UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Aplicación web progresiva con arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:

Ingeniero de Sistemas

AUTORES:

Garcia Ortiz, Angel Eduardo (orcid.org/0000-0001-9608-4824)

Garcia Tavara, Edwin Fidel (orcid.org/0000-0002-3294-0414)

ASESOR:

Mg. Agurto Marchan, Winner (orcid.org/0000-0002-0396-9349)

LÍNEA DE INVESTIGACIÓN:

Sistemas de Información y Comunicaciones

LÍNEA DE RESPONSABILIDAD SOCIAL UNIVERSITARIA:

Promoción de la salud, nutrición y salud alimentaria

PIURA – PERÚ

2023

DEDICATORIA

El presente estudio está dedicado a nuestro padre celestial, Dios, quien nos ha guiado en todo momento, y a nuestros progenitores, quienes han sido nuestros pilares y guías de formación, junto con su apoyo incondicional.

AGRADECIMIENTO

Gracias al apoyo de los docentes, con sus conocimientos rigurosos y precisos, debemos mis conocimientos a ustedes. A nuestros padres, que son el motor que impulsan nuestros sueños y esperanzas, y que además son nuestros guías en la vida. También agradecemos a nuestros amigos, quienes siempre estuvieron ahí para apoyarnos en los momentos difíciles.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS Y FIGURAS

RESUMEN

La presente investigación tuvo como objetivo determinar la mejora del proceso de gestión de pacientes infantes al implementar una PWA usando la arquitectura de monolitos modulares en un centro de salud de la ciudad de Tumbes. Fue aplicada y tuvo un enfoque cuantitativo con un diseño pre-experimental; utilizando un muestreo probabilístico aleatorio simple para seleccionar una muestra de 46 apoderados de pacientes infantes, y además 7 enfermeras del centro de salud. La recolección de datos se realizó mediante entrevistas utilizando una guía estructurada, así como encuestas mediante cuestionarios en un pre test y post test. Los resultados revelaron mejoras en el proceso de gestión de pacientes infantes, pasando de un 52.8% de valoración baja a un 90.6% de valoración alta. Además, respecto a la arquitectura, se encontraron ventajas como la flexibilidad para realizar cambios en los módulos y un despliegue más ágil. Se concluyó que la implementación del sistema mejoró el proceso estudiado, teniendo en cuenta la perspectiva de las enfermeras y de los apoderados de pacientes; de esta manera se mejoró los tiempos en la creación de reportes y búsqueda de historias clínicas, el manejo de información, el seguimiento de los pacientes, y la calidad de atención.

Palabras clave: Aplicación web progresiva (PWA), gestión de pacientes infantes, arquitectura de monolitos modulares.

ABSTRACT

The objective of this research was to determine the improvement of the management process of infant patients by implementing a PWA using the architecture of modular monoliths in a health center in the city of Tumbes. It was applied and had a quantitative approach with a pre-experimental design; using a simple random probability sampling to select a sample of 46 proxies of infant patients, and also 7 nurses of the health center. Data collection was carried out by means of interviews using a structured guide, as well as questionnaire surveys in a pre-test and post-test. The results revealed improvements in the infant patient management process, going from a 52.8% low rating to a 90.6% high rating. In addition, with respect to the architecture, advantages were found such as flexibility to make changes in the modules and a more agile deployment. It was concluded that the implementation of the system improved the process studied, taking into account the perspective of the nurses and the patients' proxies; in this way, it improved times in the creation of reports and search of clinical histories, information management, patient follow-up, and quality of care.

Keywords: Progressive web application (PWA), infant patient management, modular monolith architecture.

I. INTRODUCCIÓN

Con los últimos acontecimientos vividos a nivel mundial, sobre todo con la llegada de la emergencia sanitaria del Covid-19, demostraron ser los catalizadores hacia una transformación clínica, operativa y financiera, que la atención médica ha venido prometiendo durante un largo periodo de tiempo. Cabe mencionar que ya se tenía prevista dicha transformación al llegar al año 2040, con cambios como los que se han vivido hoy en día, pero gracias a los acontecimientos antes mencionados este lapso de tiempo se aceleró y la puso en marcha, trayendo consigo numerosas tendencias médicas que ya existían y/o que empezaban a emerger. Aun así, la pobreza y la falta de financiamiento en sistemas efectivos para satisfacer los servicios básicos de salud; son barreras importantes para la evolución de la salud en gran parte del mundo, a pesar de los esfuerzos para cerrar las brechas (Allen, 2022).

El sector salud observado de manera nacional se encuentra en un proceso de transformación mediante la implementación e identificación de soluciones digitalizadas, siendo la salud digital un papel fundamental en su modernización. Es por ello que se busca poner en marcha acciones que favorezcan la calidad dirigida a los procesos y los datos en los SI, los cuales proporcionan funcionalidades para registros relacionados a la atención médica, con el fin de fortalecer la digitalización del sector (Ministerio de Salud, 2020). Aun así, existen diferentes falencias por las que atraviesan los centros de salud en el Perú, y esto es demostrado en la investigación de Espinoza et al. (2019) que menciona una de las principales razones por las que los apoderados no llevan a sus hijos a las atenciones CRED, siendo este el tiempo de espera, quien impide que se cumplan e influyendo de manera negativa en el proceso.

El presente trabajo fue realizado en un centro de salud del departamento de Tumbes, el cual contaba con diferentes inconvenientes al momento de gestionar a los pacientes infantes, de los cuales se tenían: un ineficiente manejo de las historias clínicas; a la hora de realizar reportes de pacientes que necesitaban recibir alguna dosis establecida para su edad o pendientes de recibir atenciones CRED, el proceso se tornaba tardío y tedioso; cuando se quería buscar una historia clínica, el proceso conllevaba una gran

cantidad de tiempo; la pérdida de información; entre otros, y esto debido a que todo se hacía manual mediante fichas y archivos que a veces se traspapelaban y traían consigo inconvenientes; obteniéndose un proceso de gestión y reporte lento, así como una atención poco eficiente hacia los pacientes infantes.

En la salud, los sistemas de información representan un papel importante en el suministro de información de salud oportuna, integral y de alta calidad para tomar decisiones estratégicas y operativas que puedan salvar vidas, es por ello que las instituciones que se encargan de ofrecer servicios de salud deben contar con estos para poder brindar un eficaz y eficiente servicio a los pacientes. Lo cual, según el Ministerio de Salud (2020), esto es debido a que proporcionan la unión de la información generada por los encargados de salud, los sistemas de información y los pacientes a través del desarrollo de las TI, contribuyendo a que las personas tengan una mejor calidad de atención, privacidad y que su información esté segura.

Es por ello, que con el presente trabajo se pretendió elaborar una solución tecnológica para lograr satisfacer las falencias por las que pasaba el centro de salud, mediante la implementación de un sistema que utiliza la arquitectura de monolitos modulares que proporciona escalabilidad al sistema informático. Y todo esto para que las enfermeras puedan llegar a gestionar el proceso involucrado y las actividades que este incluye, teniendo interacciones directas y dinámicas con las historias clínicas y los respectivos reportes que se quieran generar a partir de estas. Por ello se recalca en la investigación, el proceso de gestión de pacientes infantes, así como también los indicadores necesarios para medir la mejora al implementar el sistema.

Mencionado lo anterior, se planteó como problema general: ¿En cuánto mejora una PWA basada en la arquitectura de monolitos modulares la gestión de pacientes infantes del centro de salud? Y los siguientes problemas específicos; como primero se tuvo: ¿Cómo influye una PWA basada en la arquitectura de monolitos modulares en la perspectiva del personal de salud en el proceso de gestión de pacientes infantes del centro de salud? El segundo fue: ¿Cómo influye una PWA basada en la arquitectura de monolitos modulares en la perspectiva de los apoderados de pacientes del proceso

de gestión de pacientes infantes del centro de salud? Mientras que el tercero fue: ¿Cuáles son las ventajas de utilizar la arquitectura de monolitos modulares en el desarrollo de la PWA?

La justificación social del trabajo realizado, es el incremento de la eficiencia que está afectada por la implementación de la herramienta tecnológica acordada para el centro de salud, debido a que acorta los tiempos de búsqueda, así como la creación de historias clínicas. Además, la justificación tecnológica del trabajo se sostiene en que da paso a la innovación para la búsqueda y creación de diferentes soluciones como la automatización de diferentes procesos ejercidos por las enfermeras para los pacientes infantes. Asimismo, aporta de manera investigativa al analizar las ventajas de la arquitectura de monolitos modulares en el desarrollo de aplicaciones de este tipo.

Según lo antes expuesto se estableció el objetivo general de investigación: Determinar la mejora del proceso de gestión de pacientes infantes al implementar una PWA con la arquitectura de monolitos modulares en el centro de salud. A continuación, los objetivos específicos; primero: Determinar la mejora de la perspectiva que tiene el personal de salud respecto al proceso de gestión de pacientes infantes con la implementación de la PWA que utiliza la arquitectura de monolitos modulares: como segundo: Determinar la mejora de la perspectiva que tienen los apoderados de pacientes respecto al proceso de gestión de pacientes infantes con la implementación de la PWA que utiliza la arquitectura de monolitos modulares; y como último: Analizar las ventajas que tiene utilizar la arquitectura de monolitos modulares en el desarrollo de la PWA. Los cuales permitieron definir la siguiente hipótesis principal: El proceso de gestión de pacientes infantes mejora significativamente al implementar la PWA que usa la arquitectura de monolitos modulares en el centro de salud. Además, se tuvieron las siguientes hipótesis específicas planteadas; la primera fue: La perspectiva que tiene el personal de salud respecto al proceso de gestión de pacientes infantes mejora significativamente con la implementación de la PWA que utiliza la arquitectura de monolitos modulares; y la segunda fue: La perspectiva que tienen los apoderados respecto al proceso de gestión de pacientes infantes mejora significativamente con la implementación de la PWA que utiliza la arquitectura de monolitos modulares.

II. MARCO TEÓRICO

En cuanto a la fundamentación teórica de la investigación que se desarrolló se presentaron estudios previos realizados sobre el mismo tema de investigación con el fin de darle más relevancia a lo que fue escogido para investigar. Para empezar, se obtuvieron como antecedentes internacionales los estudios de:

Según Rivera et al. (2020), en su estudio realizado con el objetivo de diseñar una aplicación web progresiva que utilice conexión en tiempo real con los datos del centro médico y que el paciente pueda acceder a esta mediante su dispositivo móvil, tuvo una investigación cuantitativa pre-experimental, y utilizó la técnica estadística ksmooth para la interpolación de datos, debido a que las variables intervinientes fueron del tipo aleatorias, obteniendo como principales resultados que los tiempos de registro de citas médicas fueron más eficientes con el sistema implementado, trayendo consigo optimización y eficiencia en los diferentes horarios de atención. Como principal conclusión se tuvo que con el uso de la tecnología de PWA, se logró proporcionar un sistema con mayor portabilidad, el cual permite que se abarque una mayor cantidad de pacientes; además, se logró una reducción significativa del tiempo de espera para el registro de citas médicas, permitiendo reducir las filas y el personal necesario para la atención de pacientes; logrando brindar una solución tecnológica que ayuda en la administración y gestión de citas médicas, mejorando el acceso y gestión de la información.

Según Tajudeen et al. (2021), en su investigación con el propósito de diseñar e implementar un sistema informático que admita gestionar pacientes para el servicio universitario de salud en Federal University of Technology Akure, la cual se basó en una investigación de tipo exploratoria y describió las fases primordiales de una metodología usada en el desarrollo de software de acuerdo al sistema a implementar. Con el sistema implementado, se obtuvo como principal resultado que se logró evitar la pérdida de los expedientes de los pacientes, así como también una mayor seguridad de los mismos; además, se redujo las colas durante el registro; mejoró y facilitó los servicios de atención médica; minimizó el uso de papel al momento de crear registros

y redujo el dinero invertido en comprar documentos y útiles para escribir. En cuanto a la conclusión más resaltante se menciona que un sistema informático bien diseñado e integrado puede representar una herramienta potente en manos de un establecimiento de salud, y esto debido a que proporciona una mejora en los diferentes servicios que se brindan, controla los costos y puede garantizar el óptimo aprovechamiento de las instalaciones de la institución.

Según Rêgo et al. (2021), en su investigación con el objetivo de lograr la confianza de los lectores para aceptar a las aplicaciones web progresivas como una factible y confiable solución en lo que respecta a la industria de la salud, la cual fue de tipo básica; tuvo como principales resultados que dentro de las diferentes organizaciones mencionadas en la investigación, Twitter y OLA Cabs son claros ejemplos de que las aplicaciones web progresivas son capaces de permitir una experiencia participativa y que la conexión de red no sea una limitación, por lo que dichas mejoras tuvieron un impacto positivo tanto para los usuarios que usan el servicio como para el negocio. Además, se presentó un software que es una prueba contundente de que este tipo de aplicaciones puede proporcionar las características necesarias para brindar una mejor calidad de atención de salud, reduciendo de esta manera errores médicos y agilizando las intervenciones clínicas; ofreciendo oportunidades de mejora en la atención médica.

Según Oñate et al. (2020), en su tesis de implementación de una PWA para gestionar pruebas para simular el ingreso a universidades e instituciones militares en un centro de nivelación y capacitación académica, que tuvo como objetivo implementar la misma en la institución; fue una investigación bibliográfica, documental y aplicada, utilizando la técnica documental; en la que se concluyó que gracias al desarrollo de esta aplicación se logró facilitar el proceso de toma, seguimiento y realización de pruebas de simulación, permitiendo ahorrar recursos y tiempo en la institución. Además, se consiguió dejar de usar las hojas y cuadernillos diariamente, lo cual generaba malestar por el esfuerzo y tiempo dedicados.

A nivel nacional se tienen a las investigaciones de:

Huamani (2018), quien en su investigación para analizar y diseñar un sistema dirigido

a la gestión hospitalaria para el mejoramiento de la atención hacia pacientes en un hospital en el departamento de Ica, tuvo como uno de sus objetivo más relevantes la implementación de este para lograr mejorar el proceso de atención de pacientes de la institución; esta fue cuantitativa, de tipo aplicada, y experimental, donde se investigó a 28 pacientes, de los que se sacó una muestra de 20, y como principales instrumentos se tuvo a una plantilla de cuestionario, una guía de entrevista y de observación; en esta investigación se concluyó que se pudo obtener toda la información suficiente y necesaria para lograr la implementación del sistema integral que permitió solucionar los problemas de atenciones hacia pacientes y la cantidad de atenciones por día en el hospital, además que, mediante el sistema se podrá generar indicadores y reportes automáticos para las distintas áreas del hospital.

Según Torres (2020), en su estudio con la finalidad de determinar el efecto que tenía el uso del sistema al momento de controlar niños en un establecimiento de salud; tuvo una investigación aplicada de diseño pre-experimental, que trabajó con una población en base a la cantidad de documentos que se registraron por los pacientes de entre 0 y 5 años, tomando un periodo de 20 días, aplicando la técnica del fichaje y la entrevista; concluyó que al implementar el sistema se consiguió un impacto favorable en el proceso de control de niños, incrementando de esta manera los porcentajes de metas objetivo y la reducción del porcentaje de retorno de archivos. Además, se demostró que con el sistema se agilizó el trabajo a comparación del método manual, evitándose el trabajo rutinario, generando reportes puntuales, y que el personal encargado de salud realice tareas mucho más complicadas.

Según Muñoz (2020), en su investigación que contó con el principal propósito de diseñar e implementar una aplicación que permita la optimización al gestionar citas en una clínica, usó una investigación exploratoria, cuantitativa y pre-experimental, con una población de 30 empleados aplicándoles un cuestionario; obteniéndose que el 6.67% de los encuestados indicó que antes de la implementación del sistema existía un bajo nivel en cuanto a la administración de citas médicas, mientras que después, se alcanzó un 76.67% de nivel alto. Concluyéndose que se logró optimizar de manera considerable la reserva, gestión y el tiempo que conllevaba la atención de citas

médicas, así como también la comunicación con los clientes hospitalarios de dicha clínica.

Según Morales (2019), en su investigación con el objetivo de desarrollar un sistema de fácil uso para gestionar las historias clínicas que brinde la posibilidad de guardar la información de aquellos pacientes que acuden al centro de salud; fue de tipo aplicada y exploratoria; de la cual se concluyó que al usar el sistema se pudo hacer el registro y búsqueda de historias de una forma más rápida. A su vez tuvo impacto en la atención de pacientes, porque permitió consultar su información de manera completa, segura y rápida. Además, se pudo evitar el uso de papel al registrar las historias, lo cual impacta de forma positiva en cuanto a pérdida de información y confusión de documentos, ahorro de espacio físico, y mitigación del acumulo de polvo causado por la excesiva cantidad de papeles. Asimismo, los reportes arrojados por el sistema brindaron la posibilidad de tomar mejores decisiones.

El autor Peralta (2019), en su investigación con el propósito principal de implementar un sistema informático que permita registrar y controlar las historias clínicas para lograr la reducción de los tiempos que tenía la atención hacia los pacientes en un hospital universitario en Piura, contó con una investigación cuantitativa, de tipo descriptiva, además fue no experimental de tipo transversal, la cual tuvo como población 600 pacientes y una muestra de 173, a los que se les aplicaron cuestionarios, y se hizo uso de una hoja de observación. Los principales resultados obtenidos fueron que el tiempo promedio que conllevaba la actividad de registrar pacientes pre test era de 223 segundos, mientras que lo ocupado en el post fue de 130 segundos; además, en cuanto al promedio de búsqueda de pacientes en pre test era de 180 segundos, y el de post test llego hasta 10 segundos. De esta investigación se obtuvo como principal conclusión que con la implementación el proceso de control y registro de las citas de los pacientes se volvió mucho más rápido, permitiendo encontrar y registrar la información de manera más eficiente.

Y por último en cuanto a antecedente local se tuvo al autor Goyburo (2018), quien en su investigación realizada tuvo el propósito de diseñar e implementar un sistema para

mejorar la gestión de pacientes en el área de admisión de un consultorio pediátrico, fue una investigación que utilizó un enfoque descriptivo, transversal y no experimental; en donde como parte de la muestra, se entregó una encuesta a diez personas, entre personal y pacientes habituales. Según los resultados, la implementación del sistema supuso una mejora considerable en la gestión de los pacientes. En particular, se automatizaron las operaciones de introducción de datos, reduciendo el tiempo necesario para registrar a los pacientes y sus historias, trayendo mejoras en la práctica, y facilidad en la gestión de los datos al permitir un acceso fácil y sistemático.

A continuación, se exponen las bases teóricas involucradas en el estudio, partiendo por explicar qué son **centros de salud**, en donde según el Ministerio de Salud (2015) son aquellos establecimientos de la salud que realizan su atención de manera ambulatoria o de internamiento, con el fin de prevenir, promover, diagnosticar, tratar y rehabilitar para mantener o restablecer la salud de las personas. La **gestión de pacientes infantes** es de suma importancia en los centros de salud, ya que según Chavez (2020), es una estrategia que brinda la posibilidad de ordenar y lograr sistematizar los procesos que intervienen en la atención sanitaria de una manera eficiente e ideal con la participación de profesionales en dicha gestión para tomar decisiones en base a los pacientes. Asimismo, según un autor Morales (2019), esta puede abarcar dos perspectivas: una que haga referencia a la gestión como sistema completo de atención que involucre tanto al paciente como al proceso puesto en práctica; y el segundo referente a la práctica médica realizada con el sistema del personal de atención. Una de las actividades más resaltantes de este proceso es el **control de pacientes infantes**, la cual según un autor Ramirez (2020), es el control que se realiza del crecimiento y desarrollo de aquellos niños que se encuentran entre cero y cinco años, la cual es una edad crucial, y esto debido a que aquí es más rápido poder hallar o prevenir distintas enfermedades, por lo tanto cuando los pacientes acudan a los centros de salud por atenciones CRED, estos se deben desarrollar de una manera eficaz y completa. Por otra parte, menciona que la **atención de pacientes infantes** es un tema delicado e importante por la relación que guarda con la salud y los derechos humanos, el cual va de la mano con la responsabilidad de los apoderados

de los pacientes. Otro concepto importante a tomar en cuenta en este proceso, es el control de vacunas, que según Giglio et al. (2018), representa una de las estrategias preventivas que tiene más efectividad en la salud pública. El control de vacunas ha reducido mucho la incidencia de muchas enfermedades infecciosas, incluso las ha eliminado.

En cuanto a las **aplicaciones informáticas** según Valarezo et al. (2018) son las herramientas que proporcionan a los usuarios acceso a un servidor web mediante la red utilizando un navegador específico. Una aplicación informática generalmente se define como un programa de computadora que se ejecuta a través de un navegador. Asimismo, según Quezada et al. (2018) entre los distintos **tipos de aplicaciones** se encuentran las aplicaciones web que son desarrolladas en lenguajes como HTML y JavaScript, las cuales son publicadas en servidores web. También están las aplicaciones móviles diseñadas para los dispositivos móviles; estas se desarrollan en distintas plataformas como Android y IOS. Y por último se detallan las híbridas o webs móviles, las cuales solo necesitan un navegador para ser ejecutadas desarrolladas en lenguajes de las mismas aplicaciones web, combinando lo mejor de las antes mencionadas. Conceptualizando de una manera más clara a las **aplicaciones web** se puede tomar como referencia a Arias (2018), quien menciona que estas aplicaciones son en esencia un medio para facilitar la ejecución de una labor especial en la web, lo que es diferente a un portal web estático que es un instrumento, pero igual fundamental para la comunicación. Asimismo, existen las **aplicaciones web progresivas**, quienes son definidas por Aguirre et al. (2019) como una aplicación web que usa las últimas tecnologías accesibles en los navegadores permitiendo que pueda agregar características que eran exclusivas de las aplicaciones nativas. A pesar de estar enfocado en los dispositivos móviles, proporciona la capacidad de ser instalado en computadoras de escritorio, aprobando constituir el desarrollo de aplicaciones sin tener en cuenta el tipo de dispositivo. Por otro lado según Morales (2019), cuando se habla de **beneficios obtenidos de un sistema de gestión de pacientes**, las principales acciones de registro y búsqueda de historias clínicas se realizarán de manera más fluidas, lo cual impactará de manera beneficiosa al momento de atender pacientes

porque permitirá obtener la información de éstos de una manera segura, aligerada, y sencilla. Además, se podrá evitar el uso de papel para realizar el registro de las historias, teniendo un impacto a favor en cuanto al ahorro de espacio de manera física, pérdida de datos e información almacenada y disminución del riesgo de una posible confusión de documentos.

Según Jaiswal (2019) la **arquitectura de software** es definida como un diseño estratégico de una actividad que se relaciona con requisitos que se caracterizan por ser globales. Dicha solución es implementada mediante estilos y patrones arquitectónicos, paradigmas de programación, ingeniería de software basada en estándares, seguridad, regularidades regidas por leyes, entre otros. Además menciona dentro de los diferentes **tipos de arquitecturas de software** a la de servidor la cual se divide en dos grupos, primero el backend como servicio conocido como 'Baas' y los servicios computarizados que se basan en la nube conocidos como 'FaaS'; también menciona a la impulsada por eventos, que se basa en productores y clientes de eventos, con la finalidad de desacoplar los componentes del sistema; y por último la de microservicios, esta se enfoca en desarrollar autónomos y pequeños servicios, de tal manera que cada servicio se encargue de resolver un inconveniente en específico o de realizar una tarea en particular; estos servicios se comunican mediante APIs. Para conceptualizar un poco más a microservicios, según la institución Red Hat (2022), mencionó que un es una arquitectura que estructura una aplicación como un conjunto de servicios poco acoplados, implementados de forma independiente y fáciles de mantener, organizados en torno a las capacidades comerciales de un equipo pequeño. En cuanto a los **beneficios de la arquitectura de software** según Gruhn et al. (2018), la ventaja más reconocida de centrarse en la arquitectura de software es mejorar la comunicación. Esta comunicación puede ser entre desarrolladores o entre los desarrolladores y las distintas partes interesadas, esto permitirá tener una base sólida del proyecto, aumentará el rendimiento y la calidad, proporcionará escalabilidad, evitará repetir código y promoverá las buenas prácticas, permitirá la implementación de modificaciones de una manera más óptima, y reducirá los costos, los tiempos de construcción y entrega de proyectos.

Según el autor Marques (2020)**, monolito** significa "trabajo construido sobre roca", usado para definir la arquitectura que se usó en algún sistema donde el código y las funciones se encuentran en un solo proceso. Lo cual contrasta el autor Newman (2019), quien menciona que un monolito hace referencia a que toda la funcionalidad del sistema está implementada o desplegada en un mismo lugar. Por otro lado, este autor menciona que también existen los **monolitos modulares**, que consisten en módulos separados, de los cuales cada uno de ellos puede ser trabajado de forma independiente, pero aún deben ser combinados para su implementación; en sí se basa en descomponer el software como tal. Asimismo, menciona algunas de las ventajas de la arquitectura de monolitos modulares, donde resalta que, para muchas de las organizaciones, esta arquitectura puede ser una de las mejores opciones, debido a que si los módulos se encuentran bien delimitados y definidos puede proporcionar un alto nivel de trabajo en paralelo, pero evitando los desafíos que da la arquitectura de microservicios con su distribución y las preocupaciones de implementación, volviendo más simple la construcción del software. Esta arquitectura está relacionada con las aplicaciones modulares, las cuales según Joshi (2016), se caracterizan por admitir complementos, y esto se debe a que admiten la carga dinámica de nuevos complementos, siempre que todos sigan la misma estructura de interfaz esperada y ejecutada por la aplicación.

A continuación, se mencionan diferentes conceptos relacionados con la investigación.

Según Casas et al. (2018), a los **frameworks** se les conoce como herramientas que permiten construir aplicaciones web de manera rápida mediante diferentes mecanismos. Estos cuentan con el soporte y la documentación necesaria para brindar una manera más sencilla en cuanto a su utilización.

Como framework utilizado para el frontend, estubo **Angular,** definido por Angular.io (2022), como una plataforma para desarrollo, la cual incluye un marco de trabajo que está basado en componentes, de esta manera crear aplicaciones web que escalen con el tiempo, lo que engloba desde pequeños proyectos hasta aplicaciones de nivel empresarial. Este framework trabaja con **RxJS**, el cual según Alabor et al. (2020), es

una biblioteca popular para implementar aplicaciones de transmisión usando JavaScript a través de principios de programación reactiva. Asimismo, se usó **Angular Material,** que para Mensah (2019), es una biblioteca conformada por componentes para las interfaces de usuario (UI) las cuales pueden ser usadas por los desarrolladores en sus proyectos de Angular, permitiendo acelerar el desarrollo. Los componentes de Angular Material son hermosos y reutilizables, de los cuales se tienen a mapas, indicaciones, tablas de datos, selectores de fechas y más. Otra tecnología usada junto a esta librería fue **TailwindCSS**, definida por Klimm (2021), como un framework que adopta un enfoque alternativo a los estilos CSS tradicionales, utilizando clases para crear un diseño personalizado sin la molestia de un estilo obstinado.

Respecto al backend, se trabajó con **Node.js,** definido por Haro et al. (2019), como una plataforma de ejecución que admite la ejecución de JavaScript a través del servidor. Cuando se combina con el framework Express.js, permite crear enrutamiento que trabaja sobre HTTP. Además, para trabajar con esta plataforma, se usó **NestJS**, que para Sabo (2020), es un framework utilizado para crear aplicaciones del lado del servidor utilizando NodeJS, el cual trabaja usualmente con TypeScript. Este permite crear aplicaciones modulares, sostenibles y escalables con el tiempo. Por otra parte, en cuanto a seguridad de autenticación se usó **JWT**, del cual Chandel (2018), menciona que permite lograr la transmisión de información de manera segura como un objeto JSON entre las partes que desean realizar comunicación. Dicha información puede ser verificada, por lo que la vuelve confiable al estar firmada digitalmente.

Asimismo para la elaboración del sistema se utilizaron principios y patrones, como los **principios SOLID**, quienes según Bipin (2016), menciona que los **principios SOLID** están basados en un conjunto de reglas y buenas prácticas a seguir, y se aplican al momento de diseñar una estructura de una clase. Además, ayudan a comprender la necesidad de determinadas arquitecturas y patrones de diseño. Por otro lado, está el patrón **Redux**, el cual según Garreau et al. (2018), es un framework general que logra un equilibrio entre la flexibilidad y estructura suficientes. Como tal, proporciona una plataforma para que los desarrolladores creen una gestión de estado personalizada para sus casos de uso, al tiempo que permite reutilizar cosas como un depurador

gráfico o middleware. Para utilizar este patrón se hizo uso de **NgRx**, el que según Fu (2018), se trata de un framework que permite la gestión de estados y se basa en el patrón Redux. De esta manera ayudar a administrar el estado de aplicaciones de gran tamaño.

Para Nordeen (2020), las **bases de datos** son una sistematización de datos alojados en una colección, las cuales permiten la manipulación y el almacenamiento de estos, por lo que facilitan de gran manera poder gestionarlos. Asimismo, el **Sistema de gestión de bases de datos** es denominado una colección de software que proporciona el ingreso a la BD, manipular datos e informar/representar estos. Una base de datos muy conocida es **PostgreSQL,** que según Anabel et al. (2020), utiliza el lenguaje de consulta estructurado, que actualmente es muy popular y utilizado en el mercado. Está muy bien catalogado por su estabilidad, rendimiento, robustez y facilidad de gestión e implementación.

Según Tarwani et al. (2016) entre las diferentes **metodologías ágiles** se encuentra Extreme Programming (XP), dónde inicialmente crean las diferentes historias de usuario y aseguran que todos los requisitos se agreguen en presencia del usuario. También está la metodología Crystal que viene siendo la más liviana, caracterizada por el tamaño y prioridades del equipo. Entre las metodologías se encuentra Scrum, centrada no solo en el desarrollo sino en el crecimiento iterativo e incremental del producto, sin descuidar los aspectos gerenciales, dividido en pequeños trabajos denominados sprints. Durante los sprint, se priorizan los requisitos y también son llamados historias de usuarios. Además, Schwaber et al. (2020) resaltan que entre los **roles de SCRUM** se encuentra el Scrum Master, el cual es un guía para los miembros del equipo, ayudando a comprender la práctica y la teoría de Scrum. Otro rol viene siendo el Product Owner, quien se encarga de maximizar el valor del producto final. Y, por último, los desarrolladores, encargados de asegurar los incrementables de cada sprint; así como también crear un plan para estos, adaptarse al plan para cumplir el objetivo y asegurar la eficacia de cada sprint terminado.

III. METODOLOGÍA

3.1. Tipo y diseño de investigación

La presente investigación fue de tipo aplicada, debido a que mediante conocimientos ya existentes se pretendió buscar la solución a un problema en específico, en este caso la implementación de una PWA con arquitectura monolítica modular para la gestión de niños en el centro de salud; este tipo de investigación según Rodriguéz (2020) cuenta con la característica que el problema está determinado y ya es conocido por el investigador, es por ello que hace uso de la investigación para responder a preguntas explícitas.

En cuanto al diseño de investigación fue experimental de tipo pre experimental porque se buscó responder a la hipótesis planteada, además de evaluar y demostrar que con la implementación de una PWA que utiliza la arquitectura de monolitos modulares para gestionar pacientes infantes en el centro de salud, es beneficiosa, haciendo un análisis antes y después de la implementación del sistema; el autor Rus (2020) dijo que la investigación experimental trata de conocer los cambios en torno a una variable que es dependiente, al momento en el que se modifica una o varias de las que depende.

3.2. Variables y operacionalización

Esta investigación tuvo como variable independiente, la Aplicación web progresiva utilizando la arquitectura de monolitos modulares, en donde PWA es definida por Aguirre et al. (2019) como aquella que usa las últimas tecnologías accesibles en los navegadores permitiendo que pueda agregar características que eran exclusivas de las aplicaciones nativas; y en cuanto a los monolitos modulares, según Newman (2019) consisten en módulos separados, de los cuales cada uno de ellos puede ser trabajado de forma independiente, pero aún deben ser combinados para su implementación; en sí se basa en descomponer el software como tal. Está variable contuvo 2 dimensiones: Estructura de base de datos y arquitectura del sistema; de las cuales solo la segunda poseyó indicadores, que fueron: Modularidad, escalabilidad y respuesta.

Por otra parte, está la variable dependiente, Proceso de gestión de pacientes infantes,

es definida por Chavez et al. (2020) como una estrategia que brinda la posibilidad de ordenar y lograr sistematizar los procesos que intervienen en la atención sanitaria de una manera eficiente e ideal con la participación de profesionales en dicha gestión para tomar decisiones en base a los pacientes. En cuanto a las dimensiones que poseyó esta variable fueron: Perspectiva de la enfermera y perspectiva del paciente, en donde la primera tuvo 3 indicadores: Nivel del manejo de información, tiempo de reportes de pacientes y respaldo de datos; mientras que la segunda dimensión tuvo como indicadores a calidad de atención, tiempo de búsqueda de historias clínica, tiempo de atención, nivel de confianza, y seguimiento del paciente.

3.3. Población, muestra y muestreo

En este trabajo de investigación que se realizó en un centro de salud del departamento de Tumbes, para los indicadores modularidad, escalabilidad y respuesta se delimitó la cantidad de 3 especialistas en desarrollo de software; además para los indicadores nivel del manejo de la información, nivel del tiempo de reportes de pacientes y frecuencia de pérdida de datos, se delimitó a una población de 7 enfermeras pertenecientes al centro de salud; mientras que para el indicador nivel de satisfacción, 123 apoderados de pacientes infantes nacidos en el primer trimestre del año 2023 que acuden a la institución. Según Gamboa (2018) la población es un conjunto de elementos sobre los cuales el investigador se interesa para poder obtener conclusiones o realizar inferencias para tomar decisiones. Además, recalca que usualmente suelen ser personas.

A la muestra es le denomina un subconjunto de la población, el cual es representativo de esta y se puede tener acceso; esto debe ser así porque con este subconjunto se realizan las mediciones pertinentes (Gamboa, 2018).

La muestra de apoderados de pacientes infantes nacidos en el primer trimestre del año 2023 que acuden al centro de salud, se determinó usando la siguiente fórmula

estadística:

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

Figura 1. Fórmula estadística para la obtención de la muestra

Dónde:

n = Muestra

N = Población

Z = Nivel de confianza al 95% (1.96) elegido para la investigación

p = Probabilidad de éxito (95% = 0.95)

q = Probabilidad de fracaso (5% = 0.05)

d = Precisión (5%)

$$n = \frac{(1.96)^2 * (123) * (0.95) * (0.05)}{(123-1) * (0.05)^2 + (1.96)^2 * (0.95) * (0.05)} = 46.04$$

Figura 2. Cálculo de muestra con los datos obtenidos de la investigación

Aplicando la fórmula estadística para calcular la muestra, se determinó que se trabajaría con una muestra de 46 apoderados.

