UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Aplicación móvil con geolocalización para el proceso de control de pedidos en CRISS NEÓN S.A.C

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:

Ingeniero de Sistemas

AUTOR:

Pachas Sifuentes, Ricardo David (ORCID: 0000-0002-4884-8372)

ASESOR:

Mgtr. More Valencia, Rubén Alexander (ORCID: 0000-0002-7496-3702)

LÍNEA DE INVESTIGACIÓN:

Sistema de Información y Comunicaciones

LIMA - PERÚ

2020

Dedicatoria

A mi familia que me apoyaron y brindaron su sincera solidaridad en este proyecto de investigación para obtener el título profesional.

A Dios por brindarme los conocimientos adquiriros y darme fuerzas para alcanzar mis metas.

Agradecimiento

A mis padres, hermana por darme la confianza y el apoyo brindando para seguir adelante ante las adversidades.

Al Mg. Rubén More Valencia por las asesorías para finalizar este trabajo de investigación.

Al Gerente General de CRISS NEON S.A.C., por la autorización para realizar el trabajo de investigación en su empresa.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS Y FIGURAS

RESUMEN

Esta tesis muestra el desarrollo de una aplicación móvil con geolocalización para el control de pedidos en CRISS NEÓN S.A.C., la empresa antes de la implementación de la aplicación presentaba deficiencia en la generación del pedido, planificación de rutas, y verificación de datos del pedido, lo cual fue vital para resolver los objetivos, determinar el efecto de una aplicación móvil con geolocalización en el incremento del porcentaje de entregados completos en el proceso de control de pedidos y determinar el efecto de una aplicación móvil con geolocalización en el decremento de entregas perfectamente recibidas en el proceso de control de pedidos.

Se usó la metodología MOBILE-D para el desarrollo de la aplicación, debido al ser una metodología enfocada a aplicaciones informáticas móviles y bajo costo.

La investigación es de tipo aplicada, el diseño es pre-experimental y de enfoque cuantitativo. La muestra fue de 218 pedidos. La técnica de recolección de datos fue el Fichaje y el instrumento fue de Ficha de Registro. Los datos se analizaron y procesaron con el software SPSS V25.

La implementación de la aplicación móvil permitió incrementar el porcentaje de entregados completos a un 32.39% y se logró decrementar las entregas perfectamente recibidas a un 32.10%.

Palabras Claves: Aplicación Móvil, Proceso de Control de Pedidos, Porcentaje de Entregados Completos, Porcentaje de Entregas Perfectamente Recibidas, MOBILE-D.

ABSTRACT

This thesis shows the development of a mobile application with geolocation for the control of orders in CRISS NEÓN SAC, the company before the implementation of the application presented deficiencies in the generation of the order, route planning, and verification of order data, what was vital to solve the objectives, determine the effect of a mobile application with geolocation on the increase in the percentage of complete deliveries in the order control process and determine the effect of a mobile application with geolocation on the decrease of deliveries perfectly received in the order control process.

The MOBILE-D methodology was used for the development of the application, due to being a methodology focused on mobile computer applications and low cost.

The research is of an applied type, the design is preexperimental and with a quantitative approach. The sample was 218 orders. The data collection technique was Registration and the instrument was Registration Form. The data were analyzed and processed with the SPSS V25 software.

The implementation of the mobile application allowed to increase the percentage of complete deliveries to 32.39% and it was possible to decrease the perfectly received deliveries to 32.10%.

Keywords: Mobile Application, Order Control Process, Percentage of Complete Deliveries, Percentage of Perfectly Received Deliveries, MOBILE-D.

I. INTRODUCCIÓN

En la era moderna, las nuevas tendencias emergentes se volvieron parte esencial en el rubro de control de pedidos y/o entrega de productos que se lleva a cabo en organizaciones, la cual ahorra tiempo y esfuerzo al cliente, lo cual el pedido solicitado por el mismo llega en un determinado tiempo así de esa manera las organizaciones implementan soluciones Ti (Tecnología de la información) para los procesos esenciales.

En el escenario internacional, según una publicación en una revista de la editorial Thomson Reuters, en Estados unidos, realizada por Bonang Waspadadi y Lintang Yuniar (2018, p.3) con el título "Implementing DSS for selecting suitable delivery services to support Smart e-commerce", indica que el éxito en el sector del comercio electrónico que se determina por las cualidades del servicio de entrega incluido la seguridad, velocidad, trazabilidad, precio y amabilidad lo cual contribuye a un mejor servicio de entrega posible y satisfacción del usuario, es la capacidad de proveer una toma de decisiones multicriterio en tiempo real para un mejor servicio de entrega posible, se implementó un modelo arquitectónico DSS, el cual desarrollaba por entrega en comercio electrónico y así los clientes puedan elegir los criterios y subcriterios que le interesen, se realizó una encuesta a 10 usuarios utilizando cuestionarios SUS(Escala de usabilidad del sistema), lo cual los resultados mostraron el puntaje de 72 puntos que significa que el sistema es suficientemente utilizable y bueno, finalmente el sistema desarrollado apoya a los clientes de los servicios de entrega que cuente con herramientas para especificar sus preferencias sobre los criterios, selecciones de subcriterios, etc, para inferir en la trazabilidad del seguimiento del movimiento del producto desde un remitente a un receptor.

En el escenario nacional, según la publicación del diario El Comercio (2020) en Perú, elaborado por la periodista Claudia Inga, indica que: "las principales aplicaciones de delivery se vieron en incremento de demanda de solicitudes de pedidos de gel anti-bacterial, alimentos y diversos productos de higiene, lo cual ha dado un incremento en sus ventas durante estos últimos días de marzo del 2020". El Country Manager de Rappi Perú quien es el sr. Ernesto La Rosa, relata que se vio un aumento significativo en cierto número de pedidos entre ellos los supermercados y se implementó desarrollos tecnológicos en camino, se

implementó a consecuencia de la gran demanda de compras por día a causa de la pandemia del COVID-19 (Enfermedad del coronavirus 2019). Se estimó en otros mercados, en los últimos 30 días, los usuarios solicitan 18% a comparado del 2019 de alimentos que son abarrotes y perecibles mediante este canal. A la vez esto señala que este proceso se encuentra trabajando en funciones que pueda realizarse lo más simple posible para poder envían el pedido al consumidor en el tiempo establecido.

La investigación se llevó a cabo en CRISS NEON S.A.C., ubicada en Jirón Huascarán 756, La Victoria donde la organización se encuentra en el rubro de publicidad exterior e interior, esta misma sustentó su funcionamiento en su estructura organizacional en el departamento de logística que realiza funciones relativas al proceso de control de pedidos, mediante la entrevista (Anexo n°02), se hizo evidencia cierta dificultades en el proceso que son los entregados completos y decremento del porcentaje de Entregas Perfectamente Recibidas ya que no cuenta con alguna estrategia para este proceso que lo realizan de manera empírica.

Este proceso comienza al momento de que uno de los colaboradores del departamento recepciona el pedido del cliente pero existe dificultades para la transmisión del pedido por vía telefónica, email o forma manual, como consecuencia ocasiona error de ubicación del cliente, confusiones de direcciones, exceso de tiempo tanto como para procesar la solicitud del pedido y en la entrega, una vez que se genera el pedido (Decremento del porcentaje de Entregas Perfectamente Recibidas) es enviado a dicho departamento para la realización de la orden de compra resumido en el diagrama de Ishikawa (Anexo n°03) y el diagrama de funciones cruzadas del proceso de control de pedidos (Anexo n°04) , obteniendo como resultado clientes insatisfechos por el exceso de tiempo, retraso de pedidos por la falta de equivocación de direcciones, solicitud del pedido que no es efectiva por la poca capacidad del repartidor para la entrega correspondiente al cliente que hacen garantía al pedido (Incremento del porcentaje de Entregados Completos).

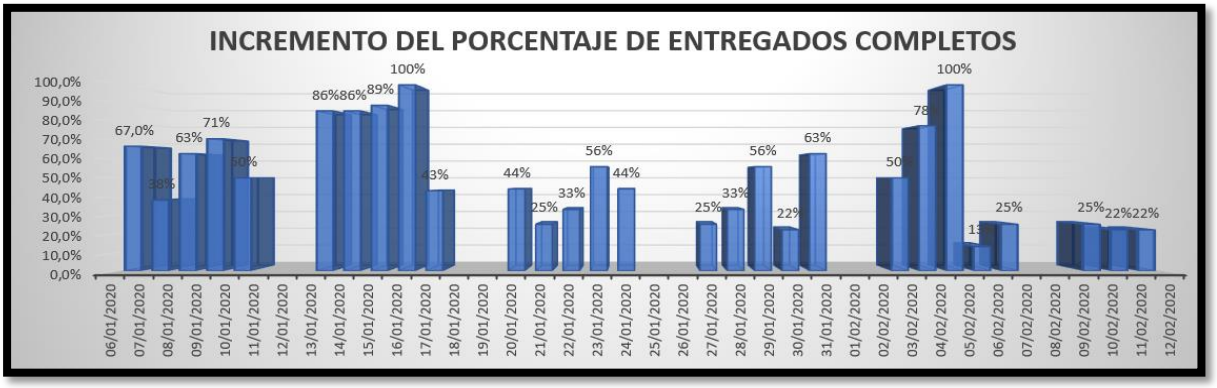


Gráfico 1. Indicador: Incremento del porcentaje de Entregados Completos (Pre-Test)

Durante el lapso de los meses de enero y febrero 2020 del indicador de incremento del porcentaje de entregados completos como lo indican el gráfico 1, solo en dos ocasiones se logró obtener entregados completos.

Durante el lapso de mes de enero y a comienzos de febrero de 2020 del indicador del decremento del porcentaje de Entregas Perfectamente Recibidas como lo indican el gráfico 2, regularmente se llega a decrementar las entregas perfectas recibidas.

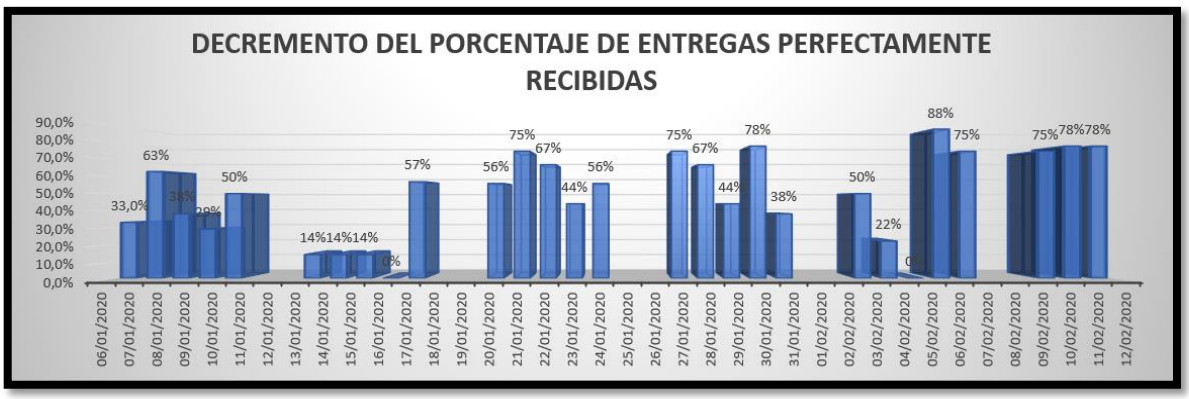


Gráfico 2. Indicador: Decremento de Entregas Perfectamente Recibidas (Pre-Test)

Debido a la realidad problemática ya mencionada, no han cumplido con las metas establecidas en el área de logística que incrementar los entregados completos y decrementar las entregas perfectamente recibidas. Por ello, se surgió el problema general: ¿Cuál es el efecto de una aplicación móvil con geolocalización en el proceso de control de pedidos en CRISS NEÓN S.A.C.? y teniendo como específicos: ¿Cuál es el efecto de una aplicación móvil con geolocalización en el incremento del porcentaje de entregados completos en el

proceso de control de pedidos en CRISS NEÓN S.A.C? y ¿Cuál es el efecto de una aplicación móvil con geolocalización en el decremento de entregas perfectamente recibidas en el proceso de control de pedidos en CRISS NEÓN S.A.C?.

Este trabajo de investigación se evidenció desde una base teórica con el propósito de brindar conocimientos verídicos sobre el uso de las aplicaciones móviles con geolocalización, cuyos resultados automatizaron el proceso que se propone, como se van a utilizar en los siguientes criterios, a continuación: desde el criterio de la relevancia social, la cual pretendió como relevancia en la sociedad, brindar una aplicación móvil que va a ahorrar tiempo y trayecto en el transcurso de una solicitud de un pedido que beneficiará la cartera de clientes de CRISS NEÓN S.A.C usando los servicios de Google y la aplicación móvil con geolocalización, citando como referencia en el marco teórico a Gupta (2019) y Abdallah (2020), desde el criterio de las implicaciones prácticas, ayudó a automatizar el proceso de las cuales implicaron las entregas fuera de tiempo y mal geolocalizadas, las cuales eran limitantes para el ahorro del proceso manual y capital a la empresa en recurso humano, citando como referencia en el marco teórico a Alarcón , Urrutia y Callejas (2016) y Babativa [et al.] (2016), desde el criterio del valor teórico, se logró alimentar el conocimiento sobre la gran importancia de las soluciones de TI desde las pymes hasta las grandes corporaciones ya que pudo cumplir el valor de aportar el conocimiento de planificar buenas prácticas de la metodología MOBILE-D para futuras investigaciones, citando como referencia en el marco teórico a Cueva (2018) y Babativa [et al.] (2016), finalmente desde el criterio de la utilidad metodológica, ayudó a realizar un instrumento confiable sobre ficha de registro para el seguimiento de la aplicación móvil con geolocalización para el proceso de control de pedidos y analizar el antes y después de la mejora de ese proceso elaborándose un marco de trabajo con los entregables de la metodología MOBILE-D citando como referencia en el marco teórico a Chavan [et al.] (2018) y Babativa [et al.] (2016).

La hipótesis general formulada en este proyecto es: La aplicación móvil con geolocalización mejora significativamente para el proceso de control de pedidos en CRISS NEÓN S.A.C. y como hipótesis específicas se tiene que La aplicación

móvil con geolocalización incrementa el porcentaje de entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C., para esta hipótesis se cita a Archiniégas [et al.] (2016), Gupta (2019) y Abdallah (2020), los cuales están contemplados sus trabajos de artículos científicos en el marco teórico y La aplicación móvil con geolocalización decrementa el porcentaje de entregas perfectamente recibidas en el proceso de control de pedidos en CRISS NEÓN S.A.C se tiene a Alcocer y Knudsen (2019), Abdallah (2020) y Vinalk (2019), los cuales están contemplados sus trabajos de artículos científicos en el marco teórico

Se definió como objetivo general: Determinar el efecto del uso de la aplicación móvil con geolocalización para mejorar el proceso del control de pedidos en CRISS NEÓN S.A.C. y como objetivos específicos se tiene: Determinar efecto de una aplicación móvil con geolocalización en el incremento del porcentaje de entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C y Determinar el efecto de una aplicación móvil con geolocalización en el decremento de entregas perfectamente recibidas en el proceso de control de pedidos en CRISS NEÓN S.A.C?.

II. MARCO TEÓRICO

Para el respaldo de esta investigación se ha buscado distintos antecedentes, tantos internacionales como nacionales, las cuales se procede a detallar:

Huamani (2018), en su investigación titulada *Sistema web para la gestión de pedidos en la empresa Impresiones Franco S.A.C.* Tuvo como objetivo de investigación determinar la influencia del sistema web en el proceso de control de pedidos en la empresa Impresiones Franco S.A.C. Fue un estudio de tipo explicativa experimental, de diseño pre-experimental, la población estuvo conformada por 319 pedidos para ambos indicadores, la muestra para el indicador de entregados completos fueron de 132 pedidos y para el indicador de Calidad de pedidos generados es de 175 pedidos y su muestreo fue probabilístico; los instrumentos empleados fue la ficha de registro. Los principales resultados de esta aplicación del sistema web es que facilitó el incremento de calidad de pedidos generados de 16,39% a su vez los entregados completos hubo un aumento de 30,84. Se concluyó que el sistema web mejoró el proceso de control de pedidos en la empresa Impresiones Franco S.A. Del presente trabajo de investigación se tuvo en cuenta el significado de la variable dependiente y las teorías relacionadas al tema.

Paima (2019), en su investigación titulada *Sistema web para el proceso de abastecimiento en la Municipalidad Provincial del Callao*. Tuvo como objetivo de investigación de determinar la influencia de un sistema web en el proceso de abastecimiento en la Municipalidad Provincial del Callao. Fue un estudio de tipo explicativa experimental, de diseño experimental, la población estuvo conformada para el indicador de Entregas perfectamente recibidas se tiene a 502 Órdenes de compra agrupados y el indicador Nivel de cumplimiento de proveedores tiene a 354 pedidos recibidos, la muestra son 218 Órdenes de compra y 184 pedidos para el indicador de Nivel de cumplimiento de proveedores y su muestreo fue probabilístico; el instrumento empleado fue la ficha de registro. Los principales resultados de la implementación del sistema web permitió disminuir las entregas perfectamente recibidas para el proceso de abastecimiento existiendo un decremento de 26.21% del antes y después de dicho indicador y a su vez se disminuyó el nivel de cumplimiento de proveedores existiendo un decremento de 22.51%. Se concluyó que el sistema web mejora el

proceso de abastecimiento en la Municipalidad Provincial del Callao. Del trabajo de investigación se tuvo en cuenta los indicadores del proyecto de investigación.

Babativa, Angélica et al. (2016), en su investigación titulada *Desarrollo Ágil de una Aplicación para Dispositivos Móviles. Caso de Estudio: Taxímetro Móvil*. Tuvo como objetivo de investigación implementar un aplicativo móvil usando una nueva tecnología denominada GPS que se incorpora en aquellos instrumentos del sistema operativo de Android. Fue un estudio de tipo experimental. la población estuvo conformada por 50 muestras, la muestra es no probabilística y su muestreo fue aleatorio simple; el instrumento empleado fue el taxímetro. Los resultados principales fueron que a través de la prueba t-student se pudo hacer válido una medición de un prototipo que obtuvo un promedio de ciertas unidades con una desviación estándar de unas 1,39 unidades. Se concluyó que la gran mejora de un desarrollo ágil para reafirman el uso de un taxímetro móvil. Este trabajo de investigación ayudó a la elaboración de diagramas UML y la metodología de investigación científica.

Baldoceda (2017), en su investigación titulada *Desarrollo de un aplicativo móvil basado en la metodología Mobile-D* para la gestión de reservas del hotel Caribe de Huaral. Tuvo como objetivo de investigación determinar la influencia de una aplicación móvil en una gestión de reserva en hotelera Caribe. Fue un estudio de tipo descriptiva. Los principales resultados de la esta implementación fueron que para estos indicadores de calidad que fueron plasmados en ciertos objetivos y descritas permitieron determinar la relación del nivel de funcionamiento del aplicativo con la portabilidad. Se concluyó que este aplicativo de gestión de reservas mejoró significativamente la gestión de reservas del hotel Caribe de Huaral. Del presente trabajo de investigación obtendremos más claro la adecuada definición de la justificación de relevancia social, implicaciones prácticas, metodológica y la elaboración de la metodología MOBILE-D.

Cueva (2018), en su investigación titulada *Aplicación móvil con geolocalización, mediante la metodología Mobile-D, para la gestión de visitas médicas en la empresa Laboratorios Siegfried S.A.C*. Tuvo como objetivo de investigación de determinar si una aplicación móvil con geolocalización mediante la metodología mobile-d, mejora la gestión de visitas médicas en la empresa Laboratorios Siegfried S.A.C. Fue un estudio aplicada experimental, de diseño

pre- experimental, la población estuvo conformada por 30 visitadores médicos, la muestra es no probabilística y no se usa ningún tipo de muestreo; los instrumentos empleados fue la ficha de observación. Los principales resultados fueron que permitió decrementar el tiempo en el registro de informes en 11.45 minutos y la ubicación de rutas en 8,53. Se concluyo que la aplicación móvil logro reducir los tiempos en la gestión de visitas médicas. Del trabajo de investigación se tuvo en cuenta el desarrollo de la metodología MOBILE-D.

Arciniégas [et al.] (2016), en su investigación titulada *Medición del desempeño de la red de suministros de medicamentos en un hospital público de tercer nivel en la ciudad de Bogotá, a través del cuadro de mando integral*. Tuvo como objetivo de investigación de desarrollar una metodología para la medición del desempeño de la red de abastecimiento de medicamentos en el Hospital Universitario de La Samaritana (HUS). Fue un estudio experimental, de diseño aplicada, la población estuvo conformada por 5 indicadores de los cuales entre ellos eran las Entregas perfectamente recibidas, etc, la muestra es no probabilística; los instrumentos empleados fue la ficha de observación. Los principales resultados fueron para el indicador de Entregas perfectas un incremento de 14.72%, para el indicador Entregas perfectamente recibidas un decremento de 6.3%, existiendo un incremento para los demás indicadores. Se concluyo que el estado actual de la gestión de la red de suministro de medicamentos en el HUS presenta notables falencias para el desempeño de la red de suministro tiene un valor porcentual notoriamente bajo. Del trabajo de investigación se tuvo en cuenta al indicador de decremento del porcentaje de entregas perfectamente recibidas y el marco teórico enfocado a pedidos.

Alcocer y Knudsen (2019), en su investigación titulada *Desempeño integral de los procesos logísticos en una cadena de suministro*. Tuvo como objetivo de investigación de desarrollar un procedimiento general para medir el desempeño integral de los procesos logísticos en una cadena de suministro. Fue un estudio experimental, de diseño experimental, la población estuvo conformada por 3 procesos Servicio al cliente que entre estos indicadores se encuentran los Pedidos entregados completos y otros indicadores, la muestra es no estratificada, por lo tanto, se procede con la medición de cada proceso con su indicador respectivo y su muestreo fue aleatorio simple; los instrumentos

empleados fue la entrevista. Los principales resultados de la implementación del desempeño integral de logística, se tuvo al indicador del indicador Pedidos entregados completos de 76% a 82%, existiendo un incremento de 6% y un incremento en los demás indicadores. Se concluyo que la aplicación del procedimiento general propuesto en las líneas de presillado de la Empresa Gráfica de Villa Clara permitió validar los resultados de dicho procedimiento a partir de la determinación del indicador Nivel de Desempeño Integral de los Procesos Logísticos logrando un mejoramiento de 0.80 a 0.87. Del trabajo de investigación se tuvo en cuenta al indicador de incremento del porcentaje de entregados completos y el marco teórico enfocado a distribución logística.

Alarcón, Urrutia y Callejas (2016), en su investigación titulada *Aplicación Móvil para la Administración de Variables Físicas en Ciclismo al Aire Libre*. Tuvo como objetivo de investigación ofrecer una herramienta de seguimiento y acompañamiento al rendimiento deportivo de un deportista. Fue un estudio de tipo descriptiva, la población estuvo conformada por rutas de entrenamiento de los deportistas y practicantes, la muestra es no probabilística y su muestreo fue aleatorio simple. Los resultados principales fueron que los valores de variables físicas se almacenaban en el servidor central por un algoritmo vinculado por Google Maps. Se concluyó que el estudio tuvo un impacto positivo en las diferentes rutas de un deportista para las sesiones de entrenamiento. Del presente trabajo de investigación se tuvo en cuenta para realizar la justificación de implicaciones prácticas y la lógica del geolocalizador con Google Maps.

Abdallah (2020), en su investigación titulada *Mobile food ordering apps: An empirical study of the factors affecting customer e-satisfaction and continued intention to reuse*. Tuvo como objetivo de investigación encontrar la implementación del aplicativo móvil en relación con la satisfacción de los clientes para utilizar dichas aplicaciones y examinar empíricamente los principales factores que hacen que sus clientes utilicen dichas apps. Fue un estudio de tipo descriptiva. Los resultados principales se basaron en la revisión en línea, la calificación en línea y seguimiento en línea hasta su expectativa de desempeño para la atención continua para el uso. Se concluyó que el estudio ha intentado proporcionar más información sobre los aspectos que podrían modelar la satisfacción de los clientes en cuestión a una aplicación móvil para pedido. Del

presente trabajo de investigación se tendrá a consideración la implementación del aplicativo en pedidos para medir su uso en cada uno de estas fases del desarrollo del aplicativo.

Gupta (2019), en su investigación titulada *A Study on Impact of Online Food delivery app on Restaurant Business special reference to zomato and swiggy*. Tuvo como objetivo de investigación conocer las estrategias de aplicaciones móviles de entrega de alimentos en la app zomato y swiggy. Fue un estudio de tipo descriptivo. Los resultados principales fueron que en la aplicación de la aplicación móvil para la entrega de alimentos tuvo un resultado satisfactorio por parte de los clientes ya que el personal se encontraba capacitado para poder competir con los demás sectores usando la tecnología como herramienta de marketing. Se concluyó que el estudio pudo comprar la comodidad de los clientes en las aplicaciones móviles mediante la expansión del lugar de comer y el comercio de alimentos, ya que, mediante la utilización de la aplicación móvil de pedidos de alimentos digitales, el personal está preparado para atraer la atención de los usuarios dándoles facilidad con el desarrollo del aplicativo compitiendo con el mercado nacional.

Chavan [et al.] (2015), en su investigación titulada *Implementing Customizable Online Food Ordering System Using Web Based Application*. Tuvo como objetivo de investigación implementar una aplicación móvil para automatizar el proceso de pedido de alimentos creando un sistema convencional reduciendo el funcionamiento de un restaurante. Fue un estudio de tipo descriptivo. Los resultados principales fueron que se pudo minimizar la imperfección del sistema convencional actual reduciendo el funcionamiento del personal de un restaurante la cual tuvo un gran nivel de satisfacción por parte de los clientes instalando el aplicativo en sus dispositivos móviles para minimizar los problemas actuales. Se concluyó que el estudio pudo implementar una aplicación móvil para automatizar el proceso de pedido de alimentos gracias a las arquitecturas tecnológicas móviles para el cumplimiento y mejora de negocios de gestión y prestación de servicios.

Vinalk, Anita et al. (2019), en su investigación titulada *The Study of Interest of Consumers In Mobile Food Ordering Apps*. Tuvo como objetivo de investigación comprender la conciencia de los consumidores sobre las aplicaciones móviles

de alimentos que comprenden la expectativa del consumidor al ordenar alimentos de una nueva aplicación para comparar diferentes alimentos en una aplicación en línea. Fue un estudio de tipo experimental. la población estuvo conformada por 134 encuestados, la muestra es no probabilística ya que es la misma que la población y su muestreo fue aleatorio simple; los instrumentos empleados fue la encuesta para la distribución de la recolección de datos. Los resultados principales fueron que 125 encuestados conocían la aplicación móvil de alimentos en línea, la cual significaba que la conciencia y el conocimiento sobre las aplicaciones de alimentos causa gran impacto en cuestión al comercio móvil para la solicitud de alimentos. Se concluyó que el estudio pudo comprobar en un análisis que la mayoría de los encuestados conocía el gran impacto de las aplicaciones móviles de alimentos como Swiggy, Foodpanda, UberEats la cual ellos consideran más importantes con respecto al método de pago fácil, mayor tiempo de entrega y recibir la orden esperada cumpliendo con un buen servicio al cliente.

En la investigación, tres autores definieron la variable dependiente que es el término Proceso de control de pedidos, en el cual para la primera definición se tiene a Tabuyo (2015) que definió que “el proceso de control de pedidos es un proceso de compra que lleva a cabo el mismo cliente que aprovisiona las mercancías fundamentales para recubrir la necesidad de un cliente” (p.124), por segundo se tuvo a Iglesias (2016) que definió que “el control de pedidos consiste en la distribución de la colección de artículos en la zona de picking con las características de un pedido” (p.95), existe un tratamiento de información para recepcionar los pedidos de clientes y para Ballou (2004), define que “El proceso de control de pedidos es una representación por el número de las labores que se cumplen en el ciclo de vida de un cliente, está incluida la preparación, entrada, transmisión, surtido e informar sobre este estado del pedido”.

En las fases del proceso de control de pedidos, para precisar cada una, en estas fases se consideró a Ballou (2004), lo cual para poder dimensionar esta variable en estas dimensiones: entrada de pedido y estado de pedido, que se ven involucradas en este proceso, en la primera fase Preparación del pedido, se definen las actividades que recopilan la información sobre lo requerido por el cliente mediante vía telefónica o correo, en la segunda fase Transmisión del

pedido, abarca la transferencia de un pedido de un punto a origen, en la tercera fase Entrada del pedido que define el ingreso de un pedido a poder realizar un levantamiento real de un pedido que debe estar las características del producto, órdenes de compra para la información del pedido, como cuarta fase Surtido del pedido se tiene una serie de actividades para la entrega del pedido y como última fase Estado del pedido, es la finalización del proceso para suministrar una calidad de servicio al cliente que va a garantizar el mismo. (Ballou, 2004, p.133).

Para la primera dimensión se tuvo entrada del pedido ya previamente definido en las fases del proceso de control de pedido (Ballou, 2004, p.133). Para su respectivo indicador se tiene al incremento del porcentaje de entregados completos que según Mora (2008), "consiste en conocer un nivel de efectividad de los despachos de mercancías hacia los clientes en cuanto a los pedidos enviados a un periodo determinado" (p.32). Para obtener el porcentaje se usará la siguiente fórmula que se comprende como: EC = Entregados Completos, NPEC = No. Pedidos entregados completos y TP = Total pedidos:

$$EC = \frac{NPEC}{TP} * 100$$

Para la segunda dimensión se tuvo a estado del pedido ya previamente definido en las fases del proceso de control de pedido (Ballou, 2004, p.133). Para el segundo indicador se tiene a decremento de entregas perfectamente recibidas, lo define Mora (2008), explica que las entregas perfectamente recibidas muestran el número y porcentaje que no han cumplido con las expectativas que se hayan establecido a la servicio y calidad. Para obtener el porcentaje se usará la siguiente fórmula que se comprende como: EPR = Entregas Perfectamente Recibidas, PR = Pedidos Rechazados, TOCR = Total de órdenes de compra recibidas.

$$EPR = \frac{PR}{TOCR} * 100$$

Para la variable independiente que fue Aplicación Móvil, tres autores definieron este término, la cual para la primera se tuvo a Serna (2016), " una aplicación móvil significa aquel pequeño paquete de software que servirá para hacer resolución de una o varias tareas en específico. " (p.20), la segunda

definición, según Santiago y Trabaldo (2015), “En una esencia, la aplicación siempre va ser un software que sirven para ordenadores de escritorio” (p.34) y por último se tuvo a López (2015), “Son aquellas plataformas de open source, los diseños a usarse son de exclusividad de Tablet tanto para dispositivos móviles otorgando una interfaz amigable al usuario. […].”

En esta investigación se definió el término de aplicación nativa, la cual se usó para el proyecto de investigación ya que según Serna (2016), "este tipo de aplicaciones usan todo el potencial de hardware de los terminales a través de la paquetería del desarrollo de sistema."(p.26), para la definición de aplicaciones híbridas y web revisar el anexo n°18.

En esta investigación se definió el concepto sobre la arquitectura de una aplicación móvil que nos define según Jiménez (2016), “esta arquitectura seguirá las guías de diseño de interfaces de aplicaciones móviles para la petición del usuario mediante peticiones HTTP para obtener datos que realicen el establecimiento de órdenes de un dispositivo.” (p.7).

A fin de poder programar aquel producto final, se tiene al software Android Studio, que se define como “aquella ventana de proyecto que consta en construir ciertos campos con nombres de una aplicación como lo muestra el S.O. de Android.”. (Smyth, 2019).

Para la implementación de la investigación se usó la geolocalización para un mayor control de pedido ya que según Beltrán (2015), “es la ubicación de una persona u objeto en el espacio, que se representa en un mapa” (p.5).

En este proyecto de investigación se usó el lenguaje de programación Java ya que, Garrido (2015), nos informa que “la tecnología Java es un lenguaje de programación de elevado nivel que va ser orientado a objeto reciente creación y tienen una sintaxis muy similar al lenguaje C o C++. (p.2-3), para definir que es ver otros lenguajes de programación, se va a revisar el anexo n°18.

El gestor de base de datos que se usó fue Firebase ya que según Moroney (2017), “el servicio de firebase es aquel gestor de base de datos no relacional que incluye el realtime database almacenada en formato JSON y sincroniza en tiempo real” (p.66). Para informar sobre base de datos, revisar el anexo n°18.

A la vez en esta investigación se definió como metodología de desarrollo de software, la metodología MOBILE – D y según Escobar y Campaña (2014), nos define que es una metodología ágil orientada a las aplicaciones móviles clasificada en diferentes fases: exploración, inicialización, producto, estabilización y pruebas, se definió las 5 fases de la metodología, la cual se tiene a Abrahamsson, Pekka et al. (2017), que son Exploración, en la cual hace realización de un establecimiento de proyecto y una planificación, que esta se conllevan a los entregables que son el establecimiento de partes interesadas, y definición de alcance ; la segunda fase conocida como Inicialización tiene el propósito de verificar de cualquier aspecto fundamental para el análisis del desarrollo de la aplicación móvil; la tercera fase que es Producción, en la cual se implementa la funcionalidad que se requiere en cierto producto y la estabilización que consiste en presentar la compatibilidad de la aplicación móvil y finalmente las pruebas que son pruebas unitarias para controlar la calidad del software. Para definir que es la metodología de desarrollo de software y otras metodologías como SCRUM y RUP revisar el anexo n°18.

De acuerdo a los resultados que se han obtenido en la Tabla 1, que se puede visualizar que esta metodología MOBILE – D obtuvo un total de 87 puntos. Por lo tanto, para desarrollar la Aplicación móvil con geolocalización se usó MOBILE-D y como se visualizan en las fichas de evaluación de expertos con las puntuaciones en el Anexo n°13.

Tabla 1

Selección de Metodología de desarrollo de software

EXPERTO	METODOLOGÍAS		
	RUP	**SCRUM**	**MOBILE - D**
Mg. Pérez Farfán, Iván Martín	**23**	**28**	**30**
Dra. Vásquez Valencia, Yesenia	**22**	**25**	**27**
Mg. More Valencia, Rubén Alexander	**26**	**28**	**30**
Total	71	81	87

III. MÉTODO

3.1. Tipo y diseño de investigación.

Se definió el tipo de investigación, Valbuena (2015), lo definió como “el tipo de investigación aplicada se halla sujetada en una investigación básica, que ha dependido de aportes teóricos de lo qué se va descubrir para llevar a fin la solución de la problemática con él para poder haber generado factores controlados, la cual implicó un proceso que proporciono una salida que generalmente produce los más fuertes, lógicos y válidos resultados que pueda concebir de la investigación científica, todo tipo de investigación experimental configura un modelo que persigue una meta con un propósito en mente.” (p.32).

Para haber realizado el proyecto de investigación se aplicó la investigación aplicada que tuvo como objetivo observar la situación actual del proceso de control de pedidos en CRISS NEÓN S.A.C y al aplicar una aplicación móvil con geolocalización a partir de los conocimientos recién proporcionados mediante una investigación de modo que se determinó si estos pueden ser aplicados para los propósitos que se definan.

Esta investigación tuvo un nivel de investigación explicativa ya que tuvo el propósito de probar hipótesis y carácter predictivo para haber pronosticado ciertos efectos la cual confían en cierta capacidad de explicación o predicción y corrección de sus disciplinas (Garza, 2009, p.16).

El enfoque cuantitativo usó la recolección de datos a que hizo un análisis de datos con el fin de responder las preguntas de investigación que se plante y probar hipótesis formuladas con anterioridad, lo cual usó la medición de instrumentos y variables en una investigación, lo cual el uso de estadística inferencial y descriptiva, probar hipótesis y es de carácter riguroso (Ñaupas et al., 2019, p.140), para esta investigación se aplicó el enfoque cuantitativo ya que se recolectó datos y analizó que el objeto de estudio es de 1 pedido, lo cual ha sido vital para el desarrollo de esta investigación.

Se definió el término de diseño de investigación, según Bilbao y Escobar (2020), que definió que “las investigaciones experimentales son un estudio en

las cuales se operan intencionalmente a más de una variable independiente (causa) para poder examinar las consecuencias del impacto de la debida manipulación que tienen entre sus variables dependientes (efectos) y sobre tipo pre-experimental significa que sea diseño con caso único (grupo experimental) y un diseño de aquel grupo con medición de un antes y después, las cuales comprenden como un diseño con solamente un grupo pre y post-test, la cual comprende un diseño sin grupo control, donde se efectúa una medición de inicio y una medición posterior pero su grado de control es riguroso.” (p.61-62).

Según este esquema del diseño**:**

$$G = \quad O1 \qquad X \qquad O2$$

Donde:

G: Grupo Experimental (objeto de estudio).

O1: Pre-test (Medición de la variable dependiente de inicio).

X: Experimento o tratamiento (Variable independiente)

O2: Post-test (Medición de una variable dependiente después de implementar una variable independiente).

En este trabajo de investigación se aplicó un diseño experimental del pre experimental de tipo pre-test y post-test, tal cual que se tuvo que evaluar las variables independientes y dependientes en un antes durante el proceso diario y un después, tomando como consideraciones:

La variable dependiente antes de modificarla (Pre Test)

La variable dependiente después de modificarla (Post Test)

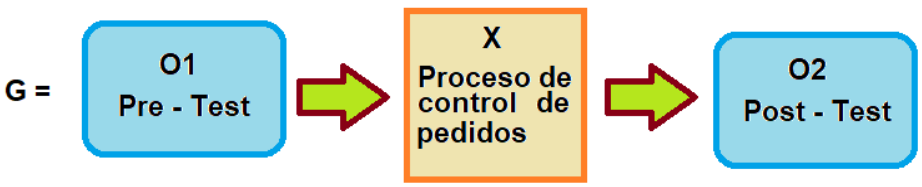


Figura 1. Clasificación de la investigación

3.2. Variables y Operacionalización.

La variable independiente que es Aplicación Móvil, se definió conceptualmente por Serna (2016), "una aplicación móvil significa aquel pequeño paquete de software que servirá para hacer resolución de una o varias tareas en específico. Estos son similares a programas de procesamiento de texto, programas de diseño, hojas de cálculo tanto como edición para video que funcionan los ordenadores de escritorio con complejidad menor y optimización para un contexto móvil" (p.20), que se definirá operacionalmente como La aplicación móvil con geolocalización optimiza el proceso de control de pedidos en la solicitud de un pedido de producto de un cliente basándose en la geolocalización; la variable dependiente es Proceso del control de pedidos quien Iglesias (2016) la define conceptualmente como “el control de pedidos consiste en la distribución de la colección de artículos en la zona de picking que se realiza teniendo en cuenta la solicitud del cliente, características del artículo, ubicación del usuario y del sistema de preparación de pedidos” (p.95), se definió operacionalmente como El uso de la aplicación móvil con geolocalización para el proceso de control de pedidos logrando el incremento del porcentaje de entregados completos y decremento del porcentaje de entregas perfectamente recibidas, esta variable dependiente se dividió en dos dimensiones, la cual la primera fue entrada del pedido que tiene como indicador que es el incremento del porcentaje de entregados completos y la segunda dimensión que es estado del pedido que tiene como indicador, el decremento del porcentaje de entregas perfectamente recibidas, el instrumento para medirlo es la ficha de registro y como escala la razón, también se puede observar en el Anexo n°6.

Para los indicadores del proceso del control de proceso se tuvo al primer indicador que es el incremento del porcentaje de entregados completos como se observa en el Anexo n°7, la cual se define según Mora (2008), “consiste en conocer un nivel de efectividad de los despachos de mercancías hacia los clientes en cuanto a los pedidos enviados a un periodo determinado” (p.32)., la técnica que se usó fue el fichaje, el instrumento que se usó fue el ficha de registro, la unidad de medida es 1 pedido y la fórmula correspondiente es:

$$EC = \frac{NPEC}{TP} * 100$$

Donde se comprendió:

EC = Entregados Completos

NPEC = No. Pedidos entregados completos

TP = Total pedidos

Como segundo indicador del proceso del control de proceso se tuvo al incremento del porcentaje de entregas perfectamente recibidas como también se observa en Anexo n°7, la cual lo define Castellano (2015) citando a Mora (2008), explica que las entregas perfectamente recibidas muestran el número y porcentaje que no han cumplido con las expectativas que se hayan establecido en cuanto al servicio y a la calidad. Para obtener el porcentaje se usará la siguiente fórmula que se comprende como: EPR = Entregas Perfectamente Recibidas, PR = Pedidos Rechazados, TOCR = Total de órdenes de compra recibidas.

$$EPR = \frac{PR}{TOCR} * 100$$

Donde se comprendió:

EPR = Entregas Perfectamente

PR = Pedidos Rechazados

TOCR = Total de órdenes de compra recibidas.

3.3. Población, muestra y muestreo.

Para este estudio se tuvo a Gutiérrez (2015), que definió que “la población significa aquel conjunto de aquellos casos enfatizado en la coincidencia con ciertas especificaciones definidas; que se hacen conjunto por todas estas unidades de análisis o significan elementos o escenarios y dependerán de un planteamiento de investigación y de aquellos alcances de estudio que enfatizan

a esta población que se va seleccionar, su selección adecuada depende del objetivo propuesto y la clave es obtener un buen marco teórico" (p.76).

En esta investigación, la población que se tomó para ambos indicadores es la conformación de 500 pedidos totales recibidos que correspondió al indicador del incremento del porcentaje de entregados completos y decremento del porcentaje de entregas perfectamente recibidas que le correspondería los meses de enero del 2020 e inicios de febrero del 2020 en un intervalo de 28 días.

Dicho grupo de objeto que cumplen con las siguientes características son los pedidos de los clientes al solicitar uno o varios productos, la cual han correspondido a los criterios de inclusión para haber determinado el estudio correspondiente, por otra parte, hay objetos que no han cumplido con las características de la población que se tomó como se tiene a los clientes, productos y monto total, esos 3 objetos formaron parte de los criterios de exclusión.

Para definir la muestra se tuvo a Argüeso, Mónica et al. (2017), que aclaran que "la muestra significa aquel subconjunto que representa la población, para la elección de muestra representativa de un aspecto fundamental, ya que esto han hecho conclusiones finales del estudio sean fiables." (p.391).

Según lo mencionado lo anterior, se obtuvo la muestra a partir de la muestra de una obtención finita.

$$n = \frac{Z^2 * N\ p\ q}{e^2(N-1) + Z^2\ p\ q}$$

Donde se obtuvo:

n: Tamaño de la muestra.

Z: Nivel de confianza escogido usado en este trabajo de investigación que es 95% equivalente a 1,96.

N: Población total de la investigación.

e: Error que se estimó (al 5%).

p: Proporción que se esperara (en este caso 5% = 0.05).

q: Precisión (Que se usó un 5%).

$$n = \frac{1.96^2 * 3360 * 0.5 * 0.5}{0.05^2(3360 - 1) + 1.96^2 * 0.5 * 0.5}$$

$$n = 218$$

Según lo obtenido de resultado de la muestra para la elaboración del trabajo de investigación se determinó 218 pedidos estratificados por 28 días. De tal modo que el número de muestra como resultado se utilizó en 28 fichas de registro en diferentes fechas por cada indicador.

La muestra fue estratificada ya que según Ross (2018), nos dice que "la estratificación se caracteriza particularmente por la efectividad que tiene para averiguar las proporciones medias en la población en lo que respecta a una pregunta en interés cuando se halla diferencias significativas entre subpoblaciones, será efectiva para conocer la receptibilidad media a la misma. (p.7)

Para haber definido el muestreo el término se tiene a Argüeso, Mónica et al. (2017), que define que "el muestreo es aquella técnica de escoger aquellos elementos de la muestra de la forma más adecuada posible, con la finalidad de poder generalizar las conclusiones del estudio a toda la población." (p.391).

Para haber definido el término de técnica de muestreo aleatorio simple se tiene a Argüeso, Mónica et al. (2017), que define que el "el muestreo aleatorio simple es el elegido en aquel tamaño de la muestra, aquellos elementos que formen parte que se escogen con anteriores, a través de una cantidad de una población dada, este muestreo adecuada nuestra población que es homogénea respecto a la característica del estudio." (p.393).

Esta investigación optó por la técnica del muestreo aleatorio simple, de modo que va a extrajo de la población finita, una cantidad cierta de pedidos subpoblaciones de un tamaño fijado con anterioridad haber sido probablemente elegido.

Se definió el término de la unidad de análisis, se tiene a Taborda y Pérez (2016) que definen que "la unidad de análisis es aquella entidad productiva, empresa o firma; puede representarse en punto de una venta, agregación geográfica o almacén en donde se puesta poder hacer el establecimiento de nivel

de ventas, etc.” (p.35). En la unidad de análisis que se obtuvo según el objeto de estudio será 1 pedido ya que se adecuan al estudio y se tiene a Marín (2015), que nos definirá el término “pedido” que “son aquellas peticiones de compras que un cliente hace conocimiento al proveedor para la suministración de productos o servicios que se solicitan, es un proceso para solicitar productos o servicios.” (p.8)

3.4. Técnicas e instrumentos de recolección de datos.

Se definió el término de Técnica ya que, Según Ibañez (2015), indicaron que “es aquel conjunto de conocimientos y habilidades que va a servir para dar resolución a problemas prácticos y un conjunto de recursos y procedimientos de que sirve un arte o ciencia.” (p.90). La técnica que se usó para esta investigación fue la del fichaje ya que, según Parraguez, Simona et al. (2017), indicaron que “el fichaje es aquella técnica que va a permitir cierto registro de información que fue optado para su proceso de investigación, la ejecución de esta va a requerir en uso de ficha para ayudarnos a organizar y recoger la información que se extrae de varias fuentes de las que sean de interés de acuerdo que hagan énfasis con estas características de la investigación.” (p.150). El fichaje se usó para la investigación para registrar data de los indicadores: incremento del porcentaje de entregados completos y decremento del porcentaje de entregas perfectamente recibidas de CRISS NEÓN S.A.C.

Para haber definido un instrumento se tiene a Ibañez (2015), nos indica que “un instrumento puede medir lo que se requiera y fiabilidad, grado en que la aplicación es sucesiva y repetida ya que va a producir iguales resultados a la vez debe reunir requisitos imprescindibles y la validez.” (p.149). El instrumento que se usó para dicha investigación será la ficha de registro ya que según Parraguez et al. (2017), nos señaló que “una ficha de registro es aquel formato, lo cual que se va hacer la recolección de datos en una forma sistemática y con una adecuación de la estructuración para la manipulación de los hechos observados.” (p.151). Por dicho motivo, en esta investigación se optó por usar la ficha de registro que va servir de instrumento para recolectar datos y con el fin de poder adecuar a una ejecución normal, de la cual se visitó frecuentemente a la organización que cumplirá el objetivo de evaluar dicho proceso de control de

pedidos para que pueda formar adecuadamente una realización para los indicadores tanto para medir este pre y post test. Dichos instrumentos y técnicas para la recolección de datos donde se visualizan en la tabla 2.

Tabla 2

Tabla de técnicas e instrumentos para la recolección de datos

Variable	Dimensión	Indicador	Técnica	Instrumento
Proceso de control de pedidos	Entrada de pedidos	Incremento del porcentaje de Entregados Completos	Fichaje	I1: Ficha de registro (ver Anexo n°09)
	Estado del pedido	Decremento del porcentaje de Entregas Perfectamente Recibidas	Fichaje	I2: Ficha de registro (ver Anexo n°09)

La entrevista es una herramienta fundamental para recolectar los datos lo cual Lusthaus (2001), nos definió que “la entrevista es aquel método de la cual se recolectara datos de un individuo que es considerada único por la posición o experiencia que posee y está determinado por una guía de entrevista y protocolo de entrevista.” (p.129).

Para haber definido la validez se tiene a Zapata, Rosa et al. (2015), nos indica que “hace énfasis al grado en que aquel instrumente va a medir realmente dicha variable que va pretenderse medir.” (p.220). Se clasificó primeramente por la validez de criterio que, según Zapata, Rosa et al. (2015), informa que “se consiste en hacer relación los constantes resultados de poder hacer la implantación del instrumento vinculado a ciertas evaluaciones de un criterio de forma externa de las medidas pretendidas a determinar.” (p.220), seguidamente por la validez de contenido que, según Zapata, Rosa et al. (2015), menciona “el grado en que aquel instrumento va a evidenciar un control que sea exacto de lo que contiene de lo que se va a calificar.” (p.220) y finalmente por la validez de constructo que, según Zapata, Rosa et al. (2015), mencionó que “se va a explicar

cómo todas las medidas de las variables de estudio se van a correlacionar con otras mediciones de cualquier juicio de teoría.” (p.221).

Para este trabajo de investigación, se definió una validez de este instrumento de medición que se hizo procedimiento a ejecutar un juicio de expertos, la cual consistió en realizar la consulta a personas expertas en el tema para evaluar la estructura de esta herramienta, la cual se tuvo en cuenta las apreciaciones y la resaltación de estimaciones obtenidas que pudieron cambiar a lo largo del tiempo que pudieron realizarse modificaciones o cambios en investigación para los indicadores como se visualiza en la tabla 3 y 4.

Para dicho indicador 1, lo cual es entregados completos se tuvo el puntaje de los expertos a continuación:

Tabla 3

Validación del Indicador – Incremento del porcentaje de Entregados Completos

Expertos	Preguntas										Total
	1	2	3	4	5	6	7	8	9	10	
Mg. Acuña Meléndez, María Eudelia	80	80	80	80	80	80	80	80	80	80	**80**
Mg. More Valencia, Rubén Alexander											

Se presentó las definiciones de la variable dependiente, definición de las dimensiones, indicador del incremento del porcentaje de entregados completos y la matriz de consistencia, la cual obtuvo una evaluación del promedio de 71%,

alcanzando un nivel de validez alto (ver Anexo n°14). Por ende, motivo, se consideró que ese instrumento sea válido y aceptado con el fin de que fue usable para recolectar datos de este trabajo de investigación.

Para el indicador 2, lo cual es decremento del porcentaje de entregas perfectamente recibidas se tuvo el puntaje de los expertos a continuación:

Tabla 4

Validación del Indicador – Decremento del porcentaje de Entregas Perfectamente Recibidas

Expertos	Preguntas										Total
	1	2	3	4	5	6	7	8	9	10	
Mg. Acuña Meléndez, María Eudelia	80	80	80	80	80	80	80	80	80	80	**80**
Mg. More Valencia, Rubén Alexander											

Se presentó las definiciones de la variable dependiente, definición de las dimensiones, indicador del decremento del porcentaje de entregas perfectamente recibidas y la matriz de consistencia, la cual se obtuvo la evaluación de un promedio de 72% que alcanzo cierto nivel de validez alto (ver Anexo n°15). Por ende, motivo, se consideró que este instrumento sea válido y aceptado cumpliendo el fin que sea usable para recolectar datos de este trabajo de investigación.

En la tabla 5 se presentan las puntuaciones del puntaje de validación para los indicadores.

Tabla 5

Puntaje de validación de indicadores

Puntuación	Significado
0 – 20%	Deficiente
21 – 50%	Regular
51 – 70%	Bueno
71 – 80%	Muy bueno
81 – 100%	Excelente

Santos (2017)

Para definir la confiabilidad del instrumento, Santos (2017) manifiesta que "también llamado precisión, se logra entender como un grado en que los puntos medidos hallan absueltos de ciertos fallos como la realización de la misma ejecución de la medición de indicadores que deben ser similares." (p.2).

El método de confiabilidad hizo manifestación de cinco niveles de conclusión en coordinación al valor que se define del p-valor de contraste (sig.) que se ha encontrado sujeto a estas limitaciones como se aprecia en la tabla a continuación:

Tabla 6

Niveles de p-valor del contraste (sig)

Escala	Nivel
0.00 < sig. < 0.20	Muy bajo
0.20 < sig. < 0.40	Bajo
0.40 < sig. < 0.60	Regular
0.60 < sig. < 0.80	Aceptable
0.80 < sig. < 1.00	Elevado

Santos (2017)

En ocasión que el valor de sig., se hizo aproximación a 1, se hace conclusión que fue un instrumento confiable, la cual se realizó cálculos estables y consistentes.

Si este valor del sig., es menor a 0.8, aquel instrumento, la cual se evaluó cierta variación heterogénea entre estos ítems.

En esta investigación se usó el método de Test – Re Test, lo cual Hernández, Fernández y Baptista (2014), que “es cierta medida que abarca la estabilidad de que va llevarse en ejecución en un procedimiento de un instrumento semejante de medición que va a aplicarse entre una a más veces en un cierto grupo de personas que van a ver en un determinado periodo luego, si su correlación entre sus resultados de estas distintas aplicaciones pertenece a un grado positivo, el instrumento será considerado confiable, a continuación se va presentar la representación gráfica sobre el proceso de cálculo con dos aplicaciones que abarca una medida de estabilidad. (p.301).

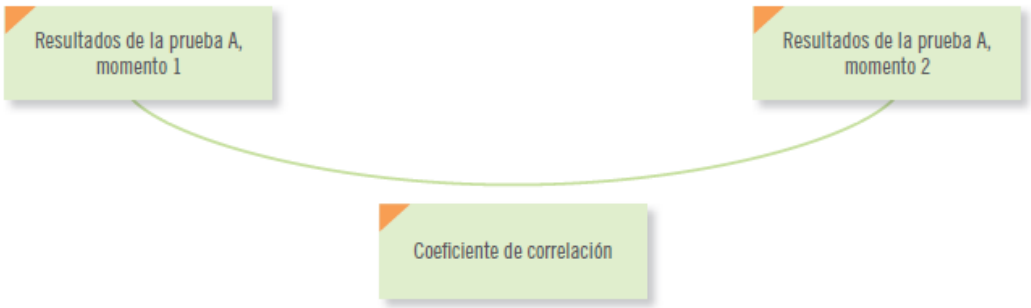


Figura 2. Proceso de cálculo con dos aplicaciones abarcando en una medida de estabilidad
Hernández, Fernández y Baptista (2014)

El coeficiente de correlación de Pearson pudo resolver el problema anterior, ya que según Rodríguez (2017), este coeficiente “requiere que la distribución de ambas variables sea semejante a la de la curva normal, destacándose en el término de asociación social pero no linealmente. “(p.62). La interpretación de este coeficiente es una prueba estadística que va a calcular un valor asociado entre estas dos variables que oscila entre -1 y +1, cuando dicho valor se acerca a +1, su relación entre las dos variables es directa y fuerte que se podría afirmar que cuando una variable se ve en aumento a la que forma similar y viceversa, en el caso de -1, la relación entre aquellas variables es inversa y fuerte que se puede afirmar que cuando la variable disminuya, la otra variable va aumentar de forma similar y viceversa como lo afirma Rodríguez (2017,p.62).

Para la realización del análisis de confiabilidad, se usó el software estadístico IBM SPSS Statics versión 25 ya que según George y Mallery (2018), “es una herramienta poderosa que es capaz de realizar casi cualquier tipo de

análisis de datos utilizado en ciencias sociales, ciencias naturales o en el mundo de los negocios a su vez es un programa estadístico complejo y poderoso según cualquier estándar". (p.9).

Tabla 7

Confiabilidad del indicador del incremento del porcentaje de Entregados Completos

		EC_Test	EC_Retest
EC_Test	Correlación de Pearson	1	,888**
	Sig. (bilateral)		,000
	N	14	14
EC_Retest	Correlación de Pearson	,888**	1
	Sig. (bilateral)	,000	
	N	14	14
**. La correlación es significativa en el nivel 0,01 (bilateral).			

Se ejecutó el análisis de confiabilidad para el indicador de incremento del porcentaje de entregados completos, (ver Anexo n°12), según el coeficiente dado como correlacional de Pearson en el software estadístico IBM SPSS Statics 25 es de 0,888, la cual se aprecia, cierto nivel elevado de confiabilidad que denota aquel instrumento que se aprecia en la tabla 7.

Tabla 8

Confiabilidad del indicador Entregas Perfectamente Recibidas

		EPR_Test	EPR_Retest
EPR_Test	Correlación de Pearson	1	,887**
	Sig. (bilateral)		,000
	N	14	14
EPR_Retest	Correlación de Pearson	,887**	1
	Sig. (bilateral)	,000	
	N	14	14
**. La correlación es significativa en el nivel 0,01 (bilateral).			

Se realizó un análisis de confiabilidad para medir el indicador de decremento del porcentaje de entregas perfectamente recibidas, (ver Anexo n°12), según el coeficiente dado en correlacional de Pearson del software estadístico IBM SPSS Statics 25 es de 0,887, en lo cual se aprecia un nivel aceptable de confiabilidad, la cual denota este instrumento en que se visualiza se aprecia en la tabla 8.

En la figura 3, se puso en evidencia esta fórmula para medir dicho coeficiente de la correlación de Pearson, tal como lo presenta Rodríguez (2017).

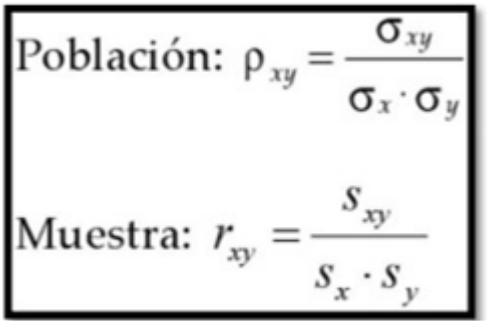


Figura 3. Fórmula de coeficiente de la correlación de Pearson

Rodríguez (2017)

Donde:

pxy= Coeficiente de la correlación de Pearson por Población

rxy= Coeficiente de la correlación de Pearson por Muestra

σxy = Sxy = Covarianza tanto para x e y

σx = Sx = Desviación típica de una variable x

σy = Sy = Desviación típica de una variable y

3.5. Procedimientos.

En esta sección se obtuvo como descripción del modo de recolección de datos de la empresa CRISS NEÓN S.A.C., el cual se usó fichas de registro para cada indicador de la investigación mediante el objeto de estudio que son los pedidos, para su recolección previa se hizo que realizar la coordinación con el gerente general de la empresa que fue accedida en la entrevista (ver Anexo n°2).

En la tabla 9, se observó lo que se consolida con lo expuesto. Se evidencia los datos de la empresa, la coordinación con el gerente general de la empresa y su proceso. Como especificaciones se obtuvo la técnica, instrumento, fuentes e informantes de cada indicador.

Tabla 9

Cuadro del procedimiento del caso de estudio

Datos generales				
Organización	CRISS NEÓN S.A.C.			
Coordinación	Gerente general de la empresa y asistente de ventas			
Recolección	Proceso de control de pedidos			
Especificaciones				
Indicador	**Técnica**	**Instrumento**	**Fuente**	**Informante**
Incremento del porcentaje de Entregados Completos	Fichaje	Ficha de registro	Proformas realizadas en el área de ventas	Gerente general
Decremento del porcentaje de Entregas Perfectamente Recibidas	Fichaje	Ficha de registro	Proformas realizadas en el área de ventas	Gerente general

3.6. Métodos de análisis de datos.

Para Hernández, Arturo et al. (2018), indica que este "método de análisis de enfoque cuantitativo se perfila a cierta minoración de categorías analíticas que se puedan considerarse frecuenciales y correlaciones entre otras, según un análisis estadístico. (p. 82).

El método de análisis de datos que se usó para este proyecto es de tipo cuantitativo pre-experimental, que en su vez se va a obtener estadísticamente, la ayuda de la comprobación si es correcta la hipótesis. Dado que la investigación hizo énfasis en buscar la comparación resultados de una data histórica que es el pre-test de los resultados al ejecutar dicha herramienta que es el post test, si el caso que la muestra es mayor a 50; de cual para haber verificado el contraste de las hipótesis va a realizarse con la prueba Z; en una distribución que es normal para calcular una probabilidad que se hace integro de la función de densidad dentro de un intervalo de la región deseada a fin de ser aceptada o rechazada, como nuestra muestra fue menor a 50, se usó la prueba de Shapiro Wilk para apoyar a la estimación valores de la población desde los datos de esta muestra y hacer pronóstico de una probabilidad en estos dos promedios.

En las pruebas de normalidad, se tiene a Vilalta (2016), que define que "estas pruebas que son realizadas que se fundamentan en la comparación de los resultados productos de la muestra con aquellos que se espera lograr si la hipótesis es correcta o nula". (p.69).

La robustez de cierta prueba estuvo de la mano de que esta muestra sea de mayor cantidad, siendo se va usar una prueba de Shapiro Wilk.

Si n > 50 = Prueba de Kolmogorov – Smirnov

Si n < 50 = Prueba de Shapiro Wilk

Estas pruebas han alojado aquellos datos de pre test y tanto para un post test de cada uno de estos indicadores usando un software estadístico IBM SPSS Statics versión 25, mediante estas premisas.

Si:

Sig < 0.05 adoptará una distribución no normal

Sig >= 0.05 adoptará una distribución normal

Donde:

Sig. Es aquel nivel o valor de criterio de contraste

En cuenta, se usó en esta investigación por los indicadores del incremento del porcentaje de entregados completos y decremento del porcentaje de entregas perfectamente recibidas, la prueba de normalidad de Shapiro Wilk, lo cual estos indicadores trabajaron según a esta muestra que va a estar estratificada en 28 fichas de registro, a su vez es menor a 50 (n<50) y sus rangos e intervalos son perimetralmente de una distribución normal.

El método estadístico que va usar en dicha validación de la hipótesis correspondería a una prueba t de Student.

La prueba de hipótesis se define Hernández, Fernández y Baptista (2014), los definen como “aquella determinación de la hipótesis poblacional que es racional con los datos que se obtienen de la muestra” (p.306). Se compone como hipótesis nula (H0), de la que se hace planteamiento de los grupos que no difieren significativamente y una hipótesis alternativa (H1 o Ha) que se puede formular cuando de manera efectiva si existen diferentes posibilidades que también se puede tomar en cuenta la hipótesis nula y de investigación. (Hernández, Fernández y Baptista, 2014, p.105).

Como hipótesis de investigación 1, se tiene la Hipótesis Específico 1 (HE1) que es que La aplicación móvil con geolocalización incrementa el porcentaje de entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C. El Indicador 1 que es Incremento del porcentaje de Entregados Completos se clasifican en IECa: Incremento del porcentaje de Entregados Completos antes de utilizar la aplicación móvil con geolocalización. y IECd: Incremento del porcentaje de Entregados Completos después de utilizar la aplicación móvil con geolocalización y como Hipótesis estadística 1 se tiene a la Hipótesis Nula (H0): La aplicación móvil con geolocalización mantiene los entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C.

H0: IECa = IECd

La cual, se interpreta que el indicador sin la aplicación móvil con geolocalización es mejor que el indicador con la aplicación móvil con geolocalización.

Como hipótesis alterna (HA) es La aplicación móvil con geolocalización incrementa los entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C.

$$HA: IEPa < IEPd$$

La cual, se interpreta que el indicador con la aplicación móvil con geolocalización es mejor que el indicador sin la aplicación móvil con geolocalización.

Como hipótesis de investigación 2, se tiene a la Hipótesis Específico 2 (HE2) que es La aplicación móvil con geolocalización decrementa las entregas perfectamente recibidas en el proceso de control de pedidos en CRISS NEÓN S.A.C. El indicador 2 que es decremento del porcentaje de entregas perfectamente recibidas se clasifican en IEPRa: Decremento del porcentaje de Entregas Perfectamente Recibidas antes de utilizar la aplicación móvil con geolocalización. y IEPRd: Decremento del porcentaje de Entregas Perfectamente Recibidas después de utilizar la aplicación móvil con geolocalización y como hipótesis estadística 2 se tiene a la Hipótesis Nula (H0) que La aplicación móvil con geolocalización no decrementa las entregas perfectamente recibidas en el proceso de control de pedidos a CRISS NEÓN S.A.C.

$$H0: IEPRa \geq IEPRd$$

La cual se interpreta que el indicador sin la aplicación móvil con geolocalización es mejor que el indicador con la aplicación móvil con geolocalización y como hipótesis alterna (HA): La aplicación móvil con geolocalización decrementa el porcentaje de entregas perfectamente recibidas en el proceso de control de pedidos a CRISS NEÓN S.A.C.

$$HA: IEPRa < IEPRd$$

La cual se interpreta que el indicador con la aplicación móvil con geolocalización es mejor que el indicador sin la aplicación móvil con

geolocalización.

El nivel de significancia que se va usar fue x = 5% (error), lo que se equivale a 0.05, esto posibilitó llevar a cabo la contraposición para que se pueda aprobar o rechazar esta hipótesis.

Nivel de confiabilidad: (1-x) = 0.95

La distribución t-student comprendió como una distribución continua que viene caracterizada por grados de libertad que son un número de datos observados menos 1 (n-1) que esta distribución es muy parecida a la normal tipificada puesto que se centra en el cero y es simétrica ya que es una distribución normal. (Gómez y López, 2017, p.276).

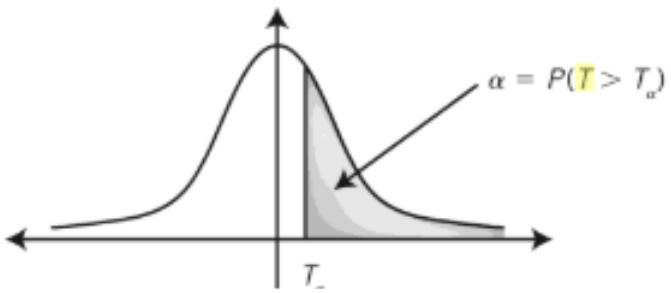


Figura 4. Distribución de T-Student
Gómez y López, 2017

3.7. Aspectos éticos.

En esta investigación se tomó los datos que se indican en este proyecto de tesis que son correspondidos reunidos de un grupo pre-experimental y control de investigación de manera correcta sin que se haya adulterado ni un detalle con honestidad para el avance del trabajo de investigación, aquellos datos estarán asentados en un instrumento que se aplicó al grupo y objeto de estudio.

Esta investigación ha seguido en base a estos lineamientos y reglamentos que otorgó la Universidad César Vallejo y la norma ISO 690, de modo que esta investigación otorgue una base confiable al momento de la sustentación del trabajo de investigación mediante las citas.

El trabajo realizo es auténtico basándose en lo anteriormente mencionado. Se ha mantenido en una confiabilidad estricta, la información los datos que han sido reunidos representativamente para cumplir con la investigación y futuros resultados que se va a obtener por la empresa CRISS NEÓN S.A.C., así como haber salvaguardado los objetivos que son parte de este estudio de investigación.

Se realizó la entrevista a la empresa CRISS NEÓN S.A.C. para que se pueda dar a pase a usar la información de la empresa y elaborar de manera correcta dicho trabajo de investigación.

IV. RESULTADOS

Como análisis descriptivo, en dicho estudio se aplicó una aplicación móvil (Variable Independiente) para evaluar el incremento del porcentaje Entregados Completos y decremento del porcentaje de Entregas Perfectamente Recibidas en el proceso de Control de Pedidos (Variable Dependiente); para ello se aplicó un Pre-Test que permitió indagar las condiciones iniciadas del indicadas, posteriormente se implementó la aplicación móvil y nuevamente se registró el incremento del porcentaje de entregados completos en el proceso de Control de Pedidos. Los resultados descriptivos de dichas medidas se observan en las tablas 10 y 11.

Para el indicador “Incremento del porcentaje de Entregados Completos”, los resultados descriptivos de dicho indicador se observan en la tabla 10.

Tabla 10

Estadísticos descriptivos del Incremento de Porcentaje de Entregados Completos

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desviación estándar
Entregados_Completos_Pretest	28	13,00	100,00	51,0357	25,87074
Entregados_Completos_Postest	28	50,00	100,00	83,4339	12,99693
N válido (por lista)	28				

En el caso del incremento del porcentaje de entregados completos en el proceso de control de pedidos, en el pre-test se obtuvo un valor de 51,04%, mientras que en el post-test fue de 83,43% tal como se aprecia en la figura 6; esto indica la gran diferencia del antes y después de la implementación de la aplicación móvil; así mismo, los entregados completos fue del 13% antes y 50% (ver Tabla 10) después de la implementación de la aplicación móvil y como se observa en el gráfico 3.

En cuanto a la dispersión del incremento del porcentaje de entregados completos, en el pre-test se tuvo una variabilidad del 25,87; sin embargo, en el post-test se tuvo un valor de 13.

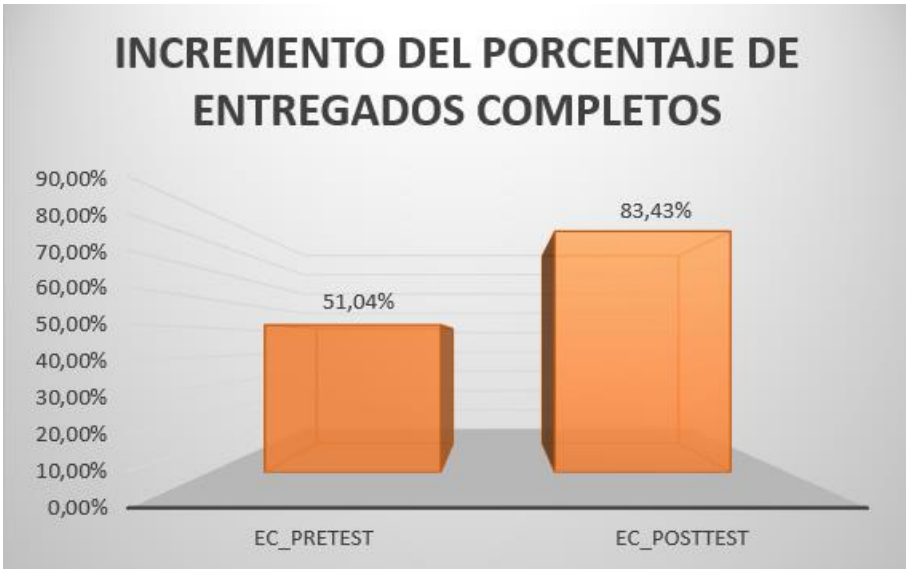


Gráfico 3. Incremento del porcentaje de entregados completos del antes y después de la implementación de la aplicación móvil

Para el indicador “Decremento del porcentaje de Entregas Perfectamente Recibidas”, los resultados descriptivos de dicho indicador se observan en la tabla 11.

Tabla 11

Estadísticos descriptivos del Decremento del porcentaje de Entregas Perfectamente Recibidas

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desviación estándar
Entregas_PR_Pretest	28	00,00	88,00	49,1071	25,91202
Entregas_PR _Postest	28	00,00	50,00	17,0125	12,61592
N válido (por lista)	28				

En el caso del decremento del porcentaje de entregas perfectamente recibidas en el proceso de control de pedidos, en el pre-test se obtuvo un valor de 49,11%, mientras que en el post-test fue de 17,01% tal como se aprecia en el gráfico 4; esto indica la gran diferencia del antes y después de la implementación de la aplicación móvil; así mismo, el porcentaje del decremento de entregas

perfectamente recibidas fue del 88% antes y 50% (ver Tabla 11) después de la implementación de la aplicación móvil y como se observa en el gráfico 4.

En cuanto a la dispersión de los entregados completos, en el pre-test se tuvo una variabilidad del 25,91; sin embargo, en el post-test se tuvo un valor de 12,62.

Gráfico 4. Decremento del porcentaje de entregas perfectamente recibidas del antes y después de la implementación de la aplicación móvil

Para el análisis inferencial se realizó las pruebas de normalidad donde se procede a la realización de pruebas de normalidad para los indicadores del incremento del porcentaje de entregados completos y decremento del porcentaje de entregas perfectamente recibidas en el proceso de control de pedidos usando el método de Shapiro-Wilk, ya que es vital en el momento de trabajar con una muestra estratificada, los cuales son las 28 fichas de registros, que, al ser menor de 50, según Hernández, Fernández y Baptista (2014, p. 376). Para realizar la prueba se debe introducir los datos precisos de cada indicador dentro del software SPSS 25.0, con un nivel de confiablidad de 95%, bajo ciertas reglas:

Si:

Sig. < 0.05 adquiere una distribución no normal. Sig. ≥ 0.05 adquiere una distribución normal.

Donde:

Sig.: P-valor o nivel crítico del contraste. Los resultados fueron los siguientes:

Con el fin de probar la hipótesis; los datos fueron debidamente supervisados para la comprobación de la correcta distribución, de esta manera se puede comprender si los datos del incremento del porcentaje de entregados completos en el área de logística poseen una distribución frecuente.

En la tabla 12, se va visualizar las pruebas de normalidad para el indicador “Incremento del porcentaje de Entregados completos”, antes y después de implementar dicha aplicación móvil bajo Shapiro Wilk.

Tabla 12

Prueba de normalidad de Entregados Completos del antes y después de la implementación de la aplicación móvil

	Shapiro-Wilk		
	Estadístico	gl	Sig.
EntregadosCompletos_PreTest	,933	28	,074
EntregadosCompletos _PostTest	,927	28	,053

Como se puede apreciar en la Tabla 12 las conclusiones de la evaluación muestran que el Sig. del incremento del porcentaje de Entregados Completos en el área de logística en el Pre-Test fue de 0,07 cuyo valor es mayor que 0,05 Por consiguiente, los datos brindados por el indicador de entregados completos se distribuye de manera normal.

Las conclusiones de la evaluación del Post-Test detallan que el Sig. del incremento del porcentaje de Entregados Completos fue de 0,05, cuyo valor es mayor o igual que 0,05, de manera que señala que los entregados completos se distribuyen de manera normal. Lo que corrobora la distribución paramétrica de ambos datos de la muestra, se puede apreciar en los gráficos 5 y 6.

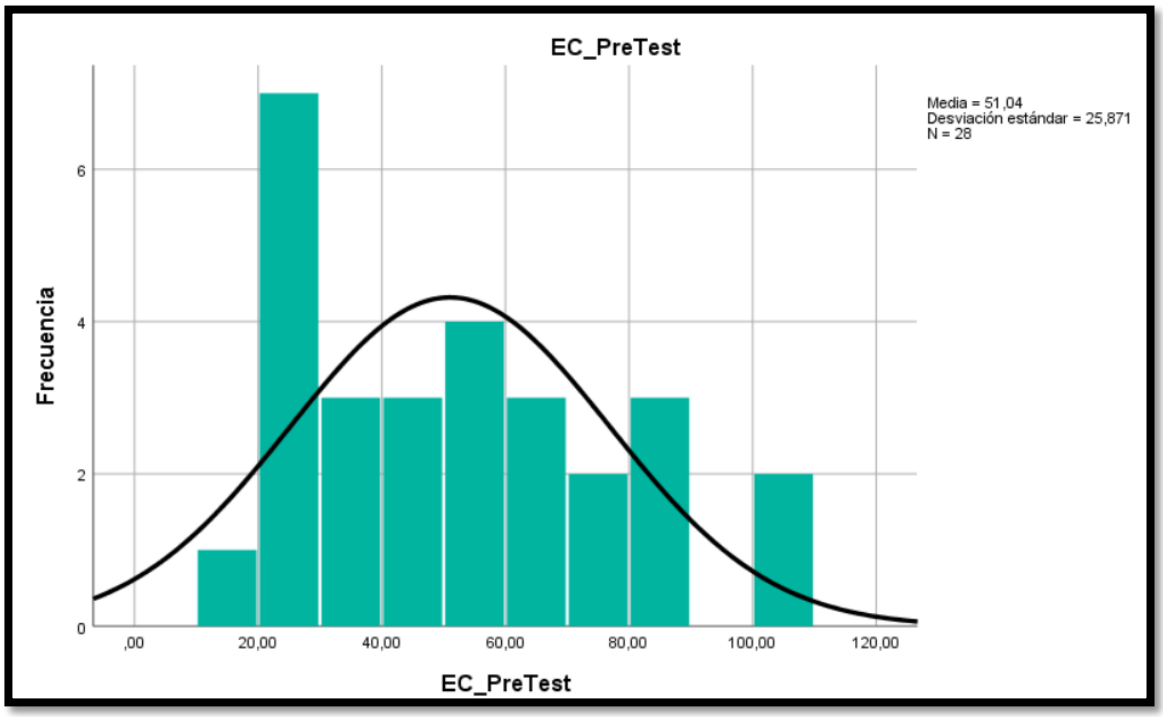


Gráfico 5. Prueba de Normalidad del Incremento del porcentaje de Entregados Completos del antes de la implementación de la aplicación móvil

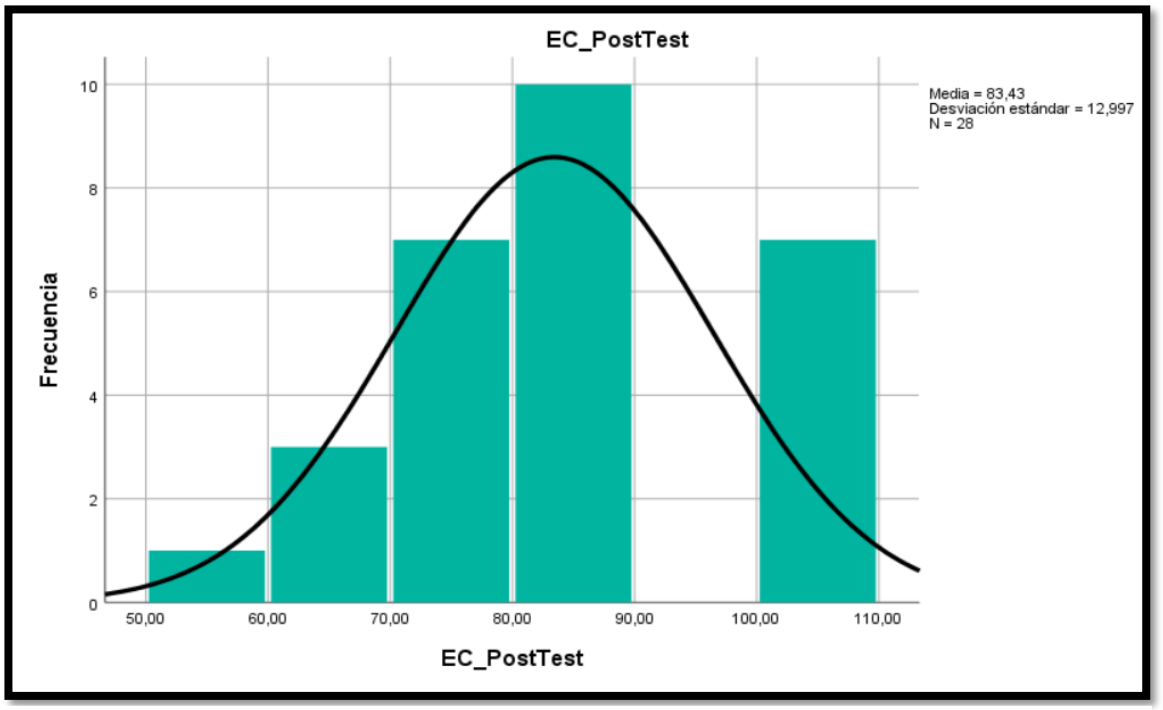


Gráfico 6. Prueba de Normalidad del Incremento del porcentaje de Entregados Completos del después de la implementación de la aplicación móvil

Con el fin de probar la hipótesis; los datos fueron debidamente supervisados para la comprobación de la correcta distribución, de esta manera se puede comprender si los datos del Decremento del porcentaje de Entregas Perfectamente Recibidas en el área de logística poseen una distribución frecuente.

En la tabla 13, se va visualizar las pruebas de normalidad para el indicador "Decremento del porcentaje de Entregas Perfectamente Recibidas", antes y después de implementar dicha aplicación móvil bajo Shapiro Wilk.

Tabla 13

Prueba de normalidad del Decremento del porcentaje de Entregas Perfectamente Recibidas del antes y después de la implementación de la aplicación móvil

Shapiro-Wilk			
	Estadístico	gl	Sig.
Entregas_PR_PreTest	,935	28	0,81
Entregas_PR_PostTest	,936	28	0,87

Como se puede apreciar en la Tabla 13 las conclusiones de la evaluación muestran que el Sig. del Decremento del porcentaje de Entregas Perfectamente Recibidas en el área de logística en el Pre-Test fue de 0,81 cuyo valor es mayor que 0,05 Por consiguiente, los datos brindados por el indicador del decremento del porcentaje de entregas perfectamente recibidas, se distribuyen de manera normal.

Las conclusiones de la evaluación del Post-Test detallan que el Sig. de los entregados completos fue de 0,87; cuyo valor es mayor que 0,05, de manera que señala que el Decremento del porcentaje de Entregas Perfectamente Recibidas se distribuye de manera normal. Lo que corrobora la distribución paramétrica de ambos datos de la muestra, se puede apreciar en los gráficos 7 y 8.

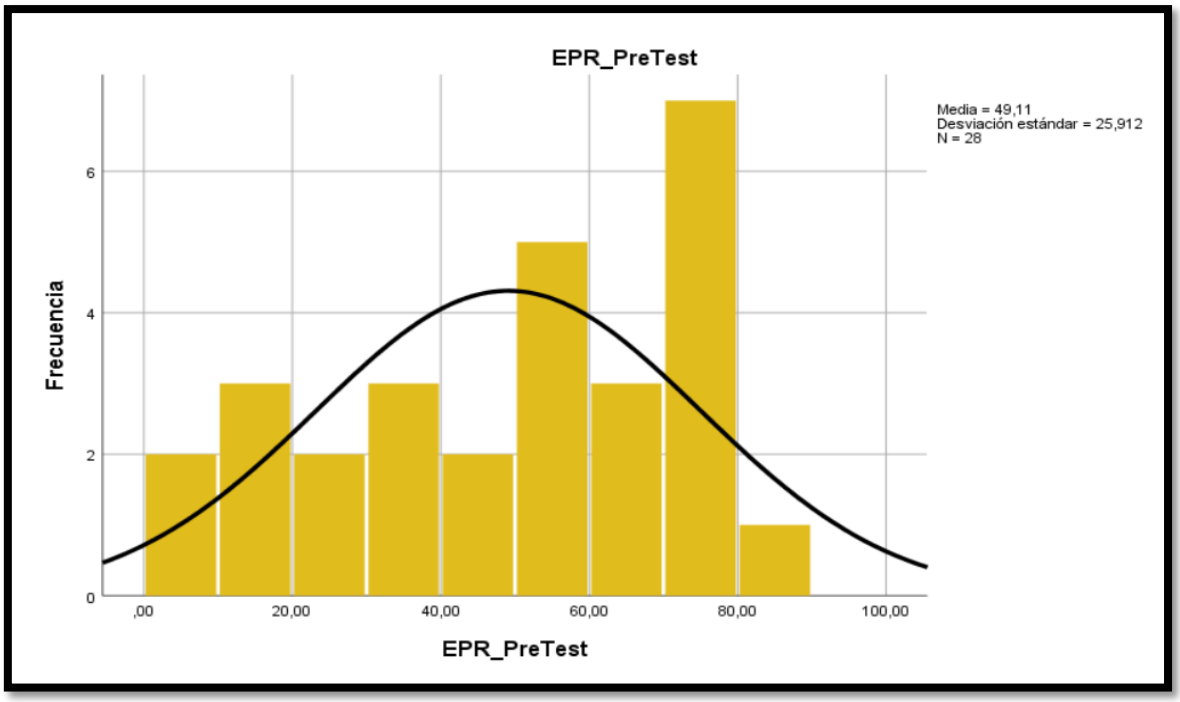


Gráfico 7. Prueba de Normalidad del Decremento del porcentaje de Entregas Perfectamente Recibidas del antes de la implementación de la aplicación móvil

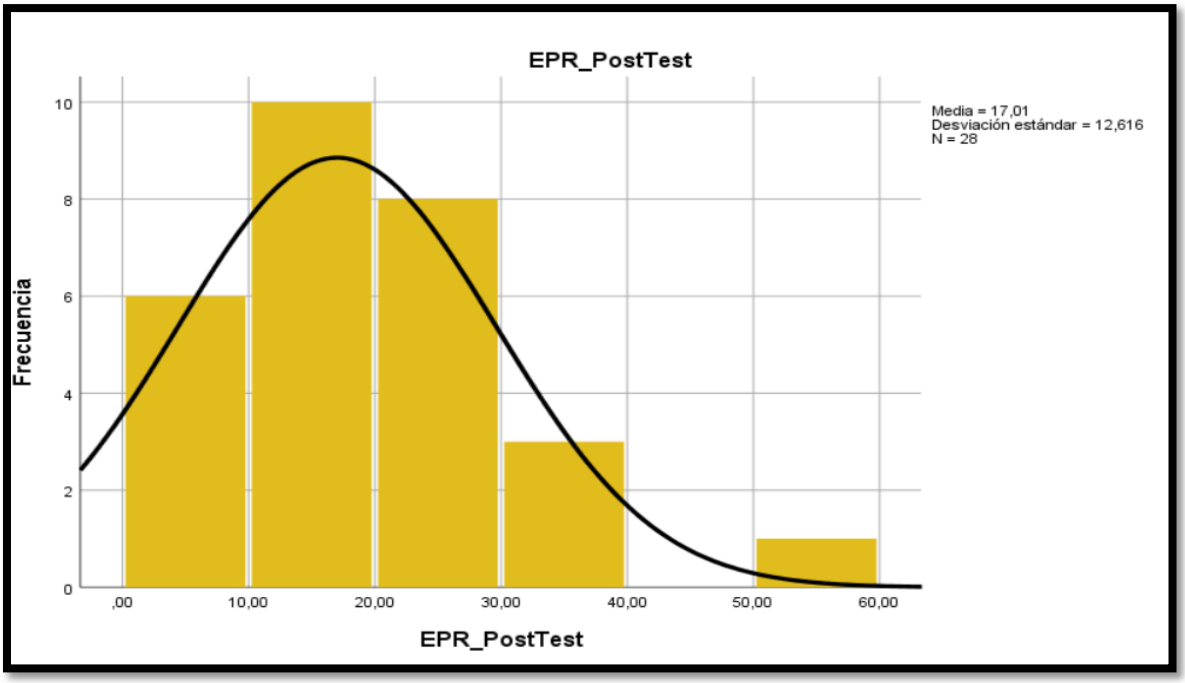


Gráfico 8. Prueba de Normalidad del Decremento del porcentaje de Entregas Perfectamente Recibidas del después de la implementación de la aplicación móvil

La hipótesis de investigación 1 fue La aplicación móvil con geolocalización incrementa el porcentaje de entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C. para el indicador de Incremento del porcentaje de entregados completos y para las Hipótesis Estadísticas se definieron las variables que son IECa: Entregados Completos antes de usar la aplicación móvil y IECd: Entregados Completos después de usar la aplicación móvil.

Cómo hipótesis nula (H0) tuvimos que La aplicación móvil con geolocalización no incrementa el porcentaje de entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C

$$H0: IECa \geq IECd$$

El indicador sin la aplicación móvil es mejor que el indicador con la aplicación móvil.

Cómo hipótesis alterna (HA) tuvimos que La aplicación móvil con geolocalización incrementa el porcentaje de entregados completos en el proceso en el proceso de control de pedidos en CRISS NEÓN S.A.C

$$H0: IECa < IECd$$

El indicador con la aplicación móvil es mejor que el indicador sin la aplicación móvil.

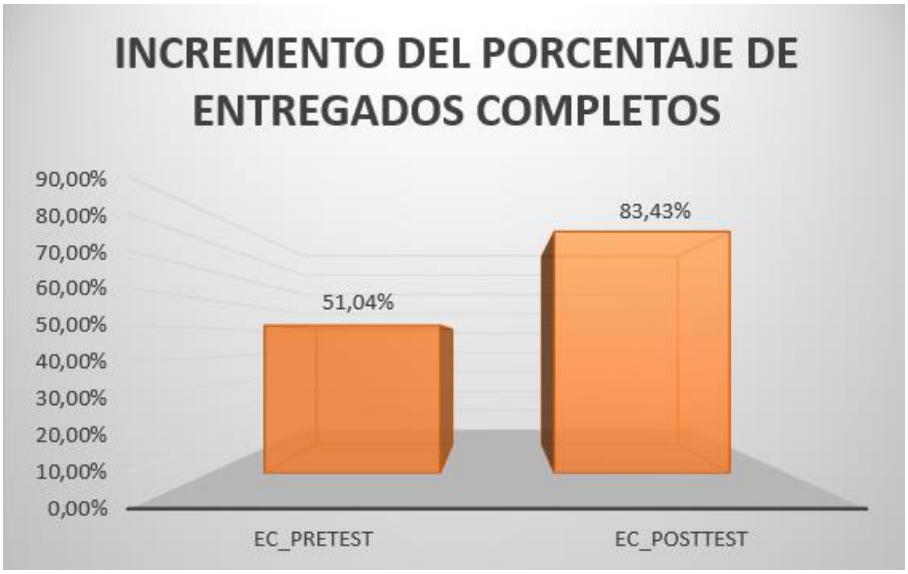


Gráfico 9. Comparativa general del Incremento del porcentaje de Entregados Completos

Se infiere del gráfico 7 que si existe un incremento en los Entregados Completos, que se reafirma al comparar las medias respectivas del pre y post, que asciende de 51,04% al valor de 83,43%.

En cuanto a la conclusión del contraste de hipótesis se utilizó la Prueba T-Student, ya que los datos adquiridos durante los tiempos establecidos (Pre-Test y Post-Test) se distribuyen normalmente. El valor de t contraste es de -7.027 , el cual es notoriamente menor que -1.708. (Ver tabla 14).

Tabla 14

Prueba de T-Student para el Incremento del porcentaje de Entregados Completos en el proceso de control de pedidos antes y después de implementar la aplicación móvil

	Prueba de T-Student			
	Media	T	gl	Sig. (bilateral)
EntregadosCompletos_PreTest	51,04	-7,027	27	,000
EntregadosCompletos_PostTest	83,43			

En ese sentido, se desaprueba la hipótesis nula y se acepta la hipótesis alterna con un 95% de confianza. Asimismo, el valor T obtenido, como se aprecia en el gráfico 8, se ubica en la zona de rechazo. Por lo tanto, la aplicación móvil con geolocalización incrementa el porcentaje de entregados completos en CRISS NEON S.A.C.

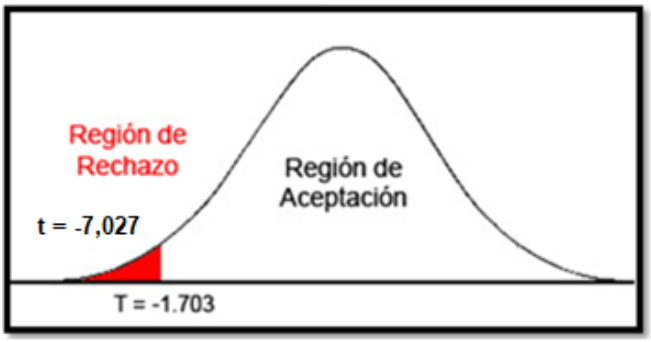


Gráfico 10. Prueba T-Student del Incremento del porcentaje de Entregados Completos

En la prueba t, cuando el p significativo es menor o igual a 0,05 se rechaza la hipótesis nula (H0). Dado que la media en el pre-test es 51,04 es menor a la media en el post-test con 83,43 se acepta la hipótesis de investigación 1 que indica que La aplicación móvil con geolocalización incrementa el porcentaje de los entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C.

La hipótesis de investigación 2 fue, La aplicación móvil con geolocalización decrementa el porcentaje de entregas perfectamente recibidas en el proceso de control de pedidos en CRISS NEÓN S.A.C para el indicador del decremento de porcentaje de entregas perfectamente recibidas y para las Hipótesis Estadísticas se definieron las variables que son IEPRa: Decremento del porcentaje de Entregas Perfectamente Recibidas antes de usar la aplicación móvil y IEPRd: Decremento del porcentaje de Entregas Perfectamente Recibidas después de usar la aplicación móvil.

Cómo hipótesis nula (H0) tuvimos que La aplicación móvil con geolocalización no decrementa el porcentaje de entregas perfectamente recibidas en el proceso de control de pedidos en CRISS NEÓN S.A.C

H0: IEPRa ≥ IEPRd

El indicador sin la aplicación móvil es mejor que el indicador con la aplicación móvil.

Cómo hipótesis alterna (HA) tuvimos que La aplicación móvil con geolocalización decrementa el porcentaje de entregas perfectamente recibidas en el proceso en el proceso de control de pedidos en CRISS NEÓN S.A.C

H0: IEPRa < IEPRd

El indicador con la aplicación móvil es mejor que el indicador sin la aplicación móvil.

Gráfico 11. Comparativa general de Decremento del porcentaje de Entregas Perfectamente Recibidas

Se infiere del gráfico 9 que si existe un decremento del porcentaje de entregas perfectamente recibidas, que se reafirma al comparar las medias respectivas del pre y post, que asciende de 49,11% al valor de 17,01%.

En cuanto a la conclusión del contraste de hipótesis se utilizó la Prueba T-Student, ya que los datos adquiridos durante los tiempos establecidos (Pre-Test y Post-Test) se distribuyen normalmente. El valor de t contraste es de -7,146, el cual es notoriamente menor que -1.708. (Ver tabla 15).

Tabla 15

Prueba de T-Student para el decremento del porcentaje de entregas perfectamente recibidas en el proceso de control de pedidos antes y después de implementar la aplicación móvil

	Prueba de T-Student			
	Media	T	gl	Sig. (bilateral)
Entregas_Perfectamente_Recibidas_PreTest	49,11	7,146	27	,000
Entregas_Perfectamente_Recibidas_PostTest	17,01			

En ese sentido, se desaprueba la hipótesis nula y se acepta la hipótesis alterna con un 95% de confianza. Asimismo, el valor T obtenido, como se aprecia en el gráfico 8, se ubica en la zona de rechazo. Por lo tanto, la aplicación móvil con geolocalización decrementa el porcentaje de entregas perfectamente recibidas en CRISS NEON S.A.C.

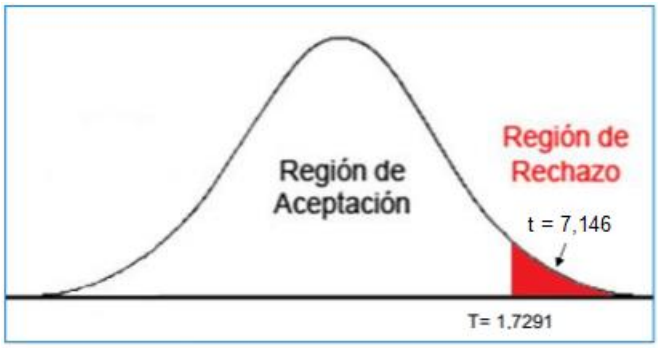


Gráfico 12. Prueba T-Student del decremento de entregas perfectamente recibidas

En la prueba t, cuando el p significativo es menor o igual a 0,05 se rechaza la hipótesis nula (H0). Dado que la media en el pre-test es 49,11 es mayor a la media en el post-test con 17,01 se acepta la hipótesis alternativa que indica que La aplicación móvil con geolocalización decrementa las entregas perfectamente recibidas en el proceso de control de pedidos en CRISS NEÓN S.A.C.

V. DISCUSIÓN

De acuerdo al objetivo, el cual fue determinar el efecto de una aplicación móvil con geolocalización para el proceso de control de pedidos, se analizó los resultados , respectó al indicador de incremento del porcentaje de Entregados completos, se justificó pérdidas de costos, falta de atención del personal de logística, la cual demandó a implementar una solución tecnológica que pueda mitigar la problemática actual en el departamento de logística, para lo cual en esta investigación se cumplieron estos factores para el adecuado proceso de control de pedidos usando un instrumento confiable como la ficha de registro; Los resultados se evaluaron por los pedidos solicitados tanto como era antes y después de la implementación de la aplicación por parte del departamento de logística donde se dividió en dos indicadores incremento del porcentaje de entregados completos y decremento del porcentaje de entregas perfectamente recibidas, se consideró los pedidos solicitados, en camino, entregados y cancelados de 218 pedidos estratificados en 28 días; en tanto para el porcentaje de los entregados completos aumentó de un 51.04% a un 83.43%, existiendo un incremento de 32.39% y se logró disminuir las entregas perfectamente recibidas de un 49.11% a un 17.01%, existiendo un decremento de 32.10%, para lo cual se valida lo respecto a Ballou (2004), que los factores para una buena calidad de pedido tiene que estar incluida la preparación, entrada, transmisión, surtido e informar sobre este estado del pedido.

Para el estudio sobre el proceso de control de pedidos, existió un problema frecuente sobre el porcentaje de entregados completos antes de la implementación, ya que la mayoría de estos pasaban del estado de en camino a cancelados, por la falta de coordinación con el mismo por lo cual en la aplicación móvil con geolocalización se implementó módulos como la selección de productos que implican tamaño, complemento, precio y cantidad también la transmisión de notificaciones cuando llega el pedido al perfil del administrador, cambia el estado de pedido de “atendido” a “en camino” notificando al cliente y al repartidor, finalmente ver el trayecto actual del pedido desde la ubicación actual hasta el destinatario, Huamaní (2018), tenía su proceso de forma manual reduciendo a gran escala por la competividad el mercado ocasionado pedidos

entregados incompletos y poca calidad del mismo, por lo cual se optó un sistema web para la gestión de pedidos con la metodología OOHDM para el cumplimiento de entregables de cada una de estas, Huamaní definió como concepto de gestión en cuanto a productos que se ofrece en una organización y en esta investigación, se definió como concepto de la representación del ciclo de vida de labores que incluyen preparación, entrada, transmisión, surtido e informar sobre el estado del pedido; para los indicadores de esta investigación se optó por entregados completos al igual que esta investigación pero se tomó el indicador en la investigación de Huamani de Calidad de pedidos generados, lo cual no se optó para esta investigación; con respecto a los resultados existió un incremento de 30.84 y en esta investigación existió un incremento de 32.39%; con respecto a la comparación de soluciones en el objeto de estudio, se optó por implementar un sistema web y para esta investigación, una aplicación móvil, para lo cual en esta investigación se podrá implementar a un futuro, un sistema multiplataforma que ayude a la empresa en estudio a poder agilizar más a la transformación digital que es un auge, actualmente frente a las empresas de su mismo sector.

Para este estudio, no había un control de pedidos eficaz, lo cual se demostró en el diagrama de Ishikawa (Anexo n°3) que afectaba a ambos indicadores tratados, los cuales ocasionaba pérdidas en ganancias y poca fidelización de clientes, consecuente se implementó tecnológicamente en la aplicación móvil con geolocalización, interfaces para clasificar el pedido por estados (atendido, en camino, entregado y cancelado) y ver un mapa por el servicio de google maps ubicando la ubicación actual del repartidor hacia el punto de entrega marcando en tiempo real la geolocalización del mismo. La falla de solicitudes de clientes se decrementó en un 32.1%, puesto que la aplicación móvil usa servicios en la nube de Google optimizando el proceso de control de pedidos, se puede asumir que existió un resultado cercano a Paima (2019) que logró decrementar las entregas perfectamente recibidas. Así mismo, según la teoría de Ballou (2004), para que se puedan cumplir las labores de un ciclo de vida de un cliente tiene que estar incluido la preparación, entrada, transmisión, surtido e informe sobre el estado de un pedido; en la investigación de Paima se toma en énfasis el proceso de abastecimiento y comparando con las fases del proceso de control de pedidos se enfoca en la primera fase que es preparación del pedido y la tercera fase que

es entrada del pedido ya que ambas fases comprueban el estrado de un crédito de cliente, se enfoca en el reparto de bienes y servicios a lo que se asimila al proceso de abastecimiento; como solución tecnológica para la problemática de esta investigación de Paima, realizó un sistema web bajo la metodología OOHDM que existió un decremento de 26.21% y para el indicador de nivel de cumplimiento de proveedores se logró decrementar 22.51%, en esta investigación, para el indicador de decremento de Entregas Perfectamente Recibidas se logró disminuir las entregas perfectamente recibidas de un 49.11% a un 17.01%, existiendo un decremento de 32.10%, no se consideró el indicador de nivel de cumplimiento de proveedores, ya que se enfocó a la entrega de pedidos por el personal de delivery en cambio en la investigación de Paima se enfocó a la gestión logística para la entrega de mercancías de los proveedores.

La aplicación móvil con geolocalización contempló las 5 fases, como preparación, la solicitud de pedido de cliente, eso involucra a la selección del producto mismo por tamaño y complemento que son auxiliares del producto final, en la fase 2, transmisión, involucra la transmisión de notificación de nuevos pedidos con Firebase Cloud Messaging para él envió de notificaciones, en la fase 3 , entrada del pedido, Tabuyo (2015), revisó la necesidad del aprovisionamiento como la forma de tener calidad en la disposición del cliente, la aplicación lo hace por actualización del producto en las características del precio, nombre, descripción, foto, tamaño y complementos y que se ponen a disposición de los clientes, en la fase 4, surtido del pedido, envío de pedidos a repartidores , cambio de estado del pedido de "solicitado" a "en camino" , envío de notificación de un nuevo pedido en el perfil del repartidor y recorrido del mismo hacia el destinatario siguiendo una línea roja de rutas cortas por un servicio de google maps, en la fase 5, estado del pedido, en la fase 5, informe sobre el estado del pedido, envío de notificación de pedido entregado al perfil del cliente y reportes de excelencia con indicadores del control del mismo, valida, respalda del trabajo de Ballou para la calidad del pedido como un proceso integral (Ballou, 2004) (Alcocer y Knudsen, 2019), así mismo para el indicador de entregados completos, existió el incremento del 6% y en esta investigación, permitió una entrega escalable y fiable a sus requerimientos iniciales, para el indicador de incremento del porcentaje de Entregados completos de 51.04% a 83.43%,

existiendo un incremento de 32.39% y no se consideró a los demás indicadores de los procesos de servicio al cliente ya que esta investigación es enfocada a la entrega de pedidos, en cambio la investigación de Alcocer y Knudsen se enfocó al proceso logístico en una cadena de suministro.

Para el estudio sobre el proceso de control de pedidos, se tiene que tomar un proceso de compra que aprovisione los productos finales de los clientes que requieran, dado que era un problema recurrente durante el proceso que se está estudiando, existía muchas inconsistencias y desacuerdos entre los colaboradores del departamento de logística cuando se dirigían a los clientes para la consulta de productos, precio y tiempo de estimación del pedido, se implementó tecnológicamente una aplicación móvil con geolocalización que tiene el módulo para el mantenimiento por cada producto implicando el tamaño, complemento y precio y en el perfil del cliente se puede visualizar la geolocalización del servicio de Google Maps del trayecto actual del repartidor hacia el destinatario del pedido, haciendo notar el tiempo referencial, nombre y teléfono del repartidor respaldando la teoría de Tabuyo (2015), para el proceso de control de pedidos es necesario un proceso de compra que aprovisiona mercancías fundamentales para cubrir las necesidades de los clientes a disposición así mismo en la investigación de Arciniégas [et al.] (2016) se tomó en cuenta, el desempeño de la red de una cadena de suministro de un abastecimiento de medicamentos; como solución para la problemática de esta investigación se realizó una mejora del nivel de desempeño de gestión logística estableciendo políticas para el control de mercaderías, existiendo un decremento de 6.3% para el indicador de entregas perfectamente recibidas y para esta investigación existió un decremento de 32.10%, para la investigación estudiada.

Para la implementación del proyecto de investigación, se implementó una aplicación móvil con geolocalización con la metodología MOBILE-D que se definió por 5 fases, exploración, establecimiento de stakeholders y levantamiento de requerimientos funcionales y no funcionales; inicialización, diagramación de arquitectura de la aplicación móvil y story cards, etc; producción, programación de la aplicación móvil con geolocalización bajo la arquitectura MVC; estabilización, adaptación de la aplicación móvil a la arquitectura móvil y

pruebas, ejecución de pruebas unitarias; lo cual tuvo un impacto positivo, en cuestión a las entregas por cada fecha que se definía en los story cards, para el diseño, programación, pruebas y verificación de pares por interface de la misma, lo cual pudo optimizar y ordenar la secuencia de la lógica de implementación de la aplicación, así mismo, Cueva (2018), usó la metodología MOBILE-D para la implementación de la aplicación móvil con geolocalización el cual le permitió entregar por fases, los entregables para un mayor control de ejecución y funcionalidad de la aplicación móvil siguiendo un conjunto de buenas prácticas de dicha metodología enfocada en aplicaciones móviles y en la investigación enfocada al proceso de control de pedidos sirvió para implementar la aplicación móvil mediante fases y hacer las pruebas correspondiente para probar la funcionalidad de la misma.

Para esta investigación, se usó la geolocalización como tecnología proveniente del servicio de Google Api Console con Google Maps usando diferentes servicios de ubicación instantánea que se vio reflejada en el diagrama de la arquitectura de la aplicación móvil enfatizada en las fases de la metodología MOBILE-D que pudo crear la lógica necesaria para la secuencia para programar el módulo del servicio del mismo, así mismo, Babativa [et al.] (2016), usó el sistema operativo Android la tecnología del GPS con diagramas UML para el modelado ágil de la implementación de la aplicación móvil con la visualización de un antes y después de la situación actual del transporte y movilidad ciudadana, en la investigación optada para el proceso de control de pedidos demostró la eficacia de una metodología de desarrollo de software como SCRUM acompañado con diagramas UML para marcar un antes y después de la situación actual de dicho proceso para una entrega óptima del software, la tecnología implementada en conjunto con la aplicación fue el GPS para el análisis y diseño de una buena mantenibilidad de software, en esta investigación se realizó una arquitectura de la aplicación móvil a realizar junto con la paquetería de tecnología GPS que es el servicio de Google Maps para optimizar el proceso de control de pedidos, analizando los requerimientos del caso de estudio.

Para la investigación, se optó por la metodología MOBILE-D para la implementación de la aplicación móvil que fue necesaria la documentación en la primera y segunda fase de la metodología para entender un antes y después sobre el impacto que tuvo la aplicación móvil durante el proceso, Baldoceda (2017), usó la metodología MOBILE-D para la implementación de la aplicación móvil a través de fases y entregables de la metodología MOBILE-D, mediante la documentación necesaria para poder haber logrado la comprensión misma de la aplicación móvil, en la investigación optada por el proceso de control de pedidos se logró implementar a través de referencias bibliográficas, documentación y antecedentes, la correcta implementación de la metodología MOBILE-D acompañado con una metodología de la investigación científica.

VI. CONCLUSIONES

1. Se determinó el efecto de una aplicación móvil con geolocalización para el proceso de control de pedidos destacando el adecuado marco de trabajo ágil para el proceso de control de pedidos, en combinación con los indicadores de la gestión logística para poder determinar el efecto de diferentes soluciones tecnológicas y logísticas para haber realizado un análisis comparativo en cuestión a la investigación estudiada.

2. El porcentaje de entregados completos se incrementó en 32.39%, logrando tener 83.43% de los entregados completos, con la implementación de la aplicación móvil con geolocalización se demostró el efecto positivo de esta con el proceso de control de pedidos en CRISS NEÓN S.A.C, además de incrementar las ventas y la fidelización de clientes.

3. El porcentaje de entregas perfectamente recibidas se decrementó en 32.10%, logrando tener 17.01% de entregas perfectamente recibidas, con la implementación de la aplicación móvil con geolocalización se demostró el efecto positivo de esta con el proceso de control de pedidos en CRISS NEÓN S.A.C, además se pudo disminuir la insatisfacción por parte de los clientes con el uso de la solución tecnológica.

VII. RECOMENDACIONES

Utilizar la inteligencia artificial para el mejoramiento de ubicación de rutas o geolocalización evaluando el impacto de la implementación comparando con la labor manual que lleva el proceso al aplicarlo y así innovar con la transformación digital para la agilización del proceso, ya que, en otras investigaciones, se demostró el gran uso de la IA desde pequeños a grandes procesos.

Incrementar la población así haciendo más valiosa la investigación y la muestra para la recolección de información de estos indicadores, así de esta manera estudiar los servicios que ofrece Google en la nube y ver su efectividad a mayor escala implementando una granja de negocios, probando la aplicación misma solo cambiando los productos, complementos que venden empresas del mismo sector como soporte de mejora usando los servicios que ofrece Google Maps para el análisis de datos para IA obteniendo resultados que beneficien a la misma organización.

Implementar en la aplicación móvil con geolocalización, la opción para puntuación de 5 estrellas y un comentario sobre el pedido que se le llegó por vía delivery para poder evaluar una mayor calidad del pedido generado y poder dar un diagnóstico sobre la fase de surtido del pedido que trata sobre la agilización del mismo.

Implementar junto con la aplicación móvil con geolocalización, un sistema web creando un sistema multiplataforma capaz de poder ser usada en todo momento ya sea en un ordenador personal o un celular que pueda agilizar el marco de trabajo respecto a las nuevas tendencias tecnológicas presentadas en hoy en día para la competividad de empresas del mismo sector.

Se sugiere implementar a un futuro, ampliar la investigación enfocado a la gestión logística, incrementando así los indicadores para poder dar un mayor control sobre la distribución, cumplimiento de despacho y de lar órdenes de compra recibidas que puedan tener la gestión logística para una mayor

satisfacción del usuario incrementando el porcentaje de pedidos analizando nuevas métricas.

REFERENCIAS

- ABDALLAH, Ali. Mobile food ordering apps: An empirical study of the factors affecting customer e-satisfaction and continued intention to reuse. ELSEVIER [en línea]. Estados Unidos: International Journal of Information Management 2020, 50(2020), pp.28-44 [consulta: 27 de abril de 2020]. ISSN: 0268-4012 Disponible en: https://www.sciencedirect.com/science/article/pii/S0268401219302038
- ABRAHAMSSON, Pekka et al. Mobile-D: An Agile Approach for Mobile Application Development [en línea]. VTT (Valtion Teknillinen Tutkimuskeskus, en inglés Technical Research Centre of Finland). 2017 [fecha de consulta: 14 de junio del 2020] Disponible en: https://webcache.googleusercontent.com/search?q=cache:PZ6FvPNbtnkJ:https://arxiv.org/pdf/1 709.06820+&cd=2&hl=es&ct=clnk&gl=pe
- ALARCÓN Andrea, URRUTIA Jorge y CALLEJAS Mauro. Aplicación Móvil para la Administración de Variables Físicas en Ciclismo al Aire Libre. Grupo Investigación en Software[en línea]. 2016, Vol. 27 (175-182) [fecha de consulta 03 de mayo del 2020]. Disponible en: https://scielo.conicyt.cl/pdf/infotec/v27n4/art19.pdf
- ALCOCER, Patricio y KNUDSEN, José. Desempeño integral de los procesos logísticos en una cadena de suministro. Universidad Libre-Barranquilla [en línea]. 2019, 1(X) [fecha de consulta 03 de octubre del 2020]. ISSN: 1815-5936. Disponible en: http://scielo.sld.cu/pdf/rii/v40n1/1815-5936-rii-40-01-78.pdf
- ARCINIÉGAS, Luis [et al.]. Medición del desempeño de la red de suministros de medicamentos en un hospital público de tercer nivel en la ciudad de Bogotá, a través del cuadro de mando integral. Universidad Libre-Barranquilla [en línea]. 2016, 12(20) [fecha de consulta 03 de octubre del 2020]. ISSN: 1909-2458. Disponible en: https://www.researchgate.net/publication/316859986_Medicion_del_desempeno_de_la_red_de_suministros_de_medicamentos_en_un_hospital_publico_de_tercer_nivel_en_la_ciudad_de_Bogota_a_traves_del_cuadro_de_mando_integral

- ARGÜESO, Mónica et.al. MATEMÁTICAS II PARA CIENCIAS SOCIALES. 2° BACHILLERATO LOMCE [en línea]. ESPAÑA: Ediciones Paraninfo, S.A., 2017. [Fecha de consulta 12 mayo 2020]. ISBN: 9788428335508. Disponible en: https://books.google.com.pe/books?id=5FU7DwAAQBAJ&dq=MATEM%C3%81TICAS+II+PARA+CIENCIAS+SOCIALES.+2%C2%B0+BACHILLERATO+LOMCE&hl=es&source=gbs_navlink s_s
- BABATIVA, Angélica et al. Desarrollo Ágil de una Aplicación para Dispositivos Móviles. Caso de Estudio: Taxímetro Móvil. Universidad Distrital Francisco José de Caldas [en línea]. 2016, 21(3) [fecha de consulta 03 de mayo del 2020]. ISSN: 2344-8393. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=5677874
- BALDOCEDA, Jean. Desarrollo de un aplicativo móvil basado en la metodología Mobile-D para la gestión de reservas del hotel Caribe de Huaral. [en línea]. Tesis (Título de Ingeniero de Sistemas y Cómputo). Perú: Universidad Inca Garcilaso de la Vega, 2017. 124p. [Consultado 2 de junio de 2020]. Disponible en: http://repositorio.uigv.edu.pe/handle/20.500.11818/1800
- BALLOU, R. Logística: administración de la cadena de suministro [en línea]. 5ta ed. México: Pearson Educación, 2004. [Fecha de consulta 13 abril 2020]. ISBN: 9702605407. Disponible en: https://books.google.com.pe/books?id=ii5xqLQ5VLgC&dq=Log%C3%ADstica:+administraci%C3%B3n+de+la+cadena+de+suministro&hl=es&source=gbs_navlinks_s
- BELTRÁN, G. Geolocalización Online: La importancia del dónde [en línea]. Barcelona: Editorial UOC. 2016. [Fecha de consulta 16 abril 2020]. ISBN: 9788491161585 Disponible en: https://books.google.com.pe/books?id=5FLeDQAAQBAJ&dq=1Geolocalizaci%C3%B3n+Online:+La+importancia+del+d%C3%B3nde&hl=es&source=gbs_navlinks_s
- BIESSEK. A. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2 [en línea]. Reino Unido:

Packt Publishing Ltd, 2019. [Fecha de consulta 21 abril 2020]. ISBN: 9781788990523. Disponible en: https://books.google.com.pe/books?id=pF6vDwAAQBAJ&dq=Flutter+for+Beginners:+An+introd uctory+guide+to+building+crossplatform+mobile+applications+with+Flutter+and+Dart+2&hl=es&source=gbs_navlinks_s

- BILBAO, J., ESCOBAR, P. Investigación y educación superior [en línea]. Estados Unidos: Lulu.com. 2020 [Fecha de consulta 02 mayo 2020]. ISBN: 9781678103903. Disponible en: https://books.google.com.pe/books?id=W67WDwAAQBAJ&dq=Investigaci%C3%B3n+y+educa ci%C3%B3n+superior&hl=es&source=gbs_navlinks_
- CABALLERO, C. UF1305 - Programación con lenguajes de guión en páginas web [en línea]. Madrid: Ediciones Praninfo S.A., 2015. [Fecha de consulta 16 abril 2020]. ISBN: 9788428396875. Disponible en: https://books.google.com.pe/books?id=N1GACwAAQBAJ&dq=UF1305+-+Programaci%C3%B3n+con+lenguajes+de+gui%C3%B3n+en+p%C3%A1ginas+web&hl=es&s ource=gbs_navlinks_s
- CAPACHO, J., NIETO, W. Diseño de base de datos [en línea]. España: Universidad del Norte, 2017. [Fecha de consulta 21 abril 2020]. ISBN: 9789587418255. Disponible en: https://books.google.com.pe/books?id=TLBJDwAAQBAJ&dq=Dise%C3%B1o+de+base+de+dat os&hl=es&source=gbs_navlinks_s
- CHAVAN, Varsha [et al.]. Implementing Customizable Online Food Ordering System Using Web Based Application. International Journal of Innovative Science, Engineering & Technology [en línea]. 2015, Vol. 2(4). 722-727. [fecha de consulta 05 de mayo del 2020]. ISSN: 23487968. Disponible en: https://www.researchgate.net/publication/340479626_Implementation_of_Responsive_Online_Food_Ordering_Application_with_Social_Media_Integration
- CUEVA, Jimmy. Aplicación móvil con geolocalización, mediante la metodología Mobile-D, para la gestión de visitas médicas en la empresa Laboratorios Siegfried S.A.C. [en línea]. Tesis (Título profesional de Ingeniero de Sistemas). Perú: Universidad César Vallejo, 2018. 122p. [Consultado 05

de setiembre 2020]. Disponible en: http://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/37415/Cueva_HJ.pdf?sequence=1&isAllowed=y

- COMBAUDON, S. MySQL 5.7: administración y optimización [en línea]. España: Ediciones ENI, 2018. [Fecha de consulta 23 abril 2020]. ISBN: 9782409008467. Disponible en: https://books.google.com.pe/books/about/MySQL_5_7.html?id=PvKjuAIA-PwC&redir_esc=y
- ESCOBAR, Guillermo, CAMPAÑA, Alex. Diseño e Implementación de una Aplicación Móvil que cumpla con la función de estación en tierra para el monitoreo de UAV'S en el centro de investigación y desarrollo de la fuerza aérea ecuatoriana [en línea]. 2014, 1 - 4 [fecha de consulta 14 de junio 2020]. Disponible en: http://repositorio.espe.edu.ec/xmlui/handle/21000/8189
- GANIVET, J. UF0929 - Gestión de pedidos y stock. [en línea]. España: Editorial Elearning S.L., 2014 [Fecha de consulta 16 abril 2020]. Disponible en: https://books.google.com.pe/books?id=b39XDwAAQBAJ&printsec=frontcover&dq=UF0929+-+Gesti%C3%B3n+de+pedidos+y+stock.&hl=es&sa=X&ved=2ahUKEwiTiPyDlqPqAhXdGrkGHY4jBnEQ6AEwAXoECAUQAg#v=onepage&q=entregas%20perfectas&f=false
- GARRIDO, P. Comenzando a programar con JAVA [en línea]. Alicante: Universidad Miguel Hernández, 2015. [Fecha de consulta 16 abril 2020]. ISBN: 9788416024247. Disponible en: https://books.google.com.pe/books?id=4v8QCgAAQBAJ&dq=Comenzando+a+programar+con+ JAVA&hl=es&source=gbs_navlinks_s
- GARZA, A. Manual de técnicas de investigación para estudiantes de ciencias sociales y humanidades [en línea]. México: El Colegio de México AC, 2009 [Fecha de consulta 02 mayo 2020]. ISBN: 9789681212988. Disponible en: https://books.google.com.pe/books?id=jdaQtk8RK2sC&dq=investigaci%C3%B3n+explicativa&hl=es&source=gbs_navlinks_s
- GEORGE, D., and MALLERY, P. IBM SPSS Statistics 25 Step by Step: A Simple Guide and Reference [en línea]. United States: Routledge, 2018.

[Fecha de consulta 21 junio 2020]. ISBN: 9781351033886. Disponible en: https://books.google.com.pe/books?id=ntNyDwAAQBAJ&dq=ibm+spss+25&hl=es&source=gbs _navlinks_s

- GÓMEZ, V., LÓPEZ, E. Teoría y problemas resueltos de matemática aplicada y estadística para farmacia [en línea]. España: Ediciones Paraninfo, 2017 [Fecha de consulta 21 de junio del 2020]. ISBN: 9788428327787. Disponible en: https://books.google.com.pe/books?id=dAMoDwAAQBAJ&dq=Teor%C3%ADa+y+problemas+r esueltos+de+matem%C3%A1tica+aplicada+y+estad%C3%ADstica+para+farmacia&hl=es&sou rce=gbs_navlinks_s
- GUIMERÁ, A. Iniciación a Android en Kotlin. Casos prácticos [en línea]. Madrid: Ediciones Paraninfo, S.A., 2018. [Fecha de consulta 16 abril 2020]. ISBN: 9788428340922. Disponible en: https://books.google.com.pe/books?id=GVJ1DwAAQBAJ&dq=Iniciaci%C3%B3n+a+Android+en +Kotlin.+Casos+pr%C3%A1cticos&hl=es&source=gbs_navlinks_s
- GUPTA, Mitali. A Study on Impact of Online Food delivery app on Restaurant Business special reference to zomato and swiggy.Cosmos Impact Factor [en línea]. 2019, 6(10). 889-893. [fecha de consulta 04 de mayo del 2020]. ISSN: 2349-5138. Disponible en: https://webcache.googleusercontent.com/search?q=cache:o8h8-vdHbdQJ:https://ijrar.com/upload_issue/ijrar_issue_20542895.pdf+&cd=1&hl=es&ct=clnk&gl=pe
- GUTIÉRREZ, F. Apuntes de conceptos básicos para muestreo estadístico: Para estudiantes de programas de doctorado en ciencias administrativas [en línea]. Estados Unidos: Lulu.com, 2015. [Fecha de consulta 12 mayo 2020]. ISBN: 9781329139152. Disponible en: https://books.google.com.pe/books?id=EPUCCwAAQBAJ&dq=Apuntes+de+conceptos+b%C3 %A1sicos+para+muestreo+estad%C3%ADstico:+Para+estudiantes+de+programas+de+doctor ado+en+ciencias+administrativas&hl=es&source=gbs_navlinks_s

- HUAMANI, Joesvel. Sistema web para la gestión de pedidos en la empresa Impresiones Franco S.A.C. [en línea]. Tesis (Título profesional de Ingeniero de Sistemas). Perú: Universidad César Vallejo, 2018. 157p. [Consultado 05 de setiembre 2020]. Disponible en: http://repositorio.ucv.edu.pe/handle/20.500.12692/35498
- HERNÁNDEZ, Arturo et al. METODOLOGÍA DE LA INVESTIGACIÓN CIENTÍFICA [en línea]. 2da ed. España: 3ciencias, 2018. 174pp [Fecha de consulta 21 de junio del 2020]. ISBN: 9788494825705. Disponible en: https://books.google.com.pe/books?id=y3NKDwAAQBAJ&dq=METODOLOG%C3%8DA+DE+LA+INVESTIGACI%C3%93N&hl=es&source=gbs_navlinks_s
- HERNÁNDEZ, Roberto, FERNÁNDEZ Carlos y BAPTISTA Pilar. Metodología de la Investigación. 6ta ed. [en línea]. México: Mc Graw-Hill., 2014. [Fecha de consulta 08 junio 2020]. ISBN: 9781456223960. Disponible en: https://periodicooficial.jalisco.gob.mx/sites/periodicooficial.jalisco.gob.mx/files/metodologia_de_l a_investigacion_-_roberto_hernandez_sampieri.pdf
- HUMAYUN Kabir, RABAYA Sultana y MAHADIR Mir. Home Delivery System: An Android Based Mobile Application [en línea]. Tesis (Bachiller en Ciencias de la computación e Ingeniería). Estados Unidos. Daffodil International Univesity, 2019. 46p. [Consultado 10 de abril 2020]. Disponible en: https://www.researchgate.net/publication/263027693_Mobile_Food_Ordering_Application_using _Android_OS_Platform
- IBAÑEZ, J. Métodos, técnicas e instrumentos de la investigación criminológica [en línea]. España: Editorial Dikynson, 2015 [Fecha de consulta 20 mayo 2020]. ISBN: 9788490318485. Disponible en: https://books.google.com.pe/books?id=ggTdBAAAQBAJ&dq=M%C3%A9todos,+t%C3%A9cnicas+e+instrumentos+de+la+investigaci%C3%B3n+criminol%C3%B3gica.&hl=es&source=gbs_n avlinks_s
- IGLESIAS, A. Distribución y logística [en línea]. Madrid: ESIC Editorial, 2016. [Fecha de consulta 13 abril 2020]. ISBN: 9788473569439.Disponible en: https://books.google.com.pe/books?id=YTXhCwAAQBAJ&dq=Distribuci%C3%B3n+y+log%C3 %ADstica&hl=es&source=gbs_navlinks_s

- INGA, Claudia. Coronavirus: Aumenta demanda de pedidos por ‘apps’ de delivery y toman medidas sanitarias para las entregas [en línea]. El Comercio, 2020. [consulta: 14 de abril de 2020]. Disponible en: https://elcomercio.pe/economia/dia-1/coronavirus-coronavirus-aumenta-demanda-de-pedidos-por-apps-de-delivery-y-toman-medidas-sanitarias-para-las-entregas-delivery-aplicativos-de-delivery-glovo-rappi-noticia/#:~:text=Aplicativos%20registran%20incremento%20de%20pedidos,frente%20a%20un%20d%C3%ADa%20normal.&text=Los%20principales%20aplicativos%20de%20delivery,y%20otros%20productos%20de%20higiene.
- JÍMENEZ, A.Desarrollo de una App para móviles de control de una plataforma IoT. Proyecto de grado de Ingeniería Informática. UPC: FIC, 2016 2016. Disponible en: https://upcommons.upc.edu/handle/2117/86513
- JOYCE, J., JOSEPH, J. Learn Ionic 2: Develop Multi-platform Mobile Apps [en línea]. United States: Apress, 2017. [Fecha de consulta 21 abril 2020]. 105pp. ISBN: 9781484226179. Disponible en: https://books.google.com.pe/books?id=WvGqDgAAQBAJ&dq=Learn+Ionic+2:+Develop+Multiplatform+Mobile+Apps&hl=es&source=gbs_navlinks_s
- LAÍNEZ, J. Desarrollo de Software Ágil: Extremme Programming y Scrum [en línea]. 2da ed. España: IT Campus Academy, 2015 [Fecha de consulta 23 abril 2020]. ISBN: 9781519620149. Disponible en: https://books.google.com.pe/books?id=M4fJCgAAQBAJ&dq=Desarrollo+de+Software+%C3%81gil:+Extreme+Programming+y+Scrum&hl=es&source=gbs_navlinks_s
- LÓPEZ, E. 100 Preguntas y Respuestas para trabajar como Desarrollador Android: o contratar al candidato adecuado [en línea]. Madrid: Editorial Enrique López Mañas, 2015. 109pp. [Fecha de consulta 16 abril 2020]. Disponible en: https://books.google.com.pe/books?id=NCPCgAAQBAJ&dq=100+Preguntas+y+Respuestas+para+trabajar+como+Desarrollador+Android:+o+contratar+al+candidato+adecuado&hl=es&source=gbs_navlinks_s
- LUSTHAUS, C. Mejorando El Desempeño de Las Organizaciones: Método de Autoevaluación [en línea]. United States: IDRC, 2001 [Fecha de consulta 08 junio 2020]. ISBN: 9780889369504. Disponible en:

https://books.google.com.pe/books?id=uy90rmtckLIC&dq=Mejorando+El+Desempe%C3%B1o+
de+Las+Organizaciones:+M%C3%A9todo+de+Autoevaluaci%C3%B3n&hl=es&source=gbs_na vlinks_s

- MARÍN, B. Preparación de pedidos y venta de productos [en línea]. España: Ediciones Paraninfo, S.A., 2014. [Fecha de consulta 20 mayo 2020]. ISBN: 9788428328890. Disponible en: https://books.google.com.pe/books?id=-AQbBAAAQBAJ&dq=Preparaci%C3%B3n+de+pedidos+y+venta+de+productos&hl=es&source =gbs_navlinks_s
- entr
- MORONEY, L. The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform [en línea]. United States: Apress, 2017. [Fecha de consulta 21 abril 2020]. ISBN: 9781484229439. Disponible en: https://books.google.com.pe/books?id=ox0-DwAAQBAJ&dq=firebase&hl=es&source=gbs_navlinks_s
- ÑAUPAS, Humberto et al. Metodología de la Investigación cuantitativa-cualitativa y redacción de la tesis [en línea]. España: Ediciones de la U.2019. [Fecha de consulta 29 junio 2020]. ISBN: 9789587628777. Disponible en: https://books.google.com.pe/books?id=KzSjDwAAQBAJ&dq=enfoque+cuantitativo&hl=es&sour ce=gbs_navlinks_s
- PAIMA, Dorcas. Sistema web para el proceso de abastecimiento en la Municipalidad Provincial del Callao [en línea]. Tesis (Título profesional de Ingeniero de Sistemas). Perú: Universidad César Vallejo, 2019. 128p. [Consultado 05 de setiembre 2020]. Disponible en: http://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/43547/Paima_RDM.pdf?sequence=1&isAllowed=y
- PANTALEO, G., RINAUDO, L. Ingeniería de Software [en línea]. España: Alfaomega Grupo Editor, 2015. [Fecha de consulta 23 abril 2020]. ISBN: 9789871609789. Disponible en: https://books.google.com.pe/books?id=a8j2DQAAQBAJ&dq=Ingenier%C3%ADa+de+Software &hl=es&source=gbs_navlinks_s
- PARRAGUEZ, Simona et al. EL ESTUDIO Y LA INVESTIGACIÓN DOCUMENTAL: ESTRATEGIAS METDOLÓGICAS Y HERRAMIENTAS TIC

[en línea]. Perú: Gerardo Chunga Chinguel, 2017 [Fecha de consulta 20 mayo 2020]. ISBN: 9786120026038. Disponible en: https://books.google.com.pe/books?id=v35KDwAAQBAJ&dq=EL+ESTUDIO+Y+LA+INVESTIG ACI%C3%93N+DOCUMENTAL:+ESTRATEGIAS+METDOL%C3%93GICAS+Y+HERRAMIENT AS+TIC&hl=es&source=gbs_navlinks_s

- PAU, J., NAVASCUÉS, R. Manual de logística integral [en línea]. España: Ediciones Díaz de Santos, 1998 [Fecha de consulta 16 abril 2020]. ISBN: 9788479783457. Disponible en: https://books.google.com.pe/books?id=dxTImJ4ipCMC&dq=Manual+de+log%C3%ADstica+integral&hl=es&source=gbs_navlinks_s
- PEÑA, S. UF1469 - SGBD e instalación [en línea]. Madrid: Ediciones Paraninfo, S.A., 2017. [Fecha de consulta 21 abril 2020]. ISBN: 9788428396561. Disponible en: https://books.google.com.pe/books?id=yVPVDQAAQBAJ&dq=UF1469+-+SGBD+e+instalaci%C3%B3n&hl=es&source=gbs_navlinks_s
- ROCHA, A., GUARDA, T. Proceedings of the International Conference on Information Technology & Systems (ICITS 2018) [en línea]. Alemania: Springer, 2018. [Fecha de consulta 23 abril 2020]. ISBN: 9783319734507. Disponible en: https://books.google.com.pe/books?id=gj5FDwAAQBAJ&dq=Proceedings+of+the+International +Conference+on+Information+Technology+%26+Systems+(ICITS+2018)&hl=es&source=gbs_n avlinks_s
- RODRÍGUEZ, A. Respuesta educativa a la diversidad del alumnado en Educación Secundaria Obligatoria. [en línea]. España: Universidad Almería, 2017. [Fecha de consulta 08 junio 2020]. ISBN: 9788416642540. Disponible en: https://books.google.com.pe/books?id=mLEnDgAAQBAJ&printsec=frontcover&dq=Respuesta+ educativa+a+la+diversidad+del+alumnado+en+Educaci%C3%B3n+Secundaria+Obligatoria&hl =es&sa=X&ved=2ahUKEwi1mcjozZ7qAhUqIbkGHdmnBk4Q6AEwAHoECAUQAg#v=onepage&

q=Respuesta%20educativa%20a%20la%20diversidad%20del%20alumnado%20en%20Educaci %C3%B3n%20Secundaria%20Obligatoria&f=false

- ROSS, S. Introducción a la estadística [en línea]. España: Reverte, 2018. [Fecha de consulta 20 junio 2020]. ISBN: 9788429194241. Disponible en: https://books.google.com.pe/books?id=Ed3eDwAAQBAJ&pg=PA7&dq=estratificaci%C3%B3n+estad%C3%ADstica&hl=es&sa=X&ved=2ahUKEwj74c6y7ZDqAhXiIbkGHUCMBoEQ6AEwAnoECAYQAg#v=onepage&q=estratificaci%C3%B3n%20estad%C3%ADstica&f=false
- SANTIAGO, R., TRABALDO, S. Mobile learning: Nuevas realidades en el aula Innovación educativa [en línea]. Madrid: Digital-Text, 2015[Fecha de consulta 16 abril 2020] . ISBN: 9788449451454. Disponible en: https://books.google.com.pe/books?id=AULhBgAAQBAJ&dq=Mobile+learning:+Nuevas+realidades+en+el+aula.+Innovaci%C3%B3n+educativa&hl=es&sa=X&ved=2ahUKEwiu9MXctp7qAhW YKLkGHWlgDCIQ6AEwAHoECAIQAg
- SANTOS, Guadalupe. Validez y Confiabilidad del cuestionario de calidad de vida SF-36 en mujeres con LUPUS, Puebla [en línea]. Tesis Pregrado. Puebla: Benemérita Universidad Autónoma de Puebla, 2017. [Consultado 20 de mayo 2020]. Disponible en: https://webcache.googleusercontent.com/search?q=cache:z9-svYc3cAJ:https://www.fcfm.buap.mx/assets/docs/docencia/tesis/ma/GuadalupeSantosSanchez.p df+&cd=1&hl=es&ct=clnk&gl=pe
- SERNA, S. Diseño de interfaces en aplicaciones móviles [en línea]. España: Grupo Editorial RA-MA. [Fecha de consulta 16 abril 2020]. ISBN: 9788499646152. Disponible en: https://books.google.com.pe/books?id=SIfDwAAQBAJ&dq=Dise%C3%B1o+de+interfaces+en+aplicaciones+m%C3%B3viles&hl=es&source=gbs_navlinks_s 20
- SMYTH, N. Android Studio 3.6 Development Essentials - Kotlin Edition: Developing Android 10 (Q) Apps Using Android Studio 3.6, Kotlin and Android Jetpack [en línea]. Estados Unidos: eBookFrenzy, 2020. [Fecha de consulta

16 abril 2020]. ISBN: 9781951442132 Disponible en: https://books.google.com.pe/books?id=UYLVDwAAQBAJ&dq=android+studio&hl=es&source=g bs_navlinks_s

- SUÁREZ, Y., MEDINA, D., and HERNÁNDEZ, P. Sistema automatizado para la gestión del mantenimiento de equipos (módulos administración y solicitud de servicio) [en línea]. 2015, diciembre, 24. 85-90 [Fecha de consulta 23 abril 2020]. ISSN: 1010-2760. Disponible en: https://revistas.unah.edu.cu/index.php/rcta/article/view/395
- TABORDA, R., PÉREZ, G. Ejercicios de econometría: Material de estudio, LECCIONES FACULTAD DE ECONOMÍA [en línea]. Colombia: Editorial Universidad del Rosario, 2016. [Fecha de consulta 20 mayo 2020]. ISBN: 9789587387124. Disponible en: https://books.google.com.pe/books?id=Ll0yDwAAQBAJ&dq=Ejercicios+de+econometr%C3%ADa:+Material+de+estudio,+LECCIONES+FACULTAD+DE+ECONOM%C3%8DA&hl=es&source =gbs_navlinks_s
- TABUYO, M. MF1182_3 - Organización y gestión de los procesos de mantenimiento de las instalaciones eléctricas en el entorno de edificios y con fines especiales [en línea]. España: Editorial Elearning, S.L., 2015. [Fecha de consulta 13 abril 2020]. ISBN: 9788416492978. Disponible en: https://books.google.com.pe/books?id=-PM-DwAAQBAJ&dq=MF1182_3+-+Organizaci%C3%B3n+y+gesti%C3%B3n+de+los+procesos+de+mantenimiento+de+las+instal aciones+el%C3%A9ctricas+en+el+entorno+de+edificios+y+con+fines+especiales&hl=es&sourc e=gbs_navlinks_s
- ULMER, Marlin et al. The Restaurant Meal Delivery Problem: Dynamic Pick-Up and Delivery with Deadlines and Random Ready Times. [en línea]. Estados Unidos: ResearchGate, 2017, pp.1-45 . [consulta: 03 de mayo de 2020]. Disponible en: https://www.researchgate.net/publication/320356850_The_Restaurant_Meal_Delivery_Problem _Dynamic_Pick-Up_and_Delivery_with_Deadlines_and_Random_Ready_Times

- VALBUENA, R. La investigación científica avanzada: Con introducción a los programas de investigación científica, la investigación internivel y el razonamiento artificial [en línea]. Venezuela: ROIMAN VALBUENA.2015. [Fecha de consulta 02 mayo 2020]. ISBN: 9789801282112. Disponible en: https://es.slideshare.net/Riman2/la-investigacin-cientficaavanzada-primera-edicin-2015
- VINALK, Anita et al. The Study of Interest of Consumers In Mobile Food Ordering Apps [en línea]. 2019, 8(1). 3424-3429. [fecha de consulta 04 de mayo del 2020]. ISSN: 2277-3878. Disponible en: https://webcache.googleusercontent.com/search?q=cache:qLK8UbJPJFsJ:https://www.ijrte.org/ wp-content/uploads/papers/v8i1/A9219058119.pdf+&cd=1&hl=es&ct=clnk&gl=pe
- VITALTA, C. Análisis de datos [en línea]. España: CIDE, 2016. [Fecha de consulta 21 de junio del 2020]. ISBN: 9786079367930. Disponible en: https://books.google.com.pe/books?id=9W84DgAAQBAJ&dq=carlos+vitalta&hl=es&source=gbs _navlinks_s
- WASPADADI, B., YUNIAR, L. Implementing DSS for selecting suitable delivery services to support smart e-commerce. International Jounal of Research Science & Management [en línea]. Estados Unidos: ResearchedID, 2018, 5(1), pp.16-22 [consulta: 14 de abril de 2020]. ISSN: 234- 5197 Disponible en: https://www.academia.edu/36098506/IMPLEMENTING_DSS_FOR_SELECTING_SUITABLE_D ELIVERY_SERVICES_TO_SUPPORT_SMART_E-COMMERCE
- ZAPATA, Rosa et al. JORNADAS INTERNACIONALES DE INVESTIGACIÓN DE INVESTIGACIÓN EN EDUCACIÓN Y SALUD [en línea].44va ed. España: Universidad Almería, 2015 [Fecha de consulta 08 junio 2020]. ISBN: 9788416027811. Disponible en: https://books.google.com.pe/books?id=yD0wBwAAQBAJ&dq=JORNADAS+INTERNACIONALES+DE+INVESTIGACI%C3%93N+DE+INVESTIGACI%C3%93N+EN+EDUCACI%C3%93N+Y+ SALUD&hl=es&source=gbs_navlinks_s

ANEXOS

ANEXO N°01: Matriz de Consistencia

<table>
<tr><th>PROBLEMAS</th><th>OBJETIVOS</th><th>HIPÓTESIS</th><th>VARIABLES</th><th>DIMENSIONES</th><th>INDICADORES</th><th>METODOLOGÍA</th></tr>
<tr><td>Problema General:

¿Cuál es el efecto de una aplicación móvil con geolocalización en el proceso de control de pedidos en CRISS NEÓN S.A.C.?</td><td>Objetivo General:

Determinar el efecto del uso de la aplicación móvil con geolocalización para mejorar el proceso del control de pedidos en CRISS NEÓN S.A.C.</td><td>Hipótesis General:

La aplicación móvil con geolocalización mejora significativamente el proceso de control de pedidos en CRISS NEÓN S.A.C.</td><td>Variable Independiente:

Aplicación móvil</td><td></td><td></td><td rowspan="4">TIPO DE INVESTIGACIÓN:
Aplicada

DISEÑO DE INVESTIGACIÓN:
Pre-Experimental

ENFOQUE DE INVESTIGACIÓN:
Cuantitativo

NIVEL DE INVESTIGACIÓN:
Explicativo

TÉCNICA:
Fichaje

INSTRUMENTO:
Ficha de registro

UNIDAD DE MEDIDA:
1 pedido

POBLACIÓN:
500 pedidos

MUESTRA:
218 pedidos</td></tr>
<tr><td>Específicos</td><td>Específicos</td><td>Específicos</td><td rowspan="3">4.2. Variable Dependiente:

Proceso de control de pedidos</td><td>D1: Entrada del pedido</td><td>Incremento del porcentaje de entregados Completos</td></tr>
<tr><td>¿Cuál es el efecto de una aplicación móvil con geolocalización en el incremento del porcentaje de entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C?

¿Cuál es el efecto de una aplicación móvil con geolocalización en el decremento de entregas perfectamente recibidas en el proceso de control de pedidos en CRISS NEÓN S.A.C?</td><td>Determinar el efecto de una aplicación móvil con geolocalización en el incremento del porcentaje de entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C

Determinar el efecto de una aplicación móvil con geolocalización en el decremento de entregas perfectamente recibidas en el proceso de control de pedidos en CRISS NEÓN S.A.C</td><td>La aplicación móvil con geolocalización incrementa el porcentaje de entregados completos en el proceso de control de pedidos en CRISS NEÓN S.A.C.

La aplicación móvil con geolocalización decrementa el porcentaje de entregas perfectamente recibidas en el proceso de control de pedidos en CRISS NEÓN S.A.C.</td><td>D2: Estado del pedido</td><td>Decremento del porcentaje de entregas perfectamente recibidas</td></tr>
</table>

ANE XO N°02: Entrevista

Criss NEON S.A.C.
FABRICANTES DE AVISOS LUMNOSOS

- LETREROS EN NEON
- PANAFLEX
- ACRILICO
- PANELES
- PLOTEO DE VINIL
- DECCRACION DE NEON
- GIGANTOGRAFIAS
- LETRAS METALICAS

ENTREVISTA

Institución : CRISS NEÓN S.A.C.
Dirección : Jirón Huascarán 756, La Victoria
Fecha : 30 de abril del 2020
Objetivo : El objetivo de esta entrevista es obtener información del proceso de pedidos de la Empresa CRISS NEÓN S.A.C.
Investigador : Pachas Sifuentes Ricardo David

Entrevistado : **Mejía Zapata Carlos**
Cargo : **Gerente general**

1. ¿Cuál es su función en el cargo que ocupa dentro de la organización?

Ocupo el cargo de Gerente General de CRISS NEÓN, y soy el encargado de la gestión del bienestar de la empresa, garantía del servicio que se brinda a todos nuestros clientes, así como también sobre la calidad y garantía de los materiales e insumos que vendemos.

2. ¿Qué tipo de productos y servicios brinda la Empresa a sus clientes?

Ofrecemos productos basados en la confección de letreros con Leds digitales, RGB, tubos de Neón, panaflex, gigantografías, acrílico, metal, etc. y también ofrecemos servicios de mantenimiento, diseños y decoraciones basadas en leds.

3. ¿Quiénes son los colaboradores involucradas en dicho proceso y cuáles son sus funciones respectivas en el pedido que se realiza?

Nuestros colaboradores desde el momento que el cliente realiza su pedido pasan ante todo por el área de logística, dependiendo del producto que solicita para dar un alcance al cliente, de cómo sería el producto

que solicita. Luego el Gerente Administrativo le confeccionará una Proforma en base al producto que el mismo solicita, la cual se le enviará por correo y una vez aceptada por el cliente, éste tendrá que abonar el 50% de la Proforma para comenzar con la fabricación del mismo. Luego los técnicos y personal de logística tendrán que coordinar con el área de ventas para la entrega del pedido.

4. ¿Cuáles son las etapas del proceso de pedidos?

La Empresa está publicitada en todas las redes sociales y es el Gerente Administrativo el que se encarga de recibir los pedidos via "online" y otros que yá son clientes antiguos que hacen su pedido por teléfono, whatssap ó por correo. Luego él osea el Ing. se encarga de coordinar con el área logística del producto determinado del usudiodel pedido solicitado y si el cliente lo aprueba se procede a realizarle su Proforma.
Y una vez que el cliente lo coordinó con el Dpto. de Ventas y se pusieron de acuerdo, se comienza con la elaboración del pedido, se coordina con el cliente la hora y dirección a enviarle y se le envía por delivery ó con un vendedor y el chofer si se trata de entregarlo en forma local y finalmente elaborará el reporte correspondiente.

5. ¿Quién toma realmente el pedido?

Cuando el Ingeniero encargado de las redes sociales y la publicidad toma el pedido, lo apunta en un papel y procede con la confección de la Proforma para el Cliente y a partir de este momento se comienza una especie de coordinación directa con el cliente para poder llegar a un punto neutro ó punto de equilibrio para ponerse de acuerdo con el monto a cobrarse, yá que por lo general los clientes siempre piden un descuento ó reajuste en la Proforma.

6. ¿Puede saber rápidamente cuantas entregas perfectas se han hecho?

No, porque al momento de la entrega se dan ocasiones que no se entregó satisfactoriamente o se han vuelto a ordenar los pedidos, también la mala administración y deficiente del vendedor en el stock al momento de hacer ventas extras, a la vez la poca capacitación del personal de delivery, equivocación de

distritos y direcciones, falta de coordinación de tiempo, falta de especificación de domicilio del cliente y se nota la escasa presión en la orientación al buen servicio al cliente.

7. ¿Cuenta la empresa con alguna estrategia para el proceso de pedidos?

En la actualidad no contamos con ninguna estrategia, sólo nuestra publicidad en las redes y los 40 años que tiene la Empresa en el mercado son la mejor garantía y experiencia que tenemos. Todo el proceso es como lo detallé anteriormente, el Ingeniero toma el pedido, datos del cliente, coordina la dirección, y se le prepara su Cotización con el fin de que el cliente lo acepté y esperamos la contra oferta hasta llegar a un punto final y comenzar con la confección del Letrero.