En cuanto al muestreo de esta investigación, este fue probabilístico de tipo aleatorio simple, y esto debido a que los datos obtenidos de la población para el muestreo pertenecen a las mismas características, poseyendo así las mismas posibilidades de ser seleccionadas en los dos tiempos estudiados, pre y post test.

3.4. Técnicas e instrumentos de recolección de datos

Para los indicadores modularidad, escalabilidad y respuesta pertenecientes a la dimensión arquitectura del sistema, se hizo uso de la técnica de la entrevista y su instrumento de recolección de datos: guía estructurada de entrevista. Para los indicadores nivel del manejo de la información, tiempo de reportes de pacientes y respaldo de datos, pertenecientes a la dimensión perspectiva de la enfermera, y los indicadores calidad de atención, tiempo de búsqueda de historia clínica, tiempo de atención, nivel de confianza y seguimiento del paciente pertenecientes a la perspectiva del paciente, la técnica fue la encuesta y su instrumento de recolección de datos fue el cuestionario para cada uno.

3.5. Procedimientos

Primero, para la aplicación de los dos cuestionarios, uno dirigido a las enfermeras y el otro a los apoderados de los pacientes, se emitió una carta de consentimiento al centro de salud para la ejecución del proyecto de ingeniería en el establecimiento, teniendo en cuenta que ambos serían trabajados en dos tiempos distintos, cuando aún no se implementaba el sistema (pre-test) y después de implementado el sistema (post-test). Seguidamente, una vez de contar ya con el permiso, se procedió a aplicar los instrumentos de recolección de datos en el tiempo pre-test; para la aplicación del primer cuestionario se hizo al momento de realizar la reunión de capacitación del software con las enfermeras mediante una encuesta de Google Forms; mientras que el segundo cuestionario se aplicó a los apoderados de pacientes infantes a través de las enfermeras que estaban de turno, realizando la pequeña encuesta al momento de terminar de atender al paciente. Después, para la segunda aplicación de los instrumentos, pero en tiempo post-test, se esperó a que se cumplan los 15 días de uso del sistema, para posteriormente contactarse con las enfermeras para aplicarles el mismo cuestionario, pero tomando en cuenta el impacto que tuvo el sistema en el proceso de gestión de pacientes infantes; y para el segundo cuestionario, también se aplicó a través de las enfermeras.

Por otra parte, mediante grupos de Facebook, red de contactos y LinkedIn se contactó

con tres especialistas en desarrollo de software para que realizaran una evaluación de la arquitectura con la que se elaboró la solución tecnológica; y de esta manera hacer la aplicación de la guía estructurada de entrevista. Se realizaron reuniones independientes por cada especialista en Google Meet, las cuales fueron grabadas con sus consentimientos para tener evidencia de las entrevistas y poder revisar y analizar sus respuestas de acuerdo con las preguntas que se les hicieron.

3.6. Método de análisis de datos

El presente trabajo de investigación usó la estadística descriptiva a partir del procesamiento de la información recolectada resultante de la implementación del pre y post test mediante los instrumentos aplicados y detallados anteriormente.

En el caso del instrumento de cuestionario se realizó una suma del puntaje de las preguntas que este presentaba calificándolas en la escala de Likert.

Para la estadística descriptiva del instrumento de cuestionario se usó gráficos estadísticos, mientras que la inferencial fue realizada con la prueba de normalidad Shapiro-Wilk sobre los indicadores “nivel del manejo de la información”, “tiempo de reportes de pacientes”, “respaldo de datos”, “calidad de atención”, “tiempo de búsqueda de historias clínica”, “tiempo de atención”, “nivel de confianza”, y “seguimiento del paciente”. Después de que se aplicó la prueba de normalidad se identificó si la distribución de los indicadores fue de carácter normal o no normal, por lo que si era normal se aplicaría la prueba de T-Student, tanto para la muestra de pacientes y enfermeras, que es definida por el autor Rodó (2019) como una distribución probabilística que se encarga de estimar el valor representativo de la media de una pequeña muestra tomada de la población que presenta una distribución normal; mientras que, si la distribución no era normal, se aplicaría la prueba no paramétrica Wilcoxon para las enfermeras y U de Mann-Whitney para los pacientes, en donde esta última según Ccanto-Curo et al. (2022) si la significación estadística p es menor a 0.05 y además no presenta una distribución normal, se hace uso de esta prueba para el análisis de la hipótesis.

3.7. Aspectos éticos

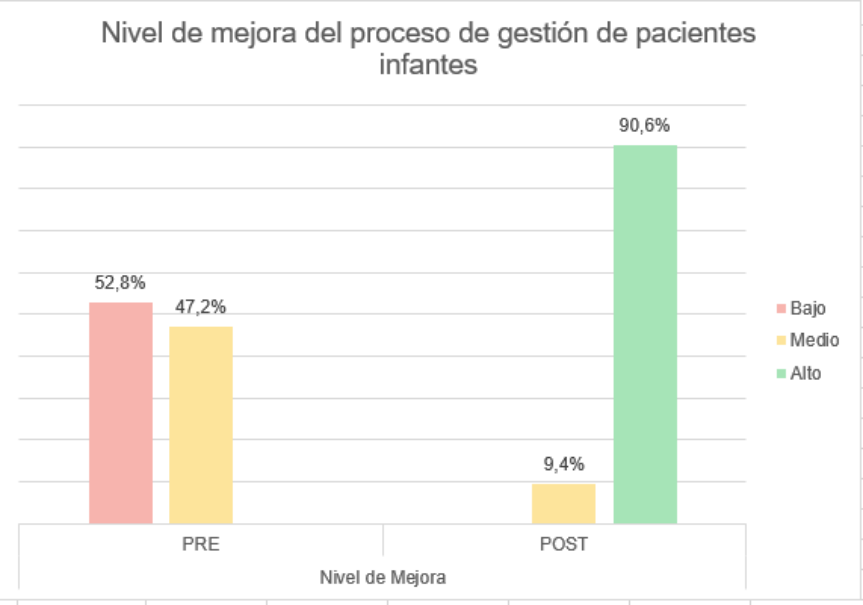
En cuanto a los aspectos éticos que estuvieron involucrados en esta investigación referente al establecimiento de salud estuvo la confidencialidad, debido a que la información que abarcó solo fue utilizada con fines investigativos, tomando muy en consideración su integridad, por lo que sus datos no serán divulgados, ni tendrán fines de lucro, así como también se reservará el nombre del centro de salud en esta investigación. Asimismo, está sujeta a la integridad académica, dado que se ha respetado la autoría de las diferentes teorías consultadas para este estudio, el proceso de citado fue realizado con los formatos ISO 690 y 690-2 según el reglamento de la universidad. Además, se involucró al aspecto de justicia, puesto que el producto elaborado fue entregado a la respectiva institución; y por último el trabajo fue analizado por la herramienta Turnitin, por lo tanto, se garantiza la originalidad.

IV. RESULTADOS

En el presente capítulo se presentaron los resultados de los datos recopilados en relación a los indicadores de estudio, donde se determinó si se produjo algún cambio después de la aplicación del sistema. Como primer paso, se llevó a cabo un Pre-Test mediante la aplicación de los diferentes instrumentos propuestos, que permitieron conocer el estado inicial de los indicadores y poder compararlos con los resultados obtenidos durante el segundo paso, el Post-Test. Permitiendo de esta manera obtener los siguientes resultados durante los dos períodos:

Estadística Descriptiva

Figura 3. Nivel de mejora del proceso de gestión de pacientes infantes

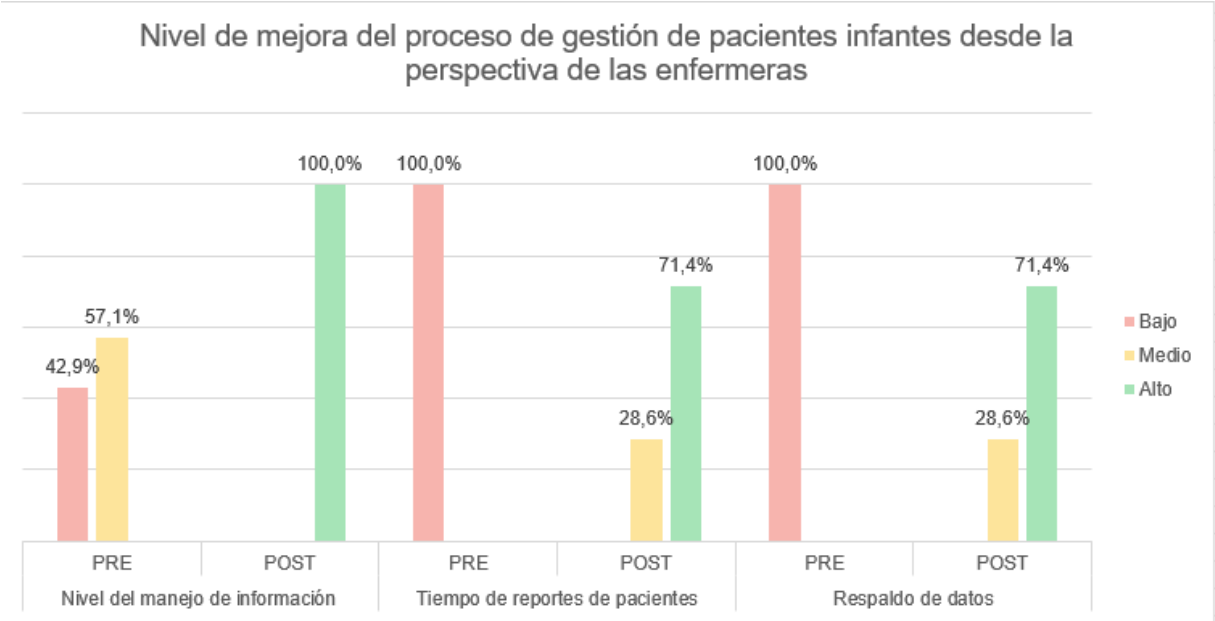


Fuente: Datos obtenidos de los cuestionarios

De acuerdo a la Figura 3, durante la realización de las encuestas a las enfermeras y a los pacientes para evaluar su perspectiva respecto al proceso de gestión de pacientes infantes, de manera general, se encontró que sin utilizar la aplicación web, el 52.8% tenía una valoración baja, y el 47.2% restante indicó una valoración media. Estos resultados revelan que un porcentaje de los individuos no estaban conformes con el proceso de gestión de pacientes infantes. Sin embargo, después de implementar la aplicación web y recolectar datos nuevamente, se observó una mejora significativa. El

90.6% una valoración alta, y el 9.4% tenía una valoración media, lo que demuestra una notable mejora en el proceso de gestión de pacientes infantes.

Figura 4. Nivel de mejora del proceso de gestión de pacientes infantes desde la Perspectiva de la enfermera



Fuente: Datos obtenidos del cuestionario de enfermeras

La Figura 4 muestra los resultados obtenidos durante la aplicación del cuestionario a las enfermeras para evaluar su perspectiva del proceso de gestión de pacientes infantes, se pudo recuperar que:

1. Indicador: Nivel del manejo de información

Según la Figura 4, respecto al manejo de información, se encontró que sin utilizar la aplicación web, el 42.9% tenía una valoración baja, y el 57.1% restante indicó una valoración media. Estos resultados revelaron que un porcentaje de las enfermeras no estaban conformes con el nivel de manejo de información. Sin embargo, después de implementar la aplicación web y recolectar datos nuevamente, se observó una mejora significativa. El 100% de las enfermeras tenía una valoración alta, lo que demostró una notable mejora en el manejo de información.

2. Indicador: Tiempo de reportes de pacientes

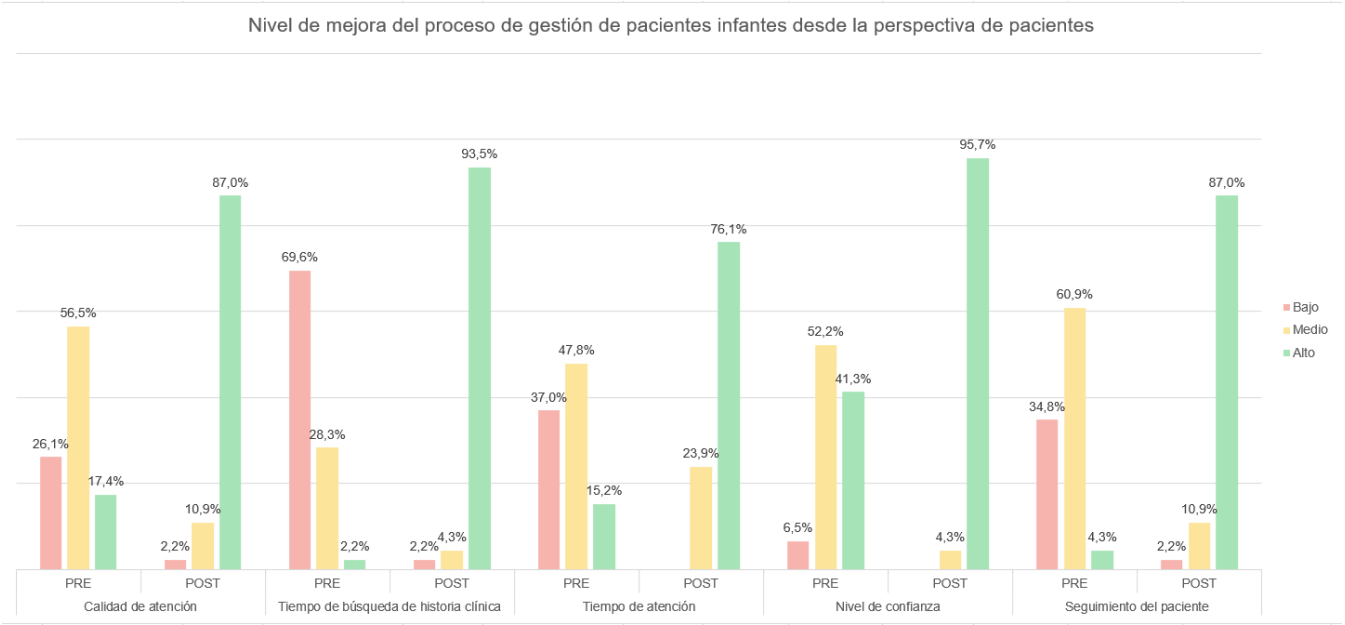
Según la Figura 4, en cuanto al tiempo de reportes de pacientes, sin utilizar la

aplicación web, se encontró que el 100% tenía una valoración baja. Estos resultados revelaron que un porcentaje de las enfermeras no estaban conformes con el tiempo de reportes de pacientes. Sin embargo, después de implementar la aplicación web y recolectar datos nuevamente, se observó una mejora significativa. El 71.4% de las enfermeras tenía una valoración buena, y el 28.6% una valoración media, lo que demostró una mejora considerable en el tiempo de reportes de pacientes.

3. Indicador: Respaldo de datos

Según la Figura 4, respecto al respaldo de datos, sin utilizar la aplicación web, se encontró que el 100% tenía una valoración baja. Estos resultados revelaron que el total de las enfermeras no están conformes con respecto al respaldo de datos. Sin embargo, después de implementar la aplicación web y recolectar datos nuevamente, se observó una mejora significativa. El 71.4% de las enfermeras tenía una valoración buena, y el 28.6% una valoración media, lo que indicó una mejora considerable en cuanto al respaldo de datos.

Figura 5. Nivel de mejora del proceso de gestión de pacientes infantes desde la Perspectiva del paciente



Fuente: Datos obtenidos del cuestionario de pacientes

La Figura 5 muestra los resultados obtenidos durante la aplicación del cuestionario a

los pacientes para evaluar su perspectiva del proceso de gestión de pacientes infantes, se pudo recuperar que:

1. Indicador: Calidad de atención

Según la Figura 5, respecto a la calidad de atención se encontró que, sin utilizar la aplicación web, el 26.1% tenía una valoración baja, el 56.5% manifestó una valoración media, y el 17.4% restante indicó una valoración buena. Estos resultados revelaron que un porcentaje de los pacientes no estaban conformes con la calidad de atención. Sin embargo, después de implementar la aplicación web y recolectar datos nuevamente, se observó una mejora significativa. El 87% de los pacientes tenía una valoración buena, el 10.9% una valoración media, y el 2.2% una valoración mala, lo que demostró una notable mejora en la calidad de atención.

2. Indicador: Tiempo de búsqueda de historia clínica

Según la Figura 5, en cuanto al tiempo de búsqueda de historia clínica se encontró que, sin utilizar la aplicación web, el 69.6% tenía una valoración mala, el 28.3% una valoración media, y el 2.2% restante indicó una valoración buena. Estos resultados revelaron que un porcentaje de los pacientes no estaban conformes con el tiempo de búsqueda de historia clínica. Sin embargo, después de implementar la aplicación web y recolectar datos nuevamente, se observó una mejora significativa. El 93.5% de los pacientes tenía una valoración buena, 4.3% una valoración media, y el 2.2% tenía una valoración mala, lo que demostró una notable mejora en el tiempo de búsqueda de historia clínica.

3. Indicador: Tiempo de atención

Según la Figura 5, respecto al tiempo de atención se encontró que, sin utilizar la aplicación web, el 37% manifestó una valoración mala, el 47.8% una valoración media, y el 15.2% restante indicó una valoración buena. Estos resultados revelaron que un porcentaje de los pacientes no estaban conformes con el tiempo de atención. Sin embargo, después de implementar la aplicación web y recolectar datos nuevamente, se observó una mejora significativa. El 76.1% de los pacientes tenía una valoración buena, y el 23.9% tenía una valoración media, lo que demostró una notable mejora en

el tiempo de atención.

4. Indicador: Nivel de confianza

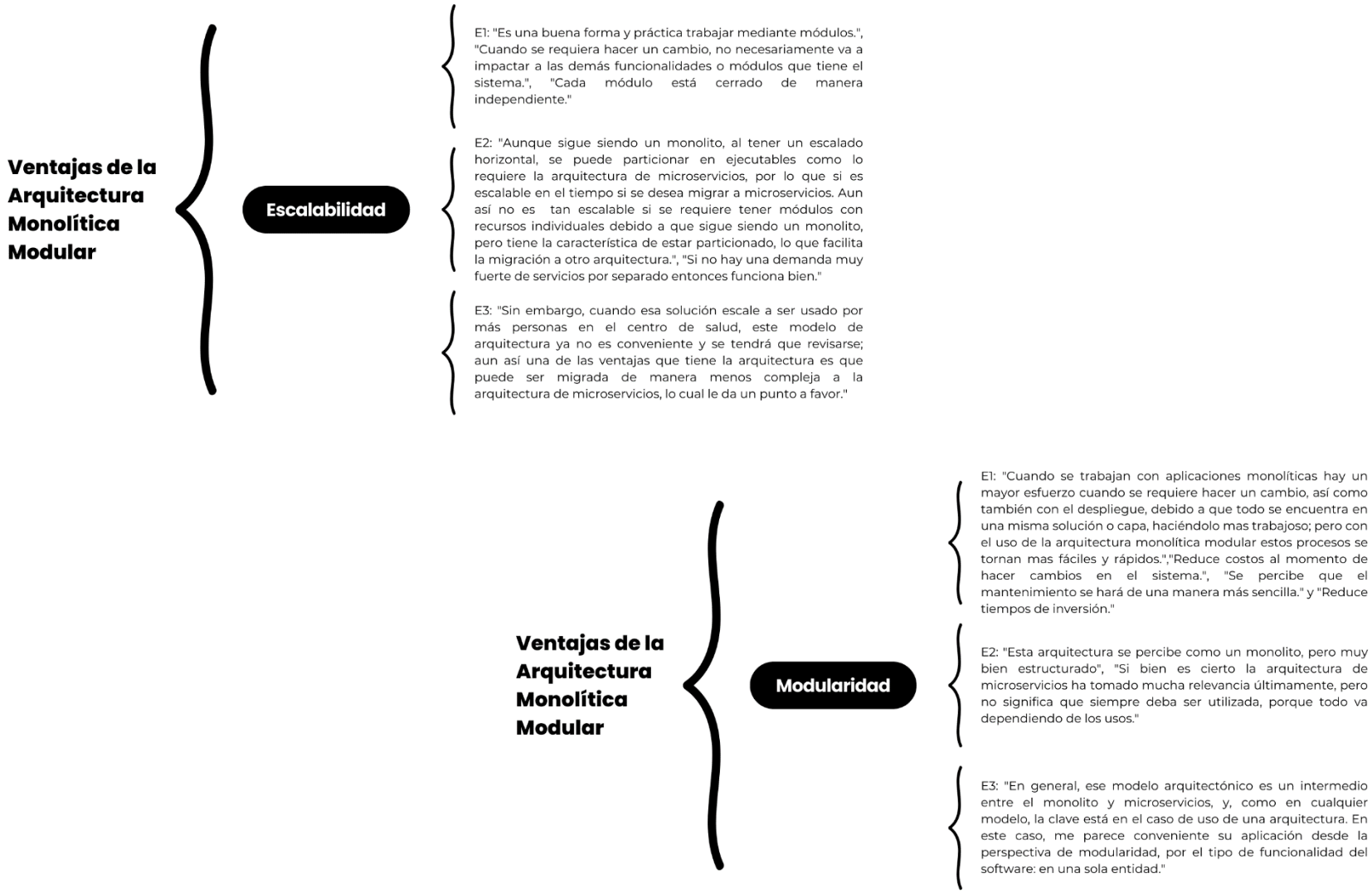
Según la Figura 5, respecto al nivel de confianza se encontró que, sin utilizar la aplicación web, el 52.2% tenía una valoración media, el 41.3% una buena, y el 6.5% restante indicó una valoración mala. Estos resultados revelaron que un porcentaje de los pacientes no están conformes con el nivel de confianza. Sin embargo, después de implementar la aplicación web y recolectar datos nuevamente, se observó una mejora significativa. El 96.7% de los pacientes tenía una valoración buena, y el 4.3% tenía una valoración media, lo que demostró una notable mejora en el nivel de confianza.

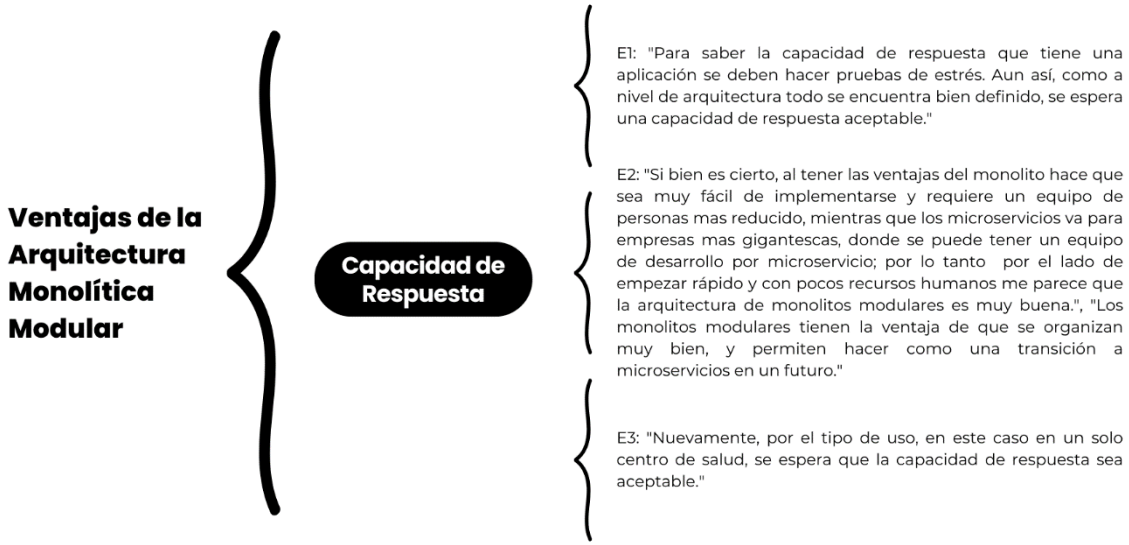
5. Indicador: Seguimiento del paciente

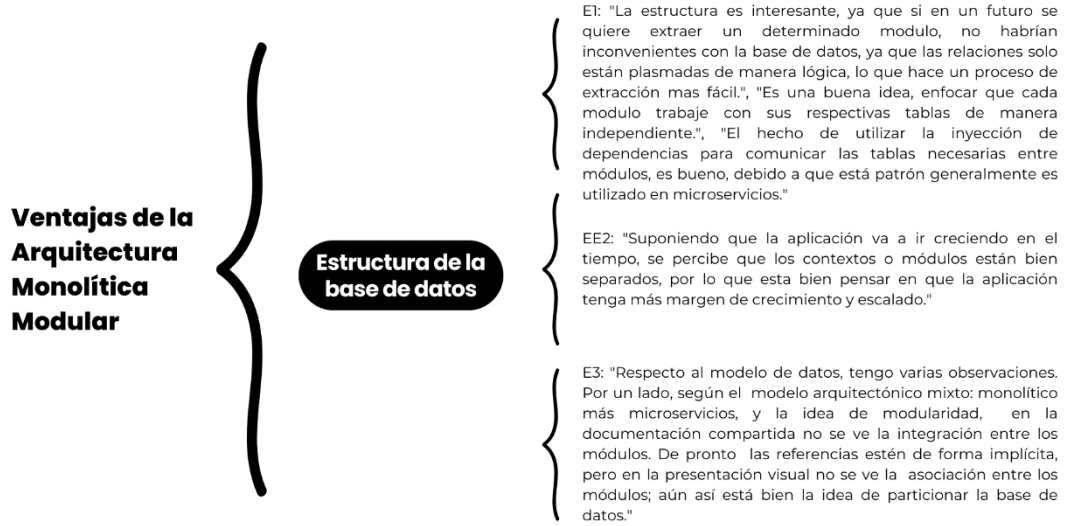
Según la Figura 5, en cuanto al indicador seguimiento del paciente se encontró que, sin utilizar la aplicación web, el 60.9% tenía una valoración media, el 34.8% una valoración mala, y el 4.3% restante indicó una valoración buena. Estos resultados revelaron que un porcentaje de los pacientes no están conformes con el seguimiento del paciente. Sin embargo, después de implementar la aplicación web y recolectar datos nuevamente, se observó una mejora significativa. El 87% de los pacientes tenía una valoración buena, 10.9% tiene una valoración media, y el 2.2% tenía una valoración mala, lo que demostró una notable mejora en el seguimiento del paciente.

Indicadores: Modularidad, escalabilidad, capacidad de respuesta y estructura de la base de datos.

Figura 6. Ventajas de la arquitectura de monolitos modulares







Fuente: entrevistas

Según lo mencionado por los entrevistados, respecto a la escalabilidad, la arquitectura modular permite trabajar de manera eficiente y práctica, ya que los cambios en un módulo no afectan necesariamente a otros. Aunque sigue siendo un monolito, se puede escalar horizontalmente mediante la partición en ejecutables, lo que facilita una eventual migración a microservicios. Sin embargo, si se necesitan módulos con recursos individuales, esta arquitectura puede no ser tan escalable. A medida que la solución se utiliza por más personas en el centro de salud, esta arquitectura se vuelve menos conveniente y será necesario revisarla. Afortunadamente, una ventaja es que puede migrarse de manera menos compleja a la arquitectura de microservicios, lo cual representa una opción favorable para su evolución.

En cuanto a la estructura de la base de datos, presenta ciertas ventajas y consideraciones. Por un lado, al estar basada en relaciones lógicas, facilita la extracción de módulos en el futuro, ya que no hay dependencias fuertes entre ellos. Es positivo que cada módulo trabaje de manera independiente con sus propias tablas y se utilice la inyección de dependencias para comunicar las tablas necesarias, lo cual es comúnmente utilizado en microservicios. En cuanto al crecimiento y escalado de la aplicación, se percibe una adecuada separación y definición de los contextos o módulos, lo que indica que existe margen para un mayor crecimiento y escalabilidad. Sin embargo, se observa una falta de integración explícita entre los módulos en la documentación compartida y la presentación visual. Aunque las referencias pueden estar implícitas, sería beneficioso mostrar de manera clara y visual la asociación entre los módulos. A pesar de esta observación, la idea de particionar la base de datos sigue siendo válida y puede ser una base sólida para la evolución del sistema.

A nivel de modularidad, la arquitectura monolítica modular ofrece numerosas ventajas en comparación con las aplicaciones monolíticas tradicionales. Al dividir el sistema en módulos independientes, se simplifica y agiliza tanto el proceso de desarrollo como el despliegue de cambios. Esto conlleva una reducción de costos y tiempos de inversión, así como un mantenimiento más sencillo. Si bien la arquitectura monolítica modular sigue siendo un enfoque diferente al de los microservicios, no significa que deba descartarse en todos los casos. La elección de una arquitectura depende de los casos

de uso y las necesidades específicas del sistema. En este contexto, la arquitectura monolítica modular resulta conveniente debido a la naturaleza de la funcionalidad del software en una única entidad.

En torno a la capacidad de respuesta de una aplicación, se puede evaluar a través de pruebas de estrés, aunque se espera que la arquitectura modular de monolitos proporcione una respuesta aceptable debido a la forma en que está definida. La arquitectura de monolitos modulares ofrece ventajas significativas, especialmente en términos de implementación rápida y eficiente con equipos más pequeños. A diferencia de los microservicios, que suelen ser más adecuados para empresas más grandes con equipos de desarrollo por microservicio, los monolitos modulares son una buena opción cuando se busca comenzar rápidamente con recursos humanos limitados. Además, los monolitos modulares permiten una transición más suave hacia una arquitectura de microservicios en el futuro, lo que los convierte en una opción versátil. Dado el contexto de uso en un solo centro de salud, se espera que la respuesta del sistema sea aceptable, tomando en cuenta las características y necesidades específicas de dicho entorno.

Análisis Inferencial

Objetivo general

Prueba de Normalidad

Tabla 1. *Prueba de normalidad de la variable Proceso de gestión de pacientes infantes*

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
PRE_TEST_GENERAL	,091	53	,200*	,979	53	,478
POST_TEST_GENERAL	,158	53	,002	,917	53	,001

*. Esto es un límite inferior de la significación verdadera.

a. Corrección de significación de Lilliefors

Fuente: Datos obtenidos de los cuestionarios

Los resultados de la tabla 1 muestran que, para el primer indicador, la prueba de normalidad realizada en los dos periodos arrojó un nivel de significancia de 0.200 y 0.002 respectivamente, lo que indica que el primero tiene una distribución normal ya que es mayor que el valor de significancia establecido de 0.05, y el segundo valor tiene una distribución no normal ya que es menor al valor de significancia establecido de 0.05. Por lo tanto, se realizaron las pruebas no paramétricas, ya que uno de los valores tiene una distribución no normal usando la prueba de U de Mann-Whitney.

Comprobación de hipótesis general:

1. Planteamiento de hipótesis

H0: El proceso de gestión de pacientes infantes no mejora significativamente al implementar una PWA usando la arquitectura de monolitos modulares en el centro de salud.

HA: El proceso de gestión de pacientes infantes mejora significativamente al implementar una PWA usando la arquitectura de monolitos modulares en el centro de salud.

2. Fijación de α

α > 0.05 Normal => Se reconoce la hipótesis nula

α < 0.05 No Normal => Se reconoce la hipótesis alterna.

3. Estadístico de prueba

Se empleó el test U de Mann-Whitney para analizar los datos proporcionados durante el estudio del proceso de gestión de pacientes infantes.

Tabla 2. *Rangos para la variable Proceso de gestión de pacientes infantes*

	GRUPO	N	Rango promedio	Suma de rangos
TEST_GENERAL	Pre_test	53	27,12	1437,50
	Post_test	53	79,88	4233,50
	Total	106		

Fuente: Datos obtenidos de los cuestionarios

Tabla 3. *Estadísticos de prueba para la variable Proceso de gestión de pacientes infantes*

	TEST_GENERAL
U de Mann-Whitney	6,500
W de Wilcoxon	1437,500
Z	-8,844
Sig. asintótica(bilateral)	,000

a. Variable de agrupación: GRUPO

Fuente: Datos obtenidos de los cuestionarios

4. Decisión estadística

La prueba de hipótesis reveló que el valor p asintótico en el proceso de gestión de pacientes infantes fue de 0.000, y el valor de Z fue de -8.844. Como resultado, la hipótesis alternativa fue aceptada.

5. Conclusión

La perspectiva general respecto al proceso de gestión de pacientes infantes mejora significativamente con la implementación de la PWA que utiliza la arquitectura de monolitos modulares.

Objetivo específico 1 - Dimensión Perspectiva de la enfermera

Tabla 4. *Pruebas de normalidad de la dimensión Perspectiva de la enfermera*

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
PRE_TEST	,323	7	,026	,734	7	,009
POST_TEST	,279	7	,105	,868	7	,177

a. Corrección de significación de Lilliefors

Fuente: Datos obtenidos del cuestionario de enfermeras

Los resultados de la tabla 4 muestran que, para el primer indicador, la prueba de normalidad realizada en los dos periodos arrojó un nivel de significancia de 0.009 y 0.177 respectivamente, lo que indica que el primero tiene una distribución no normal ya que es menor que el valor de significancia establecido de 0.05, y el segundo valor tiene una distribución normal ya que es mayor al valor de significancia establecido de 0.05. Por lo tanto, se realizaron las pruebas no paramétricas, ya que uno de los valores tiene una distribución no normal usando la prueba de Wilcoxon.

Comprobación de hipótesis de específica 1:

1. Planteamiento de hipótesis

H0: La perspectiva que tiene el personal de salud respecto al proceso de gestión de pacientes infantes no mejora significativamente con la implementación de la PWA que utiliza la arquitectura de monolitos modulares.

HA: La perspectiva que tiene el personal de salud respecto al proceso de gestión de pacientes infantes mejora significativamente con la implementación de la PWA que utiliza la arquitectura de monolitos modulares.

2. Fijación de α

α > 0.05 Normal => Se reconoce la hipótesis nula

α < 0.05 No Normal => Se reconoce la hipótesis alterna.

3. Estadístico de prueba

Se empleó el test de Wilcoxon para analizar los datos proporcionados durante el estudio de la dimensión Perspectiva de la Enfermera.

Tabla 5. *Rangos de la dimensión Perspectiva de la enfermera*

		N	Rango promedio	Suma de rangos
POST_TEST - PRE_TEST	Rangos negativos	0[a]	,00	,00
	Rangos positivos	7[b]	4,00	28,00
	Empates	0[c]		
	Total	7		

a. POST_TEST < PRE_TEST

b. POST_TEST > PRE_TEST

c. POST_TEST = PRE_TEST

Fuente: Datos obtenidos del cuestionario de enfermeras

Tabla 6. *Estadísticos de prueba de la dimensión Perspectiva de la enfermera*

	POST_TEST - PRE_TEST
Z	-2,371[b]
Sig. asintótica(bilateral)	,018

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos negativos.