8. ¿El sistema vigente organizado en la empresa se ajusta a los requerimientos de esta?

No tenemos adaptado un sistema definido para administrar o controlar nuestros pedidos, solo se realiza de manera empírica.

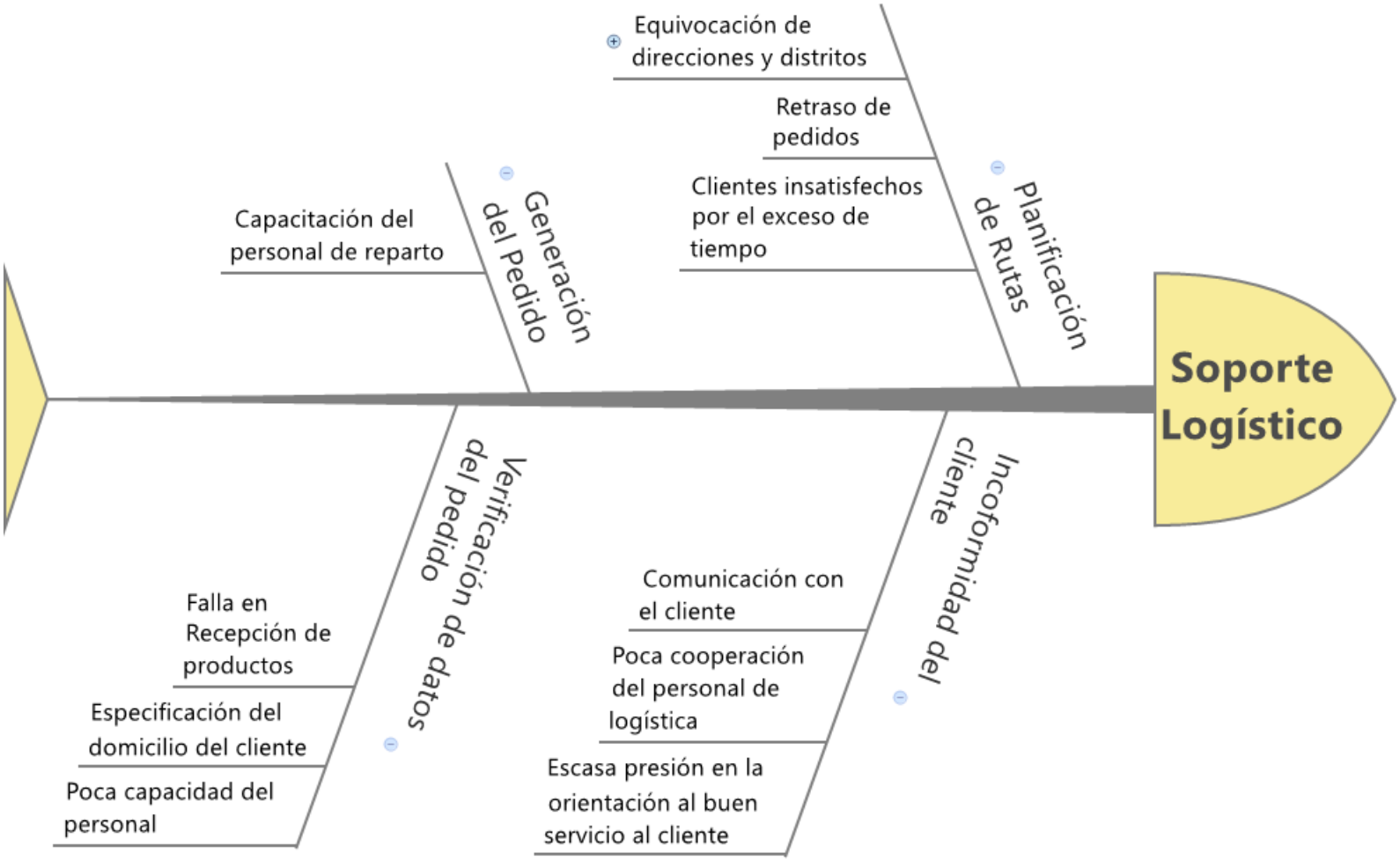
9. ¿Han pensado uds. Implementar alguna solución para el proceso de pedidos?

Por supuesto, pero lamentablemente es muy costoso elaborar dicho proceso, consultamos con un ingeniero y nos señaló que, en caso de una reingeniería del proceso de pedidos, se tiene que dar una capacitación y para eso tendrían que dejar de realizar sus labores o pagarles horas extras para que se queden, lo cual tampoco es factible para su labor que la producción no podrá esperar debido a la gran solicitud de pedidos que se tiene. Lo que se desea es un programa que permita comunicarse de forma inmediata con los vendedores para que la información llegue rápido y verificarse ahí mismo.

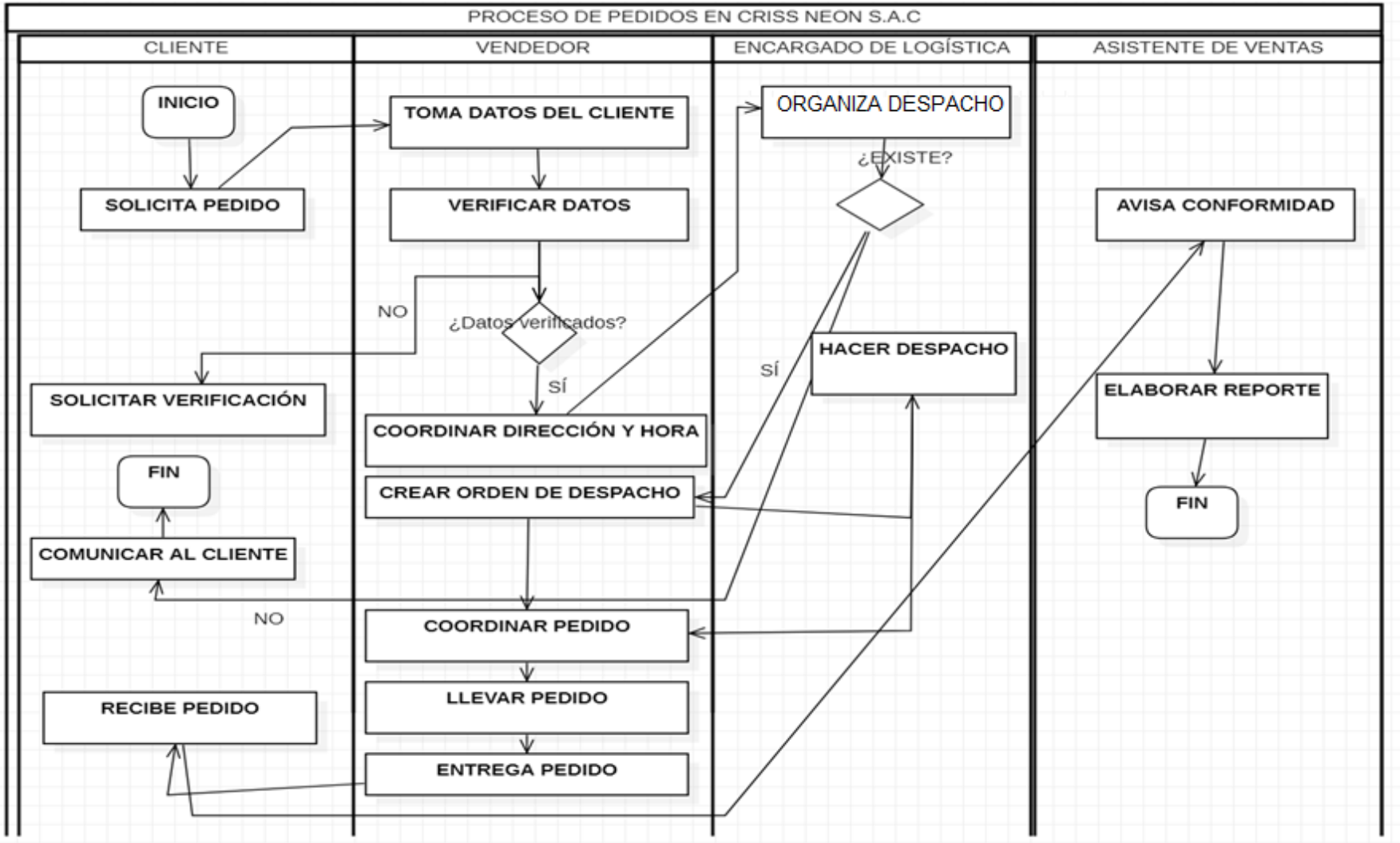
Criss NEON
CARLOS MEJIA ZAPATA
GERENTE GENERAL

Nombre: **Carlos Francisco Mejia Zapata**
Cargo: **(Gerente General)**

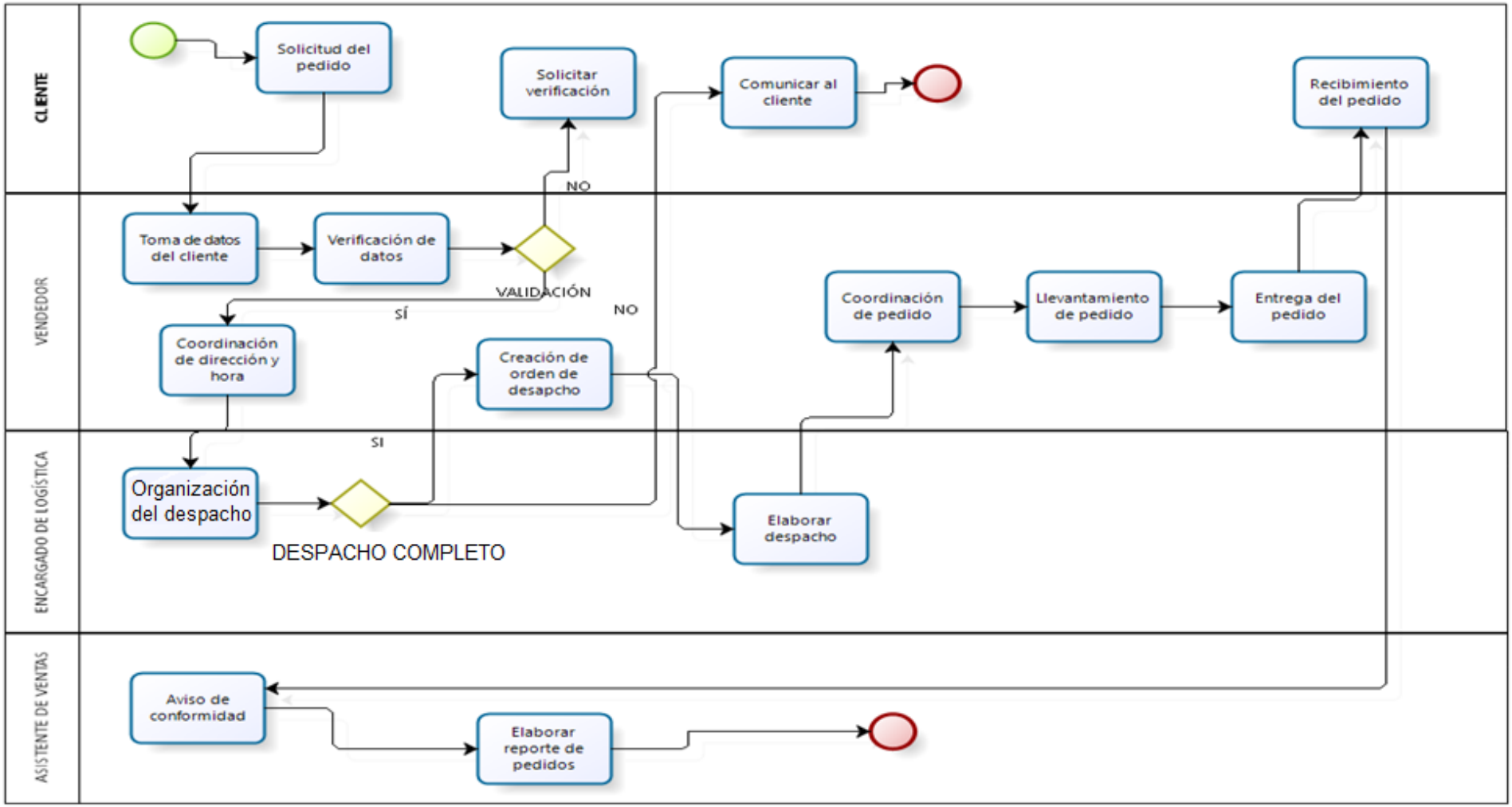
ANEXO N°03: Diagrama de Ishikawa de proceso del control de pedidos



ANEXO N°04: Diagrama de flujo de funciones cruzadas del proceso de control de pedidos



ANEXO N°05: Diagrama de flujo de proceso de control de pedidos



ANEXO N°06: Matriz de operacionalización de variables

Variable	Definición conceptual	Definición Operacional	Dimensiones	Indicadores	Instrumento	Escala
Variable Independiente Aplicación Móvil	Serna (2016), "una aplicación móvil es aquel pequeño paquete de software que sirve para resolver una o varias tareas en específico. Estos son similares procesadores de texto, programas de diseño, hojas de cálculo y edición de video de los ordenadores de escritorio con complejidad menor y optimización para un contexto móvil" (p.20).	La aplicación móvil con geolocalización optimiza el proceso de control de pedidos en la solicitud de un pedido de producto de un cliente basándose en la geolocalización.				
Variable Dependiente Proceso de control de pedidos	Iglesias (2016) define que “el control de pedidos consiste en la distribución de la colección de artículos en la zona de picking que se realiza teniendo en cuenta la solicitud del cliente, características del artículo, ubicación del usuario y del sistema de preparación de pedidos” (p.95).	El uso de la aplicación móvil con geolocalización para el proceso de control de pedidos logra el incremento del porcentaje de entregados completos y el decremento del porcentaje de Entregas Perfectamente Recibidas	D1: Entrada del pedido	Incremento del porcentaje de Entregados Completos	Ficha de Registro	Razón
			D2: Estado del pedido	Decremento del porcentaje de Entregas Perfectamente Recibidas		

ANEXO N°07: Tabla de indicadores del proceso de control de pedidos

Indicador	Descripción	Técnica	Instrumento	Unidad de medida	Fórmula
I1: Incremento del porcentaje de Entregados Completos	Mora (2008), "consiste en conocer un nivel de efectividad de los despachos de mercancías hacia los clientes en cuanto a los pedidos enviados a un periodo determinado" (p.32).	Fichaje	Ficha de registro	1 Pedido	$EC = \frac{NPEC}{\square\square} * 100$ **Donde se comprende:** **EC =** Entregados Completos **PP =** No. Pedidos entregados completos **TP =** Total pedidos
I2: Decremento del porcentaje de Entregas Perfectamente Recibidas	Castellano (2015) citando a Mora (2008), explica que las entregas perfectamente recibidas muestran el número y porcentaje que no han cumplido con las expectativas que se hayan establecido en cuanto al servicio y a la calidad.	Fichaje	Ficha de registro	1 Pedido	$EPR = \frac{PR}{TOCR} * 100$ **Donde se comprende:** **EPR =** Entregas Perfectamente Recibidas **PR =** Pedidos Rechazados **TOCR =** Total de órdenes de compra recibidas.

ANEXO N°08: Ficha técnica. Instrumento de recolección de datos

Autor	Pachas Sifuentes, Ricardo David
Nombre del instrumento	Ficha de registro
Lugar	CRISS NEÓN S.A.C.
Fecha de aplicación	06 de enero del 2020
Objetivo	Determinar la influencia de una aplicación móvil con geolocalización para mejorar el proceso del control de pedidos en CRISS NEÓN S.A.C.
Tiempo de duración	28 días (de lunes a viernes)

Elección de técnica e instrumento

Variable	Técnica	Instrumento
Variable dependiente Proceso de control de pedidos	Fichaje	Ficha de registro
Variable independiente Aplicación móvil	--------	----------

Fuente: Elaboración Propia

ANEXO N°09: Instrumento de Investigación (Pre-Test)

Indicador: Incremento del porcentaje de Entregados Completos

FICHA DE REGISTRO			
INVESTIGADOR	Pachas Sifuentes Ricardo David	**TIPO DE PRUEBA**	Pre - Test
INSTITUCIÓN INVESTIGADA	CRISS NEÓN S.A.C.		
DIRECCIÓN	Jirón Huascarán 756. La Victoria		
FECHA DE INICIO	06/01/2020	**FECHA DE FIN**	12/02/2020

VARIABLE	DIMENSIÓN	INDICADOR	MEDIDA	FÓRMULA
Proceso de control de pedidos	Entrada del pedido	Incremento del porcentaje de Entregados Completos	Pedidos	$EC = \frac{NPEC}{TP} * 100$

ITEM	FECHA	NO. PEDIDOS ENTREGADOS COMPLETOS (NPEC)	TOTAL PEDIDOS (TP)	ENTREGADOS COMPLETOS (EC)
1	06/01/2020	4	6	67%
2	07/01/2020	3	8	38%
3	08/01/2020	5	8	63%
4	09/01/2020	5	7	71%
5	10/01/2020	5	10	50%
6	13/01/2020	6	7	86%
7	14/01/2020	6	7	86%
8	15/01/2020	8	9	89%
9	16/01/2020	7	7	100%
10	17/01/2020	3	7	43%
11	20/01/2020	4	9	44%
12	21/01/2020	2	8	25%
13	22/01/2020	2	6	33%
14	23/01/2020	5	9	56%
15	24/01/2020	4	9	44%
16	27/01/2020	2	8	25%
17	28/01/2020	2	6	33%
18	29/01/2020	5	9	56%
19	30/01/2020	2	9	22%
20	31/01/2020	5	8	63%
21	03/02/2020	3	6	50%
22	04/02/2020	7	9	78%
23	05/02/2020	8	8	100%
24	06/02/2020	1	8	13%
25	07/02/2020	2	8	25%
26	10/02/2020	1	4	25%
27	11/02/2020	2	9	22%
28	12/02/2020	2	9	22%

CARLOS MEJIA ZAPATA
GERENTE GENERAL

Indicador: Decremento del porcentaje de Entregas Perfectamente Recibidas

FICHA DE REGISTRO			
INVESTIGADOR	Pachas Sifuentes Ricardo David	**TIPO DE PRUEBA**	Pre - Test
INSTITUCIÓN INVESTIGADA	CRISS NEÓN S.A.C.		
DIRECCIÓN	Jirón Huascarán 756. La Victoria		
FECHA DE INICIO	02/11/2019	**FECHA DE FIN**	11/12/2019

VARIABLE	DIMENSIÓN	INDICADOR	MEDIDA	FÓRMULA
Proceso de control de pedidos	Estado del pedido	Decremento del porcentaje de Entregas Perfectamente Recibidas	Pedidos	$EPR = \frac{PR}{TOCR} * 100$

ITEM	FECHA	PEDIDOS RECHAZADOS (PR)	TOTAL DE ÓRDENES DE COMPRA RECIBIDAS (TOCR)	ENTREGAS PERFECTAMENTE RECIBIDAS (EPR)
1	06/01/2020	2	6	33%
2	07/01/2020	5	8	63%
3	08/01/2020	3	8	38%
4	09/01/2020	2	7	29%
5	10/01/2020	5	10	50%
6	13/01/2020	1	7	14%
7	14/01/2020	1	7	14%
8	15/01/2020	1	9	11%
9	16/01/2020	0	7	0%
10	17/01/2020	4	7	57%
11	20/01/2020	5	9	56%
12	21/01/2020	6	8	75%
13	22/01/2020	4	6	67%
14	23/01/2020	4	9	44%
15	24/01/2020	5	9	56%
16	27/01/2020	6	8	75%
17	28/01/2020	4	6	67%
18	29/01/2020	4	9	44%
19	30/01/2020	7	9	78%
20	31/01/2020	3	8	38%
21	03/02/2020	3	6	50%
22	04/02/2020	2	9	22%
23	05/02/2020	0	8	0%
24	06/02/2020	7	8	88%
25	07/02/2020	6	8	75%
26	10/02/2020	3	4	75%
27	11/02/2020	7	9	78%
28	12/02/2020	7	9	78%

Criss NEON
CARLOS MEJIA ZAPATA
GERENTE GENERAL

ANEXO N°09: Instrumento de Investigación (Post-Test)

Indicador: Incremento del porcentaje de Entregados Completos

FICHA DE REGISTRO			
INVESTIGADOR	Pachas Sifuentes Ricardo David	**TIPO DE PRUEBA**	Post - Test
INSTITUCIÓN INVESTIGADA	CRISS NEÓN S.A.C.		
DIRECCIÓN	Jirón Huascarán 756. La Victoria		
FECHA DE INICIO	07/09/2020	**FECHA DE FIN**	30/10/2020

VARIABLE	DIMENSIÓN	INDICADOR	MEDIDA	FÓRMULA
Proceso de control de pedidos	Entrada del pedido	Incremento del porcentaje de Entregados Completos	Pedidos	$EC = \frac{NPEC}{TP} * 100$

ITEM	FECHA	NO. PEDIDOS ENTREGADOS COMPLETOS (NPEC)	TOTAL PEDIDOS (TP)	ENTREGADOS COMPLETOS (EC)
1	07/09/2020	5	6	83,33%
2	08/09/2020	8	8	100%
3	09/09/2020	8	8	100%
4	10/09/2020	6	7	85,71%
5	17/09/2020	8	10	80%
6	23/09/2020	7	7	100%
7	24/09/2020	7	7	100%
8	25/09/2020	9	9	100%
9	27/09/2020	7	7	100%
10	28/09/2020	6	7	85,71%
11	29/09/2020	8	9	88,89%
12	30/09/2020	8	8	100%
13	01/10/2020	5	6	83,33%
14	05/10/2020	8	9	88,89%
15	07/10/2020	7	9	77,78%
16	09/10/2020	6	8	75%
17	10/10/2020	5	6	83,33%
18	11/10/2020	6	9	66,67%
19	12/10/2020	7	9	77,78%
20	13/10/2020	6	8	75%
21	14/10/2020	4	6	66,67%
22	15/10/2020	7	9	77,78%
23	16/10/2020	6	8	75%
24	17/10/2020	6	8	75%
25	23/10/2020	5	8	62,5%
26	28/10/2020	2	4	50%
27	29/10/2020	8	9	88,89%
28	30/10/2020	8	9	88,89%

CARLOS MEJIA ZAPATA
GERENTE GENERAL

Indicador: Decremento del porcentaje de Entregas Perfectamente Recibidas

FICHA DE REGISTRO			
INVESTIGADOR	Pachas Sifuentes Ricardo David	**TIPO DE PRUEBA**	Post - Test
INSTITUCIÓN INVESTIGADA	CRISS NEÓN S.A.C.		
DIRECCIÓN	Jirón Huascarán 756. La Victoria		
FECHA DE INICIO	07/09/2020	**FECHA DE FIN**	30/10/2020

VARIABLE	DIMENSIÓN	INDICADOR	MEDIDA	FÓRMULA
Proceso de control de pedidos	**Estado del pedido**	**Decremento del porcentaje de Entregas Perfectamente Recibidas**	**Pedidos**	$EPR = \frac{PR}{TOCR} * 100$

ITEM	FECHA	PEDIDOS RECHAZADOS (PR)	TOTAL DE ÓRDENES DE COMPRA RECIBIDAS (TOCR)	ENTREGAS PERFECTAMENTE RECIBIDAS (EPR)
1	07/09/2020	1	6	16,67%
2	08/09/2020	0	8	0%
3	09/09/2020	0	8	0%
4	10/09/2020	1	7	14,29%
5	17/09/2020	2	10	20%
6	23/09/2020	0	7	0%
7	24/09/2020	0	7	0%
8	25/09/2020	0	9	0%
9	27/09/2020	0	7	0%
10	28/09/2020	1	7	14,29%
11	29/09/2020	1	9	11,11%
12	30/09/2020	1	8	12,5%
13	01/10/2020	1	6	16,67%
14	05/10/2020	1	9	11,11%
15	07/10/2020	2	9	22,22%
16	09/10/2020	2	8	25%
17	10/10/2020	1	6	16,67%
18	11/10/2020	3	9	33,33%
19	12/10/2020	2	9	22,22%
20	13/10/2020	2	8	25%
21	14/10/2020	2	6	33,33%
22	15/10/2020	2	9	22,22%
23	16/10/2020	2	8	25%
24	17/10/2020	2	8	25%
25	23/10/2020	3	8	37,5%
26	28/10/2020	2	4	50%
27	29/10/2020	1	9	11,11%
28	30/10/2020	1	9	11,11%

CARLOS MEJIA ZAPATA
GERENTE GENERAL

ANEXO N°11: Base de datos experimental

Orden	Incremento del porcentaje Entregados Completos		Decremento del porcentaje de Entregas Perfectamente Recibidas	
	PreTest	PostTest	PreTest	PostTest
1	67%	83,33%	33%	16,67%
2	38%	100%	63%	0%
3	63%	100%	38%	0%
4	71%	85,71%	29%	14,29%
5	50%	80%	50%	20%
6	86%	100%	14%	0%
7	86%	100%	14%	0%
8	89%	100%	11%	0%
9	100%	100%	0%	0%
10	43%	85,71%	57%	14,29%
11	44%	88,89%	56%	11,11%
12	25%	100%	75%	12,5%
13	33%	83,33%	67%	16,67%
14	56%	88,89%	44%	11,11%
15	44%	77,78%	56%	22,22%
16	25%	75%	75%	25%
17	33%	83,33%	67%	16,67%
18	56%	66,67%	44%	33,33%
19	22%	77,78%	78%	22,22%
20	63%	75%	38%	25%
21	50%	66,67%	50%	33,33%
22	78%	77,78%	22%	22,22%
23	100%	75%	0%	25%
24	13%	75%	88%	25%
25	25%	62,5%	75%	37,5%
26	25%	50%	75%	50%
27	22%	88,89%	78%	11,11%
28	22%	88,89%	78%	11,11%

ANEXO N°12: Resultado de la confiabilidad del instrumento

Indicador: Incremento del porcentaje de Entregados Completos

FICHA DE REGISTRO			
INVESTIGADOR	Pachas Sifuentes Ricardo David	**TIPO DE PRUEBA**	Test
INSTITUCIÓN INVESTIGADA	CRISS NEÓN S.A.C.		
DIRECCIÓN	Jirón Huascarán 756. La Victoria		
FECHA DE INICIO	06/01/2020	**FECHA DE FIN**	23/01/2020

VARIABLE	DIMENSIÓN	INDICADOR	MEDIDA	FÓRMULA
Proceso de control de pedidos	**Entrada del pedido**	**Incremento del porcentaje de Entregados Completos**	**Pedidos**	$EC = \frac{NPEC}{TP} * 100$

ITEM	FECHA	NO. PEDIDOS ENTREGADOS COMPLETOS (NPEC)	TOTAL PEDIDOS (TP)	ENTREGADOS COMPLETOS (EC)
1	**06/01/2020**	4	6	67%
2	**07/01/2020**	3	8	38%
3	**08/01/2020**	5	8	63%
4	**09/01/2020**	5	7	71%
5	**10/01/2020**	5	10	50%
6	**13/01/2020**	6	7	86%
7	**14/01/2020**	6	7	86%
8	**15/01/2020**	8	9	89%
9	**16/01/2020**	7	7	100%
10	**17/01/2020**	3	7	43%
11	**20/01/2020**	4	9	44%
12	**21/01/2020**	2	8	25%
13	**22/01/2020**	2	6	33%
14	**23/01/2020**	5	9	56%

Criss NEON
CARLOS MEJIA ZAPATA
GERENTE GENERAL

FICHA DE REGISTRO			
INVESTIGADOR	Pachas Sifuentes Ricardo David	**TIPO DE PRUEBA**	Re-Test
INSTITUCIÓN INVESTIGADA	CRISS NEÓN S.A.C.		
DIRECCIÓN	Jirón Huascarán 756. La Victoria		
FECHA DE INICIO	24/01/2020	**FECHA DE FIN**	12/02/2020

VARIABLE	DIMENSIÓN	INDICADOR	MEDIDA	FÓRMULA
Proceso de control de pedidos	**Entrada del pedido**	**Incremento del porcentaje de Entregados Completos**	**Pedidos**	$EC = \frac{NPEC}{TP} * 100$

ITEM	FECHA	NO. PEDIDOS ENTREGADOS COMPLETOS (NPEC)	TOTAL PEDIDOS (TP)	ENTREGADOS COMPLETOS (EC)
1	**24/01/2020**	4	9	44%
2	**27/01/2020**	2	8	25%
3	**28/01/2020**	2	6	33%
4	**29/01/2020**	5	9	56%
5	**30/01/2020**	2	9	22%
6	**31/01/2020**	5	8	63%
7	**03/02/2020**	3	6	50%
8	**04/02/2020**	7	9	78%
9	**05/02/2020**	8	8	100%
10	**06/02/2020**	1	8	13%
11	**07/02/2020**	2	8	25%
12	**10/02/2020**	1	4	25%
13	**11/02/2020**	2	9	22%
14	**12/02/2020**	2	9	22%

Criss NEON
CARLOS MEJIA ZAPATA
GERENTE GENERAL

Indicador: Decremento del porcentaje de Entregas Perfectamente Recibidas

FICHA DE REGISTRO			
INVESTIGADOR	Pachas Sifuentes Ricardo David	**TIPO DE PRUEBA**	Test
INSTITUCIÓN INVESTIGADA	CRISS NEÓN S.A.C.		
DIRECCIÓN	Jirón Huascarán 756. La Victoria		
FECHA DE INICIO	06/01/2020	**FECHA DE FIN**	23/01/2020

VARIABLE	DIMENSIÓN	INDICADOR	MEDIDA	FÓRMULA
Proceso de control de pedidos	**Estado del pedido**	**Decremento del porcentaje de Entregas Perfectamente Recibidas**	**Pedidos**	$EPR = \frac{PR}{TOCR} * 100$

ITEM	FECHA	PEDIDOS RECHAZADOS (PR)	TOTAL DE ÓRDENES DE COMPRA RECIBIDAS (TOCR)	ENTREGAS PERFECTAMENTE RECIBIDAS (EPR)
1	06/01/2020	2	6	33%
2	07/01/2020	5	8	63%
3	08/01/2020	3	8	38%
4	09/01/2020	2	7	29%
5	10/01/2020	5	10	50%
6	13/01/2020	1	7	14%
7	14/01/2020	1	7	14%
8	15/01/2020	1	9	11%
9	16/01/2020	0	7	0%
10	17/01/2020	4	7	57%
11	20/01/2020	5	9	56%
12	21/01/2020	6	8	75%
13	22/01/2020	4	6	67%
14	23/01/2020	4	9	44%

Criss NEON
CARLOS MEJIA ZAPATA
GERENTE GENERAL

FICHA DE REGISTRO			
INVESTIGADOR	Pachas Sifuentes Ricardo David	**TIPO DE PRUEBA**	Re-Test
INSTITUCIÓN INVESTIGADA	CRISS NEÓN S.A.C.		
DIRECCIÓN	Jirón Huascarán 756. La Victoria		
FECHA DE INICIO	24/01/2020	**FECHA DE FIN**	12/02/2020

VARIABLE	DIMENSIÓN	INDICADOR	MEDIDA	FÓRMULA
Proceso de control de pedidos	**Estado del pedido**	**Decremento del porcentaje de Entregas Perfectamente Recibidas**	**Pedidos**	$EPR = \frac{PR}{TOCR} * 100$

ITEM	FECHA	PEDIDOS RECHAZADOS (PR)	TOTAL DE ÓRDENES DE COMPRA RECIBIDAS (TOCR)	ENTREGAS PERFECTAMENTE RECIBIDAS (EPR)
1	24/01/2020	7	9	77,78%
2	27/01/2020	6	8	75%
3	28/01/2020	5	6	83,33%
4	29/01/2020	6	9	66,67%
5	30/01/2020	7	9	77,78%
6	31/01/2020	6	8	75%
7	03/02/2020	4	6	66,67%
8	04/02/2020	7	9	77,78%
9	05/02/2020	6	8	75%
10	06/02/2020	6	8	75%
11	07/02/2020	5	8	62,5%
12	10/02/2020	2	4	50%
13	11/02/2020	8	9	88,89%
14	12/02/2020	8	9	88,89%

Criss NEON
CARLOS MEJIA ZAPATA
GERENTE GENERAL

		EP_Test	EP_Retest
EC_Test	Correlación de Pearson	1	,888**
	Sig. (bilateral)		,000
	N	14	14
EC_Retest	Correlación de Pearson	,888**	1
	Sig. (bilateral)	,000	
	N	14	14
**. La correlación es significativa en el nivel 0,01 (bilateral).			

La correlación de Pearson indica un grado elevado de confiabilidad de 0,88, por lo cual se demuestra que el instrumento del indicador del incremento del porcentaje de Entregados Completos es confiable.

		EPR_Test	EPR_Retest
EPR_Test	Correlación de Pearson	1	,887**
	Sig. (bilateral)		,000
	N	14	14
EPR_Retest	Correlación de Pearson	,887**	1
	Sig. (bilateral)	,000	
	N	14	14
**. La correlación es significativa en el nivel 0,01 (bilateral).			

La correlación de Pearson indica un grado elevado de confiabilidad de 0,87, por lo cual se demuestra que el instrumento del indicador del decremento del porcentaje de Entregas Perfectamente Recibidas es aceptable.

ANEXO N°13: Validación de la metodología de desarrollo de software

TABLA DE EVALUACIÓN DE EXPERTOS
(Metodología de desarrollo de Software)

Datos del experto:

1. Apellidos y Nombres: Pérez Farfán Iván Martin
2. Cargo que sustenta: Encargado de Investigación
3. Título y/o Grado: Mgtr. Ingeniería de Sistemas
4. Universidad que labora: Universidad César Vallejo Lima Norte.
5. Autor: Pachas Sifuentes, Ricardo David.
6. Fecha: 19/06/2020

TESIS:

APLICACIÓN MÓVIL CON GEOLOCALIZACIÓN PARA EL PROCESO DE PEDIDOS EN CRISS NEÓN S.A.C.

Mediante la tabla de evaluación de expertos, usted tiene la facultad de calificar las metodologías involucradas, mediante una serie de preguntas llenando con puntuación especificada al final de la tabla. Asimismo, le exhortamos en la corrección de los ítems indicando sus observaciones y/o sugerencias, con la finalidad de mejorar la coherencia de las preguntas sobre la metodología.

		METODOLOGIAS			
ITEM	**PREGUNTAS**	**RUP**	**SCRUM**	**MOBILE-D**	**OBSERVACIONES**
1	Metodología de rápida y ágil implementación.	2	3	3	
2	La metodología nos ayuda a construir un software de calidad.	3	3	3	
3	Es una metodología flexible y preparada a los cambios durante un pro	2	3	3	
4	Metodología de rápido desarrollo de software.	2	3	3	
5	Metodología más práctica para los proyectos móviles	2	2	3	
6	El cliente es parte de uno de los integrantes del equipo.	3	3	3	
7	Los requerimientos están priorizados.	3	3	3	
8	Mayor énfasis en la construcción de una aplicación móvil.	2	2	3	
9	Desarrollo muy rápido en equipos muy pequeños.	2	3	3	
10	Tamaño del proyecto (bajo costo, facilidad de comunicación con el usuario)	2	3	3	
TOTAL		23	28	30	

Evaluar con la siguiente puntuación:

1: Bajo 2: Regular 3: Bueno

Firma del experto

UCV
UNIVERSIDAD CÉSAR VALLEJO

TABLA DE EVALUACIÒN DE EXPERTOS
(Metodología de desarrollo de Software)

Datos del experto:

1. **Apellidos y Nombres: VASQUEZ VALENCIA YESENIA**
2. **Cargo que sustenta: DTC**
3. **Título y/o Grado: DOCTORA**
4. **Universidad que labora: Universidad César Vallejo Lima Norte.**
5. **Autor: Pachas Sifuentes, Ricardo David.**
6. **Fecha: 22/06/2020**

TESIS:

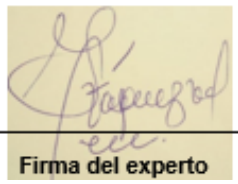
APLICACIÒN MÒVIL CON GEOLOCALIZACIÒN PARA EL PROCESO DE PEDIDOS EN CRISS NEÓN S.A.C.

Mediante la tabla de evaluación de expertos, usted tiene la facultad de calificar las metodologías involucradas, mediante una serie de preguntas llenando con puntuación especificada al final de la tabla. Asimismo, le exhortamos en la corrección de los ítems indicando sus observaciones y/o sugerencias, con la finalidad de mejorar la coherencia de las preguntas sobre la metodología.

		METODOLOGIAS			
ITEM	PREGUNTAS	RUP	SCRUM	MOBILE-D	OBSERVACIONES
1	Metodología de rápida y ágil implementación.	2	2	3	
2	La metodología nos ayuda a construir un software de calidad.	2	2	3	
3	Es una metodología flexible y preparada a los cambios durante un pro	2	2	3	
4	Metodología de rápido desarrollo de software.	2	3	2	
5	Metodología más práctica para los proyectos móviles	2	3	2	
6	El cliente es parte de uno de los integrantes del equipo.	2	3	2	
7	Los requerimientos están priorizados.	2	2	3	
8	Mayor énfasis en la construcción de una aplicación móvil.	2	2	3	
9	Desarrollo muy rápido en equipos muy pequeños.	2	3	3	
10	Tamaño del proyecto (bajo costo, facilidad de comunicación con el usuario)	2	3	3	
TOTAL		22	25	27	

Evaluar con la siguiente puntuación:

1: Bajo 2: Regular 3: Bueno



Firma del experto

UCV
UNIVERSIDAD CÉSAR VALLEJO

TABLA DE EVALUACIÓN DE EXPERTOS
(Metodología de desarrollo de Software)

Datos del experto:

1. **Apellidos y Nombres: More Valencia Rubén Alexander**
2. **Cargo que sustenta: Docente Universidad César Vallejo**
3. **Título y/o Grado: Ingeniero Informático - Mg. Administración de la Educación**
4. **Universidad que labora: Universidad César Vallejo Piura.**
5. **Autor: Pachas Sifuentes, Ricardo David.**
6. **Fecha: 23/06/2020**

TESIS:

APLICACIÓN MÓVIL CON GEOLOCALIZACIÓN PARA EL PROCESO DE PEDIDOS EN CRISS NEÓN S.A.C.

Mediante la tabla de evaluación de expertos, usted tiene la facultad de calificar las metodologías involucradas, mediante una serie de preguntas llenando con puntuación especificada al final de la tabla. Asimismo, le exhortamos en la corrección de los ítems indicando sus observaciones y/o sugerencias, con la finalidad de mejorar la coherencia de las preguntas sobre la metodología.

		METODOLOGÍAS			
ITEM	PREGUNTAS	RUP	SCRUM	MOBILE-D	OBSERVACIONES
1	Metodología de rápida y ágil implementación.	2	3	3	
2	La metodología nos ayuda a construir un software de calidad.	3	3	3	
3	Es una metodología flexible y preparada a los cambios durante un pro	2	2	3	
4	Metodología de rápido desarrollo de software.	3	3	3	
5	Metodología mas práctica para los proyectos móviles	2	3	3	
6	El cliente es parte de uno de los integrantes del equipo.	3	3	3	
7	Los requerimientos están priorizados.	3	3	3	
8	Mayor énfasis en la construcción de una aplicación móvil.	2	3	3	
9	Desarrollo muy rápido en equipos muy pequeños.	3	3	3	
10	Tamaño del proyecto (bajo costo, facilidad de comunicación con el usuario)	3	2	3	
TOTAL		26	28	**30**	

Evaluar con la siguiente puntuación:

1: Bajo **2: Regular** **3: Bueno**

ANEXO N°14: Validación del Instrumento del Indicador (Entregados Completos)

Ficha de experto 1

VALIDACIÓN DEL INSTRUMENTO PARA EL INDICADOR: ENTREGADOS COMPLETOS

Datos del experto:

1. **Apellidos y Nombres:** …Acuña Meléndez María……..……
2. **Cargo que sustenta:** Docente……..……
3. **Título y/o Grado:** Magister……..……..
4. **Universidad que labora:** Universidad César Vallejo Lima Norte
5. **Autor: Pachas Sifuentes, Ricardo David.**
6. **Fecha: _23__/_11__/2020**

TESIS:
APLICACIÓN MÓVIL CON GEOLOCALIZACIÓN PARA EL PROCESO DE CONTROL DE PEDIDOS EN CRISS NEÓN S.A.C.

Tabla de Evaluación de Expertos para el Indicador: Entregados Completos

$EC = \frac{NPEC}{TP} * 100$	Donde: EC = Entregados Completos NP = N° de Pedidos entregados completos TP = Total pedidos

ITEMS	CRITERIOS	Deficiente 0 – 20%	Regular 21 – 50%	Bueno 51 – 70%	Muy Bueno 71 – 80%	Excelente 71 – 80%
1. Claridad	Esta formulado con el lenguaje apropiado.				80%	
2. Objetividad	Esta expresado en conducta observable.				80%	
3. Actualidad	Es adecuado al avance de la ciencia.				80%	
4. Organización	Existe una organización lógica.				80%	
5. Suficiencia	Comprende los aspectos de cantidad y calidad.				80%	
6. Intencionalidad	Adecuado para valorar aspectos del sistema metodológico y científico.				80%	
7. Consistencia	Está basado en aspectos teóricos, científicos acordes a la tecnología educativa.				80%	
8. Coherencia	Entre los índices, indicadores y dimensiones.				80%	
9. Metodología	Responde al propósito del trabajo bajo los objetivos a lograr.				80%	
10. Pertinencia	El instrumento es adecuado al tipo de investigación				80%	
PROMEDIO DE VALIDACIÓN					80%	

Promedio de Valoración: _____80%_____

Observaciones: _____El instrumento es aplicable_____

Firma del experto

Ficha de experto 3

ANEXO N°15: Validación del Instrumento del Indicador (Decremento del porcentaje de Entregas Perfectamente Recibidas)

Ficha de experto 1

VALIDACIÓN DEL INSTRUMENTO PARA EL INDICADOR: ENTREGAS PERFECTAMENTE RECIBIDAS

Datos del experto:

1. **Apellidos y Nombres:** Acuña Meléndez María……
2. **Cargo que sustenta:** Docente……
3. **Título y/o Grado:** Magister……..
4. **Universidad que labora:** Universidad César Vallejo Lima Norte.
5. **Autor: Pachas Sifuentes, Ricardo David.**
6. **Fecha: __22_/_11__/_2020__**

TESIS:

APLICACIÓN MÓVIL CON GEOLOCALIZACIÓN PARA EL PROCESO DE CONTROL DE PEDIDOS EN CRISS NEÓN S.A.C.

Tabla de Evaluación de Expertos para el Indicador: Entregas Perfectamente Recibidas

$EPR = \frac{PR}{TOCR}$	Donde: EC = Entregas perfectamente recibidas PEC = Pedidos rechazados TP = Total de órdenes de compra recibidas

ITEMS	CRITERIOS	Deficiente 0 – 20%	Regular 21 – 50%	Bueno 51 – 70%	Muy Bueno 71 – 80%	Excelente 71 – 80%
1. Claridad	Esta formulado con el lenguaje apropiado.				80%	
2. Objetividad	Esta expresado en conducta observable.				80%	
3. Actualidad	Es adecuado al avance de la ciencia.				80%	
4. Organización	Existe una organización lógica.				80%	
5. Suficiencia	Comprende los aspectos de cantidad y calidad.				80%	
6. Intencionalidad	Adecuado para valorar aspectos del sistema metodológico y científico.				80%	
7. Consistencia	Está basado en aspectos teóricos, científicos acordes a la tecnología educativa.				80%	
8. Coherencia	Entre los índices, indicadores y dimensiones.				80%	
9. Metodología	Responde al propósito del trabajo bajo los objetivos a lograr.				80%	
10. Pertinencia	El instrumento es adecuado al tipo de investigación				80%	
PROMEDIO DE VALIDACIÓN					80%	

Promedio de Valoración: ____80%_____

Observaciones: El instrumento es aplicable_____

_____

Firma del experto

Ficha de experto 3

ANEXO N°16: Carta de aprobación de la empresa

CONSTANCIA

El que suscribe, **Carlos Francisco MEJÍA Zapata**, Gerente General de la empresa CRISS NEÓN S.A.C, hace constar por el presente documento, que el estudiante **Ricardo David PACHAS Sifuentes**, identificado con el número de DNI 70579023, viene realizando en esta empresa un trabajo de investigación titulado "Aplicación móvil con geolocalización para el proceso de control de pedidos en CRISS NEÓN S.A.C.", para poder obtener el grado de Ingeniero de Sistemas en la Universidad César Vallejo.

Se expide la presente constancia para los fines que el interesado estime conveniente.

Lima, 19 de noviembre del 2020

Criss NEON
CARLOS MEJIA ZAPATA
GERENTE GENERAL

Nombre: **Carlos Francisco Mejia Zapata**
Cargo: **(Gerente General)**

Huascarán 756, La Victoria – Telf. 999015859 - E-mail: crissneon@hotmail.com

ANEXO N°17: Acta de implementación de la aplicación móvil

ACTA DE IMPLEMENTACIÓN DE LA APLICACIÓN MÓVIL CON GEOLOCALIZACIÓN PARA EL PROCESO DE CONTROL DE PEDIDOS EN CRISS NEÓN S.A.C.

Estimado: Sr. Ricardo David Pachas Sifuentes

Mediante la presente acta de implementación se confirma y ampara que se realizó la implementación de la aplicación móvil a partir del 05 de setiembre del año 2020, cuyo título es "Aplicación móvil con geolocalización para el proceso de control de pedidos en CRISS NEÓN S.A.C.", con el fin de contribuir a la organización de manera eficiente y eficaz, cumpliendo los requerimientos planteados al inicio del proyecto.

Firma en señal de conformidad.

Criss NEON
CARLOS MEJIA ZAPATA
GERENTE GENERAL

Nombre: **Carlos Francisco Mejia Zapata**
Cargo: **(Gerente General)**

Huascarán 756, La Victoria – Telf. 999015859 - E-mail: crissneon@hotmail.com

ANEXO N°18: Bases teóricas

En esta investigación definiremos sobre los tipos de aplicaciones móviles que comprende en primer lugar por las aplicaciones nativas que lo define según Serna (2016), "este tipo de aplicaciones usan lenguajes de programación nativos del sistema operativo y usan todo el potencial de hardware de los terminales a través de la paquetería del desarrollo de sistema, los sistemas operativos promueven su desarrollo cuando se trata de software que requiere un uso importante del hardware y sensores, en android se desarrolla en lenguaje java y se instalan una paquetería en específico que distribuye a través de los mercados de aplicaciones."(p.26); en segundo lugar por las aplicaciones híbridas que lo define según Serna (2016), "combinan diversas tecnologías de los lenguajes del sistema operativo que hacen fusión con elementos web en su interfaz, en este tipo de apps es común usar elementos incrustados que presentan partes del navegador para visualizar la interfaz web, que presentan interfaces construidas en HTML (Lenguaje de Marcas de Hipertexto) y CSS (Hoja de estilos en cascada), etc ; su costo es económico que no va ser necesario la mano de obra que va especializarse en capacitaciones de entornos de lenguajes de la programación o por otras tecnologías webs conocidas." (p.27) y por último las aplicaciones web que lo define según Serna (2016), "presenta como una respuesta que enriquece los tradicionales servicios en línea, su apariencia puede ser similar a una aplicación nativa pero marca diferencia en que usa su totalidad de tecnologías webs así como la adaptación de un modelo adaptative design y responsive permitiendo el uso de una sola plantilla de HTML que se adapte al lenguaje CSS o de algún servidor, esta técnica también es usada para las aplicaciones híbridas." (p.27).

En nuestra investigación también mencionaremos al lenguaje de programación, que se define según Caballero (2015), “es una sintaxis de símbolos y códigos usados para guiar la programación de estructuras bajo variables e identificadores, operadores y expresiones, estructuras de control e instrucciones de entrada/salida.[1]” (p.8), las cuales entre estos lenguajes de

[1] CABALLERO, C. UF1305 - Programación con lenguajes de guión en páginas web [en línea]. Madrid: Ediciones Praninfo S.A., 2015. [Fecha de consulta 16 abril 2020]. ISBN: 9788428396875. Disponible en: https://books.google.com.pe/books?id=N1GACwAAQBAJ&dq=UF1305+-

programación se encuentran Java, que lo define según Garrido (2015), “la tecnología Java es aquel lenguaje de programación como conjunto de plataformas especializadas: J2SE (Java SE), J2EE (Java EE), J2ME (Java ME) y JavaCard, es aquel lenguaje de programación de elevado nivel que va ser orientado a objeto reciente creación y tienen una sintaxis muy similar al lenguaje C o C++, por lo que es sencillo y multiplataforma.[2]” (p.2-3), como segundo lenguaje de programación, se tiene a Kotlin. Según Guimerá (2018), “es un lenguaje de programación que fue creado en 2010 por JetBrains, es una alternativa a Java, que sustituye a varios de los problemas que se encuentra en dicho lenguaje; este lenguaje a diferencia de Java: ahorra código, es seguro contra nulos y fácil de usar.[3]” (p.3) y por último se tiene a flutter que según Biessek (2019), “es un moderno framework que requiere un lenguaje moderno de alto nivel para ser capaz de proporcionar la mejor experiencia al desarrollado móvil, esta tecnología se basa en lenguaje dart para poder realizar estas aplicaciones.[4]” (p.7).

+Programaci%C3%B3n+con+lenguajes+de+gui%C3%B3n+en+p%C3%A1ginas+web&hl=es&source=gbs_navlinks_s

[2] GARRIDO, P. Comenzando a programar con JAVA [en línea]. Alicante: Universidad Miguel Hernández, 2015. [Fecha de consulta 16 abril 2020]. ISBN: 9788416024247. Disponible en: https://books.google.com.pe/books?id=4v8QCgAAQBAJ&dq=Comenzando+a+programar+con+JAVA&hl=es&source=gbs_navlinks_s

[3] GUIMERÁ, A. Iniciación a Android en Kotlin. Casos prácticos [en línea]. Madrid: Ediciones Paraninfo, S.A., 2018. [Fecha de consulta 16 abril 2020]. ISBN: 9788428340922. Disponible en: https://books.google.com.pe/books?id=GVJ1DwAAQBAJ&dq=Iniciaci%C3%B3n+a+Android+en+Kotlin.+Casos+pr%C3%A1cticos&hl=es&source=gbs_navlinks_s

[4] BIESSEK. A. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2 [en línea]. Reino Unido: Packt Publishing Ltd, 2019. [Fecha de consulta 21 abril 2020]. ISBN: 9781788990523. Disponible en: https://books.google.com.pe/books?id=pF6vDwAAQBAJ&dq=Flutter+for+Beginners:+An+introductory+guide+to+building+cross-platform+mobile+applications+with+Flutter+and+Dart+2&hl=es&source=gbs_navlinks_s

	Java	Kotlin	Flutter (Dart)
Ventajas	Según Garrido (2015): - Interactivo. - Independiente de arquitectura de harware. - Interpetrado y rápido. - Fácil de aprender - Muy Comercial. - Gratuito	Según Guimerá (2018): - Codificación corta. - Interoperabilidad - Mismo trabajo que el lenguaje Java. - Estilo orientado a objetos con enfoque funcional.	Según Biessek (2019): - Multiplataforma - Ahorra hardware en dispositivos móviles.
Desventajas	Según Garrido (2015): - Librerías illimitadas.	Según Guimerá (2018): - Legibilidad inicial del código. - Difícil de entender la sintaxis.	Según Biessek (2019): - Poco conocimiento de uso. - Difícil de entender la sintaxis. - Librerías limitadas.

Por la cual, según la tabla anterior, se va a determinar las ventajas y desventajas de cada lenguaje de programación para el IDE de Android Studio, por eso se determinó que se realizará con en lenguaje Java para la realización de la aplicación móvil con geolocalización, ya que es fácil de entender la sintaxis del código y es el lenguaje favorito de los desarrolladores.

En nuestra investigación definiremos el término base de datos que Según Capacho y Nieto (2017), “una base de datos significa aquel conjunto de

elementos interrelacionados, un conjunto de serie de programas que se permite a varios usuarios poder obtener acceso a estos archivos para bien sean actualizados o consultados.[5]" (p.13) y consta de dos tipos de base de datos que uno de ellos es la relacional que lo define Capacho y Nieto (2017), manifiestan que "es un modelo representativo de modelo relacional, lo cual las relaciones y datos entre datos que se representan por medio de una serie de tablas cada una de las cuales tienen varias columnas con nombres únicos." (p.18) y la no relacional que lo define Capacho y Nieto (2017), manifiesta que "a diferencia de las bases de datos relacionales, estos van a poseer cierto identificador de la cual los relacione entre un cierto conjunto de datos y otros, la información se va a organizar normalmente mediante una serie de documentos y no existe un esquema que será exacto de almacenamiento." (p.18).

En nuestra investigación definiremos el término base de datos que Según Capacho y Nieto (2017), "una base de datos significa aquel conjunto de elementos interrelacionados, un conjunto de serie de programas que se permite a varios usuarios poder obtener acceso a estos archivos para bien sean actualizados o consultados.[6]" (p.13) y consta de dos tipos de base de datos que uno de ellos es la relacional que lo define Capacho y Nieto (2017), manifiestan que "es un modelo representativo de modelo relacional, lo cual las relaciones y datos entre datos que se representan por medio de una serie de tablas cada una de las cuales tienen varias columnas con nombres únicos." (p.18) y la no relacional que lo define Capacho y Nieto (2017), manifiesta que "a diferencia de las bases de datos relacionales, estos van a poseer cierto identificador de la cual los relacione entre un cierto conjunto de datos y otros, la información se va a organizar normalmente mediante una serie de documentos y no existe un esquema que será exacto de almacenamiento." (p.18).

[5] CAPACHO, J., NIETO, W. Diseño de base de datos [en línea]. España: Universidad del Norte, 2017. [Fecha de consulta 21 abril 2020]. ISBN: 9789587418255. Disponible en: https://books.google.com.pe/books?id=TLBJDwAAQBAJ&dq=Dise%C3%B1o+de+base+de+datos&hl=es&source=gbs_navlinks_s

[6] CAPACHO, J., NIETO, W. Diseño de base de datos [en línea]. España: Universidad del Norte, 2017. [Fecha de consulta 21 abril 2020]. ISBN: 9789587418255. Disponible en: https://books.google.com.pe/books?id=TLBJDwAAQBAJ&dq=Dise%C3%B1o+de+base+de+datos&hl=es&source=gbs_navlinks_s

En esta investigación mencionaremos sobre los gestores de base de datos que lo define según Peña (2017), “es un conjunto de aplicaciones informáticas que van a permitir manejar diversas bases de datos, las cuales estos programas van a servir para que usuarios y base de datos se comuniquen de forma eficiente, estos sistemas comprenden programas para la gestión de los datos que incluyen propios datos almacenados que están relacionados.[7]” (p.3). Entre estos gestores se encuentra Firebase que según Moroney (2017), “el servicio de firebase es aquel gestor de base de datos no relacional que incluía el realtime database que a la vez es aquella base de datos no relacional alojada en cloud, las cuales, su data es procesada y almacenados en formato JSON (Notación de Objetos de JavaScript) y hace posible la sincronización en tiempo real con cada usuario que se conecta. Cuando se compila aplicaciones multiplataforma con Android, SDK y Java Script, todos los usuarios comparten instancias de Realtime Database y reciben actualizaciones automáticamente con datos más actuales.[8]” (p.2).

En la anterior figura, se puede observar que firebase tiene herramientas para desarrolladores como es cierta base de datos en tiempo real, autenticación, almacenamiento en la nube, que se puede guardar imágenes, hacer tests de prueba y para comercializar, la parte de analytics, lo cual mide su potencial en el mercado informático, en segundo gestor de BD se tiene a SQLite que lo define según Joyce y Joseph (2017), “SQLite es aquel gestor de base de datos relacional, la cual va a permitir que cada aplicación en el dispositivo móvil tenga su propia base de datos SQL. Al igual que el almacenamiento nativo, es sólo adaptable a aplicaciones móviles y no para aplicaciones HTML5. Esta base de

[7] PEÑA, S. UF1469 - SGBD e instalación [en línea]. Madrid: Ediciones Paraninfo, S.A., 2017. [Fecha de consulta 21 abril 2020]. ISBN: 9788428396561. Disponible en: https://books.google.com.pe/books?id=yVPVDQAAQBAJ&dq=UF1469+-+SGBD+e+instalaci%C3%B3n&hl=es&source=gbs_navlinks_s

[8] MORONEY, L. The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform [en línea]. United States: Apress, 2017. [Fecha de consulta 21 abril 2020]. ISBN: 9781484229439. Disponible en: https://books.google.com.pe/books?id=ox0-DwAAQBAJ&dq=firebase&hl=es&source=gbs_navlinks_s

datos proporciona una forma de almacenar data de forma persistente con datos ilimitados administrables utilizando la sintaxis SQL.[9]" (p.66) y por último MySQL quien lo define según Combaudon (2018), "MySQL es aquel sistema de gestión de base de datos relacional que comprende el open source más popular a nivel internacional, conocido por el rendimiento y fiabilidad que otorga, este gestor ayudará permitiendo optimizar el código y obtener aplicaciones rápidas y seguras, proviene de la familia de Oracle y proporciona un servicio en la nube.[10] (p.17).

A la vez en esta investigación se define la metodología de desarrollo de software para el proyecto pero primero definiremos este término que según Pantaleo y Rinaudo (2015), "este tipo de metodología enfatiza a un framework que es un entorno o marco de trabajo que sirve para planear, estructurar y controlar el proceso de desarrollo de softwares y aplicaciones.[11]" (p.6), entre estas, se tiene a la primera que es la metodología Mobile-D que lo define según Rocha y Guarda (2018), "es una metodología con el enfoque ágil para desarrollar aplicaciones móviles".[12] (p.814), las cuales se dividen en cinco fases que comprende, la primera por la fase de Exploración que se comprende por los programadores que precisan características y plan de un mismo proyecto, que se va a descomponer ciertas actividades como el establecimiento de conjunto de actores, definición de alcance un proyecto y aclarar el proyecto. (p.814), en segunda comprende a la fase de Iniciación, en aquella fase, este equipo de programación va a configurar el proyecto preparándolo e identificar cualquier

[9] JOYCE, J., JOSEPH, J. Learn Ionic 2: Develop Multi-platform Mobile Apps [en línea]. United States: Apress, 2017. [Fecha de consulta 21 abril 2020]. 105pp. ISBN: 9781484226179. Disponible en: https://books.google.com.pe/books?id=WvGqDgAAQBAJ&dq=Learn+Ionic+2:+Develop+Multi-platform+Mobile+Apps&hl=es&source=gbs_navlinks_s

[10] COMBAUDON, S. MySQL 5.7: administración y optimización [en línea]. España: Ediciones ENI, 2018. [Fecha de consulta 23 abril 2020]. ISBN: 9782409008467. Disponible en: https://books.google.com.pe/books/about/MySQL_5_7.html?id=PvKjuAIA-PwC&redir_esc=y

[11] PANTALEO, G., RINAUDO, L. Ingeniería de Software [en línea]. España: Alfaomega Grupo Editor, 2015. [Fecha de consulta 23 abril 2020]. ISBN: 9789871609789. Disponible en: https://books.google.com.pe/books?id=a8j2DQAAQBAJ&dq=Ingenier%C3%ADa+de+Software&hl=es&source=gbs_navlinks_s

[12] ROCHA, A., GUARDA, T. Proceedings of the International Conference on Information Technology & Systems (ICITS 2018) [en línea]. Alemania: Springer, 2018. [Fecha de consulta 23 abril 2020]. ISBN: 9783319734507. Disponible en: https://books.google.com.pe/books?id=gj5FDwAAQBAJ&dq=Proceedings+of+the+International+Conference+on+Information+Technology+%26+Systems+(ICITS+2018)&hl=es&source=gbs_navlinks_s

recurso indispensable. Para esto, va elaborarse los planes que para su siguiente fase y va a tenerse que establecer el equipo técnico, recursos físicos, tecnológicos y comunicaciones que se descompone por cuatro fases que consiste en configurar el proyecto, planificar un día, día de trabajo y liberación. (p.814), por tercera comprende la fase de Producto. En esta fase, se hace repetición de la programación de esos tres días, que es de forma iterativa hasta implementarse todas las funcionalidades que hace de desear. Se tiene que planificar las iteraciones de trabajo y tareas por realizar. (p.815), por la cuarta que comprende la fase de Estabilización, que se define que, en esta fase, se realiza una integración de un sistema en grupo para la aseguración del funcionamiento correcto. En dicha fase es la más esencial en un proyecto, lo cual se va a asegurar la estabilidad de este desarrollo del aplicativo. (p.815) y en la última fase de pruebas, que se define en esta determinada fase si el sistema está con la disponibilidad tanto para la versión estable como funcional, se realizará el testeo de la aplicación para hacer la comprobación de errores que conllevada a corregirlas, pero no se realizarán nuevos avances o implementos finalizando el proceso ya se descompondría el ciclo.(p.815); la segunda metodología definida es RUP (Proceso Unificado de Rational) , la cual, según Suárez, Medina y Hernández (2015), "es una metodología de desarrollo iterativo, la cual facilita el buen enfoque en un desarrollo de software, la cual otorgará relaciones puntualizadas en los elementos como rol, artefactos, tarea y disciplina, puede ser adaptable a propósitos de proyecto según el tipo y tamaño.[13]"(p.86), se compone en cuatro fases, que comprende la primera fase que es de Inicio, la cual se define que en dicha fase se enfoca en un modelado de negocio tanto para la especificación para requisitos, lo cual establecen cierto punto de observación con el fin de hacer una comparación de determinados gastos que ocurren en el transcurso de la implementación pero para pasar a una nueva fase, deberá existir un acuerdo de las estimaciones, prioridades, riesgos, proceso de desarrollo, costo y horario, riesgos identificados. (p.86), en la segunda fase se tiene a la Elaboración que en esta fase se desarrollan casos de uso y lista de

[13] SUÁREZ, Y., MEDINA, D., and HERNÁNDEZ, P. Sistema automatizado para la gestión del mantenimiento de equipos (módulos administración y solicitud de servicio) [en línea]. 2015, diciembre, 24. 85-90 [Fecha de consulta 23 abril 2020]. ISSN: 1010-2760. Disponible en: https://revistas.unah.edu.cu/index.php/rcta/article/view/395

riesgos que deben estar documentado, a la vez como ejecutar cierto plan de desarrollo con el fin de implementar el proyecto (p.86), en la tercera fase se tiene a la Construcción, que se define que en aquella fase se realizarán diversas iteraciones a un ciclo que va incorporar casos de uso en sucesiva de acuerdo a factores de riesgos de un proyecto y por último se tiene a la fase de Transición que nos indica tener una determinada versión inicial beta del software para terminar la fase de producción. (Suárez, Medina y Hernández, 2015, p.86); la última metodología que se definirá es la SCRUM que es un proceso, metodología y a la vez un framework para desarrollar software a modo incremental en entornos donde exista la complejidad y sus requisitos de estos no estén claros o se cambian frecuentemente, es muy usado actualmente, ya que posee características que cuadran con el tipo de profesional en ámbito tecnológico. Su objetivo es proporcionar de un proceso conveniente para grandes y pequeños proyectos de cuales su desarrollo se orientada a objetos, aportara ventajas como la flexibilidad, trabajo en equipo y proporcionar un software que funcione de una manera incremental. Sus principales factores son el backlog, sprints, equipo de desarrollo, reuniones a diario, meets de revisiones y presentaciones de demos. Los integrantes, los cuales se componen como el Scrum Master quien protege al equipo de riesgos e interferencias externas y exceso de optimismo y desmotivación a la vez es quien mantiene las informaciones de las reuniones del Sprint visibles para todos los participantes y en otros el equipo de desarrollo y el producto owner. Se componen a cuatro fases, lo cuales son la Planificación del sprint (presupuesto general, lo que se va a necesitar), Etapa de desarrollo (Garantización de objetivos), Revisión del sprint (analizar y evaluar resultados) y Retroalimentación (feedback de entregado de resultados)[14]. (Laínez, 2015, p.127-128).

[14] LAÍNEZ, J. Desarrollo de Software Ágil: Extremme Programming y Scrum [en línea]. 2da ed. España: IT Campus Academy, 2015 [Fecha de consulta 23 abril 2020]. ISBN: 9781519620149. Disponible en: https://books.google.com.pe/books?id=M4fJCgAAQBAJ&dq=Desarrollo+de+Software+%C3%81gil:+Extreme+Programming+y+Scrum&hl=es&source=gbs_navlinks_s

ANEXO N°19: Desarrollo de la metodología MOBILE-D

FASE I: EXPLORACIÓN

Como primera etapa, se establece los requerimientos y el alcance del proyecto que servirán para un desarrollo adecuado.

- **Establecimiento de los Stakeholders**

Para el desarrollo de esta actividad se definió a las siguientes personas involucradas.

- ✓ **Programador:** Persona encargada del modelamiento de base de datos en Firebase, programación MVC del diseño y desarrollo de la aplicación móvil con geolocalización.
- ✓ **Gerente general de Criss Neón:** Persona interesada en el proyecto, la cual aplicarán el software.
- ✓ **Personal de delivery:** Personal de Criss Neón que usará la aplicación móvil con geolocalización para guiarse en la entrega del pedido al cliente.
- ✓ **Clientes de Criss Neón:** Personas de la empresa Criss Neón quienes utilizarán el aplicativo para solicitar el proceso de control de pedidos.

- **Alcance.**

Desarrollar una aplicación móvil con geolocalización para el proceso de control de pedidos en CRISS NEON S.A.C.

- **Limitaciones.**

El sistema podrá ser usado tanto como el administrador, personal de delivery y clientes de CRISS NEON S.A.C.

La aplicación móvil con geolocalización solo tendrá disponibilidad de hacer pedidos que contengan productos al instante mas no que se programen en un intervalo de días.

La ejecución de la aplicación móvil con geolocalización se re va a realizar únicamente en sistemas operativos de móvil Android desde la versión 5.1 o API Lollipop 22.

Para loguearse, usar el servicio de firebase realtime database y geolocalización será necesario estar conectado a una red de internet.

- **Definición del proyecto.**

Para esta fase se definió el ambiente técnico, la cual será las reuniones con el cliente de CRISS NEON S.A.C. para los avances del aplicativo mediante Zoom, se desarrollará bajo la arquitectura MVC (Modelo Vista, Controlador).

- **Requerimientos Iniciales.**

Se pretende realizar una aplicación móvil con geolocalización, que permita registrar los pedidos solicitados por clientes y el cliente tanto como el repartidor de delivery puedan ver en tiempo real la ubicación del repartidor hasta la llegada del pedido.

- **Requerimientos Funcionales.**

La siguiente tabla se describe los requerimientos funcionales:

Código	Requerimiento	Prioridad
RF01	La aplicación móvil debe tener un módulo de login para el usuario (cliente), administrador y repartidor.	ALTA
RF02	La aplicación móvil debe permitir registrar al cliente por correo electrónico o número de celular.	ALTA
RF03	La aplicación móvil debe permitir enviar un mensaje de texto si el usuario se registra por número de celular.	ALTA
RF04	La aplicación móvil debe permitir visualizar los productos favoritos haciendo click se visualiza el detalle	BAJA
RF05	La aplicación móvil debe listar las categorías registradas	ALTA
RF06	La aplicación móvil debe listar los productos registrados por categoría	ALTA

RF07	La aplicación móvil debe permitir añadir o eliminar complementos, si el cliente desea por cada producto.	ALTA
RF08	La aplicación móvil debe configurar el tamaño del producto.	ALTA
RF09	La aplicación móvil debe permitir comentar y puntualizar cada producto por cliente.	MEDIA
RF10	La aplicación móvil debe permitir visualizar los la puntuación de máximo 5 estrellas calificada por producto en cada sesión del cliente.	MEDIA
RF11	La aplicación móvil debe permitir añadir y eliminar productos a un carrito de compras.	ALTA
RF12	La aplicación móvil obligará al usuario a buscar su dirección para el pedido apoyada por el google places	ALTA
RF13	La aplicación móvil debe permitir registrar una solicitud de pedido.	ALTA
RF14	La aplicación móvil debe permitir repetir el pedido ya solicitado al carrito de compras.	ALTA
RF15	La aplicación móvil debe permitir editar los datos del cliente.	ALTA
RF16	La aplicación móvil debe permitir visualizar el trayecto en tiempo real de cada pedido.	ALTA
RF17	La aplicación móvil debe permitir cancelar un pedido antes que el repartidor comienza el trayecto por el cliente.	ALTA
RF18	La aplicación móvil debe enviar una notificación al administrador de una nueva solicitud de pedido.	ALTA
RF19	La aplicación móvil deberá permitir o denegar el permiso para la subscripción de noticias emitidas por el administrador por parte del cliente.	ALTA

RF20	La aplicación móvil debe permitir agregar, modificar y eliminar las categorías.	ALTA
RF21	La aplicación móvil debe permitir agregar, buscar, modificar y eliminar los productos.	ALTA
RF22	La aplicación móvil debe permitir agregar, modificar y eliminar complementos por cada producto.	ALTA
RF23	La aplicación móvil debe permitir agregar, modificar y eliminar el tamaño por cada producto.	ALTA
RF24	La aplicación móvil debe listar los pedidos por atendido, en camino, entregado y cancelado.	ALTA
RF25	La aplicación móvil debe permitir asignar un pedido a los repartidores permitidos por el administrador.	ALTA
RF26	La aplicación móvil deberá visual el detalle de productos por cada pedido.	ALTA
RF27	La aplicación móvil debe permitir al administrador desactivar y activar las cuentas de los repartidores.	ALTA
RF28	El administrador mediante la aplicación móvil debe enviar noticias nuevas de la empresa a los clientes.	ALTA
RF29	La aplicación móvil puede eliminar, cambiar al estado de cancelado o repetir un pedido por el administrador.	ALTA
RF30	La aplicación móvil debe listar los pedidos asignados por repartidor.	ALTA
RF31	La aplicación móvil debe permitir visualizar el api de google maps del pedido marcando la latitud y longuitud del pedido bajo una línea roja.	ALTA
RF32	La aplicación móvil debe permitir buscar una dirección aparte por el repartidor marcando una línea amarilla.	ALTA
RF33	La aplicación móvil debe permitir marcar con una línea negrita en tiempo real el trayecto del repartidor.	ALTA

RF34	La aplicación móvil debe permitir llamar al cliente que solicito el pedido o repartidor del pedido.	ALTA
RF35	La aplicación móvil debe eliminar el pedido siendo el repartidor dando por terminado la entrega completa.	ALTA
RF36	La aplicación móvil debe notificar al cliente cuando su pedido se entregó con satisfacción.	ALTA
RF37	La aplicación móvil debe permitir visualizar al administrador un reporte en formato PDF del pedido solicitado por el cliente mostrando el subtotal, IGV y total del mismo.	MEDIA
RF38	La aplicación móvil debe elaborar reportes del incremento del porcentaje de entregados completos en un intervalo de días.	ALTA
RF39	La aplicación móvil debe elaborar reportes del decremento del porcentaje de entregas perfectamente recibidas en un intervalo de días.	ALTA
RF40	La aplicación móvil debe elaborar reportes de Entregas Perfectamente Recibidas en un intervalo de días.	ALTA

- **Requerimientos No Funcionales.**

La siguiente tabla se describe los requerimientos

Código	Requerimiento
RNF01	La aplicación móvil se desarrollará en el entorno de lenguaje de programación Java.
RNF02	La aplicación móvil se desarrollará con el ide de Android Studio.
RNF03	La aplicación móvil tendrá como soporte de gestor de base de datos Firebase y SQLite para el carrito de compras.
RNF04	El sistema estará dividido tres aplicaciones separadas para cada pefil (administrador, usuario (cliente) y repartidor).

RNF05	La aplicación móvil usara el api de google maps y sus servicios para la visualización de rutas del pedido.
RNF06	La aplicación móvil se desarrollará bajo el patrón de arquitectura Modelo-Vista-Controlador.
RNF07	La aplicación móvil deberá contar con los permisos permitidos por el usuario para observar la ubicación en tiempo real, llamar y descargar documentos en el aplicativo.

- **Módulos de procesos de la aplicación móvil.**
  - ✓ **Perfil: Cliente**

Módulo	Código	Proceso	Requerimientos
Módulo de Login	**MC01**	La app móvil iniciará con una pantalla de identificación del usuario si en caso no tiene cuenta se puede registrar.	**RF01, RF02, RF03**
Módulo de Edición de Datos Personales	**MC02**	La app móvil deberá permitir editar los datos del cliente (nombre, teléfono) si él lo desea.	**RF16**
Módulo de Carrito de Compras	**MC03**	La app. móvil deberá añadir los productos con sus complementos y su tamaño por categoría y se añadirá al actual carrito de compras.	**RF04, RF05, RF06, RF07, RF08, RF09, RF10, RF11**
Módulo de Solicitud del Pedido	**MC04**	La app. móvil registra el pedido integrado con el servicio de Google Places para confirmar la	**RF12, RF13, RF14**

		dirección destinataria del pedido con el actual carrito de compras.	
Módulo de Control de Pedido	**MC05**	La app móvil deberá visualizar los pedidos por fecha, hora, comentario y estado que pueda repetir el pedido a su vez ver el trayecto actual del repartidor o cancelarlo antes que sea asignado a un repartidor.	**RF16, RF17, RF18, RF33, RF34**
Módulo de Envío de Noticias	**MC06**	La app móvil deberá permitir al cliente subscribirse o dejar de suscribirse a las noticias emitidas por el administrador.	**RF19**

- ✓ **Perfil: Administrador**

Módulo	Código	Proceso	Requerimientos
Módulo de Login	**MA01**	La app móvil iniciará con una pantalla de identificación del administrador si en caso no tiene cuenta se puede registrar.	**RF01, RF02, RF03**
Módulo de Categorías	**MA02**	La app móvil deberá listar las categorías permitiendo agregar, editar o eliminar las categorías.	**RF05, RF20**
Módulo de Productos	**MA03**	La app móvil deberá listar los productos permitiendo agregar, buscar, editar o eliminar los productos también por complemento y tamaño	**RF06, RF21, RF22, RF23**
Módulo de Control de Pedidos	**MA04**	La app móvil deberá listar los pedidos por 4 estados (atendido, en camino, completado o cancelado) que podrá asignarle a un repartidor, eliminar o editar el estado.	**RF24, RF25, RF26, RF29, RF31, RF33, RF37**

Módulo de Repartidores	**MA05**	La app móvil deberá listar a los repartidores registrados y decidir si activar o no su cuenta.	**RF27**
Módulo de Envío de Noticias	**MA06**	La app móvil deberá tener una ventana emergente para enviar noticias a los clientes por parte de los administradores sobre nuevas ofertas, etc.	**RF28**
Módulo de Reportes	**MA07**	La app móvil deberá tener un módulo para visualizar diferentes reportes por fechas.	**RF38, RF39, RF40**

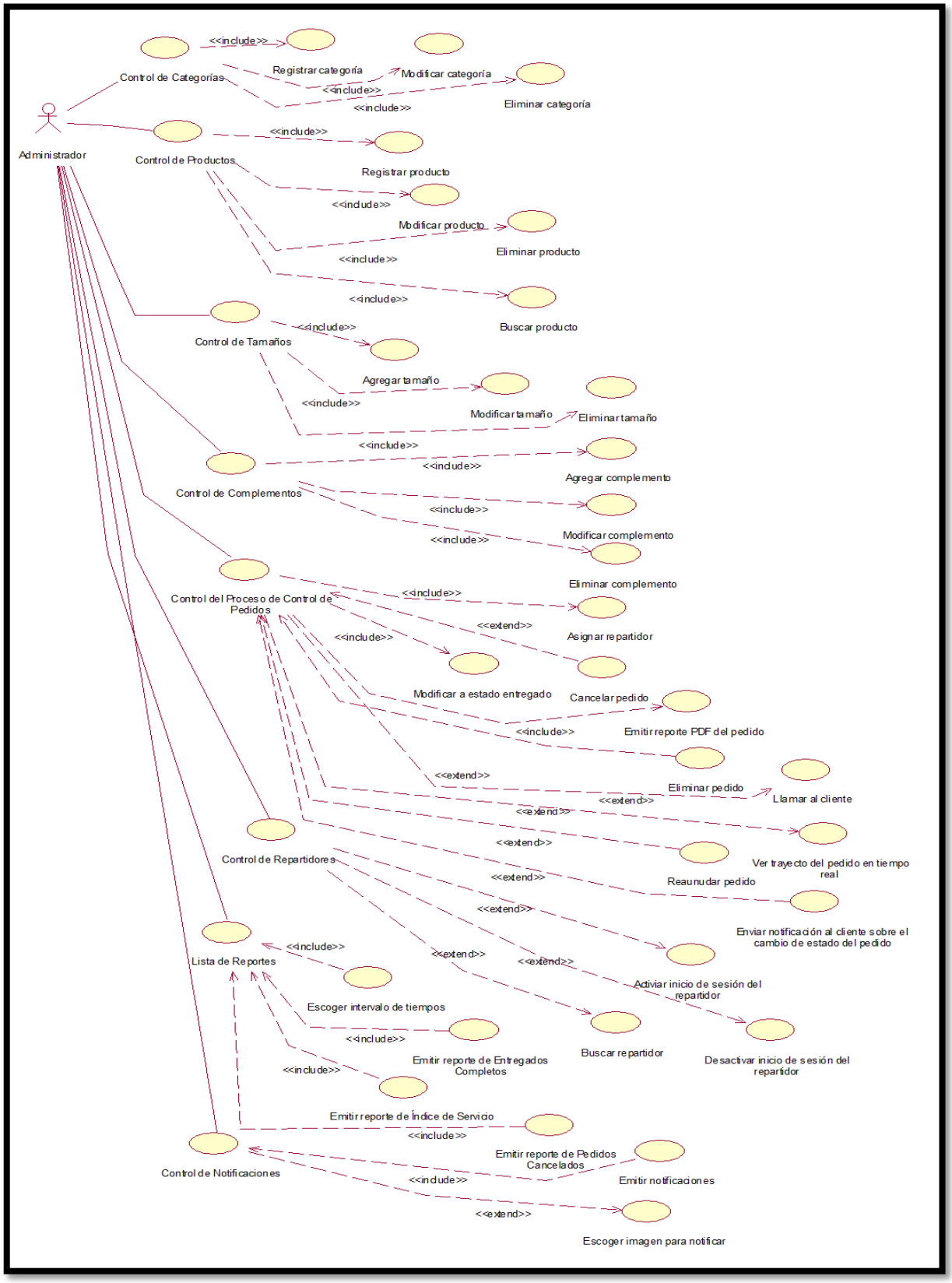
- ✓ **Perfil: Repartidor**

Módulo	Código	Proceso	Requerimientos
Módulo Login	**MR01**	La app móvil iniciará con una pantalla de identificación del repartidor si en caso no tiene cuenta se puede registrar y el repartidor debe pedir permiso al administrador para usar su cuenta.	**RF01, RF02, RF03**
Módulo de Pedidos Asignados	**MR02**	La app móvil deberá listar los pedidos asignados por repartidor, pero no podrá empezar un nuevo pedido sino finalizo el que tiene en trayecto.	**RF30**
Módulo de Trayecto del	**MR03**	La app móvil será capaz de visualizar el servicio de Google Maps ubicando en la ubicación	**RF31, RF32, RF33, RF34, RF35, RF36**

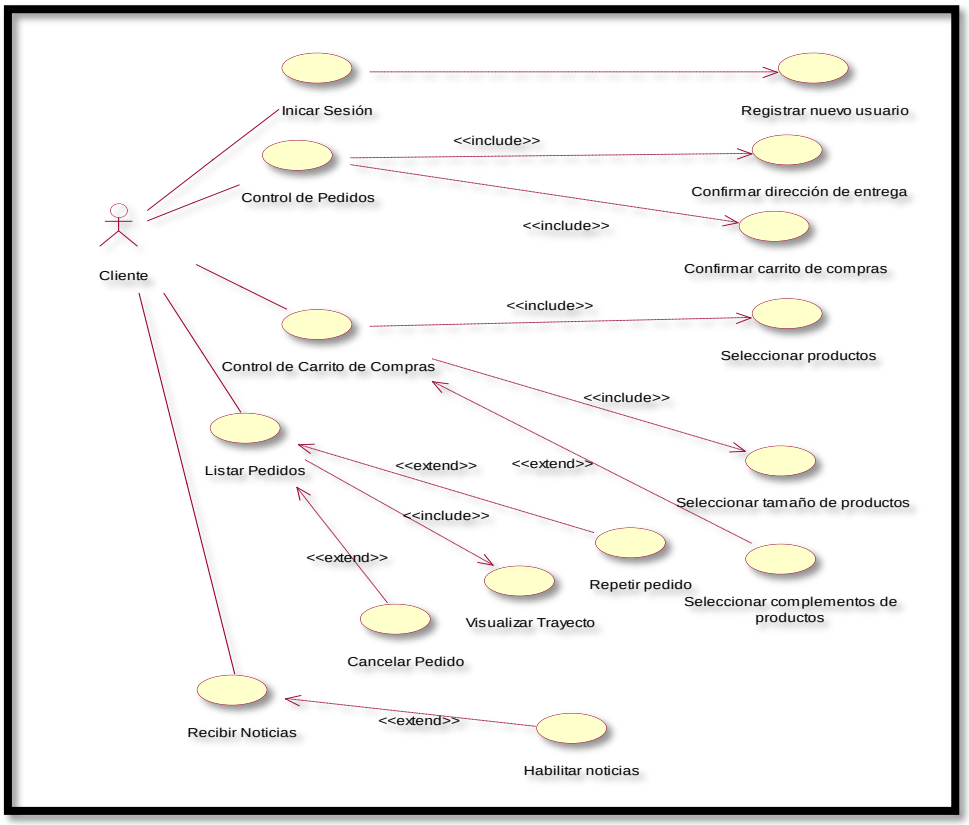
pedido con Geolocaliza-ción.		real del repartidor hasta la ubicación del pedido destinatario marcadando una línea roja y una línea negra por el trayecto en tiempo real del repartidor, que podrá llamar al cliente y ver el trayecto de otra ruta marcando una línea amarilla buscando una dirección, al final tendrá que confirmar entrega completada para empezar con un nuevo pedido notificando al cliente sobre su entrega**.**	

FASE II. INICIALIZACIÓN.

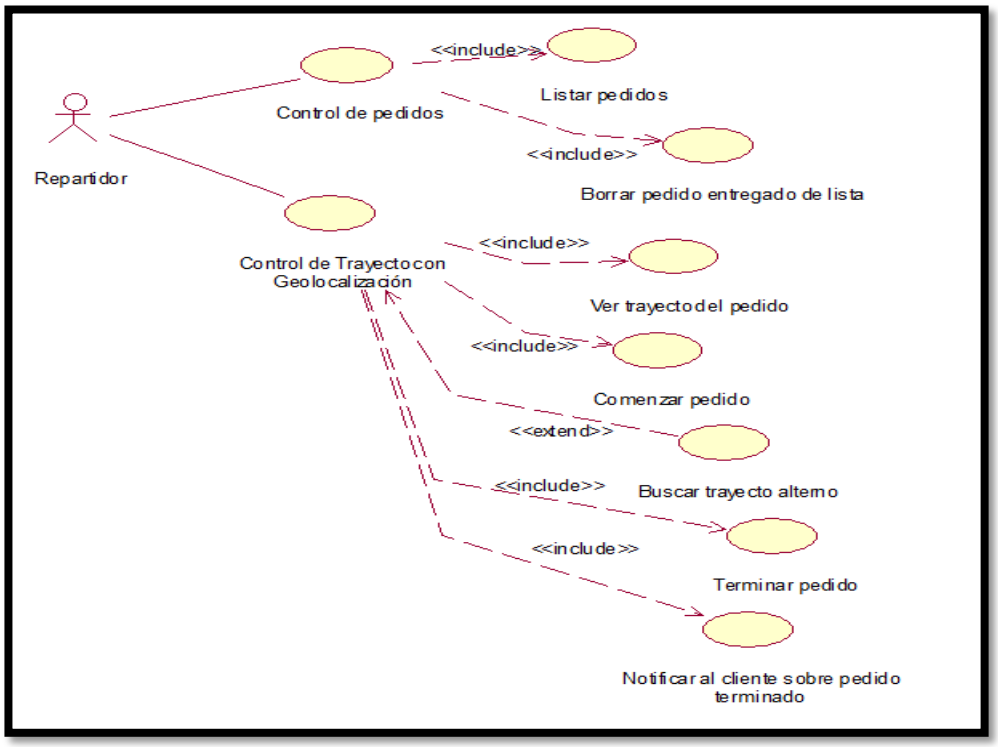
- **Diagrama de clases de la aplicación móvil con geolocalización del usuario Administrador.**



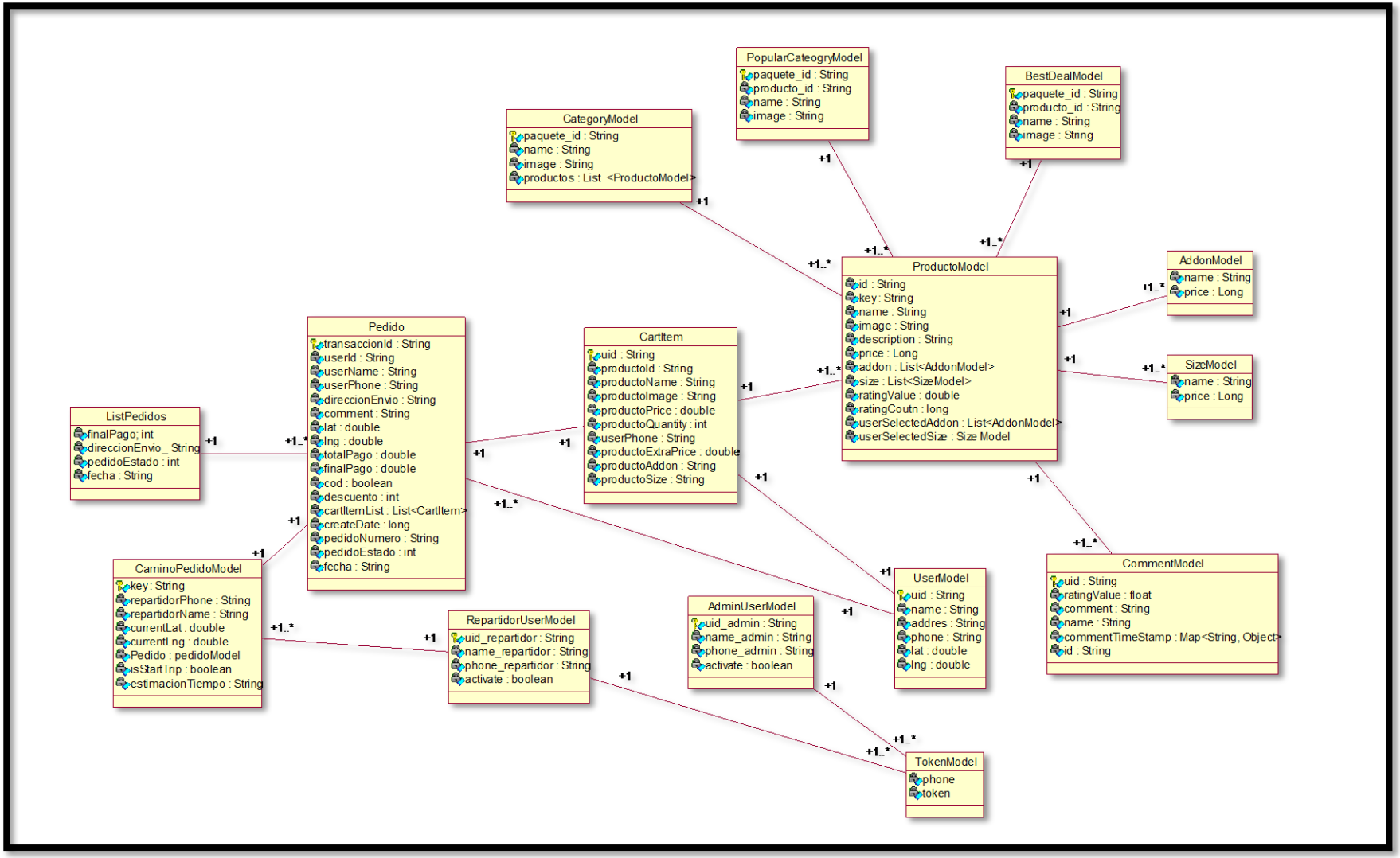
- **Diagrama de clases de la aplicación móvil con geolocalización del usuario Cliente.**



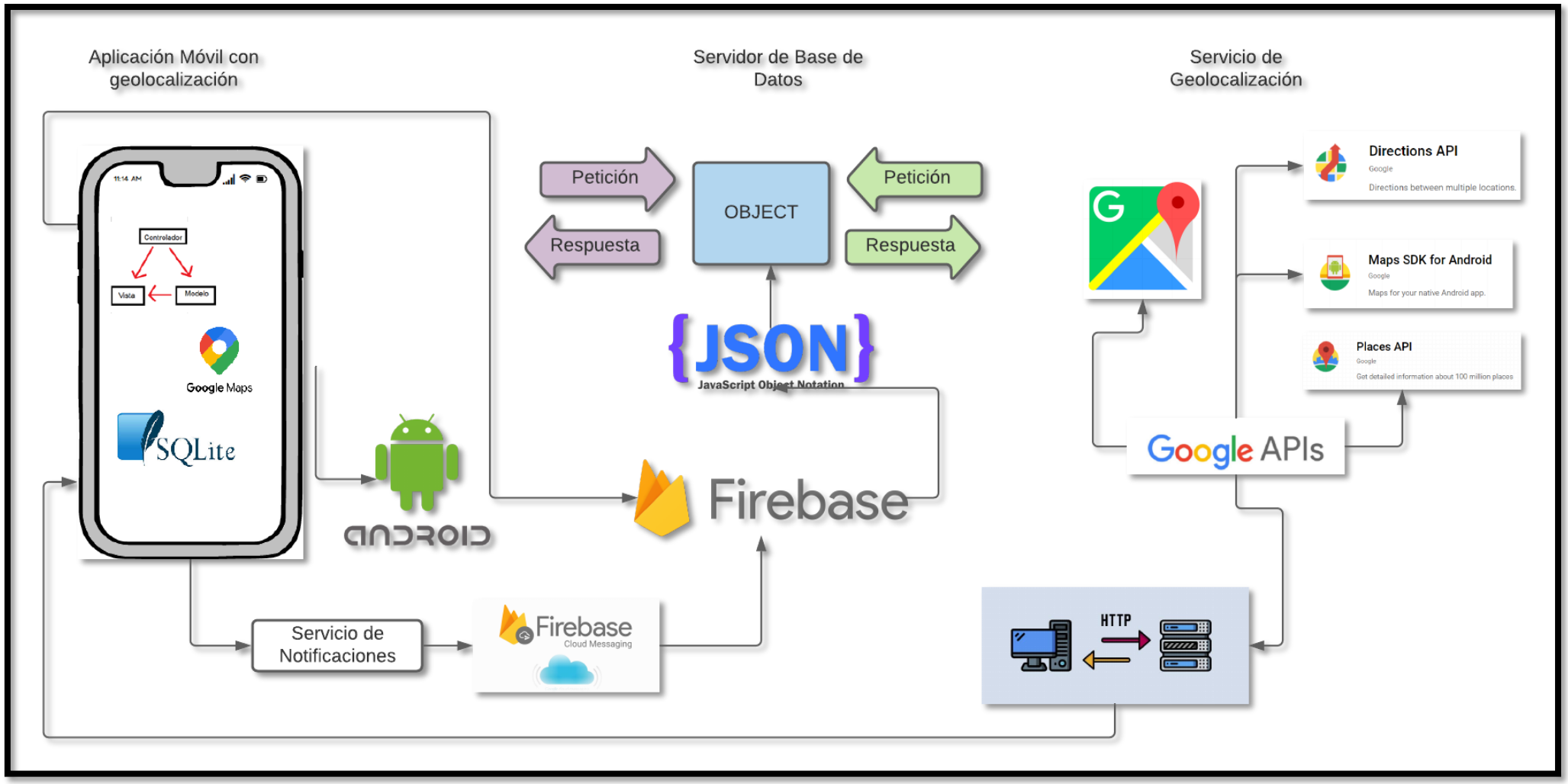
- **Diagrama de clases de la aplicación móvil con geolocalización del usuario Repartidor.**



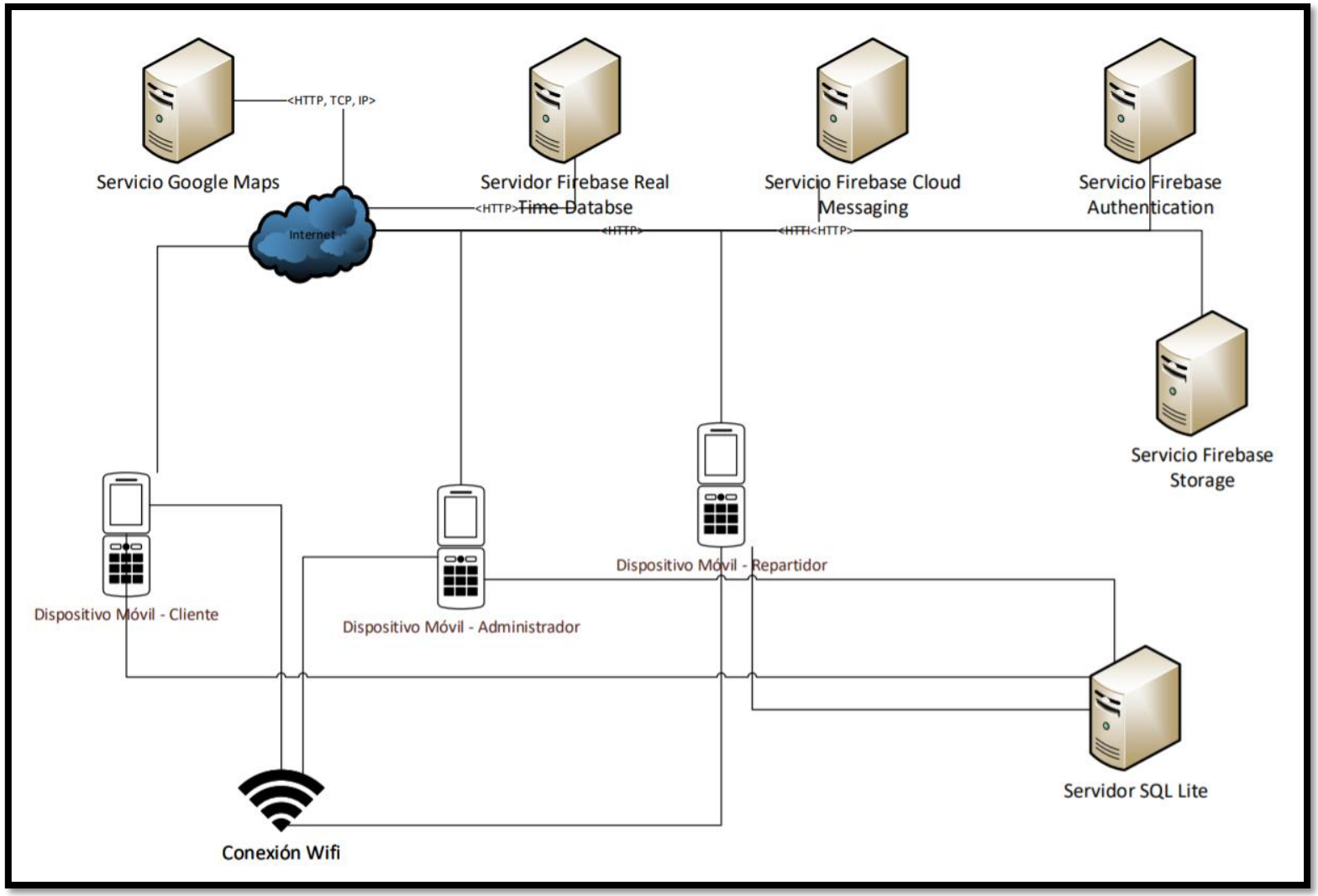
- **Diagrama de clases de la aplicación móvil con geolocalización.**



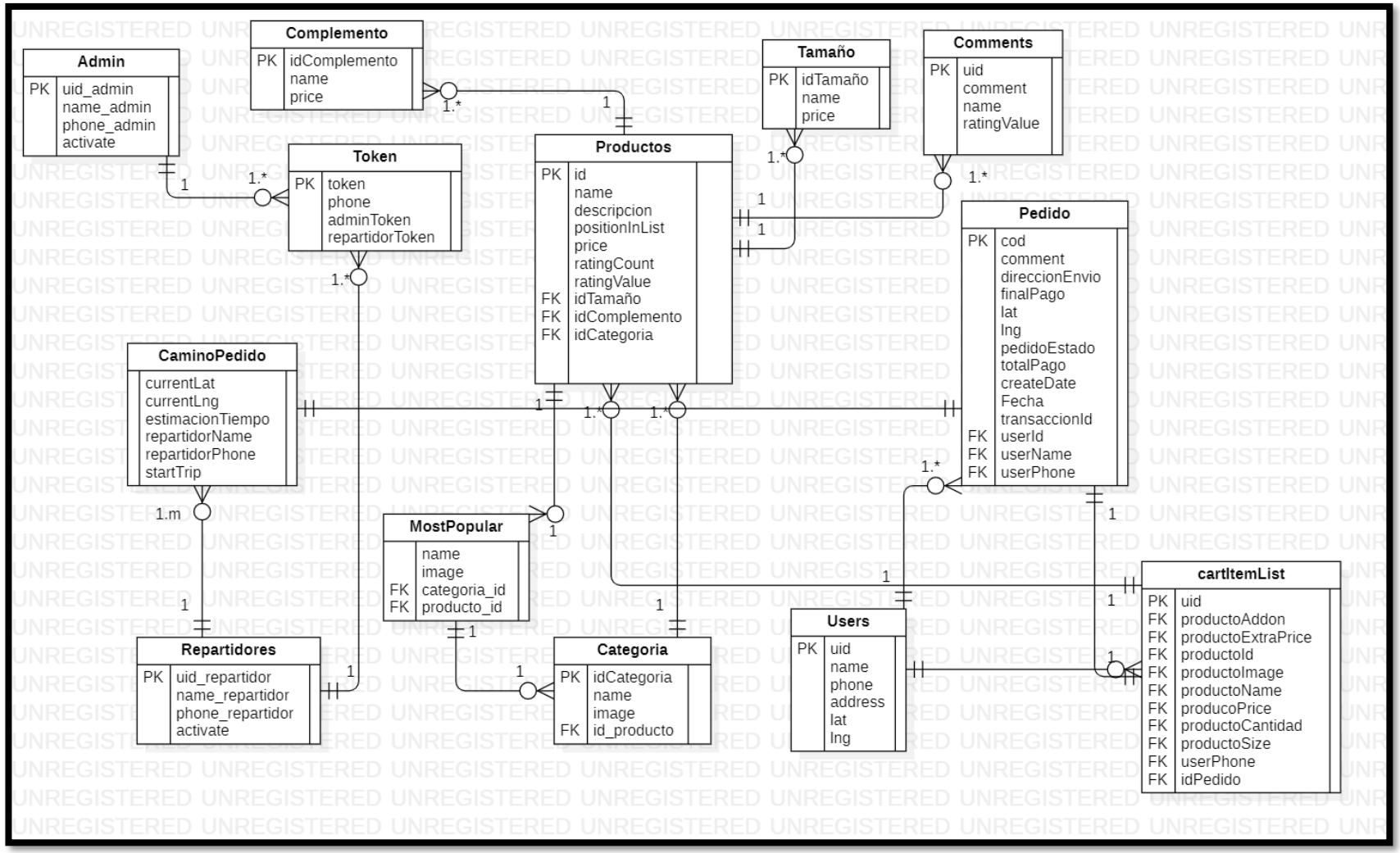
- **Arquitectura de la aplicación móvil con geolocalización (Perfil: Administrador, Cliente y Repartidor)**



- **Arquitectura física de la aplicación móvil con geolocalización.**



- **Diagrama de base de datos elaborada en firebase de la aplicación móvil con geolocalización.**



- **Diccionario de la base de datos.**

  - ✓ **Diccionario BD**

Tablas	Descripción
Admin	Almacena la información de los administradores de la empresa.
Complemento	Almacena la información de los complementos que se brinda por cada producto.
Token	Almacena las solicitudes del usuario basado en inicios de sesión por el usuario o cliente, administradores y repartidores de la empresa.
CaminoPedido	Almacena la información temporal para el repartidor sobre el pedido, latitud y longitud del pedido.
Repartidores	Almacena la información de los repartidores de la empresa.
MostPopular	Almacena la información de los productos más populares en la empresa.
Productos	Almacena la información de los productos que se venden en la empresa.
Categoria	Almacena la información de las categorías que existen en la empresa.
Tamaño	Almacena la información de los tamaños por producto que son adquiridos en la empresa.
Users	Almacena la información de los clientes que usan la aplicación móvil.
Comments	Almacena la información de los comentarios con las puntuaciones realizadas por cliente sobre los productos.
Pedido	Almacena la información del pedido realizado por el cliente.
cartItemList	Almacena la información detallada del pedido por la cantidad, tamaño, complementos de los productos.

A continuación, se describe la información de las tablas en la base de datos no relacional Firebase:

- ✓ **Tabla: Admin**

Columna	Descripción
uid_admin	Código del administrador.
name_admin	Nombre del administrador.
phone_admin	Celular del administrador.
activate	Activación de inicio de sesión del administrador.

- ✓ **Tabla: Complemento**

Columna	Descripción
idComplemento	Código del complemento.
name	Nombre del complemento.
price	Precio del complemento.

- ✓ **Tabla: Tamaño**

Columna	Descripción
idTamano	Código del tamaño.
name	Nombre del tamaño.
price	Precio del tamaño.

- ✓ **Tabla: Comments**

Columna	Descripción
uid	Código del comentario.
comment	Detalle del comentario.
name	Nombre del comentario.
ratingValue	Puntuación del producto.

- ✓ **Tabla: Token**

Columna	Descripción
token	Código del token del cliente.
phone	Celular del cliente.
adminToken	Código del token del administrador.
repartidorToken	Código del token del repartidor.

- ✓ **Tabla: Repartidores**

Columna	Descripción
uid_repartidor	Código del repartidor.
name_repartidor	Nombre del repartidor.
phone_repartidor	Celular del repartidor.
activate	Activación de inicio de sesión del repartidor.

- ✓ **Tabla: MostPoupular**

Columna	Descripción
name	Nombre del producto popular
image	Imagen del producto popular
categoria_id	Id de la categoría del producto popular.
producto_id	Id del producto popular.

- ✓ **Tabla: Categoria**

Columna	Descripción
idCategoria	Código de la categoría.
name	Nombre de la categoría.
image	Imagen de la categoría.
id_producto	Id del producto.

- ✓ **Tabla: Productos**

Columna	Descripción
id	Código del producto.
name	Nombre del producto.
descripción	Descripción del producto.
positionInList	Posición en lista del producto.
price	Precio del producto.
ratingCount	Puntuación del producto.
ratingValue	Puntuación de valoración total del producto.
idTamano	Código del tamaño del producto.
idComplemento	Código del complemento del producto
idCategoria	Código de la categoría del producto.

- ✓ **Tabla: Pedido**

Columna	Descripción
cod	Código del pedido.
comment	Comentario o Indicaciones del pedido.
direccionEnvio	Dirección de envío del pedido.
finalPago	Monto total del pedido.
lat	Latitud de la dirección de envío del pedido.
lng	Longitud de la dirección de envío del pedido.
pedidoEstado	Estado del pedido.
totalPago	Total del final del pedido.
createDate	Código encriptado que incluye fecha y hora del pedido.
Fecha	Fecha del pedido.
transaccionId	Forma de transacción del pedido.
userId	Código del cliente.
userName	Nombre del cliente.

userPhone	Celular del cliente.

- ✓ **Tabla: cartItemList**

Columna	Descripción
uid	Código del detalle del pedido.
productoAddon	Nombre del complemento del pedido.
productoExtraPrice	Precio Extra del complemento del pedido por producto.
productoId	Código del producto adquirido.
productoImage	Imágen del producto.
productoName	Nombre del producto.
productoPrice	Precio del producto.
productoCantidad	Cantidad del producto.
productoSize	Nombre del tamaño del producto
userPhone	Celular del cliente.
idPedido	Código del pedido.

- ✓ **Tabla: Users**

Columna	Descripción
uid	Código del cliente o usuario.
name	Nombre del cliente.
phone	Celular del usuario.
address	Dirección del usuario.
lat	Latitud de la dirección del usuario.
lng	Longitud de la dirección del usuario.

- **Configuración del ambiente de desarrollo.**

En esta fase se define los recursos hardware y software que se usará para el desarrollo del proyecto de investigación. Los recursos técnicos utilizados son los siguientes:

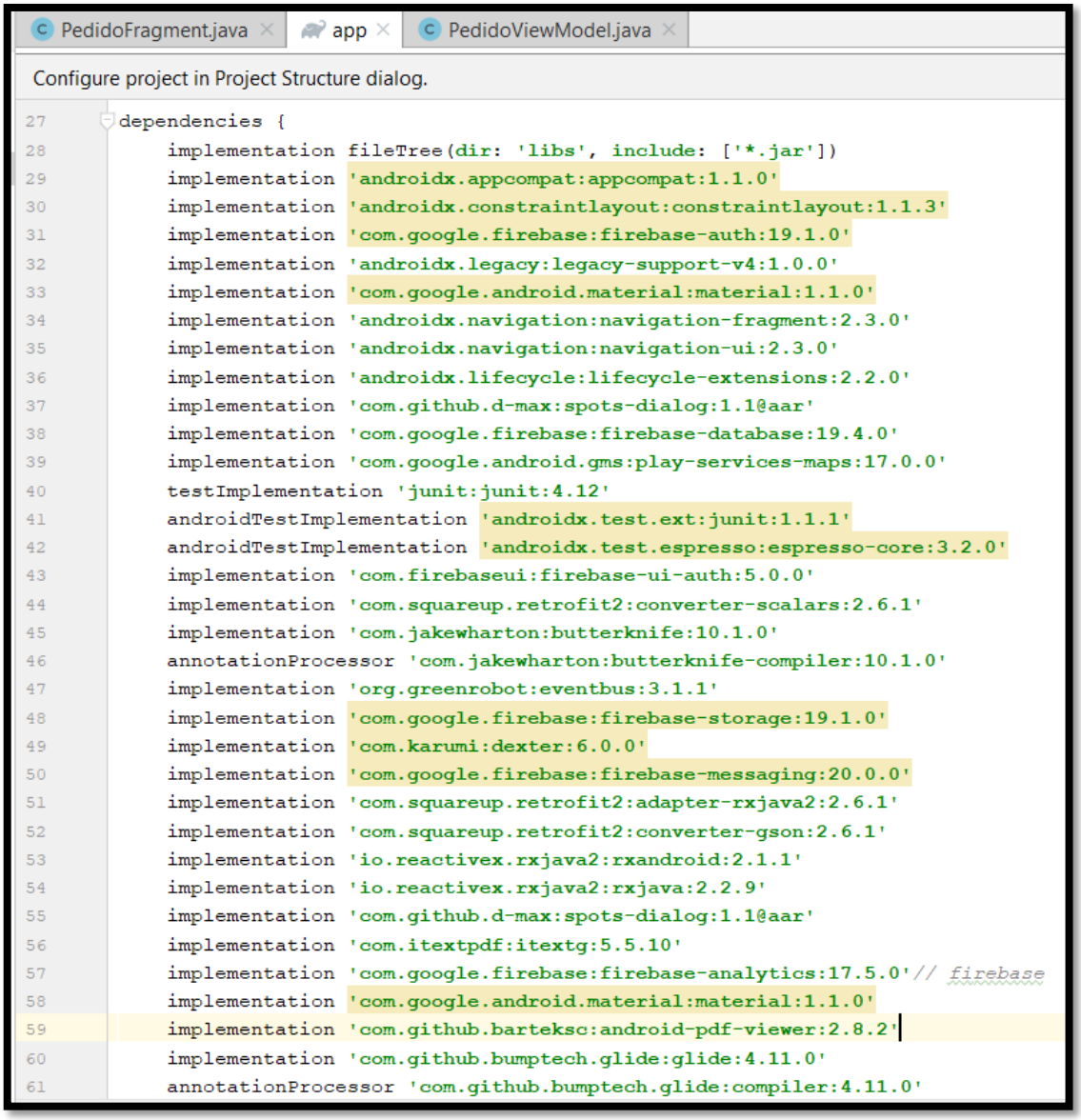
- **Hardware:** Laptop Lenovo Y7000 16gb RAM, 237Gb, Laptop HP Pavilion G4-2055, Intel Core i5 3rd Generation, Dispositivo Móvil Samsung Galaxy J6 (Celular)
- **Software:** IDE Android Studio 3.5, Lenguaje Java, Base de datos Firebase y SQLite, Google APIs: Places API, Directions API, Maps SDK for Android.

- **Configuración de Android Studio.**

Se utilizarán las siguientes librerías y componentes:

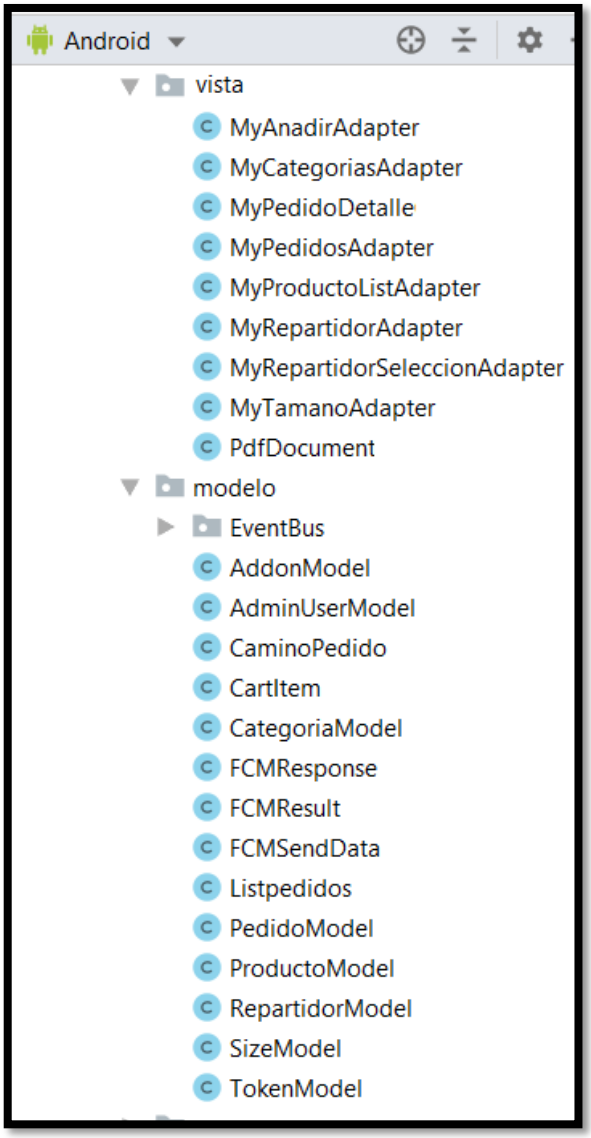
- Google Maps Apis
- Karumi Dexter
- Play Services Location
- Firebase Cloud Messaging (FCM)
- RxJava
- Eventbus
- Glide Bumptech
- Retrofit 2
- Firebase Auth
- Firebase Database
- Play Service Maps
- ButterKnife

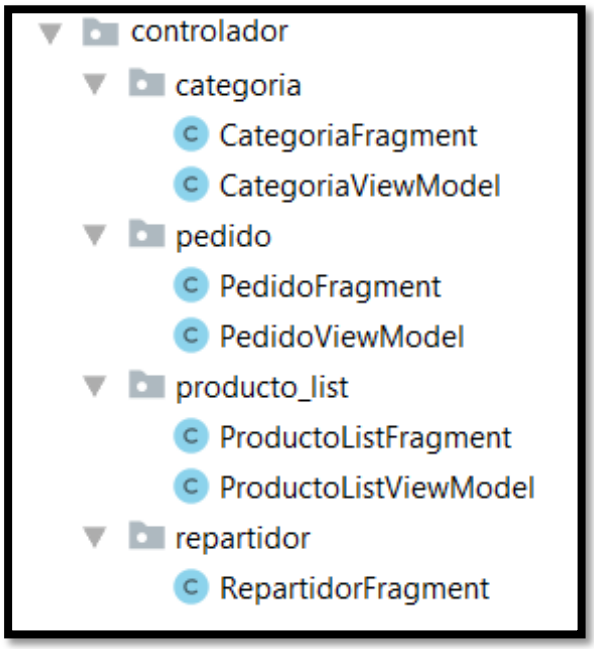
- Servicios API Rest



En la anterior figura se visualiza las dependencias y librerías concatenadas para poder cumplir el debido funcionamiento de la aplicación móvil con geolocalización.

- **Configuración de la arquitectura Modelo-Vista-Controlador (MVC).**





En las figuras mostradas con anterior se visualiza un ejemplo como es la configuración de la arquitectura Modelo Vista Controlador, la vista son las clases que ayudan a visualizar junto con los layouts la información, el modelo son los atributos por clases y el controlador para configuración de conexión con firebase y la interacción con el usuario.

- **Storycard 01: Selección de inicio de sesión.**

NÚMERO	01	DIFICULTAD	Medio	PRIORIDAD	Alta

DESCRIPCION

El usuario (cliente de CRISS NEON S.A.C.), tiene dos opciones para iniciar sesión o registrarse en el sistema al igual que el módulo de administradores y repartidores.

EXCEPCIONES

- Solo podrá acceder los clientes, administradores y repartidores de CRISS NEON S.A.C. al sistema, los repartidores deben esperar hasta que el administrador le habilite su ingreso al sistema.
- Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.

FECHA	ESTADO	COMENTARIO
03/08/2020	Definido	
04/08/2020	Hecho	
22/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 02: Inicio de sesión por correo**

NÚMERO	02	DIFICULTAD	Medio	PRIORIDAD	Alta

DESCRIPCION

El usuario (cliente de CRISS NEON S.A.C.), tendrá la opción para iniciar sesión por correo electrónico al igual que los administradores y repartidores.

EXCEPCIONES

- Solo podrá acceder los clientes, administradores y repartidores de CRISS NEON S.A.C. al sistema, los repartidores deben esperar hasta que el administrador le habilite su ingreso al sistema.
- Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.

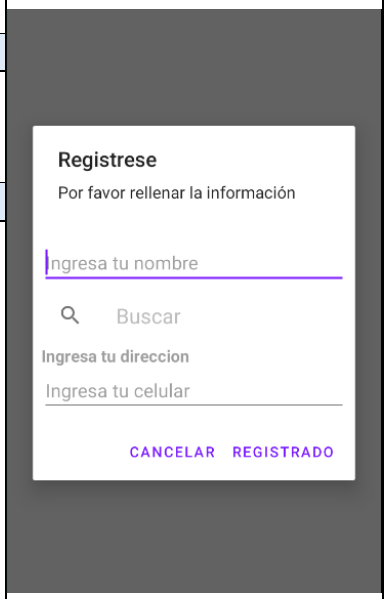
FECHA	ESTADO	COMENTARIO
03/08/2020	Definido	
04/08/2020	Hecho	
22/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 03 : Inicio de sesión por teléfono**

NÚMERO	03	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El usuario (cliente de CRISS NEON S.A.C.), tendrá la opción para registrarse en el sistema mediante su número de teléfono móvil al igual que los administradores y repartidores.					
EXCEPCIONES					
• Solo podrá ingresar los números de celular válidos para el envío del mensaje de 6 dígitos como verificación de ingreso o registro al sistema. • Si el celular no tiene conexión a internet, este proceso no se realizará satisfactoriamente.					
FECHA	**ESTADO**	**COMENTARIO**			
03/08/2020	Definido				
04/08/2020	Hecho				
22/08/2020	Implementado				
07/09/2020	Prueba				
08/09/2020	Verificado				

- **Storycard 04 : Registro de Usuarios**

NÚMERO	04	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El usuario al momento de crear la cuenta de usuario tendrá que digitar el nombre, celular y la dirección tendrá que buscar el usuario mediante el servicio de Google Places.					
EXCEPCIONES					
• Solo se podrá registrar las direcciones válidas localizadas por el Google Places Maps. • Si no se completa todos los campos, no se permitirá el registro. • El cliente debe tener permitido el servicio de ubicación de su celular y el permiso de ubicación del aplicativo que pregunta al comienzo de iniciar la app. • Si el celular no tiene conexión a internet, el registro no será completado.					
FECHA	**ESTADO**	**COMENTARIO**			
03/08/2020	Definido				
04/08/2020	Hecho				
22/08/2020	Implementado				
07/09/2020	Prueba				



08/09/2020	Verificado

- **Storycard 05 : Selección rápida del producto.**

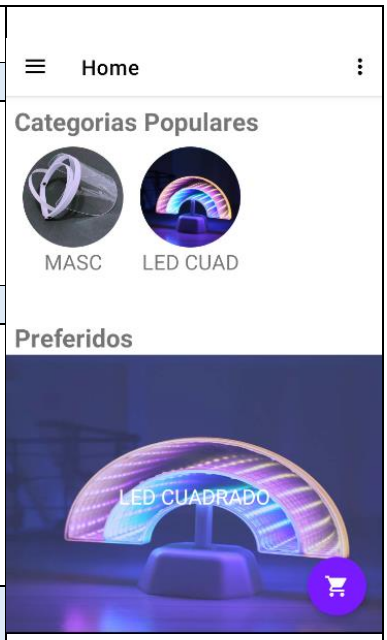
NÚMERO	05	DIFICULTAD	Baja	PRIORIDAD	Alta

DESCRIPCION

El usuario (cliente de CRISS NEON S.A.C.), podrá visualizar los productos populares y preferidos por el público que al hacer click accede de forma inmediata a la información detallada del producto y en el botón de la forma de un carrito, el cliente podrá ver su actual carrito de compras.

EXCEPCIONES

- Solo podrá acceder los clientes, administradores y repartidores de CRISS NEON S.A.C. al sistema.
- Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.



FECHA	ESTADO	COMENTARIO
03/08/2020	Definido	
04/08/2020	Hecho	
22/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 06 : Subscripción a noticias.**

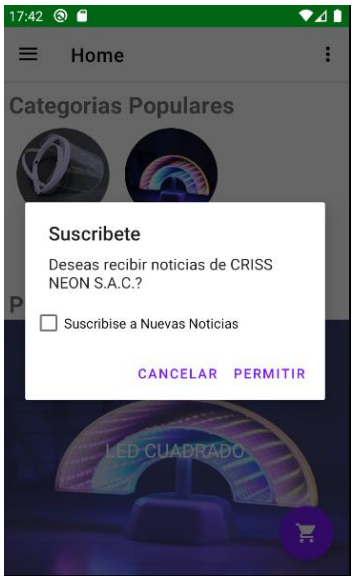
NÚMERO	06	DIFICULTAD	Medio	PRIORIDAD	Alta

DESCRIPCION

El usuario (cliente de CRISS NEON S.A.C.), tendrá la opción de suscribirse a las noticias emitidas de ofertas, etc que van a ser emitidas por CRISS NEON S.A.C marcando el checklist y dando click a Permitir, lado contrario ya no desean recibir noticias dejan de marcar el checklist y le dan click a Permitir.

EXCEPCIONES

- Solo podrá acceder a ese módulo los clientes registrados en el sistema.
- Si el celular no tiene conexión a internet, no se recibirá las noticias correspondientes hasta activar la conexión.



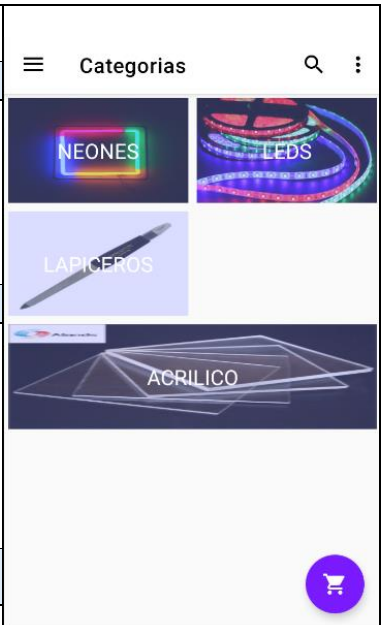
FECHA	ESTADO	COMENTARIO
03/08/2020	Definido	
04/08/2020	Hecho	

22/08/2020	Implementado
07/09/2020	Prueba
08/09/2020	Verificado

- **Storycard 07 : Listar Categorías**

NÚMERO	07	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El usuario (cliente de CRISS NEON S.A.C.), podrá visualizar las categorías disponibles que a su vez podrá buscar la categoría que desee visualizar, al hacer click en cualquier categoría, el cliente visualizará los productos disponibles de la categoría seleccionada.					
EXCEPCIONES					
• Solo podrá acceder a ese módulo los clientes registrados en el sistema. • Si el celular no tiene conexión a internet, la función de no se realizará satisfactoriamente.					

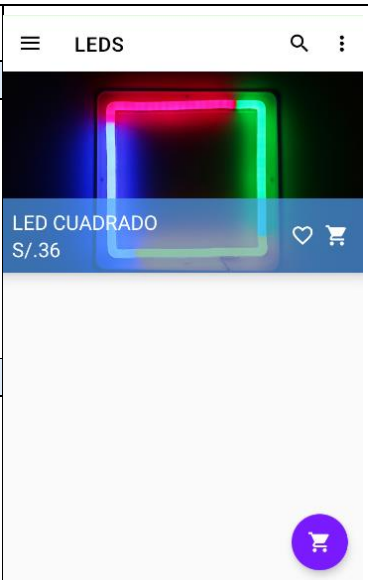
FECHA	ESTADO	COMENTARIO
03/08/2020	Definido	
04/08/2020	Hecho	
22/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 08 : Listar Productos por Categoría.**

NÚMERO	08	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El usuario (cliente de CRISS NEON S.A.C.), tendrá la opción para visualizar los productos disponibles por cada categoría, en la parte superior habrá un icono de una lupa que permitirá al cliente buscar su producto deseado, luego, el cliente al darle click al producto que desea visualizar a detalle y comprar por tamaño y complemento o si no le hace click al icono del carrito para comprar defrente el producto.					
EXCEPCIONES					
• Solo podrá acceder a ese módulo los clientes registrados en el sistema. • Si el celular no tiene conexión a internet, la función de no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
03/08/2020	Definido	



04/08/2020	Hecho
22/08/2020	Implementado
07/09/2020	Prueba
08/09/2020	Verificado

- **Storycard 09 : Detalle del producto.**

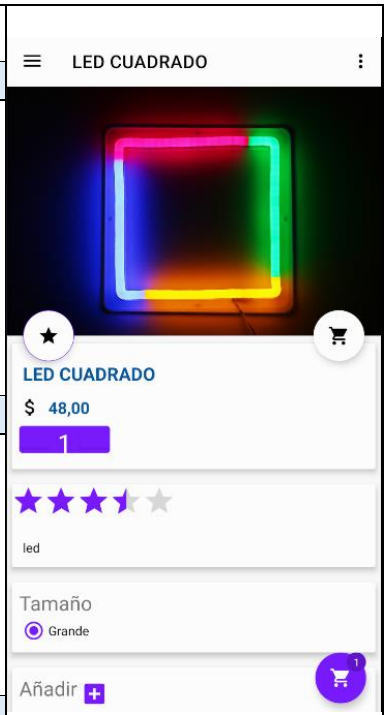
NÚMERO	09	DIFICULTAD	Medio	PRIORIDAD	Alta

DESCRIPCION

El usuario (cliente de CRISS NEON S.A.C.), al escoger el producto que quiso ver a detalle, podrá hacer una puntuación respectiva en el botón de una forma de una estrella, escoger el tamaño del producto y si quiere añadir algún complemento en el icono del "+", también si quiere añadir más de 1 producto, el cliente ya decidido de su producto a añadir en el carrito, puede hacer click en el botón del carrito para agregar dicho producto.

EXCEPCIONES

- Solo podrá acceder a ese módulo los clientes registrados en el sistema.
- Si el celular no tiene conexión a internet, la función de no se realizará satisfactoriamente.

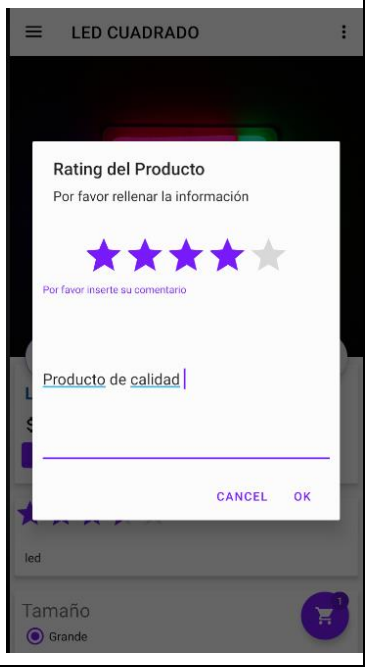


FECHA	ESTADO	COMENTARIO
03/08/2020	Definido	
04/08/2020	Hecho	
22/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 10 : Ranking del Producto.**

NÚMERO	10	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El usuario (cliente de CRISS NEON S.A.C.), podrá evaluar el producto con una calificación de máximo 5 estrellas y un comentario visualizándose en la reputación del producto.					
EXCEPCIONES					
• Solo podrá acceder a ese módulo los clientes registrados en el sistema. • Si el celular no tiene conexión a internet, la función de no se realizará satisfactoriamente.					

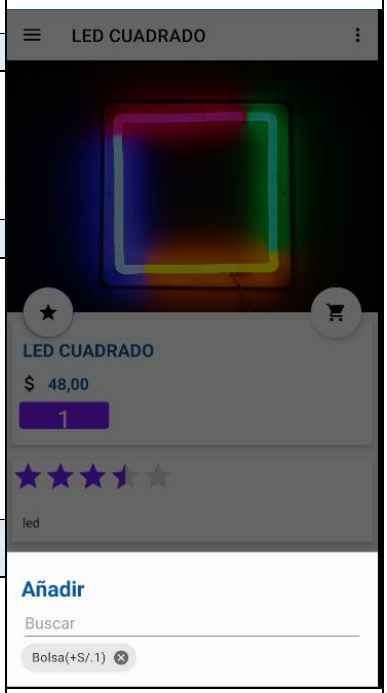
FECHA	ESTADO	COMENTARIO
03/08/2020	Definido	
04/08/2020	Hecho	
22/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 11 : Añadidura del complemento por producto.**

NÚMERO	11	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El usuario (cliente de CRISS NEON S.A.C.), podrá añadir complementos al producto que desee comprar y eso aumentará el costo del producto mostrado en la interfaz del producto a comprar.					
EXCEPCIONES					
• Solo podrá acceder a ese módulo los clientes registrados en el sistema. • Si el celular no tiene conexión a internet, la función de no se realizará satisfactoriamente.					

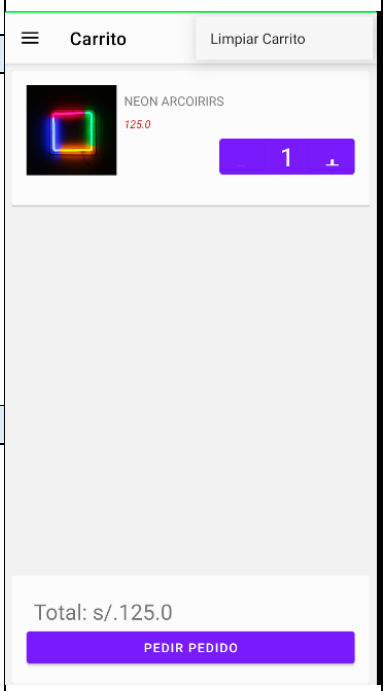
FECHA	ESTADO	COMENTARIO
03/08/2020	Definido	
04/08/2020	Hecho	
22/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 12 : Carrito de Compras**

NÚMERO	12	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El usuario (cliente de CRISS NEON S.A.C.), podrá ver los productos que desea comprar en el carrito al hacer click en el icono del carrito de comprar que aparece en cada interfaz del sistema y puede aún seleccionar si desea aumentar o disminuir la cantidad de productos adquiridos, eso de forma automática modifica el total del carrito, en la parte superior se visualiza tres puntos que permitirá limpiar todo el carrito de compras y si despliega un producto tendrá la opción de eliminarlo de su carrito de compras.					
EXCEPCIONES					
• Solo podrá acceder a ese módulo los clientes registrados en el sistema. • Si el celular no tiene conexión a internet, la función de no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
03/08/2020	Definido	
04/08/2020	Hecho	
22/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



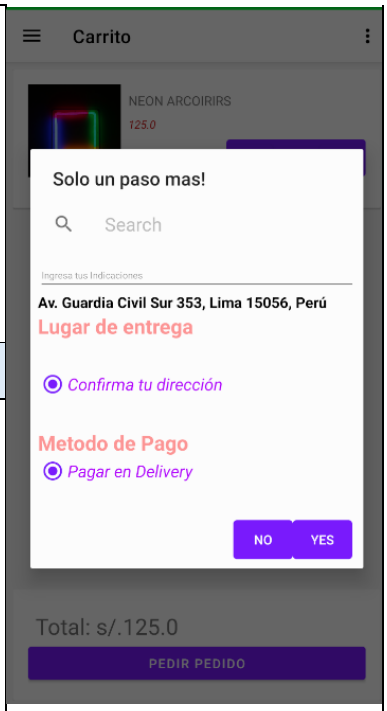
- **Storycard 13 : Solicitud del Pedido**

NÚMERO	13	DIFICULTAD	Alta	PRIORIDAD	Alta
DESCRIPCION					
El usuario (cliente de CRISS NEON S.A.C.), podrá solicitar su pedido del carrito de compras de los productos que desee, pero tendrá que validar la dirección registrada mostrada en la interfaz o sino tiene la opción de reemplazar por otra dirección localizada, dar indicaciones sobre el lugar para llegar.					
EXCEPCIONES					
• Solo podrá acceder a ese módulo los clientes registrados en el sistema. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente. • Solo se podrá registrar las direcciones válidas localizadas por el Google Places Maps.					

- Si no se completa todos los campos, no se permitirá hacer la solicitud del pedido.

- El cliente debe tener permitido el servicio de ubicación de su celular y el permiso de ubicación del aplicativo que pregunta al comienzo de iniciar la app.

FECHA	ESTADO	COMENTARIO
04/08/2020	Definido	
05/08/2020	Hecho	
23/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 14 : Listar Pedidos**

NÚMERO	14	DIFICULTAD	Alta	PRIORIDAD	Alta

DESCRIPCION

El usuario (cliente de CRISS NEON S.A.C.), podrá visualizar sus pedidos realizados que van a contener la fecha, hora, indicaciones o comentario, estado y foto del primer producto del pedido. Y tendrá la opción que, al desplazar de un pedido, podrá repetir el mismo pedido adjuntando lo solicitado en el carrito de compras, ver el trayecto actual del pedido y si desea cancelar su pedido.

EXCEPCIONES

- Solo podrá acceder a ese módulo los clientes registrados en el sistema.

- El cliente debe tener permitido el servicio de ubicación de su celular y el permiso de ubicación del aplicativo que pregunta al comienzo de iniciar la app.

- El cliente va poder acceder al trayecto del pedido siempre y cuando el pedido este en transcurso o camino.

- El cliente va poder eliminar el pedido siempre y cuando el pedido no haya sido asignado a un repartidor.

- Si el celular no tiene conexión a internet, la función de no se realizará satisfactoriamente.

FECHA	**ESTADO**	**COMENTARIO**
05/08/2020	Definido	
06/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 15 : Edición de datos del Cliente**

NÚMERO	**15**	**DIFICULTAD**	**Medio**	**PRIORIDAD**	**Alta**

DESCRIPCION

El usuario (cliente de CRISS NEON S.A.C.), podrá editar su nombre y/o dirección.

EXCEPCIONES

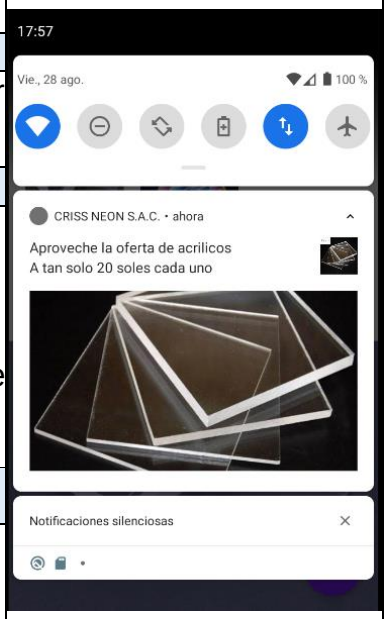
- Solo podrá acceder a ese módulo los clientes registrados en el sistema.
- Solo se podrá registrar las direcciones válidas localizadas por el Google Places Maps.
- El cliente debe tener permitido el servicio de ubicación de su celular y el permiso de ubicación del aplicativo que pregunta al comienzo de iniciar la app.
- Si el celular no tiene conexión a internet, la modificación no será completada.

FECHA	**ESTADO**	**COMENTARIO**
05/08/2020	Definido	
06/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 16 : Mostrar noticias**

NÚMERO	16	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El usuario (cliente de CRISS NEON S.A.C.), podrá visualizar las noticias emitidas por la empresa CRISS NEON S.A.C.					
EXCEPCIONES					
• Solo podrá visualizar las noticias, los clientes registrados en el sistema y se hayan suscrito a las noticias. • Si el celular no tiene conexión a internet, el envío de noticias, no se realizará satisfactoriamente.					

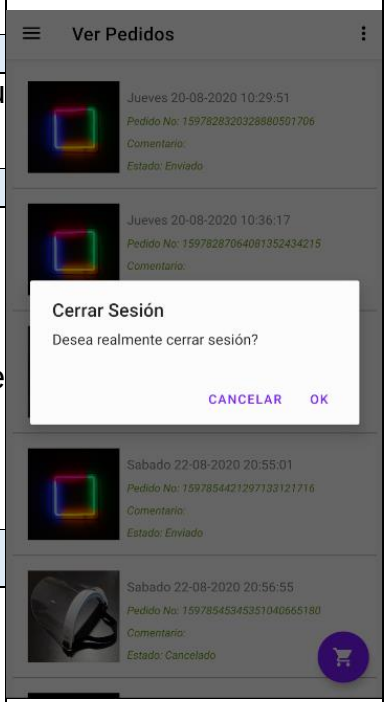
FECHA	ESTADO	COMENTARIO
05/08/2020	Definido	
06/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 17 : Cerrar Sesión**

NÚMERO	17	DIFICULTAD	Baja	PRIORIDAD	Alta
DESCRIPCION					
El usuario (cliente de CRISS NEON S.A.C.), podrá cerrar su sesión en el aplicativo móvil.					
EXCEPCIONES					
• Solo podrá visualizar las noticias, los clientes registrados en el sistema y se hayan suscrito a las noticias. • Si el celular no tiene conexión a internet, la función no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
05/08/2020	Definido	
06/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 18: Geolocalización del trayecto del pedido del módulo cliente**

NÚMERO	18	DIFICULTAD	Alta	PRIORIDAD	Alta

DESCRIPCION

El usuario (cliente de CRISS NEON S.A.C.), va poder ver el transcurso actual del pedido visualizando el nombre, celular del repartidor y el tiempo estimado de la llegada del pedido, otorgando el tiempo real el transcurso del pedido por una línea roja de la localización actual del repartidor y el destino final del cliente, finalmente en la parte superior el cliente podrá llamar al repartidor de manera inmediata haciendo click al botón “LLAMAR AL REPARTIDOR”.

EXCEPCIONES

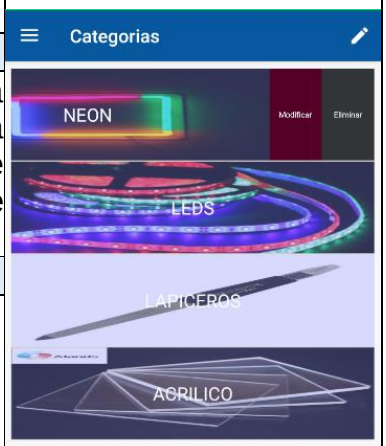
- Solo podrá acceder a ese módulo los clientes registrados en el sistema.
- El cliente debe tener permitido el servicio de ubicación de su celular y el permiso de ubicación del aplicativo que pregunta al comienzo de iniciar la app.
- Si el celular no tiene conexión a internet, el registro no será completado.

FECHA	ESTADO	COMENTARIO
06/08/2020	Definido	
21/08/2020	Hecho	
01/09/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 19 : Listar Edición de Categorías**

NÚMERO	19	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá visualizar las categorías vigentes en la empresa a su vez de modificar o eliminar la categoría desplazándose de la parte derecha de la categoría y en la parte superior se observa un icono de lápiz que al hacer click se puede agregar una categoría nueva.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

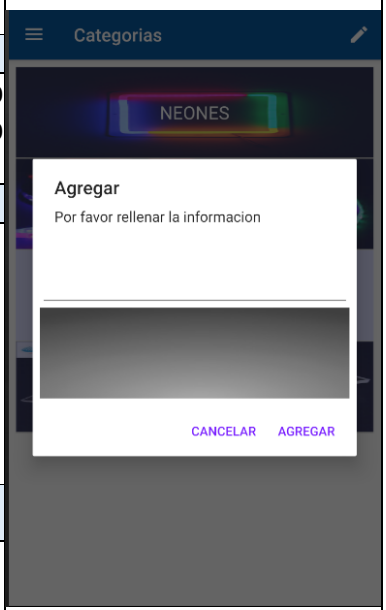
FECHA	ESTADO	COMENTARIO
05/08/2020	Definido	
06/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 20 : Agregar Categorías**

NÚMERO	20	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá agregar nuevas categorías ingresando el nombre y la foto de la categoría en la parte superior del icono del lápiz.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
05/08/2020	Definido	
06/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 21 : Modificar Categorías**

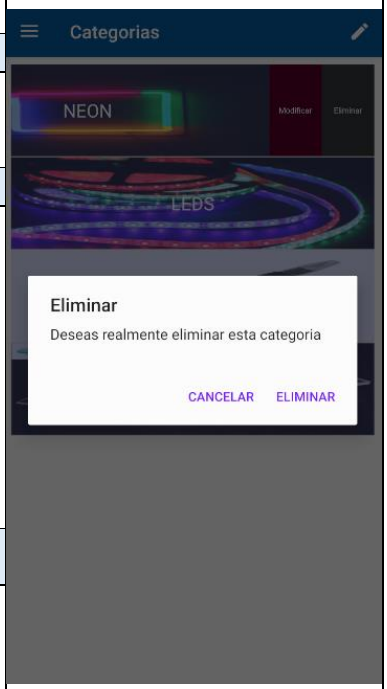
NÚMERO	21	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá modificar las categorías existentes editando el nombre o foto de la categoría.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
05/08/2020	Definido	
06/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 22 : Eliminar Categorías**

NÚMERO	22	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá eliminar una categoría existente.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
05/08/2020	Definido	
06/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 23 : Listar Edición de Productos por Categoría**

NÚMERO	23	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá visualizar los productos vigentes por categoría en la empresa a su vez de modificar eliminar la categoría, agregar complementos y tamaño desplazándose de la parte derecha de la categoría y en la parte superior se observa un icono de lápiz que al hacer click se puede agregar un nuevo producto y una lupa para buscar un producto en concreto.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
06/08/2020	Definido	
07/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 24 : Registrar complementos por producto**

NÚMERO	24	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá agregar, editar o eliminar un complemento del producto seleccionado mediante la barra desplazadora del producto.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
06/08/2020	Definido	
07/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 25 : Registrar tamaños por producto**

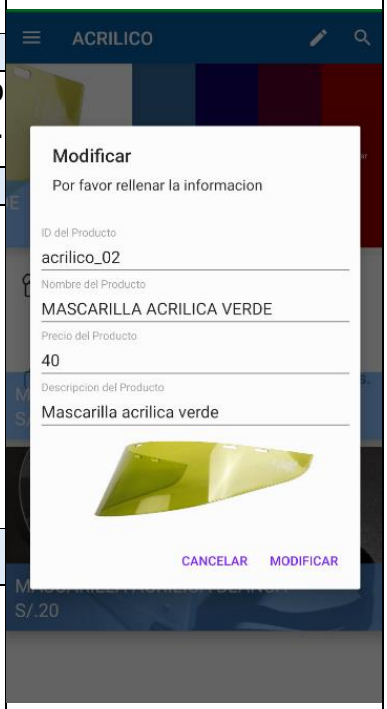
NÚMERO	25	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá agregar, editar o eliminar un tamaño del producto seleccionado mediante la barra desplazadora del producto.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	FECHA	COMENTARIO
06/08/2020	Definido	
07/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 26 : Modificar productos**

NÚMERO	26	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá modificar el producto seleccionado editando el id, nombre, precio, descripción y/o foto del producto.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

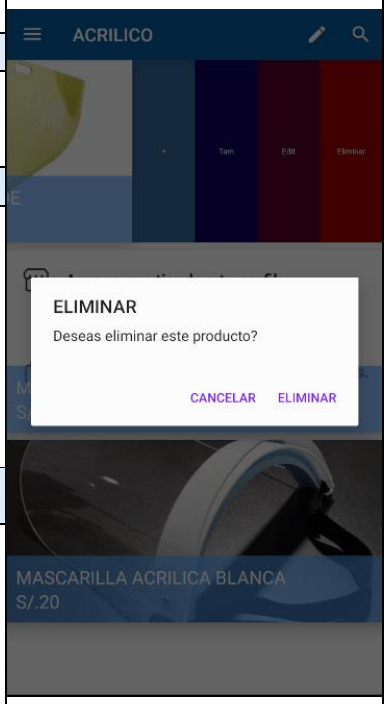
FECHA	ESTADO	COMENTARIO
06/08/2020	Definido	
07/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 27 : Eliminar productos**

NÚMERO	27	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá eliminar un producto existente.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

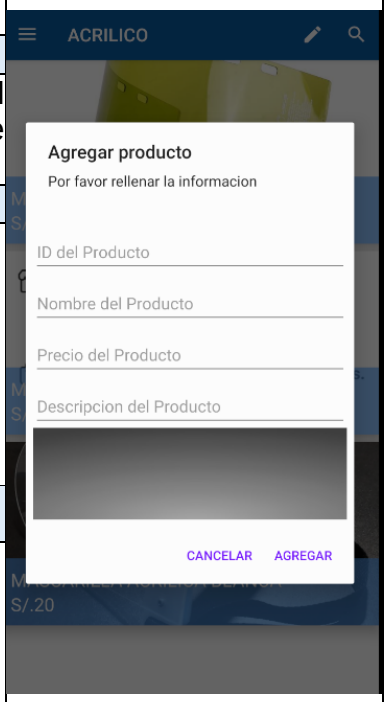
FECHA	ESTADO	COMENTARIO
06/08/2020	Definido	
07/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 28: Agregar productos**

NÚMERO	28	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá agregar nuevos productos ingresando el id, nombre, precio y descripción del producto en la parte superior del icono del lápiz.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
06/08/2020	Definido	
07/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 29: Emitir noticias**

NÚMERO	29	DIFICULTAD	Medio	PRIORIDAD	Alta

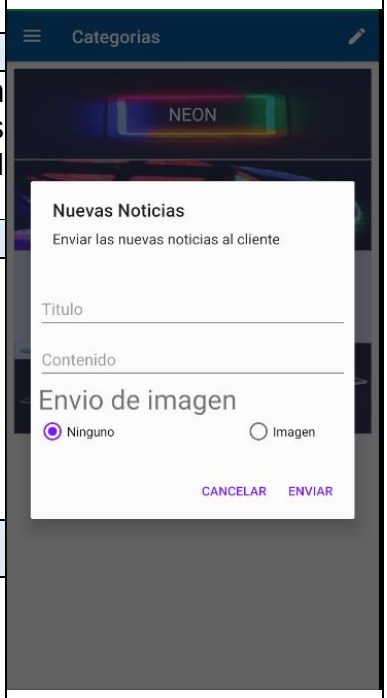
DESCRIPCION

El administrador podrá emitir una nueva noticia ya se solo un contenido o un contenido con imagen enviando a los usuarios que han permitido la difusión de noticias de CRISS NEÓN S.A.C.

EXCEPCIONES

- Solo podrá acceder los administradores de CRISS NEÓN S.A.C.
- Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.

FECHA	ESTADO	COMENTARIO
05/08/2020	Definido	
06/08/2020	Hecho	
25/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 30 : Listar Pedidos**

NÚMERO	30	DIFICULTAD	Medio	PRIORIDAD	Alta

DESCRIPCION

El administrador podrá visualizar la lista y cantidad de pedidos emitidos por el estado de este permitiendo editar el pedido, eliminarlo, llamar al cliente, ver la ubicación actual del repartidor e imprimir el pedido.