Fuente: Datos obtenidos del cuestionario de enfermeras

4. Decisión estadística

La prueba de hipótesis reveló que el valor p asintótico en la dimensión "Perspectiva de la enfermera" fue de 0.018, y el valor de Z fue de -2.371. Como resultado, la hipótesis alternativa fue aceptada.

5. Conclusión

La perspectiva que tiene el personal de salud respecto al proceso de gestión de pacientes infantes mejora significativamente con la implementación de la PWA que utiliza la arquitectura de monolitos modulares.

Objetivo específico 2 - Dimensión Perspectiva del paciente

Tabla 7. *Pruebas de normalidad de la dimensión Perspectiva del paciente*

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
PRE_TEST	,101	46	,200*	,983	46	,742
POST_TEST	,163	46	,004	,905	46	,001

*. Esto es un límite inferior de la significación verdadera.

a. Corrección de significación de Lilliefors

Fuente: Datos obtenidos del cuestionario de pacientes

Los resultados de la tabla 7 muestran que, para el primer indicador, la prueba de normalidad realizada en los dos periodos arrojó un nivel de significancia de 0.742 y 0.001 respectivamente, lo que indica que el primero tiene una distribución normal ya que es mayor que el valor de significancia establecido de 0.05, y el segundo valor tiene una distribución no normal ya que es menor al valor de significancia establecido de 0.05. Por lo tanto, se realizaron las pruebas no paramétricas, ya que uno de los valores tiene una distribución no normal usando la prueba de U de Mann-Whitney.

Comprobación de hipótesis de específica 2:

1. Planteamiento de hipótesis

H0: La perspectiva que tienen los apoderados respecto al proceso de gestión de pacientes infantes no mejora significativamente con la implementación de la PWA que utiliza la arquitectura de monolitos modulares.

HA: La perspectiva que tienen los apoderados respecto al proceso de gestión de pacientes infantes mejora significativamente con la implementación de la PWA que utiliza la arquitectura de monolitos modulares.

2. Fijación de α

α > 0.05 Normal => Se reconoce la hipótesis nula

α < 0.05 No Normal => Se reconoce la hipótesis alterna.

3. Estadístico de prueba

Se empleó el test U de Mann-Whitney para analizar los datos proporcionados durante el estudio de la dimensión Perspectiva del Paciente.

Tabla 8. *Rangos de la dimensión Perspectiva del paciente*

	Condicion	N	Rango promedio	Suma de rangos
TEST	Pre_test	46	24,99	1149,50
	Post_test	46	68,01	3128,50
	Total	92		

Fuente: Datos obtenidos del cuestionario de pacientes

Tabla 9. *Estadísticos de prueba de la dimensión de Perspectiva de pacientes*[a]

	TEST
U de Mann-Whitney	68,500
W de Wilcoxon	1149,500
Z	-7,741
Sig. asintótica(bilateral)	,000

a. Variable de agrupación: Condicion

Fuente: Datos obtenidos del cuestionario de pacientes

4. Decisión estadística

La prueba de hipótesis reveló que el valor p asintótico en la dimensión "Perspectiva del paciente" fue de 0.00, y el valor de Z fue de -7.741. Como resultado, la hipótesis alternativa fue aceptada.

5. Conclusión

La perspectiva que tienen los apoderados respecto al proceso de gestión de pacientes infantes mejora significativamente con la implementación de la PWA que utiliza la arquitectura de monolitos modulares.

V. DISCUSIÓN

Respecto al objetivo general, se logró demostrar que el proceso de gestión de pacientes infantes mejoró significativamente, de tal manera que después de la implementación del sistema alcanzó un nivel de mejora alto, con un 90,6% de perspectiva general. Esto se relaciona con la investigación de Peralta (2019), quien implementó un sistema informático que permitió registrar y controlar las historias clínicas mejorando el proceso de atención hacia los pacientes gracias a la reducción de los tiempos, permitiendo encontrar y registrar la información de manera más eficiente. Se relaciona además con la investigación de Goyburo (2018), dónde implementó un sistema informático que ayudó a mejorar la gestión de pacientes; esto se evidenció principalmente en el proceso de registro de datos, el cual solía ser más lento y presentaba problemas al recolectar información de manera autónoma, pero, con la implementación del sistema, se obtuvo una reducción del tiempo empleado en el registro de pacientes y en el registro de historial clínico. La investigación se relaciona con el concepto teórico de gestión de pacientes infantes puesto que se logró la sistematización de este proceso y engloba la perspectiva del paciente y del personal de salud, que según el autor Chavez (2020), es una estrategia que da la posibilidad de sistematizar y ordenar los diferentes procesos que se dan en una atención sanitaria para las decisiones teniendo en cuenta a los pacientes y según el autor Morales (2019), la gestión de pacientes cuenta con dos perspectivas: la primera que engloba de manera general al paciente y al proceso, y la segunda referente a la práctica con el personal de salud.

Haciendo referencia al primer objetivo específico, se obtuvieron por resultados que al implementar el sistema, cambió la perspectiva del personal de salud en el proceso de gestión de pacientes infantes respecto al nivel de manejo de la información por parte de las enfermeras, lográndose una mejora en el manejo de las historias clínicas en un 57.1%, los datos son comparados con lo concluido por cierto autor Rivera et al. (2020) quien indicó que con el sistema implementado los tiempos de registro de citas médicas fueron más eficientes concluyendo que una PWA trajo consigo mayor portabilidad al sistema, abarcando una mayor cantidad de pacientes y logrando una mejora

significativa del tiempo de espera al registrar citas médicas, además de reducir el personal necesario para la atención de pacientes. Se relaciona también con el concepto teórico de sistema de control de pacientes que según detalló el autor Morales (2019), las acciones como son búsqueda y registro de historias clínicas se realizarán más fluidamente impactando de manera beneficiosa a la gestión de pacientes porque la obtención de información se hace de manera segura, sencilla y rápida. Asimismo, según el indicador de la investigación, se logró reducir la pérdida de datos en el proceso de gestión de pacientes infantes ya que antes el 100% no estaban conformes con este tema, pero después un 71.4% tenían una perspectiva diferente sobre esto, lo cual se puede relacionar con lo obtenido por el autor Tajudeen et al. (2021), dónde su sistema logró evitar la pérdida de expedientes de los pacientes, así como una mayor seguridad para los mismos, reduciendo las colas durante los registros, minimizando el uso del papel y facilitando los servicios de atención médica, además de reducir el dinero invertido en documentos y útiles para escribir. Menciona también, que un sistema bien diseñado representa una herramienta potente para el establecimiento de salud, gracias al proporcionar mejoras en los servicios, control de costos y aprovechamiento de las instalaciones de la institución. En cuanto al último indicador de la investigación, se obtuvo una mejora significativa respecto al tiempo que toma la generación de reportes de pacientes, y esto debido a que después de la implementación tomó mucho menos tiempo que de manera manual, donde causaba un 100% de inconformidad por parte de las enfermeras, pasando a un 71.4% de mejora respecto a los reportes, esto se puede aseverar con lo que precisó el autor Huamani (2018) quien mencionó que la implementación del sistema permitió solucionar los problemas de atenciones hacia los pacientes, asimismo de la cantidad de atenciones por día en el hospital, aludiendo que mediante el sistema se lograban generar reportes automáticos beneficiando al personal de salud para las distintas áreas del hospital.

De acuerdo al segundo objetivo específico de la investigación, se identificó ciertos indicadores que debían medirse para completar el proceso de gestión de pacientes infantes, los cuales causaban una disconformidad por parte de los pacientes, pero gracias a la implementación del sistema el tiempo de búsqueda de historia clínica se

redujo mostrando un 93.5% de mejora en este indicador, evidenciando que el tiempo de atención de los pacientes pasó de un 47.8% con valoración media a 76.1% con valoración buena, esto se relacionó con el estudio realizado de un autor Oñate et al. (2020), quien desarrolló una PWA para gestionar pruebas que simulen el ingreso a universidades e instituciones militares, dónde gracias a la implementación de la aplicación se logró ahorrar recursos y tiempo del personal en la institución, simultáneamente se prescindió de usar hojas y cuadernillos diariamente, lo cual generaba malestar por el esfuerzo y tiempo dedicados. De acuerdo a otro indicador de la presente investigación el cual fue calidad de atención, se mejoró en un 60.9% respecto a la perspectiva anterior, esto se relacionó con uno de los resultados que se detallaron en el estudio realizado por Morales (2019), quien desarrolló un sistema para la gestión de historias clínicas que guarda la información de los pacientes del centro de salud, dónde el mayor impacto estuvo en la atención de pacientes, pues el consultar información se pudo hacer de manera completa y rápida, mejorando así la calidad de las atenciones del centro de salud; además se relacionó con el concepto teórico de atención de pacientes infantes que según el autor Ramirez (2020), se necesita brindar una atención de calidad para conseguir una percepción satisfactoria y se cumpla el respectivo control de la mejor manera posible. Según el indicador seguimiento del paciente, se obtuvo una mejora significativa respecto a las vacunas o atenciones que este necesitaba, pues el seguimiento se volvió mucho más dinámico y fácil de realizar pasando de un 34.8% con valoración mala a un 87% con valoración buena, esto se relaciona con la investigación de Muñoz (2020), donde diseñó e implementó un sistema que permite la optimización de la gestión de citas, luego de la implementación de este sistema, se logró optimizar de manera considerable la reserva, gestión y el tiempo de las atenciones, así como el seguimiento de los pacientes de dicha clínica. Respecto al indicador nivel de confianza, gracias a la implementación del sistema, se obtuvo una mejora del nivel de confianza de los pacientes hacia el centro de salud en un 44.5% de aumento en la perspectiva, todo gracias a lo mencionado anteriormente, esto se asocia con la investigación de Rêgo et al. (2021) que tenía el objetivo de lograr la confianza de los lectores respecto a las PWA como una solución confiable a la industria de la salud, donde se presentó un software que es la prueba contundente de

que las PWA proporcionan características necesarias para mejorar la calidad de la atención de salud, reduciendo errores médicos y agilizando diferentes procesos clínicos, mejorando así la confianza de los pacientes.

En cuanto al análisis de las ventajas de la arquitectura de monolitos modulares utilizada en el desarrollo del sistema, mediante las entrevistas se determinó que: a diferencia de los microservicios, que suelen ser más adecuados para empresas más grandes con equipos de desarrollo por microservicio, los monolitos modulares son una buena opción cuando se busca comenzar rápidamente con recursos humanos limitados, y así de esta manera evitar la complejidad que trae la arquitectura de microservicios; además la arquitectura permite trabajar de manera eficiente y práctica, debido a que los cambios en un módulo no afectan necesariamente a otros porque son independientes; lo cual es contrastado por la teoría del autor Newman (2019), el cual menciona que los monolitos modulares consisten en módulos separados, en los que cada uno de ellos puede ser trabajado de forma independiente, pero aún deben ser combinados para su implementación. Asimismo, menciona algunas de las ventajas de la arquitectura de monolitos modulares, donde resalta que, para muchas de las organizaciones, la arquitectura de monolitos modulares puede ser una de las mejores opciones, y esto debido a que si los módulos se encuentran bien delimitados y definidos puede proporcionar un alto nivel de trabajo en paralelo, pero evitando los desafíos que da la arquitectura de microservicios con su distribución y las preocupaciones de implementación, volviendo más simple la construcción del software en estos aspectos. Por otra parte, otra de las ventajas encontradas fue respecto a la estructura de la base de datos, mencionando que, al estar basada en relaciones lógicas, facilita la extracción de módulos en el futuro, debido a que no hay dependencias fuertes entre ellos, lo cual es algo positivo, ya que cada módulo trabaja de manera independiente con sus propias tablas y se utiliza la inyección de dependencias para comunicar las tablas necesarias, lo cual es comúnmente utilizado en microservicios. Esto es algo favorable debido a que según el autor Newman (2019), resalta que uno de los más grandes desafíos de los monolitos modulares es que la base de datos carece de la descomposición que se da en los módulos a nivel de

código, lo cual es un inconveniente si en un futuro se desea extraer el monolito y migrar a una arquitectura de, por ejemplo, microservicios.

VI. CONCLUSIONES

Después de realizar el análisis de los resultados obtenidos, se presentan las siguientes conclusiones:

1. Acerca de la mejora del proceso de gestión de pacientes infantes al implementar la aplicación web progresiva usando la arquitectura de monolitos modulares en el centro de salud, se evidenció una mejora significativa (P = 0.000) en las dimensiones estudiadas luego de la implementación del sistema; por lo que se puede concluir que el proceso de gestión de pacientes infantes mejoró, tanto desde la perspectiva de pacientes, como el de las enfermeras.

2. En cuanto al primer objetivo específico, la implementación de la PWA mejoró la perspectiva del personal de salud respecto al proceso de gestión de pacientes infantes de manera significativa (P = 0.018), demostrándose un nivel de mejora alto en el manejo de la información, tiempo de reportes de pacientes y el respaldo de los datos del centro de salud.

3. Acerca del segundo objetivo específico, la implementación de la PWA mejoró la perspectiva de los apoderados de pacientes respecto al proceso de gestión de pacientes infantes de manera significativa (P = 0.000), demostrándose un nivel de mejora alto en la calidad de atención, tiempo de búsqueda de historias clínicas, y tiempo de atención, así como también en el nivel de confianza y el seguimiento realizado a los pacientes del centro de salud.

4. Referente al tercer objetivo específico, dentro de las ventajas de utilizar la arquitectura de monolitos modulares en el desarrollo de la PWA, se encontró que permite cambios eficientes y prácticos, debido a que los cambios en un módulo no afectan necesariamente a otros; la estructura de la base de datos facilita la extracción de módulos y la comunicación entre ellos; ofrece ventajas en desarrollo, despliegue y mantenimiento en comparación con las aplicaciones monolíticas tradicionales; y aunque no es lo mismo que los microservicios, es una opción versátil para comenzar rápidamente con recursos limitados y facilitar la transición hacia otra arquitectura de microservicios en el futuro.

VII. RECOMENDACIONES

Se recomienda a futuros investigadores especialistas en arquitectura de software que, para obtener resultados más precisos en cuanto a los indicadores de la arquitectura de monolitos modulares, se realice una comparación entre esta arquitectura y otras arquitecturas convencionales utilizadas en centros de salud en términos de eficiencia, calidad del cuidado y satisfacción del paciente. De esta manera. involucraría grupos de control que utilicen diferentes enfoques arquitectónicos para permitir una comparación más precisa.

Para futuros investigadores y profesionales, si se quiere tener un mayor control sobre la investigación, se puede optar por un diseño cuasiexperimental; de esta manera se seleccionan dos departamentos, áreas o grupos de la empresa que sean comparables en términos de tamaño y otros factores relevantes; de tal forma que se pueda medir la eficiencia del proceso de gestión de pacientes en ambos grupos antes y después de la implementación del sistema, para obtener una idea más sólida de los efectos del sistema.

Como investigador en el campo de la arquitectura de sistemas de información, se puede evaluar la posibilidad de transferir la arquitectura de Monolitos Modulares a otros centros de salud y evaluar su aplicabilidad y beneficios en diferentes contextos. Esto podría implicar un análisis de factibilidad y considerar los desafíos y adaptaciones necesarias para su implementación en otros entornos de atención médica.

Para futuras investigaciones que puedan tomar a este proyecto realizado en el centro de salud, se podría realizar un seguimiento a largo plazo para evaluar los efectos sostenidos de la intervención. Esto permitirá obtener una visión más completa de los beneficios y limitaciones de la arquitectura de Monolitos Modulares a lo largo del tiempo.

REFERENCIAS

Agile methodologies in software maintenance: a systematic review. **Tarwani, Sandhya and Chug, Anuradha. 2016.** 4, 2016, Vol. 40. 1854-3871.

Aguirre, Verónica, et al. 2019. *PWA para unificar el desarrollo Desktop, Web y Mobile.* Facultad de informática, Instituto de Investigación en Informática LIDI (III-LIDI). Facultad de Informática – Universidad Nacional de La Plata. La Plata : s.n., 2019. p. 9.

Alabor, Manuel and Stolze, Markus. 2020. *Debugging of RxJS-based applications.* New York, NY, USA : Association for Computing Machinery, 2020. pp. 15–24. 978-1-4503-8188-8.

Allen, Stephanie. 2022. Deloitte. [Online] Enero 2022. [Cited: Agosto 7, 2022.] https://www2.deloitte.com/pe/es/pages/life-sciences-and-healthcare/articles/perspectivas-del-sector-cuidado-de-la-salud-2022.html.

Angular.io. 2022. Angular. [Online] Febrero 28, 2022. [Cited: Julio 29, 2022.] https://angular.io/guide/what-is-angular.

Arias Muñoz, Marco Antonio. 2018. *Desarrollo de una aplicación web para la mejora del control de asistencia de personal en la Escuela Tecnológica Superior de la Universidad Nacional de Piura.* Universidad Inca Garcilaso de la Vega, Lima, Perú : 2018.

Bipin, Joshi. 2016. *Beginning SOLID Principles and Design Patterns for ASP.NET Developers.* s.l. : Apress, 2016. p. 415. 978-1-4842-1848-8.

Chandel, Munish. 2018. *Cracking Spring Microservices Interviews: A quick refresher for Java and Spring Cloud Developers.* s.l. : Shunya Foundation, 2018. p. 168. VWVTPZ3A.

Comparación de tendencias tecnológicas en aplicaciones web. **Valarezo Pardo, Milton Rafael, et al. 2018.** 3, Alcoy : 3ciencias, 2018, Vol. 7. 2254-3376.

Designing and Implementation of a Computerized Patient Management System for University Health Service of Federal University Of Technology Akure, Ondo State. **Tajudeen Temitayo, Adebayo and T, B llori. 2021.** 1, Akure : s.n., 2021, Vol. 3. 2505-0141.

Eficacia, efectividad e impacto en vacunas: ¿es lo mismo? **Giglio, Norberto, Bakir, Julia and Gentile , Angela. 2018.** Buenos Aires : Revista pediatrica HNRG, 2018. 2314-1239.

Eficiencia y sostenibilidad en la gestión clínica en el Perú en tiempos de pandemia. **Chavez, Annil. 2020.** 2, Lima : South Sustainability, Diciembre 31, 2020, South

Sustainability, Vol. 1, p. 4. 2708-7077.

Espinoza Luna, Teresa Adith and Valderrama Marin, Cynthia Pilar. 2019. *Factores sociales, culturales e institucionales en la irregularidad de los controles de crecimiento y desarrollo del niño menor de 1 año del Centro de Salud Perú Korea Bellavista – Callao, 2019.* Universidad Nacional del Callao, Callao : 2019.

Estadística aplicada a la investigación educativa. **Gamboa Graus, Michel Enrique. 2018.** Enero 1, 2018. 2007-7890.

Fu, Cheng. 2018. State Management with NgRx. *Build Mobile Apps with Ionic 4 and Firebase: Hybrid Mobile App Development.* Berkeley, CA : 978-1-4842-3775-5, 2018.

Garreau, Marc and Faurot, Will. 2018. *Redux in Action.* s.l. : Simon and Schuster, 2018. p. 463. 978-1-63835-625-7.

Goyburo Moscoso, Bruno Alexander. 2018. *Diseño e implementación de un sistema informático para la gestión de pacientes en el consultorio pediátrico del doctor torres, tumbes 2017.* Universidad Católica Los Ángeles De Chimbote, Tumbes : 2018.

Gruhn, Volker and Striemer, Rüdiger. 2018. *The Essence of Software Engineering.* Primera. Berlín : Springer, 2018. p. 251. 978-3030088804.

Haro, E, et al. 2019. *Backend development for web applications, restful web services: Node.js vs spring boot.* Brazil : 309RISTI, N.º E17, 2019. 1646-9895.

Huamani Gonzales, Clinton Juvenal. 2018. *Análisis y Diseño de un Sistema de Gestión hospitalaria para mejorar el proceso de atención a pacientes en el Hospital Santa María del Socorro de Ica.* Universidad Nacional San Luis Gonzaga de Ica, Ica, Perú : 2018.

Joshi, Bipin. 2016. *Beginning SOLID Principles and Design Patterns for ASP.NET Developers.* s.l. : Apress, 2016. 978-1-4842-1848-8.

Klimm, Marvin Christian. 2021. *Design Systems for Micro Frontends.* University of Applied Sciences, Gummersbach : 2021.

Marques Fernandes, Henrique. 2020. ¿Qué es un sistema/aplicación monolito/monolítico? [Online] Julio 08, 2020. [Cited: Julio 29, 2022.] https://marquesfernandes.com/es/tecnologia-es/o-que-e-um-sistema-aplicacao-monolith-monolithic/.

Mensah, Oliver. 2019. Creating Beautiful Apps with Angular Material. [Online] Setiembre 14, 2019. [Cited: Julio 29, 2022.] https://auth0.com/blog/creating-beautiful-apps-with-angular-material/.

Ministerio de Salud. 2020. *Documento Técnico: Agenda Digital del Sector Salud 2020-2025.* Lima, Ministerio de Salud. Lima : s.n., 2020. p. 54.

—. 2015. *Infrastructura y equipamiento de los establecimientos de salud de primer nivel de atención.* Perú : Dirección general de infraestructura, equipamiento y mantenimiento, 2015.

Morales Larrañaga, Carlos Alberto. 2019. *SISTEMA WEB PARA LA GESTIÓN DE PACIENTES EN LA CLÍNICA DENTAL MY DENTIST CORIN, CALLAO - 2019SISTEMA WEB PARA LA GESTIÓN DE PACIENTES EN LA CLÍNICA DENTAL MY DENTIST CORIN, CALLAO - 2019.* UNIVERSIDAD PRIVADA TELESUP, Lima : UNIVERSIDAD PRIVADA TELESUP, 2019.

Morales Ordinola, Alan German. 2019. *Análisis y diseño de un sistema de gestión de historias clínicas para pacientes del Centro de Salud Pachitea.* Universidad de Piura, Piura, Perú : 2019.

Muñoz Sánchez, Virgilio Alexander. 2020. *Diseño e implementación de un sistema web para la gestión de citas médicas en la Clínica FEM SALUD S.A.C, 2020.* Universidad Peruana de la Américos, Lima, Perú : 2020.

Nordeen, Alex . 2020. *Learn SQL in 24 Hours.* Wisconsin : Guru99, 2020. p. 152. 978-0672335419.

Oñate Calderón, Wilmer Rolando and Aldás Flores, Clay Fernando. 2020. *Implementación de una aplicación web progresiva para la gestión de pruebas de simulación para el ingreso a instituciones militares y universidades en el centro de capacitación y nivelación académica SMARTEL.* Universidad Técnica de Ambato, Ambato, Ecuador : 2020.

Peralta Purizaca , Rensson Ruben. 2019. *Implementación de un sistema informático de registro y control de historias clínicas para reducir los tiempos de atención a los pacientes del hospital universitario de la universidad nacional de Piura.* Universidad Nacional de Piura, Piura : 2019.

Programa "Aulamatics Virtual" para fortalecer las competencias matemáticas genéricas en estudiantes de educación superior. **Ccanto-Curo, Ruth Melania and Acebedo Jara, Carlos Orlando. 2022.** Lousada : s.n., Julio 2022, p. 323. 16469895.

Quezada Dávila, Manuel Alejandro and Villarreal Agüero, Renzo Geraldo. 2018. *Para la obtención del Título Profesional de Ingeniero de Sistemas de Información.* Universidad Peruana de Ciencias Aplicadas, Lima : 2018.

Ramirez Flores, Nataly Maria. 2020. *Calidad de atención y cumplimiento del control del niño sano en madres de la Asociación Pro-vivienda San Hilarión, 2020.* Universidad Cesar Vallejo, Lima : 2020.

Red Hat. 2022. What are microservices? [Online] Mayo 11, 2022. [Cited: Julio 29, 2022.] https://www.redhat.com/en/topics/microservices/what-are-microservices.

Rendimiento de MariaDB y PostgreSQL. **Pilicita Garrido, Anabel, Borja López, Yolanda and Gutiérrez, Constante. 2020.** 2, Ecuador : UPSE, Diciembre 2020, Vol. 7.

Rodó, Paula. 2019. Distribución t de Student. [Online] Noviembre 5, 2019. [Cited: Agosto 6, 2022.] https://economipedia.com/definiciones/distribucion-t-de-student.html.

Rodriguéz, Daniela. 2020. Investigación Aplicada: Características, definición, ejemplos. [Online] Setiembre 17, 2020. [Cited: Agosto 6, 2022.] https://www.lifeder.com/investigacion-aplicada/.

Rus Arias, Enrique. 2020. Investigación Experimental. [Online] Diciembre 10, 2020. [Cited: Agosto 6, 2022.] https://economipedia.com/definiciones/investigacion-experimental.html.

Sabo, Mario. 2020. *NestJS.* Universidad Strossmayer de Osijek, Osijek, Croacia : 2020.

Sam Newman. 2019. *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith.* Londres : O'Reilly Media, Inc, 2019. p. 272. 1492047813, 9781492047810.

Schwaber, Ken and Sutherland, Jeff. 2020. *The Scrum Guide.* 2020.

Software Architecture and Software Design. **Jaiswal, Manishaben. 2019.** 303, Cumberland : IRJET, 2019, Vol. 6. 2395-0056.

Torres Arias, Nilton Cesar. 2020. *Sistema web para automatizar el proceso de control integral en niños de 0 a 5 años en el C.S Unidad Vecinal N°3.* Universidad César Vallejo, Lima, Perú : 2020.

Towards PWA in Healthcare. **Rêgo, Felipe, Portela, Filipe and Santos, Manuel Filipe. 2021.** s.l. : Procedia Computer Science, 2021, Vol. 160. 1877-0509.

Usabilidad de framework web: identificación de problemas y propuesta de evaluación. **Casas, Sandra I and Constanzo, Marcela A. 2018.** 2018. 978-950-658-472-6.

Use of e-Health as an Accessibility and Management Strategy Within Health Centers in Ecuador Through the Implementation of a Progressive Web Application as a Tool for Technological Development and Innovation. **Rivera, Joel, et al. 2020.** Cham : Springer International Publishing, 2020. 978-3-030-63329-5.

ANEXOS

Anexo 1: Matriz de Operacionalización de Variables

VARIABLE DE ESTUDIO	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL	DIMENSIÓNES	INDICADORES	ESCALA DE MEDICIÓN
Aplicación web progresiva utilizando la arquitectura de monolitos modulares	Según (Aguirre, y otros, 2019) es aquella que usa las últimas tecnologías accesibles en los navegadores permitiendo que pueda agregar características que eran exclusivas de las aplicaciones nativas. En cuanto a los monolitos modulares, (Smith, 2021) menciona que es un enfoque que crea e implementa una sola aplicación (que es la parte "monolítica"), pero crea la aplicación de modo que el código para cada característica requerida por la aplicación se divida en módulos separados. Este enfoque reduce las dependencias de los módulos para que pueda actualizar/modificar módulos sin afectar a otros.	La Aplicación Web Progresiva que uso la arquitectura de monolitos modulares será estudiada mediante las dimensiones siguientes: Estructura de base de datos y arquitectura del sistema. Las cuáles serán trabajadas con la técnica de la entrevista.	**Estructura de base de datos**		
			Arquitectura del sistema	Modularidad	
				Escalabilidad	
				Respuesta	
Proceso de gestión de pacientes infantes	Según el artículo de la revista (Eficiencia y sostenibilidad en la gestión clínica en el Perú en tiempos de pandemia, 2020) es una estrategia que brinda la posibilidad de ordenar y lograr sistematizar los procesos que intervienen en la atención sanitaria de una manera eficiente e ideal con la participación de profesionales en dicha gestión para tomar decisiones en base a los pacientes.	El proceso de gestión de pacientes infantes se estudiará basándose en las dimensiones siguientes: Control de pacientes infantes y Reportes de pacientes infantes. Las cuáles serán trabajadas con la técnica de la encuesta.	**Perspectiva de la enfermera**	Nivel del manejo de la información.	Ordinal
				Tiempo de reportes de pacientes.	
				Respaldo de datos.	
			Perspectiva del paciente	Calidad de atención	
				Tiempo de búsqueda de historia clínica	
				Tiempo de atención	
				Nivel de confianza	
				Seguimiento del paciente	

Anexo 2: Instrumentos de recolección de datos

Cuestionario a Pacientes

Cuestionario							
Nombres de la investigación		Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023					
Investigadores		García Ortiz Angel Eduardo					
		García Távara Edwin Fidel					
Tipo de prueba		Pre y Post test					
Empresa investigada		El centro de salud					
Fecha de inicio		-	Fecha de fin		-		
Variable		Gestión de pacientes infantes					
Dimensión		Perspectiva del paciente					
Escala de medición		Ordinal					
Instrucciones		Estimado paciente, a continuación, encontrará las interrogantes relacionadas a la gestión de pacientes infantes del centro de salud de Tumbes. Lea cuidadosamente cada una de ellas y responda marcando con un aspa (X) una de las alternativas presentadas, de acuerdo a su punto de vista					
N°	Indicador	Pregunta	PÉSIMO	MALO	REGULAR	BUENO	MUY BUENO
1	Calidad de atención	¿Cómo califica usted, la atención brindada al acudir al centro de salud para realizar las atenciones CRED de su hijo?					
2		¿Cómo califica usted, la atención brindada por el centro de salud al acudir a los controles de vacunas de su hijo?					
3	Tiempo de búsqueda de historia clínica	¿Cómo califica usted, el tiempo demandado para ubicar la historia clínica de su hijo?					
4	Tiempo de atención	¿Cómo califica usted, el tiempo demandado en las atenciones realizadas?					
5	Nivel de confianza	¿Cómo califica usted, el nivel de confianza que brinda el centro de salud?					
6	Seguimiento del paciente	¿Cómo califica usted, el seguimiento que se realiza a los pacientes infantes para sus controles de vacuna y atenciones CRED?					

Cuestionario a Enfermeras

Cuestionario							
Nombres de la investigación		Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023					
Investigadores		García Ortiz Angel Eduardo					
		García Távara Edwin Fidel					
Tipo de prueba		Pre y Post test					
Empresa investigada		El centro de salud					
Fecha de inicio		-	Fecha de fin	-			
Variable		Gestión de pacientes infantes					
Dimensión		Perspectiva de la enfermera					
Escala de medición		Ordinal					
Instrucciones		Estimado colaborador, a continuación, encontrará las interrogantes relacionadas a la gestión de pacientes infantes del centro de salud de Tumbes. Lea cuidadosamente cada una de ellas y responda marcando con un aspa (X) una de las alternativas presentadas, de acuerdo a su punto de vista					
N°	Indicador	Pregunta	PÉSIMO	MALO	REGULAR	BUENO	MUY BUENO
1	Nivel del manejo de información	¿Cómo califica usted, el manejo de información al gestionar las historias clínicas de los pacientes infantes?					
2		¿Cómo califica usted, el manejo de información al generar reportes de pacientes infantes pendientes de vacunar?					
3	Tiempo de reportes de pacientes	¿Cómo califica usted, el tiempo demandado para generar reportes de pacientes infantes pendientes de vacunar?					
4		¿Cómo califica usted, el tiempo demandado para generar reportes de pacientes infantes pendientes de recibir atenciones CRED?					
5	Respaldo de datos	¿Cómo califica usted, el respaldo que se tiene ante la perdida de datos de los pacientes infantes?					

Entrevista

Relevamiento de información

“Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023”

1. **Datos del entrevistado**
    - **Nombres y Apellidos**:
    - **Profesión**:
    - **Centro de estudios**:
    - **Especialización**:
    - **País**:

2. **Relevamiento de información:**
    - ¿Cómo percibe la modularidad del sistema informático elaborado?

      _____

      _____

      _____

      _____

      _____

      _____

      _____

      _____

      _____

    - ¿Cómo percibe la escalabilidad del sistema informático elaborado?

      _____

      _____

      _____

      _____

      _____

      _____

_____
_____
_____

- ¿Cómo percibe la capacidad de respuesta del sistema informático elaborado?

_____
_____
_____
_____
_____
_____
_____
_____
_____

- ¿Cómo percibe la estructura de la base de datos del sistema informático elaborado?

_____
_____
_____
_____
_____
_____
_____
_____
_____

Anexo 3: Constancias de Validación

UNIVERSIDAD CÉSAR VALLEJO

CONSTANCIA DE VALIDACIÓN

Yo, **José Irenio Abarca Ramírez**, con DNI N° **40242174,** de profesión **Ingeniero de Sistemas.**

Por medio de la presente hago constar que he revisado con fines de Validación el **Cuestionario para las enfermeras del centro de salud**, para la investigación titulada, **Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023**, elaborada por los estudiantes García Ortiz, Ángel Eduardo y García Távara, Edwin Fidel.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Cuestionario para las enfermeras del centro de salud	DEFICIENTE	ACEPTABLE	BUENO	MUY BUENO	EXCELENTE
1.Claridad				X	
2.Objetividad				X	
3.Actualidad				X	
4.Organización				X	
5.Suficiencia				X	
6.Intencionalidad					X
7.Consistencia					X
8.Coherencia					X
9.Metodología				X	

En señal de conformidad firmo la presente en la ciudad de Piura a los **18 días** del mes de Noviembre del Dos mil veintidós.

JOSE IRENIO ABARCA RAMIREZ
INGENIERO DE SISTEMAS
Reg. CIP N° 229298

Firma del Experto

UNIVERSIDAD CÉSAR VALLEJO

CONSTANCIA DE VALIDACIÓN

Yo, **José Irenio Abarca Ramírez,** con DNI Nº **40242174,** de profesión **Ingeniero de Sistemas**.

Por medio de la presente hago constar que he revisado con fines de Validación el **Cuestionario para los pacientes del centro de salud**, para la investigación titulada, **Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023**, elaborada por los estudiantes García Ortiz, Ángel Eduardo y García Távara, Edwin Fidel.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Cuestionario para los pacientes del centro de salud	DEFICIENTE	ACEPTABLE	BUENO	MUY BUENO	EXCELENTE
1.Claridad				X	
2.Objetividad				X	
3.Actualidad				X	
4.Organización				X	
5.Suficiencia				X	
6.Intencionalidad					X
7.Consistencia					X
8.Coherencia					X
9.Metodología				X	

En señal de conformidad firmo la presente en la ciudad de Piura a los ____ días del mes de Noviembre del Dos mil veintidós.

JOSE IRENIO ABARCA RAMIREZ
INGENIERO DE SISTEMAS
Reg. CIP N° 229298

Firma del Experto

UNIVERSIDAD CÉSAR VALLEJO

CONSTANCIA DE VALIDACIÓN

Yo, **José Irenio Abarca Ramírez,** con DNI Nº **40242174 de** profesión Ingeniero de sistemas.