EXCEPCIONES

- Solo podrá acceder los administradores de CRISS NEÓN S.A.C.
- Solo se podrá observar la ubicación actual del repartidor en el estado Pedido en Camino y el repartidor haya empezado el transcurso.
- Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.

FECHA	ESTADO	COMENTARIO
09/08/2020	Definido	
10/08/2020	Hecho	
26/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 31 : Clasificación de los pedidos por Estado**

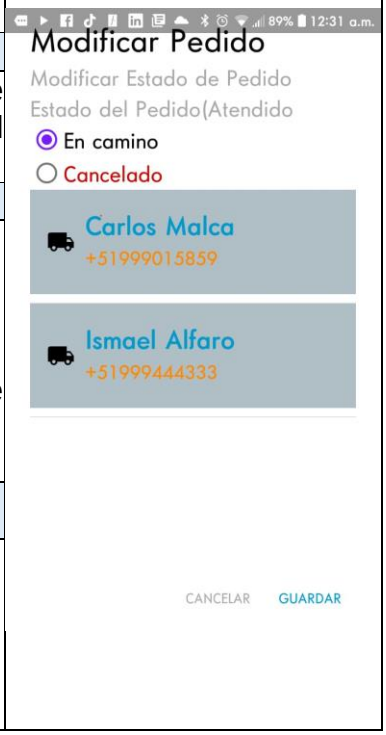
NÚMERO	31	DIFICULTAD	Alta	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá visualizar los pedidos emitidos por los clientes por 4 estados (Atendido, En camino, Entregado, Cancelado) haciendo click en las 3 rayas de la parte superior visualizará una ventana inferior que lo apoyará a seleccionar que estado de pedido quiere visualizar.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
10/08/2020	Definido	
13/08/2020	Hecho	
26/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 32: Modificar pedido en estado Atendido**

NÚMERO	32	DIFICULTAD	Alta	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá asignar el pedido solicitado de un cliente a un repartidor o sino cancelar el pedido finalmente guardar el estado del pedido que es “en camino”.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

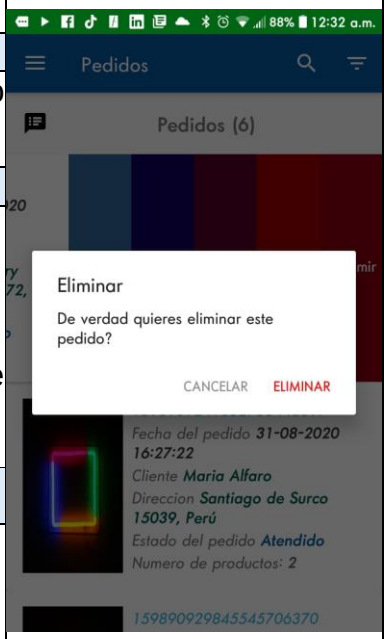
FECHA	ESTADO	COMENTARIO
13/08/2020	Definido	
14/08/2020	Hecho	
26/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 33 : Eliminar Pedido**

NÚMERO	33	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá eliminar un pedido borrando el pedido solicitado del cliente					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

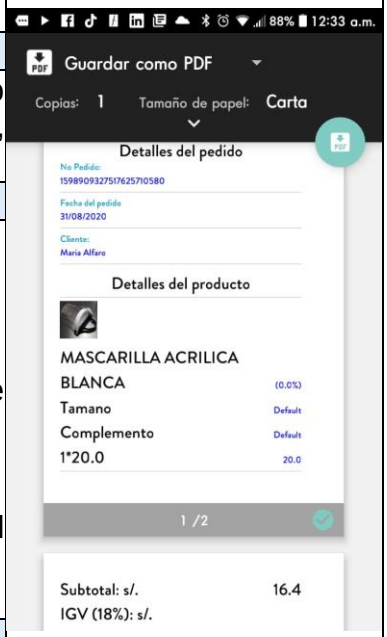
FECHA	ESTADO	COMENTARIO
13/08/2020	Definido	
14/08/2020	Hecho	
26/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 34 : Emitir PDF del pedido.**

NÚMERO	34	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá visualizar un archivo pdf del pedido detallando el número, fecha, cliente, productos, tamaño, complementos, cálculos (subtotal, IGV y total).					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente. • El dispositivo del administrador debe estar permitido el permiso de descarga de archivos.					

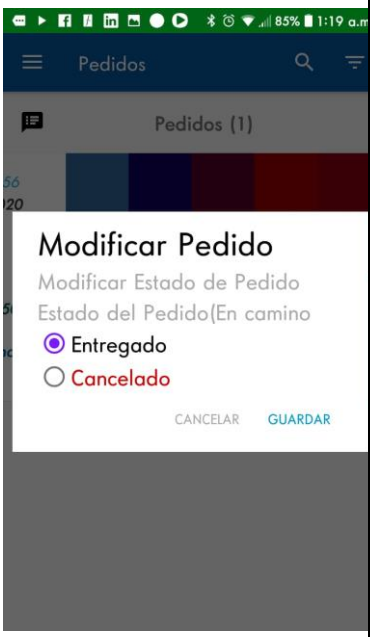
FECHA	ESTADO	COMENTARIO
14/08/2020	Definido	
15/08/2020	Hecho	
27/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 35 : Modificar pedido en estado En camino**

NÚMERO	35	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá modificar el pedido en el estado en camino para que el pedido pueda cambiarse al estado entregado o cancelado.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

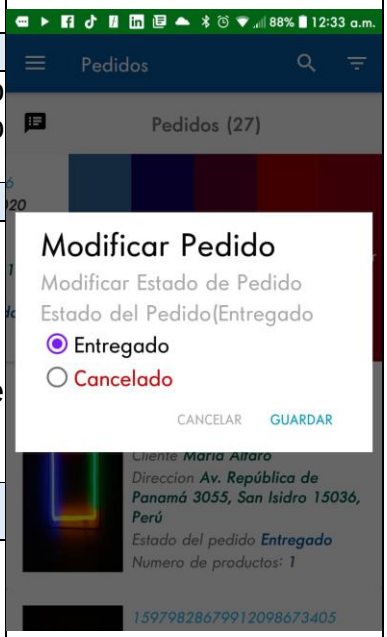
FECHA	ESTADO	COMENTARIO
14/08/2020	Definido	
15/08/2020	Hecho	
27/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 36: Modificar pedido en estado Entregado**

NÚMERO	36	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá modificar el pedido en el estado entregado para que el pedido pueda cambiarse al estado cancelado.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

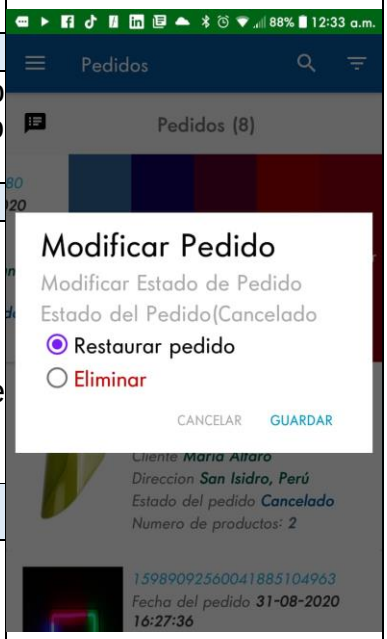
FECHA	ESTADO	COMENTARIO
15/08/2020	Definido	
16/08/2020	Hecho	
27/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 37 : Modificar pedido en estado Cancelado**

NÚMERO	37	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá modificar el pedido en el estado cancelado para que el pedido pueda cambiarse al estado atendido o eliminar el pedido.					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
16/08/2020	Definido	
17/08/2020	Hecho	
27/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 38 : Detalle del Pedido**

NÚMERO	38	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El usuario y administrador podrán visualizar el detalle del pedido de los productos, cantidad, complemento y tamaño.					
EXCEPCIONES					
• Solo podrá acceder los administradores y usuarios registrados en el sistema de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
16/08/2020	Definido	
17/08/2020	Hecho	
27/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 39 : Geolocalización del trayecto del pedido del módulo administrador**

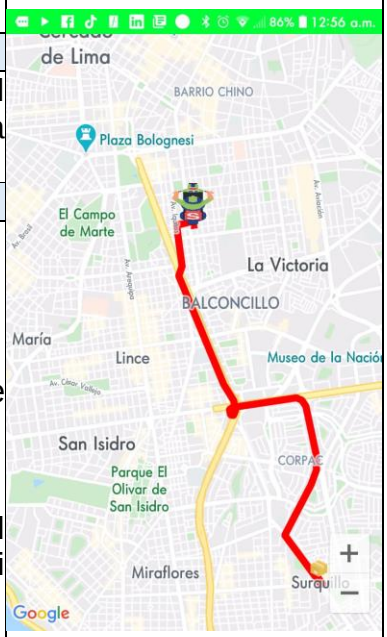
NÚMERO	39	DIFICULTAD	Alta	PRIORIDAD	Alta

DESCRIPCION

El administrador podrá visualizar el trayecto en tiempo actual del transcurso del pedido definiéndose en una línea roja, la ruta del repartidor será representada por una línea negra.

EXCEPCIONES

- Solo podrá acceder los administradores y usuarios registrados en el sistema de CRISS NEÓN S.A.C.
- Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.
- El dispositivo móvil del administrador debe activar el permiso de ubicación para permitir al aplicativo usar el api de Google Maps.
- Debe estar activado el servicio de ubicación el dispositivo móvil del administrador.

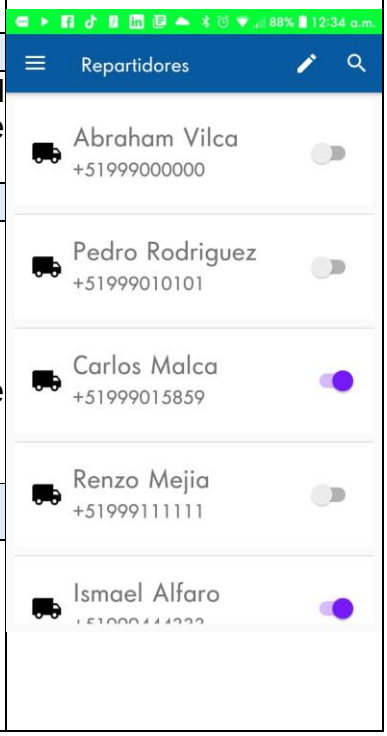


FECHA	ESTADO	COMENTARIO
20/08/2020	Definido	
22/08/2020	Hecho	
28/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 40 : Listar Repartidores**

NÚMERO	40	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá ver los repartidores registrados, él decide si activa o desactiva la cuenta del repartidor además que puede buscar al repartidor deseado.					
EXCEPCIONES					
• Solo podrá acceder los administradores y usuarios registrados en el sistema de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
23/08/2020	Definido	
24/08/2020	Hecho	
28/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 41 : Menú para acceder a reportes**

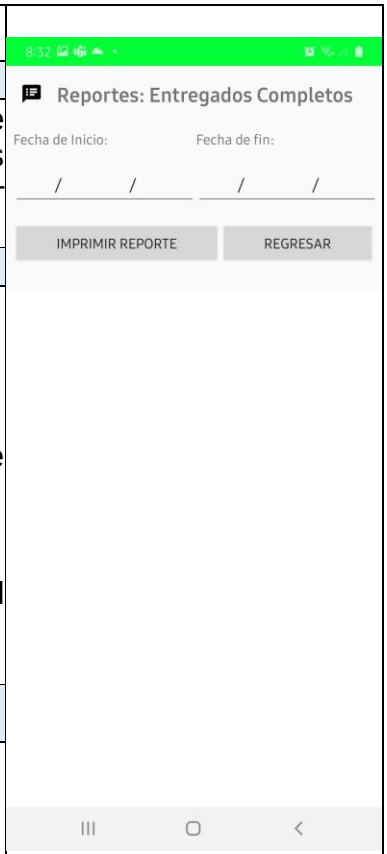
NÚMERO	41	DIFICULTAD	Baja	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá visualizar una ventana emergente que le indica los reportes por % de entregados completos, % de entregas perfectamente recibidas e índice de servicio.					
EXCEPCIONES					
• Solo podrá acceder los administradores y usuarios registrados en el sistema de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.					

FECHA	ESTADO	COMENTARIO
01/09/2020	Definido	
02/09/2020	Hecho	
05/09/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 41: Reporte de Entregados Completos**

NÚMERO	41	DIFICULTAD	Alta	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá seleccionar dentro de un rango de fechas (inicio y fin) para emitir un reporte de los entregados completos en pdf de ese rango haciendo click a “Imprimir Reporte”					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente. • El dispositivo del administrador debe estar permitido el permiso de descarga de archivos.					

FECHA	ESTADO	COMENTARIO
01/09/2020	Definido	
02/09/2020	Hecho	
05/09/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 42 : Reporte de Índice de Servicio**

NÚMERO	42	DIFICULTAD	Alta	PRIORIDAD	Alta
DESCRIPCION					
El administrador podrá seleccionar dentro de un rango de fechas (inicio y fin) para emitir un reporte del índice de servicio en pdf de ese rango haciendo click a “Imprimir Reporte”					
EXCEPCIONES					
• Solo podrá acceder los administradores de CRISS NEÓN S.A.C. • Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente. • El dispositivo del administrador debe estar permitido el permiso de descarga de archivos.					

FECHA	ESTADO	COMENTARIO
01/09/2020	Definido	

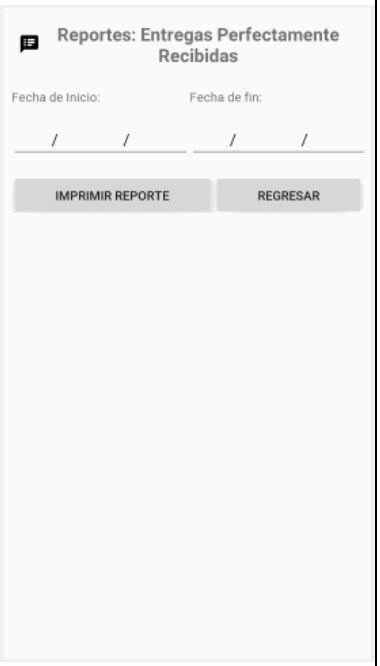
02/09/2020	Hecho
05/09/2020	Implementado
07/09/2020	Prueba
08/09/2020	Verificado

- **Storycard 43 : Reporte de Entregas Perfectamente Recibidas**

NÚMERO	**43**	**DIFICULTAD**	**Alta**	**PRIORIDAD**	**Alta**

DESCRIPCION

El administrador podrá seleccionar dentro de un rango de fechas (inicio y fin) para emitir un reporte de Entregas Perfectamente Recibidas en pdf de ese rango haciendo click a "Imprimir Reporte"



EXCEPCIONES

- Solo podrá acceder los administradores de CRISS NEÓN S.A.C.
- Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.
- El dispositivo del administrador debe estar permitido el permiso de descarga de archivos.

FECHA	**ESTADO**	**COMENTARIO**
01/09/2020	Definido	
02/09/2020	Hecho	
05/09/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 44: Listar Pedidos Asignados por Repartidor**

NÚMERO	44	DIFICULTAD	Alta	PRIORIDAD	Alta

DESCRIPCION

El repartidor podrá visualizar los reportes asignados por el administrador.

EXCEPCIONES

- Solo podrá acceder los repartidores de CRISS NEÓN S.A.C.
- Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.

FECHA	ESTADO	COMENTARIO
20/08/2020	Definido	
21/08/2020	Hecho	
28/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

- **Storycard 45: Geolocalización de la ruta actual del perfil del repartidor hacia el destinario del pedido**

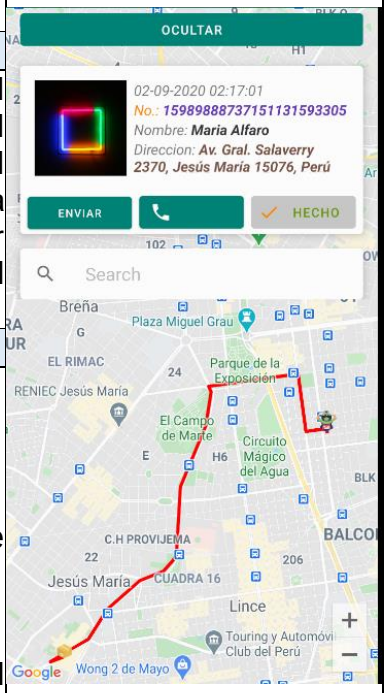
NÚMERO	45	DIFICULTAD	Alta	PRIORIDAD	Alta

DESCRIPCION

El repartidor podrá visualizar la ruta en línea roja hacia el destinatario del cliente, al momento de darle click en Enviar, el cliente y el administrador podrán visualizar la ruta acutal del repartidor, podrá llamar al repartidor y dar click en “Hecho” para concretar el pedido entregado, podrá buscar otra dirección por la caja de texto donde indica “Search” desde el punto actual del repartidor.

EXCEPCIONES

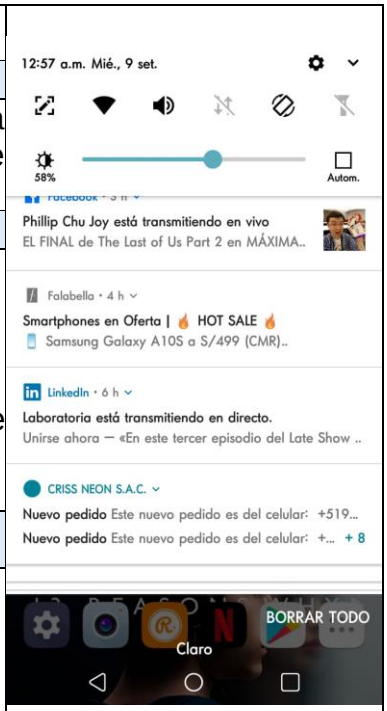
- Solo podrá acceder los repartidores registrados en el sistema de CRISS NEÓN S.A.C.
- Si el celular no tiene conexión a internet, el proceso de logueo no se realizará satisfactoriamente.
- El dispositivo móvil del administrador debe activar el permiso de ubicación para permitir al aplicativo usar el api de Google Maps.



• Debe estar activado el servicio de ubicación el dispositivo móvil del administrador.			
FECHA	**ESTADO**	**COMENTARIO**	
20/08/2020	Definido		
21/08/2020	Hecho		
28/08/2020	Implementado		
07/09/2020	Prueba		
08/09/2020	Verificado		

- **Storycard 46: Notificación de un nuevo pedido para el perfil del administrador por parte del perfil del cliente.**

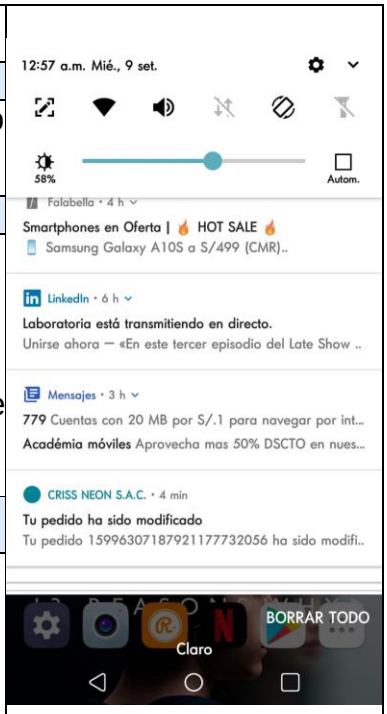
NÚMERO	**46**	**DIFICULTAD**	**Medio**	**PRIORIDAD**	**Alta**	
DESCRIPCION						
El administrador podrá visualizar notificaciones emitidas por la aplicación móvil al llegar un pedido y el número del cliente que lo solicita.						
EXCEPCIONES						
• Solo le llegará las notificaciones al administrador logueado en el sistema. • Si el celular no tiene conexión a internet, el envío de noticias no se efectuará.						
FECHA	**ESTADO**	**COMENTARIO**				
23/08/2020	Definido					
24/08/2020	Hecho					
31/08/2020	Implementado					
07/09/2020	Prueba					
08/09/2020	Verificado					



- **Storycard 47: Notificación de un nuevo pedido modificado por parte del administrador hacia el cliente**

NÚMERO	47	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El cliente podrá visualizar cuando su pedido ha sido asignado a un repartidor o ha sido cancelado por el número del pedido.					
EXCEPCIONES					
• Solo le llegará las notificaciones al cliente logueado en el sistema. • Si el celular no tiene conexión a internet, el envío de noticias no se efectuará.					

FECHA	ESTADO	COMENTARIO
23/08/2020	Definido	
24/08/2020	Hecho	
31/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	



- **Storycard 48: Notificación de pedido entregado satisfactoriamente en el perfil del cliente/usuario.**

NÚMERO	48	DIFICULTAD	Medio	PRIORIDAD	Alta
DESCRIPCION					
El cliente podrá visualizar cuando su pedido ha sido entregado de forma satisfactoria y el número del repartidor que lo ha entregado.					
EXCEPCIONES					
• Solo le llegará las notificaciones al cliente logueado en el sistema. • Si el celular no tiene conexión a internet, el envío de noticias no se efectuará.					

FECHA	ESTADO	COMENTARIO
23/08/2020	Definido	
24/08/2020	Hecho	
31/08/2020	Implementado	
07/09/2020	Prueba	
08/09/2020	Verificado	

FASE III: PRODUCCIÓN

- **Storycard 01 : Seleccionar Inicio de Sesión**

Interfaz	Descripción
	Se puede visualizar la interfaz para seleccionar el mode inicio de sesión ya sea por teléfono o correo electrónico.

```
MainActivity.java
103     private void init() {
104
105         Places.initialize( context: this,"AIzaSyA38X4HCspBdvWIMla87MVrhU-C46hVBvY");
106         placesClient = Places.createClient( context: this);
107         providers = Arrays.asList(new AuthUI.IdpConfig.PhoneBuilder().build(),
108                 new AuthUI.IdpConfig.EmailBuilder().build());
109         dialog = new SpotsDialog.Builder().setCancelable(false).setContext(this).build();
110
111
112
113         userRef = FirebaseDatabase.getInstance().getReference(Common.USER_REFERENCES);
114         firebaseAuth = FirebaseAuth.getInstance();
115
116         listener = firebaseAuth -> {
117
118             Dexter.withActivity(this)
119                     .withPermission(Manifest.permission.ACCESS_FINE_LOCATION)
120                     .withListener(new PermissionListener() {
121                         @Override
122                         public void onPermissionGranted(PermissionGrantedResponse response) {
123                             FirebaseUser user = firebaseAuth.getCurrentUser();
124                             if (user != null) {
125                                 //Toast.makeText(MainActivity.this, "Accediste al login", Toast.LENGTH_SHORT).show();
126                                 checkUserFromFirebase(user);
127                             } else {
128
129                                 phoneLogin();
130
131                             }
132                         }
133
134                         @Override
    MainActivity  showRegisterDialog()
```

En la sintaxis de código mostrada anteriormente, se usó las librerías firebase-auth:19.9.9 y firebase-ui-auth para la autentificación de usuarios y asu vez la misma base de datos NoSQL “Firebase”, extraiga módulo de registros por teléfono y correo electrónico.

- **Storycard 02 : Inicio de sesión por correo**

Interfaz	Descripción
	Se puede visualizar la interfaz para digitar el correo electrónico y de manera automática saldrá la contraseña para digitar y pasará al módulo de registro si en caso no está registrado.

```
MainActivity.java
145         };
146     }
147
148     private void checkUserFromFirebase(FirebaseUser user) {
149         dialog.show();
150         userRef.child(user.getUid())
151                 .addListenerForSingleValueEvent(new ValueEventListener() {
152                     @Override
153                     public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
154                         if (dataSnapshot.exists()) {
155                             Toast.makeText( context: MainActivity.this, text: "You already registed", Toast.LENGTH_SHORT).show();
156                             UserModel userModel = dataSnapshot.getValue(UserModel.class);
157                             goToHomeActivity(userModel);
158
159                         } else {
160                             showRegisterDialog(user);
161                         }
162                         dialog.dismiss();
163
164                     }
165
166                     @Override
167                     public void onCancelled(@NonNull DatabaseError databaseError) {
168                         dialog.dismiss();
169                         Toast.makeText( context: MainActivity.this, text: "" + databaseError.getMessage(), Toast.LENGTH_SHORT).show();
170
171                     }
172                 });
173
174
175     }
176
```

En la sintaxis de código mostrada anteriormente, se usó el método checkUserFromFirebase() instanciado con el modelo User para poder obtener mediante la obtención de datos que es el dataSnapshot para comprobar la existencia del usuario mediante el Firebase Database y Firebase Authetication.

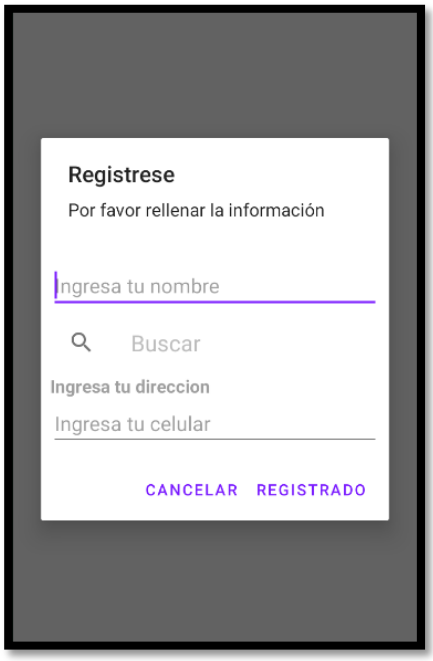
- **Storycard 03 : Inicio de sesión por teléfono**

Interfaz	Descripción
	Se puede visualizar la interfaz para digitar el número de teléfono y de manera inmediata el usuario al verificarlo, recibirá un SMS con su clave de 6 dígitos y pasara al módulo de registro si en caso no está registrado.

```
private void phoneLogin() {

    startActivityForResult(AuthUI.getInstance() AuthUI
                    .createSignInIntentBuilder() AuthUI.SignInIntentBuilder
            .setLogo(R.drawable.crissneon) SignInIntentBuilder
            .setTheme(R.style.LoginTheme) SignInIntentBuilder
                    .setAvailableProviders(providers).build(),
            APP_REQUEST_CODE);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == APP_REQUEST_CODE) {
        IdpResponse response = IdpResponse.fromResultIntent(data);
        if (resultCode == RESULT_OK) {
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        } else {
            Toast.makeText( context: this, text: "Fallado", Toast.LENGTH_SHORT).show();
        }
    }
```

En la sintaxis de código mostrada anteriormente, se usó el método phoneLogin(), se usó la librería AuthUI para la instancia de una librería en Firebase para la autenticación de números de celular mediante un requestCode que requerida el código para el registro de usuarios.

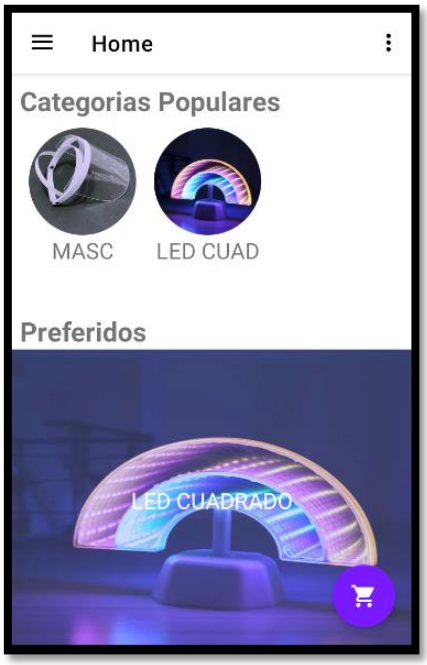
- **Storycard 04 : Registro de Usuarios**

Interfaz	Descripción
	Se puede visualizar dialogShow, para el registro de usuarios por su nombre, la dirección buscada por el servicio de Google Places Maps y el número de celular que previamente esta confirmado por el mismo usuario.

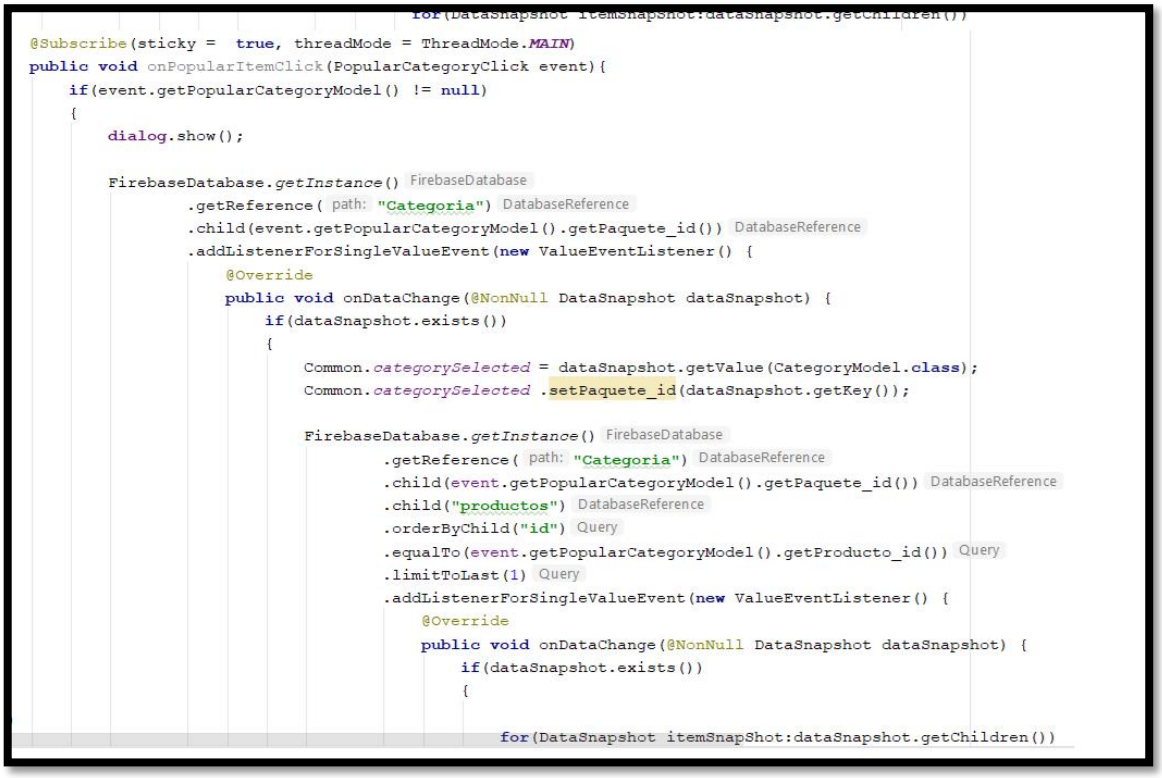
```
MainActivity.java | app

    }

    private void showRegisterDialog(FirebaseUser user) {
        androidx.appcompat.app.AlertDialog.Builder builder = new androidx.appcompat.app.AlertDialog.Builder( context: this);
        builder.setTitle("Registrese");
        builder.setMessage("Por favor rellenar la información");

        View itemView = LayoutInflater.from(this).inflate(R.layout.layout_register, root: null);

        EditText edt_name = (EditText) itemView.findViewById(R.id.edt_name);
        TextView txt_detalle_direccion = (TextView) itemView.findViewById(R.id.txt_detalle_direccion);
        EditText edt_phone = (EditText) itemView.findViewById(R.id.edt_phone);

        places_fragment = (AutocompleteSupportFragment)getSupportFragmentManager()
                .findFragmentById(R.id.places_autocomplete_fragment);
        places_fragment.setPlaceFields(placeFields);
        places_fragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
            @Override
            public void onPlaceSelected(@NonNull Place place) {
                placeSelected = place;
                txt_detalle_direccion.setText(place.getAddress());


            }

            @Override
            public void onError(@NonNull Status status) {
                Toast.makeText( context: MainActivity.this, text: ""+status.getStatusMessage(), Toast.LENGTH_SHORT).show();

            }
```

En la sintaxis de código mostrada anteriormente, se usó las librerías firebase-auth:19.9.9 y firebase-ui-auth para la autentificación de usuarios y asu vez la misma base de datos NoSQL “Firebase”, extraiga el módulo de registros por teléfono y correo electrónico.

- **Storycard 05 : Selección rápida del producto.**

Interfaz	Descripción
	Se puede visualizar la visualización de CircleImageViews e imagénes que al dar click se puede visualizar el producto a detalle.

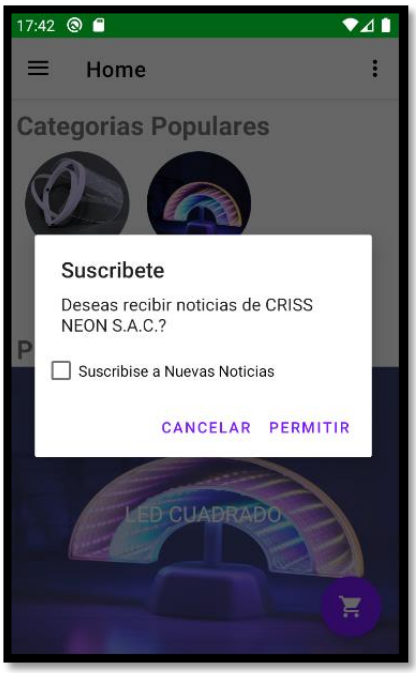
```
@Subscribe(sticky =  true, threadMode = ThreadMode.MAIN)
public void onBestDealItemClick(BestDealItemClick event){
    if(event.getBestDealModel() != null)
    {
        dialog.show();

        FirebaseDatabase.getInstance() FirebaseDatabase
                .getReference( path: "Categoria") DatabaseReference
                .child(event.getBestDealModel().getPaquete_id()) DatabaseReference
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                        if(dataSnapshot.exists())
                        {
                            Common.categorySelected = dataSnapshot.getValue(CategoryModel.class);
                            Common.categorySelected.setPaquete_id(dataSnapshot.getKey());

                            FirebaseDatabase.getInstance() FirebaseDatabase
                                    .getReference( path: "Categoria") DatabaseReference
                                    .child(event.getBestDealModel().getPaquete_id()) DatabaseReference
                                    .child("productos") DatabaseReference
                                    .orderByChild("id") Query
                                    .equalTo(event.getBestDealModel().getProducto_id()) Query
                                    .limitToLast(1) Query
                                    .addListenerForSingleValueEvent(new ValueEventListener() {
                                        @Override
                                        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                                            if(dataSnapshot.exists())
                                            {

                                                for(DataSnapshot itemSnapShot:dataSnapshot.getChildren())
```



```
                                    for(DataSnapshot itemSnapShot:dataSnapshot.getChildren())
@Subscribe(sticky =  true, threadMode = ThreadMode.MAIN)
public void onPopularItemClick(PopularCategoryClick event){
    if(event.getPopularCategoryModel() != null)
    {
        dialog.show();

        FirebaseDatabase.getInstance() FirebaseDatabase
                .getReference( path: "Categoria") DatabaseReference
                .child(event.getPopularCategoryModel().getPaquete_id()) DatabaseReference
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                        if(dataSnapshot.exists())
                        {
                            Common.categorySelected = dataSnapshot.getValue(CategoryModel.class);
                            Common.categorySelected .setPaquete_id(dataSnapshot.getKey());

                            FirebaseDatabase.getInstance() FirebaseDatabase
                                    .getReference( path: "Categoria") DatabaseReference
                                    .child(event.getPopularCategoryModel().getPaquete_id()) DatabaseReference
                                    .child("productos") DatabaseReference
                                    .orderByChild("id") Query
                                    .equalTo(event.getPopularCategoryModel().getProducto_id()) Query
                                    .limitToLast(1) Query
                                    .addListenerForSingleValueEvent(new ValueEventListener() {
                                        @Override
                                        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                                            if(dataSnapshot.exists())
                                            {

                                                for(DataSnapshot itemSnapShot:dataSnapshot.getChildren())
```

En las dos sintaxis de código mostradas anteriormente, se usó FirebaseDatabase para la consulta de la tabla BestDeal y PopularDeal para la consulta de los mejores productos solicitados por clientes en la empresa a su vez al hacer click nos trae defrente al producto eso es posible a que el Common hace una instancia con la tabla de Categoria y productos para que nos pueda traer por id el producto que se selecciona mediante un arreglo “for” que usa el DataSnapshot para la devolución de datos mediante una llamada de instancia de la consulta NoSQL.

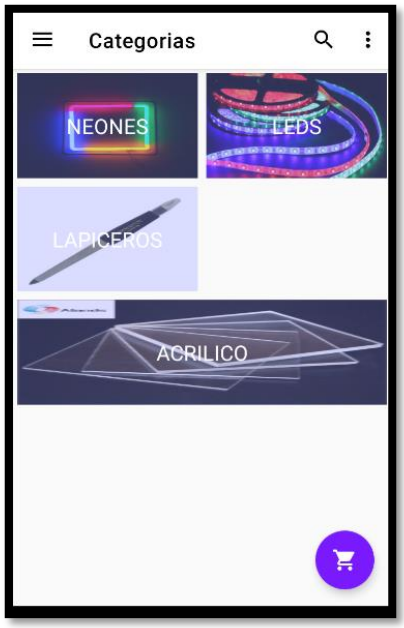
- **Storycard 06 : Subscripción a noticias.**

Interfaz	Descripción
	Se puede visualizar dialogShow, para el registro de usuarios por su nombre, la dirección buscada por el servicio de Google Places Maps y el número de celular que previamente esta confirmado por el mismo usuario.

```
private void showSuscribirseNoticias() {
    Paper.init( context: this);
    androidx.appcompat.app.AlertDialog.Builder builder = new androidx.appcompat.app.AlertDialog.Builder( context: this);
    builder.setTitle("Suscribete");
    builder.setMessage("Deseas recibir noticias de CRISS NEON S.A.C.?");

    View itemView = LayoutInflater.from(this).inflate(R.layout.layout_suscribirse_noticias, root: null);
    CheckBox ckb_noticias = (CheckBox)itemView.findViewById(R.id.ckb_suscripcion_noticias);
    boolean isSuscribirseNoticias = Paper.book().read(Common.IS_SUBSCRIBE_NEWS, defaultValue: false);
    if(isSuscribirseNoticias)
        ckb_noticias.setChecked(true);
    builder.setNegativeButton( text: "CANCELAR", (dialogInterface, i) -> {
        dialogInterface.dismiss();
    }).setPositiveButton( text: "PERMITIR", (dialogInterface, i) -> {
        if(ckb_noticias.isChecked())
        {
            Paper.book().write(Common.IS_SUBSCRIBE_NEWS,true);
            FirebaseMessaging.getInstance()
                    .subscribeToTopic(Common.NEWS_TOPIC)
                    .addOnFailureListener(e -> Toast.makeText( context: HomeActivity.this,e.getMessage(), Toast.LENGTH_SHORT).show()).addOnSuccessLis

        }
        else
        {
            Paper.book().delete(Common.IS_SUBSCRIBE_NEWS);
            FirebaseMessaging.getInstance()
                    .unsubscribeFromTopic(Common.NEWS_TOPIC)
                    .addOnFailureListener(e -> Toast.makeText( context: HomeActivity.this,e.getMessage(), Toast.LENGTH_SHORT).show()).addOnSuccessLis
        }

    });
    builder.setView(itemView);
    AlertDialog dialog = builder.create();
    dialog.show();


}
```

En la sintaxis de código mostrada anteriormente, se usó la librería Paper.book que va sirve como un motor de almacenamiento para las noticias guardadas en notificaciones del cliente y el Firebase Messaging para que se transmita mediante la nube las noticias, al hacer check permite la transmisión de noticias pero si deja de marcar el check deja de recibir las noticias mediante el método unsubscribeFromTopic.

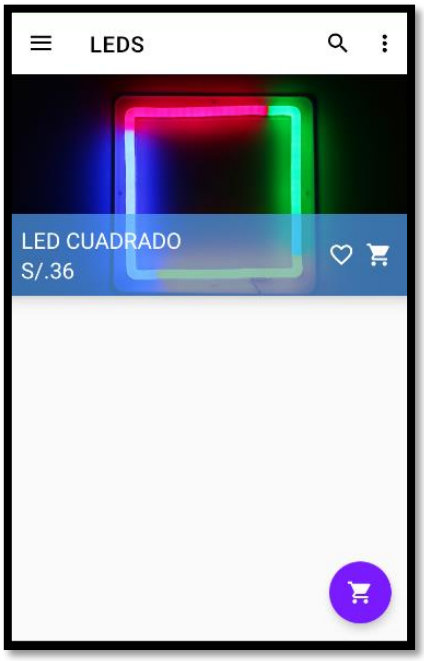
- **Storycard 07 : Listar Categorías.**

Interfaz	Descripción
	Se puede visualizar la lista de categorías registradas en el firebase con su nombre e imagen y en la parte superior visualizamos una lupa que servirá para buscar la categoría deseada.


```
public void loadCategories(){
      List<CategoryModel> tempList = new ArrayList<>();
    DatabaseReference categoryRef = FirebaseDatabase.getInstance().getReference(Common.CATEGORY_REF);
    categoryRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for(DataSnapshot itemSnapShot:dataSnapshot.getChildren())
            {
                CategoryModel categoryModel = itemSnapShot.getValue(CategoryModel.class);
                categoryModel.setPaquete_id(itemSnapShot.getKey());
                tempList.add(categoryModel);

            }
              categoryCallbackListener.onCategoryDealLoadSucess(tempList);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            categoryCallbackListener.onCategoryDealLoadFailed(databaseError.getMessage());

        }
    });
}
  public MutableLiveData<String> getMessageError() { return messageError; }

  @Override
  public void onCategoryDealLoadSucess(List<CategoryModel> categoryModelList) {
      categoryListMultable.setValue(categoryModelList);

  }
```

En la sintaxis de código mostrada anteriormente, se usó el método loadCategories que usa una list para usar un arreglo que listen mediante el DatabaseReference para las categorías mediante el setPaquete_id para obtenerlo que lo carga en conjunto con el método de categoryListMutable y el dataSnapshot obtendrá los datos del firebase.

- **Storycard 08 : Listar Productos por Categoría.**

Interfaz	Descripción
	Se puede visualizar la lista de productos por categoría, al darle click al carrito por cada producto se añadirá al carrito actual, al dar click al producto se visualizará con mayor detalle el producto con los tamaños y complementos y si se le da click al botón de carrito morado se puede visualizar el carrito de compras.

```
public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
    Glide.with(context).load(productoModelList.get(position).getImage()).into(holder.img_producto_image);
    holder.txt_producto_price.setText(new StringBuilder("S/.")
    .append(productoModelList.get(position).getPrice()));
    holder.txt_producto_name.setText(new StringBuilder("")
    .append(productoModelList.get(position).getName()));

    holder.setListener((view, pos) -> {
        Common.selectedProducto = productoModelList.get(pos);
        Common.selectedProducto.setKey(String.valueOf(pos));
        EventBus.getDefault().postSticky(new ProductoItemClick( success: true, productoModelList.get(pos)));
    });

    holder.img_cart.setOnClickListener(v -> {
        CartItem cartItem = new CartItem();
        cartItem.setUid(Common.currentUser.getUid());
        cartItem.setUserPhone(Common.currentUser.getPhone());

        cartItem.setProductoId(productoModelList.get(position).getId());
        cartItem.setProductoName(productoModelList.get(position).getName());
        cartItem.setProductoImage(productoModelList.get(position).getImage());
        cartItem.setProductoPrice(Double.valueOf(String.valueOf(productoModelList.get(position).getPrice())));
        cartItem.setProductoQuantity(1);
        cartItem.setProductoExtraPrice(0.0);
        cartItem.setProductoAddon("Default");
        cartItem.setProductoSize("Default");

        cartDataSource.getItemWithAllOptionsInCart(Common.currentUser.getUid(),
                cartItem.getProductoId(),
                cartItem.getProductoSize(),
                cartItem.getProductoAddon())
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new SingleObserver<CartItem>() {
                    @Override
                    public void onSubscribe(Disposable d) {
```

```
        ImageView closeButton = (ImageView)searchView.findViewById(R.id.search_close_btn);
        closeButton.setOnClickListener(view -> {
            EditText ed = (EditText)searchView.findViewById(R.id.search_src_text);

            ed.setText("");

            searchView.setQuery( query: "", submit: false);

            searchView.onActionViewCollapsed();

            menuItem.collapseActionView();

            productoListViewModel.getMutableLiveDataProductoList();


        });


    }

    private void startBuscar(String s) {
        List<ProductoModel> resultList = new ArrayList<>();
        for(int i=0;i<Common.categorySelected.getProductos().size();i++)
        {
            ProductoModel productoModel = Common.categorySelected.getProductos().get(i);
            if(productoModel.getName().toLowerCase().contains(s))
                resultList.add(productoModel);

        }
        productoListViewModel.getMutableLiveDataProductoList().setValue(resultList);
    }

    @Override
    public void onDestroy() {
        EventBus.getDefault().postSticky(new MenuItemBack());
        super.onDestroy();
    }
```

En las dos sintaxis de código mostrada anteriormente, se usó el método onBindViewHolder en el controlador para traer una lista de productos por sus atributos y para la posición correcta de datos se usa el position, para buscar el producto se usa el startBuscar que busca el producto mediante un “for” por un arreglo que obtiene una lista de productos.

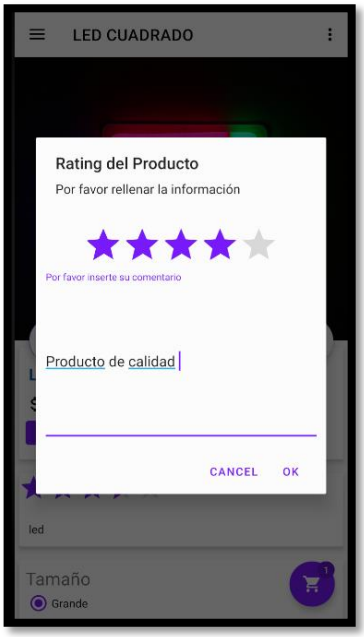
- **Storycard 09 : Detalle del producto.**

Interfaz	Descripción
	Se puede visualizar el detalle del producto, con el precio del producto, al dar click al botón de estrella se va observar una ventana emergente para calificar el producto, y el carrito de icono negro para comprarlo, con los tamaños y complementos.

```
ProductoListFragment.java | MyProductoListAdapter.java

public void onSuccess(CartItem cartItemFromDB) {
    if(cartItemFromDB.equals(cartItem))
    {
     //database
        cartItemFromDB.setProductoExtraPrice(cartItem.getProductoExtraPrice());
        cartItemFromDB.setProductoAddon(cartItem.getProductoAddon());
        cartItemFromDB.setProductoSize(cartItem.getProductoSize());
        cartItemFromDB.setProductoQuantity(cartItemFromDB.getProductoQuantity() + cartItem.getProductoQuantity());

        cartDataSource.updateCartItems(cartItemFromDB)
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new SingleObserver<Integer>() {
                    @Override
                    public void onSubscribe(Disposable d) {

                    }

                    @Override
                    public void onSuccess(Integer integer) {
                        Toast.makeText(context, text: "Carrito modificado satisfactoriamente", Toast.LENGTH_SHORT).show();
                        EventBus.getDefault().postSticky(new CounterCartEvent( success: true));

                    }

                    @Override
                    public void onError(Throwable e) {
                        Toast.makeText(context, text: "[UPDATE CART]"+e.getMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
    }
    else{
        compositeDisposable.add(cartDataSource.insertOrReplaceAll(cartItem)
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(() -> {
                    Toast.makeText(context, text: "Añadido al carrito de forma satisfactoria", Toast.LENGTH_SHORT).show();
                    EventBus.getDefault().postSticky(new CounterCartEvent( success: true));
```

En la sintaxis de código mostrada anteriormente, se usó el método onSucess que instancia de la base de datos temporal SQLite para que almacene mediante el símbolo del carrito el producto que se seleccione y muestra el producto a detalle mediante una instancia del DatabaseReference para mostrar los productos detallados por tamaño, complemento, precio, producto listado por un array.

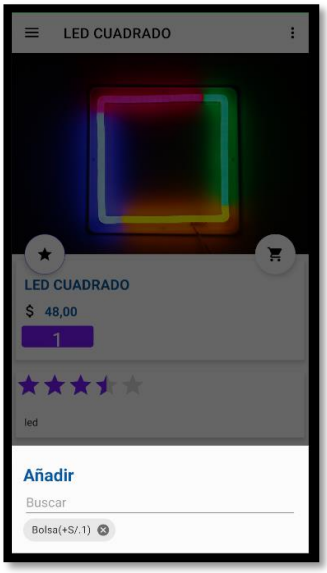
- **Storycard 10 : Ranking del Producto.**

Interfaz	Descripción
	Se puede observar un dialogShow al hacer click al icono de estrella para poder calificar el producto según un una calificación de máximo 5 estrellas y un comentario.

```
ProductoDetailFragment.java
370    private void addRatingToProducto(float ratingValue) {
371        FirebaseDatabase.getInstance() FirebaseDatabase
372                .getReference(Common.CATEGORY_REF) DatabaseReference
373                .child(Common.categorySelected.getPaquete_id()) DatabaseReference
374                .child("productos") DatabaseReference
375                .child(Common.selectedProducto.getKey()) DatabaseReference
376                .addListenerForSingleValueEvent(new ValueEventListener() {
377                    @Override
378                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
379                        if(dataSnapshot.exists())
380                        {
381                            ProductoModel productoModel = dataSnapshot.getValue(ProductoModel.class);
382                            productoModel.setKey(Common.selectedProducto.getKey());
383
384                            if(productoModel.getRatingValue() == null )
385                                productoModel.setRatingValue(0d);
386                            if(productoModel.getRatingCount() == null)
387                                productoModel.setRatingCount(0l);
388                            double sumRating = productoModel.getRatingValue()+ratingValue;
389                            long ratingCount = productoModel.getRatingCount()+1;
390
391                            Map<String,Object> updateData = new HashMap<>();
392                            updateData.put("ratingValue",sumRating);
393                            updateData.put("ratingCount",ratingCount);
394
395                            productoModel.setRatingValue(sumRating);
396                            productoModel.setRatingCount(ratingCount);
397                            dataSnapshot.getRef()
398                                    .updateChildren(updateData)
399                                    .addOnCompleteListener(task -> {
400                                        waitingDialog.dismiss();
401                                        if(task.isSuccessful())
402                                        {
403                                            Toast.makeText(getContext(), text: "Gracias", Toast.LENGTH_SHORT).show();
404                                            Common.selectedProducto = productoModel;
405                                            productoDetailViewModel.setProductoModel(productoModel);
406                                        }
407                                    });
```

En la sintaxis de código mostrada anteriormente, se usó el método addRatingToProducto va obtener de la información de los productos mediante un dataSnapshot , para permitir agregar una nueva tabla sobre los comentarios y puntuación de cada producto, sumando la cantidad de estrellas y guardando el comentario del producto por el set.

- **Storycard 11 : Añadidura del complemento por producto.**

Interfaz	Descripción
	Se puede observar un dialogShow para poder añadir los complementos o buscarlo para poder añadir por producto y eso incrementará al precio del producto.

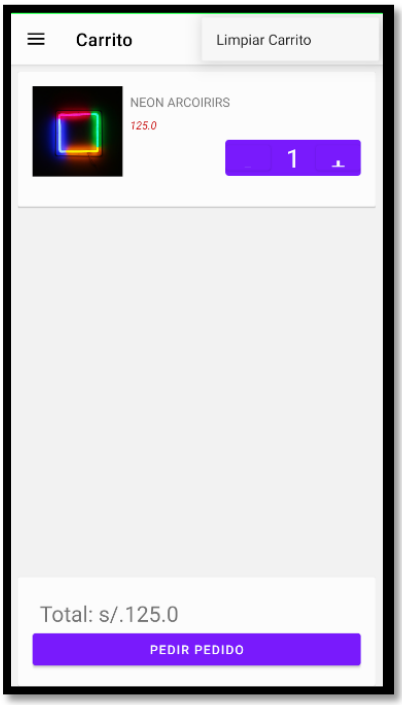
ProductoDetailFragment.java

```
328
329     private void displayUserSelectedAddon() {
330         if(Common.selectedProducto.getUserSelectedAddon() != null &&
331         Common.selectedProducto.getUserSelectedAddon().size()>0)
332         {
333             chip_group_user_selected_addon.removeAllViews();
334             for(AddonModel addonModel : Common.selectedProducto.getUserSelectedAddon())
335             {
336                 Chip chip = (Chip)getLayoutInflater().inflate(R.layout.layout_chip_with_delete_icon,  root: null);
337                 chip.setText(new StringBuilder(addonModel.getName()).append("(+S/.")
338                 .append(addonModel.getPrice()).append(")"));
339                 chip.setClickable(false);
340                 chip.setOnCloseIconClickListener(view -> {
341                     chip_group_user_selected_addon.removeView(view);
342                     Common.selectedProducto.getUserSelectedAddon().remove(addonModel);
343                     calculateTotalPrice();
344
345                 });
346                 chip_group_user_selected_addon.addView(chip);
347             }
348         }else
349             chip_group_user_selected_addon.removeAllViews();
350     }
351
352     private void submitRatingToFirebase(CommentModel commentModel) {
353         waitingDialog.show();
354         FirebaseDatabase.getInstance() FirebaseDatabase
355                 .getReference(Common.COMMENT_REF) DatabaseReference
356                 .child(Common.selectedProducto.getId()) DatabaseReference
357                 .push() DatabaseReference
358                 .setValue(commentModel) Task<Void>
359                 .addOnCompleteListener(task -> {
```

ProductoDetailFragment › btnCart

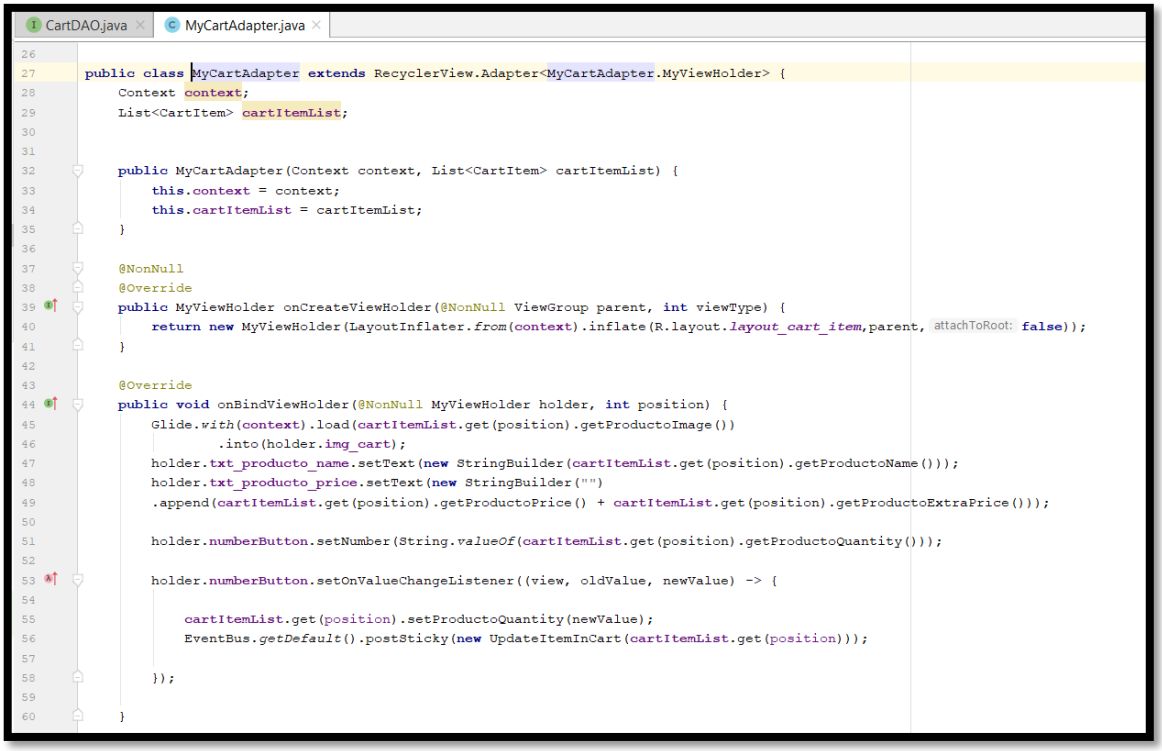
En la sintaxis de código mostrada anteriormente, se usó el método displaySelectedProducto va obtener de la información de los complementos productos mediante un dataSnapshot , para permitir agregar una un nuevo complemento por producto mediante un método repetitivo en incrementar el precio del producto.

- **Storycard 12 : Carrito de Compras**

Interfaz	Descripción
	Se puede observar la lista de productos en el carrito de compras, si se desglosa en la parte derecha se puede eliminar el producto o también agregar la cantidad de productos a solicitar y en la parte superior observamos un botón para limpiar el carrito y al dar click al botón “PEDIR PEDIDO”, saldrá un dialogShow para solicitar el pedido.

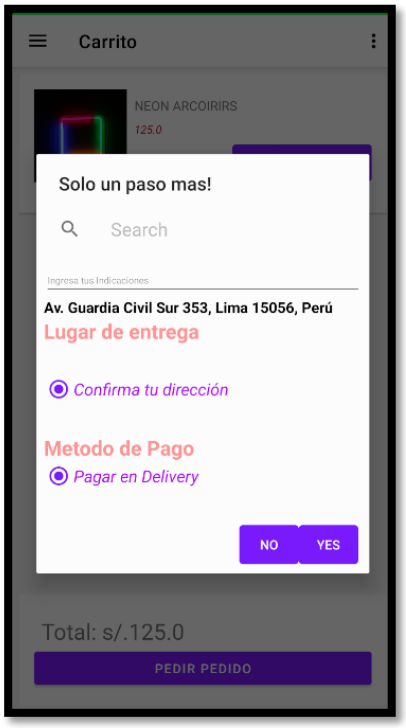
```
CartDAO.java

@Dao
public interface CartDAO {
    @Query("SELECT * FROM Cart WHERE uid=:uid")
    Flowable<List<CartItem>> getAllCart(String uid);

    @Query("SELECT SUM(productoQuantity) from Cart WHERE uid=:uid")
    Single<Integer> countItemInCart(String uid);

    @Query("SELECT SUM((productoPrice+productoExtraPrice) * productoQuantity) FROM Cart WHERE uid=:uid")
    Single<Double> sumPriceInCart(String uid);

    @Query("SELECT * FROM Cart WHERE productoId=:productoId AND uid=:uid")
    Single<CartItem> getItemInCart(String productoId, String uid);

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    Completable insertOrReplaceAll(CartItem... cartItems);

    @Update(onConflict = OnConflictStrategy.REPLACE)
    Single<Integer> updateCartItems(CartItem cartItems);

    @Delete
    Single<Integer> deleteCartItem(CartItem cartItem);

    @Query("DELETE FROM Cart WHERE uid=:uid")
    Single<Integer> cleanCart(String uid);

    @Query("SELECT * FROM Cart WHERE productoId=:productoId AND uid=:uid AND productoSize=:productoSize AND productoAddon =:productoAddon")
    Single<CartItem> getItemWithAllOptionsInCart(String uid, String  productoId, String productoSize, String productoAddon);


}
```

En la primera sintaxis de código mostrada anteriormente, se puede visualizar la interface clase CartDAO para poder hacer las consultas en SQLite almacenadas en el carrito de compras y se refleja la construcción del carrito de compras almacenado en el celular.



En la segunda sintaxis de código mostrada anteriormente, se puede visualizar el controlador del Carrito para que se muestre lo almacenado en el carrito de compras mediante position para posicionar los productos listados y mediante el método remove se removerá si en caso el cliente quiere limpiar todo o un producto del carrito de la librería EventBus.

- **Storycard 13 : Solicitud del Pedido**

Interfaz	Descripción
	Se puede observar un dialogShow que nos permitirá solicitar el pedido presentando la dirección registrada y a su vez confirmar y buscar la dirección exacta del pedido gracias al Google Places Maps y dar las indicaciones necesarias, y se le da click a Yes para continuar el pedido y No para cancelar la solicitud del pedido.

```
CartFragment.java | MyCartAdapter.java

        return root;
    }

    private void initLocation() {
        buildLocationRequest();
        buidLocationCallback();
        fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(getContext());
        fusedLocationProviderClient.requestLocationUpdates(locationRequest,locationCallback, Looper.getMainLooper());
    }

    private void buidLocationCallback() {
        locationCallback = (LocationCallback) onLocationResult(locationResult) → {
                super.onLocationResult(locationResult);
                currentLocation = locationResult.getLastLocation();
        };
    }

    private void buildLocationRequest() {
        locationRequest = new LocationRequest();
        locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        locationRequest.setInterval(5000);
        locationRequest.setFastestInterval(3000);
        locationRequest.setSmallestDisplacement(10f);
    }
       @Override
       public void onLoadTimeSuccess(Pedido pedido, long estimateTimeInMs) {
           pedido.setCreateDate(estimateTimeInMs);
           pedido.setPedidoEstado(3);
           writePedidoToFirebase(pedido);
       }

       @Override
       public void onLoadTimeFailed(String message) {
           Toast.makeText(getContext(), text: ""+message, Toast.LENGTH_SHORT).show();

       }
```

```
private void pagarCOD(String direccion, String comment, String fecha) {
    compositeDisposable.add(cartDataSource.getAllCart(Common.currentUser.getUid())
    .subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .subscribe(cartItems -> {

        cartDataSource.sumPriceInCart(Common.currentUser.getUid())
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new SingleObserver<Double>() {
                    @Override
                    public void onSubscribe(Disposable d) {

                    }

                    @Override
                    public void onSuccess(Double precioTotal) {
                        double precioFinal = precioTotal;
                        Pedido pedido = new Pedido();
                        pedido.setUserId(Common.currentUser.getUid());
                        pedido.setUserName(Common.currentUser.getName());
                        pedido.setUserPhone(Common.currentUser.getPhone());
                        pedido.setDireccionEnvio(direccion);
                        pedido.setFecha(fecha);
                        pedido.setComment(comment);
                        //placeSelected.getLatLng();

                        if(currentLocation != null)
                        {
                            pedido.setLat(placeSelected.getLatLng().latitude);
                            pedido.setLng(placeSelected.getLatLng().longitude);

                        }else{


                            pedido.setLat(currentLocation.getLatitude());
                            pedido.setLng(currentLocation.getLongitude());

                            //pedido.setLat(-0.1f);
                            //pedido.setLng(-0.1f);
                        }
                        pedido.setCartItemList(cartItems);
                        pedido.setTotalPago(precioTotal);
                        pedido.setDescuento(0);
                        pedido.setFinalPago(precioFinal);
                        pedido.setCod(true);
                        pedido.setTransaccionId("Pagar en Delivery");

                        //writePedidoToFirebase(pedido);

                        syncLocalTimeWithGlobaltime(pedido);


                    }
```

En las dos sintaxis de código mostradas anteriormente se usó el método pagarCOD se va obtener la información almacenada en el SQLite del carrito de compras que va extraer el usuario, dirección, fecha, indicaciones, longitud y latitud de la dirección por el servicio de Google Maps, en el método initLocation va ayudar a obtener la dirección confirmada por el usuario, mientras que el método onLoadTimeSucess almacena el pedido en la base de datos Firebase otorgándole un estado de “3” que es de Atendido o Recibido.

- **Storycard 14 : Listar Pedidos**

Interfaz	Descripción
	Se puede visualizar en la interfaz que se tiene la lista de pedidos implementado con recycler view para listar los productos obtenidos por cada usuario, visualizamos 3 opciones distintas para el pedido ya que MySwipeHelper es una clase que va generar diferentes menús de tipo desplegables para mayor interacción con el usuario.


```
private void loadPedidosFromFirebase() {
    List<Pedido> pedidoList = new ArrayList<>();
    FirebaseDatabase.getInstance().getReference(Common.PEDIDO_REF) DatabaseReference
            .orderByChild("userId") Query
            .equalTo(Common.currentUser.getUid()) Query
            .limitToLast(100) Query
            .addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    for(DataSnapshot pedidoSnapShot:dataSnapshot.getChildren()){
                        Pedido pedido = pedidoSnapShot.getValue(Pedido.class);
                        pedido.setPedidoNumero(pedidoSnapShot.getKey());
                        pedidoList.add(pedido);
                    }
                    listener.onLoadPedidoSuccess(pedidoList);

                }

                @Override
                public void onCancelled(@NonNull DatabaseError databaseError) {
                    listener.onLoadOrderFailed(databaseError.getMessage());

                }
            });
```

En la primera sintaxis de código mostrada anteriormente,se tiene el método loadPedidosFromFirebase(), que nos ayudará u obtener una lista de pedidos mediante un arrayList obtenidos por el usuario que obtendrá la data mediante DataSnapshot.

```
ViewPedidosFragment.java

buf.add(new MyButton(getContext(), text: "Ver Trayecto", textSize: 30, imageResId: 0, Color.parseColor( colorString: "#001970"),
        pos -> {
            Pedido pedido = ((MyPedidosAdapter)recycler_pedidos.getAdapter()).getItemAtPosition(pos);

            //fetch from firebase
            FirebaseDatabase.getInstance() FirebaseDatabase
                    .getReference(Common.CAMINO_PEDIDO_REF) DatabaseReference
                    .child(pedido.getPedidoNumero()) DatabaseReference
                    .addListenerForSingleValueEvent(new ValueEventListener() {
                        @Override
                        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                            if(dataSnapshot.exists())
                            {
                                Common.currentCaminoPedido = dataSnapshot.getValue(CaminoPedidoModel.class);
                                Common.currentCaminoPedido.setKey(dataSnapshot.getKey());
                                if(Common.currentCaminoPedido.getCurrentLat() != -1 &&
                                Common.currentCaminoPedido.getCurrentLng() != -1)
                                {
                                    startActivity(new Intent(getContext(), TrayectoPedidoActivity.class));

                                }else {
                                    Toast.makeText(getContext(), text: "Su pedido aun no esta en camino, espere por favor que se le asigne un repart
                                }

                            }else{
                                Toast.makeText(getContext(), text: "Tu pedido esta en proceso, por favor espera que este en camino", Toast.LENGTH_SH
                            }

                        }

                        @Override
```

En la segunda sintaxis de código mostrada anteriormente, mediante el MySwipeHelper que sirve como un selector o menú desplegable, se va instanciar de FirebaseDatabase para localizar la latitud y longitud de la tabla CaminoPedido por usuario y repartidor.

```
ViewPedidosFragment.java

        }));

buf.add(new MyButton(getContext(), text: "Repetir Pedido", textSize: 30, imageResId: 0, Color.parseColor( colorString: "#5d4037"),
        pos -> {
            Pedido pedido = ((MyPedidosAdapter)recycler_pedidos.getAdapter()).getItemAtPosition(pos);

            //fetch from firebase
            dialog.show();
            cartDataSource.cleanCart(Common.currentUser.getUid()) //limpiar la carta
            .subscribeOn(Schedulers.io())
                    .observeOn(AndroidSchedulers.mainThread())
                    .subscribe(new SingleObserver<Integer>() {
                        @Override
                        public void onSubscribe(Disposable d) {

                        }

                        @Override
                        public void onSuccess(Integer integer) {
                            CartItem[] cartItems = pedido
                                    .getCartItemList().toArray(new CartItem[pedido.getCartItemList().size()]);

                            compositeDisposable.add(cartDataSource.insertOrReplaceAll(cartItems)
                                    .subscribeOn(Schedulers.io())
                                    .observeOn(AndroidSchedulers.mainThread())
                                    .subscribe(() -> {
                                        dialog.dismiss();
                                        Toast.makeText(getContext(), text: "Anadido nuevamente a  la lista de pedidos", Toast.LENGTH_SHORT).s
                                            EventBus.getDefault().postSticky(new CounterCartEvent( success: true));
                                    },throwable -> {
                                        dialog.dismiss();
                                        Toast.makeText(getContext(), text: ""+throwable.getMessage(), Toast.LENGTH_SHORT).show();
```

En la tercera sintaxis de código mostrada anteriormente mediante el MySwipeHelper que sirve como un selector o menú desplegable, va permitir al usuario o cliente repetir el pedido mediante la tabla de Pedidos almacenándolo mediante un array a la base de datos temporal SQLite.

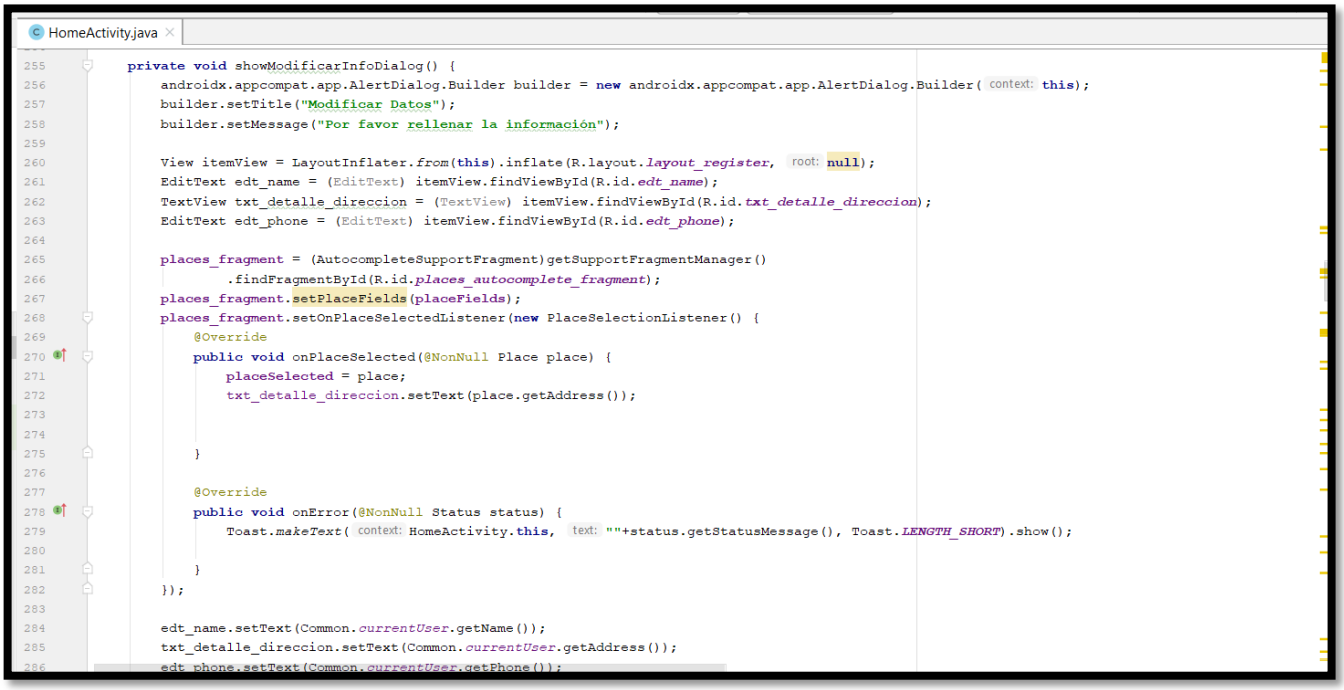
MyPedidosAdapter.java

```
    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        Glide.with(context).load(pedidoList.get(position).getCartItemList().get(0).getProductoImage())
                .into(holder.img_pedido);
        calendar.setTimeInMillis(pedidoList.get(position).getCreateDate());
        Date date = new Date(pedidoList.get(position).getCreateDate());
        holder.txt_pedido_fecha.setText(new StringBuilder(Common.getDateOfWeek(calendar.get(Calendar.DAY_OF_WEEK)))
                .append(" ")
                .append(simpleDateFormat.format(date)));
        holder.txt_numero_pedido.setText(new StringBuilder("Pedido No: ").append(pedidoList.get(position).getPedidoNumero()));
        holder.txt_cometario_pedido.setText(new StringBuilder("Comentario: ").append(pedidoList.get(position).getComment()));
        holder.txt_pedido_estado.setText(new StringBuilder("Estado: ").append(Common.convertEstadoToText(pedidoList.get(position).getPedidoEstado())));

        holder.setRecyclerClickListener((view, pos) -> {
            showDialog(pedidoList.get(pos).getCartItemList());
        });

    }

    private void showDialog(List<CartItem> cartItemList) {
        View layout_dialog = LayoutInflater.from(context).inflate(R.layout.layout_dialog_pedido_detalle, root: null);
        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        builder.setView(layout_dialog);

        Button btn_ok = (Button)layout_dialog.findViewById(R.id.btn_ok);
        RecyclerView recycler_pedido_detalle = (RecyclerView)layout_dialog.findViewById(R.id.recycler_pedido_detalle);
        recycler_pedido_detalle.setHasFixedSize(true);
        LinearLayoutManager layoutManager = new LinearLayoutManager(context);
        recycler_pedido_detalle.setLayoutManager(layoutManager);
        recycler_pedido_detalle.addItemDecoration(new DividerItemDecoration(context,layoutManager.getOrientation()));

        MyPedidoDetalleControlador myPedidoDetalleControlador = new MyPedidoDetalleControlador(context,cartItemList);
```

En la cuarta sintaxis de código mostrada anteriormente, se va poder observar la obtención de la lista de pedidos mediante holders que obtienen del Firebase e interactúa con la vista del usario.

- **Storycard 15 : Edición de datos del Cliente**

Interfaz	Descripción
	Se puede visualizar en la interfaz que se tiene un dialogShow para la edición de usuarios trayendo los datos actuales del usuario, pudiendo editar la dirección por Google places y su nombre.

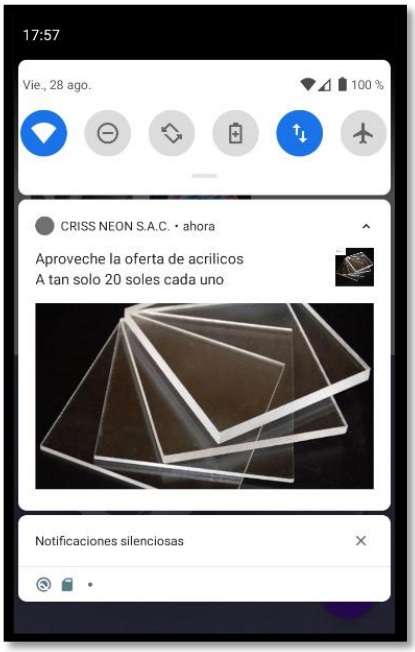


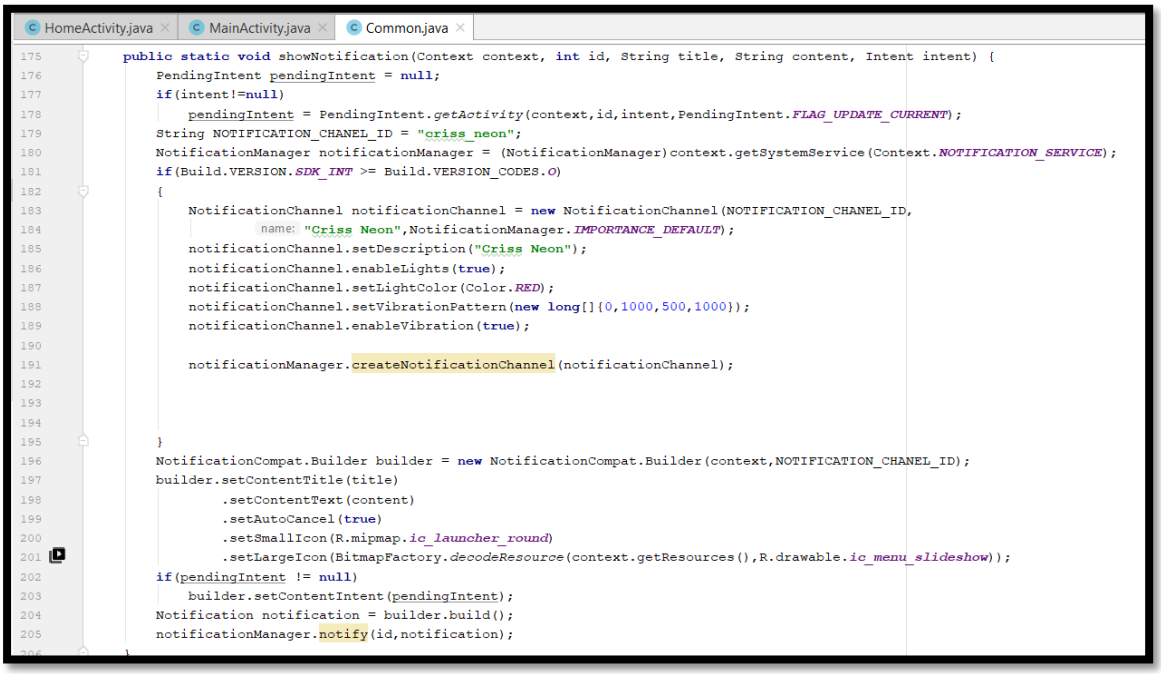
```
HomeActivity.java

private void showModificarInfoDialog() {
    androidx.appcompat.app.AlertDialog.Builder builder = new androidx.appcompat.app.AlertDialog.Builder( context: this);
    builder.setTitle("Modificar Datos");
    builder.setMessage("Por favor rellenar la información");

    View itemView = LayoutInflater.from(this).inflate(R.layout.layout_register, root: null);
    EditText edt_name = (EditText) itemView.findViewById(R.id.edt_name);
    TextView txt_detalle_direccion = (TextView) itemView.findViewById(R.id.txt_detalle_direccion);
    EditText edt_phone = (EditText) itemView.findViewById(R.id.edt_phone);

    places_fragment = (AutocompleteSupportFragment)getSupportFragmentManager()
            .findFragmentById(R.id.places_autocomplete_fragment);
    places_fragment.setPlaceFields(placeFields);
    places_fragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
        @Override
        public void onPlaceSelected(@NonNull Place place) {
            placeSelected = place;
            txt_detalle_direccion.setText(place.getAddress());


        }

        @Override
        public void onError(@NonNull Status status) {
            Toast.makeText( context: HomeActivity.this, text: ""+status.getStatusMessage(), Toast.LENGTH_SHORT).show();

        }
    });

    edt_name.setText(Common.currentUser.getName());
    txt_detalle_direccion.setText(Common.currentUser.getAddress());
    edt_phone.setText(Common.currentUser.getPhone());
```

En la primera sintaxis de código mostrada anteriormente, se muestra la codificación para traer el dialog show de la edición de datos mediante métodos get para obtener entre el método Common y get la obtención de datos actuales del usuario.

En la segunda sintaxis de código mostrada anteriormente, se muestra la codificación para modificar la información del usuario mediante la data nueva modificada en el firebase instanciándolo mediante servicios como Google Places por la latitud y longitud de la dirección.

- **Storycard 16 : Mostrar noticias**

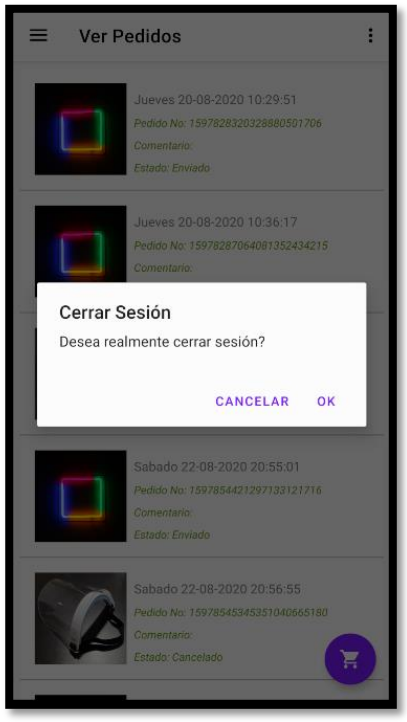
Interfaz	Descripción
	Se puede visualizar la transmisión de noticias ya permitido la subscripción de noticias por el usuario dando como cuerpo la foto, título y contexto.



```
HomeActivity.java | MainActivity.java | Common.java

public static void showNotification(Context context, int id, String title, String content, Intent intent) {
    PendingIntent pendingIntent = null;
    if(intent!=null)
        pendingIntent = PendingIntent.getActivity(context,id,intent,PendingIntent.FLAG_UPDATE_CURRENT);
    String NOTIFICATION_CHANEL_ID = "criss_neon";
    NotificationManager notificationManager = (NotificationManager)context.getSystemService(Context.NOTIFICATION_SERVICE);
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
    {
        NotificationChannel notificationChannel = new NotificationChannel(NOTIFICATION_CHANEL_ID,
                name: "Criss Neon",NotificationManager.IMPORTANCE_DEFAULT);
        notificationChannel.setDescription("Criss Neon");
        notificationChannel.enableLights(true);
        notificationChannel.setLightColor(Color.RED);
        notificationChannel.setVibrationPattern(new long[]{0,1000,500,1000});
        notificationChannel.enableVibration(true);

        notificationManager.createNotificationChannel(notificationChannel);



    }
    NotificationCompat.Builder builder = new NotificationCompat.Builder(context,NOTIFICATION_CHANEL_ID);
    builder.setContentTitle(title)
            .setContentText(content)
            .setAutoCancel(true)
            .setSmallIcon(R.mipmap.ic_launcher_round)
            .setLargeIcon(BitmapFactory.decodeResource(context.getResources(),R.drawable.ic_menu_slideshow));
    if(pendingIntent != null)
        builder.setContentIntent(pendingIntent);
    Notification notification = builder.build();
    notificationManager.notify(id,notification);
}
```

En la sintaxis de código mostrada anteriormente, se muestra la codificación para visualizar las notificaciones mediante el Firebase Mesagging y servicios como NOTIFICATION_SERVICE que servirá para la visualización de noticias emitido por el administrador.

- **Storycard 17 : Cerrar Sesión**

Interfaz	Descripción
	Se puede visualizar la transmisión de noticias ya permitido la subscripción de noticias por el usuario dando como cuerpo la foto, título y contexto.

```
private void signOut(){
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Cerrar Sesión")
            .setMessage("Desea realmente cerrar sesión?")
            .setNegativeButton( text: "CANCELAR", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    dialogInterface.dismiss();

                }
            }).setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int i) {

            Common.selectedProducto = null;
            Common.categorySelected = null;
            Common.currentUser = null;
            FirebaseAuth.getInstance().signOut();

            Intent intent = new Intent( packageContext: HomeActivity.this,MainActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(intent);
            finish();

        }
    });
    AlertDialog dialog = builder.create();
    dialog.show();
}
```

En la sintaxis de código mostrada anteriormente, se muestra la codificación para cerrar cesión mediante actividades derivadas al FLAG cerrando toda actividad hecha por el usuario.

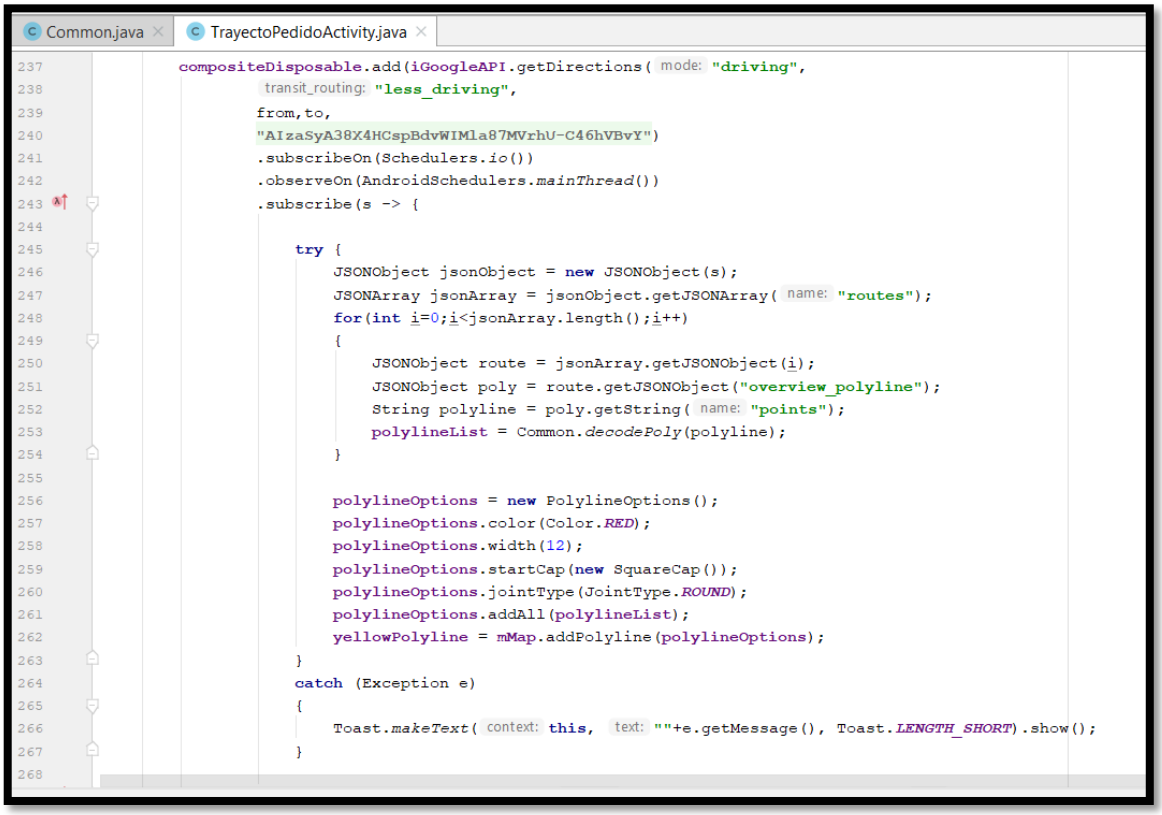
- **Storycard 18: Geolocalización del trayecto del pedido del módulo cliente**

Interfaz	Descripción
	Se puede visualizar el servicio de Google Maps mostrada para visualizar el trayecto en tiempo en real del pedido del cliente marcándose por una línea roja, viéndose el celular y tiempo referencial del pedido, en la parte superior se observa un botón “Llamar al Repartidor” que servirá para llamar de forma inmediata al repartidor.

```
private void drawRoutes() {
        LatLng locationPedido = new LatLng(Common.currentCaminoPedido.getPedidoModel().getLat(),
                Common.currentCaminoPedido.getPedidoModel().getLng());
    LatLng locationRepartidor = new LatLng(Common.currentCaminoPedido.getCurrentLat(),Common.currentCaminoPedido.getCurrentLng());

    //anadimos al cliente en la geolocalizacion

    mMap.addMarker(new MarkerOptions()
            .icon(BitmapDescriptorFactory.fromResource(R.drawable.box))
            .title(Common.currentCaminoPedido.getPedidoModel().getUserName())
            .snippet(Common.currentCaminoPedido.getPedidoModel().getDireccionEnvio())
            .position(locationPedido));

    //anadimos al repartidor en la geolocalizacion
    if(repartidorMarker == null)
    {

        int height,width;
        height = width=80;
        BitmapDrawable bitmapDrawable = (BitmapDrawable) ContextCompat
                .getDrawable( context: TrayectoPedidoActivity.this,R.drawable.crissrepartidor);
        Bitmap resized = Bitmap.createScaledBitmap(bitmapDrawable.getBitmap(),width,height, filter: false);

        repartidorMarker =  mMap.addMarker(new MarkerOptions()
                .icon(BitmapDescriptorFactory.fromBitmap(resized))
                .title(new StringBuilder("Repartidor: ").append(Common.currentCaminoPedido.getRepartidorName()).toString())
                .snippet(new StringBuilder("Celular: ").append(Common.currentCaminoPedido.getRepartidorPhone())
                        .append("\n")
                        .append("Tiempo Referencial: ")
                        .append(Common.currentCaminoPedido.getEstimacionTiempo()).toString())
                .position(locationRepartidor));
```

En la primera sintaxis de código mostrada anteriormente, se muestra la codificación para el trazado de la línea roja mediante la tabla de CaminoPedido de Firebase que va arrastrar la longitud y latitud del pedido a entregar, para

mostrar el nombre , celular del repartidor y tiempo referencial del pedido se uso los servicios de Google Maps y Places para hacer una estimación de tiempo de la ubicación actual del repartidor hacia el destinatario el tiempo referencial del mismo.



En la segunda sintaxis de código mostrada anteriormente, se muestra la codificación de un objeto JSON para trazar las rutas mediante Google Poly, quien facilitara el trazado de rutas de color red, así mostrando al cliente que el repartidor siga la línea roja destinataria al pedido.

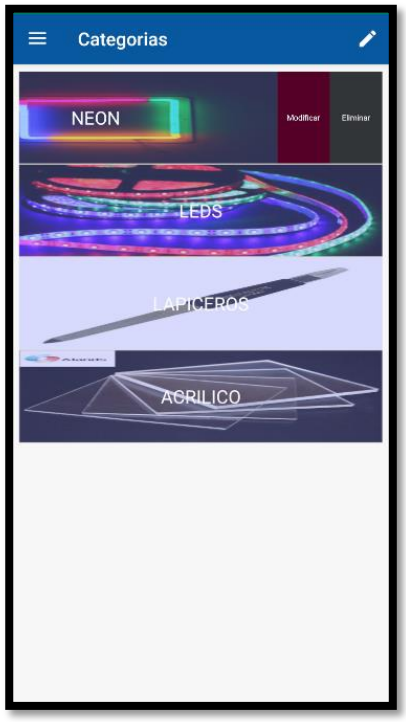
```
public static List<LatLng> decodePoly(String encoded) {
    List poly = new ArrayList();
    int index=0,len=encoded.length();
    int lat=0,lng=0;
    while (index < len)
    {
        int b, shift=0,result=0;
        do{
            b=encoded.charAt(index++)-63;
            result |= (b &0x1f) << shift;
            shift+=5;
        }while (b >= 0x20);
        int dlat = ((result & 1) != 0 ? ~(result >> 1):(result >> 1));
        lat += dlat;

        shift = 0;
        result = 0;
        do {
            b=encoded.charAt(index++)-63;
            result |= (b & 0x1f) << shift;
            shift +=5;
        }while (b >= 0x20);
        int dlng = ((result & 1) != 0 ? ~(result >> 1):(result >> 1));
        lng += dlng;

        LatLng p = new LatLng((((double)lat / 1E5)),
                (((double)lng/1E5)));
        poly.add(p);

    }
    return poly;
}
```

En la tercera sintaxis de código mostrada anteriormente, se muestra la obtención de la latitud y longitud en el objeto JSON en Google Maps mediante métodos repetitivos que nos ayudará a obtener las coordenadas del mismo mediante el trazado de rutas.

```
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
    //guardamos la antigua posicion
    String from = new StringBuilder()
            .append(Common.currentCaminoPedido.getCurrentLat())
            .append(",")
            .append(Common.currentCaminoPedido.getCurrentLng())
            .toString();
    ///modificar pposicion
    Common.currentCaminoPedido = dataSnapshot.getValue(CaminoPedidoModel.class);
    Common.currentCaminoPedido.setKey(dataSnapshot.getKey());
    //guardar nueva posicion
    String to = new StringBuilder()
            .append(Common.currentCaminoPedido.getCurrentLat())
            .append(",")
            .append(Common.currentCaminoPedido.getCurrentLng())
            .toString();
    if(dataSnapshot.exists())
    {
        if(isInit)
            moveMakerAnimation(repartidorMarker,from,to);
        else
            isInit=true;
    }
```

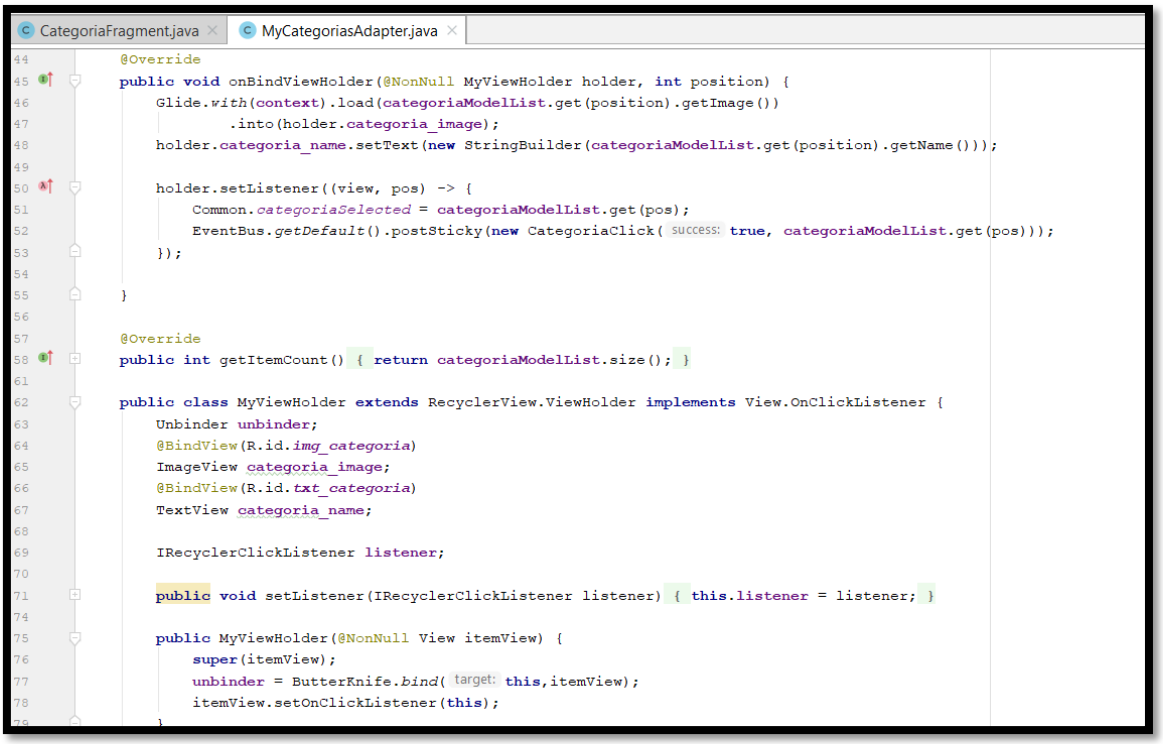
En la cuarta sintaxis de código mostrada anteriormente, se muestra la codificación de la obtención de la latitud y longitud mediante el DataSnapshot de la tabla CaminoPedido obteniendo esto y si existe forma la línea roja trazada.

- **Storycard 19 : Listar Edición de Categorías**

Interfaz	Descripción
	Se puede visualizar la lista de categorías existentes por el administrador y en la parte desplegable las opciones de Modificar y Eliminar, en la parte superior se visualiza un lápiz que ayudará a agregar nuevas categorías.

```
private void initViews() {

    storage = FirebaseStorage.getInstance();
    storageReference = storage.getReference();

    dialog = new SpotsDialog.Builder().setContext(getContext()).setCancelable(false).build();
    //dialog.show();
    LinearLayoutManager layoutManager = new LinearLayoutManager(getContext());
    recycler_menu.setLayoutManager(layoutManager);
    recycler_menu.addItemDecoration(new DividerItemDecoration(getContext(),layoutManager.getOrientation()));

    MySwipeHelper mySwipeHelper = new MySwipeHelper(getContext(),recycler_menu, buttonWidth: 200) {
        @Override
        public void instantiateMyButton(RecyclerView.ViewHolder viewHolder, List<MyButton> buf) {

            buf.add(new MyButton(getContext(), text: "Eliminar", textSize: 30, imageResId: 0, Color.parseColor( colorString: "#333639"),
                    pos -> {

                        Common.categoriaSelected = categoriaModels.get(pos);

                        showEliminarCategoria();
                    }));

            buf.add(new MyButton(getContext(), text: "Modificar", textSize: 30, imageResId: 0, Color.parseColor( colorString: "#560027"),
                    pos -> {

                Common.categoriaSelected = categoriaModels.get(pos);

                showUpdateDialog();
                    }));
```

En la primera sintaxis de código mostrada anteriormente, se muestra la codificación con el MySwiperHelper para el desglosamiento de las opciones de Eliminar mediante un showDialog la opción.



En la segunda sintaxis de código mostrada anteriormente , se muestra la obtención de datos de categorías existentes mediante la clase Common para poder visualizar mediante RecyclerView la imagen del producto y categoría, se usa el ButterKnifer para la relación de elementos de información de Categorías.

- **Storycard 20 : Agregar Categorías**

Interfaz	Descripción
	Se puede visualizar un showDialog para agregar nuevas categorías mediante el nombre y una imagen de la categoría.

```
builder.setPositiveButton( text: "AGREGAR", (dialogInterface, i) -> {


    CategoriaModel categoriaModel = new CategoriaModel();
    categoriaModel.setName(edt_categoria_name.getText().toString());
    categoriaModel.setProductos(new ArrayList<>());
    if(imageUri != null)
    {
        dialog.setMessage("Agregando categoria.....");
        dialog.show();

        String unique_name = UUID.randomUUID().toString();
        StorageReference imageFolder = storageReference.child("images/"+unique_name);

        imageFolder.putFile(imageUri)
                .addOnFailureListener(e -> {
                    dialog.dismiss();
                    Toast.makeText(getContext(), text: ""+e.getMessage(), Toast.LENGTH_SHORT).show();
                }).addOnCompleteListener(task -> {
            dialog.dismiss();
            imageFolder.getDownloadUrl().addOnSuccessListener(uri -> {
                categoriaModel.setImage(uri.toString());
                agregarCategoria(categoriaModel);
            });
        }).addOnProgressListener(taskSnapshot -> {
            double progress = (100.00* taskSnapshot.getBytesTransferred() / taskSnapshot.getTotalByteCount());
            dialog.setMessage(new StringBuilder("Agregando: ").append(progress).append("%"));
        });

    }else{
        agregarCategoria(categoriaModel);
    }
```

En la sintaxis de código mostrada anteriormente, se muestra la codificación de la agregación una nueva categoría mediante un showDialog almacenando los datos por la obtención de datos y Firebase Storage extrayendo imágenes del dispositivo del celular y un ProgressListener para mostrar la carga de la categoría.

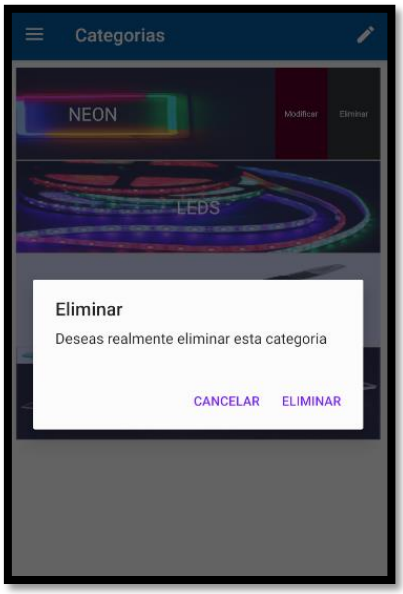
- **Storycard 21 : Modificar Categorías**

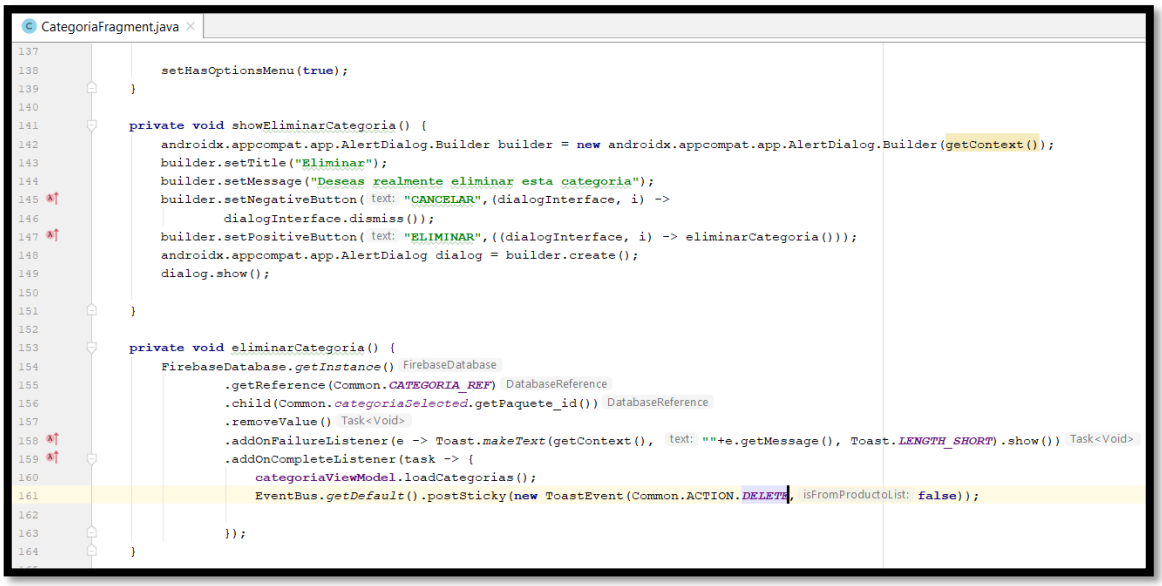
Interfaz	Descripción
	Se puede visualizar un showDialog para modificar las categorías existentes pudiendo reemplazar el nombre o foto.

```java
builder.setPositiveButton( text: "MODIFICAR", (dialogInterface, i) -> {
    Map<String,Object> updateData = new HashMap<>();
    updateData.put("name",edt_categoria_name.getText().toString());
    if(imageUri != null)
    {
        dialog.setMessage("Modificando categoria.....");
        dialog.show();

        String unique_name = UUID.randomUUID().toString();
        StorageReference imageFolder = storageReference.child("images/"+unique_name);

        imageFolder.putFile(imageUri)
                .addOnFailureListener(e -> {
                    dialog.dismiss();
                    Toast.makeText(getContext(), text: ""+e.getMessage(), Toast.LENGTH_SHORT).show();
                }).addOnCompleteListener(task -> {
            dialog.dismiss();
            imageFolder.getDownloadUrl().addOnSuccessListener(uri -> {
                updateData.put("image", uri.toString());
                updateCategory(updateData);
            });
        }).addOnProgressListener(taskSnapshot -> {
            double progress = (100.00* taskSnapshot.getBytesTransferred() / taskSnapshot.getTotalByteCount());
            dialog.setMessage(new StringBuilder("Modificando: ").append(progress).append("%"));
        });

    }else{
        updateCategory(updateData);
    }

});
builder.setView(itemView);
```

En la sintaxis de código mostrada anteriormente, se muestra la codificación para modificar la categoría almacenando las nuevas imágenes del producto mediante el FirebaseStorage y los datos por el método put.

- **Storycard 22 : Eliminar Categorías**

Interfaz	Descripción
	Se puede visualizar un showDialog para eliminar la categoría existente.



```
CategoriaFragment.java

        setHasOptionsMenu(true);
    }

    private void showEliminarCategoria() {
        androidx.appcompat.app.AlertDialog.Builder builder = new androidx.appcompat.app.AlertDialog.Builder(getContext());
        builder.setTitle("Eliminar");
        builder.setMessage("Deseas realmente eliminar esta categoria");
        builder.setNegativeButton( text: "CANCELAR",(dialogInterface, i) ->
                dialogInterface.dismiss());
        builder.setPositiveButton( text: "ELIMINAR",((dialogInterface, i) -> eliminarCategoria()));
        androidx.appcompat.app.AlertDialog dialog = builder.create();
        dialog.show();

    }

    private void eliminarCategoria() {
        FirebaseDatabase.getInstance() FirebaseDatabase
                .getReference(Common.CATEGORIA_REF) DatabaseReference
                .child(Common.categoriaSelected.getPaquete_id()) DatabaseReference
                .removeValue() Task<Void>
                .addOnFailureListener(e -> Toast.makeText(getContext(), text: ""+e.getMessage(), Toast.LENGTH_SHORT).show()) Task<Void>
                .addOnCompleteListener(task -> {
                    categoriaViewModel.loadCategorias();
                    EventBus.getDefault().postSticky(new ToastEvent(Common.ACTION.DELETE, isFromProductoList: false));

                });
    }
```

En la sintaxis de código mostrada anteriormente, se muestra la codificación para eliminar la categoría mediante un showDialog para eliminarla mediante la obtención de datos del FirebaseDatabase que lo remueve con ayuda del Common para eliminar la categoría seleccionada.

- **Storycard 23 : Listar Edición de Productos por Categoría**

Interfaz	Descripción
	Se puede visualizar la lista de productos existentes por el administrador y en la parte desplegable las opciones de registrar complementos, tamaños, editar y eliminar por producto, como visualizamos en la parte superior, el icono del lápiz servirá para agregar nuevos productos y el icono de lupa para buscar el producto.

```
ProductoListFragment.java | MyProductoListAdapter.java

    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        Glide.with(context).load(productoModelList.get(position).getImage()).into(holder.img_producto_image);
        holder.txt_producto_price.setText(new StringBuilder("S/.")
                .append(productoModelList.get(position).getPrice()));
        holder.txt_producto_name.setText(new StringBuilder("")
                .append(productoModelList.get(position).getName()));

        holder.setListener((view, pos) -> {
            Common.selectedProducto = productoModelList.get(pos);
            Common.selectedProducto.setKey(String.valueOf(pos));
        });



    }

    @Override
    public int getItemCount() { return productoModelList.size(); }

    public ProductoModel getItemAtPosition(int pos) { return productoModelList.get(pos); }

    public class MyViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
        private Unbinder unbinder;
        @BindView(R.id.txt_producto_name)
        TextView txt_producto_name;
        @BindView(R.id.txt_producto_price)
        TextView txt_producto_price;
        @BindView(R.id.img_producto_image)
        ImageView img_producto_image;
```

En la primera sintaxis de código mostrada anteriormente, se muestra la codificación para mostrar la lista de productos por sus datos mediante las

instancias de Firebase mostrando en cada componente del activity la información de cada producto.

En la segunda sintaxis de código mostrada anteriormente, se muestra con ayuda de MySwipeHelper, eliminar el producto mediante el llamado de un showDialog para eliminar la categoría seleccionada por producto eliminándolo de la lista y edición de datos llamando a un showDialog para editar los campos del producto.

En la tercera sintaxis de código mostrada anteriormente, se muestra con ayuda de MySwipeHelper, para registrar nuevos tamaños por productos de tal manera con el tamaño con EventBus para decrementar los errores en esta sección.

```
private void startBuscarProducto(String s) {
    List<ProductoModel> resultProducto = new ArrayList<>();
    for(int i=0; i<Common.categoriaSelected.getProductos().size();i++) {
        ProductoModel productoModel = Common.categoriaSelected.getProductos().get(i);
        if (productoModel.getName().toLowerCase().contains(s.toLowerCase()))
        {
            productoModel.setPositionInList(i);
            resultProducto.add(productoModel);
        }
    }
        productoListViewModel.getMutableLiveDataProductoList().setValue(resultProducto);

}
```

En la cuarta sintaxis de código mostrada anteriormente, se muestra la codificación para buscar el producto mediante un arreglo que contenga un for que traiga el producto pro categoría buscado.

```
public void onCreateOptionsMenu(@NonNull Menu menu, @NonNull MenuInflater inflater) {
    inflater.inflate(R.menu.producto_list_menu,menu);

    MenuItem menuItem = menu.findItem(R.id.action_buscar);
    SearchManager searchManager = (SearchManager)getActivity().getSystemService(Context.SEARCH_SERVICE);
    SearchView searchView = (SearchView) menuItem.getActionView();
    searchView.setSearchableInfo(searchManager.getSearchableInfo(getActivity().getComponentName()));

    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String s) {
            startBuscarProducto(s);
            return true;
        }

        @Override
        public boolean onQueryTextChange(String s) { return false; }
    });

    ImageView closeButton = (ImageView)searchView.findViewById(R.id.search_close_btn);
    closeButton.setOnClickListener(v -> {
        EditText ed = (EditText)searchView.findViewById(R.id.search_src_text);

        ed.setText("");

        searchView.setQuery( query: "", submit: false);
        searchView.onActionViewCollapsed();
        menuItem.collapseActionView();
        productoListViewModel.getMutableLiveDataProductoList().setValue(Common.categoriaSelected.getProductos());

    });
```

En la quinta sintaxis de código mostrada anteriormente, se muestra la codificación para traer la imagen y nombre del producto mediante el método startBuscarProducto que nos devuelve la consulta en Firebase sobre el producto buscado.

- **Storycard 24 : Registrar complementos por producto**

Interfaz	Descripción
	Se puede visualizar una interfaz traída por el MySwiperHelper por producto para añadir, eliminar, editar complementos, en la parte superior visualizamos un icono de guardar y otro para regresar.

```
@NonNull
@Override
public MyAnadirAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    return new MyAnadirAdapter.MyViewHolder(LayoutInflater.from(context)
            .inflate(R.layout.layout_tamano_complemento_display,parent, attachToRoot: false));
}



@Override
public void onBindViewHolder(@NonNull MyAnadirAdapter.MyViewHolder holder, int position) {
    holder.txt_name.setText(addonModels.get(position).getName());
    holder.txt_price.setText(String.valueOf(addonModels.get(position).getPrice()));

    holder.img_delete.setOnClickListener(view -> {
        addonModels.remove(position);

        notifyItemRemoved(position);
        updateAddonModel.setAddonModels(addonModels);
        EventBus.getDefault().postSticky(updateAddonModel);
    });

    holder.setListener(new IRecyclerClickListener() {
        @Override
        public void onItemClickListener(View view, int pos) {
            editPos = position;
            EventBus.getDefault().postSticky(new SelectAddonModel(addonModels.get(pos)));

        }
    });
}
```

En la primera sintaxis de código mostrada anteriormente, se muestra la codificación para remover el complemento a través de las posiciones por la librería EventBus obtenido la data con un get.

```
@Override
public int getItemCount() { return addonModels.size(); }

public void addNewSize(AddonModel addonModel){
    addonModels.add(addonModel);
    notifyItemInserted( position: addonModels.size()-1);
    updateAddonModel.setAddonModels(addonModels);
    EventBus.getDefault().postSticky(updateAddonModel);
}

public void editSize(AddonModel addonModel){
    if(editPos != -1)
    {
        addonModels.set(editPos,addonModel);
        notifyItemChanged(editPos);
        editPos = -1;

        updateAddonModel.setAddonModels(addonModels);
        EventBus.getDefault().postSticky(updateAddonModel);
    }

}
```

En la segunda sintaxis de código mostrada anteriormente se muestra la codificación para editar los complementos mediante la librería EventBus modificando por posición.

TamanoComplementoEditActivity.java

```
        {
            edt_name.setText(event.getSizeModel().getName());
            edt_price.setText(String.valueOf(event.getSizeModel().getPrice()));

            btn_editar.setEnabled(true);
        }else{
            btn_editar.setEnabled(false);
        }
    }


    @Subscribe(sticky =  true,threadMode =  ThreadMode.MAIN)
    public void onSelectedAddonModel(SelectAddonModel event)
    {
        if(event.getAddonModel() != null)
        {
            edt_name.setText(event.getAddonModel().getName());
            edt_price.setText(String.valueOf(event.getAddonModel().getPrice()));

            btn_editar.setEnabled(true);
        }else{
            btn_editar.setEnabled(false);
        }
    }
}
```

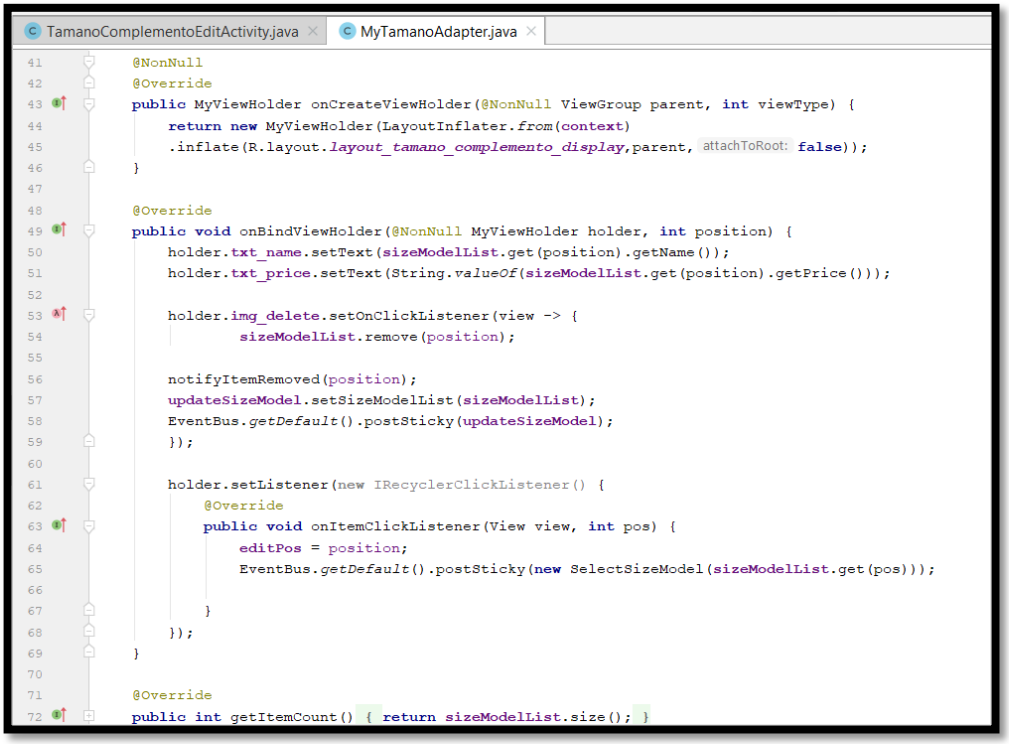
En la tercera sintaxis de código mostrada anteriormente, se agregar un nuevo complemento por un evento mediante un get que se comunica con el FirebaseDatabase para suscribirse al registro por producto de complementos.

- **Storycard 25 : Registrar tamaños por producto**

Interfaz	Descripción
	Se puede visualizar una interfaz traída por el MySwiperHelper por producto para añadir, eliminar, editar tamaños por productos, en la parte superior visualizamos un icono de guardar y otro para regresar.

```
private void saveData() {
    if(productoEditPosition != -1)
    {
        Common.categoriaSelected.getProductos().set(productoEditPosition,Common.selectedProducto);

        Map<String,Object> updateData = new HashMap<>();
        updateData.put("productos",Common.categoriaSelected.getProductos());

        FirebaseDatabase.getInstance() FirebaseDatabase
                .getReference(Common.CATEGORIA_REF) DatabaseReference
                .child(Common.categoriaSelected.getPaquete_id()) DatabaseReference
                .updateChildren(updateData) Task<Void>
                .addOnFailureListener(e -> Toast.makeText( context: TamanoComplementoEditActivity.this,  text: ""+e.getMessage(), Toast.LENGTH_SHORT).show())
                .addOnCompleteListener(task -> {
                    if(task.isSuccessful())
                    {
                        Toast.makeText( context: this,  text: "Recarga satisfactoria", Toast.LENGTH_SHORT).show();
                        needSave=false;
                        edt_price.setText("0");
                        edt_name.setText("");
                    }

                });
    }
```

En la primera sintaxis de código mostrada anteriormente, se muestra la codificación para agregar nuevos tamaños por producto mediante el método get para poder añadirlos.



En la segunda sintaxis de código mostrada anteriormente, se muestra la codificación para mostrar los compelementos mediante RecyclerView por el método onBinViewHolder que sirve como controlador para listar dichos complementos.

En la tercera sintaxis de código mostrada anteriormente, se muestra la codificación para editar y eliminar los complementos mediante las liberías de EventBus que va disminuir las fallas en cuestión a cantidad masiva de datos que contenga el tamaño por producto.

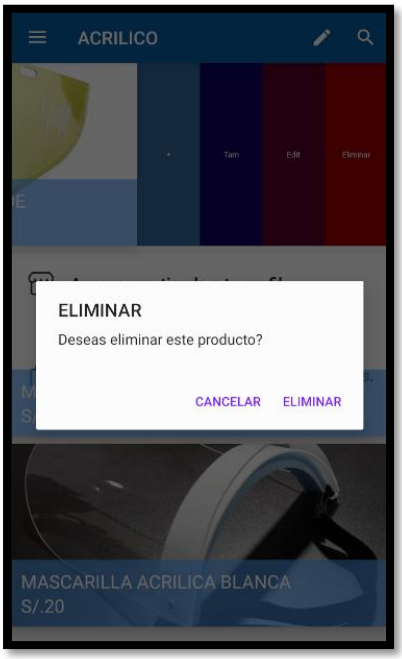
- **Storycard 26 : Modificar productos**

Interfaz	Descripción
	Se puede visualizar un dialogShow para la edición de datos, editando el id, nombre, precio, descripción o la imagen.

```
private void updateProducto(List<ProductoModel> productos, Common.ACTION action ) {
    Map<String,Object> updateData = new HashMap<>();
    updateData.put("productos",productos);

    FirebaseDatabase.getInstance() FirebaseDatabase
            .getReference(Common.CATEGORIA_REF) DatabaseReference
            .child(Common.categoriaSelected.getPaquete_id()) DatabaseReference
            .updateChildren(updateData) Task<Void>
            .addOnFailureListener(e -> Toast.makeText(getContext(), text: ""+e.getMessage(), Toast.LENGTH_SHORT).show()) Task<Void>
            .addOnCompleteListener(task -> {
                if(task.isSuccessful())
                {
                    productoListViewModel.getMutableLiveDataProductoList();
                    EventBus.getDefault().postSticky(new ToastEvent(action, isFromProductoList: true));

                }
            });
}
```

En la sintaxis de código mostrada anteriormente, se muestra la codificación para modificar los productos existentes por los datos que se desee modificar mediante la comunicación de get en la clase del Producto comunicándose con FirebaseDatabase para seleccionar por categoría de los productos.

- **Storycard 27 : Eliminar productos**

Interfaz	Descripción
	Se puede visualizar un dialogShow para la eliminación de un producto confirmando dicha eliminación.

```
buf.add(new MyButton(getContext(), text: "Eliminar", textSize: 30, imageResId: 0, Color.parseColor( colorString: "#9b0000"),
        pos -> {
    if(productoModelList != null)
        Common.selectedProducto = productoModelList.get(pos);
        AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
        builder.setTitle("ELIMINAR")
                .setMessage("Deseas eliminar este producto?")
                .setNegativeButton( text: "CANCELAR",((dialogInteface, i) -> { dialogInteface.dismiss(); }))
                .setPositiveButton( text: "ELIMINAR",(((dialogInterface, i) -> {
                    ProductoModel productoModel = adapter.getItemAtPosition(pos);
                    if(productoModel.getPositionInList() == -1)
                      Common.categoriaSelected.getProductos().remove(pos);
                    else
                        Common.categoriaSelected.getProductos().remove(productoModel.getPositionInList());
                    updateProducto(Common.categoriaSelected.getProductos(),Common.ACTION.DELETE);
                })));
        AlertDialog eliminarDialog = builder.create();
        eliminarDialog.show();
    }));
```

En la sintaxis de código mostrada anteriormente, se muestra la codificación para traer el showDialog y eliminar un producto existente mediante el controlador y el Common para la acción de eliminar por posición en Lista removiendo por posición en la lista de productos.

- **Storycard 28: Agregar productos**

Interfaz	Descripción
	Se puede visualizar un dialogShow para agregar un nuevo producto por id, nombr

```
private void showAgregarDialog() {
    androidx.appcompat.app.AlertDialog.Builder builder = new androidx.appcompat.app.AlertDialog.Builder(getContext());
    builder.setTitle("Agregar producto");
    builder.setMessage("Por favor rellenar la informacion");

    View itemView = LayoutInflater.from(getContext()).inflate(R.layout.layout_modificar_producto,  root: null);
    EditText edt_producto_id = (EditText)itemView.findViewById(R.id.edt_producto_id) ;
    EditText edt_producto_name = (EditText)itemView.findViewById(R.id.edt_producto_name);
    EditText edt_producto_price = (EditText)itemView.findViewById(R.id.edt_producto_price);
    EditText edt_producto_description = (EditText)itemView.findViewById(R.id.edt_producto_description);
    img_producto = (ImageView)itemView.findViewById(R.id.img_producto_image);



    Glide.with(getContext()).load(R.drawable.negro_ic).into(img_producto);

    img_producto.setOnClickListener(v -> {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(Intent.createChooser(intent,  title: "Seleccionar foto de producto"),PICK_IMAGE_REQUEST);
    });

    builder.setNegativeButton( text: "CANCELAR",((dialogInterface, i) -> dialogInterface.dismiss()))
            .setPositiveButton( text: "AGREGAR",((dialogInterface, i) -> {

                ProductoModel updateProducto = new ProductoModel();

                updateProducto.setId(edt_producto_id.getText().toString());
                updateProducto.setName(edt_producto_name.getText().toString());
                updateProducto.setDescription(edt_producto_description.getText().toString());
                updateProducto.setPrice(TextUtils.isEmpty(edt_producto_price.getText()) ? 0 :
```

En la primera sintaxis de código mostrada anteriormente, se muestra la codificación para traer el showDialog de agregar un producto almacenando la imagen mediante la galería del celular.

En la segunda sintaxis de código mostrada anteriormente, se muestra la codificación para la subida de imágenes mediante el Firebase Storage, la comunicación del Common y la clase Productos del get para agregar la información necesaria del nuevo producto.

- **Storycard 29: Emitir noticias**

Interfaz	Descripción
	Se puede visualizar un dialogShow para emitir una noticia con el título, contenido, si se desea se puede añadir una imagen.

```
private void showNewsDialog() {
    androidx.appcompat.app.AlertDialog.Builder builder = new   androidx.appcompat.app.AlertDialog.Builder( context: this);
    builder.setTitle("Nuevas Noticias");
    builder.setMessage("Enviar las nuevas noticias al cliente");
    View itemView = LayoutInflater.from(this).inflate(R.layout.layout_news_system, root: null);

    EditText edt_titulo = (EditText)itemView.findViewById(R.id.edt_titulo);
    EditText edt_contenido = (EditText)itemView.findViewById(R.id.edt_contenido);
    EditText edt_link = (EditText)itemView.findViewById(R.id.edt_link);
    img_imagen = (ImageView)itemView.findViewById(R.id.img_imagen);
    RadioButton rdi_ninguno = (RadioButton)itemView.findViewById(R.id.rdi_ninguno);
    RadioButton rdi_link = (RadioButton)itemView.findViewById(R.id.rdi_link);
    RadioButton rdi_imagen = (RadioButton)itemView.findViewById(R.id.rdi_imagen);

    rdi_ninguno.setOnClickListener(view -> {
        edt_link.setVisibility(View.GONE);
        img_imagen.setVisibility(View.GONE);
    });
    rdi_link.setOnClickListener(view -> {
        edt_link.setVisibility(View.VISIBLE);
        img_imagen.setVisibility(View.GONE);

    });
    rdi_imagen.setOnClickListener(view -> {
        edt_link.setVisibility(View.GONE);
        img_imagen.setVisibility(View.VISIBLE);

    });
    img_imagen.setOnClickListener(view -> {
```

En la primera sintaxis de código mostrada anteriormente, se muestra la codificación para traer el showDialog para agregar una nueva noticia y la visualización para añadir una imagen nueva.

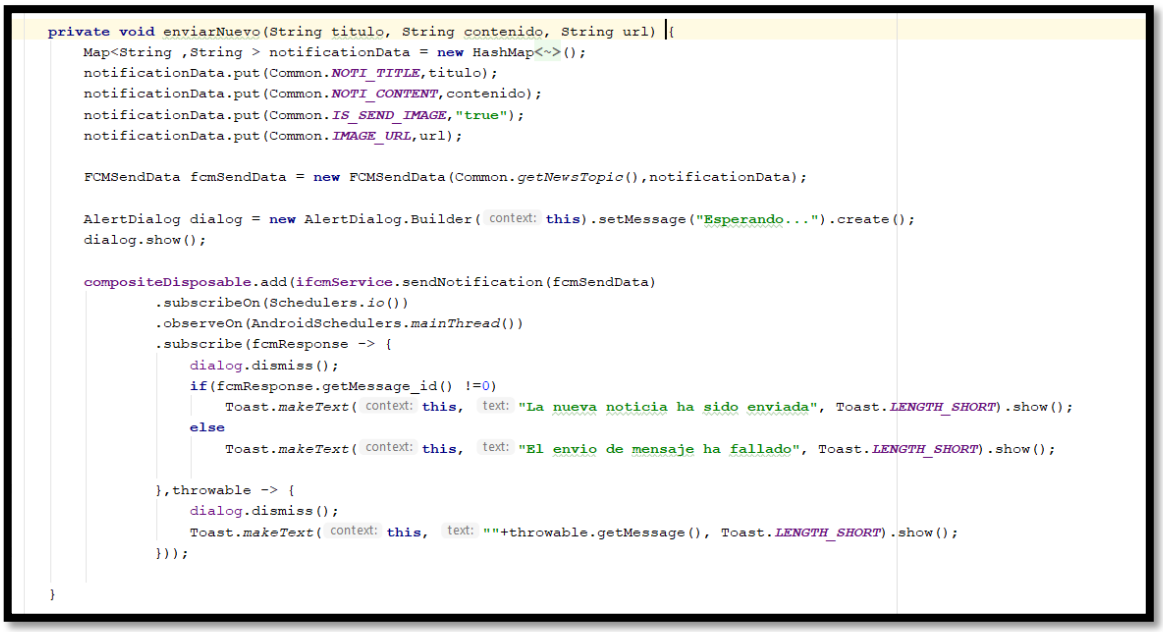
```
builder.setPositiveButton( text: "ENVIAR", (dialogInterface, i) -> {

    if(rdi_ninguno.isChecked())
    {
        enviarNuevo(edt_titulo.getText().toString(),edt_contenido.getText().toString());
    }
    else if(rdi_link.isChecked())
    {
        enviarNuevo(edt_titulo.getText().toString(),edt_contenido.getText().toString(),edt_link.getText().toString());
    }
    else if(rdi_imagen.isChecked())
    {
      if(imgUri != null)
      {
          AlertDialog dialog = new AlertDialog.Builder( context: this).setMessage("Modificando...").create();
          dialog.show();

          String file_name = UUID.randomUUID().toString();
          StorageReference newsImages = storageReference.child("news/"+file_name);
          newsImages.putFile(imgUri)
                  .addOnFailureListener(e -> {
                      dialog.dismiss();
                      Toast.makeText( context: this,  text: ""+e.getMessage(), Toast.LENGTH_SHORT).show();

                  }).addOnSuccessListener(taskSnapshot -> {
                      dialog.dismiss();
                      newsImages.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                          @Override
                          public void onSuccess(Uri uri) {
                              enviarNuevo(edt_titulo.getText().toString(),edt_contenido.getText().toString(),uri.toString());
                          }
```

En la segunda sintaxis de código mostrada anteriormente, se muestra la codificación para la subida de imágenes mediante la subida al FirebaseStorage y emisión por un api key mostrada en la figura “” para la transmisión de nuevas noticias para el perfil de usuario del cliente.

IFCMService.java

```
package com.example.proyectoserver.remote;

import ...

public interface IFCMService {
    @Headers({
            "Content-Type:application/json",
            "Authorization:key=AAAA9yY_UI4:APA91bGEcsW2CxAyvtPsZIgaaPSxzKxAn7VZfDLWmpOjbGIdsW6pfuJ9XcNCx6MD-5njz8FOLAIrGf9ys2-gc3whud-78TmZ9G5E4HzSBCIaui
    })
    @POST("fcm/send")
    Observable<FCMResponse> sendNotification(@Body FCMSendData body);
}
```

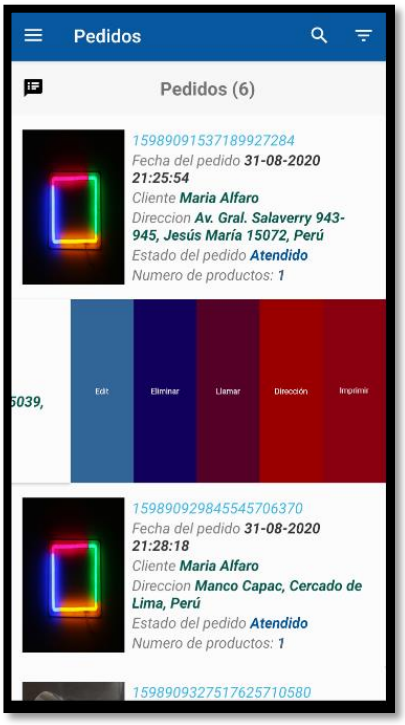
En la tercera sintaxis de código mostrada anteriormente, se muestra la interface clase IFCMService que contiene una api en formato application/json del la clave de Firebase para que los ususarios que estén subscritos a las noticias puedan visualizarlo.



```
private void enviarNuevo(String titulo, String contenido, String url) {
    Map<String ,String > notificationData = new HashMap<~>();
    notificationData.put(Common.NOTI_TITLE,titulo);
    notificationData.put(Common.NOTI_CONTENT,contenido);
    notificationData.put(Common.IS_SEND_IMAGE,"true");
    notificationData.put(Common.IMAGE_URL,url);

    FCMSendData fcmSendData = new FCMSendData(Common.getNewsTopic(),notificationData);

    AlertDialog dialog = new AlertDialog.Builder( context: this).setMessage("Esperando...").create();
    dialog.show();

    compositeDisposable.add(ifcmService.sendNotification(fcmSendData)
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(fcmResponse -> {
                dialog.dismiss();
                if(fcmResponse.getMessage_id() !=0)
                    Toast.makeText( context: this,  text: "La nueva noticia ha sido enviada", Toast.LENGTH_SHORT).show();
                else
                    Toast.makeText( context: this,  text: "El envio de mensaje ha fallado", Toast.LENGTH_SHORT).show();

            },throwable -> {
                dialog.dismiss();
                Toast.makeText( context: this,  text: ""+throwable.getMessage(), Toast.LENGTH_SHORT).show();
            }));

}
```

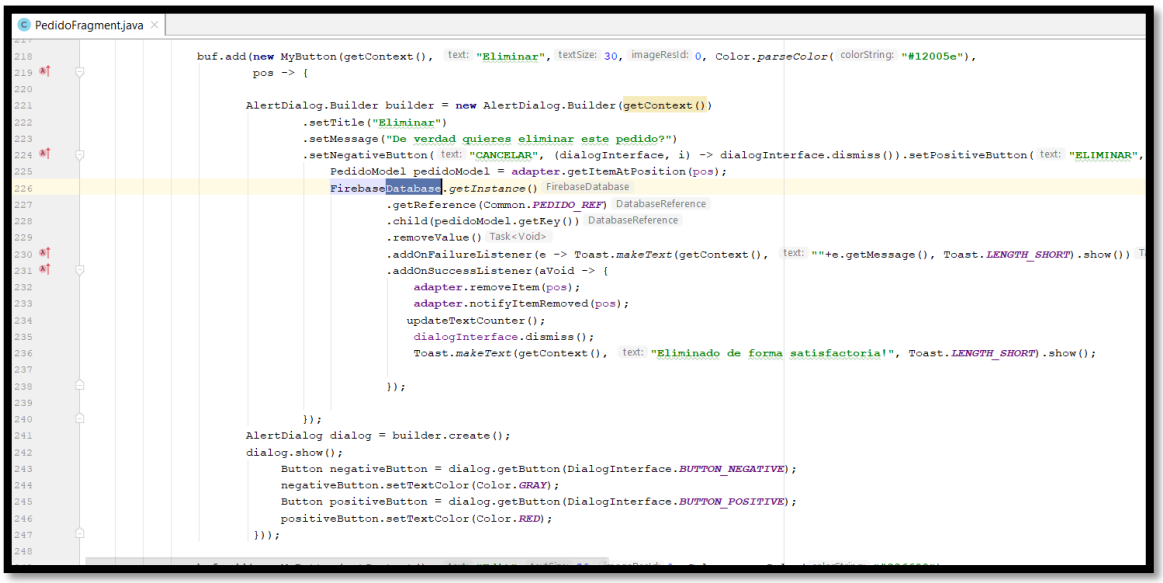
En la cuarta sintaxis de código mostrada anteriormente, se muestra el método para poder enviar notificiaciones a otros usaurios mediante la subscripción del api key de firebase para que esta pueda obtener el mensaje en un FCMSendData que pueda enviar la información a los dispositivos móviles de los usuarios.

- **Storycard 30 : Listar Pedidos**

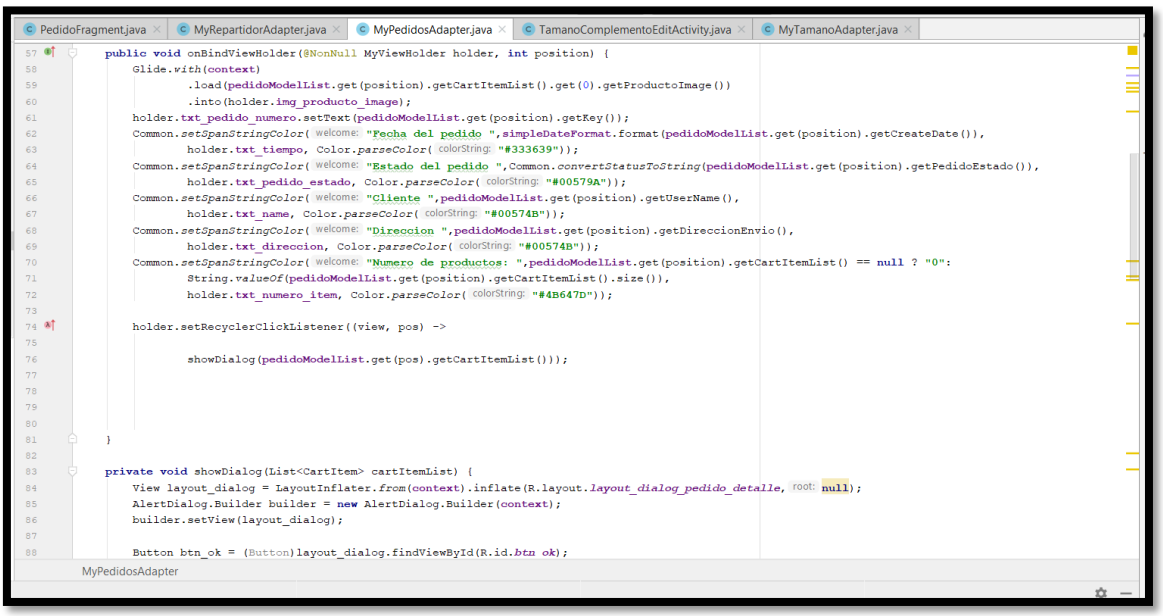
Interfaz	Descripción
	Se puede visualizar la lista de pedidos que esta contenida en un recyclerview, contienen los datos: código, fecha,hora, cliente, dirección, estado y número de productos, con la ayuda del MySwipeHelper se visualiza por cada producto, los selectores de editar (editar estado de pedido), eliminar el pedido, llamar al cliente del pedido, ver la dirección del pedido (sólo si el pedido está en camino) e imprimir el detalle del pedido en tipo PDF.

En la primera sintaxis de código mostrada anteriormente, se muestra la codificación por parte del MySwiperHelper para imprimir el detalle del pedido bajo el permiso del WRITE_EXTERNAL_STORAGE conjunto con EventBus para la transmisión del reporte del detalle del producto.

En la segunda sintaxis de código mostrada anteriormente, se muestra el permiso para ver la ubicación en tiempo real del repartidor habilitando el permiso para que la aplicación móvil pueda acceder al permiso de localización del dispositivo móvil, pero solo se verá la localización del pedido siempre y cuando el repartidor haya empezado el recorrido del pedido del cliente en conjunto accede a otra interfaz con la tabla del Pedido del cliente seleccionado, ahora vemos otro botón desplegable para llamar al cliente, consultando a la base de datos NoSQL el número celular del cliente con el permiso de CALL_PHONE.

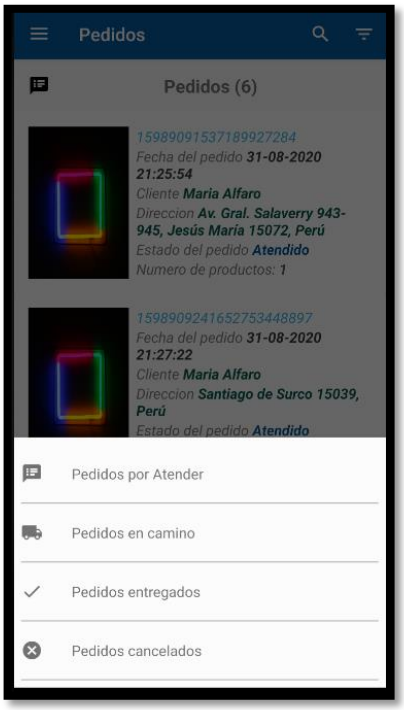


En la tercera sintaxis de código mostrada anteriormente, se muestra el método desplegable para eliminar el pedido según la clase PedidoModel para interactuar con el FirebaseDatabase referenciado con la base de datos NoSQL, removiendo mediante un adaptador por la posición.



En la cuarta sintaxis de código mostrada anteriormente, se muestra el controlador del pedido interactuando con la vista de la interfaz del pedido que va visualizarse en conjunto con Firebase para mostrar los pedidos que corresponda según el estado (Atendido, En camino, Entregado y Cancelado).

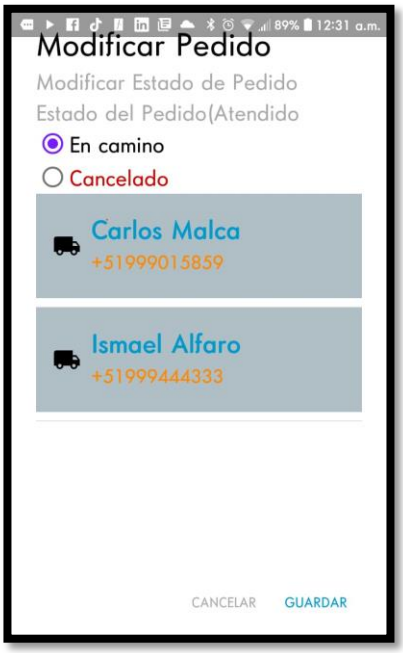
- **Storycard 31: Clasificación de los pedidos por Estado**

Interfaz	Descripción
	Se puede visualizar una ventana emergente que al presionar las tres rayas de la parte superior se visualiza cuatro estados de pedidos que cargará los pedidos en ese estado.

```
public class BottomHojaPedidoFragment extends BottomSheetDialogFragment {

    @OnClick(R.id.atendido_filtro)
    public void onAtendidoFiltroClick(){
        EventBus.getDefault().postSticky(new LoadPedidoEvent( estado: 3));
        dismiss();
    }
    @OnClick(R.id.encamino_filtro)
    public void onCaminoFiltroClick(){
        EventBus.getDefault().postSticky(new LoadPedidoEvent( estado: 2));
        dismiss();
    }
    @OnClick(R.id.entregado_filtro)
    public void onEntregadoFiltroClick(){
        EventBus.getDefault().postSticky(new LoadPedidoEvent( estado: 1));
        dismiss();
    }
    @OnClick(R.id.cancelado_filtro)
    public void onCanceladoFiltroClick(){
        EventBus.getDefault().postSticky(new LoadPedidoEvent( estado: 0));
        dismiss();
    }

    private Unbinder unbinder;
    private static BottomHojaPedidoFragment instance;

    public static BottomHojaPedidoFragment getInstance() {
        return instance == null ? new BottomHojaPedidoFragment() : instance;
    }
```

En la sintaxis de código mostrada anteriormente, se muestra la clase BottomHojaPedidoFragment acompañado con el estilo de BottomSheetDialog para poder adaptar la interfaz a una ventana emergente, en cada método public void se visualiza la carga con FirebaseDatabase y EventBus para el cambio de estado de pedidos.

- **Storycard 32: Modificar pedido en estado Atendido**

Interfaz	Descripción
	Se puede visualizar una interfaz para modificar el pedido en estado atendido por en camino que tiene que seleccionar al repartidor o cancelar el pedido.

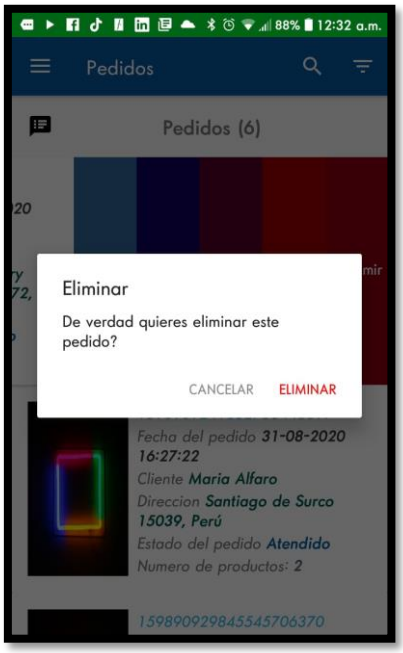
```
private void showEditarDialog(PedidoModel pedidoModel, int pos) {
    View layout_dialog;
    AlertDialog.Builder builder;
    if(pedidoModel.getPedidoEstado() == 3)
    {
        layout_dialog = LayoutInflater.from(getContext())
                .inflate(R.layout.layout_dialog_camino, root: null);

        recycler_repartidor = layout_dialog.findViewById(R.id.recycler_repartidores);

        builder = new AlertDialog.Builder(getContext(),android.R.style.Theme_Material_Light_NoActionBar_Fullscreen)
        .setView(layout_dialog);
    }
    else if(pedidoModel.getPedidoEstado() == 0)
    {
        layout_dialog = LayoutInflater.from(getContext())
                .inflate(R.layout.layout_dialog_cancelado, root: null);
        builder = new AlertDialog.Builder(getContext())
        .setView(layout_dialog);
    } else
    {
        layout_dialog = LayoutInflater.from(getContext())
                .inflate(R.layout.layout_dialog_entregado, root: null);
        builder = new AlertDialog.Builder(getContext())
        .setView(layout_dialog);
    }

    Button btn_ok = (Button)layout_dialog.findViewById(R.id.btn_ok);
    Button btn_cancelar = (Button)layout_dialog.findViewById(R.id.btn_cancelar);
```

En la primera sintaxis de código mostrada anteriormente, se muestra el método showEditarDialog para interactual con la clase Pedido y el Firebase mediante el recyclerView, la cual cambia a un estado 0 si es cancelado y aun estado 2 si es un pedido en camino.

```
private void loadRepartidorList(int pos, PedidoModel pedidoModel, AlertDialog dialog, Button btn_ok, Button btn_cancelar, RadioButton rdi_camino,
  List<RepartidorModel>  tempList = new ArrayList<>();
    DatabaseReference repartidorRef = FirebaseDatabase.getInstance().getReference(Common.REPARTIDOR);
    Query repartidorActive = repartidorRef.orderByChild("activate").equalTo(true);
    repartidorActive.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for(DataSnapshot repartidorSnapshot:dataSnapshot.getChildren())
            {
                RepartidorModel repartidorModel = repartidorSnapshot.getValue(RepartidorModel.class);
                repartidorModel.setKey_repartidor(repartidorSnapshot.getKey());
                tempList.add(repartidorModel);

            }
            repartidorLoadCallbackListener.onRepartidorLoadSucess(pos,pedidoModel,tempList,
                    dialog,
                    btn_ok,btn_cancelar,
                    rdi_camino,rdi_entregado,rdi_cancelado,rdi_eliminar,rdi_restaurar_pedido);

        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            repartidorLoadCallbackListener.onRepartidorLoadFailed(databaseError.getMessage());

        }
    });
}
```

En la segunda sintaxis de código mostrada anteriormente, se muestra el método para listar los repartidores mediante un arrayList que hace referencia a FirebaseDatabase que se obtiene de la tabla Repartidores pero solamente si están activados por el administrador.