Por medio de la presente hago constar que he revisado con fines de Validación la **Guía de Entrevista para los programadores especialistas**, para la investigación titulada, **Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023**, elaborada por los estudiantes García Ortiz, Angel Eduardo y García Távara, Edwin Fidel.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Entrevista para los programadores especialistas	DEFICIENTE	ACEPTABLE	BUENO	MUY BUENO	EXCELENTE
1.Claridad				X	
2.Objetividad				X	
3.Actualidad				X	
4.Organización				X	
5.Suficiencia				X	
6.Intencionalidad					X
7.Consistencia					X
8.Coherencia					X
9.Metodología				X	

En señal de conformidad firmo la presente en la ciudad de Piura a los **18 días** del mes de Noviembre del Dos mil veintidós.

JOSE IRENIO ABARCA RAMIREZ
INGENIERO DE SISTEMAS
Reg. CIP Nº 229299

Firma del Experto

UNIVERSIDAD CÉSAR VALLEJO

CONSTANCIA DE VALIDACIÓN

Yo, Eduardo R. Pérez Zamora, con DNI Nº **17639065** de profesión Ingeniero en Computación e Informática y el grado de Maestro.

Por medio de la presente hago constar que he revisado con fines de Validación el **Cuestionario para los pacientes del centro de salud**, para la investigación titulada, **Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023**, elaborada por los estudiantes García Ortiz, Ángel Eduardo y García Távara, Edwin Fidel.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Cuestionario para los pacientes del centro de salud	DEFICIENTE	ACEPTABLE	BUENO	MUY BUENO	EXCELENTE
1.Claridad				X	
2.Objetividad					X
3.Actualidad					X
4.Organización					X
5.Suficiencia					X
6.Intencionalidad					X
7.Consistencia					X
8.Coherencia					X
9.Metodología					X

En señal de conformidad firmo la presente en la ciudad de Piura a los 23 días del mes de Noviembre del Dos mil veintidós.

EDUARDO RAUL PEREZ ZAMORA
INGENIERO EN COMPUTACIÓN E INFORMATICA
Reg. CIP Nº 212391

UNIVERSIDAD CÉSAR VALLEJO

CONSTANCIA DE VALIDACIÓN

Yo, Eduardo R. Pérez Zamora, con DNI Nº **17639065** de profesión Ingeniero en Computación e Informática y el grado de Maestro.

Por medio de la presente hago constar que he revisado con fines de Validación la **Guía de Entrevista para los programadores especialistas**, para la investigación titulada, **Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023**, elaborada por los estudiantes García Ortiz, Angel Eduardo y García Távara, Edwin Fidel.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Entrevista para los programadores especialistas	DEFICIENTE	ACEPTABLE	BUENO	MUY BUENO	EXCELENTE
1.Claridad				X	
2.Objetividad					X
3.Actualidad					X
4.Organización					X
5.Suficiencia					X
6.Intencionalidad					X
7.Consistencia					X
8.Coherencia					X
9.Metodología					X

En señal de conformidad firmo la presente en la ciudad de Piura a los ____ días del mes de Noviembre del Dos mil veintidós.

EDUARDO RAUL PEREZ ZAMORA
INGENIERO EN COMPUTACIÓN E INFORMATICA
Reg. CIP N° 212391

UNIVERSIDAD CÉSAR VALLEJO

CONSTANCIA DE VALIDACIÓN

Yo, Eduardo R. Pérez Zamora, con DNI Nº **17639065** de profesión Ingeniero en Computación e Informática y el grado de Maestro.

Por medio de la presente hago constar que he revisado con fines de Validación el **Cuestionario para las enfermeras del centro de salud**, para la investigación titulada, **Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023**, elaborada por los estudiantes García Ortiz, Ángel Eduardo y García Távara, Edwin Fidel.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Cuestionario para las enfermeras del centro de salud	**DEFICIENTE**	**ACEPTABLE**	**BUENO**	**MUY BUENO**	**EXCELENTE**
1.Claridad				X	
2.Objetividad					X
3.Actualidad					X
4.Organización					X
5.Suficiencia					X
6.Intencionalidad					X
7.Consistencia					X
8.Coherencia					X
9.Metodología					X

En señal de conformidad firmo la presente en la ciudad de Piura a los ____ días del mes de Noviembre del Dos mil veintidós.

EDUARDO RAUL PEREZ ZAMORA
INGENIERO EN COMPUTACIÓN E INFORMATICA
Reg. CIP Nº 212391

UNIVERSIDAD CÉSAR VALLEJO

CONSTANCIA DE VALIDACIÓN

Yo, JUAN CARLOS RUFINO SERNAQUE, con DNI N.º 72633860 **de** profesión Ingeniero Industrial y de sistemas.

Por medio de la presente hago constar que he revisado con fines de Validación la **Guía de Entrevista para los programadores especialistas**, para la investigación titulada, **Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023**, elaborada por los estudiantes García Ortiz, Angel Eduardo y García Távara, Edwin Fidel.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Entrevista para los programadores especialistas	DEFICIENTE	ACEPTABLE	BUENO	MUY BUENO	EXCELENTE
1.Claridad				X	
2.Objetividad				X	
3.Actualidad					X
4.Organización					X
5.Suficiencia					X
6.Intencionalidad				X	
7.Consistencia				X	
8.Coherencia					X
9.Metodología					X

En señal de conformidad firmo la presente en la ciudad de Piura a los 28 días del mes de Noviembre del Dos mil veintidós.

JUAN CARLOS
RUFINO SERNAQUE
Ingeniero Industrial y
de Sistemas
CIP Nº 270985

Firma del Experto

UNIVERSIDAD CÉSAR VALLEJO

CONSTANCIA DE VALIDACIÓN

Yo, JUAN CARLOS RUFINO SERNAQUE, con DNI N.º **72633860** de profesión Ingeniero Industrial y de Sistemas.

Por medio de la presente hago constar que he revisado con fines de Validación el **Cuestionario para los pacientes del centro de salud**, para la investigación titulada, **Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023**, elaborada por los estudiantes García Ortiz, Ángel Eduardo y García Távara, Edwin Fidel.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Cuestionario para los pacientes del centro de salud	DEFICIENTE	ACEPTABLE	BUENO	MUY BUENO	EXCELENTE
1.Claridad				X	
2.Objetividad					X
3.Actualidad				X	
4.Organización					X
5.Suficiencia					X
6.Intencionalidad					X
7.Consistencia				X	
8.Coherencia					X
9.Metodología				X	

En señal de conformidad firmo la presente en la ciudad de Piura a los 28 días del mes de Noviembre del Dos mil veintidós.

JUAN CARLOS
RUFINO SERNAQUE
Ingeniero Industrial -
de Sistemas
CIP Nº 270985

Firma del Experto

UNIVERSIDAD CÉSAR VALLEJO

CONSTANCIA DE VALIDACIÓN

Yo, JUAN CARLOS RUFINO SERNAQUE, con DNI N.° **72633860** de profesión Ingeniero Industrial y de Sistemas.

Por medio de la presente hago constar que he revisado con fines de Validación el **Cuestionario para las enfermeras del centro de salud**, para la investigación titulada, **Aplicación web progresiva utilizando arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023**, elaborada por los estudiantes García Ortiz, Ángel Eduardo y García Távara, Edwin Fidel.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Cuestionario para las enfermeras del centro de salud	DEFICIENTE	ACEPTABLE	BUENO	MUY BUENO	EXCELENTE
1.Claridad			X		
2.Objetividad					X
3.Actualidad					X
4.Organización				X	
5.Suficiencia				X	
6.Intencionalidad					X
7.Consistencia					X
8.Coherencia					X
9.Metodología					X

En señal de conformidad firmo la presente en la ciudad de Piura a los 28 días del mes de Noviembre del Dos mil veintidós.

JUAN CARLOS
RUFINO SERNAQUE
Ingeniero Industrial
de Sistemas
CIP N° 270985

Firma del Experto

Anexo 4: Confiabilidad

	ITEMS						
ENCUESTADOS	1	2	3	4	5	6	SUMA
E1	5	5	4	5	5	4	28
E2	5	5	5	4	4	4	27
E3	5	5	5	4	5	5	29
E4	5	4	5	5	4	5	28
E5	5	5	5	5	4	4	28
E6	5	4	5	5	5	5	29
E7	5	5	5	5	5	5	30
E8	5	5	5	4	5	4	28
E9	5	5	5	5	5	5	30
E10	5	5	4	5	5	5	29
E11	4	4	5	5	5	5	28
E12	4	5	5	5	5	5	29
E13	5	5	5	5	5	5	30
E14	5	5	5	4	5	5	29
E15	5	5	4	5	5	5	29
E16	5	5	5	5	5	5	30
E17	5	4	5	5	5	5	29
E18	4	5	5	5	5	5	29
E19	5	5	5	5	5	5	30
E20	3	3	4	3	4	4	21
E21	4	4	5	3	4	4	24
E22	5	5	5	4	5	4	28
E23	3	3	4	3	4	5	22
E24	4	4	5	5	5	3	26
E25	3	3	4	3	4	3	20
E26	4	4	5	4	5	4	26
E27	4	4	4	3	4	5	24
E28	2	2	3	3	3	4	17
E29	4	4	4	3	4	3	22
E30	4	4	4	3	4	3	22
E31	4	5	4	5	4	5	27
E32	5	4	4	4	5	4	26
E33	5	4	4	5	4	5	27
E34	5	4	4	4	4	4	25
E35	5	5	5	5	5	5	30
E36	4	5	4	5	5	4	27
E37	5	5	5	4	5	4	28
E38	5	4	4	5	4	4	26
E39	4	4	4	3	4	3	22
E40	4	4	5	4	4	5	26
E41	3	4	5	4	4	5	25
E42	4	4	3	5	3	4	23
E43	4	4	5	3	4	4	24
E44	3	4	4	4	4	4	23
E45	4	5	5	5	4	4	27
E46	5	4	2	3	4	2	20
VARIANZA	0.581	0.488	0.467	0.671	0.335	0.568	
SUMATORIA DE VARIANZAS	3.109						
VARIANZA DE LA SUMA DE LOS ÍTEMS	10.269						

$$\alpha = \frac{K}{K-1}\left[1 - \frac{\sum S_i^2}{S_T^2}\right]$$

α:	Coeficiente de confiabilidad del cuestionario	0.84
k:	Número de ítems del instrumento	6
$\sum_{i=1}^{k} S_i^2$:	Sumatoria de las varianzas de los ítems.	3.109
S_t^2:	Varianza total del instrumento.	10.269

RANGO	CONFIABILIDAD	
0.53 a menos	Confiabilidad nula	
0.54 a 0.59	Confiabilidad baja	
0.60 a 0.65	Confiable	
0.66 a 0.71	Muy confiable	
0.72 a 0.99	Excelente confiabilidad	0.84 Nuestro instrumento es de excelente confiabilidad
1	Confiabilidad perfecta	

	ITEMS					
ENCUESTADOS	1	2	3	4	5	SUMA
E1	4	5	5	4	3	21
E2	4	4	5	5	4	22
E3	4	5	4	4	5	22
E4	4	4	4	3	3	18
E5	4	4	5	5	5	23
E6	1	1	1	3	2	8
E7	1	2	2	1	1	7
E8	1	2	3	2	3	11
E9	1	2	2	1	2	8
E10	2	2	1	3	2	10
E11	2	3	1	2	3	11
E12	2	3	2	3	2	12
E13	2	3	1	2	1	9
E14	2	3	2	2	3	12
E15	1	3	3	3	2	12
VARIANZA	1.556	1.262	2.196	1.449	1.396	
SUMATORIA DE VARIANZAS	7.858					
VARIANZA DE LA SUMA DE LOS ÍTEMS	30.996					

α:	Coeficiente de confiabilidad del cuestionario	→	**0.90**
k:	Número de ítems del instrumento	→	6
$\sum_{i=1}^{k} S_i^2$:	Sumatoria de las varianzas de los ítems.	→	7.858
S_t^2:	Varianza total del instrumento.	→	30.996

RANGO	CONFIABILIDAD	
0.53 a menos	Confiabilidad nula	
0.54 a 0.59	Confiabilidad baja	
0.60 a 0.65	Confiable	
0.66 a 0.71	Muy confiable	
0.72 a 0.99	Excelente confiabilidad	0.90 Nuestro instrumento es de excelente confiabilidad
1	Confiabilidad perfecta	

Anexo 5: Carta de aceptación

Nota: El nombre del establecimiento y las respectivas firmas de autorización están ocultas por razones de preservación del nombre de la empresa y confidencialidad.

TUMBES, 13 de abril del 2023

Señores:

UNIVERSIDAD CESAR VALLEJO

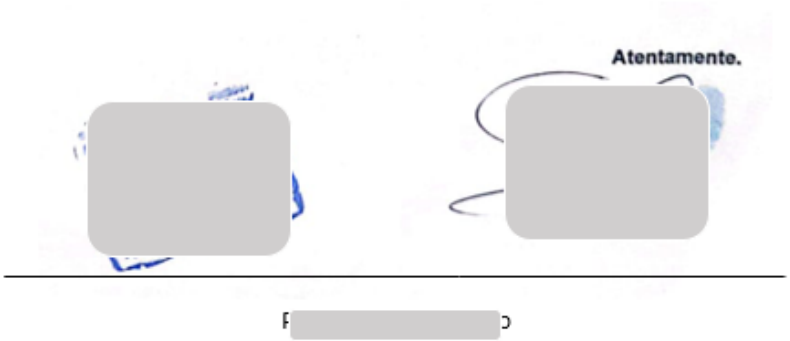
La que suscribe, [redacted], Gerente del **CLAS** [redacted] **– Tumbes**.

Autoriza:

Que los señores **ÁNGEL EDUARDO GARCÍA ORTIZ** y **EDWIN FIDEL GARCÍA TÁVARA**, recopilen los datos de la organización y apliquen los instrumentos necesarios para desarrollar su proyecto de investigación "Aplicación web progresiva con arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023" para la obtención de su título profesional.

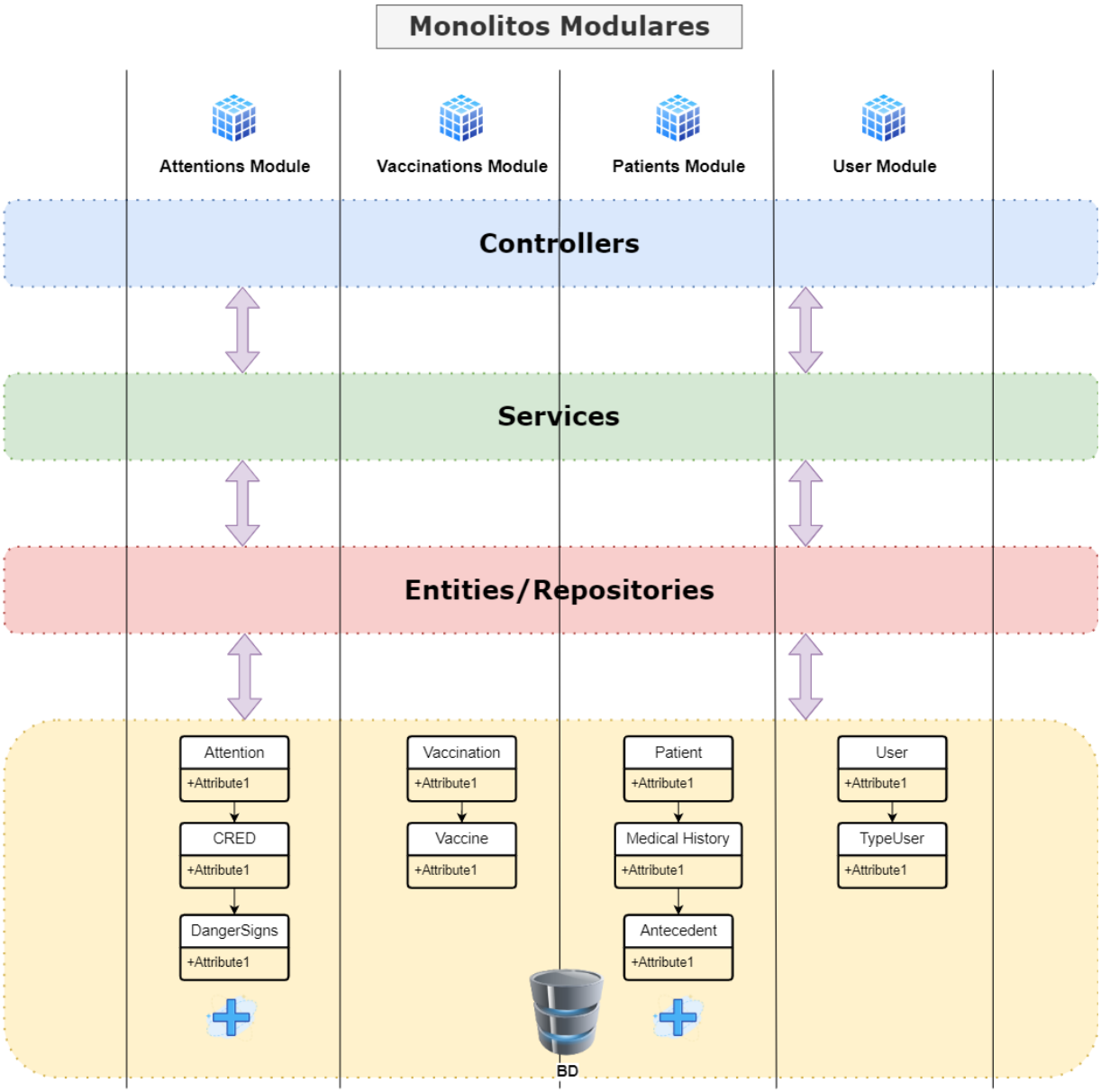
Sin otro asunto en particular, me despido.

Atentamente:



[redacted]

Anexo 6: Diagrama de Arquitectura



Anexo 7: Metodología de desarrollo de software

METODOLOGÍA SCRUM

Introducción

El siguiente documento presencia la implementación de la metodología de trabajo ágil Scrum, para el desarrollo de la **"APLICACIÓN WEB PROGRESIVA UTILIZANDO ARQUITECTURA MONOLÍTICA MODULAR PARA LA GESTIÓN DE PACIENTES INFANTES EN UN CENTRO DE SALUD, TUMBES 2023"**, debido a que proporciona una visión global del proyecto a desarrollar para de esta manera saber qué ocurre en el proyecto y cómo ocurre mediante entregas de gran medida funcionales y disponibles de forma iterativa y progresiva, en periodos de tiempo de 2 a 4 semanas.

Alcance

Al realizar un análisis de la problemática planteada en la investigación se determinan algunas de las características y capacidades que debe presentar el sistema para satisfacer los objetivos prioritarios.

- El sistema permitirá registrar pacientes niños recién nacidos.
- El sistema permite gestionar las atenciones de los niños
- El sistema permite gestionar las fechas de vacunación de los niños.
- El sistema permitirá gestionar las vacunaciones aplicadas a los niños.
- El sistema proporcionará reportes de los niños que son candidatos a recibir sus vacunas.

Valores

En cuanto a los valores que deben ser practicados por cada uno de los miembros involucrados para que la metodología pueda ser exitosa son:

- Puntualidad
- Compromiso

- Respeto
- Responsabilidad
- Enfoque
- Transparencia

Roles de la Metodología

Nombre de Roles del Proyecto

Rol	Nombre
Scrum Master	García Távara Edwin Fidel
Team Member	García Távara Edwin Fidel García Ortiz Ángel Eduardo
Product Owner	Gerente del centro de salud

Implicados del Proyecto

Comprometidos	Implicados
Scrum Master	Equipo de desarrollo
Team Member	
Product Owner	El centro de salud

Product Backlog

El propietario del producto precisa el Product Backlog, el cual detalla los requerimientos importantes como entregables, en este trabajo, se relaciona con el proceso de control de pacientes infantes.

Requerimientos funcionales

PRODUCT BACKLOG (LISTA DE PRODUCTO)			
Código	**Requerimiento Funcional**	**Días estimados**	**Prioridad**
RF1	El sistema permite el inicio de sesión de todos los tipos de usuarios que tendrán acceso al sistema, así mismo realiza la validación de acuerdo al rol asignado.	4	1
RF2	El sistema permite, el registro, modificación, consulta y eliminación de los usuarios con acceso al sistema.	4	2
RF3	El sistema permite, el registro, modificación, consulta y eliminación de pacientes infantes y sus respectivas historias clínicas.	5	1
RF4	El sistema permite, el registro, modificación, consulta y eliminación de atenciones realizadas a pacientes infantes.	5	1
RF5	El sistema permite, el registro, modificación, consulta y eliminación de las vacunas aplicadas a pacientes infantes.	4	1

RF6	El sistema permite listar a los pacientes infantes de recibir una determinada dosis y atenciones CRED pendientes.	3	1
RF7	El sistema permite generar reporte exportados a Excel de la lista de pacientes infantes pendientes de recibir una determinada dosis.	3	2
RF8	El sistema permite visualizar el estado del cronograma de controles CRED y vacunas.	4	2

Requerimientos No Funcionales

Código	Tipo	Requerimiento
RNF1	Usabilidad	El periodo de tiempo para que un determinado usuario aprenda a utilizar el sistema deberá ser corto.
RNF2	Fiabilidad	El sistema tiene que asegurar los datos que maneje, para de esta manera protegerlos del acceso no autorizado.
RNF3	Disponibilidad	El sistema debe estar disponible siempre en su totalidad para el personal de la empresa.
RNF4	Soporte	El Sistema debe tener la capacidad de un fácil análisis y de modificación para corregir posibles fallas.
RNF5	Seguridad	El acceso al sistema debe estar restringido, pudiendo acceder a él solamente a través de un usuario y contraseña, por lo tanto, sólo podrán ingresar las personas que se encuentren registradas. Además, a cada usuario se le asignara un rol para acceder a determinadas funciones.

Historias de usuario

Por consiguiente, se describen cada uno de los requerimientos proporcionados que el usuario precisa en el sistema, los cuales fueron presentados y discutidos con el Product Owner en previas reuniones.

Login

Historia de Usuario	
Número: 1	**Usuario:** Todos
Nombre de Historia: Login	
Prioridad de Negocio: Alta	**Riesgo de desarrollo:** Medio
Descripción: Esta funcionalidad permite el inicio de sesión de todos los tipos de usuarios que tendrán acceso al sistema.	
Condiciones y Restricciones: En el formulario de la interfaz de login el usuario debe ingresar de manera obligatoria su usuario y contraseña asignados, los cuales deben ser válidos, y en caso de que no, se mostraran los mensajes respectivos. Asimismo, el sistema realizará la validación de acuerdo al rol asignado al determinado usuario, para de esta manera mostrar las respectivas funcionalidades disponibles de acuerdo al rol.	

Gestionar Usuarios

Historia de Usuario	
Número: 2	**Usuario:** Administrador
Nombre de Historia: Gestionar usuarios	
Prioridad de Negocio: Medio	**Riesgo de desarrollo:** Bajo
Descripción: Esta funcionalidad permite gestionar a los usuarios que tendrán acceso al sistema, teniendo como funciones registrar, modificar, consultar y eliminar a los usuarios, en donde se tendrá en cuenta los campos de: usuario, nombres, contraseña, colegiatura, tipo de usuario y dni.	
Condiciones y Restricciones: Cuando se registre un nuevo usuario se deben llenar todos los campos de forma correcta, y de no ser así, se mostrarán los respectivos mensajes de validación. Se debe tener en cuenta que los campos como dni, usuario y colegiatura deben ser únicos e irrepetibles. Al momento de modificar todos los campos deben ser obligatorios y tener sus respectivas validaciones, pero no necesariamente todos deben ser modificados. Además, el campo de la contraseña aparecerá vacío, y se debe dejarse así si no se desea cambiar la contraseña, pero en caso de que si, entonces se deberá ingresar la nueva contraseña. Cuando se quiera consultar o buscar datos, solo se podrá hacer mediante el dni, colegiatura, usuario y nombres. Al eliminar un usuario, este debe ser borrado solo lógicamente de la base de datos.	

Gestionar Pacientes Infantes

Historia de Usuario	
Número: 3	**Usuario:** Todos
Nombre de Historia: Gestionar Pacientes Infantes	
Prioridad de Negocio: Alto	**Riesgo de desarrollo:** Medio
Descripción: Esta funcionalidad permite gestionar a los pacientes infantes que se agregaran al sistema, contando con las funciones de registrar, modificar, consultar y eliminar. Para este módulo se tendrá en cuenta los campos de los formatos dados por la organización.	
Condiciones y Restricciones: Cuando se registre a un nuevo paciente infante se deben llenar todos los campos de forma correcta, dentro de los cuales todos son requeridos, y en caso de que no se cumplan estas condiciones, se mostraran los respectivos mensajes de validación. Se debe tomar en cuenta, que el campo que identifique al paciente infante debe ser único e irrepetible. Se puede modificar uno o varios campos, pero teniendo en cuenta sus restricciones y validaciones. Cuando un usuario modifique un paciente infante, se deberá guardar que fue él quien lo elimino. Al querer buscar o consultar datos, se podrá hacer mediante los nombres del paciente infante, identificación, número de historia clínica y fecha de nacimiento. El sistema debe mostrar alertas de vacunas pendientes o atenciones CRED pendientes cuando se entre a una historia clínica de un paciente infante. Si se elimina por error, se podrá restaurar en otra pestaña, ya que la eliminación deber ser lógica en la base de datos.	

Gestionar Atenciones de Pacientes Infantes

Historia de Usuario	
Número: 4	**Usuario:** Todos
Nombre de Historia: Gestionar Atenciones de Pacientes Infantes	
Prioridad de Negocio: Alto	**Riesgo de desarrollo:** Medio
Descripción: Esta funcionalidad permite gestionar la atención a los pacientes infantes que se agregaran al sistema, contando con las funciones de registrar, modificar, consultar y eliminar. En este apartado se debe de tener en cuenta los formatos brindados por la organización para la atención de pacientes infantes.	
Condiciones y Restricciones: Cuando se registre una nueva atención a un paciente infante se deben llenar todos los campos de forma correcta y que se encuentren como requeridos, de otro modo, se mostraran los respectivos mensajes de validación. Se debe tomar en cuenta que, el valor ingresado en el campo número de historia clínica debe existir. Se puede modificar uno o varios campos, pero se debe tener en cuenta sus restricciones. Al querer buscar o consultar datos de la atención, se podrá hacer mediante la fecha de atención, historia clínica del paciente, identificación y nombres del paciente. Si se elimina por error, se podrá restaurar en otra pestaña, ya que la eliminación debe ser lógica en la base de datos	

Gestionar Vacunas Aplicadas en Edades de Pacientes Infantes

Historia de Usuario	
Número: 5	**Usuario:** Todos
Nombre de Historia: Gestionar Vacunas Aplicadas en Edades de Pacientes Infantes	
Prioridad de Negocio: Alto	**Riesgo de desarrollo:** Medio
Descripción: El sistema permitirá el registro, modificación, consulta y eliminación de las vacunas que son aplicadas en determinadas edades de pacientes infantes. Se debe tener en cuenta que los campos que contendrá los formatos dados por la organización.	
Condiciones y Restricciones: Cuando se registre una nueva vacunación se deben llenar todos los campos de forma correcta, dentro de los cuales todos son requeridos, y en caso de que no se cumplan estas condiciones, se mostraran los respectivos mensajes de validación. Se debe tener en cuenta que, los datos ingresados en los campos de vacuna y número de historia clínica deben existir. Se puede modificar uno o varios campos, pero teniendo en cuenta sus respectivas validaciones. Cuando un usuario modifique un registro de vacunación, se deberá guardar que fue él quien lo modifico. Al querer buscar o consultar datos, se podrá hacer mediante el código de paciente, id de vacuna, número de historia clínica, identificación y fecha de vacunación. Si se elimina por error, se podrá restaurar en otra pestaña, ya que la eliminación debe ser lógica en la base de datos. Cuando un usuario elimine un registro de vacunación, se deberá guardar que fue él quien lo eliminó.	

Listar Pacientes Infantes Pendientes en Recibir Dosis

Historia de Usuario	
Número: 6	**Usuario:** Todos
Nombre de Historia: Listar Pacientes Infantes Pendientes en Recibir Dosis y Atenciones CRED	
Prioridad de Negocio: Alta	**Riesgo de desarrollo:** Alto
Descripción: El sistema deberá permitir listar a los diferentes pacientes infantes que tengan pendientes dosis y atenciones CRED correspondientes a su edad actual. En el caso de vacunas se podrán listar por por fecha de las dosis correspondientes, en orden alfabético y por vacuna pendiente. Al querer buscar un paciente en específico se puede hacer por el nombre del paciente infante, con su identificación y con su número de historia.	
Condiciones y Restricciones: Los pacientes que se muestren serán los que aún están pendientes de recibir sus dosis, en caso el paciente este al día con sus dosis, este no aparecerá en la lista.	

Generar Reporte en Excel de Pacientes Infantes con Dosis Pendiente

Historia de Usuario	
Número: 7	**Usuario:** Todos
Nombre de Historia: Generar Reporte en Excel de Pacientes Infantes con Dosis Pendiente	
Prioridad de Negocio: Alta	**Riesgo de desarrollo:** Medio
Descripción: El sistema deberá permitir generar diferentes reportes como archivo Excel de los pacientes infantes que tengan dosis pendientes de las vacunas, mostrando los campos más fundamentales como pueden ser nombres y apellidos, identificación, contacto, dirección, nombres de los padres, dosis pendientes, entre otros.	
Condiciones y Restricciones: Los reportes mostraran los datos más importantes de los pacientes infantes, en caso un paciente infante no tenga dosis pendientes, este no aparecerá en el reporte generado en un archivo Excel. Se podrán establecer filtros como rango de fecha y vacunas a aplicar. Se podrá buscar por número de historia clínica, nombre de paciente, e identificación.	

Generar el cronograma de controles CRED

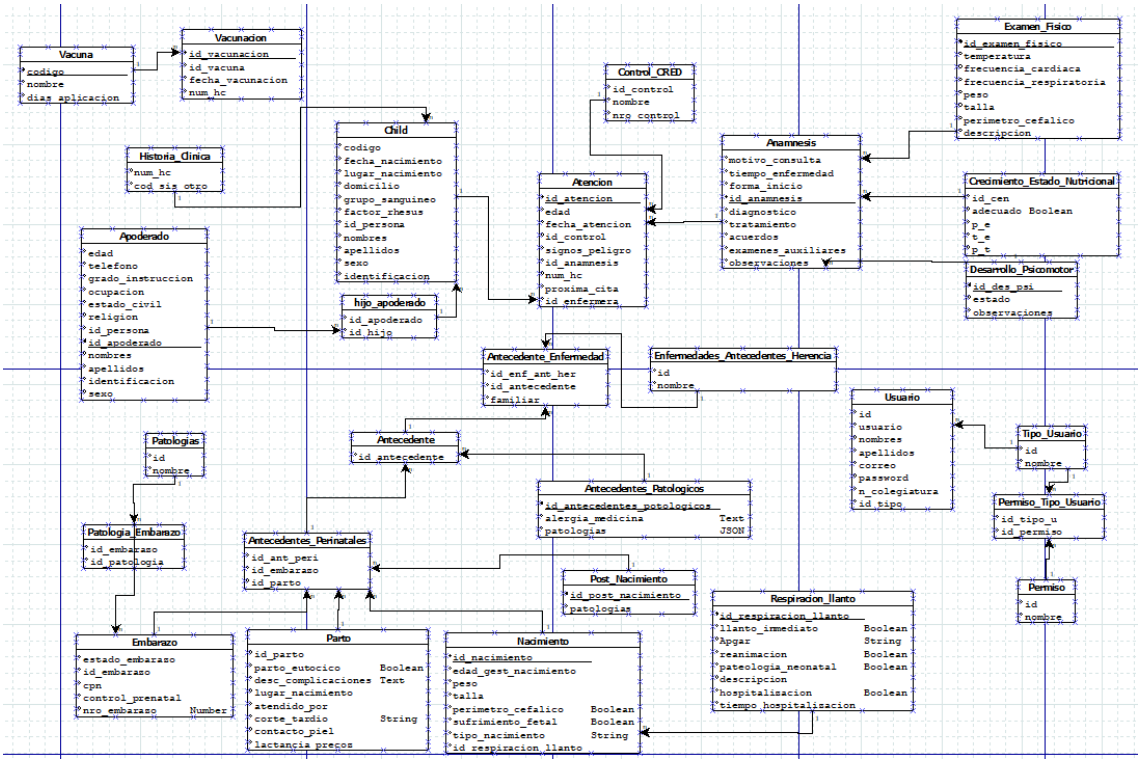
Historia de Usuario	
Número: 8	**Usuario:** Todos
Nombre de Historia: Generar el cronograma de controles CRED.	
Prioridad de Negocio: Media	**Riesgo de desarrollo:** Medio
Descripción: El sistema deberá permitir generar el cronograma de los controles CRED correspondientes del paciente infante.	
Condiciones y Restricciones: El cronograma se generará cuando el niño sea registrado por primera vez en el sistema, este debe contener las fechas en específico las cuales el niño debe acudir a su atención CRED.	

Sprint Backlog

N° Historia	Sprint	Descripción de tarea	Estado	Estimación (dias)
	Sprint 1: Diseño de base de datos	Diseño de base de datos: Identificación de entidades	Pendiente	2
		Diseño de base de datos: Diseño del modelo lógico	Pendiente	2
		Diseño de base de datos: Diseño del modelo físico	Pendiente	1
1 y 2	**Sprint 2:** Módulo de usuarios	Mockups	Pendiente	1
		Creación de entidades (Backend)	Pendiente	1
		Creación de servicios (Backend)	Pendiente	2
		Creación de controladores (Backend)	Pendiente	1
		Creación de DTOs y validaciones (Backend)	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	3
		Comunicación entre backend y frontend	Pendiente	1
		Sprint Review	Pendiente	1
5	**Sprint 3:** Módulo de vacunaciones	Mockups	Pendiente	1
		Creación de entidades (Backend)	Pendiente	1
		Creación de servicios (Backend)	Pendiente	2
		Creación de controladores (Backend)	Pendiente	1
		Creación de DTOs y validaciones (Backend)	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	3
		Comunicación entre backend y frontend	Pendiente	1
		Sprint Review	Pendiente	1
4	**Sprint 4:** Módulo de atenciones	Mockups	Pendiente	2
		Creación de entidades (Backend)	Pendiente	1
		Creación de servicios (Backend)	Pendiente	3
		Creación de controladores (Backend)	Pendiente	1
		Creación de DTOs y validaciones (Backend)	Pendiente	2
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	4
		Comunicación entre backend y frontend	Pendiente	1
		Sprint Review	Pendiente	1
3, 8	**Sprint 5:** Módulo de pacientes	Mockups	Pendiente	1
		Creación de entidades (Backend)	Pendiente	1
		Creación de servicios (Backend)	Pendiente	2
		Creación de controladores (Backend)	Pendiente	1
		Creación de DTOs y validaciones (Backend)	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	3
		Comunicación entre backend y frontend	Pendiente	1
		Sprint Review	Pendiente	1
6 y 7	**Sprint 6:** Módulo de reportes	Mockups	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	1
		Comunicación entre backend y frontend	Pendiente	1
	Sprint 7: Producción y capacitación	Llenado de la base de datos	Pendiente	1
		Mandar la aplicación a producción	Pendiente	3
		Capacitación de los usuarios finales	Pendiente	1

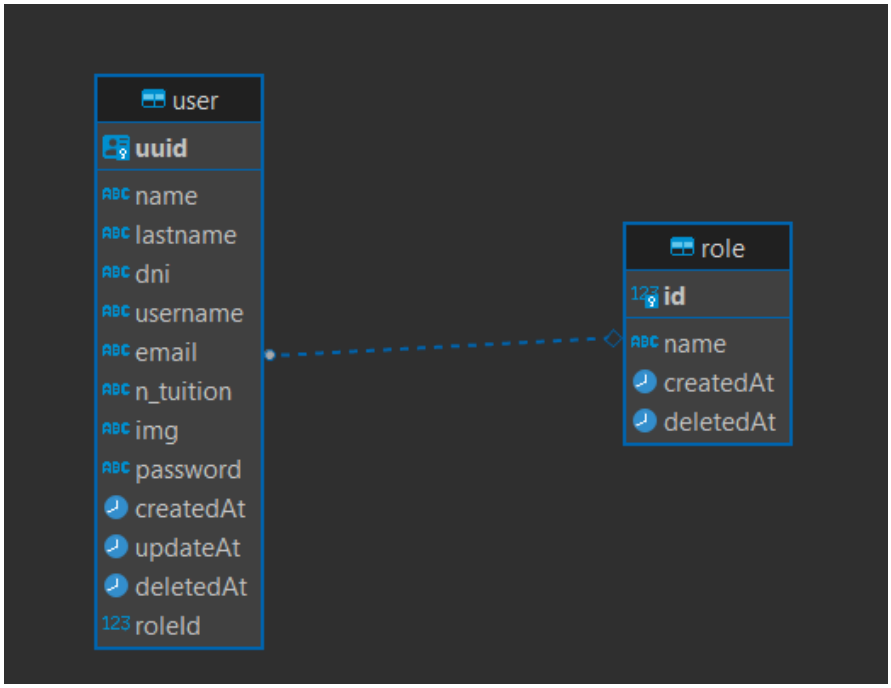
SPRINT 1: Diseño de base de datos

- **Diseño del modelo lógico**

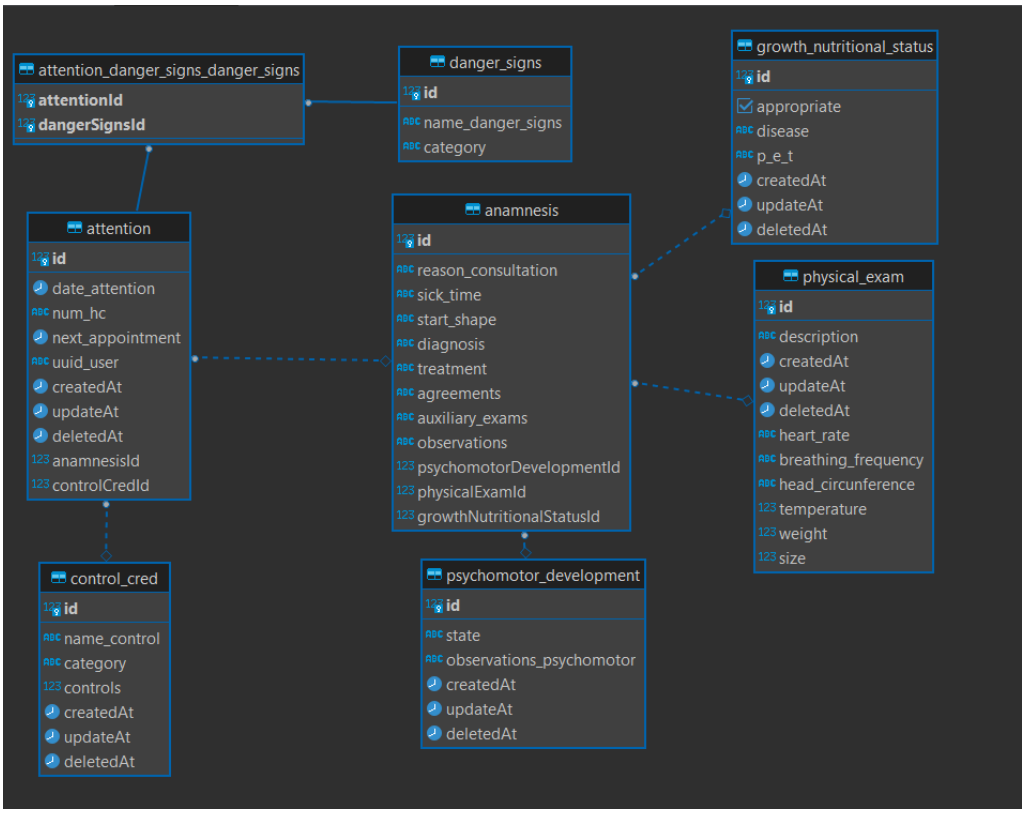


- **Diseño del modelo físico**

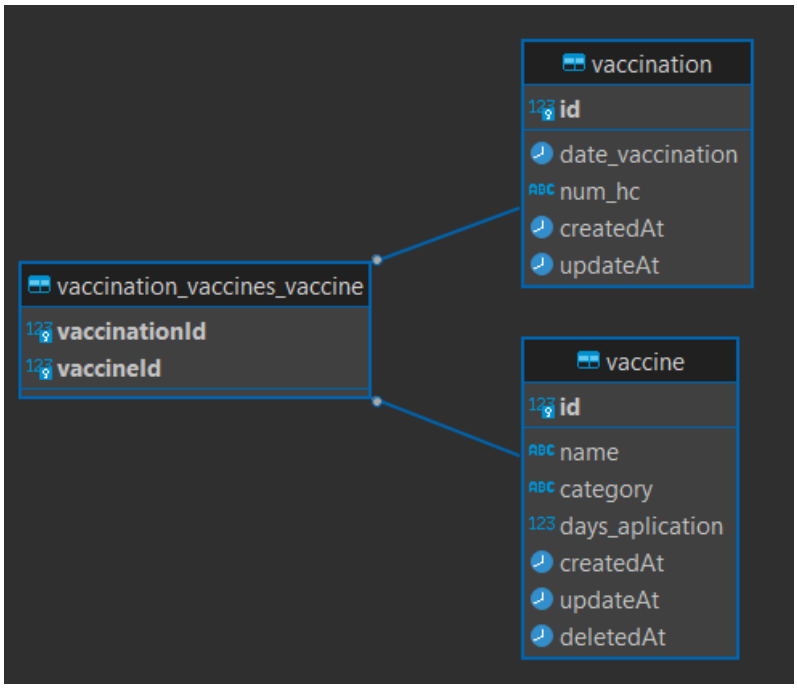
Módulo Usuarios



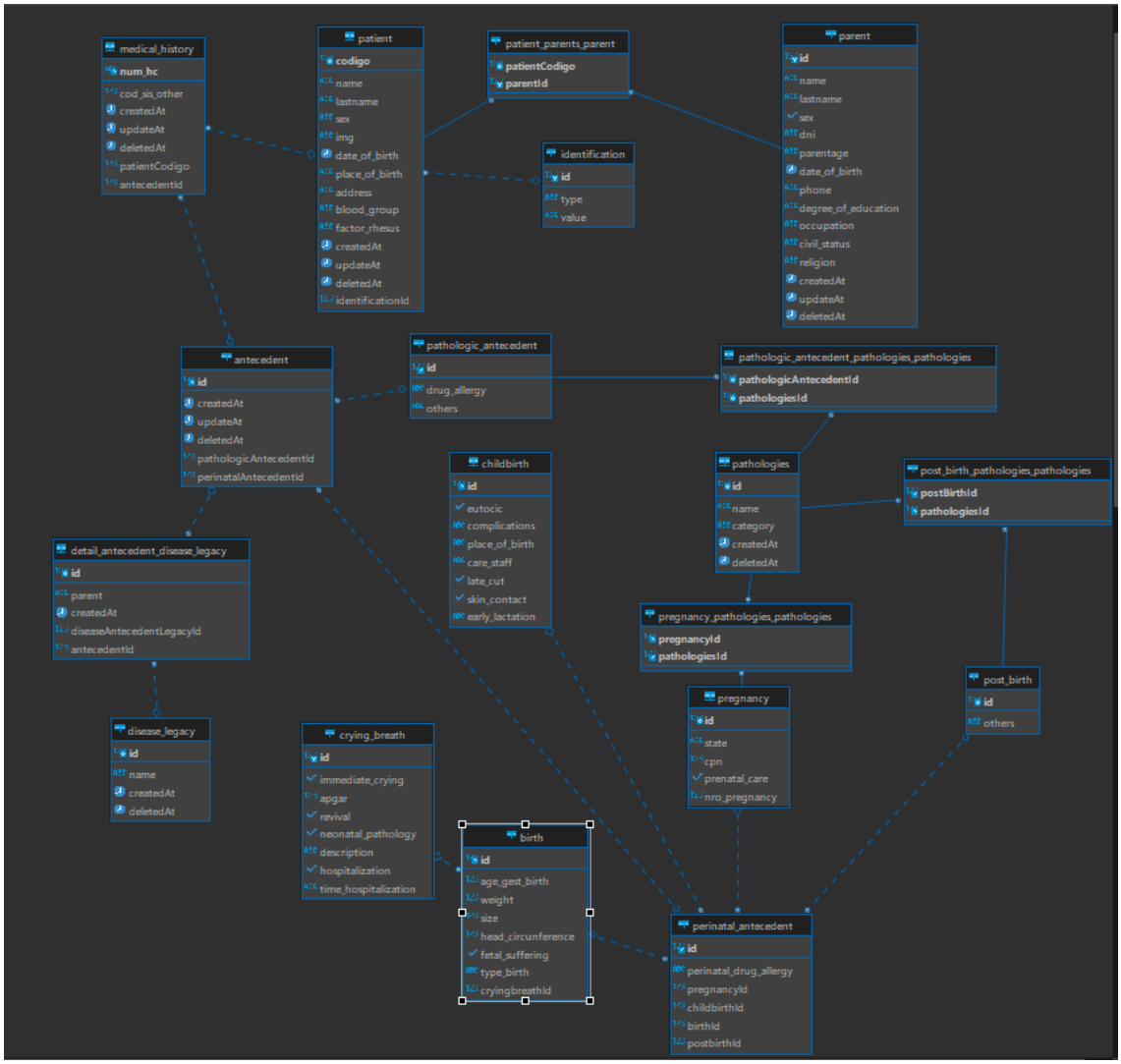
Módulo Atenciones



Módulo Vacunaciones

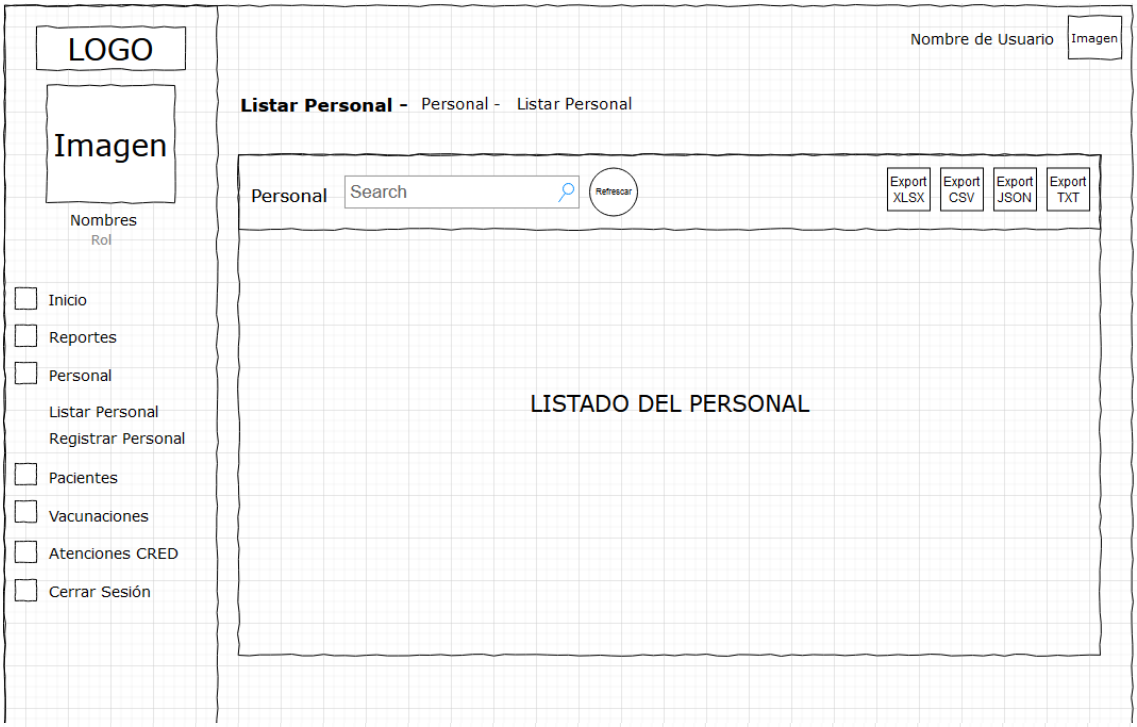


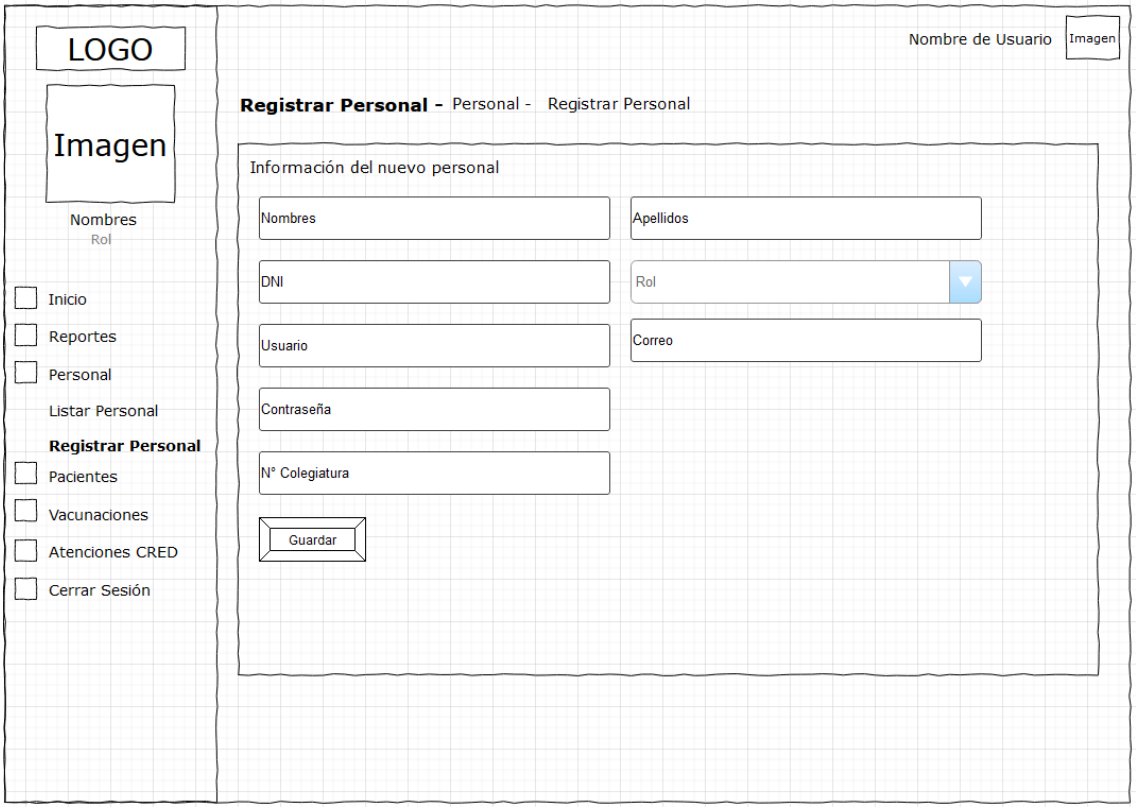
Módulo Pacientes



SPRINT 2: Módulo Usuarios

- **Mockups**





- **Creación de entidades**

Código fuente que muestra a las entidades de la base de datos pertenecientes al módulo de usuarios; las cuales son: user y role.

```typescript
import { Column, CreateDateColumn, DeleteDateColumn, Entity, OneToMany, PrimaryGeneratedColumn } from "typeorm";
import { User } from "./user.entity";

@Entity()
export class Role {
    @PrimaryGeneratedColumn()
    id: number;

    @Column({ nullable: false })
    name: string;//Nombre rol

    @CreateDateColumn()
    createdAt: Date;

    //SoftDelete
    @DeleteDateColumn()
    deletedAt: Date;

    @OneToMany(() => User, typeUserToUser => typeUserToUser.role)
    users: User[];
}
```

```typescript
import { Column, CreateDateColumn, DeleteDateColumn, Entity, ManyToOne, PrimaryGeneratedColumn, UpdateDateColumn } from "typeorm";
import { Role } from "./rol.entity";

@Entity()
export class User {
    @PrimaryGeneratedColumn("uuid")
    uuid: string;

    @Column('text', { nullable: false })
    name: string//nombres

    @Column('text', { nullable: false })
    lastname: string;//apellidos

    @Column('text', { nullable: true })
    dni: string;

    @Column('text', { nullable: false, unique: true })
    username: string;//nombre de usuario

    @Column('text', { nullable: false })
    email: string;//correo electronico

    @Column('text', { nullable: false })
    n_tuition: string;//N° de colegiatura

    @Column('text', { nullable: true, default: 'assets/images/user/admin.jpg' })
    img: string;

    @Column('text', { select: false })
    password: string;

    @CreateDateColumn()
    createdAt: Date;

    @UpdateDateColumn()
    updateAt: Date;

    //SoftDelete
    @DeleteDateColumn()
    deletedAt: Date;

    @ManyToOne(() => Role, (typeUser) => typeUser.users)
    role: Role;
}
```

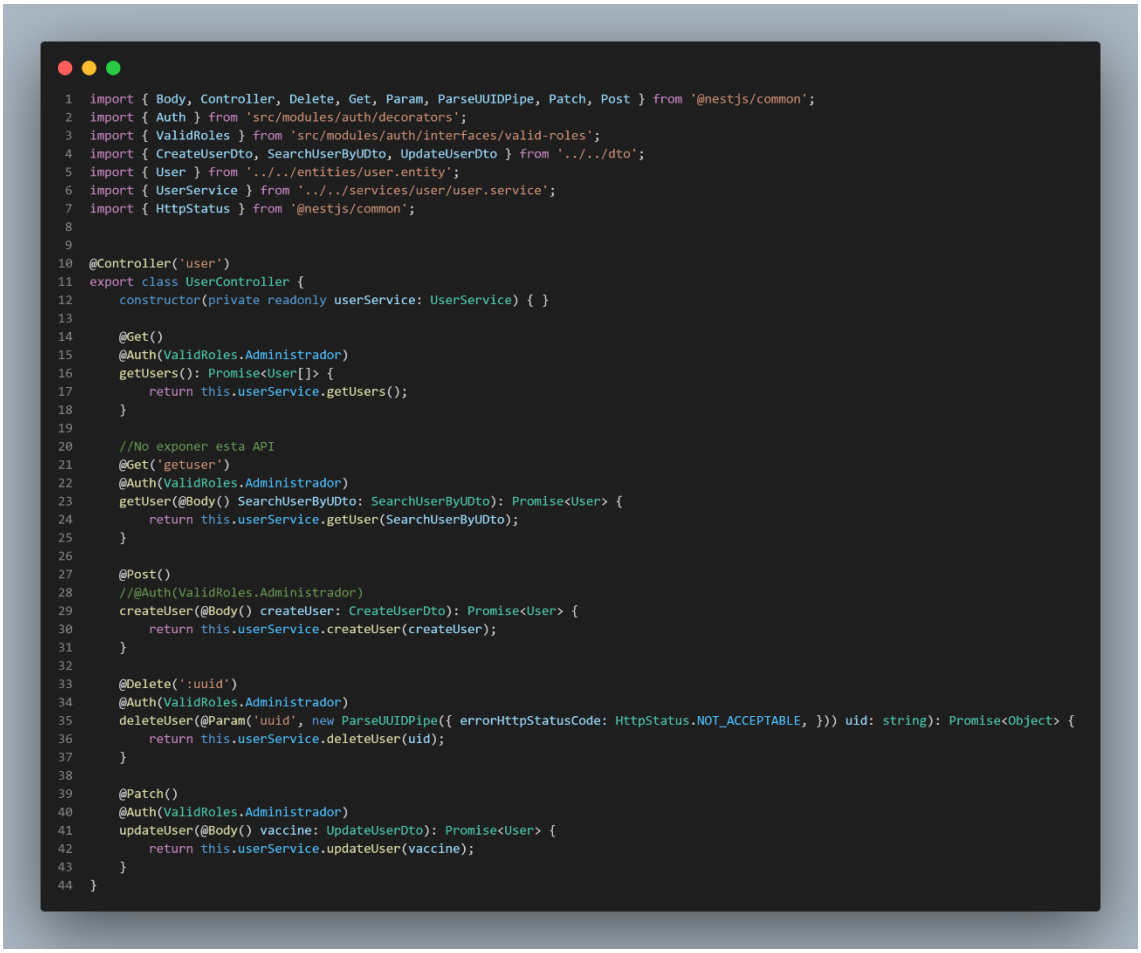
- **Creación de servicios**

Código fuente que muestra a los servicios utilizados para consultar a las entidades mencionadas anteriormente.

```
import { BadRequestException, HttpException, HttpStatus, Injectable, InternalServerErrorException, NotFoundException } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { CreateUserDto, SearchUserByIDDto, UpdateUserDto } from '../../dto';
import { Role } from '../../entities/rol.entity';
import { User } from '../../entities/user.entity';
import * as bcrypt from 'bcrypt'

@Injectable()
export class UserService {
    constructor(
        @InjectRepository(User) private readonly userRepository: Repository<User>,
        @InjectRepository(Role) private readonly roleRepository: Repository<Role>,

    ) { }

    async getUsers(): Promise<User[]> {
        const users = await this.userRepository.find({
            relations: {
                role: true,
            },
        });

        //Mensaje para cuando salga mal la consulta
        if (!users) throw new NotFoundException("Algo salió mal.");

        //Mensaje para cuando no se encuentren registros
        if (users && users.length == 0) throw new HttpException({
            status: HttpStatus.NOT_FOUND,
            error: 'No hay usuarios para mostrar.',
        }, HttpStatus.NOT_FOUND)

        return users;
    }

    async getUser({ username }: SearchUserByIDDto): Promise<User> {
        const user_response = await this.userRepository.findOne({ where: { username: username }, });
        //Mensaje para cuando salga mal la consulta
        if (!user_response) throw new NotFoundException("Algo salió mal.");

        //Mensaje para cuando no se encuentren registros
        if (user_response && user_response == null) throw new HttpException({
            status: HttpStatus.ACCEPTED,
            error: 'El usuario no existe.',
        }, HttpStatus.ACCEPTED)

        return user_response;
    }

    async createUser(createUserDto: CreateUserDto): Promise<User> {
        const { password, role, ...userData } = createUserDto;
        const frole = await this.roleRepository.findOne({ where: { id: role }, });

        const newUser = this.userRepository.create({ ...userData, role: frole, password: bcrypt.hashSync(password, 10) });

        return this.userRepository.save(newUser).catch((e) => {
            this.handleDBErrors(e)
        });
    }

    async deleteUser(uid: string): Promise<Object> {
        const deleteResponse = await this.userRepository.softDelete(uid);
        if (!deleteResponse.affected) {
            throw new NotFoundException('Usuario no encontrado.');
        }

        return {
            status: HttpStatus.ACCEPTED,
            message: 'Usuario eliminado exitosamente.',
        }
    }

    async updateUser(updateUserDto: UpdateUserDto) {
        const { password, id_role, ...userData } = updateUserDto;
        const role_user = await this.roleRepository.findOne({ where: { id: id_role }, });

        // const userToUpdate = await this.userRepository.findOne({
        //     where: { uuid: uuid }
        // });
        // if (!userToUpdate) {
        //     throw new HttpException({
        //         statusCode: HttpStatus.NOT_FOUND,
        //         message: 'El usuario a actualizar no existe.',
        //     }, HttpStatus.NOT_FOUND);
        // }

        let userUpdate: User;

        if (password != "") {
            if (password.length < 6) throw new HttpException({
                statusCode: HttpStatus.BAD_REQUEST,
                message: 'La contraseña debe contener como mínimo 6 caracteres.',
            }, HttpStatus.ACCEPTED)
            userUpdate = await this.userRepository.preload({
                ...userData, role: role_user, password: bcrypt.hashSync(password, 10)
            });
        } else {
            userUpdate = await this.userRepository.preload({
                ...userData, role: role_user
            });
        }

        if (!userUpdate) {
            throw new NotFoundException('Usuario no encontrado.')
        }

        await this.userRepository.save(userUpdate).catch((e) => {
            const errors = [];
            if (/(value_identification)[\s\S]*(already exists)/.test(e.detail)) errors.push('La identificación ingresada ya pertenece a un usuario.')

            if (/(username)[\s\S]*(already exists)/.test(e.detail)) errors.push('El usuario ingresado ya pertenece a un registro.')

            if (errors.length > 0) throw new BadRequestException(errors);

            return e;
        });

        return await this.userRepository.findOne({
            where: { uuid: userUpdate.uuid },
            relations: {
                role: true,
            },
        });

    }

    private handleDBErrors(error: any): never {
        if (error.code === '23505')
            throw new BadRequestException(error.detail);
        console.log(error)
        throw new InternalServerErrorException('Please check server logs');
    }

}
```

- **Creación de controladores**

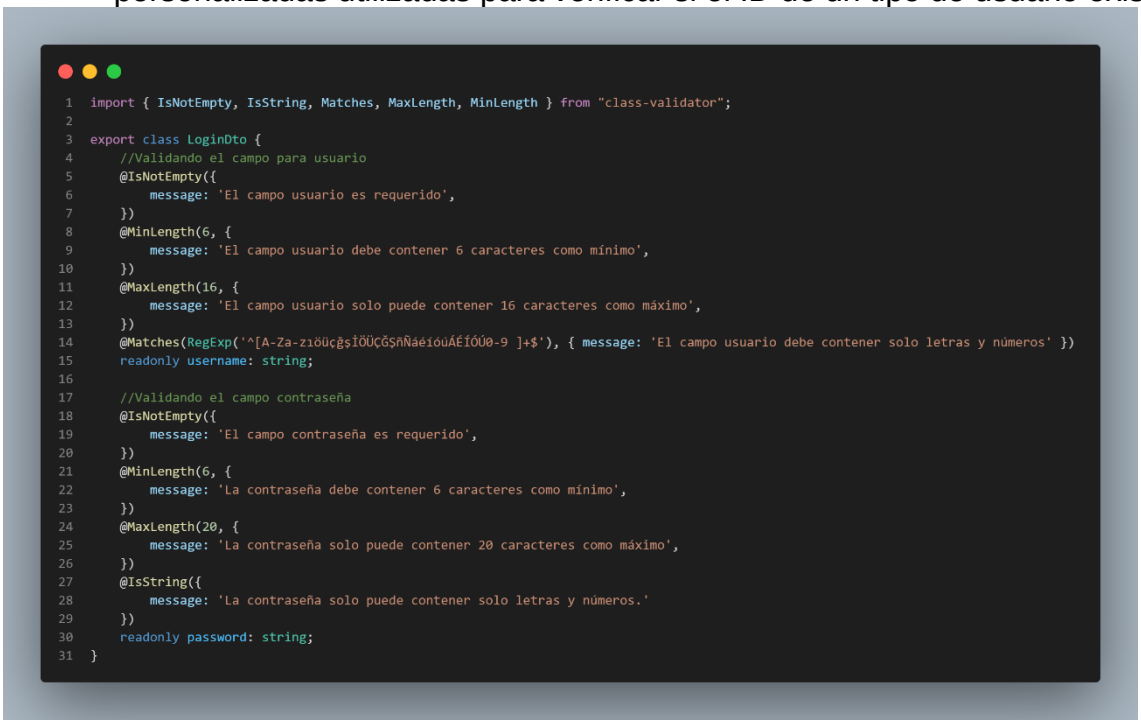
Código fuente que contiene las APIs a las cuales se debe consultar para llamar a un determinado servicio.



```typescript
import { Body, Controller, Delete, Get, Param, ParseUUIDPipe, Patch, Post } from '@nestjs/common';
import { Auth } from 'src/modules/auth/decorators';
import { ValidRoles } from 'src/modules/auth/interfaces/valid-roles';
import { CreateUserDto, SearchUserByUDto, UpdateUserDto } from '../../dto';
import { User } from '../../entities/user.entity';
import { UserService } from '../../services/user/user.service';
import { HttpStatus } from '@nestjs/common';


@Controller('user')
export class UserController {
    constructor(private readonly userService: UserService) { }

    @Get()
    @Auth(ValidRoles.Administrador)
    getUsers(): Promise<User[]> {
        return this.userService.getUsers();
    }

    //No exponer esta API
    @Get('getuser')
    @Auth(ValidRoles.Administrador)
    getUser(@Body() SearchUserByUDto: SearchUserByUDto): Promise<User> {
        return this.userService.getUser(SearchUserByUDto);
    }

    @Post()
    //@Auth(ValidRoles.Administrador)
    createUser(@Body() createUser: CreateUserDto): Promise<User> {
        return this.userService.createUser(createUser);
    }

    @Delete(':uuid')
    @Auth(ValidRoles.Administrador)
    deleteUser(@Param('uuid', new ParseUUIDPipe({ errorHttpStatusCode: HttpStatus.NOT_ACCEPTABLE, })) uid: string): Promise<Object> {
        return this.userService.deleteUser(uid);
    }

    @Patch()
    @Auth(ValidRoles.Administrador)
    updateUser(@Body() vaccine: UpdateUserDto): Promise<User> {
        return this.userService.updateUser(vaccine);
    }
}
```

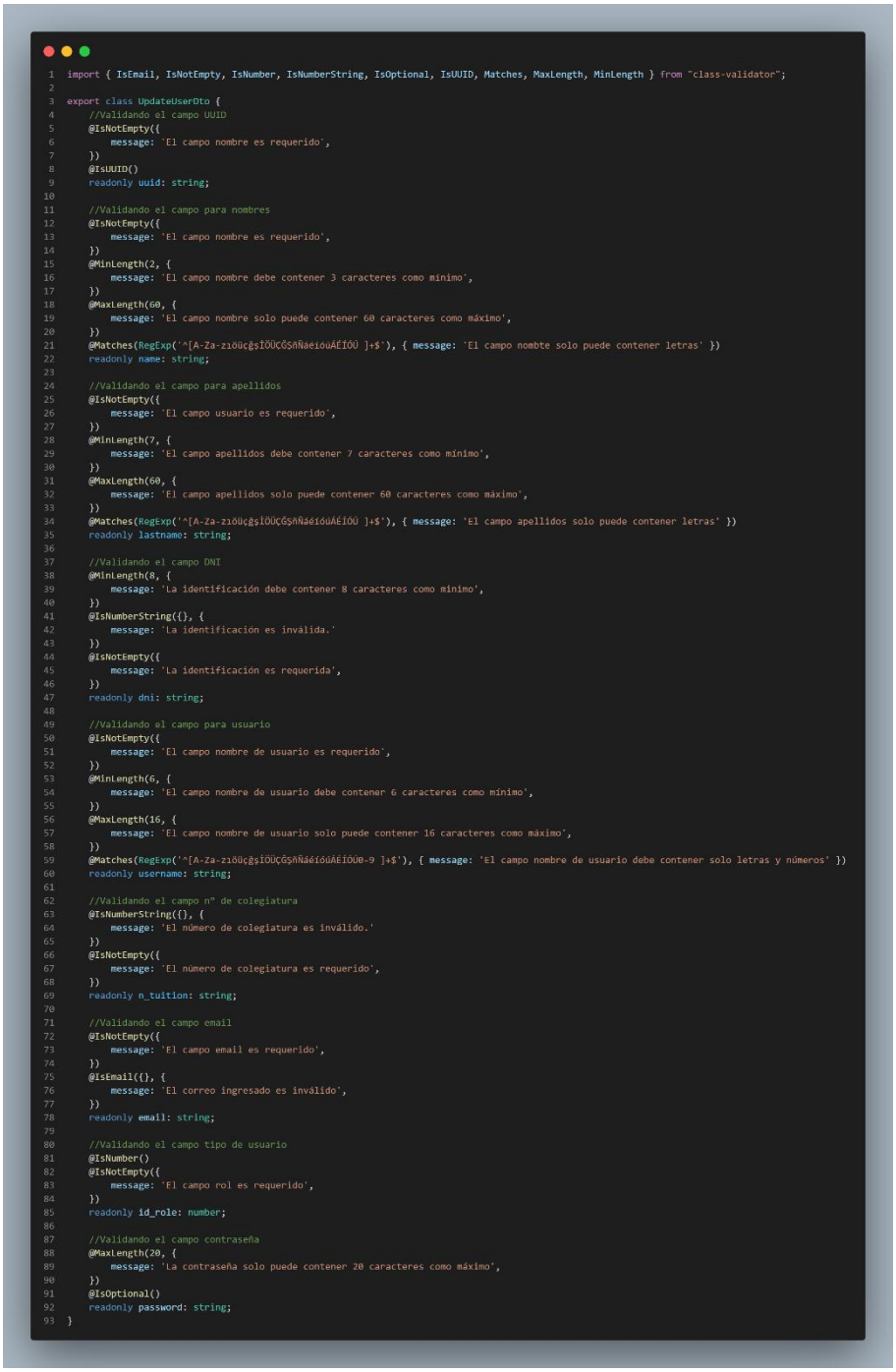
- **Creación de DTOs y validaciones**

Objeto de transferencia de datos que se utiliza al momento de enviar datos al backend para los diferentes casos de usos. Además, las validaciones personalizadas utilizadas para verificar si el ID de un tipo de usuario existe.