- **Storycard 33 : Eliminar Pedido**

Interfaz	Descripción
	Se puede visualizar un showDialog que a través del menú desplegable de cada pedido eliminar un pedido y eso mismo se notificará al cliente de su dispositivo móvil.


```
private void eliminarPedido(int pos, PedidoModel pedidoModel) {
    if(!TextUtils.isEmpty(pedidoModel.getKey()))
    {

        FirebaseDatabase.getInstance() FirebaseDatabase
                .getReference(Common.PEDIDO_REF) DatabaseReference
                .child(pedidoModel.getKey()) DatabaseReference
                .removeValue() Task<Void>
                .addOnFailureListener(e -> Toast.makeText(getContext(), text: ""+e.getMessage(), Toast.LENGTH_SHORT).show()) Task<Void>
                .addOnSuccessListener(aVoid -> {
                    adapter.removeItem(pos);
                    adapter.notifyItemRemoved(pos);
                    updateTextCounter();
                    Toast.makeText(getContext(), text: "Pedido eliminado satisfactoriamente", Toast.LENGTH_SHORT).show();

                });


    }else {
        Toast.makeText(getContext(), text: "El número de pedido no debe ser nulo o vacio", Toast.LENGTH_SHORT).show();
    }
}
```

En la primera sintaxis de código mostrada anteriormente, se muestra el método eliminarPedido, para eliminar el pedido seleccionado referenciandolo de FirebaseDatabase de la tabla Pedido, removiendo de la lista de pedidos descontando por el método updateTextCounter en la figura, la cantidad de pedidos.

```
    private void updateTextCounter() {
        txt_pedido_filtro.setText(new StringBuilder("Pedidos (")
                .append(adapter.getItemCount())
                .append(")"));
    }
```

En la segunda sintaxis de código mostrada anteriormente, se muestra el contador para contabilizar los pedidos por estado.

```
PedidoFragment.java

    private void updatePedido(int pos, PedidoModel pedidoModel, int estado)
    {
        if(!TextUtils.isEmpty(pedidoModel.getKey()))
        {
            Map<String,Object> updateData = new HashMap<>();
            updateData.put("pedidoEstado",estado);

            FirebaseDatabase.getInstance() FirebaseDatabase
                    .getReference(Common.PEDIDO_REF) DatabaseReference
                    .child(pedidoModel.getKey()) DatabaseReference
                    .updateChildren(updateData) Task<Void>
                    .addOnFailureListener(e -> Toast.makeText(getContext(), text: ""+e.getMessage(), Toast.LENGTH_SHORT).show()) Task<Void>
                    .addOnSuccessListener(aVoid -> {

                        android.app.AlertDialog dialog = new SpotsDialog.Builder().setContext(getContext()).setCancelable(false).build();
                        dialog.show();
                        FirebaseDatabase.getInstance() FirebaseDatabase
                                .getReference(Common.TOKEN_REF) DatabaseReference
                                .child(pedidoModel.getUserId()) DatabaseReference
                                .addListenerForSingleValueEvent(new ValueEventListener() {
                                    @Override
                                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                                        if(dataSnapshot.exists())
                                        {
                                            TokenModel tokenModel = dataSnapshot.getValue(TokenModel.class);
                                            Map<String,String> notiData = new HashMap<>();
                                            notiData.put(Common.NOTI_TITLE,"Tu pedido ha sido modificado");
                                            notiData.put(Common.NOTI_CONTENT,new StringBuilder("Tu pedido ")
                                                    .append(pedidoModel.getKey())
                                                    .append(" ha sido modificado en ")
```

En la tercera sintaxis de código mostrada anteriormente, se muestra el método updatePedido para poder notificar al cliente que su pedido ha sido modificado en el dispositivo del cliente con el TokenModel que ayudará a localizar las actividades del administrador para ser llevado al dispositivo móvil del cliente.

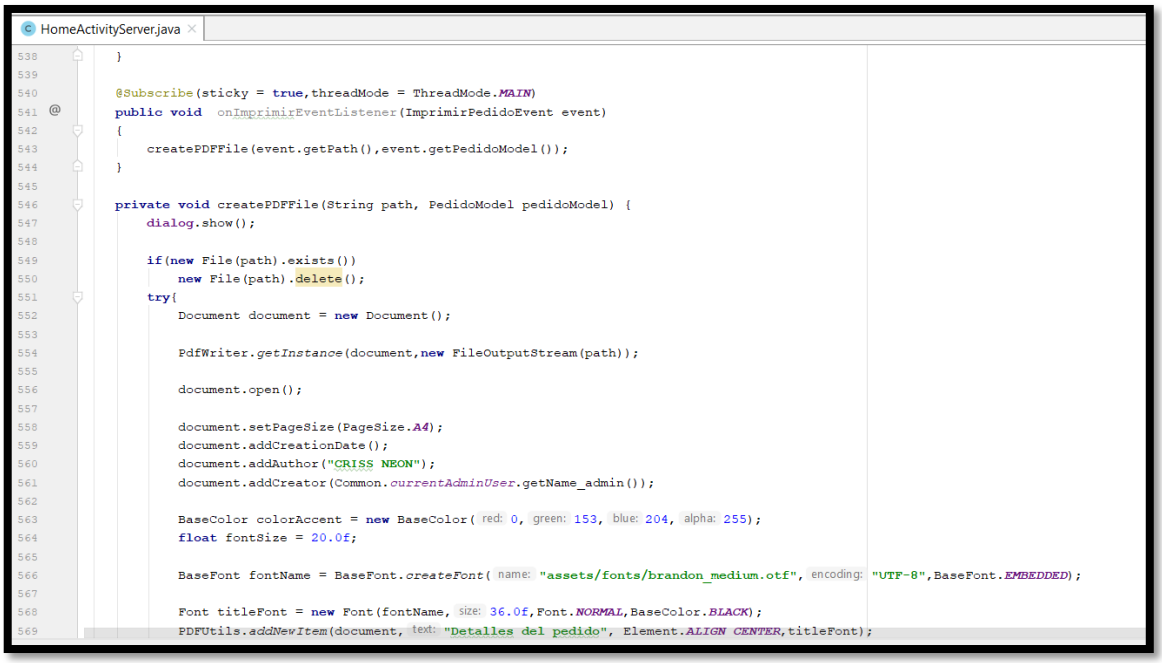
- **Storycard 34 : Emitir PDF del pedido.**

Interfaz	Descripción
	Se puede visualizar la emisión de un reporte detallado del PDF del pedido seleccionado que contiene, el número, fecha, cliente, productos, tamaño, complementos, subtotal, IGV y total del pedido.

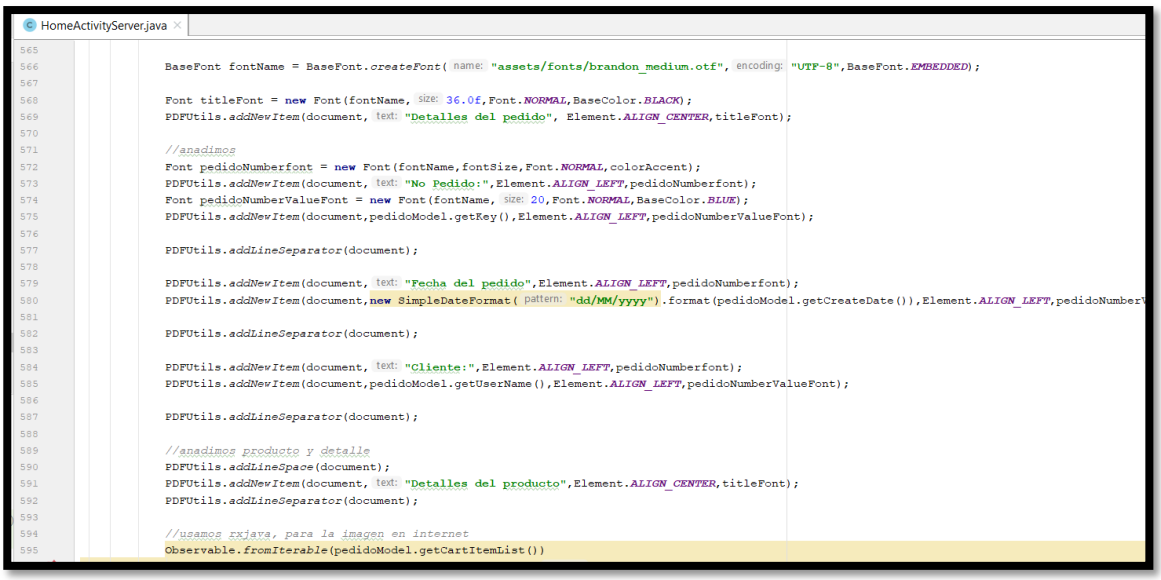
```java
public class PDFUtils {
    public static void addNewItem(Document document, String text, int align, Font font) throws DocumentException {
        Chunk chunk = new Chunk(text,font);
        Paragraph p = new Paragraph(chunk);
        p.setAlignment(align);
        document.add(p);

    }

    public static void addLineSeparator(Document document) throws DocumentException {
        LineSeparator lineSeparator = new LineSeparator();
        lineSeparator.setLineColor(new BaseColor( red: 0, green: 0, blue: 0, alpha: 68));
        addLineSpace(document);
        document.add(new Chunk(lineSeparator));
        addLineSpace(document);
    }

    public static void addLineSpace(Document document) throws DocumentException {
        document.add(new Paragraph( string: ""));
    }

    public static void addNewItemWithLeftAndRight(Document document, String leftText, String rightText, Font leftFont, Font rightFont)
            throws DocumentException {
        Chunk chunkTextLeft = new Chunk(leftText,leftFont);
        Chunk chunkTextRight = new Chunk(rightText,rightFont);
        Paragraph p = new Paragraph(chunkTextLeft);
        p.add(new Chunk(new VerticalPositionMark()));
        p.add(chunkTextRight);
        document.add(p);
    }
}
```

En la primera sintaxis de código mostrada anteriormente, se muestra la clase PDFUtils que consta de 4 método, addNewItem, lo cual junto con el chunk que es una instancia que ayudará al tamaño y lectura del texto, addLineSeparator para la separación de texto, addLineSpace para añadir un formato de líneas en

el documento y addNewItemWithLeftAndRight para agregar un margen en el lado izquierdo y derecho del PDF.



En la segunda sintaxis de código mostrada anteriormente, se muestra el método onImprimirEventListener para crear el archivo PDF obteniendo de la clase de Pedido referenciándose de la base de datos en Firebase, en el método createPDFFile, a través de PDFWriter va crear el PDF acompañado de la clase PDFUtils por una página A4 y con los estilos del PDF.



En la tercera sintaxis de código mostrada anteriormente, se muestra el detalle del PDF obteniendo de la clase de PedidoModel interactuando con DatabaseReference de Firebase por el pedido que se seleccione para descargar conjungando con la clase de PDFUtils para darle estilo al documento PDF.

En la cuarta sintaxis de código mostrada anteriormente, se muestra el formato del PDFUtils para la suma de valores del subtotal, IGV y total del pedido.

- **Storycard 35 : Modificar pedido en estado En camino**

Interfaz	Descripción
	Se puede visualizar un dialogShow para modificar el pedido en estado en camino por entregado o cancelado.


```
private void showDialog(int pos, PedidoModel pedidoModel, AlertDialog dialog, Button btn_ok, Button btn_cancelar, RadioButton rdi_camino, RadioButton

    dialog.show();

    dialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
    dialog.getWindow().setGravity(Gravity.CENTER);

    btn_cancelar.setOnClickListener(view -> dialog.dismiss());
    btn_ok.setOnClickListener(view -> {
        if(rdi_cancelado != null  && rdi_cancelado.isChecked())
        {
            updatePedido(pos,pedidoModel, estado: 0);
            dialog.dismiss();
        }
        else if(rdi_camino != null  && rdi_camino.isChecked())
        {
          //updatePedido(pos, pedidoModel, 2);
            RepartidorModel repartidorModel = null;
            if(myRepartidorSelectedAdapter != null)
            {
                repartidorModel = myRepartidorSelectedAdapter.getSeleccionarRepartidor();
                if(repartidorModel != null)
                {
                    createCaminoPedido(pos, repartidorModel,pedidoModel,dialog);
                }
                else
                    Toast.makeText(getContext(), text: "Por favor selecciona, un repartidor", Toast.LENGTH_SHORT).show();
            }

        }
        else if(rdi_entregado != null  && rdi_entregado.isChecked())
        {
            updatePedido(pos,pedidoModel, estado: 1);
            dialog.dismiss();
```

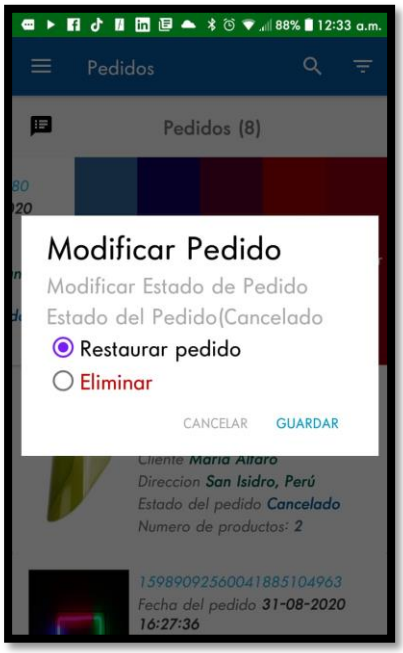
En la sintaxis de código mostrada anteriormente, se muestra el método showDialog para poder marcar el pedido como cancelado o entregado y aquello interactúa con el servicio Firebase para poder cambiar el estado del pedido y remover el pedido actual de en camino a uno de los dos estados seleccionados.

- **Storycard 36: Modificar pedido en estado Entregado**

Interfaz	Descripción
	Se puede visualizar un dialogShow para modificar el pedido en estado entregado hacia cancelado.

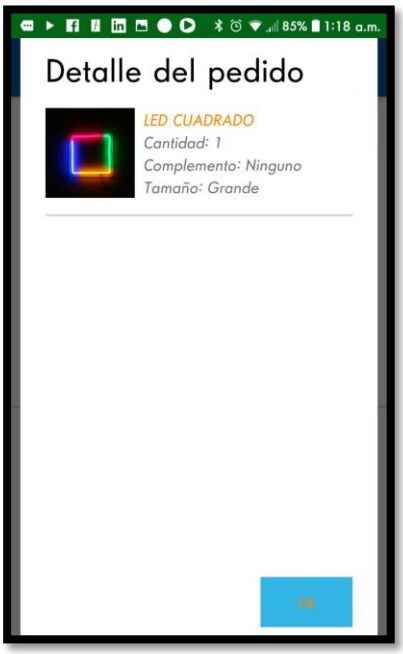
Siguiendo la lógica de la figura …., ahora en la figura …, se muestra el método para poder cambiar al estado del pedido entregado a cancelado y aquello interactúa con el servicio Firebase para poder cambiar el estado del pedido y remover el pedido actual de en camino a uno de los dos estados seleccionados.

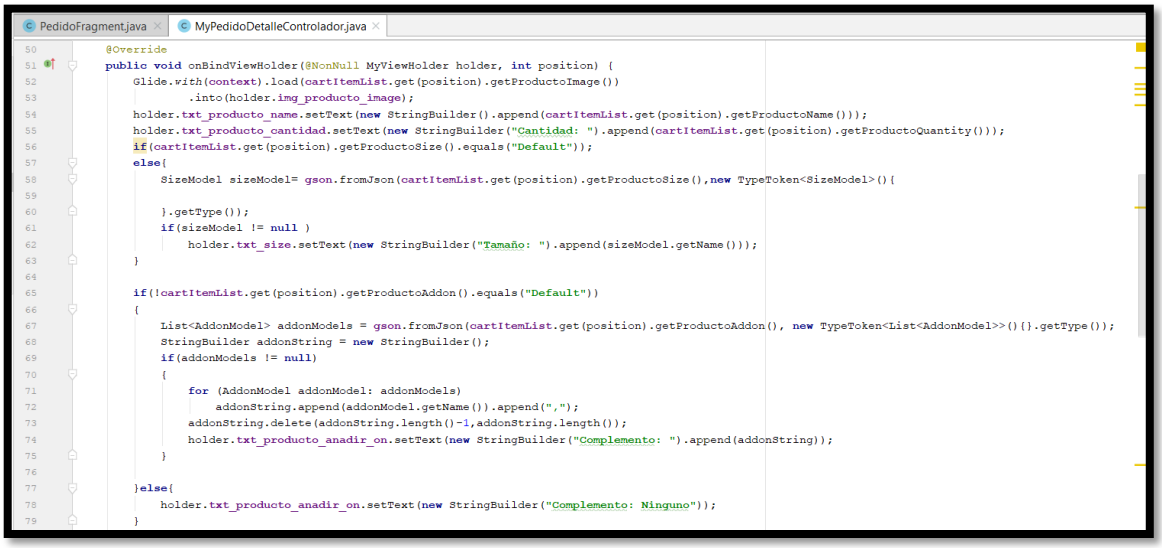
- **Storycard 37 : Modificar pedido en estado Cancelado**

Interfaz	Descripción
	Se puede visualizar un dialogShow para modificar el pedido en estado entregado hacia cancelado.

Siguiendo la lógica del storycard 36, ahora en la sintaxis de código mostrada anteriormente, se muestra el método para poder cambiar al estado del pedido cancelado a atendido o eliminarlo y aquello interactúa con el servicio Firebase para poder cambiar el estado del pedido y remover el pedido actual de en camino a uno de los dos estados seleccionados mediante el controlador que permite interactuar con la data de Firebase.

- **Storycard 38 : Detalle del Pedido**

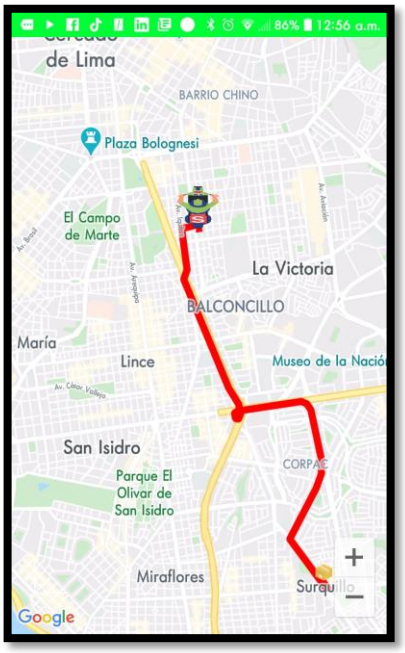
Interfaz	Descripción
	Se puede visualizar el detalle del pedido por producto, cantidad, complemento y tamaño del mismo dando click a cualquier pedido.



```
PedidoFragment.java | MyPedidoDetalleControlador.java

    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        Glide.with(context).load(cartItemList.get(position).getProductoImage())
                .into(holder.img_producto_image);
        holder.txt_producto_name.setText(new StringBuilder().append(cartItemList.get(position).getProductoName()));
        holder.txt_producto_cantidad.setText(new StringBuilder("Cantidad: ").append(cartItemList.get(position).getProductoQuantity()));
        if(cartItemList.get(position).getProductoSize().equals("Default"));
        else{
            SizeModel sizeModel= gson.fromJson(cartItemList.get(position).getProductoSize(),new TypeToken<SizeModel>(){

            }.getType());
            if(sizeModel != null )
                holder.txt_size.setText(new StringBuilder("Tamaño: ").append(sizeModel.getName()));
        }

        if(!cartItemList.get(position).getProductoAddon().equals("Default"))
        {
            List<AddonModel> addonModels = gson.fromJson(cartItemList.get(position).getProductoAddon(), new TypeToken<List<AddonModel>>(){}.getType());
            StringBuilder addonString = new StringBuilder();
            if(addonModels != null)
            {
                for (AddonModel addonModel: addonModels)
                    addonString.append(addonModel.getName()).append(",");
                addonString.delete(addonString.length()-1,addonString.length());
                holder.txt_producto_anadir_on.setText(new StringBuilder("Complemento: ").append(addonString));
            }

        }else{
            holder.txt_producto_anadir_on.setText(new StringBuilder("Complemento: Ninguno"));
        }
```

En la sintaxis de código mostrada anteriormente, observamos que en el controlador se obtiene los pedidos a través de la obtención de data por el Firebase y el PedidoModel, los complementos están dentro de un arreglo ya pueden contemplar de uno a muchos complementos, extrayendo la información del pedido solicitado por el cliente.

- **Storycard 39 : Geolocalización del trayecto del pedido del módulo administrador**

Interfaz	Descripción
	Se puede visualizar el servicio de Google Maps mostrada para visualizar el trayecto en tiempo en real del pedido del cliente marcándose por una línea roja del repartidor por pedido así podrá controlar el pedido del cliente.

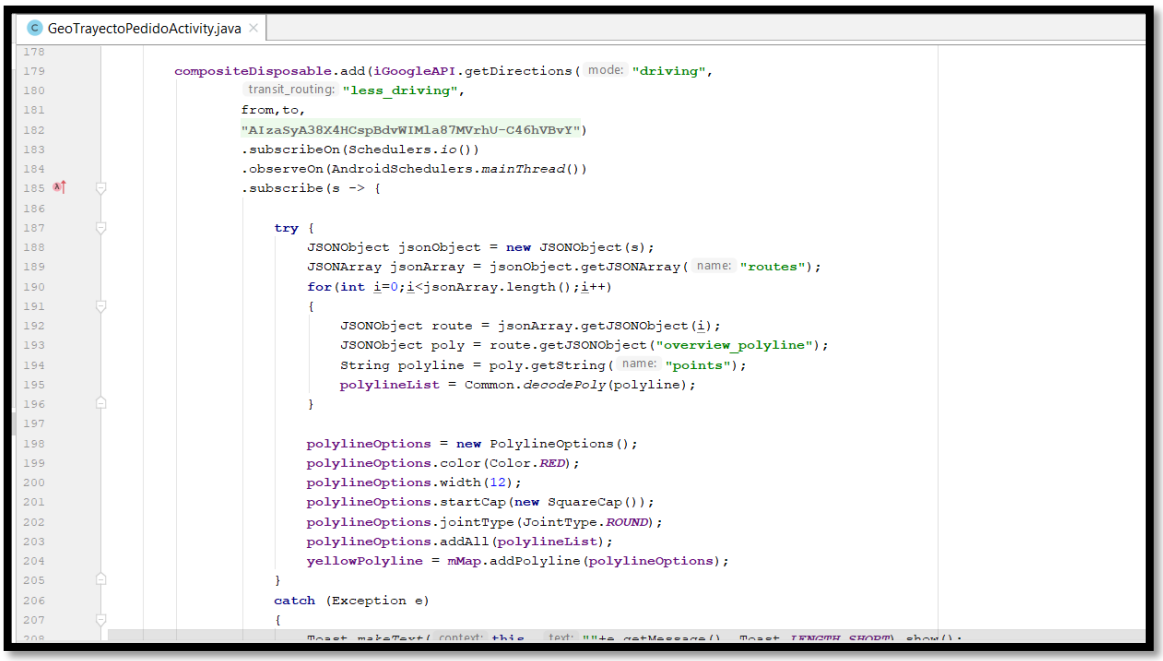
```
PedidoFragment.java

private void createCaminoPedido(int pos,RepartidorModel repartidorModel, PedidoModel pedidoModel, AlertDialog dialog) {
    CaminoPedido caminoPedido = new CaminoPedido();
    caminoPedido.setRepartidorPhone(repartidorModel.getPhone_repartidor());
    caminoPedido.setRepartidorName(repartidorModel.getName_repartidor());
    caminoPedido.setPedidoModel(pedidoModel);
    caminoPedido.setStartTrip(false);
    caminoPedido.setCurrentLat(-1.0);
    caminoPedido.setCurrentLng(-1.0);

    FirebaseDatabase.getInstance() FirebaseDatabase
            .getReference(Common.CAMINO_PEDIDO_REF) DatabaseReference
            .child(pedidoModel.getKey())//cambiar push .push()
            .setValue(caminoPedido) Task<Void>
            .addOnFailureListener(e -> {
                dialog.dismiss();
                Toast.makeText(getContext(), text: "" + e.getMessage(), Toast.LENGTH_SHORT).show();
            }) Task<Void>
            .addOnCompleteListener(task -> {
                if(task.isSuccessful())
                {
                    dialog.dismiss();
                    FirebaseDatabase.getInstance() FirebaseDatabase
                            .getReference(Common.TOKEN_REF) DatabaseReference
                            .child(repartidorModel.getKey_repartidor()) DatabaseReference
                            .addListenerForSingleValueEvent(new ValueEventListener() {
                                @Override
                                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                                    if(dataSnapshot.exists())
                                    {
```

En la primera sintaxis de código mostrada anteriormente, observamos en el método createCaminoPedido, al asignar un pedido a un repartidor se crea cierta información con la tabla CaminoPedido de firebase que va almacenar el nombre, teléfono del cliente y repartidor, latitud y longitud de la dirección a recorrer.

En la segunda sintaxis de código mostrada anteriormente,observamos en el método checkPedidoFromFirebase(), la instancia de la base de datos NoSQL para traer la interfaz de la ubicación actual del pedido por la tabla caminoPedido obteniendo la data mediante el dataSnapshot y ayudándose del método iSingleTrayectoPedidoLoadCallbackListener.

En la tercera sintaxis de código mostrada anteriormente, observamos en el método onSingleTrayectoPedidoLoadSucess, es la carga del google maps

mediante la solicitud de pedido en firebase por latitud y longitud del pedido, y la visualización del motorizado acompañado mediante la geolocalización.

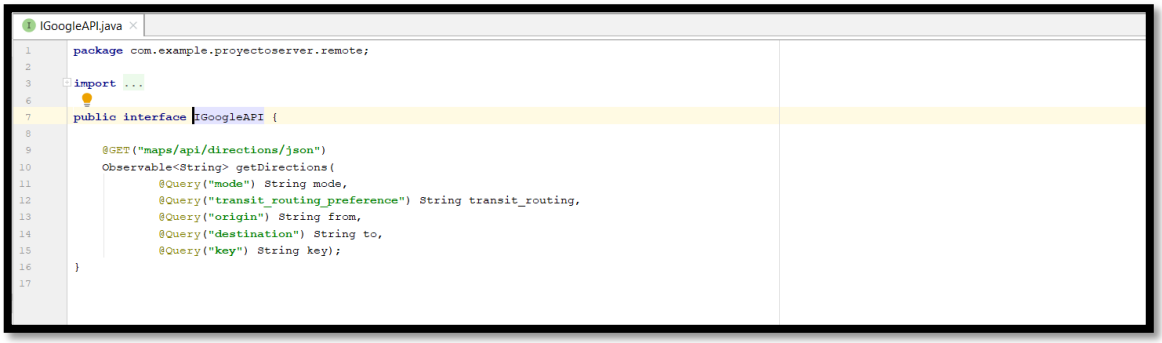


```
GeoTrayectoPedidoActivity.java

        compositeDisposable.add(iGoogleAPI.getDirections( mode: "driving",
                transit_routing: "less_driving",
                from,to,
                "AIzaSyA38X4HCspBdvWIMla87MVrhU-C46hVBvY")
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(s -> {

                    try {
                        JSONObject jsonObject = new JSONObject(s);
                        JSONArray jsonArray = jsonObject.getJSONArray( name: "routes");
                        for(int i=0;i<jsonArray.length();i++)
                        {
                            JSONObject route = jsonArray.getJSONObject(i);
                            JSONObject poly = route.getJSONObject("overview_polyline");
                            String polyline = poly.getString( name: "points");
                            polylineList = Common.decodePoly(polyline);
                        }

                        polylineOptions = new PolylineOptions();
                        polylineOptions.color(Color.RED);
                        polylineOptions.width(12);
                        polylineOptions.startCap(new SquareCap());
                        polylineOptions.jointType(JointType.ROUND);
                        polylineOptions.addAll(polylineList);
                        yellowPolyline = mMap.addPolyline(polylineOptions);
                    }
                    catch (Exception e)
                    {
```

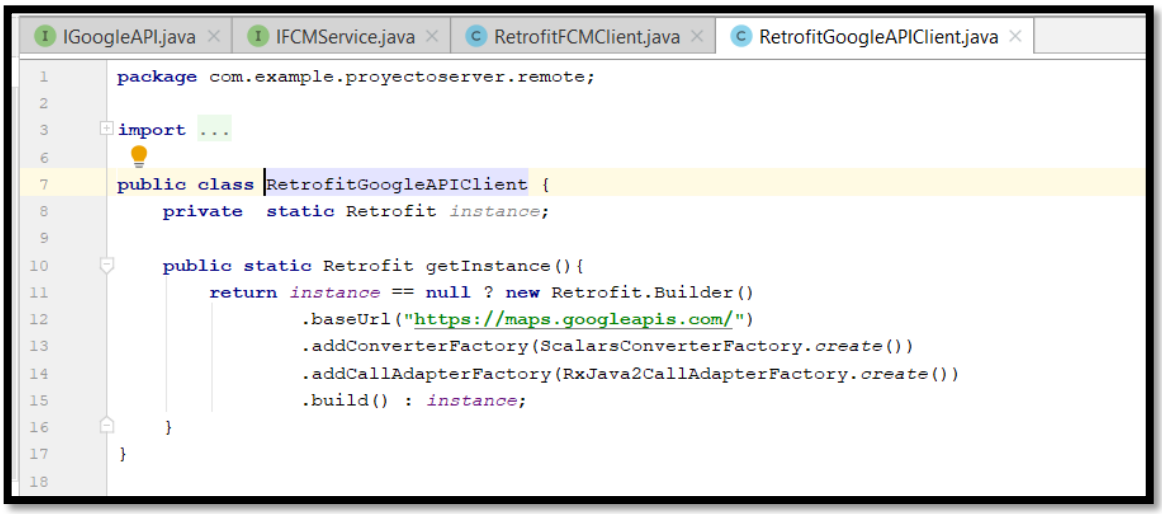
En la cuarta sintaxis de código mostrada anteriormente, observamos la subscripción al api de Google Maps mediante un objeto JSON para poder visualizar la ruta roja, de la ubicación actual del repartidor hasta la ruta destinataria del pedido acompañado por el método repetitivo for para recorrer el pedido que forma la línea roja, la librería poly.

```
GeoTrayectoPedidoActivity.java

    private void moveMakerAnimation(Marker repartidorMarker, String from, String to) {
        compositeDisposable.add(iGoogleAPI.getDirections( mode: "driving",
                transit_routing: "less_driving",
                from,to,
                "AIzaSyA38X4HCspBdvWIMla87MVrhU-C46hVBvY")
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(returnResult -> {

                    try {
                        JSONObject jsonObject = new JSONObject(returnResult);
                        JSONArray jsonArray = jsonObject.getJSONArray( name: "routes");
                        for(int i=0;i<jsonArray.length();i++)
                        {
                            JSONObject route = jsonArray.getJSONObject(i);
                            JSONObject poly = route.getJSONObject("overview_polyline");
                            String polyline = poly.getString( name: "points");
                            polylineList = Common.decodePoly(polyline);
                        }

                        polylineOptions = new PolylineOptions();
                        polylineOptions.color(Color.GRAY);
                        polylineOptions.width(12);
                        polylineOptions.startCap(new SquareCap());
                        polylineOptions.jointType(JointType.ROUND);
                        polylineOptions.addAll(polylineList);
                        grayPolyline = mMap.addPolyline(polylineOptions);

                        blackPolylineOptions= new PolylineOptions();
                        blackPolylineOptions.color(Color.BLACK);
```

En la quinta sintaxis de código mostrada anteriormente, observamos el método para que el administrador pueda ver el movimiento en tiempo real del vehículo del repartidor mediante una api instancia a GoogleMapsSDKPlatform y un arreglo para visualizar la línea roja de la ubicación actual del repartidor hasta el destinatario del repartidor.

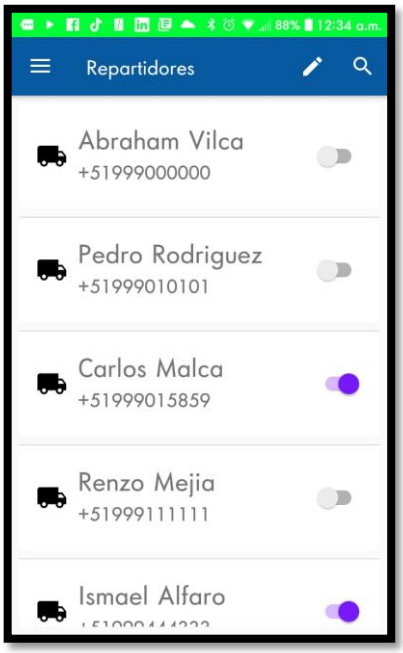


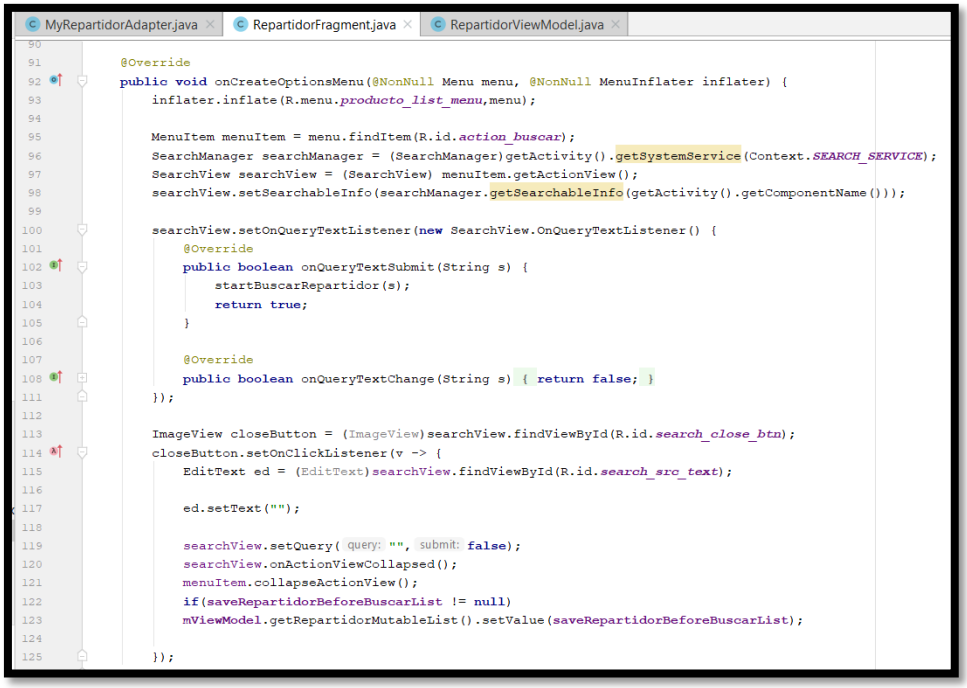
En la sexta sintaxis de código mostrada anteriormente, observamos la interface clase para instanciar las direcciones de Google Maps con paquetería json para elaborar la ruta del repartidor hasta la ruta destinataria.



En la séptima sintaxis de código mostrada anteriormente, observamos la clase RetrofitGoogleAPIClient para poder hacer llamadas de red de GoogleMaps Api y obtener el resultado estructurado de la ubicación real del pedido mediante un HttpClient.

- **Storycard 40 : Listar Repartidores**

Interfaz	Descripción
	Se puede visualizar la lista de repartidores, se observa un SwitchCompat para activar el uso e inicio de la aplicación móvil en perfil repartidor, en la parte superior se visualizar una lupa que servirá para buscar al repartidor que se desea buscar.



```
@Override
public void onCreateOptionsMenu(@NonNull Menu menu, @NonNull MenuInflater inflater) {
    inflater.inflate(R.menu.producto_list_menu,menu);

    MenuItem menuItem = menu.findItem(R.id.action_buscar);
    SearchManager searchManager = (SearchManager)getActivity().getSystemService(Context.SEARCH_SERVICE);
    SearchView searchView = (SearchView) menuItem.getActionView();
    searchView.setSearchableInfo(searchManager.getSearchableInfo(getActivity().getComponentName()));

    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String s) {
            startBuscarRepartidor(s);
            return true;
        }

        @Override
        public boolean onQueryTextChange(String s) { return false; }
    });

    ImageView closeButton = (ImageView)searchView.findViewById(R.id.search_close_btn);
    closeButton.setOnClickListener(v -> {
        EditText ed = (EditText)searchView.findViewById(R.id.search_src_text);

        ed.setText("");

        searchView.setQuery( query: "", submit: false);
        searchView.onActionViewCollapsed();
        menuItem.collapseActionView();
        if(saveRepartidorBeforeBuscarList != null)
        mViewModel.getRepartidorMutableList().setValue(saveRepartidorBeforeBuscarList);

    });
```

En la primera sintaxis de código mostrada anteriormente, , observamos la interacción del diseño del layout_repartidor con el con el método onCreateOptionsMenu para la muestra de datos del Firebase para visualizar la data cargada alojada.

```
28      private void startBuscarRepartidor(String s) {
29          List<RepartidorModel> resultRepartidor = new ArrayList<>();
30          for(int i=0;i<repartidorModelList.size();i++)
31          {
32              RepartidorModel repartidorModel = repartidorModelList.get(i);
33              if(repartidorModel.getPhone_repartidor().toLowerCase().contains(s.toLowerCase()))
34              {
35                  resultRepartidor.add(repartidorModel);
36
37              }else if(repartidorModel.getName_repartidor().toLowerCase().contains(s.toLowerCase()))
38              {
39                  resultRepartidor.add(repartidorModel);
40
41              }
42
43          }
44          mViewModel.getRepartidorMutableList().setValue(resultRepartidor);
45      }
46
47      private void initViews() {
48
49          setHasOptionsMenu(true);
50          dialog = new SpotsDialog.Builder().setContext(getContext()).setCancelable(false).build();
51          LinearLayoutManager layoutManager = new LinearLayoutManager(getContext());
52          recycler_repartidor.setLayoutManager(layoutManager);
53          recycler_repartidor.addItemDecoration(new DividerItemDecoration(getContext(),layoutManager.getOrientation()));
54      }
```

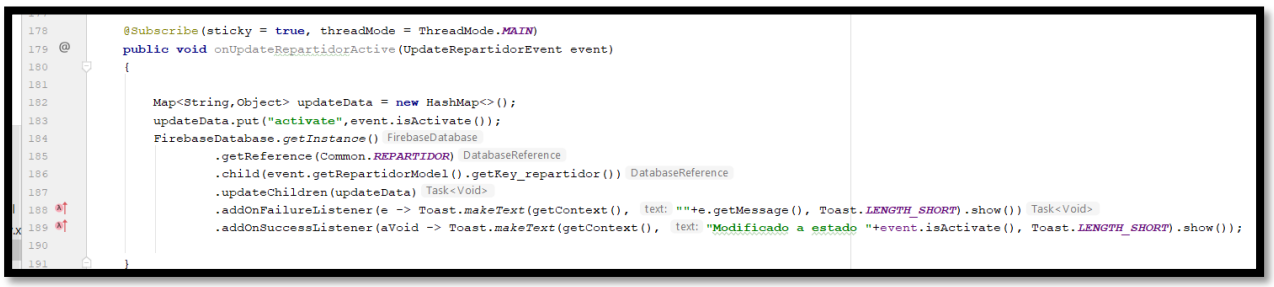
En la segunda sintaxis de código mostrada anteriormente, observamos en el método startBuscarRepartidor, un list para listar a los repartidores buscadosmediante un ArrayList y un método repetitivo for para buscar a los a los repartidores por su nombre o apellido.

MyRepartidorAdapter.java | RepartidorFragment.java | RepartidorViewModel.java

```
44      }
45
46      private void loadRepartidor() {
47          List<RepartidorModel> tempList = new ArrayList<>();
48          DatabaseReference repartidorRef = FirebaseDatabase.getInstance().getReference(Common.REPARTIDOR);
49          repartidorRef.addListenerForSingleValueEvent(new ValueEventListener() {
50              @Override
51              public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
52                  for(DataSnapshot repartidorSnapShot:dataSnapshot.getChildren())
53                  {
54                      RepartidorModel repartidorModel = repartidorSnapShot.getValue(RepartidorModel.class);
55                      repartidorModel.setKey_repartidor(repartidorSnapShot.getKey());
56                      tempList.add(repartidorModel);
57                  }
58                  repartidorLoadCallbackListener.onRepartidorLoadSucess(tempList);
59              }
60
61              @Override
62              public void onCancelled(@NonNull DatabaseError databaseError) {
63                  repartidorLoadCallbackListener.onRepartidorLoadFailed(databaseError.getMessage());
64
65              }
66          });
67      }
68
```

En la tercera sintaxis de código mostrada anteriormente,observamos en el método loadRepartidor, un list para listar a los repartidores mediante un ArrayList

y un método repetitivo for para observar el resultado interactuando con el layout_repartidor.

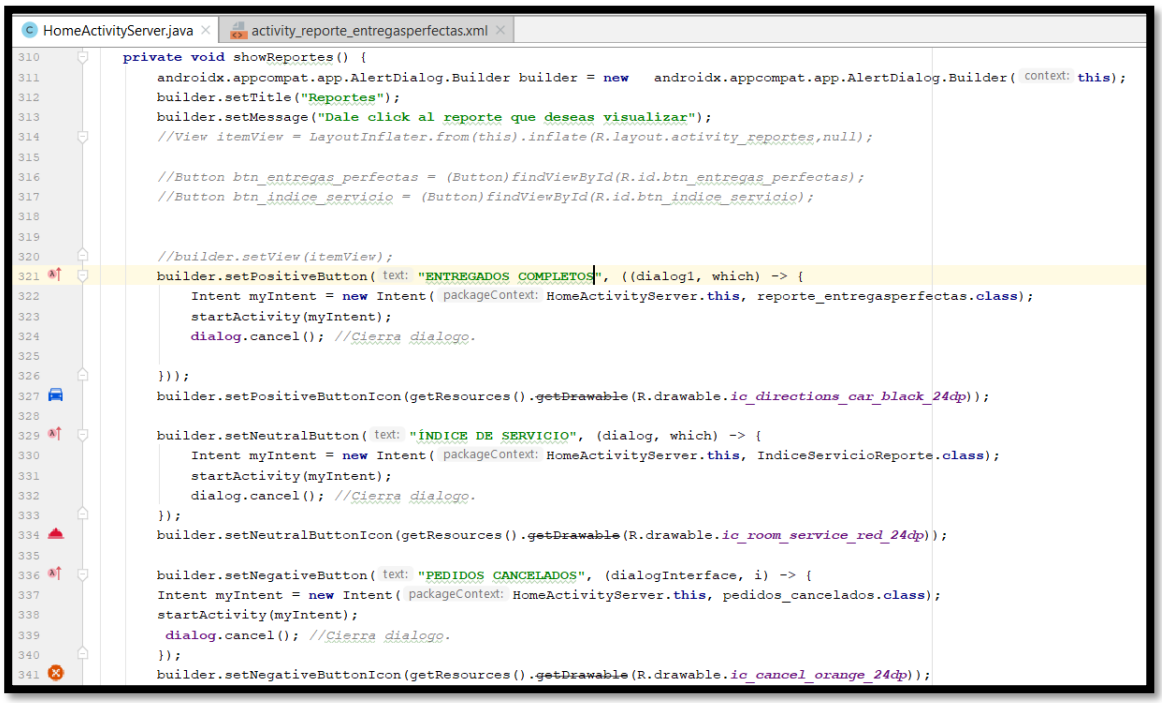


```
178     @Subscribe(sticky = true, threadMode = ThreadMode.MAIN)
179 @   public void onUpdateRepartidorActive(UpdateRepartidorEvent event)
180     {
181
182         Map<String,Object> updateData = new HashMap<>();
183         updateData.put("activate",event.isActivate());
184         FirebaseDatabase.getInstance() FirebaseDatabase
185                 .getReference(Common.REPARTIDOR) DatabaseReference
186                 .child(event.getRepartidorModel().getKey_repartidor()) DatabaseReference
187                 .updateChildren(updateData) Task<Void>
188                 .addOnFailureListener(e -> Toast.makeText(getContext(), text: ""+e.getMessage(), Toast.LENGTH_SHORT).show()) Task<Void>
189                 .addOnSuccessListener(aVoid -> Toast.makeText(getContext(), text: "Modificado a estado "+event.isActivate(), Toast.LENGTH_SHORT).show());
190
191     }
```

En la cuarta sintaxis de código mostrada anteriormente, observamos en el método onUpdateRepartidorActivate para poder configurar el estado del repartidor si es activado o desactivado interactuando con Firebase para cambiar el estado y así controlar su inicio de sesión.

- **Storycard 41 : Menú para acceder a reportes**

Interfaz	Descripción
	Se puede visualizar la lista de reportes a seleccionar, si es por entregados completos, Entregas Perfectamente Recibidas o índice de servicio.



```
HomeActivityServer.java | activity_reporte_entregasperfectas.xml

private void showReportes() {
    androidx.appcompat.app.AlertDialog.Builder builder = new   androidx.appcompat.app.AlertDialog.Builder( context: this);
    builder.setTitle("Reportes");
    builder.setMessage("Dale click al reporte que deseas visualizar");
    //View itemView = LayoutInflater.from(this).inflate(R.layout.activity_reportes,null);

    //Button btn_entregas_perfectas = (Button)findViewById(R.id.btn_entregas_perfectas);
    //Button btn_indice_servicio = (Button)findViewById(R.id.btn_indice_servicio);


    //builder.setView(itemView);
    builder.setPositiveButton( text: "ENTREGADOS COMPLETOS", ((dialog1, which) -> {
        Intent myIntent = new Intent( packageContext: HomeActivityServer.this, reporte_entregasperfectas.class);
        startActivity(myIntent);
        dialog.cancel(); //Cierra dialogo.

    }));
    builder.setPositiveButtonIcon(getResources().getDrawable(R.drawable.ic_directions_car_black_24dp));

    builder.setNeutralButton( text: "ÍNDICE DE SERVICIO", (dialog, which) -> {
        Intent myIntent = new Intent( packageContext: HomeActivityServer.this, IndiceServicioReporte.class);
        startActivity(myIntent);
        dialog.cancel(); //Cierra dialogo.
    });
    builder.setNeutralButtonIcon(getResources().getDrawable(R.drawable.ic_room_service_red_24dp));

    builder.setNegativeButton( text: "PEDIDOS CANCELADOS", (dialogInterface, i) -> {
    Intent myIntent = new Intent( packageContext: HomeActivityServer.this, pedidos_cancelados.class);
    startActivity(myIntent);
     dialog.cancel(); //Cierra dialogo.
    });
    builder.setNegativeButtonIcon(getResources().getDrawable(R.drawable.ic_cancel_orange_24dp));
```

En la primera sintaxis de código mostrada anteriormente, se va visualizar la codificación del AlertDialog para entrar a cada interfaz del reporte: entregados completos, índice de servicio o Entregas Perfectamente Recibidas mediante intent.

- **Storycard 41: Reporte de Entregados Completos**

<table>
<tr><th>Interfaz</th><th>Descripción</th></tr>
<tr><td> 
</td><td>Se puede visualizar la interfaz del reporte de entregados completos seleccionando en un rango de fechas (fecha de inicio y fecha de fin) a través de un DateTimePicker, cuando se hace click al botón Imprimir Reporte, almacena el pdf emitido en una carpeta ubicada en descargas con el reporte previamente seleccionado las fechas y su rango.</td></tr>
</table>

```java
btn_imprimir_reporte.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String firstdate = date.toString();
        String seconddate = date2.toString();
        System.out.println(edt_fecha_inicio.getText().toString()+"-"+edt_fecha_fin.getText().toString());
        System.out.println(firstdate+"-"+seconddate);
        //<<<<<<<<<<<<<<<====================REPORT PDF================

        list_fechas_unique.clear();
        arrays_totales.clear();
        filtrar_pedidos_fecha(edt_fecha_inicio.getText().toString(),edt_fecha_fin.getText().toString());
        //filtrar_pedidos_fecha("2020-06-01","2020-06-01");
        generar_pdf();
        Toast.makeText( context: reporte_entregasperfectas.this,  text: "SE CREO EL PDF", Toast.LENGTH_LONG).show();

        PDFView pdfView = (PDFView) findViewById(R.id.pdfView);
        File file = new File( pathname: Environment.getExternalStorageDirectory().getAbsolutePath() + "/Download/MisPDFs/MiPDF.pdf");
        pdfView.fromFile(file).defaultPage(1).load();
        //<<<<<<<<<<<<<<<====================REPORT PDF================

    }
});
```

En la primera sintaxis de código mostrada anteriormente, se visualizar la codificación la creación de lógica con el botón de “Imprimir Reporte” con el layout de entregados completos para que interactúen entre ellos y la interacción del archivo pdf almacenado en el dispositivo del usuario.

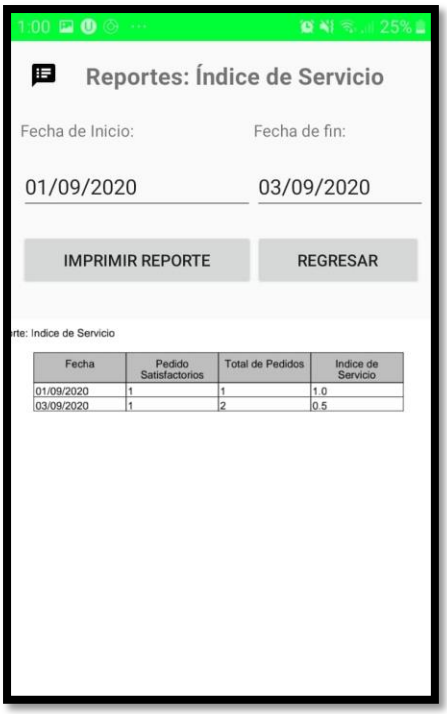
```
public void fecha_pedido_entregado() {
    databaseReference.child("Pedido").orderByChild("pedidoEstado").equalTo(1).addValueEventListener(new ValueEventListener()
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            list_entregados.clear();
            for (DataSnapshot objSnaptshot : dataSnapshot.getChildren()){
                Listpedidos p = objSnaptshot.getValue(Listpedidos.class);
                list_entregados.add(p.getFecha());
            }
            Log.e( tag: "Datos: " , msg: ""+list_entregados);
        }
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}
    });
}
```

En la segunda sintaxis de código mostrada anteriormente, se visualiza la codificación para traer el estado del pedido Entregado que el estado 1 con el método DataSnapshot de Firebase mediante la fecha del mismo al reporte.

```
public void operaciones_pedido(String fecha) {

    ArrayList<String> itemIds = new ArrayList<~>();//Lista todos los pedidos de estado
    ArrayList<String> itemIds2 = new ArrayList<~>();// lista todos los estado

    for (int i=0; i< list_entregados.size(); i++)
    {
        Log.e( tag: "Datos: " , msg: list_entregados+"------------++");
        Log.e( tag: "Datos: " , msg: fecha+"------------+++");
        if((list_entregados.get(i)).equals(fecha)){
            itemIds.add(list_entregados.get(i));
            Log.e( tag: "Datos: " , msg: list_entregados.get(i)+"22");
        }
    }

    Log.e( tag: "Datos: " , msg: itemIds.size()+"--------*****************");
    //================================================
    for (int i=0; i< list_totales.size(); i++)
    {
        if((list_totales.get(i)).equals(fecha)){
            itemIds2.add(list_totales.get(i));
        }
    }

    double rest1 = Double.parseDouble(String.valueOf(itemIds.size()));
    double rest2 = Double.parseDouble(String.valueOf(itemIds2.size()));
    double porcet = (rest1/rest2)*100;
    int result = Integer.valueOf((int) porcet);
```

En la tercera sintaxis de código mostrada anteriormente, se visualizar la codificación para traer los datos extraídos según el rango de fechas mediante un array list, por un método repetitivo for para poder listar y hacer la fórmula correspondiente para dicho indicador.

- **Storycard 42 : Reporte de Índice de Servicio**

Interfaz	Descripción
	Se puede visualizar la interfaz del reporte de índice de servicio seleccionando en un rango de fechas (fecha de inicio y fecha de fin) a través de un DateTimePicker, cuando se hace click al botón Imprimir Reporte, almacena el pdf emitido en una carpeta ubicada en descargas con el reporte previamente seleccionado las fechas y su rango.

En la primera sintaxis de código mostrada anteriormente, se visualizar el método para filtrar pedidos mediante el método repetitivo for recorra las fechas hasta encontrar pedidos con las fechas indicadas y mostrar en el reporte.

```
public void fecha_pedido_total() {
    databaseReference
            .child("Pedido")
            .addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    list_totales.clear();
                    for (DataSnapshot objSnaptshot : dataSnapshot.getChildren()){
                        Listpedidos p = objSnaptshot.getValue(Listpedidos.class);
                        list_totales.add(p.getFecha());
                    }
                    Log.e( tag: "Datos: " , msg: ""+list_totales);
                }
                @Override
                public void onCancelled(@NonNull DatabaseError databaseError) { }
            });
}
```

En la segunda sintaxis de código mostrada anteriormente, se visualiza el método para extraer la lista de pedidos mediante un DataSnapshot del firebase mediante las fechas solicitadas por el administrador.

```
public void generar_pdf() {

    Document documento = new Document();
    try {
        File file = crearFichero(NOMBRE_DOCUMENTO);
        FileOutputStream ficheroPDF = new FileOutputStream(file.getAbsolutePath());
        PdfWriter writer = PdfWriter.getInstance(documento, ficheroPDF);
        documento.open();
        documento.add(new Paragraph( string: "Reporte: Indice de Servicio \n\n"));
        // Insertamos una tabla
        PdfPTable table = new PdfPTable( numColumns: 4);// 5 columas
        // ==============Header row==============
        PdfPCell cell1 = new PdfPCell(new Phrase( string: "Fecha"));
        cell1.setHorizontalAlignment(Element.ALIGN_CENTER);
        cell1.setPadding(5.0f);
        cell1.setBackgroundColor(new GrayColor( floatGray: 0.75f));
        table.addCell(cell1);

        PdfPCell cell2 = new PdfPCell(new Phrase( string: "Pedido Satisfactorios"));
        cell2.setHorizontalAlignment(Element.ALIGN_CENTER);
        cell2.setPadding(5.0f);
        cell2.setBackgroundColor(new GrayColor( floatGray: 0.75f));
        table.addCell(cell2);

        PdfPCell cell3 = new PdfPCell(new Phrase( string: "Total de Pedidos"));
        cell3.setHorizontalAlignment(Element.ALIGN_CENTER);
        cell3.setPadding(5.0f);
        cell3.setBackgroundColor(new GrayColor( floatGray: 0.75f));
        table.addCell(cell3);

        PdfPCell cell4 = new PdfPCell(new Phrase( string: "Indice de Servicio"));
        cell4.setHorizontalAlignment(Element.ALIGN_CENTER);
        cell4.setPadding(5.0f);
        cell4.setBackgroundColor(new GrayColor( floatGray: 0.75f));
        table.addCell(cell4);
```

En la tercera sintaxis de código mostrada anteriormente, se visualiza el método generar_pdf, descargar de un fichero ajustando las celdas con los datos para generar el reporte del índice de servicio.

- **Storycard 43 : Reporte de Entregas Perfectamente Recibidas**

Interfaz	Descripción
	Se puede visualizar la interfaz del reporte de Entregas Perfectamente Recibidas, seleccionando en un rango de fechas (fecha de inicio y fecha de fin) a través de un DateTimePicker, cuando se hace click al botón Imprimir Reporte, almacena el pdf emitido en una carpeta ubicada en descargas con el reporte previamente seleccionado las fechas y su rango.

```
public void fecha_pedido_entregado() {
    databaseReference.child("Pedido").orderByChild("pedidoEstado").equalTo(0).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            list_entregados.clear();
            for (DataSnapshot objSnaptshot : dataSnapshot.getChildren()){
                Listpedidos p = objSnaptshot.getValue(Listpedidos.class);
                list_entregados.add(p.getFecha());
            }
            Log.e( tag: "Datos: " , msg: ""+list_entregados);
        }
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}
    });
}
```

En la primera sintaxis de código mostrada anteriormente, se visualiza la codificación de la creación de lógica para visualizar los pedidos según el estado 0 que es el cancelado mediante el DataSnapshot para extraer la data para listar los pedidos mediante el rango de fechas indicados.

pedidos_cancelados.java

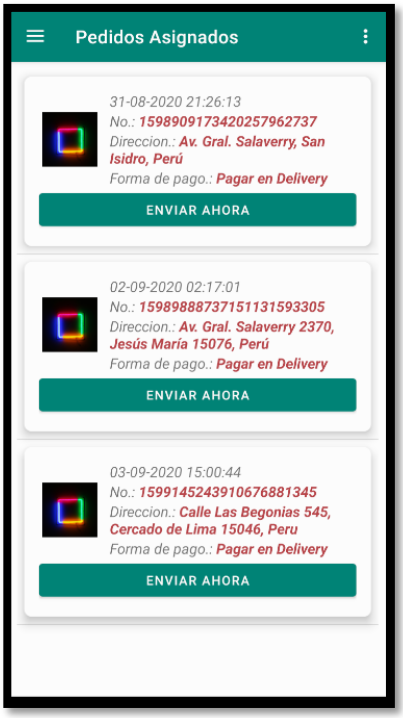
```
265     public void operaciones_pedido(String fecha) {
266
267         ArrayList<String> itemIds = new ArrayList<~>();//Lista todos los pedidos de estado 1
268         ArrayList<String> itemIds2 = new ArrayList<~>();// lista todos los estado
269
270         for (int i=0; i< list_entregados.size(); i++)
271         {
272             Log.e( tag: "Datos: " , msg: list_entregados+"------------++");
273             Log.e( tag: "Datos: " , msg: fecha+"------------+++");
274             if((list_entregados.get(i)).equals(fecha)){
275                 itemIds.add(list_entregados.get(i));
276                 Log.e( tag: "Datos: " , msg: list_entregados.get(i)+"22");
277             }
278         }
279
280         Log.e( tag: "Datos: " , msg: itemIds.size()+"--------***************");
281         //=================================================
282         for (int i=0; i< list_totales.size(); i++)
283         {
284             if((list_totales.get(i)).equals(fecha)){
285                 itemIds2.add(list_totales.get(i));
286             }
287         }
288
289         double rest1 = Double.parseDouble(String.valueOf(itemIds.size()));
290         double rest2 = Double.parseDouble(String.valueOf(itemIds2.size()));
291         double porcet = (rest1/rest2)*100;
292         int result = Integer.valueOf((int) porcet);
293
294 //          Log.e("Datos: " , ""+fecha);
295 //          Log.e("Datos: " , ""+itemIds.size());
296 //          Log.e("Datos: " , ""+itemIds2.size());
297 //          Log.e("Datos: " , result+"%");
298
299         arrays_totales.add(fecha);
300         arrays_totales.add(String.valueOf((itemIds.size())));
301         arrays_totales.add(String.valueOf(itemIds2.size()));
302         arrays_totales.add(result+"%");
```

En la segunda sintaxis de código mostrada anteriormente, se visualiza la codificación para hacer un arrayList para listar los pedidos por fecha por un método repetitivo “for” y así hacer el cálculo correspondiente para las Entregas Perfectamente Recibidas.

```
304     }
305
306     public void generar_pdf() {
307
308         Document documento = new Document();
309         try {
310             File file = crearFichero(NOMBRE_DOCUMENTO);
311             FileOutputStream ficheroPDF = new FileOutputStream(file.getAbsolutePath());
312             PdfWriter writer = PdfWriter.getInstance(documento, ficheroPDF);
313             documento.open();
314             documento.add(new Paragraph( string: "Reporte: Pedidos Cancelados \n\n"));
315             // Insertamos una tabla
316             PdfPTable table = new PdfPTable( numColumns: 4);// 5 columas
317             // ==============Header row==============
318             PdfPCell cell1 = new PdfPCell(new Phrase( string: "Fecha"));
319             cell1.setHorizontalAlignment(Element.ALIGN_CENTER);
320             cell1.setPadding(5.0f);
321             cell1.setBackgroundColor(new GrayColor( floatGray: 0.75f));
322             table.addCell(cell1);
323
324             PdfPCell cell2 = new PdfPCell(new Phrase( string: "Pedidos Cancelados"));
325             cell2.setHorizontalAlignment(Element.ALIGN_CENTER);
326             cell2.setPadding(5.0f);
327             cell2.setBackgroundColor(new GrayColor( floatGray: 0.75f));
328             table.addCell(cell2);
329
330             PdfPCell cell3 = new PdfPCell(new Phrase( string: "Total Entregas"));
331             cell3.setHorizontalAlignment(Element.ALIGN_CENTER);
332             cell3.setPadding(5.0f);
333             cell3.setBackgroundColor(new GrayColor( floatGray: 0.75f));
334             table.addCell(cell3);
335
336             PdfPCell cell4 = new PdfPCell(new Phrase( string: "Porcentaje de Pedidos Cancelados"
337             cell4.setHorizontalAlignment(Element.ALIGN_CENTER);
338             cell4.setPadding(5.0f);
339             cell4.setBackgroundColor(new GrayColor( floatGray: 0.75f));
340             table.addCell(cell4);
```

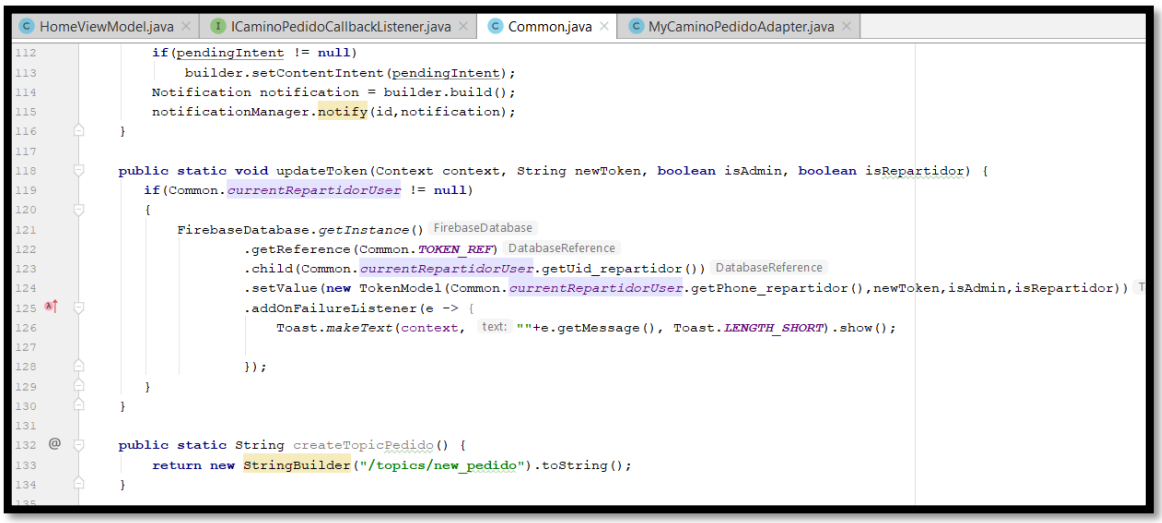
En la tercera sintaxis de código mostrada anteriormente, se visualiza la codificación para registrar los datos emitidos por el dataSnapshot del Firebase y así observa los pedidos y su porcentaje en el rango seleccionado.

- **Storycard 44: Listar Pedidos Asignados por Repartidor**

Interfaz	Descripción
	Se puede visualizar la interfaz de la lista de pedidos asignados por el repartidor visualizándose en cada recuadro, la fecha, hora, número, dirección y modo de envío, al hace click en el botón “Enviar Ahora” se va visualizar el mapa de la ubicación actual del repartidor hasta el destinatario del cliente, si el repartidor ya tiene asignado un pedido en camino, por más quiera iniciar un pedido nuevo sin completar el actual, le arrojará el pedido que tiene que entregar.

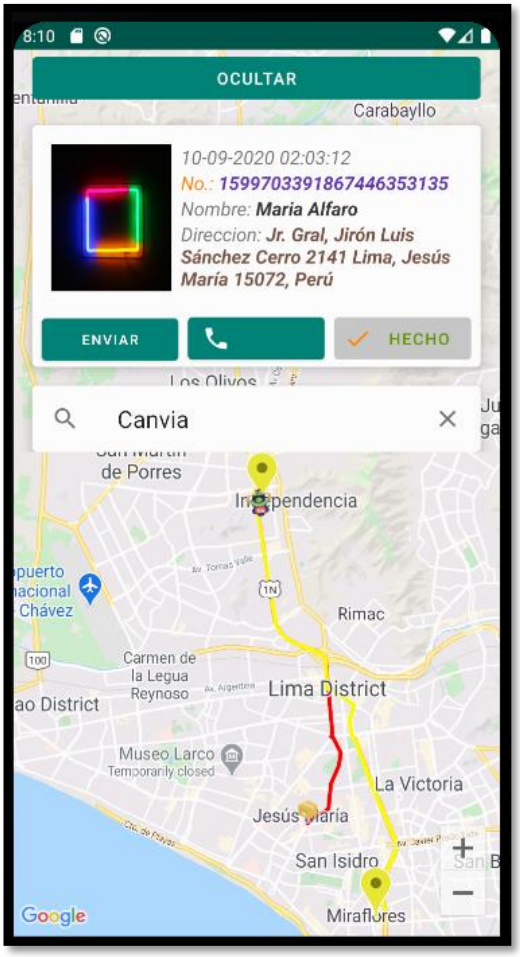

```
private void loadPedidoByRepartidor(String repartidorPhone) {
    List<CaminoPedidoModel> tempList = new ArrayList<>();
    Query pedidoRef = FirebaseDatabase.getInstance().getReference(Common.CAMINO_PEDIDO_REF)
            .orderByChild("repartidorPhone")
            .equalTo(Common.currentRepartidorUser.getPhone_repartidor());
    pedidoRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
           for(DataSnapshot pedidoSnapshot:dataSnapshot.getChildren())
           {
               CaminoPedidoModel  caminoPedidoModel = pedidoSnapshot.getValue(CaminoPedidoModel.class);
               caminoPedidoModel.setKey(pedidoSnapshot.getKey());
               tempList.add(caminoPedidoModel);
           }
           listener.onCaminoPedidoLoadSucess(tempList);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            listener.onCaminoPedidoLoadFailed(databaseError.getMessage());

        }
    });
}
```

En la primera sintaxis de código mostrada anteriormente, se visualizar la codificación para la carga de pedidos por repartidor, clasificando por la tabla “Repartidores” de la columna “repartidorPhone”, lo relacionan con la tabla “CaminoPedido” para poder encontrar las coincidencias del número del repartidor con los pedidos que se le asignó mediante la DataSnapshot para obtener los pedidos.



En la segunda sintaxis de código mostrada anteriormente, se visualizar la codificación para la carga de pedidos por repartidor por el método updateToken por el token del repartidor para poder coincidir los pedidos asignados por el repartidor.

- **Storycard 45: Geolocalización de la ruta actual del perfil del repartidor hacia el destinario del pedido**

Interfaz	Descripción
	Se puede visualizar la interfaz del servicio de api de Google maps marcando con una línea roja la ubicación actual del repartidor, el repartidor le da click al botón de “Enviar” para poder comenzar el transcurso del pedido y así que tanto como el administrador y el usuario visualice la localización real del pedido, en el botón para llamar al repartidor de forma inmediata activando previamente el permiso para realizar llamadas, en el botón de Hecho es para indicar que el pedido ha sido entregado completamente notificando al usuario sobre el pedido con el código y celular del repartidor, en el botón ocultar lo que hará es ocultar la barra de datos del cliente y la caja de texto de buscar, en la caja de texto Search, lo que va ser es que el repartidor digite la dirección que en caso le indicará una nueva el mismo cliente, por el servicio de Google places Maps api que se traza por una línea amarilla.

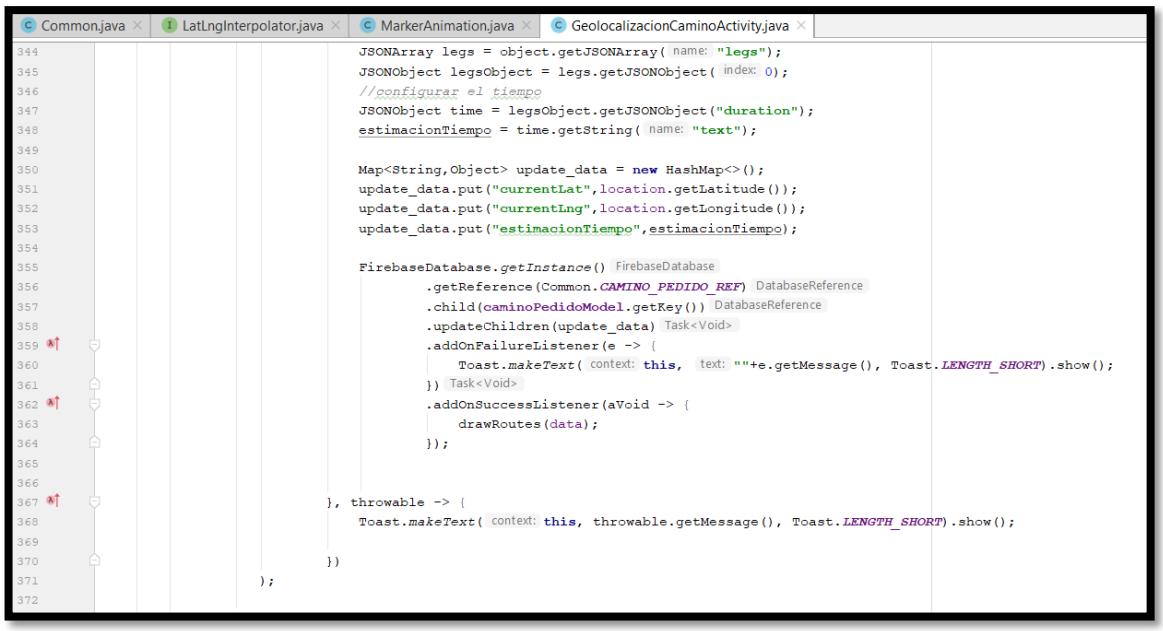
```java
    @OnClick(R.id.btn_ver)
    void onVerClick() {
        if (expandable_layout.isExpanded())
            btn_ver.setText("VER COMPLETO");
        else
            btn_ver.setText("OCULTAR");
        expandable_layout.toggle();
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    @OnClick(R.id.btn_llamar)
    void onCallClick() {
        if (caminoPedidoModel != null) {

            if (checkSelfPermission(Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
//necesita permis
                Dexter.withActivity(this)
                        .withPermission(Manifest.permission.CALL_PHONE)
                        .withListener(new PermissionListener() {
                            @Override
                            public void onPermissionGranted(PermissionGrantedResponse response) {

                            }

                            @Override
                            public void onPermissionDenied(PermissionDeniedResponse response) {
                                Toast.makeText( context: GeolocalizacionCaminoActivity.this,  text: "Debes activar el permiso para llamar", Toast.LENGTH_SH

                            }
```

En la primera sintaxis de código mostrada anteriormente, se visualiza la codificación para poder ocultar y mostrar la ventana de datos del destino del pedido y el permiso para llamar al cliente por el aplicativo mediante la tabla obtenida por Firebase que es caminoPedido, el teléfono del usuario.