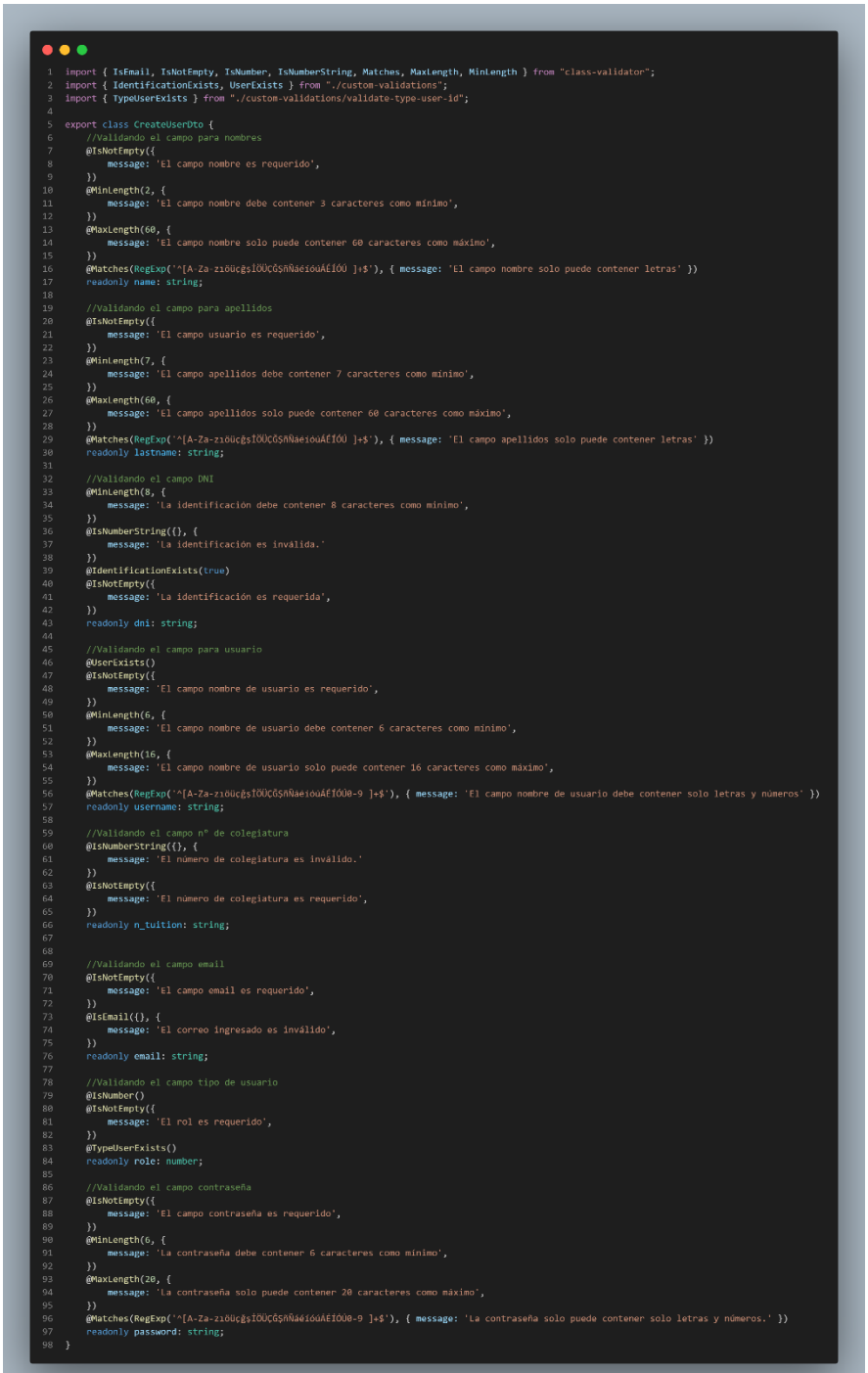


```dto
import { IsNotEmpty, IsString, Matches, MaxLength, MinLength } from "class-validator";

export class LoginDto {
    //Validando el campo para usuario
    @IsNotEmpty({
        message: 'El campo usuario es requerido',
    })
    @MinLength(6, {
        message: 'El campo usuario debe contener 6 caracteres como mínimo',
    })
    @MaxLength(16, {
        message: 'El campo usuario solo puede contener 16 caracteres como máximo',
    })
    @Matches(RegExp('^[A-Za-zıöüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ0-9 ]+$'), { message: 'El campo usuario debe contener solo letras y números' })
    readonly username: string;

    //Validando el campo contraseña
    @IsNotEmpty({
        message: 'El campo contraseña es requerido',
    })
    @MinLength(6, {
        message: 'La contraseña debe contener 6 caracteres como mínimo',
    })
    @MaxLength(20, {
        message: 'La contraseña solo puede contener 20 caracteres como máximo',
    })
    @IsString({
        message: 'La contraseña solo puede contener solo letras y números.'
    })
    readonly password: string;
}
```

```dto
import { IsNotEmpty, Matches, MaxLength, MinLength } from "class-validator";

export class SearchUserByUDto {
    //Validando el campo para usuario
    @IsNotEmpty({
        message: 'El campo usuario es requerido',
    })
    @MinLength(6, {
        message: 'El campo usuario debe contener 6 caracteres como mínimo',
    })
    @MaxLength(16, {
        message: 'El campo usuario solo puede contener 16 caracteres como máximo',
    })
    @Matches(RegExp('^[A-Za-zıöüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ0-9 ]+$'), { message: 'El campo usuario debe contener solo letras y números' })
    readonly username: string;
}

```



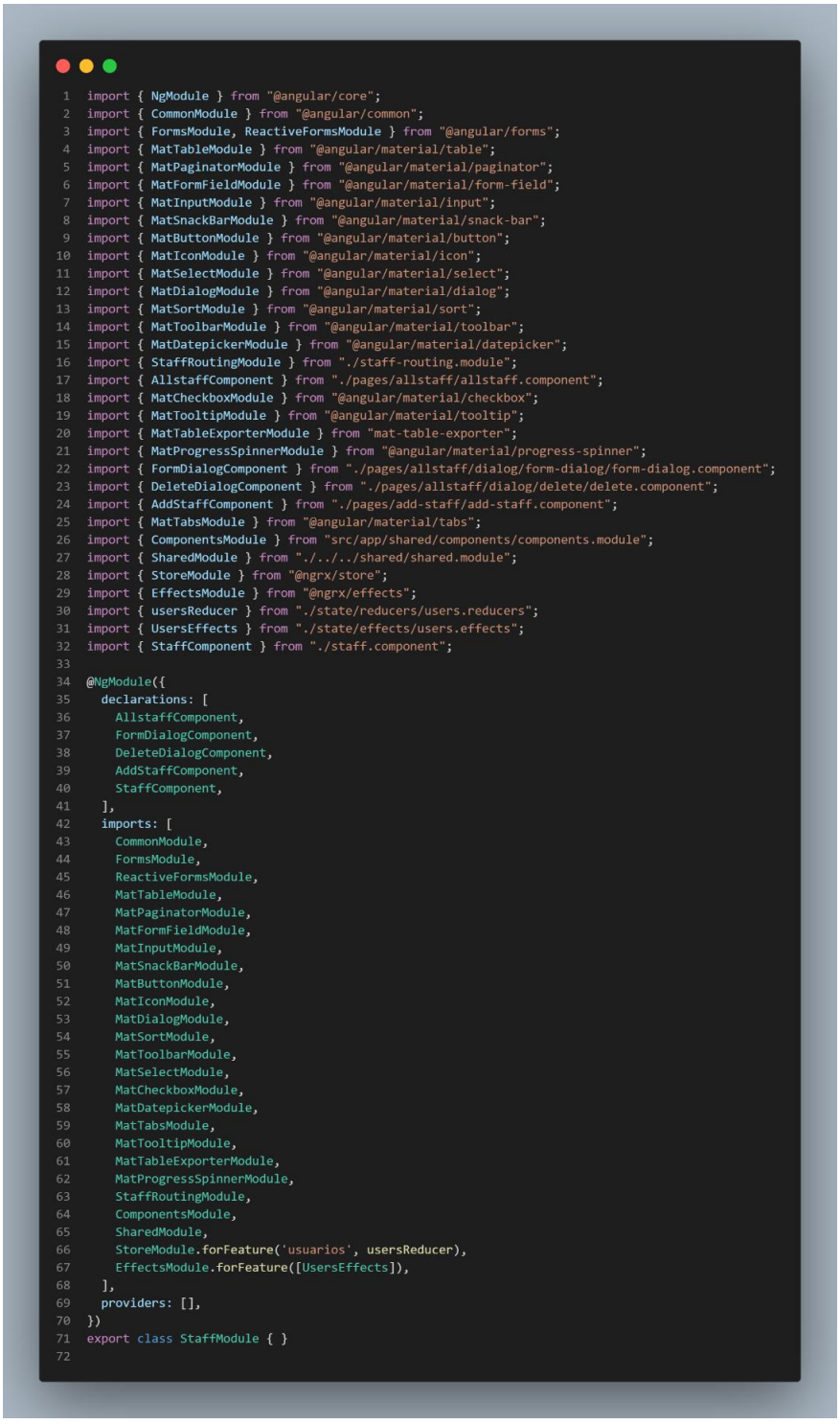
```ts
import { IsEmail, IsNotEmpty, IsNumber, IsNumberString, IsOptional, IsUUID, Matches, MaxLength, MinLength } from "class-validator";

export class UpdateUserDto {
    //Validando el campo UUID
    @IsNotEmpty({
        message: 'El campo nombre es requerido',
    })
    @IsUUID()
    readonly uuid: string;

    //Validando el campo para nombres
    @IsNotEmpty({
        message: 'El campo nombre es requerido',
    })
    @MinLength(2, {
        message: 'El campo nombre debe contener 3 caracteres como minimo',
    })
    @MaxLength(60, {
        message: 'El campo nombre solo puede contener 60 caracteres como máximo',
    })
    @Matches(RegExp('^[A-Za-z1öüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ ]+$'), { message: 'El campo nombte solo puede contener letras' })
    readonly name: string;

    //Validando el campo para apellidos
    @IsNotEmpty({
        message: 'El campo usuario es requerido',
    })
    @MinLength(7, {
        message: 'El campo apellidos debe contener 7 caracteres como mínimo',
    })
    @MaxLength(60, {
        message: 'El campo apellidos solo puede contener 60 caracteres como máximo',
    })
    @Matches(RegExp('^[A-Za-z1öüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ ]+$'), { message: 'El campo apellidos solo puede contener letras' })
    readonly lastname: string;

    //Validando el campo DNI
    @MinLength(8, {
        message: 'La identificación debe contener 8 caracteres como mínimo',
    })
    @IsNumberString({}, {
        message: 'La identificación es inválida.'
    })
    @IsNotEmpty({
        message: 'La identificación es requerida',
    })
    readonly dni: string;

    //Validando el campo para usuario
    @IsNotEmpty({
        message: 'El campo nombre de usuario es requerido',
    })
    @MinLength(6, {
        message: 'El campo nombre de usuario debe contener 6 caracteres como mínimo',
    })
    @MaxLength(16, {
        message: 'El campo nombre de usuario solo puede contener 16 caracteres como máximo',
    })
    @Matches(RegExp('^[A-Za-z1öüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ0-9 ]+$'), { message: 'El campo nombre de usuario debe contener solo letras y números' })
    readonly username: string;

    //Validando el campo n° de colegiatura
    @IsNumberString({}, {
        message: 'El número de colegiatura es inválido.'
    })
    @IsNotEmpty({
        message: 'El número de colegiatura es requerido',
    })
    readonly n_tuition: string;

    //Validando el campo email
    @IsNotEmpty({
        message: 'El campo email es requerido',
    })
    @IsEmail({}, {
        message: 'El correo ingresado es inválido',
    })
    readonly email: string;

    //Validando el campo tipo de usuario
    @IsNumber()
    @IsNotEmpty({
        message: 'El campo rol es requerido',
    })
    readonly id_role: number;

    //Validando el campo contraseña
    @MaxLength(20, {
        message: 'La contraseña solo puede contener 20 caracteres como máximo',
    })
    @IsOptional()
    readonly password: string;
}
```



```ts
import { IsEmail, IsNotEmpty, IsNumber, IsNumberString, Matches, MaxLength, MinLength } from "class-validator";
import { IdentificationExists, UserExists } from "./custom-validations";
import { TypeUserExists } from "./custom-validations/validate-type-user-id";

export class CreateUserDto {
    //Validando el campo para nombres
    @IsNotEmpty({
        message: 'El campo nombre es requerido',
    })
    @MinLength(2, {
        message: 'El campo nombre debe contener 3 caracteres como mínimo',
    })
    @MaxLength(60, {
        message: 'El campo nombre solo puede contener 60 caracteres como máximo',
    })
    @Matches(RegExp('^[A-Za-zıöüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ ]+$'), { message: 'El campo nombre solo puede contener letras' })
    readonly name: string;

    //Validando el campo para apellidos
    @IsNotEmpty({
        message: 'El campo usuario es requerido',
    })
    @MinLength(7, {
        message: 'El campo apellidos debe contener 7 caracteres como mínimo',
    })
    @MaxLength(60, {
        message: 'El campo apellidos solo puede contener 60 caracteres como máximo',
    })
    @Matches(RegExp('^[A-Za-zıöüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ ]+$'), { message: 'El campo apellidos solo puede contener letras' })
    readonly lastname: string;

    //Validando el campo DNI
    @MinLength(8, {
        message: 'La identificación debe contener 8 caracteres como mínimo',
    })
    @IsNumberString({}, {
        message: 'La identificación es inválida.'
    })
    @IdentificationExists(true)
    @IsNotEmpty({
        message: 'La identificación es requerida',
    })
    readonly dni: string;

    //Validando el campo para usuario
    @UserExists()
    @IsNotEmpty({
        message: 'El campo nombre de usuario es requerido',
    })
    @MinLength(6, {
        message: 'El campo nombre de usuario debe contener 6 caracteres como mínimo',
    })
    @MaxLength(16, {
        message: 'El campo nombre de usuario solo puede contener 16 caracteres como máximo',
    })
    @Matches(RegExp('^[A-Za-zıöüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ0-9 ]+$'), { message: 'El campo nombre de usuario debe contener solo letras y números' })
    readonly username: string;

    //Validando el campo nº de colegiatura
    @IsNumberString({}, {
        message: 'El número de colegiatura es inválido.'
    })
    @IsNotEmpty({
        message: 'El número de colegiatura es requerido',
    })
    readonly n_tuition: string;


    //Validando el campo email
    @IsNotEmpty({
        message: 'El campo email es requerido',
    })
    @IsEmail({}, {
        message: 'El correo ingresado es inválido',
    })
    readonly email: string;

    //Validando el campo tipo de usuario
    @IsNumber()
    @IsNotEmpty({
        message: 'El rol es requerido',
    })
    @TypeUserExists()
    readonly role: number;

    //Validando el campo contraseña
    @IsNotEmpty({
        message: 'El campo contraseña es requerido',
    })
    @MinLength(6, {
        message: 'La contraseña debe contener 6 caracteres como mínimo',
    })
    @MaxLength(20, {
        message: 'La contraseña solo puede contener 20 caracteres como máximo',
    })
    @Matches(RegExp('^[A-Za-zıöüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ0-9 ]+$'), { message: 'La contraseña solo puede contener solo letras y números.' })
    readonly password: string;
}
```

- **Creación de módulos**

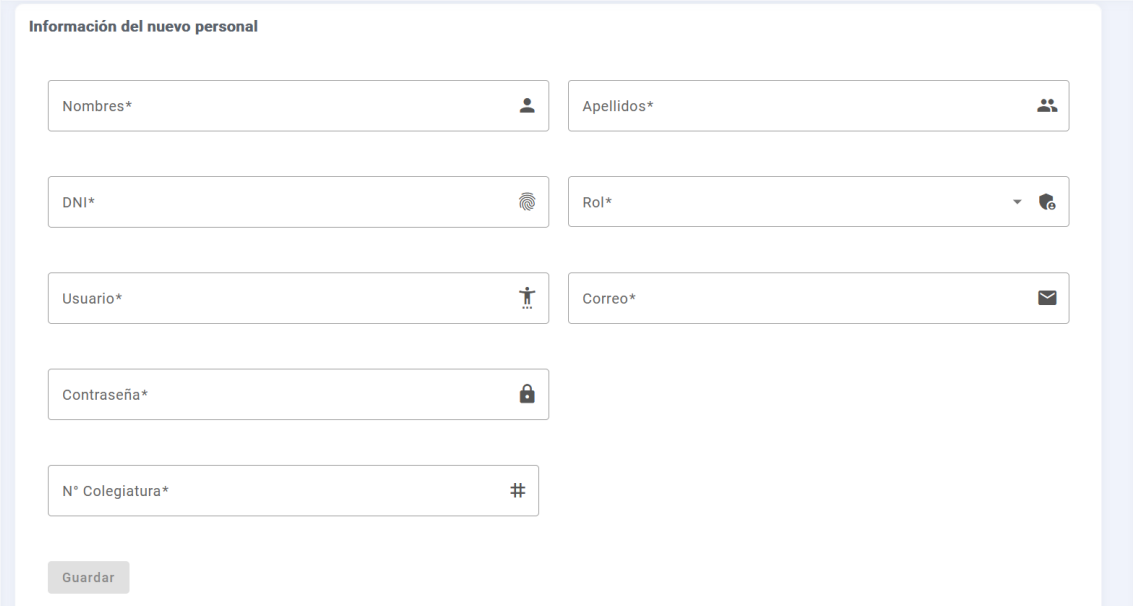
Creación del módulo en el frontend que permite administrar todas las vistas, importaciones y módulos externos correspondientes a este.

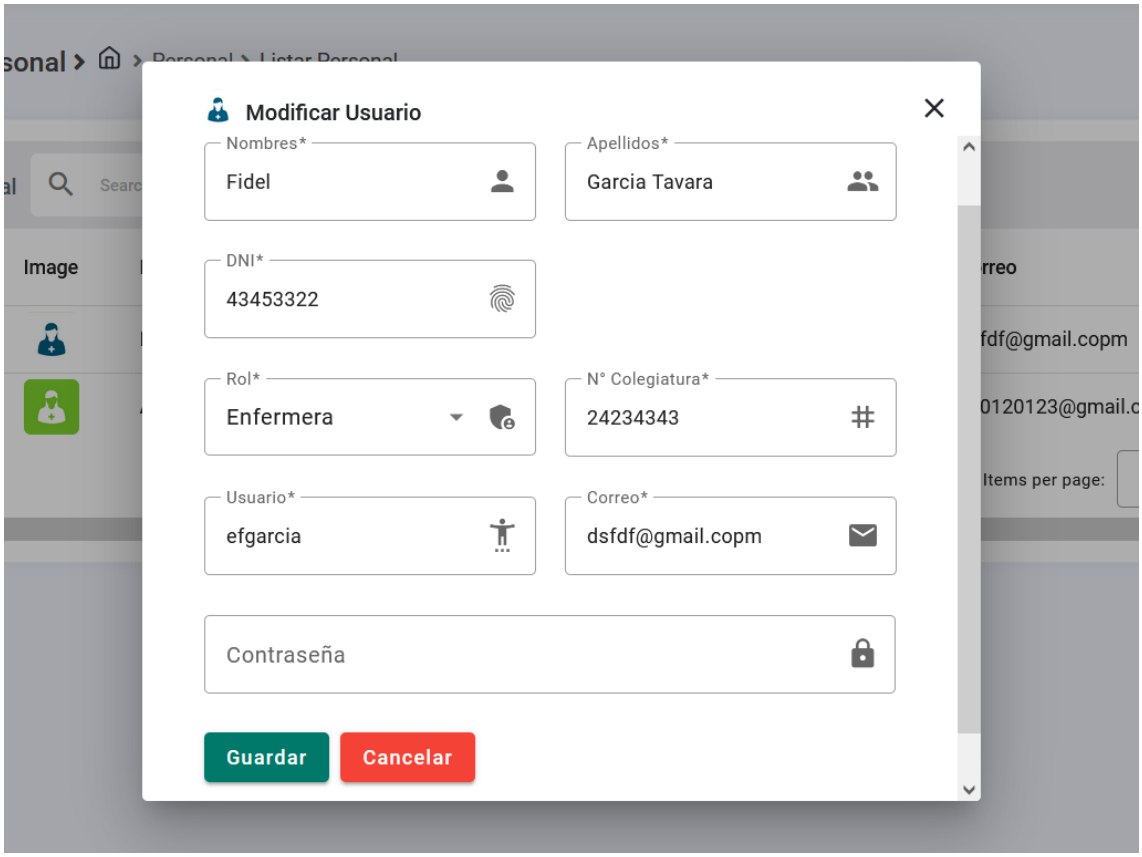


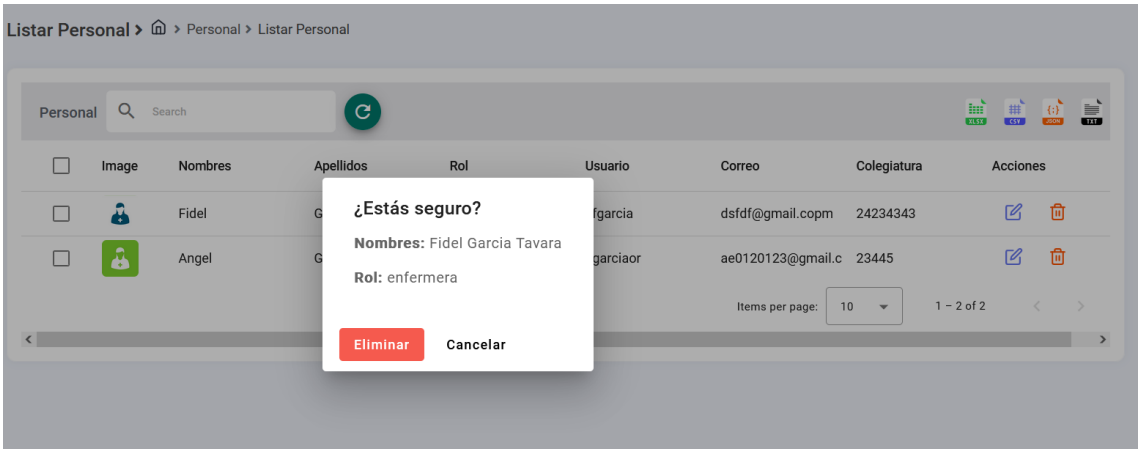
```
import { NgModule } from "@angular/core";
import { CommonModule } from "@angular/common";
import { FormsModule, ReactiveFormsModule } from "@angular/forms";
import { MatTableModule } from "@angular/material/table";
import { MatPaginatorModule } from "@angular/material/paginator";
import { MatFormFieldModule } from "@angular/material/form-field";
import { MatInputModule } from "@angular/material/input";
import { MatSnackBarModule } from "@angular/material/snack-bar";
import { MatButtonModule } from "@angular/material/button";
import { MatIconModule } from "@angular/material/icon";
import { MatSelectModule } from "@angular/material/select";
import { MatDialogModule } from "@angular/material/dialog";
import { MatSortModule } from "@angular/material/sort";
import { MatToolbarModule } from "@angular/material/toolbar";
import { MatDatepickerModule } from "@angular/material/datepicker";
import { StaffRoutingModule } from "./staff-routing.module";
import { AllstaffComponent } from "./pages/allstaff/allstaff.component";
import { MatCheckboxModule } from "@angular/material/checkbox";
import { MatTooltipModule } from "@angular/material/tooltip";
import { MatTableExporterModule } from "mat-table-exporter";
import { MatProgressSpinnerModule } from "@angular/material/progress-spinner";
import { FormDialogComponent } from "./pages/allstaff/dialog/form-dialog/form-dialog.component";
import { DeleteDialogComponent } from "./pages/allstaff/dialog/delete/delete.component";
import { AddStaffComponent } from "./pages/add-staff/add-staff.component";
import { MatTabsModule } from "@angular/material/tabs";
import { ComponentsModule } from "src/app/shared/components/components.module";
import { SharedModule } from "./../../shared/shared.module";
import { StoreModule } from "@ngrx/store";
import { EffectsModule } from "@ngrx/effects";
import { usersReducer } from "./state/reducers/users.reducers";
import { UsersEffects } from "./state/effects/users.effects";
import { StaffComponent } from "./staff.component";

@NgModule({
  declarations: [
    AllstaffComponent,
    FormDialogComponent,
    DeleteDialogComponent,
    AddStaffComponent,
    StaffComponent,
  ],
  imports: [
    CommonModule,
    FormsModule,
    ReactiveFormsModule,
    MatTableModule,
    MatPaginatorModule,
    MatFormFieldModule,
    MatInputModule,
    MatSnackBarModule,
    MatButtonModule,
    MatIconModule,
    MatDialogModule,
    MatSortModule,
    MatToolbarModule,
    MatSelectModule,
    MatCheckboxModule,
    MatDatepickerModule,
    MatTabsModule,
    MatTooltipModule,
    MatTableExporterModule,
    MatProgressSpinnerModule,
    StaffRoutingModule,
    ComponentsModule,
    SharedModule,
    StoreModule.forFeature('usuarios', usersReducer),
    EffectsModule.forFeature([UsersEffects]),
  ],
  providers: [],
})
export class StaffModule { }

```

- **Creación de interfaces de usuario**

Creación de interfaces de usuario del Frontend.







- **Comunicación entre backend y frontend**

Código fuente que muestra los métodos utilizados para consultar al backend desde el frontend.

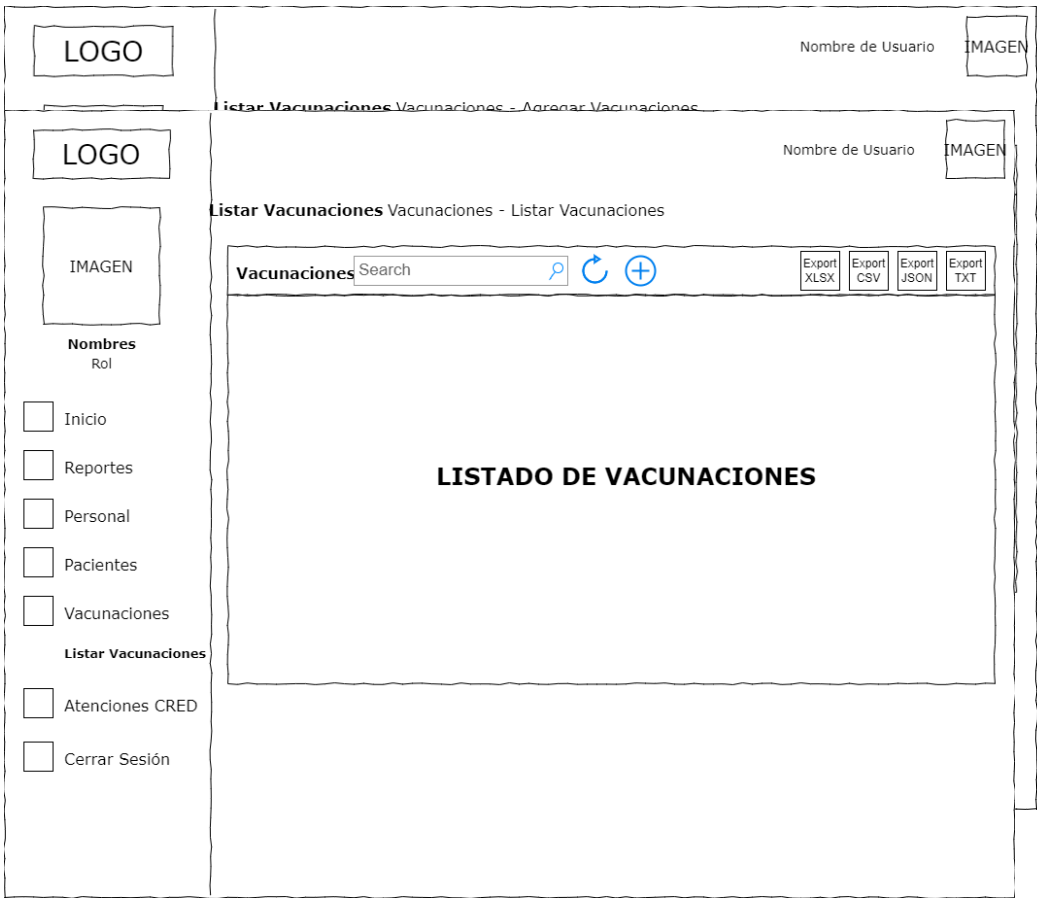
```
import { Injectable } from '@angular/core';
import { HttpClient, HttpErrorResponse, HttpHeaders } from '@angular/common/http';
import { BehaviorSubject, Observable } from 'rxjs';
import { delay } from 'rxjs/operators';
import { CreateUserDto } from '../dto/create-user.dto';
import { UnsubscribeOnDestroyAdapter } from 'src/app/shared/UnsubscribeOnDestroyAdapter';
import { User } from '../models/user.model';
import { Global } from 'src/app/config/global_url';
import { UpdateUserDto } from '../dto/update-user.dto';

@Injectable({
  providedIn: 'root'
})
export class UserService extends UnsubscribeOnDestroyAdapter {

  public url: string;
  public headers: HttpHeaders;

  isTblLoading = true;
  dataChange: BehaviorSubject<User[]> = new BehaviorSubject<User[]>([]);
  // Temporarily stores data from dialogs
  dialogData: any;

  constructor(
    private _http: HttpClient,
  ) {
    super();
    this.url = Global.url;
  }

  get data(): User[] {
    return this.dataChange.value;
  }

  getDialogData() {
    return this.dialogData;
  }

  /** CRUD METHODS */
  getUsers(): void {
    this.subs.sink = this._http.get<User[]>(this.url + 'user').subscribe(
      (data) => {
        this.isTblLoading = false;
        this.dataChange.next(data);
      },
      (error: HttpErrorResponse) => {
        this.isTblLoading = false;
      }
    );
  }

  getUserss(): Observable<any> {

    return this._http.get(this.url + 'user', { headers: this.headers }).pipe(delay(1000));
  }

  addUser(dto: CreateUserDto): void {
    this.dialogData = User;

    this._http.post(this.url + 'user', dto).subscribe(data => {
      this.dialogData = data;
    },
      (err: HttpErrorResponse) => {
        // error code here
      });
  }

  saveUser(object: CreateUserDto): Observable<any> {
    this.dialogData = User;

    return this._http.post(this.url + 'user', object, { headers: this.headers }).pipe(delay(2000));
  }

  deleteUser(id: any): Observable<any> {
    return this._http.delete(this.url + 'user/' + id, { headers: this.headers }).pipe(delay(2000));
  }

  updateUser(dto: UpdateUserDto): Observable<any> {
    //let headers = new HttpHeaders().set('Content-Type', 'application/x-www-form-urlencoded');
    return this._http.patch(this.url + 'user', dto, { headers: this.headers }).pipe(delay(2000));
  }
}
```

- **Sprint Review**

N° Historia	Sprint	Descripción de tarea	Estado	Estimación (dias)
	Sprint 1: Diseño de base de datos	Diseño de base de datos: Identificación de entidades	Concluido	2
		Diseño de base de datos: Diseño del modelo lógico	Concluido	2
		Diseño de base de datos: Diseño del modelo físico	Concluido	1
1 y 2	**Sprint 2:** Módulo de usuarios	Mockups	Concluido	1
		Creación de entidades (Backend)	Concluido	1
		Creación de servicios (Backend)	Concluido	2
		Creación de controladores (Backend)	Concluido	1
		Creación de DTOs y validaciones (Backend)	Concluido	1
		Prueba de APIs	Concluido	1
		Creación de módulos (Frontend)	Concluido	1
		Creación de interfaces de usuario (Frontend)	Concluido	3
		Comunicación entre backend y frontend	Concluido	1
		Sprint Review	Concluido	1
5	**Sprint 3:** Módulo de vacunaciones	Mockups	Pendiente	1
		Creación de entidades (Backend)	Pendiente	1
		Creación de servicios (Backend)	Pendiente	2
		Creación de controladores (Backend)	Pendiente	1
		Creación de DTOs y validaciones (Backend)	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	3
		Comunicación entre backend y frontend	Pendiente	1
		Sprint Review	Pendiente	1
4	**Sprint 4:** Módulo de atenciones	Mockups	Pendiente	2
		Creación de entidades (Backend)	Pendiente	1
		Creación de servicios (Backend)	Pendiente	3
		Creación de controladores (Backend)	Pendiente	1
		Creación de DTOs y validaciones (Backend)	Pendiente	2
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	4
		Comunicación entre backend y frontend	Pendiente	1
		Sprint Review	Pendiente	1
3, 8	**Sprint 5:** Módulo de pacientes	Mockups	Pendiente	1
		Creación de entidades (Backend)	Pendiente	1
		Creación de servicios (Backend)	Pendiente	2
		Creación de controladores (Backend)	Pendiente	1
		Creación de DTOs y validaciones (Backend)	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	3
		Comunicación entre backend y frontend	Pendiente	1
		Sprint Review	Pendiente	1
6 y 7	**Sprint 6:** Módulo de reportes	Mockups	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	1
		Comunicación entre backend y frontend	Pendiente	1
	Sprint 7: Producción y capacitación	Llenado de la base de datos	Pendiente	1
		Mandar la aplicación a producción	Pendiente	3
		Capacitación de los usuarios finales	Pendiente	1

SPRINT 3: Módulo Vacunaciones

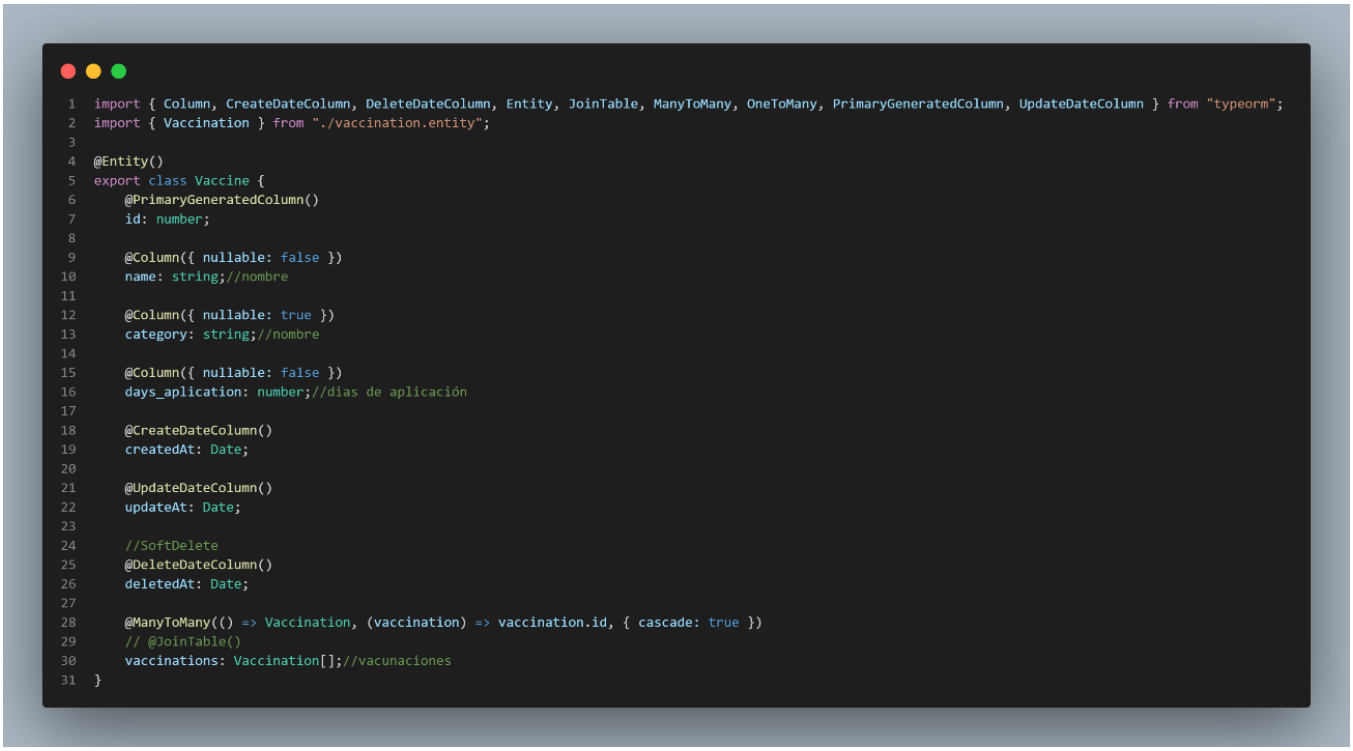
- **Mockups**



- **Creación de entidades**

Código fuente que muestra a las entidades de la base de datos pertenecientes al módulo de usuarios; las cuales son: vaccination y vaccine.

```typescript
import { Column, CreateDateColumn, Entity, JoinTable, ManyToMany, OneToMany, PrimaryGeneratedColumn, UpdateDateColumn } from "typeorm";
import { Vaccine } from "./vaccine.entity";

@Entity()
export class Vaccination {
    @PrimaryGeneratedColumn()
    id: number;

    @Column({ nullable: false })
    date_vaccination: Date;//Fecha de vacunación

    @Column({ nullable: false })
    num_hc: string;//número de historia

    @CreateDateColumn()
    createdAt: Date;

    @UpdateDateColumn()
    updateAt: Date;

    @ManyToMany(() => Vaccine, (vaccine) => vaccine.id, { cascade: true, eager: true })
    @JoinTable()
    vaccines: Vaccine[];//vacunas
}
```



```ts
import { Column, CreateDateColumn, DeleteDateColumn, Entity, JoinTable, ManyToMany, OneToMany, PrimaryGeneratedColumn, UpdateDateColumn } from "typeorm";
import { Vaccination } from "./vaccination.entity";

@Entity()
export class Vaccine {
    @PrimaryGeneratedColumn()
    id: number;

    @Column({ nullable: false })
    name: string;//nombre

    @Column({ nullable: true })
    category: string;//nombre

    @Column({ nullable: false })
    days_aplication: number;//dias de aplicación

    @CreateDateColumn()
    createdAt: Date;

    @UpdateDateColumn()
    updateAt: Date;

    //SoftDelete
    @DeleteDateColumn()
    deletedAt: Date;

    @ManyToMany(() => Vaccination, (vaccination) => vaccination.id, { cascade: true })
    // @JoinTable()
    vaccinations: Vaccination[];//vacunaciones
}
```

- **Creación de servicios**

Código fuente que muestra a los servicios utilizados para consultar a las entidades mencionadas anteriormente.

```ts
import { HttpException, HttpStatus, Injectable, NotFoundException } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { CreateVaccineDto, UpdateVaccineDto } from '../../dto';
import { Vaccine } from '../../entities/vaccine.entity';
import { VaccineInterface } from '../interfaces/vaccine.interface';

@Injectable()
export class VaccineService implements VaccineInterface {

    constructor(
        @InjectRepository(Vaccine) private readonly vaccineRepository: Repository<Vaccine>
    ) {
    }

    async getVaccines(): Promise<Vaccine[]> {
        const vaccines = await this.vaccineRepository.find({
            select: { id: true, name: true, category: true, days_aplication: true, createdAt: true },
            order: {
                days_aplication: 'asc'
            }
        });

        //Mensaje para cuando salga mal la consulta
        if (!vaccines) throw new NotFoundException("Algo salió mal.");

        //Mensaje para cuando no se encuentren registros
        if (vaccines && vaccines.length == 0) throw new HttpException({
            status: HttpStatus.BAD_REQUEST,
            error: 'No hay vacunas para mostrar.',
        }, HttpStatus.ACCEPTED)

        return vaccines;
    }

    async createVaccine({ name, days_aplication }: CreateVaccineDto) {
        const newVaccine = this.vaccineRepository.create({ name, days_aplication });
        return this.vaccineRepository.save(newVaccine);
    }

    async deleteVaccine(id: number): Promise<Object> {
        const deleteResponse = await this.vaccineRepository.softDelete(id);
        if (!deleteResponse.affected) {
            throw new NotFoundException(['Vacuna no encontrada.']);
        }

        return {
            status: HttpStatus.ACCEPTED,
            message: ['Vacuna eliminada exitosamente.'],
        }
    }

    async updateVaccine({ id, name, days_aplication }: UpdateVaccineDto) {
        const vaccine: Vaccine = await this.vaccineRepository.preload({
            id, name, days_aplication
        });

        if (!vaccine) {
            throw new NotFoundException(['Vacuna no encontrada.'])
        }

        this.vaccineRepository.save(vaccine);

        return vaccine;
    }
}

```

- **Creación de controladores**

Código fuente que contiene las APIs a las cuales se debe consultar para llamar a un determinado servicio.

```typescript
import { Body, Controller, Delete, Get, Param, Patch, Post } from '@nestjs/common';
import { CreateVaccinationDto, UpdateVaccinationDto } from '../../dto';

import { Vaccination } from '../../entities/vaccination.entity';
import { VaccinationService } from '../../services/vaccination/vaccination.service';

@Controller('vaccination')
export class VaccinationController {
    constructor(private readonly vaccinationService: VaccinationService) { }

    @Get()
    getVaccinations(): Promise<Vaccination[]> {
        return this.vaccinationService.getVaccinations();
    }

    @Post()
    createVaccination(@Body() createVaccination: CreateVaccinationDto): Promise<Vaccination> {
        return this.vaccinationService.createVaccination(createVaccination);
    }

    @Delete(':id')
    deleteVaccination(@Param('id') id: number): Promise<Object> {
        return this.vaccinationService.deleteVaccination(id);
    }

    @Patch()
    updateVaccination(@Body() vaccination: UpdateVaccinationDto): Promise<Vaccination> {
        return this.vaccinationService.updateVaccination(vaccination);
    }
}
```

```typescript
import { Body, Controller, Delete, Get, Param, Patch, Post, ParseIntPipe } from '@nestjs/common';
import { CreateVaccineDto, UpdateVaccineDto } from '../../dto';
import { Vaccine } from '../../entities/vaccine.entity';
import { VaccineService } from '../../services/vaccine/vaccine.service';

@Controller('vaccine')
export class VaccineController {
    constructor(private readonly vaccineService: VaccineService) { }

    @Get()
    getVaccines(): Promise<Vaccine[]> {
        return this.vaccineService.getVaccines();
    }

    @Post()
    //@HttpCode(HttpStatus.NO_CONTENT)  //Cambiar codigo de estado
    createVaccine(@Body() createVaccine: CreateVaccineDto): Promise<Vaccine> {
        return this.vaccineService.createVaccine(createVaccine);
    }

    @Delete(':id')
    deleteVaccine(@Param('id') id: number): Promise<Object> {
        return this.vaccineService.deleteVaccine(id);
    }

    @Patch(':id')
    updateVaccine(@Body() vaccine: UpdateVaccineDto): Promise<Vaccine> {
        return this.vaccineService.updateVaccine(vaccine);
    }
}
```

- **Creación de DTOs y validaciones**

Objeto de transferencia de datos que se utiliza al momento de enviar datos al backend para los diferentes casos de usos. Además, las validaciones personalizadas utilizadas para verificar si el ID de un tipo de usuario existe.

```typescript
import { IsDate, IsNotEmpty, IsAlphanumeric, ArrayMinSize, IsArray } from "class-validator";
import { HCExists } from "../custom/custom-validations/validate-num-hc-exists.validate";

export class CreateVaccinationDto {
    //Validación para el número de Historia Clínica
    @IsAlphanumeric('es-ES',
        {
            message: 'El N° de historia debe ser numérico.'
        }
    )
    @IsNotEmpty({
        message: 'El N° de historia es requerido.',
    })
    @HCExists()
    readonly num_hc: string;

    //Validación para la fecha de vacunación
    @IsDate({
        message: 'El valor ingresado debe de ser una fecha.'
    })
    @IsNotEmpty({
        message: 'La fecha de vacunación es requerida.',
    })
    readonly date_vaccination: Date;

    //Validación para las ID de las vacunas que se aplicaron en una vacunación
    @IsArray({ message: 'El array de vacunas debe ser de números.' })
    @ArrayMinSize(1, { message: 'Por favor enviar la lista de vacunas correctamente.', })
    @IsNotEmpty({
        message: 'El array de vacunas es requerido.',
    })
    readonly vaccines: Array<number>;
}
```

```typescript
import { IsInt, IsString, MaxLength, MinLength } from "class-validator";

export class CreateVaccineDto {
    //Validación para el nombre de la vacuna
    @MinLength(3, {
        message: 'El campo nombre debe contener 3 caracteres como mínimo.',
    })
    @MaxLength(50, {
        message: 'El campo nombre solo puede contener 50 caracteres como máximo.',
    })
    @IsString(
        {
            message: 'El campo nombre debe contener solo letras.'
        }
    )
    readonly name: string;

    //Validación para el día despues de nacido el niño en el que se aplicará dicha vacuna
    @IsInt(
        {
            message: 'Los dias de aplicación deben ser numericos.'
        }
    )
    readonly days_aplication: number;

}
```

```ts
import { IsInt, IsNotEmpty, IsNumber, IsString, MaxLength, MinLength } from "class-validator";
import { VaccineExists } from "../custom/custom-validations/validate-vaccine-id";

export class UpdateVaccineDto {
    //Validación para el numero de ID
    @IsNotEmpty({
        message: 'El id es requerido.',
    })
    @IsNumber({}, {
        message: 'El id debe ser numérico.'
    })
    @VaccineExists()
    readonly id: number;

    //Validación para el nombre de la vacuna
    @MinLength(3, {
        message: 'El campo nombre debe contener 3 caracteres como mínimo.',
    })
    @MaxLength(50, {
        message: 'El campo nombre solo puede contener 50 caracteres como máximo.',
    })
    @IsString(
        {
            message: 'El campo nombre debe contener solo letras.'
        }
    )
    readonly name: string;

    //Validación para el día despues de nacido el niño en el que se aplicará dicha vacuna
    @IsInt(
        {
            message: 'Los dias de aplicación deben ser numericos.'
        }
    )
    readonly days_aplication: number;

}

```

```ts
import { IsDate, IsNotEmpty, IsAlphanumeric, IsArray, ArrayMinSize, IsNumber } from "class-validator";

export class UpdateVaccinationDto {
    @IsNumber()
    @IsNotEmpty({
        message: 'El campo rol es requerido',
    })
    readonly id: number;

    //Validación para el número de Historia Clinica
    @IsAlphanumeric('es-ES',
        {
            message: 'El Nº de historia debe ser numerico.'
        }
    )
    @IsNotEmpty({
        message: 'El Nº de historia es requerido.',
    })
    readonly num_hc: string;

    //Validación para la fecha de vacunación
    @IsDate({
        message: 'El valor ingresado debe de ser una fecha.'
    })
    @IsNotEmpty({
        message: 'La fecha de vacunación es requerida.',
    })
    readonly date_vaccination: Date;

    //Validación para las ID de las vacunas que se aplicaron en una vacunación
    @IsArray({ message: 'El array de vacunas debe ser de números.' })
    @ArrayMinSize(1, { message: 'Porfavor enviar la lista de vacunas correctamente.', })
    @IsNotEmpty({
        message: 'El array de vacunas es requerido.',
    })
    readonly vaccines: Array<number>;
}
```

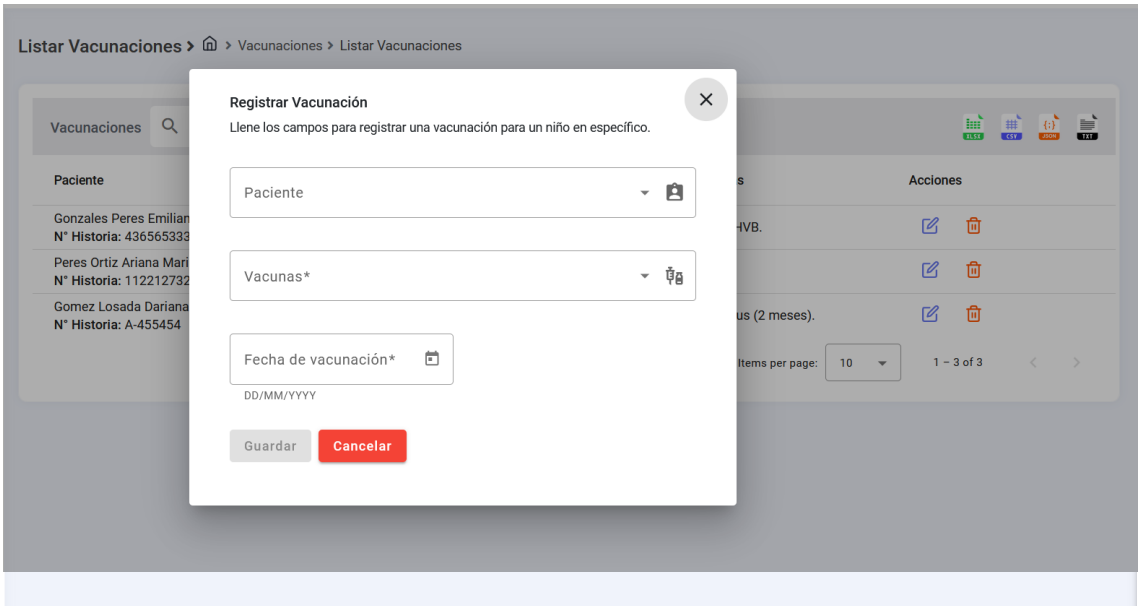
- **Creación de módulos**

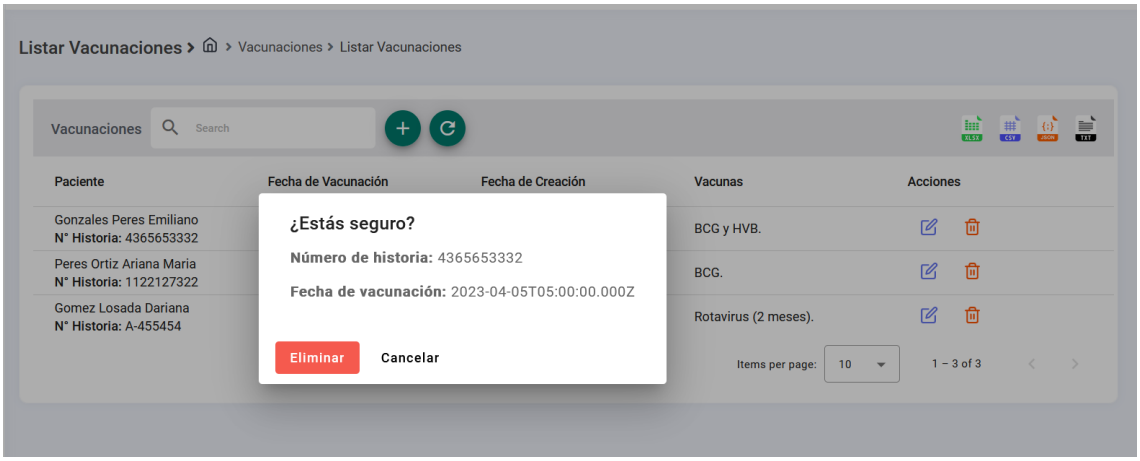
Creación del módulo en el frontend que permite administrar todas las vistas, importaciones y módulos externos correspondientes a este.

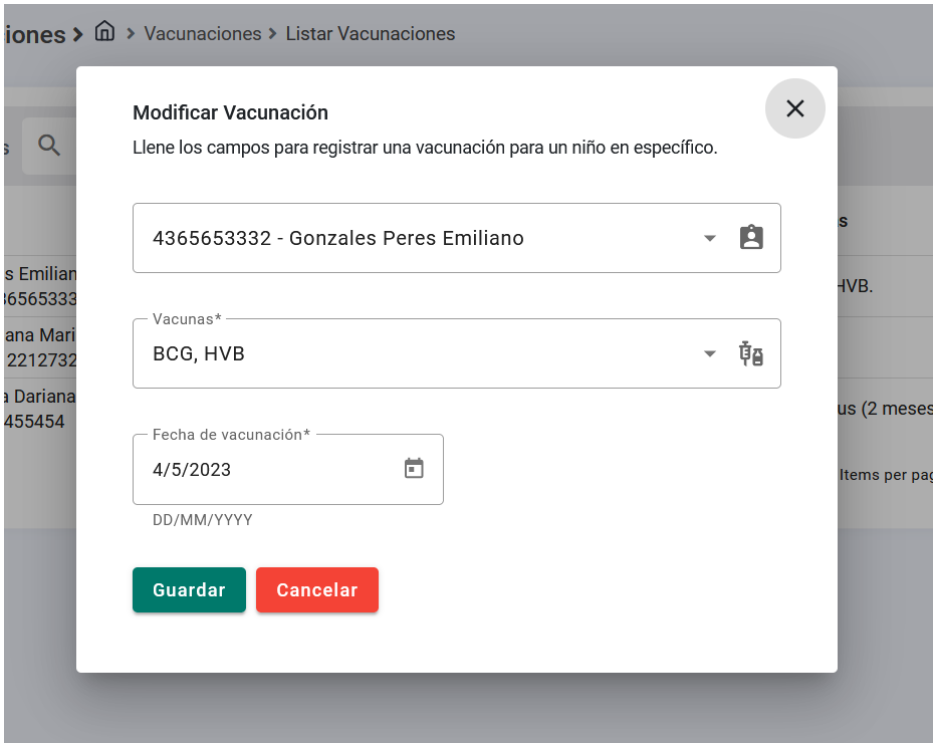
```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { VaccinationComponent } from './vaccination.component';
import { VaccinationsRoutingModule } from './vaccinations-routing.module';
import { AllVaccinationsComponent } from './pages/all-vaccinations/all-vaccinations.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { MatTableModule } from '@angular/material/table';
import { MatPaginatorModule } from '@angular/material/paginator';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import { SharedModule } from 'src/app/shared/shared.module';
import { ComponentsModule } from 'src/app/shared/components/components.module';
import { StaffRoutingModule } from '../staff/staff-routing.module';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { MatTableExporterModule } from 'mat-table-exporter';
import { MatTooltipModule } from '@angular/material/tooltip';
import { MatTabsModule } from '@angular/material/tabs';
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatSelectModule } from '@angular/material/select';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatSortModule } from '@angular/material/sort';
import { MatDialogModule } from '@angular/material/dialog';
import { MatIconModule } from '@angular/material/icon';
import { MatButtonModule } from '@angular/material/button';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import { EffectsModule } from '@ngrx/effects';
import { StoreModule } from '@ngrx/store';
import { vaccinationsReducer } from './state/reducers/vaccination.reducers';
import { VaccinationsEffects } from './state/effects/vaccination.effects';
import { DeleteDialogComponent } from './pages/all-vaccinations/dialog/delete/delete.component';
import { FormDialogComponent } from './pages/all-vaccinations/dialog/form-dialog/form-dialog.component';
import { NgxMatSelectSearchModule } from 'ngx-mat-select-search';

@NgModule({
  declarations: [
    VaccinationComponent,
    AllVaccinationsComponent,
    FormDialogComponent,
    DeleteDialogComponent,
  ],
  imports: [
    CommonModule,
    VaccinationsRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    MatTableModule,
    MatPaginatorModule,
    MatFormFieldModule,
    MatInputModule,
    MatSnackBarModule,
    MatButtonModule,
    MatIconModule,
    MatDialogModule,
    MatSortModule,
    MatToolbarModule,
    MatSelectModule,
    MatCheckboxModule,
    MatDatepickerModule,
    MatTabsModule,
    MatTooltipModule,
    MatTableExporterModule,
    MatProgressSpinnerModule,
    NgxMatSelectSearchModule,
    StaffRoutingModule,
    ComponentsModule,
    SharedModule,
    StoreModule.forFeature('vacunaciones', vaccinationsReducer),
    EffectsModule.forFeature([VaccinationsEffects]),
  ],
})
export class VaccinationModule { }

```

- **Creación de interfaces de usuario**

  Creación de interfaces de usuario del Frontend.



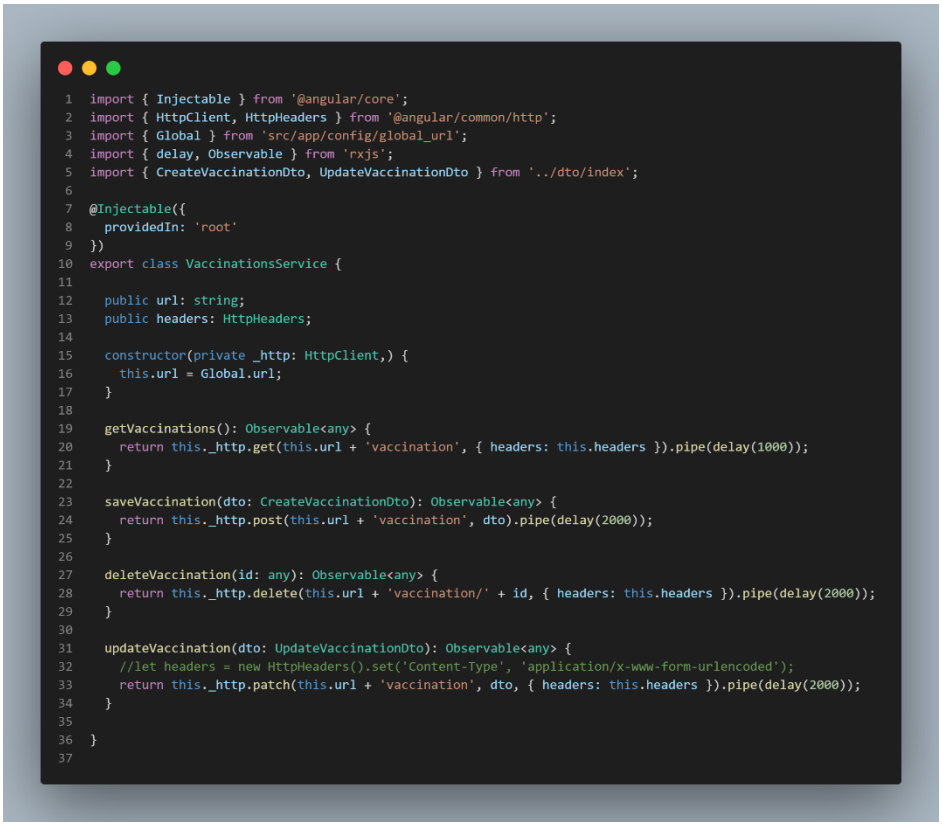



iones > > Vacunaciones > Listar Vacunaciones
Modificar Vacunación
Llene los campos para registrar una vacunación para un niño en específico.
4365653332 - Gonzales Peres Emiliano
Vacunas*
BCG, HVB
Fecha de vacunación*
4/5/2023
DD/MM/YYYY
Guardar
Cancelar

- **Comunicación entre backend y frontend**

Código fuente que muestra los métodos utilizados para consultar al backend desde el frontend.

```typescript
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';
import { delay } from 'rxjs/operators';
import { Global } from 'src/app/config/global_url';
import { CreateVaccineDto, UpdateVaccineDto } from '../dto/index';

@Injectable({
  providedIn: 'root'
})
export class VaccineService {

  public url: string;
  public headers: HttpHeaders;

  constructor(
    private _http: HttpClient,
  ) {
    this.url = Global.url;
  }

  /** CRUD METHODS */
  getVaccines(): Observable<any> {
    return this._http.get(this.url + 'vaccine', { headers: this.headers }).pipe(delay(1000));
  }

  saveVaccine(dto: CreateVaccineDto): Observable<any> {
    return this._http.post(this.url + 'vaccine', dto).pipe(delay(2000));
  }

  deleteVaccine(id: any): Observable<any> {
    return this._http.delete(this.url + 'vaccine/' + id, { headers: this.headers }).pipe(delay(2000));
  }

  updateVaccine(dto: UpdateVaccineDto): Observable<any> {
    //let headers = new HttpHeaders().set('Content-Type', 'application/x-www-form-urlencoded');
    return this._http.patch(this.url + 'vaccine', dto, { headers: this.headers }).pipe(delay(2000));
  }


}

```



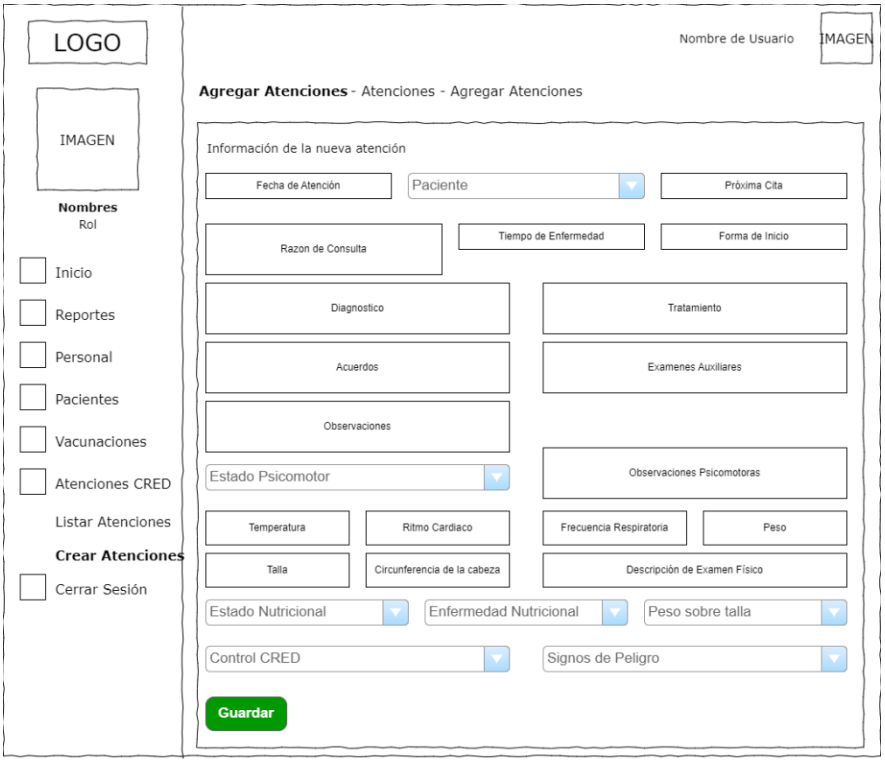
```ts
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Global } from 'src/app/config/global_url';
import { delay, Observable } from 'rxjs';
import { CreateVaccinationDto, UpdateVaccinationDto } from '../dto/index';

@Injectable({
  providedIn: 'root'
})
export class VaccinationsService {

  public url: string;
  public headers: HttpHeaders;

  constructor(private _http: HttpClient,) {
    this.url = Global.url;
  }

  getVaccinations(): Observable<any> {
    return this._http.get(this.url + 'vaccination', { headers: this.headers }).pipe(delay(1000));
  }

  saveVaccination(dto: CreateVaccinationDto): Observable<any> {
    return this._http.post(this.url + 'vaccination', dto).pipe(delay(2000));
  }

  deleteVaccination(id: any): Observable<any> {
    return this._http.delete(this.url + 'vaccination/' + id, { headers: this.headers }).pipe(delay(2000));
  }

  updateVaccination(dto: UpdateVaccinationDto): Observable<any> {
    //let headers = new HttpHeaders().set('Content-Type', 'application/x-www-form-urlencoded');
    return this._http.patch(this.url + 'vaccination', dto, { headers: this.headers }).pipe(delay(2000));
  }

}

```

- **Sprint Review**

N° Historia	Sprint	Descripción de tarea	Estado	Estimación (dias)
	Sprint 1: Diseño de base de datos	Diseño de base de datos: Identificación de entidades	Concluido	2
		Diseño de base de datos: Diseño del modelo lógico	Concluido	2
		Diseño de base de datos: Diseño del modelo físico	Concluido	1
1 y 2	**Sprint 2:** Módulo de usuarios	Mockups	Concluido	1
		Creación de entidades (Backend)	Concluido	1
		Creación de servicios (Backend)	Concluido	2
		Creación de controladores (Backend)	Concluido	1
		Creación de DTOs y validaciones (Backend)	Concluido	1
		Prueba de APIs	Concluido	1
		Creación de módulos (Frontend)	Concluido	1
		Creación de interfaces de usuario (Frontend)	Concluido	3
		Comunicación entre backend y frontend	Concluido	1
		Sprint Review	Concluido	1
5	**Sprint 3:** Módulo de vacunaciones	Mockups	Concluido	1
		Creación de entidades (Backend)	Concluido	1
		Creación de servicios (Backend)	Concluido	2
		Creación de controladores (Backend)	Concluido	1
		Creación de DTOs y validaciones (Backend)	Concluido	1
		Prueba de APIs	Concluido	1
		Creación de módulos (Frontend)	Concluido	1
		Creación de interfaces de usuario (Frontend)	Concluido	3
		Comunicación entre backend y frontend	Concluido	1
		Sprint Review	Concluido	1
4	**Sprint 4:** Módulo de atenciones	Mockups	Pendiente	2
		Creación de entidades (Backend)	Pendiente	1
		Creación de servicios (Backend)	Pendiente	3
		Creación de controladores (Backend)	Pendiente	1
		Creación de DTOs y validaciones (Backend)	Pendiente	2
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	4
		Comunicación entre backend y frontend	Pendiente	1
		Sprint Review	Pendiente	1
3, 8	**Sprint 5:** Módulo de pacientes	Mockups	Pendiente	1
		Creación de entidades (Backend)	Pendiente	1
		Creación de servicios (Backend)	Pendiente	2
		Creación de controladores (Backend)	Pendiente	1
		Creación de DTOs y validaciones (Backend)	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	3
		Comunicación entre backend y frontend	Pendiente	1
		Sprint Review	Pendiente	1
6 y 7	**Sprint 6:** Módulo de reportes	Mockups	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	1
		Comunicación entre backend y frontend	Pendiente	1
	Sprint 7: Producción y capacitación	Llenado de la base de datos	Pendiente	1
		Mandar la aplicación a producción	Pendiente	3
		Capacitación de los usuarios finales	Pendiente	1

SPRINT 4: Módulo Atenciones

- **Mockups**

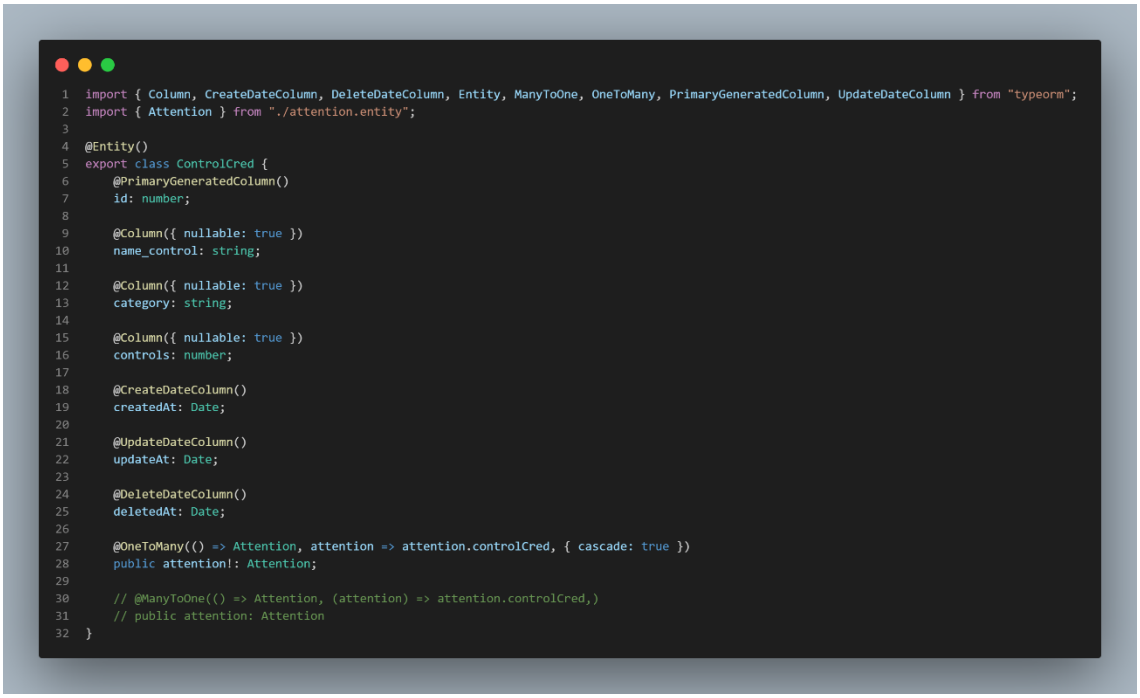


- **Creación de entidades**

Código fuente que muestra a las entidades de la base de datos pertenecientes al módulo de usuarios; las cuales son: attention, anamnesis, ControlCRED, entre otras.

```typescript
import { Column, CreateDateColumn, DeleteDateColumn, Entity, JoinColumn, ManyToOne, OneToMany, OneToOne, PrimaryGeneratedColumn, UpdateDateColumn } from "typeorm";
import { Attention } from "./attention.entity";
import { PsychomotorDevelopment } from "./psychomotor_development.entity";
import { PhysicalExam } from "./physical_exam.entity";
import { GrowthNutritionalStatus } from "./growth_nutritional_status.entity";

@Entity()
export class Anamnesis {
    @PrimaryGeneratedColumn()
    id: number;

    @Column({ nullable: true })
    reason_consultation: string;

    @Column({ nullable: true })
    sick_time: string;

    @Column({ nullable: true })
    start_shape: string;

    @Column({ nullable: true })
    diagnosis: string;

    @Column({ nullable: true })
    treatment: string;

    @Column({ nullable: true })
    agreements: string;

    @Column({ nullable: true })
    auxiliary_exams: string;

    @Column({ nullable: true })
    observations: string;

    @OneToOne(() => PsychomotorDevelopment, { cascade: true, eager: true } )
    @JoinColumn()
    psychomotorDevelopment: PsychomotorDevelopment;

    @OneToOne(() => PhysicalExam, { cascade: true, eager: true } )
    @JoinColumn()
    physicalExam: PhysicalExam;

    @OneToOne(() => GrowthNutritionalStatus, { cascade: true, eager: true } )
    @JoinColumn()
    growthNutritionalStatus: GrowthNutritionalStatus;
}
```