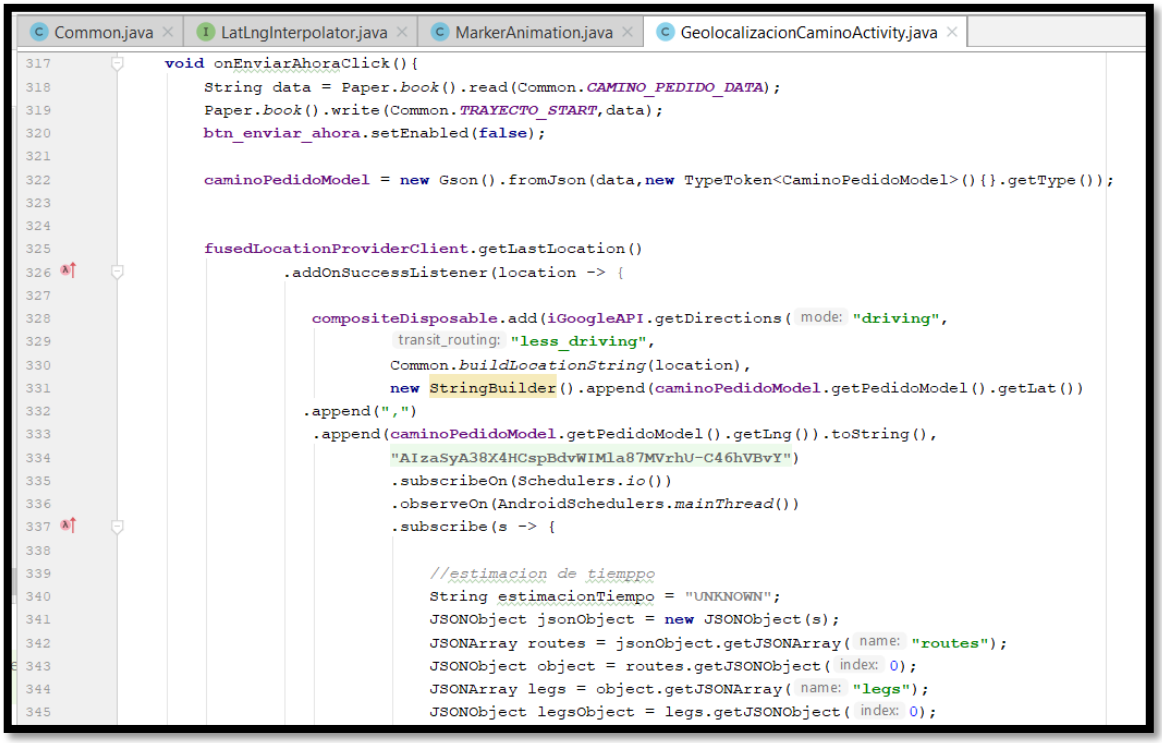
```java
Common.java | LatLngInterpolator.java | MarkerAnimation.java | GeolocalizacionCaminoActivity.java

    @OnClick(R.id.btn_hecho)
    void onHechoClick() {
        if(caminoPedidoModel != null)
        {
            AlertDialog.Builder builder = new AlertDialog.Builder( context: this)
                    .setTitle("Confirmar entrega completa")
                    .setMessage("Confirmar que realmente se ha entregado el pedido correctamente")
                    .setNegativeButton( text: "CANCELAR", (dialogInterface, i) -> dialogInterface.dismiss())
                    .setPositiveButton( text: "SI", (dialogInterface, i) -> {

                        AlertDialog dialog = new AlertDialog.Builder( context: this)
                                .setCancelable(false)
                                .setMessage("Esperando..")
                                .create();

                        Map<String,Object> update_data = new HashMap<>();
                        update_data.put("pedidoEstado", 1);
                        update_data.put("uid_repartidor",Common.currentRepartidorUser.getUid_repartidor());

                        FirebaseDatabase.getInstance() FirebaseDatabase
                                .getReference(Common.PEDIDO_REF) DatabaseReference
                                .child(caminoPedidoModel.getPedidoModel().getKey()) DatabaseReference
                                .updateChildren(update_data) Task<Void>
                                .addOnFailureListener(e -> Toast.makeText( context: GeolocalizacionCaminoActivity.this, e.getMessage(), Toast.LENGTH_SHO
                                .addOnSuccessListener(aVoid -> {

                                    //eliminar pedido
                                    FirebaseDatabase.getInstance() FirebaseDatabase
                                            .getReference(Common.CAMINO_PEDIDO_REF) DatabaseReference
                                            .child(caminoPedidoModel.getPedidoModel().getKey()) DatabaseReference
                                            .removeValue() Task<Void>
```


```
.child(caminoPedidoModel.getPedidoModel().getKey())
.removeValue()
.addOnFailureListener(e -> Toast.makeText( context: GeolocalizacionCaminoActivity.this, e.getMessage(), Toast.LENGTH_SHORT).show())
.addOnSuccessListener(aVoid1 -> {

    //elimina pedido
    //avisa notificacion al usuario
    FirebaseDatabase.getInstance()
            .getReference(Common.TOKEN_REF)
            .child(caminoPedidoModel.getPedidoModel().getUserId())
            .addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                    if(dataSnapshot.exists())
                    {
                        TokenModel tokenModel = dataSnapshot.getValue(TokenModel.class);
                        Map<String,String> notiData = new HashMap<>();
                        notiData.put(Common.NOTI_TITLE,"Tu pedido ha sido entregado satisfactoriamente");
                        notiData.put(Common.NOTI_CONTENT,new StringBuilder("Ha sido entregado por  ")
                                .append(Common.currentRepartidorUser.getPhone_repartidor()).toString());

                        FCMSendData sendData = new FCMSendData(tokenModel.getToken(),notiData);

                        compositeDisposable.add(ifcmService.sendNotification(sendData)
                                .subscribeOn(Schedulers.io())
                                .observeOn(AndroidSchedulers.mainThread())
                                .subscribe(fcmResponse -> {
                                    dialog.dismiss();
                                    if(fcmResponse.getSuccess() == 1)
                                    {
                                        Toast.makeText( context: GeolocalizacionCaminoActivity.this,  text: "Finalizado el pedido", Toast.LENGTH_SHORT
                                    }else{
                                        Toast.makeText( context: GeolocalizacionCaminoActivity.this,  text: "Pedido modificado satisfactoriamente pero
                                    }
```

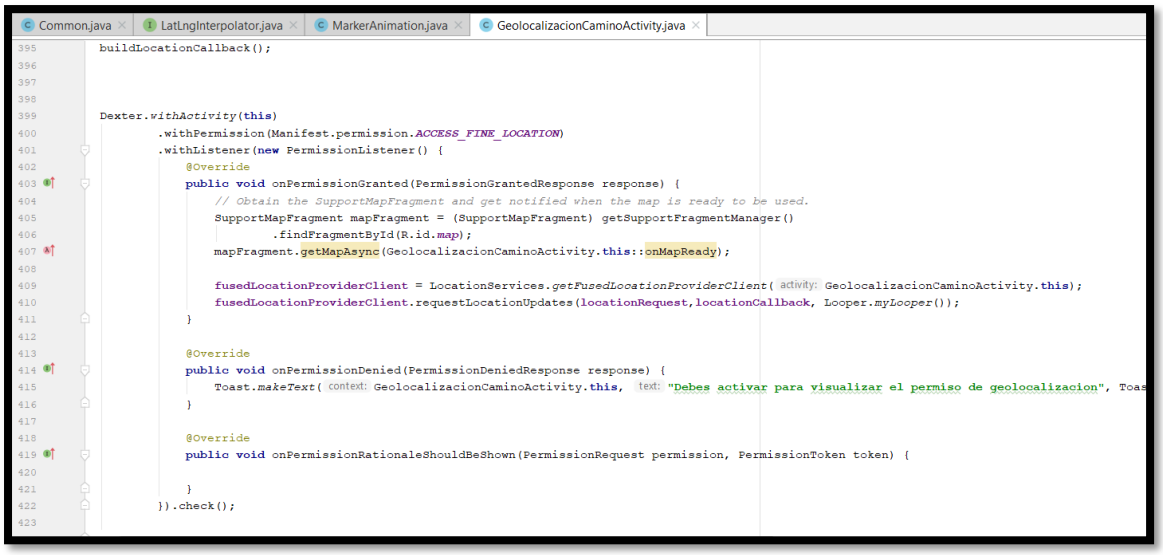
En la segunda y tercera sintaxis de código mostrada anteriormente, se visualiza la codificación para poder terminar el pedido entregado completo, cambiándolo al estado “1” que es el estado terminado, lo cual el método FirebaseDatabse.getReference va actualizar el estado de pedido del cliente de “2” (en camino), a “1”(completo), notificando al cliente sobre su entrega completa y eliminar de la lista de pedidos el pedido ya asignado.



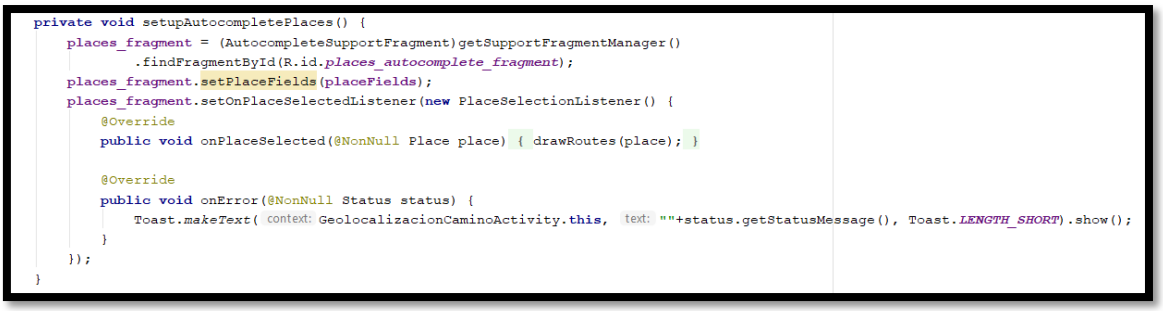

```
                    JSONArray legs = object.getJSONArray( name: "legs");
                    JSONObject legsObject = legs.getJSONObject( index: 0);
                    //configurar el tiempo
                    JSONObject time = legsObject.getJSONObject("duration");
                    estimacionTiempo = time.getString( name: "text");

                    Map<String,Object> update_data = new HashMap<>();
                    update_data.put("currentLat",location.getLatitude());
                    update_data.put("currentLng",location.getLongitude());
                    update_data.put("estimacionTiempo",estimacionTiempo);

                    FirebaseDatabase.getInstance()
                            .getReference(Common.CAMINO_PEDIDO_REF)
                            .child(caminoPedidoModel.getKey())
                            .updateChildren(update_data)
                            .addOnFailureListener(e -> {
                                Toast.makeText( context: this,  text: ""+e.getMessage(), Toast.LENGTH_SHORT).show();
                            })
                            .addOnSuccessListener(aVoid -> {
                                drawRoutes(data);
                            });


                }, throwable -> {
                    Toast.makeText( context: this, throwable.getMessage(), Toast.LENGTH_SHORT).show();

                })
        );
```



En la cuarta y quinta sintaxis de código mostrada anteriormente, se visualiza la codificación para comenzar el trayecto del pedido mediante la geolocalización por el api de Google maps, calculando la estimación de tiempo para que pueda verse en la interfaz del cliente, el tiempo referencial del pedido mediante un objecto JSON, permitiendo que mediante el FirebaseDatabse y Map<String,Object>, se visualize la ruta real del repartidor en los mapas del administrador y usuario.

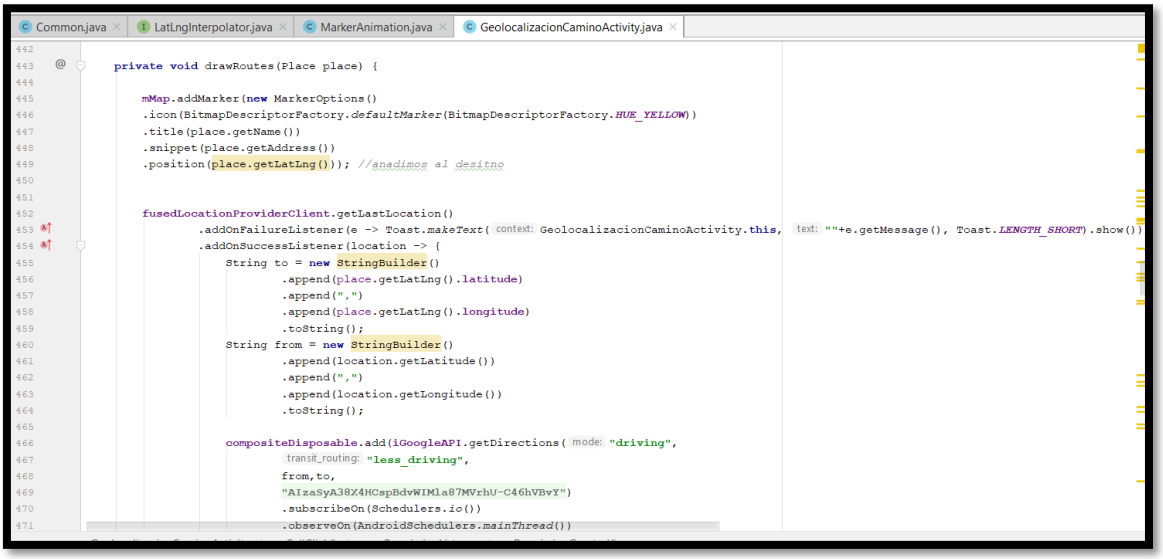


En la sexta sintaxis de código mostrada anteriormente, se visualizar la codificación para el permiso de acceder la ubicación del repartidor y al servid



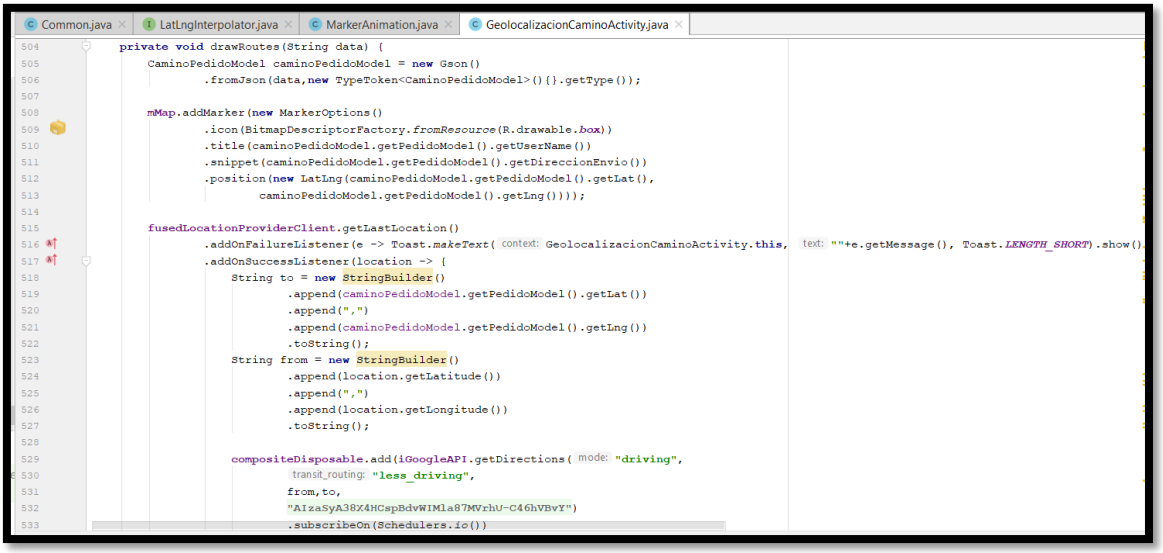
```
private void setupAutocompletePlaces() {
    places_fragment = (AutocompleteSupportFragment)getSupportFragmentManager()
            .findFragmentById(R.id.places_autocomplete_fragment);
    places_fragment.setPlaceFields(placeFields);
    places_fragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
        @Override
        public void onPlaceSelected(@NonNull Place place) { drawRoutes(place); }

        @Override
        public void onError(@NonNull Status status) {
            Toast.makeText( context: GeolocalizacionCaminoActivity.this,  text: ""+status.getStatusMessage(), Toast.LENGTH_SHORT).show();
        }
    });
}
```

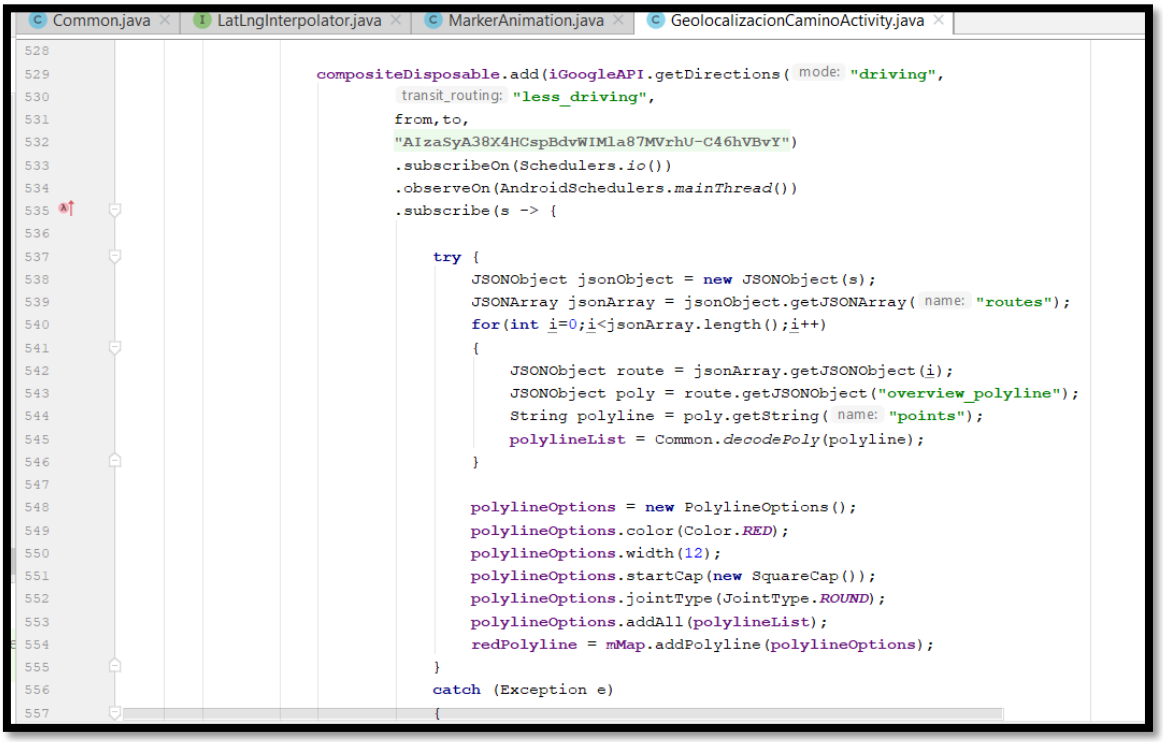
En la séptima sintaxis de código mostrada anteriormente, se visualizar la codificación para la carga de la ubicación alterna digitada y ubicada correctamente por el Google Places Maps trazando una línea amarilla conjunto con el método dwawRoutes(place) para trazar la ruta.



```
Common.java | LatLngInterpolator.java | MarkerAnimation.java | GeolocalizacionCaminoActivity.java

442
443     private void drawRoutes(Place place) {
444
445         mMap.addMarker(new MarkerOptions()
446         .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_YELLOW))
447         .title(place.getName())
448         .snippet(place.getAddress())
449         .position(place.getLatLng())); //anadimos al desitno
450
451
452         fusedLocationProviderClient.getLastLocation()
453                 .addOnFailureListener(e -> Toast.makeText( context: GeolocalizacionCaminoActivity.this,  text: ""+e.getMessage(), Toast.LENGTH_SHORT).show())
454                 .addOnSuccessListener(location -> {
455                     String to = new StringBuilder()
456                             .append(place.getLatLng().latitude)
457                             .append(",")
458                             .append(place.getLatLng().longitude)
459                             .toString();
460                     String from = new StringBuilder()
461                             .append(location.getLatitude())
462                             .append(",")
463                             .append(location.getLongitude())
464                             .toString();
465
466                     compositeDisposable.add(iGoogleAPI.getDirections( mode: "driving",
467                             transit_routing: "less_driving",
468                             from,to,
469                             "AIzaSyA38X4HCspBdvWIMla87MVrhU-C46hVBvY")
470                             .subscribeOn(Schedulers.io())
471                             .observeOn(AndroidSchedulers.mainThread())
```

En la octava sintaxis de código mostrada anteriormente, se visualizar la codificación para dibujar la ruta desde la latitud y longitud de la ubicación real del repartidor hasta la latitud y longitud del destinatario con ayuda del api de Google Maps.





En la novena y décima sintaxis de código mostrada anteriormente, se visualizar la codificación para poder dibujar la ruta trazada desde latitud hasta latitud y longitud desde la tabla caminoPedido interactuando en tiempo real del repartidor por el fusedLocationProviderclient que es un método para visualizar el camino por el Google Maps API marcando una línea roja trazada por un objeto JSON.

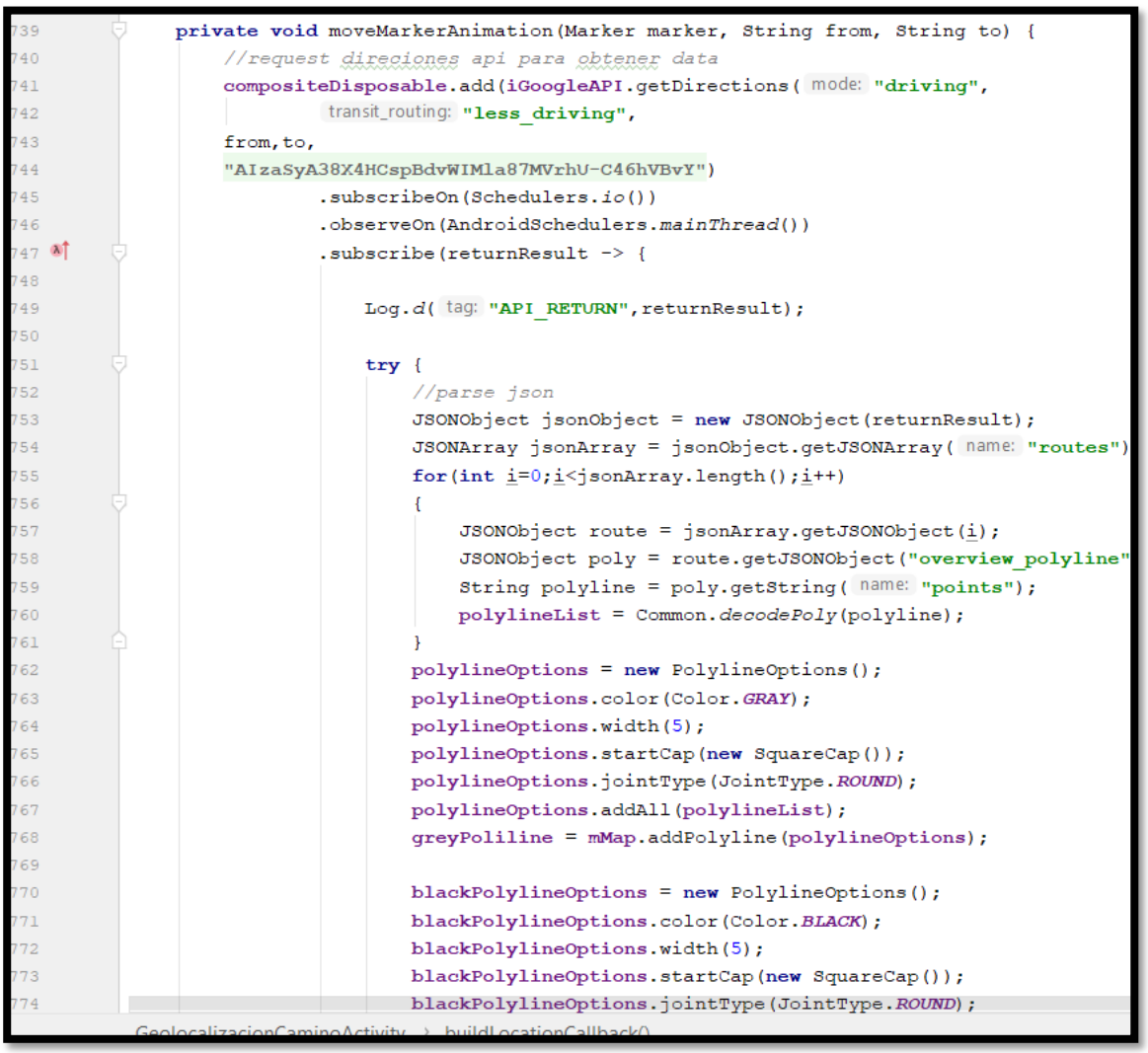
En la onceava sintaxis de código mostrada anteriormente, se visualiza la carga de datos del pedido adjunto a entregar al cliente solicitando mediante peticiones get, el nombre, número, dirección del cliente mediante la tabla Pedido y CaminoPedido.

```
private void buildLocationCallback() {
    locationCallback = new LocationCallback(){
        @Override
        public void onLocationResult(LocationResult locationResult) {
            super.onLocationResult(locationResult);
            // Add a marker in Sydney and move the camera
            LatLng locationRepartidor = new LatLng(locationResult.getLastLocation().getLatitude(),
                    locationResult.getLastLocation().getLongitude());

            updateLocation(locationResult.getLastLocation());
            if(repartidorMarker == null)
            {
                //inflate drawable interfaz
                int height,width;
                height = width = 80;
                BitmapDrawable bitmapDrawable = (BitmapDrawable) ContextCompat
                        .getDrawable( context: GeolocalizacionCaminoActivity.this,R.drawable.crissrepartidor);
                Bitmap resized = Bitmap.createScaledBitmap(bitmapDrawable.getBitmap(),width,height, filter: false);
                repartidorMarker = mMap.addMarker(new MarkerOptions()
                        .icon(BitmapDescriptorFactory.fromBitmap(resized))
                        .position(locationRepartidor).title("Tu"));

                mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(locationRepartidor, v: 18));

            }

            if(isInit && previousLocation != null)
            {
                String from = new StringBuilder()
                        .append(previousLocation.getLatitude())
                        .append(",")
                        .append(previousLocation.getLongitude())
                        .toString();
                String to = new StringBuilder()
```

En la doceava sintaxis de código mostrada anteriormente, se visualizar la codificación para la carga de la imagen de la moto del repartidor y la interacción con el api de Google Maps para que puedan reflejarse en tiempo real el pedido.

```
Common.java | LatLngInterpolator.java | MarkerAnimation.java | GeolocalizacionCaminoActivity.java
675         }
676
677         private void updateLocation(Location lastLocation) {
678
679
680             String data= Paper.book().read(Common.TRAYECTO_START);
681             if(!TextUtils.isEmpty(data))
682             {
683                 CaminoPedidoModel caminoPedidoModel = new Gson()
684                         .fromJson(data,new TypeToken<CaminoPedidoModel>(){}.getType());
685
686                 if(caminoPedidoModel !=null)
687                 {
688                     compositeDisposable.add(iGoogleAPI.getDirections( mode: "driving",
689                             transit_routing: "less_driving",
690                             Common.buildLocationString(lastLocation),
691                             new StringBuilder().append(caminoPedidoModel.getPedidoModel().getLat())
692                                     .append(",")
693                                     .append(caminoPedidoModel.getPedidoModel().getLng()).toString(),
694                             "AIzaSyA38X4HCspBdvWIMla87MVrhU-C46hVBvY")
695                             .subscribeOn(Schedulers.io())
696                             .observeOn(AndroidSchedulers.mainThread())
697                             .subscribe(s -> {
698
699                                 //estimacion de tiemppo
700                                 String estimacionTiempo = "UNKNOWN";
701                                 JSONObject jsonObject = new JSONObject(s);
702                                 JSONArray routes = jsonObject.getJSONArray( name: "routes");
703                                 JSONObject object = routes.getJSONObject( index: 0);
704                                 JSONArray legs = object.getJSONArray( name: "legs");
705                                 JSONObject legsObject = legs.getJSONObject( index: 0);
706                                 //configurar el tiempo
707                                 JSONObject time = legsObject.getJSONObject("duration");
708                                 estimacionTiempo = time.getString( name: "text");
709
710                                 Map<String,Object> update_data = new HashMap<>();
711                                 update_data.put("currentLat",lastLocation.getLatitude());
```

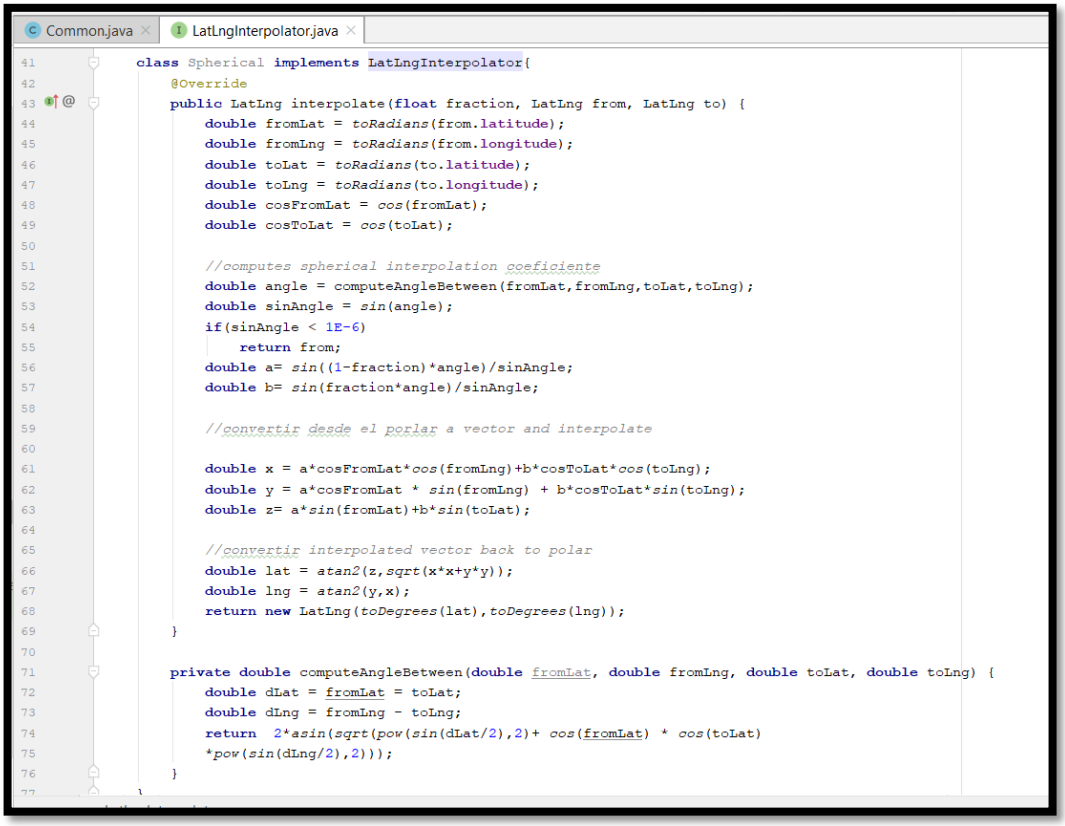
En la treceava sintaxis de código mostrada anteriormente, se visualiza el método updateLocation para poder conectarse el servicio Google Maps mediante un objeto JSON, dando así una estimación de tiempo según Google Maps de la ruta actual del repartidor hacia el destinatario por su latitud y longitud.



```
739     private void moveMarkerAnimation(Marker marker, String from, String to) {
740         //request direciones api para obtener data
741         compositeDisposable.add(iGoogleAPI.getDirections( mode: "driving",
742                 transit_routing: "less_driving",
743         from,to,
744         "AIzaSyA38X4HCspBdvWIMla87MVrhU-C46hVBvY")
745                 .subscribeOn(Schedulers.io())
746                 .observeOn(AndroidSchedulers.mainThread())
747                 .subscribe(returnResult -> {
748
749                     Log.d( tag: "API_RETURN",returnResult);
750
751                     try {
752                         //parse json
753                         JSONObject jsonObject = new JSONObject(returnResult);
754                         JSONArray jsonArray = jsonObject.getJSONArray( name: "routes")
755                         for(int i=0;i<jsonArray.length();i++)
756                         {
757                             JSONObject route = jsonArray.getJSONObject(i);
758                             JSONObject poly = route.getJSONObject("overview_polyline"
759                             String polyline = poly.getString( name: "points");
760                             polylineList = Common.decodePoly(polyline);
761                         }
762                         polylineOptions = new PolylineOptions();
763                         polylineOptions.color(Color.GRAY);
764                         polylineOptions.width(5);
765                         polylineOptions.startCap(new SquareCap());
766                         polylineOptions.jointType(JointType.ROUND);
767                         polylineOptions.addAll(polylineList);
768                         greyPoliline = mMap.addPolyline(polylineOptions);
769
770                         blackPolylineOptions = new PolylineOptions();
771                         blackPolylineOptions.color(Color.BLACK);
772                         blackPolylineOptions.width(5);
773                         blackPolylineOptions.startCap(new SquareCap());
774                         blackPolylineOptions.jointType(JointType.ROUND);
```

En la quinceava sintaxis de código mostrada anteriormente, se visualizar el método moveMarkerAnimation para usar la librería poly tanto de la línea roja para el trazado del pedido, como la línea negra para la ubicación real del repartidor y la línea amarilla para el trazado de la ruta opcional digitada por el repartidor.

En la dieciseisava sintaxis de código mostrada anteriormente, se visualiza la clase MarkerAnimation para poder obtener un mayor control de la ubicación real del pedido del repartidor hacia el destinatario del cliente mediante Interpolator para dar una animación escalable, la velocidad del motorizado influenciada con el aplicativo en tiempo real.

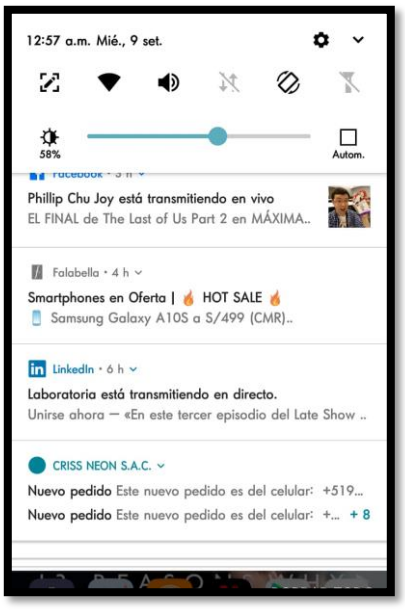


En la diecisieteava sintaxis de código mostrada anteriormente, se visualiza la clase LatLngInterpolator para trazar por radianes la latitud y longitud siguiente una lógica matemática que ayudará al camino en tiempo real del pedido mediante ángulos y fórmulas matemáticas.

```
public static float getBearing(LatLng begin, LatLng end) {
    double lat = Math.abs(begin.latitude-end.latitude);
    double lng = Math.abs(begin.longitude - end.longitude);

    if(begin.latitude < end.latitude && begin.longitude < end.longitude)
        return (float) (Math.toDegrees(Math.atan(lng/lat)));
    else if(begin.latitude >= end.latitude && begin.longitude < end.longitude)
        return (float) ((90 - Math.toDegrees(Math.atan(lng/lat)))+90);
    else if(begin.latitude >= end.latitude && begin.longitude >= end.longitude)
        return (float) (Math.toDegrees(Math.atan(lng/lat))+180);
    else if(begin.latitude < end.latitude && begin.longitude >= end.longitude)
        return (float) ((90 - Math.toDegrees(Math.atan(lng/lat)))+270);
    return -1;
}
```

En la dieciochoavo sintaxis de código mostrada anteriormente, se visualizar la clase getBearing() que va obtener la latitud y longitud del pedido mediante fórmulas ubicadas en ellas para traducir la latitud y longitud almacenada en el Firebase en el mapa geográfico de Google Maps.

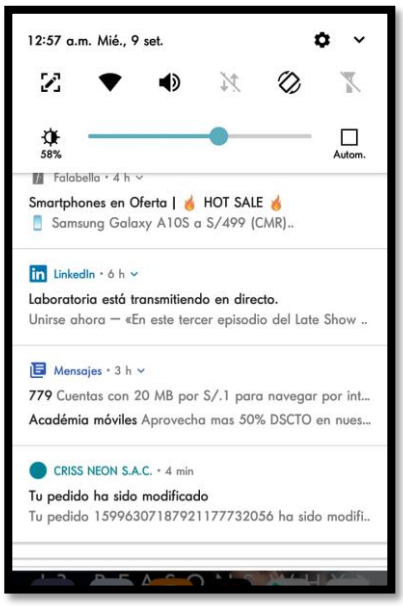
- **Storycard 46: Notificación de un nuevo pedido para el perfil del administrador por parte del perfil del cliente.**

Interfaz	Descripción
	Se puede visualizar la transmisión de un nuevo pedido de manera inmediata en el aplicativo del perfil del administrador que permite visualizar las notificaciones de pedidos recién solicitados.

```
FirebaseDatabase.getInstance()
        .getReference(Common.PEDIDO_REF)
        .child(Common.createPedidoNumber())
        .setValue(pedido)
        .addOnFailureListener(e -> {
            Toast.makeText(getContext(), text: ""+e.getMessage(), Toast.LENGTH_SHORT).show();
        }).addOnCompleteListener(task -> {
            cartDataSource.cleanCart(Common.currentUser.getUid())
                    .subscribeOn(Schedulers.io())
                    .observeOn(AndroidSchedulers.mainThread())
                    .subscribe(new SingleObserver<Integer>() {
                        @Override
                        public void onSubscribe(Disposable d) {

                        }

                        @Override
                        public void onSuccess(Integer integer) {
                            Map<String,String> notiData = new HashMap<>();
                            notiData.put(Common.NOTI_TITLE,"Nuevo pedido");
                            notiData.put(Common.NOTI_CONTENT, "Este nuevo pedido es del celular:  "+Common.currentUser.getPhone());

                            FCMSendData sendData = new FCMSendData(Common.createTopicPedido(),notiData);
                            compositeDisposable.add(ifcmService.sendNotification(sendData)
                            .subscribeOn(Schedulers.io())
                            .observeOn(AndroidSchedulers.mainThread())
                            .subscribe(fcmResponse -> {

                                Toast.makeText(getContext(), text: "Pedido solicitado de forma satisfactoria!", Toast.LENGTH_SHORT).show();
                                EventBus.getDefault().postSticky(new CounterCartEvent( success: true));

                            }, throwable -> {
                                Toast.makeText(getContext(), text: "Pedido fue enviado pero fallado la notificacion", Toast.LENGTH_SHORT).show();
                                EventBus.getDefault().postSticky(new CounterCartEvent( success: true));
```

En la sintaxis de código mostrada anteriormente, se muestra la codificación para el envío de notificación de un pedido de un cliente hacia el perfil del todos los administradores de CRISS NEON S.A.C mediante la clase FCMSendData para la transmisión de datos y Firebase Messaging.

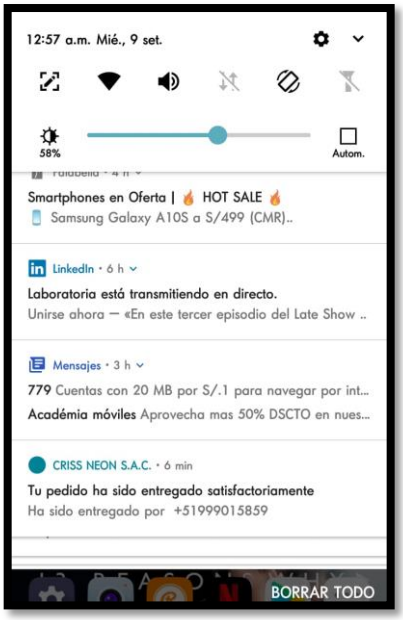
- **Storycard 47: Notificación de un nuevo pedido modificado por parte del administrador hacia el cliente**

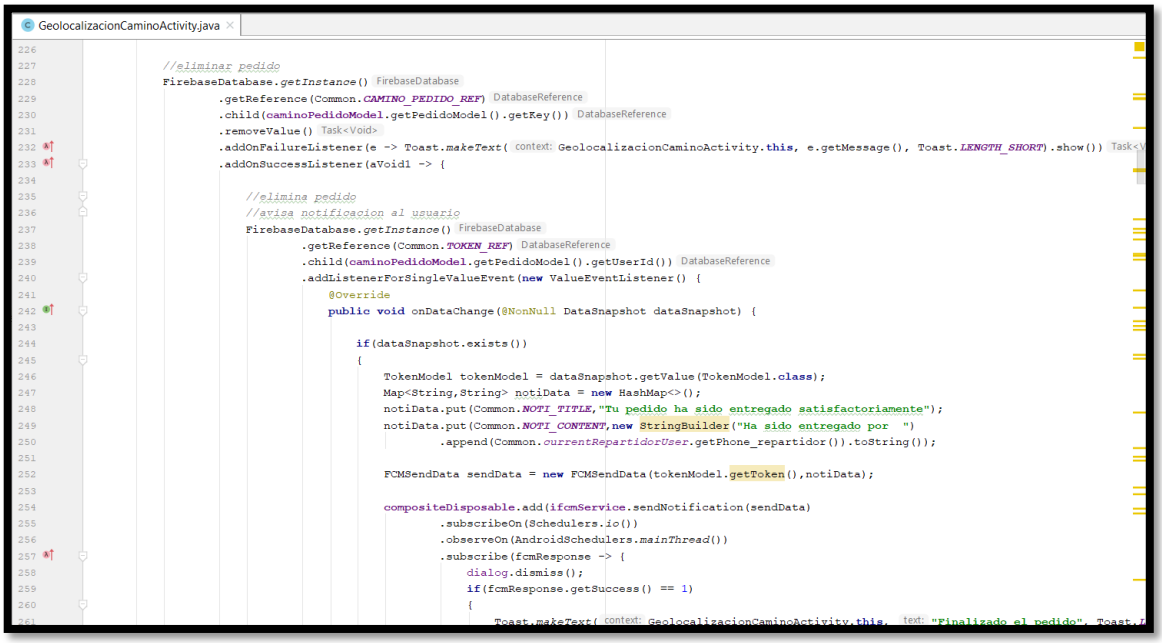
Interfaz	Descripción
	Se puede visualizar la transmisión de pedidos modificados ya sean cancelados, eliminados o asignados a un repartidor hacia el aplicativo del perfil del cliente o usuario.

```
PedidoFragment.java | PedidoViewModel.java

FirebaseDatabase.getInstance() FirebaseDatabase
        .getReference(Common.TOKEN_REF) DatabaseReference
        .child(pedidoModel.getUserId()) DatabaseReference
        .addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                if(dataSnapshot.exists())
                {
                    TokenModel tokenModel = dataSnapshot.getValue(TokenModel.class);
                    Map<String,String> notiData = new HashMap<>();
                    notiData.put(Common.NOTI_TITLE,"Tu pedido ha sido modificado");
                    notiData.put(Common.NOTI_CONTENT,new StringBuilder("Tu pedido ")
                            .append(pedidoModel.getKey())
                            .append(" ha sido modificado en ")
                            .append(Common.convertStatusToString(estado)).toString());

                    FCMSendData sendData = new FCMSendData(tokenModel.getToken(),notiData);

                    compositeDisposable.add(ifcmService.sendNotification(sendData)
                    .subscribeOn(Schedulers.io())
                    .observeOn(AndroidSchedulers.mainThread())
                    .subscribe(fcmResponse -> {
                        dialog.dismiss();
                        if(fcmResponse.getSuccess() == 1)
                            {
                                Toast.makeText(getContext(), text: "Pedido modificado satisfactoriamente", Toast.LENGTH_SHORT).show();
                            }else{
                            Toast.makeText(getContext(), text: "Pedido modificado satisfactoriamente pero no se envio la notificacion", Toast.LENGTH_SHORT)

                            }

                    }, throwable -> {
                        dialog.dismiss();
                        Toast.makeText(getContext(), text: ""+throwable.getMessage(), Toast.LENGTH_SHORT).show();

                    }));
```

En la sintaxis de código mostrada anteriormente, se muestra la codificación de transmisión de notificaciones mediante FirebaseMessaging y la tabla Token que ayuda a la transmisión de notificaciones mediante FCMSendData que una clase para él envió d noticias por usuario y su pedido cuando se cambia a cancelado, eliminado o asignado.

- **Storycard 48: Notificación de pedido entregado satisfactoriamente en el perfil del cliente/usuario.**

Interfaz	Descripción
	Se puede visualizar la transmisión de noticias de pedidos ya entregados satisfactoriamente, cuando el repartidor confirma la entrega completa.



```
GeolocalizacionCaminoActivity.java

        //eliminar pedido
        FirebaseDatabase.getInstance() FirebaseDatabase
                .getReference(Common.CAMINO_PEDIDO_REF) DatabaseReference
                .child(caminoPedidoModel.getPedidoModel().getKey()) DatabaseReference
                .removeValue() Task<Void>
                .addOnFailureListener(e -> Toast.makeText( context: GeolocalizacionCaminoActivity.this, e.getMessage(), Toast.LENGTH_SHORT).show()) Task<V
                .addOnSuccessListener(aVoid1 -> {

                    //elimina pedido
                    //avisa notificacion al usuario
                    FirebaseDatabase.getInstance() FirebaseDatabase
                            .getReference(Common.TOKEN_REF) DatabaseReference
                            .child(caminoPedidoModel.getPedidoModel().getUserId()) DatabaseReference
                            .addListenerForSingleValueEvent(new ValueEventListener() {
                                @Override
                                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                                    if(dataSnapshot.exists())
                                    {
                                        TokenModel tokenModel = dataSnapshot.getValue(TokenModel.class);
                                        Map<String,String> notiData = new HashMap<>();
                                        notiData.put(Common.NOTI_TITLE,"Tu pedido ha sido entregado satisfactoriamente");
                                        notiData.put(Common.NOTI_CONTENT,new StringBuilder("Ha sido entregado por  ")
                                                .append(Common.currentRepartidorUser.getPhone_repartidor()).toString());

                                        FCMSendData sendData = new FCMSendData(tokenModel.getToken(),notiData);

                                        compositeDisposable.add(ifcmService.sendNotification(sendData)
                                                .subscribeOn(Schedulers.io())
                                                .observeOn(AndroidSchedulers.mainThread())
                                                .subscribe(fcmResponse -> {
                                                    dialog.dismiss();
                                                    if(fcmResponse.getSuccess() == 1)
                                                    {
                                                        Toast.makeText( context: GeolocalizacionCaminoActivity.this, text: "Finalizado el pedido", Toast.
```

En la sintaxis de código mostrada anteriormente, se muestra la codificación para enviar notificaciones al Cliente o Usuario cuando el repartidor confirma que el pedido ha sido entregado de forma satisfactoria por la clase FCMSendData y el FirebaseMessaging informando al cliente por pedido entregado.

FASE IV: ESTABILIZACIÓN

En esta fase se logra integrar la funcionalidad la aplicación de la cual se comprueba que la aplicación funcione con normalidad.

Recomendaciones del dispositivo móvil

Dispositivo Móvil	
Hardware	• 4GB Ram • Procesor Quad-core 1.4GHz. • Conexión Wifi 802.11.
Software	• Lollipop Api 22 como mínimo. • Android

FASE V: PRUEBAS

- ✓ **Perfil: Cliente**

- **Prueba unitaria 01: Módulo de Login**

Código de PU:	PU01
Código del módulo:	MC01
Nombre del módulo:	Módulo de Login
Descripción	Verificar el funcionamiento de un registro de un nuevo usuario a través de un número de celular del usuario.
Objetivo	Registrarse y Autentificarse en la aplicación móvil con geolocalización.
Condiciones	1) Presionar el botón “Acceder con el teléfono” 2) Digitar el número de celular del dispositivo móvil. 3) Revisar el código de confirmación de 6 dígitos enviado. 4) Digitar el código de 6 dígitos enviado. 5) El sistema arrojará una ventana emergente de registro. 6) Escribir el nombre. 7) Buscar y seleccionar la dirección con funcionamiento del api Google Places. 8) Presionar el botón “Registrar”. 9) El sistema carga la interfaz de la selección de mejores productos.
Resultado esperado	Al momento de registrarse en el aplicativo debe emitir un mensaje de usuario registrado y la pantalla de selección de mejores productos.
Resultado obtenido	La prueba fue aprobatoria, los datos se almacenaron correctamente en la base de datos NoSql Firebase y entra de frente a la pantalla de selección de productos.

- **Prueba unitaria 02: Módulo de Edición de Datos Personales**

Código de PU:	PU02
Código del módulo:	**MC02**
Nombre del módulo:	Módulo de Edición de Datos Personales
Descripción	Verificar el funcionamiento de la edición de datos personales en la aplicación móvil con geolocalización del perfil usuario o cliente.
Objetivo	Editar datos del usuario.
Condiciones	1) Presionar las tres rayas de la parte superior. 2) Presionar en la parte donde dice “Editar Datos” 3) El sistema arrojará una ventana emergente para modificar datos. 4) El usuario puede modificar su nombre o buscar la dirección geolocalizada por Google Places. 5) Presionar en el botón Modificar. 6) El sistema almacena los cambios.
Resultado esperado	Al momento de modificar datos en el aplicativo, el sistema guarda los cambios
Resultado obtenido	La prueba fue aprobatoria, los datos modificaron correctamente en la base de datos NoSql Firebase.

- **Prueba unitaria 03: Módulo de Carrito de Compras**

Código de PU:	PU03
Código del módulo:	**MC03**
Nombre del módulo:	Módulo de Carrito de Compras
Descripción	Escoger productos según complemento y tamaño añadiendo en el carrito de compras, comentar y puntualizar con máximo 5 estrellas el producto.
Objetivo	Escoger productos para elaborar el carrito de compras.
Condiciones	1) Presionar las tres rayas de la parte superior. 2) Presionar en la parte donde dice “Categorias” 3) Seleccionar una categoría. 4) El sistema arroja todos los productos de la categoría seleccionada. 5) Seleccionar el producto a desear. 6) Desplazar el celular hacia abajo. 7) Seleccionar el tamaño del producto. 8) Seleccionar el botón “+” al lado de añadir y dar click a los complementos que se desea agregar. 9) Dar click en cualquier lado de la pantalla para cerrar la ventana emergente de complementos. 10) Desplazarse hacia arriba de la pantalla para visualizar el botón de estrella. 11) Dar click al botón de estrella. 12) Seleccionar el número de estrella y comentar sobre producto. 13) Dar al botón OK de la ventana emergente de la puntuación del producto y el sistema cerrará la ventana emergente. 14) Presionar el botón del carrito para agregar el producto al carrito de compra y el sistema muestra de producto agregado. 15) Presionar el botón de color morado del carrito.

	16) Desplazar el producto de la parte derecha y aparecerá una opción para eliminar.
Resultado esperado	Los Productos se añadan de forman satisfactoria al carrito de compras según tamaño, complemento, comentario y puntuación del mismo y si se desea se puede eliminar el mismo.
Resultado obtenido	La prueba fue aprobatoria, se añadió los productos al carrito de compras según tamaño, complemento, comentario y puntuación a la base de datos temporal de SQLite almacenándose en la memoria interna de la aplicación.

- **Prueba unitaria 04: Módulo de Solicitud del Pedido**

Código de PU:	PU04
Código del módulo:	**MC04**
Nombre del módulo:	Módulo de Solicitud del Pedido
Descripción	Registrar el pedido solicitando el carrito de compras previamente escogido.
Objetivo	Solicitar el pedido con los productos escogidos en el carrito de compras.
Condiciones	1) Presionar el botón de color morado del carrito. 2) Presionar el botón “PEDIR PEDIDO”. 3) Buscar y seleccionar la dirección con funcionamiento del api Google Places. 4) Presionar el botón “YES” para confirmar la solicitud del pedido. 5) El sistema mandará una notificación a todos los usuarios registrados como administradores de un nuevo pedido y el celular del mismo.
Resultado esperado	El pedido ha sido enviado de forma satisfactoria para que el administrador pueda asignar a un repartidor.

Resultado obtenido	La prueba fue aprobatoria, se envió la solicitud del pedido para que el administrador pueda asignar un repartidor a dicho pedido.

- **Prueba unitaria 05: Módulo de Control de Pedido**

Código de PU:	PU05
Código del módulo:	**MC05**
Nombre del módulo:	Módulo de Control de Pedido
Descripción	Monitorear el pedido solicitado si se quiere repetir el mismo, el trayecto en tiempo real o cancelarlo.
Objetivo	Monitorear el pedido solicitado.
Condiciones	1) Presionar las tres rayas de la parte superior. 2) Presionar en la parte donde dice “Ver Pedidos” 3) Desplazarse hacia abajo y buscar el pedido solicitado. 4) El pedido solicitado, desplazar de la parte derecha. 5) Al dar click en “Repetir Pedido” repetiremos los productos solicitados con su tamaño y complemento hacia el carrito de compras. 6) Al dar click en “Ver Trayecto”, se puede visualizar el recorrido en tiempo real del mismo, ya que el repartidor esta hiendo en camino con el pedido. 7) Al dar click en “Cancelar Pedido”, se canceló un pedido previo que no fue asignado por el administrador a un repartidor para cancelarlo.
Resultado esperado	El pedido ha sido monitoreado de forma satisfactoria ya que se pudo repetir, ver el trayecto y cancelar el pedido.
Resultado obtenido	La prueba fue aprobatoria, se repitió el pedido al carrito de compras, se vio el trayecto y canceló el pedido mismo.

- **Prueba unitaria 06: Módulo de Envío de Noticias**

Código de PU:	PU06
Código del módulo:	**MC06**
Nombre del módulo:	Módulo de Envío de Noticias
Descripción	Recibir notificaciones o noticias de parte de los administradores sobre la empresa.
Objetivo	Recibir notificaciones o noticias sobre la empresa.
Condiciones	1) Presionar las tres rayas en la parte superior. 2) Presionar en la parte donde dice “Suscribirse a Noticias” 3) El sistema arrojará una ventana emergente. 4) Presionar en el check de Suscribirse a Nuevas Noticias. 5) Presionar el botón “Permitir”. 6) Revisar la bandeja de notificaciones para el recibo de notificaciones.
Resultado esperado	El usuario recibe las noticias o notificaciones por parte del administrador.
Resultado obtenido	La prueba fue aprobatoria, se emitió las noticias o notificaciones por parte de los administradores a través de Firebase Cloud Messagging.

- ✓ **Perfil: Administrador**

- **Prueba unitaria 07: Módulo de Login**

Código de PU:	PU07
Código del módulo:	**MA01**
Nombre del módulo:	Módulo de Login
Descripción	Verificar el funcionamiento del perfil de sesión del administrador.
Objetivo	Autentificarse en la aplicación móvil en modo administrador.
Condiciones	1) Presionar el botón “Acceder con el teléfono” 2) Digitar el número registrado. 3) Revisar el código de confirmación de 6 dígitos enviado. 4) Digitar el código de 6 dígitos enviado. 5) El sistema arrojará la pantalla principal del administrador que es la edición de categorías.
Resultado esperado	Al momento de loguearse en el aplicativo, debe arrojar la pantalla principal del administrador.
Resultado obtenido	La prueba fue aprobatoria, se logueo de forma correcta con el usuario registrado de administrador en Firebase.

- **Prueba unitaria 08: Módulo de Categorías**

Código de PU:	PU08
Código del módulo:	**MA02**
Nombre del módulo:	Módulo de Categorías
Descripción	Verificar el funcionamiento de un registro, modificación y eliminación de una categoría.
Objetivo	Realizar el registro, modificación y eliminación de una categoría.
Condiciones	1) Presionar el botón del lápiz en la parte superior. 2) Escribir el nombre de la categoría.

	3) Seleccionar la foto de la categoría. 4) Presionar el botón de “Agregar”. 5) El sistema agrega la nueva categoría. 6) Desplazar en la parte derecha de una categoría. 7) Dar click en “Modificar” 8) Cambiar el nombre de la categoría. 9) Seleccionar una nueva foto de la categoría. 10) Dar click en el botón “Modificar” 11) Se modifica la categoría de forma automática. 12) Desplazar en la parte derecha de una categoría. 13) Dar click en “Eliminar” 14) El sistema arrojará una ventana emergente. 15) Dar click a “Eliminar” 16) El sistema eliminará la categoría seleccionada.
Resultado esperado	Al momento de registrar, eliminar o modificar una categoría se debe hacer de forma satisfactoria interactuando con la base de datos Firebase.
Resultado obtenido	La prueba fue aprobatoria, se agregó, modifico y elimino de manera correcta la categoría seleccionado interactuando con la base de datos.

- **Prueba unitaria 09: Módulo de Productos**

Código de PU:	PU09
Código del módulo:	**MC03**
Nombre del módulo:	Módulo de Productos
Descripción	Verificar el funcionamiento de un registro, modificación, eliminación y búsqueda de un producto incluido el registro, modificación y eliminación de su tamaño y complemento.
Objetivo	Realizar el registro, modificación, eliminación y búsqueda de un producto incluido el registro, modificación y eliminación de su tamaño y complemento.

Condiciones	1) Seleccionar una categoría. 2) El sistema arroja la lista de productos por categoría. 3) Seleccionar un producto y desplazar en la parte derecha 4) Dar click en “Edit”. 5) El sistema arrojará una ventana emergente para la edición de esta misma. 6) Cambiar el nombre, precio, producto o descripción del producto. 7) Seleccionar una nueva foto del producto. 8) Dar click en el botón “Modificar” 9) Se modifica el producto de forma automática. 10) Desplazar en la parte derecha de un producto. 11) Dar click en “Eliminar” 12) El sistema arrojará una ventana emergente. 13) Dar click a “Eliminar” 14) El sistema eliminará el producto correspondiente. 15) Dar click en el icono de lupa y digitar el producto buscado y dar click en el botón de lupa. 16) Desplazar en la parte derecha de un producto. 17) Dar click en “+”. 18) El sistema arrojará una ventana emergente para añadir complementos. 19) Digitar el nombre y precio del complemento. 20) Presionar el botón “Crear”. 21) El sistema agregará el nuevo complemento. 22) Seleccionar un complemento. 23) Editar el nombre y precio del complemento. 24) Dar click en “Editar”. 25) El sistema edita de forma automática el complemento. 26) Seleccionar un complemento y a su lado presionar el icono del tachito para eliminarlo.

	27) Seleccionar el icono de un disquete para guardar los cambios en la parte superior. 28) Seleccionar el icono de regreso para regresar a la lista de productos. 29) Dar click en el icono de lupa y digitar el producto buscado y dar click en el botón de lupa. 30) Desplazar en la parte derecha de un producto. 31) Dar click en “Tam”. 32) El sistema arrojará una ventana emergente para añadir tamaños. 33) Digitar el nombre y precio del tramaño. 34) Presionar el botón “Crear”. 35) El sistema agregará el nuevo tamaño. 36) Seleccionar un tamaño. 37) Editar el nombre y precio del tamaño. 38) Dar click en “Editar”. 39) El sistema edita de forma automática el tamaño. 40) Seleccionar un tamaño y a su lado presionar el icono del tachito para eliminarlo. 41) Seleccionar el icono de un disquete para guardar los cambios en la parte superior. 42) Seleccionar el icono de regreso para regresar a la lista de productos.
Resultado esperado	Al momento de registrar, eliminar o modificar producto, tamaño y complemento del mismo se debe hacer de forma satisfactoria interactuando con la base de datos Firebase.
Resultado obtenido	La prueba fue aprobatoria, se agregó, modifico y elimino de manera correcta el producto seleccionado incluyendo su tamaño y complemento del mismo interactuando con la base de datos.

- **Prueba unitaria 10: Módulo de Control de Pedidos**

Código de PU:	PU10
Código del módulo:	**MA04**
Nombre del módulo:	Módulo de Control de Pedidos
Descripción	Verificar el orden correcto de los pedidos por Estado, el funcionamiento y validaciones dependiendo del estado que se encuentre el pedido.
Objetivo	Realizar la verificación de un control de pedidos adecuado ordenándolos por estado y comprobando su funcionalidad mediante las validaciones asignadas por estado.
Condiciones	1) Presionar las tres rayas en la parte superior. 2) Presionar en la parte donde dice “Pedidos” 3) El sistema conlleva a una interfaz de los pedidos por atender. 4) Deslizar un pedido hacia la derecha. 5) Presionar el botón “Edit”. 6) El sistema nos lleva a otra interfaz para asignar a un repartidor dicho pedido. 7) Presionar en el repartidor asignado. 8) Click en el botón “Guardar”. 9) El sistema indica un mensaje “Pedido modificado satisfactoriamente” 10) Seleccionar un pedido y deslizarlo hacia la derecha. 11) Presionar “Eliminar”. 12) El sistema arroja una ventana emergente de la confirmación de eliminación de un pedido. 13) El usuario le da click a la opción “ELIMINAR” y el sistema elimina el pedido de la lista. 14) Seleccionar un pedido y deslizarlo hacia la derecha. 15) Presionar “Llamar”

	16) El sistema lanza una ventana emergente de confirmación de permiso para llamadas. 17) Presionar “Permitir” y el sistema lleva a la interfaz de la marcación de celular con el número de celular del cliente indicado del pedido. 18) Regresar a la aplicación móvil. 19) Seleccionar un pedido y deslizarlo hacia la derecha. 20) Presionar “Dirección”. 21) El sistema informa con una notificación que el pedido es atendido y no tiene aún la ubicación. 22) Seleccionar un pedido y deslizarlo hacia la derecha. 23) Presionar “Imprimir”. 24) El sistema imprime un reporte en pdf sobre los productos solicitados adjuntando el detalle, subtotal, igv y total del pedido. 25) Regresar a la aplicación móvil. 26) Seleccionar de la parte superior la opción de las tres rayas. 27) Dar click a “Pedidos en camino”. 28) El sistema arroja los pedidos en camino que fueron asignados a un repartidor. 29) Deslizar un pedido hacia la derecha. 30) Presionar el botón “Edit”. 31) El sistema nos lleva para modificar el pedido de llevarlo al estado de entregado o cancelado. 32) Presionar la opción “Entregado”. 33) Click en el botón “Guardar”. 34) El sistema indica un mensaje “Pedido modificado satisfactoriamente” modificando el estado y quintándolo de la lista. 35) Seleccionar un pedido y deslizarlo hacia la derecha. 36) Presionar “Eliminar”.

	37) El sistema arroja una ventana emergente de la confirmación de eliminación de un pedido. 38) El usuario le da click a la opción “ELIMINAR” y el sistema elimina el pedido de la lista. 39) Seleccionar un pedido y deslizarlo hacia la derecha. 40) Presionar “Llamar” 41) El sistema lanza una ventana emergente de confirmación de permiso para llamadas. 42) El sistema lleva a la interfaz de la marcación de celular con el número de celular del cliente indicado del pedido. 43) Regresar a la aplicación móvil. 44) Seleccionar un pedido y deslizarlo hacia la derecha. 45) Presionar “Dirección”. 46) El sistema conlleva a otra ventana del aplicativo visualizando la ruta real del pedido en Google maps del geolocalizador sobre el pedido. 47) Seleccionar un pedido y deslizarlo hacia la derecha. 48) Presionar “Imprimir”. 49) El sistema imprime un reporte en pdf sobre los productos solicitados adjuntando el detalle, subtotal, igv y total del pedido. 50) Regresar a la aplicación móvil. 51) Seleccionar de la parte superior la opción de las tres rayas. 52) Dar click a “Pedidos entregados”. 53) El sistema arroja los pedidos entregados. 54) Deslizar un pedido hacia la derecha. 55) Presionar el botón “Edit”. 56) El sistema nos lleva para modificar el pedido de llevarlo al estado de cancelado 57) Presionar la opción “Cancelado”. 58) Click en el botón “Guardar”.

	59) El sistema indica un mensaje “Pedido modificado satisfactoriamente” modificando el estado y quintándolo de la lista. 60) Seleccionar un pedido y deslizarlo hacia la derecha. 61) Presionar “Eliminar”. 62) El sistema arroja una ventana emergente de la confirmación de eliminación de un pedido. 63) El usuario le da click a la opción “ELIMINAR” y el sistema elimina el pedido de la lista. 64) Seleccionar un pedido y deslizarlo hacia la derecha. 65) Presionar “Llamar” 66) El sistema lleva a la interfaz de la marcación de celular con el número de celular del cliente indicado del pedido. 67) Regresar a la aplicación móvil. 68) Seleccionar un pedido y deslizarlo hacia la derecha. 69) Presionar “Dirección”. 70) El sistema informa que el pedido ha sido entregado y no tiene coordenadas mediante un mensaje. 71) Seleccionar un pedido y deslizarlo hacia la derecha. 72) Presionar “Imprimir”. 73) El sistema imprime un reporte en pdf sobre los productos solicitados adjuntando el detalle, subtotal, igv y total del pedido. 74) Regresar a la aplicación móvil. 75) Seleccionar de la parte superior la opción de las tres rayas. 76) Dar click a “Entregas Perfectamente Recibidas”. 77) El sistema arroja los Entregas Perfectamente Recibidas. 78) Deslizar un pedido hacia la derecha. 79) Presionar el botón “Edit”.

	80) El sistema nos lleva para restaurar o eliminar el pedido. 81) Presionar la opción “Restaurar Pedido”. 82) Click en el botón “Guardar”. 83) El sistema indica un mensaje “Pedido modificado satisfactoriamente” modificando el estado, quitando de la lista y volviéndole al estado “Atendido”. 84) Seleccionar un pedido y deslizarlo hacia la derecha. 85) Presionar “Eliminar”. 86) El sistema arroja una ventana emergente de la confirmación de eliminación de un pedido. 87) El usuario le da click a la opción “ELIMINAR” y el sistema elimina el pedido de la lista. 88) Seleccionar un pedido y deslizarlo hacia la derecha. 89) Presionar “Llamar” 90) El sistema lleva a la interfaz de la marcación de celular con el número de celular del cliente indicado del pedido. 91) Regresar a la aplicación móvil. 92) Seleccionar un pedido y deslizarlo hacia la derecha. 93) Presionar “Dirección”. 94) El sistema informa que el pedido ha sido cancelado y no tiene coordenadas mediante un mensaje. 95) Seleccionar un pedido y deslizarlo hacia la derecha. 96) Presionar “Imprimir”. 97) El sistema imprime un reporte en pdf sobre los productos solicitados adjuntando el detalle, subtotal, igv y total del pedido. 98) Regresar a la aplicación móvil.
Resultado esperado	Al momento de hacer un control de pedido, se puedan visualizar los pedidos realizados por estado y se

	puedan tener las validaciones correspondientes por el estado del mismo.
Resultado obtenido	La prueba fue aprobatoria, se pudo hacer un control de pedidos por estado del mismo a su vez ver la dirección real del mismo siempre y cuando el repartidor haya iniciado el pedido, llamando al cliente e imprimiendo un reporte de detalle de pedido.

- **Prueba unitaria 11: Módulo de Repartidores**

Código de PU:	PU11
Código del módulo:	**MA05**
Nombre del módulo:	Módulo de Repartidores
Descripción	Verificar el funcionamiento de la activación de incio de sesión de los repartidores.
Objetivo	Realizar la activación del repartidor en el aplicativo.
Condiciones	1) Presionar las tres rayas en la parte superior. 2) Presionar en la parte donde dice “Repartidores” 3) Dar click en el icono de buscar del icono de lupa para buscar al repartidor por su nombre. 4) Dar click en el icono de lupa para buscar el repartidor. 5) Activar mediante el selector del repartidor para activar el inicio de sesión del repartidor.
Resultado esperado	Al momento de activar o desactivar el inicio a la aplicación del repartidor, puede navegar con su usuario.
Resultado obtenido	La prueba fue aprobatoria, se activó al repartidor interactuando con la base de datos para que el repartidor pueda navegar en la aplicación móvil.

- **Prueba unitaria 12: Módulo de Envío de Noticias**

Código de PU:	PU12
Código del módulo:	**MA06**
Nombre del módulo:	Módulo de Envío de Noticias
Descripción	Enviar notificaciones o noticias a los usuarios de la empresa por parte de los administradores.
Objetivo	Emitir notificaciones o noticias a los usuarios.
Condiciones	1) Presionar las tres rayas en la parte superior. 2) Presionar en la parte donde dice “Nuevas Noticias” 3) El sistema arrojará una ventana emergente para publicar noticias a los usuarios. 4) Digitar el título y contenido de la notificación. 5) Presionar en enviar para seleccionar la noticiar. 6) Repetir los pasos 1, 2 y 4. 7) Seleccionar la opción Imagen. 8) Dar click en el símbolo de Android y subir la imagen correspondiente. 9) Presionar en Enviar.
Resultado esperado	La emisión correcta de noticias o notificaciones de parte del administrador.
Resultado obtenido	La prueba fue aprobatoria, se envió las notificaciones a los clientes y usuarios de parte del administrador de la empresa.

- **Prueba unitaria 13: Módulo de Reportes**

Código de PU:	PU13
Código del módulo:	**MA07**
Nombre del módulo:	Módulo de Reportes
Descripción	Visualizar reportes en un rango de fechas de los indicadores de entregados completos, índice de servicio y entregas perfectamente recibidas.
Objetivo	Emitir reportes según un rango de fechas de los indicadores del control de pedidos.
Condiciones	1) Presionar las tres rayas en la parte superior. 2) Presionar en la parte donde dice “Reportes” 3) El sistema arrojará una ventana emergente para visualizar los reportes correspondientes al control de pedidos. 4) Dar click en “Entregados Completos”. 5) Seleccionar en las dos fechas, un rango de fechas de incio y fin para. 6) Dar click en el botón “Imprimir Reporte” para visualizar el pdf. 7) Dar click en el botón “Regresar” para regresar a la pantalla principal del administrador. 8) Repetir los pasos 1, 2, 3 y 4. 9) Dar click en “Índice de Servicio”. 10) Repetir los pasos 5, 6 y 7. 11) Repetir los pasos 1,2,3 y 4. 12) Dar click en “Entregas Perfectamente Recibidas”. 13) Repetir los pasos 5, 6 y 7.
Resultado esperado	La emisión correcta de los reportes en pdf para realizar un seguimiento por día del control de pedidos.
Resultado obtenido	La prueba fue aprobatoria, se emitió los reportes correspondientes con su rango de fechas consultando con la base de datos de Firebase.

- ✓ **Perfil: Repartidor**

- **Prueba unitaria 14: Módulo de Login**

Código de PU:	PU14
Código del módulo:	**MR01**
Nombre del módulo:	Módulo de Login
Descripción	Verificar el funcionamiento de un registro de un nuevo usuario a través de un número de celular del repartidor.
Objetivo	Registrarse y Autentificarse del perfil del repartidor en la aplicación móvil con geolocalización.
Condiciones	1) Presionar el botón “Acceder con el teléfono” 2) Digitar el número de celular del dispositivo móvil. 3) Revisar el código de confirmación de 6 dígitos enviado. 4) Digita el código de 6 dígitos enviado. 5) El sistema arrojará una ventana emergente de registro. 6) Escribir el nombre 7) Presionar el botón “Registrar”. 8) El repartidor tiene que avisar al administrador para que le puedan activar su inicio de sesión.
Resultado esperado	Al momento de registrarse en el aplicativo debe emitir un mensaje que tiene que pedir permiso al administrador para el inicio de sesión.
Resultado obtenido	La prueba fue aprobatoria, los datos se almacenaron correctamente en la base de datos NoSql Firebase y el repartidor tiene que pedir permiso al administrador para el inicio de sesión.

- **Prueba unitaria 15: Módulo de Pedidos Asignados**

Código de PU:	PU15
Código del módulo:	**MA02**
Nombre del módulo:	Módulo de Pedidos Asignados
Descripción	Visualizar los pedidos asignados por el administrador y ver que un pedido que no ha sido entregado solamente ve eso, pero no otro.
Objetivo	Verificar pedidos asignados por repartidor y validar pedido en camino.
Condiciones	1) Presionar las tres rayas en la parte superior. 2) Presionar en la parte donde dice “Pedidos Asignados” 3) Se visualiza los pedidos asignados por repartidor. 4) Click a “Enviar Ahora”. 5) El sistema arrojará los detalles del pedido seleccionado. 6) Se seleccionó otro pedido que no ha sido comenzado y el sistema arrojó el pedido que esta hiendose en camino.
Resultado esperado	Se visualiza los pedidos asignados y la validación del pedido en camino para evitar confusiones.
Resultado obtenido	La prueba fue aprobatoria, se visualizan los pedidos asignados por repartidor y se validó el pedido en camino por más que el repartidor de click en otro pedido para ver el detalle solo se verá el detalle del pedido que está en trayecto.

- **Prueba unitaria 16: Módulo de Trayecto del pedido con Geolocalización.**

Código de PU:	PU16
Código del módulo:	**MR03**
Nombre del módulo:	Módulo de Trayecto del pedido con Geolocalización.
Descripción	Visualizar el trayecto real del pedido avanzado que se vea trazado por una línea roja la ruta que debe seguir el repartidor y la línea negra trazada del repartidor es la ruta que actualmente sigue el mismo, llamar al cliente, buscar una dirección alterna por el servicio de Google Maps, iniciar y terminar un pedido.
Objetivo	Apoyar al repartidor en la entrega completa del pedido y atendiendo la solicitud del cliente hasta la llegada final de su pedido.
Condiciones	1) Seleccionar un pedido asignado y darle click a “Enviar” 2) El sistema arroja una ventana para visualizar el servicio del api de Google Maps y visualizar la localización exacta del repartidor hasta el destinatario del pedido marcando una línea roja. 3) Click a “VER COMPLETO”. 4) El sistema arroja el detalle del pedido. 5) Click a “Enviar”. 6) El sistema activa el pedido como en camino para que sea validado como único pedido para visualizar y puedan ver el trayecto real tanto el cliente y el administrador. 7) Mientras el repartidor recorre el camino, el sistema traza una línea negra que es el trayecto del repartidor. 8) Click al icono de llamar.

	9) El sistema arroja a otra ventana llamando directamente al usuario 10) Terminar la llamada y de frente el sistema vuelve al trayecto del pedido. 11) Digitar en donde dice “Search” para buscar una dirección alterna. 12) El sistema arroja una línea amarilla trazada desde el punto actual del repartidor hacia la dirección localizada. 13) Cuando el repartidor haya entregado de forma correcta el pedido, dar click a “Hecho”. 14) El sistema arroja una ventana emergente para confirmar pedido entregado completo. 15) Dar click en “SI” 16) El pedido se desvanece y avisa con una notificación al cliente que solicitó el pedido que este mismo ha sido entregado con el número del repartidor.
Resultado esperado	El sistema hace un seguimiento real de la ruta del repartidor hacia el destino del pedido siguiendo coordenadas de latitud y longitud del mismo por los servicios de Google Maps Apis y conlleva a una ruta alterna por el mismo marcando una línea amarilla que pueda empezar, finalizar el pedido y llamar al cliente; cuando se haya finalizado el pedido se enviará una notificación al cliente del pedido entregado de manera satisfactoria por el número.
Resultado obtenido	La prueba fue aprobatoria, se pudo corroborar la funcionalidad correcta del api de Google Maps para el trayecto real, correcto y alterno del pedido del usuario en tiempo real, empezar y terminar un pedido enviando una notificación de entrega completada por último se pudo llamó al cliente.