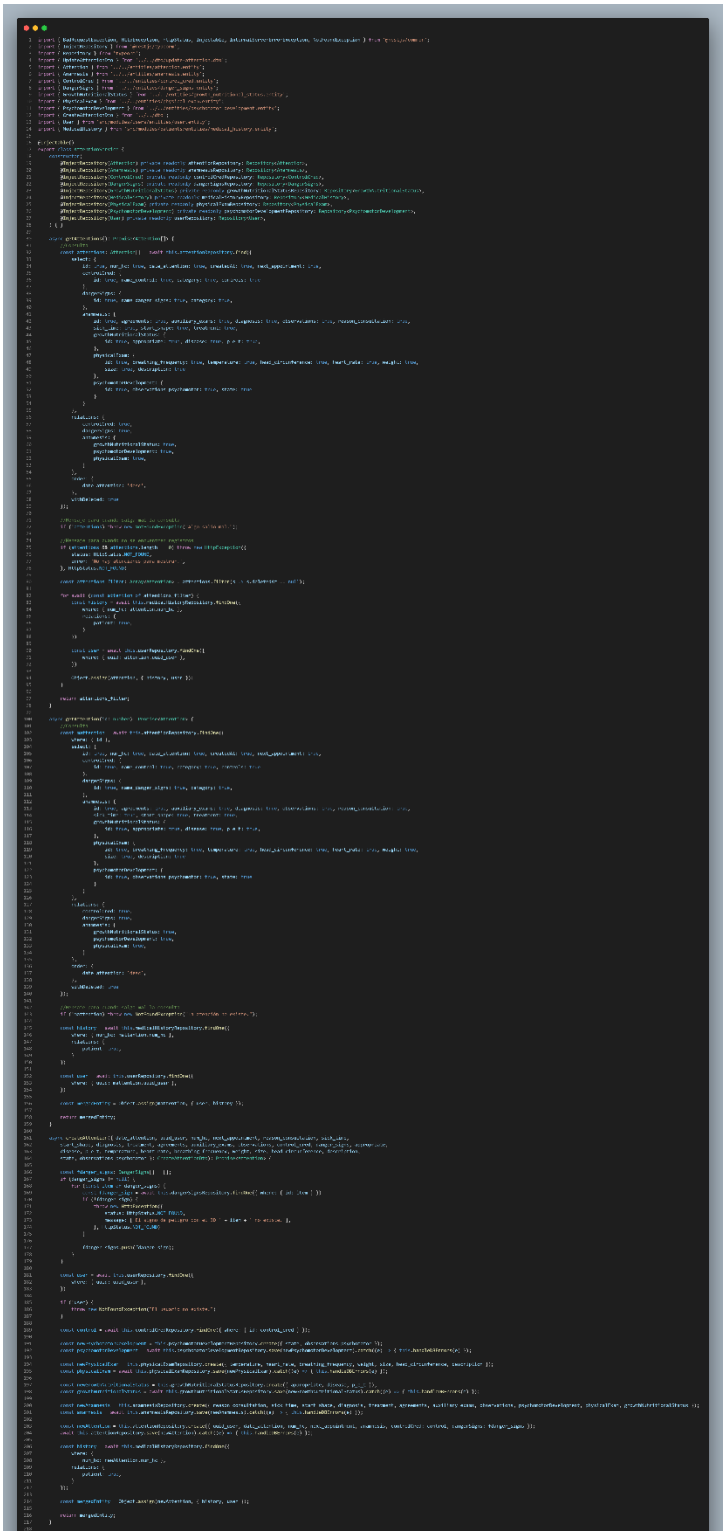
```typescript
import { Column, CreateDateColumn, DeleteDateColumn, Entity, OneToMany, JoinTable, ManyToMany, PrimaryGeneratedColumn, UpdateDateColumn, ManyToOne, OneToOne, JoinColumn, BeforeInsert } from "typeorm";
import { Anamnesis } from "./anamnesis.entity";
import { ControlCred } from "./control_cred.entity";
import { DangerSigns } from "./danger_signs.entity";

@Entity()
export class Attention {
    @PrimaryGeneratedColumn()
    id: number;

    // @Column({ nullable: true })
    // age: number;

    @Column({ nullable: true })
    date_attention: Date;

    @Column({ nullable: true })
    num_hc: string;

    @Column({ nullable: true })
    next_appointment: Date;

    @Column({ nullable: true })
    uuid_user: string;

    @CreateDateColumn()
    createdAt: Date;

    @UpdateDateColumn()
    updatedAt: Date;

    @DeleteDateColumn()
    deletedAt: Date;

    // @BeforeInsert()
    // async beforeInsert() {
    //     const now = new Date();
    //     this.date_attention = now.toISOString();
    // }

    //ESTAS SON RELACIONES
    @OneToOne(() => Anamnesis, { cascade: true })
    @JoinColumn()
    anamnesis: Anamnesis;

    @ManyToOne(() => ControlCred, (controlCred) => controlCred.attention,)
    public controlCred: ControlCred

    @ManyToMany(() => DangerSigns, (dangerSigns) => dangerSigns.id, { cascade: true })
    @JoinTable()
    dangerSigns: DangerSigns[];

}
```



```ts
import { Column, CreateDateColumn, DeleteDateColumn, Entity, ManyToOne, OneToMany, PrimaryGeneratedColumn, UpdateDateColumn } from "typeorm";
import { Attention } from "./attention.entity";

@Entity()
export class ControlCred {
    @PrimaryGeneratedColumn()
    id: number;

    @Column({ nullable: true })
    name_control: string;

    @Column({ nullable: true })
    category: string;

    @Column({ nullable: true })
    controls: number;

    @CreateDateColumn()
    createdAt: Date;

    @UpdateDateColumn()
    updateAt: Date;

    @DeleteDateColumn()
    deletedAt: Date;

    @OneToMany(() => Attention, attention => attention.controlCred, { cascade: true })
    public attention!: Attention;

    // @ManyToOne(() => Attention, (attention) => attention.controlCred,)
    // public attention: Attention
}
```

- **Creación de servicios**

Código fuente que muestra a los servicios utilizados para consultar a las entidades mencionadas anteriormente.



```typescript
import { HttpException, HttpStatus, Injectable, NotFoundException } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { ControlCred } from '../../entities/control_cred.entity';
import { DangerSigns } from '../../entities/danger_signs.entity';

@Injectable()
export class DangerSignsService {

    constructor(
        @InjectRepository(DangerSigns) private readonly dangerSignsRepository: Repository<DangerSigns>
    ) {}

    async getDangerSigns(): Promise<DangerSigns[]> {
        const dangerSigns = await this.dangerSignsRepository.find({
            select: { id: true, name_danger_signs: true, category: true }
        });

        //Mensaje para cuando salga mal la consulta
        if (!dangerSigns) throw new NotFoundException("Algo salió mal.");

        //Mensaje para cuando no se encuentren registros
        if (dangerSigns && dangerSigns.length == 0) throw new HttpException({
            status: HttpStatus.BAD_REQUEST,
            error: 'No hay signos de peligro para mostrar.',
        }, HttpStatus.ACCEPTED)

        return dangerSigns;
    }

}

```

```typescript
import { HttpException, HttpStatus, Injectable, NotFoundException } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { ControlCred } from '../../entities/control_cred.entity';

@Injectable()
export class ControlCredService {

    constructor(
        @InjectRepository(ControlCred) private readonly controlCredRepository: Repository<ControlCred>
    ) { }

    async getControlCred(): Promise<ControlCred[]> {
        const controlCred = await this.controlCredRepository.find({
            select: { id: true, controls: true, name_control: true, category: true },
            order: {
                controls: 'asc'
            }
        });

        //Mensaje para cuando salga mal la consulta
        if (!controlCred) throw new NotFoundException("Algo salió mal.");

        //Mensaje para cuando no se encuentren registros
        if (controlCred && controlCred.length == 0) throw new HttpException({
            status: HttpStatus.BAD_REQUEST,
            error: 'No hay controles CRED para mostrar.',
        }, HttpStatus.ACCEPTED)

        return controlCred;
    }

}

```

- **Creación de controladores**

Código fuente que contiene las APIs a las cuales se debe consultar para llamar a un determinado servicio.

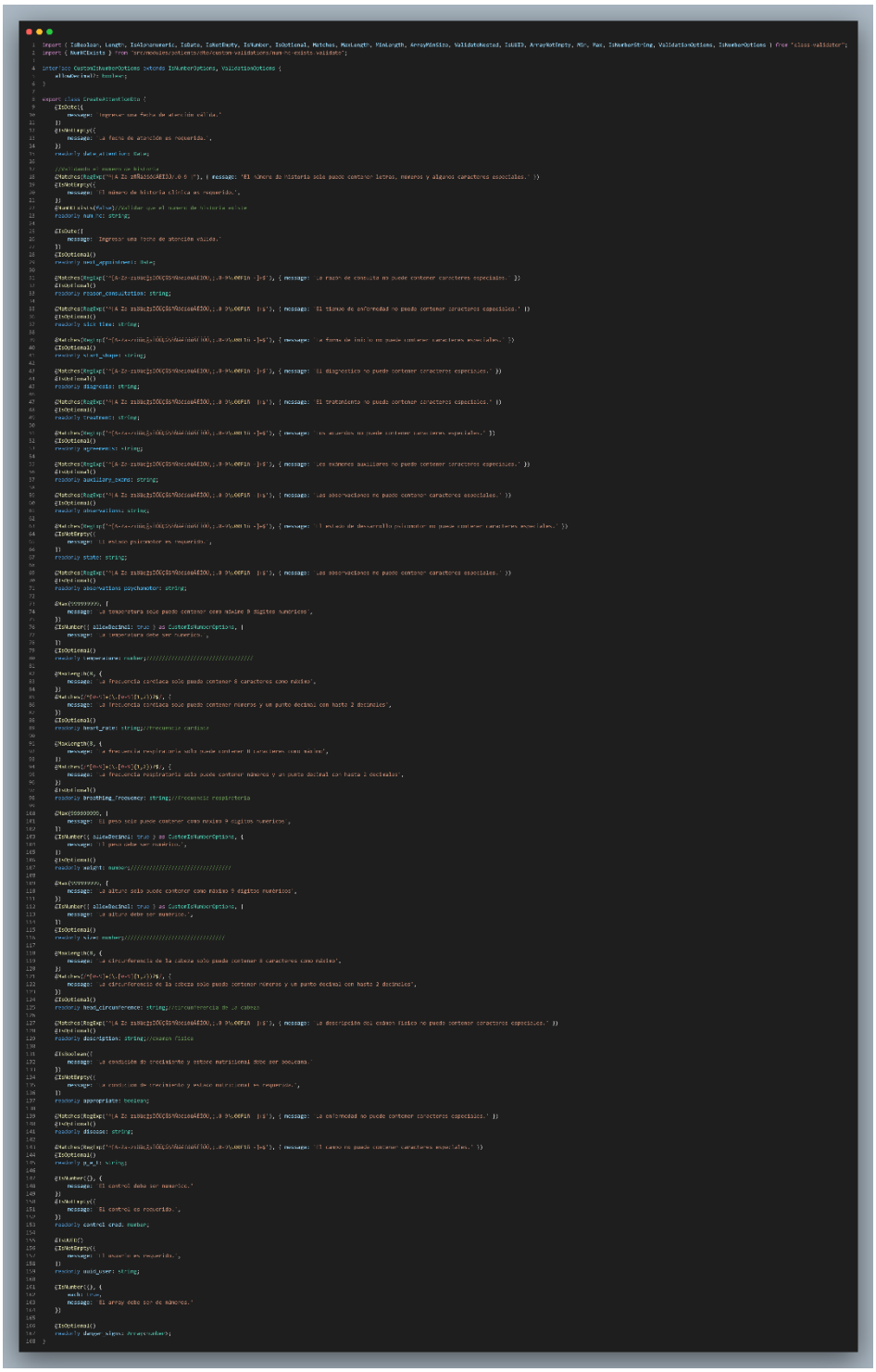
```
import { Controller, ParseIntPipe } from '@nestjs/common';
import { Body, Param, Get, Delete, Patch, Post } from '@nestjs/common/decorators';
import { CreateAttentionDto } from '../../dto/create-attention.dto';
import { UpdateAttentionDto } from '../../dto/update-attention.dto';
import { Attention } from '../../entities/attention.entity';
import { AttentionService } from '../../services/attention/attention.service';
import { Auth } from 'src/modules/auth/decorators';
import { ValidRoles } from 'src/modules/auth/interfaces/valid-roles';

@Controller('attention')
export class AttentionController {
    constructor(private readonly attentionService: AttentionService) { }

    @Get()
    getAttentions(): Promise<Attention[]> {
        return this.attentionService.getAttentions();
    }

    @Get('view/:id')
    // @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    getAttention(@Param('id') id,): Promise<Attention> {
        return this.attentionService.getAttention(id);
    }

    @Post()
    createAttention(@Body() createAttention: CreateAttentionDto): Promise<Attention> {
        return this.attentionService.createAttention(createAttention);
    }

    @Delete(':id')
    deleteAttention(@Param('id') id: number): Promise<Object> {
        return this.attentionService.deleteAttention(id);
    }

    @Patch()
    updateAttentions(@Body() attention: UpdateAttentionDto): Promise<Attention> {
        return this.attentionService.updateAttention(attention);
    }
}

```

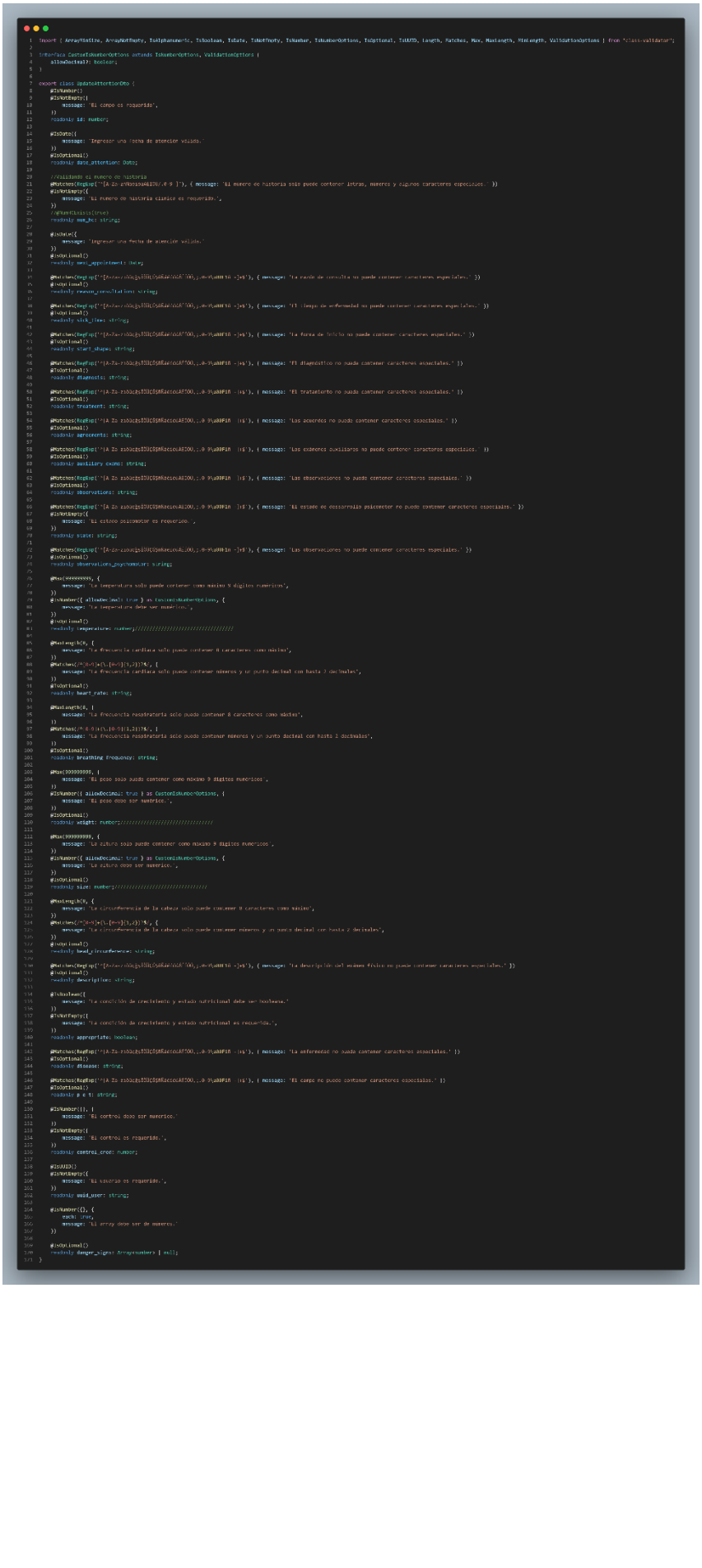
```ts
import { Controller } from '@nestjs/common';
import { Get } from '@nestjs/common/decorators';
import { ControlCred } from '../../entities/control_cred.entity';
import { ControlCredService } from '../../services/controlCred/control_cred.service';

@Controller('control-cred')
export class ControlCredController {
    constructor(private readonly controlCredService: ControlCredService) { }

    @Get()
    getControlCred(): Promise<ControlCred[]> {
        return this.controlCredService.getControlCred();
    }

}

```

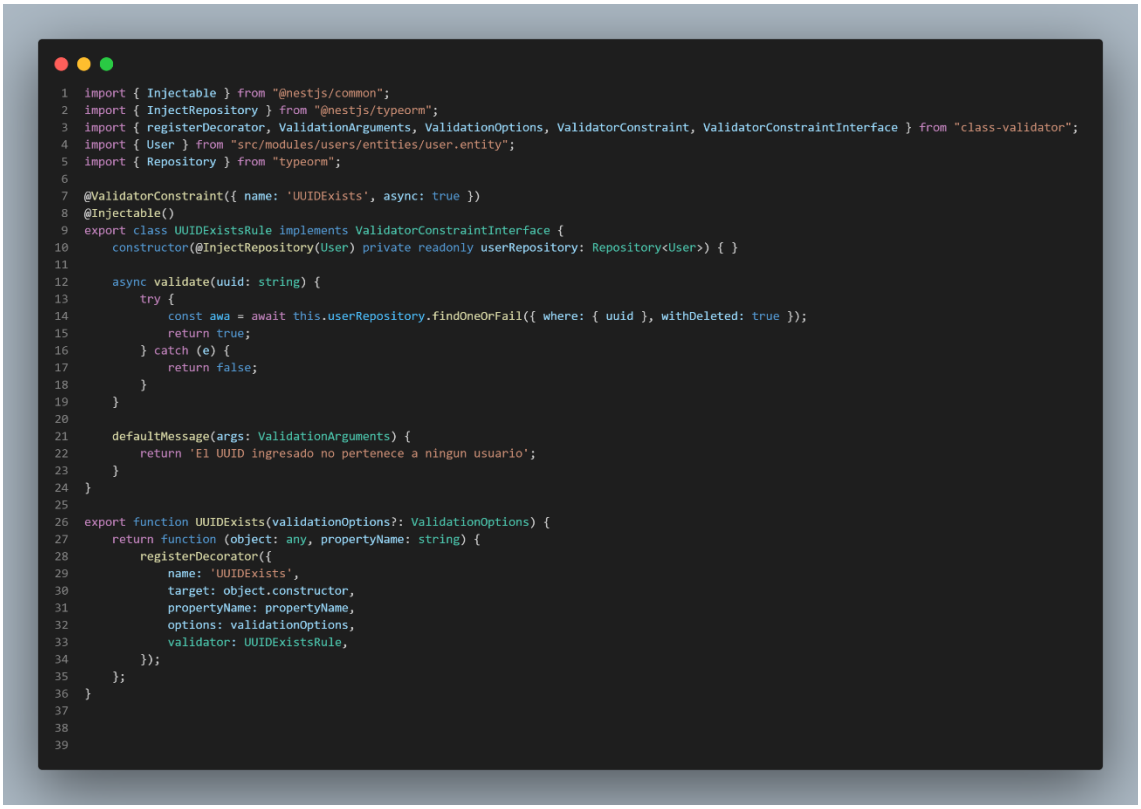
```ts
import { Controller } from '@nestjs/common';
import { Get } from '@nestjs/common/decorators';
import { DangerSigns } from '../../entities/danger_signs.entity';
import { DangerSignsService } from '../../services/dangerSigns/danger_signs.service';

@Controller('danger-signs')
export class DangerSignsController {
    constructor(private readonly dangerSignsService: DangerSignsService) { }

    @Get()
    getDangerSigns(): Promise<DangerSigns[]> {
        return this.dangerSignsService.getDangerSigns();
    }

}

```

- **Creación de DTOs y validaciones**

Objeto de transferencia de datos que se utiliza al momento de enviar datos al backend para los diferentes casos de usos. Además, las validaciones personalizadas utilizadas para verificar si el ID de un tipo de usuario existe.







```ts
import { Injectable } from "@nestjs/common";
import { InjectRepository } from "@nestjs/typeorm";
import { registerDecorator, ValidationArguments, ValidationOptions, ValidatorConstraint, ValidatorConstraintInterface } from "class-validator";
import { User } from "src/modules/users/entities/user.entity";
import { Repository } from "typeorm";

@ValidatorConstraint({ name: 'UUIDExists', async: true })
@Injectable()
export class UUIDExistsRule implements ValidatorConstraintInterface {
    constructor(@InjectRepository(User) private readonly userRepository: Repository<User>) { }

    async validate(uuid: string) {
        try {
            const awa = await this.userRepository.findOneOrFail({ where: { uuid }, withDeleted: true });
            return true;
        } catch (e) {
            return false;
        }
    }

    defaultMessage(args: ValidationArguments) {
        return 'El UUID ingresado no pertenece a ningun usuario';
    }
}

export function UUIDExists(validationOptions?: ValidationOptions) {
    return function (object: any, propertyName: string) {
        registerDecorator({
            name: 'UUIDExists',
            target: object.constructor,
            propertyName: propertyName,
            options: validationOptions,
            validator: UUIDExistsRule,
        });
    };
}
```

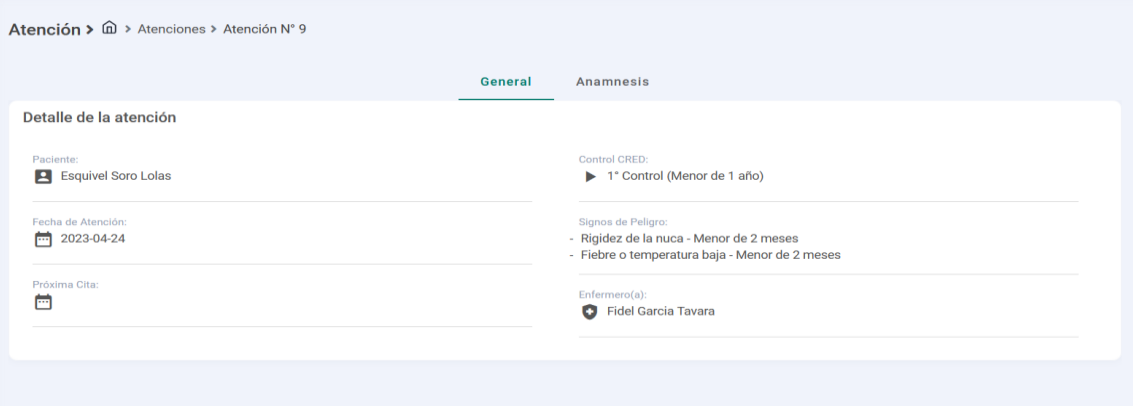
- **Creación de módulos**

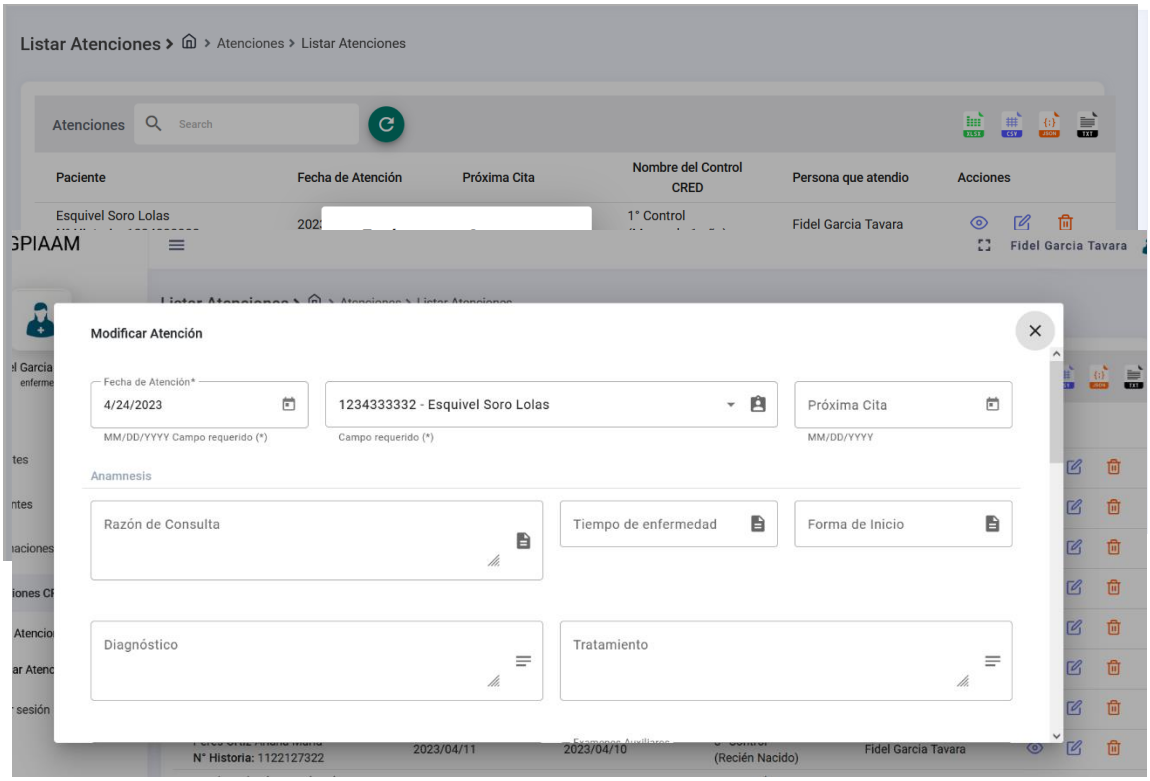
Creación del módulo en el frontend que permite administrar todas las vistas, importaciones y módulos externos correspondientes a este.

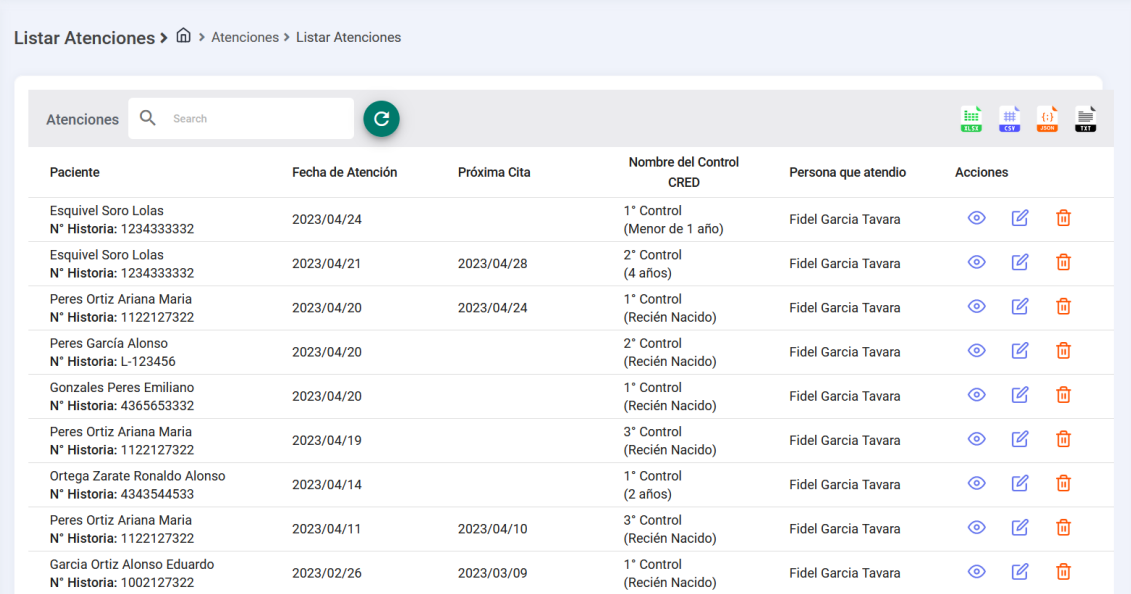
```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { AttentionComponent } from './attention.component';
import { AllAttentionsComponent } from './pages/all-attentions/all-attentions.component';
import { StoreModule } from '@ngrx/store';
import { EffectsModule } from '@ngrx/effects';
import { AttentionRoutingModule } from './attention-routing.module';
import { attentionsReducer } from './state/reducers/attention.reducers';
import { AttentionsEffects } from './state/effects/attention.effects';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { MatTableModule } from '@angular/material/table';
import { MatPaginatorModule } from '@angular/material/paginator';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import { MatButtonModule } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';
import { MatDialogModule } from '@angular/material/dialog';
import { MatSortModule } from '@angular/material/sort';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatSelectModule } from '@angular/material/select';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatTabsModule } from '@angular/material/tabs';
import { MatTooltipModule } from '@angular/material/tooltip';
import { MatTableExporterModule } from 'mat-table-exporter';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { NgxMatSelectSearchModule } from 'ngx-mat-select-search';
import { StaffRoutingModule } from '../staff/staff-routing.module';
import { ComponentsModule } from 'src/app/shared/components/components.module';
import { SharedModule } from 'src/app/shared/shared.module';
import { AddAttentionsComponent } from './pages/add-attentions/add-attentions.component';
import { DeleteDialogComponent } from './pages/all-attentions/dialog/delete/delete.component';
import { FormDialogComponent } from './pages/all-attentions/dialog/form-dialog/form-dialog.component';
import { ViewAttentionComponent } from './pages/view-attention/view-attention.component';
import { NgxSkeletonLoaderModule } from 'ngx-skeleton-loader';
import { NgApexchartsModule } from 'ng-apexcharts';
import { MatChipsModule } from '@angular/material/chips';
import { MatStepperModule } from '@angular/material/stepper';

@NgModule({
  declarations: [
    AttentionComponent,
    AllAttentionsComponent,
    AddAttentionsComponent,
    FormDialogComponent,
    DeleteDialogComponent,
    ViewAttentionComponent,
  ],
  imports: [
    CommonModule,
    AttentionRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    MatTableModule,
    MatPaginatorModule,
    MatFormFieldModule,
    MatInputModule,
    MatSnackBarModule,
    MatButtonModule,
    MatIconModule,
    MatDialogModule,
    MatStepperModule,
    MatSortModule,
    MatToolbarModule,
    MatSelectModule,
    MatCheckboxModule,
    MatDatepickerModule,
    MatTabsModule,
    MatChipsModule,
    MatTooltipModule,
    MatTableExporterModule,
    MatProgressSpinnerModule,
    NgApexchartsModule,
    NgxMatSelectSearchModule,
    StaffRoutingModule,
    ComponentsModule,
    SharedModule,
    NgxSkeletonLoaderModule,
    StoreModule.forFeature('atenciones', attentionsReducer),
    EffectsModule.forFeature([AttentionsEffects]),
  ],
})
export class AttentionModule { }
```

- **Creación de interfaces de usuario**

  Creación de interfaces de usuario del Frontend.







Paciente	Fecha de Atención	Próxima Cita	Nombre del Control CRED	Persona que atendio	Acciones
Esquivel Soro Lolas **N° Historia:** 1234333332	2023/04/24		1° Control (Menor de 1 año)	Fidel Garcia Tavara	
Esquivel Soro Lolas **N° Historia:** 1234333332	2023/04/21	2023/04/28	2° Control (4 años)	Fidel Garcia Tavara	
Peres Ortiz Ariana Maria **N° Historia:** 1122127322	2023/04/20	2023/04/24	1° Control (Recién Nacido)	Fidel Garcia Tavara	
Peres García Alonso **N° Historia:** L-123456	2023/04/20		2° Control (Recién Nacido)	Fidel Garcia Tavara	
Gonzales Peres Emiliano **N° Historia:** 4365653332	2023/04/20		1° Control (Recién Nacido)	Fidel Garcia Tavara	
Peres Ortiz Ariana Maria **N° Historia:** 1122127322	2023/04/19		3° Control (Recién Nacido)	Fidel Garcia Tavara	
Ortega Zarate Ronaldo Alonso **N° Historia:** 4343544533	2023/04/14		1° Control (2 años)	Fidel Garcia Tavara	
Peres Ortiz Ariana Maria **N° Historia:** 1122127322	2023/04/11	2023/04/10	3° Control (Recién Nacido)	Fidel Garcia Tavara	
Garcia Ortiz Alonso Eduardo **N° Historia:** 1002127322	2023/02/26	2023/03/09	1° Control (Recién Nacido)	Fidel Garcia Tavara	

- **Comunicación entre backend y frontend**

Código fuente que muestra los métodos utilizados para consultar al backend desde el frontend.

```typescript
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { delay, Observable } from 'rxjs';
import { Global } from 'src/app/config/global_url';
import { CreateAttentionDto, UpdateAttentionDto } from '../dto';

@Injectable({
  providedIn: 'root'
})
export class AttentionsService {

  public url: string;
  public headers: HttpHeaders;

  constructor(private _http: HttpClient,) {
    this.url = Global.url;
  }

  getAttentions(): Observable<any> {
    return this._http.get(this.url + 'attention', { headers: this.headers }).pipe(delay(1000));
  }

  getAttention(id: number): Observable<any> {
    return this._http.get(this.url + 'attention/view/' + id, { headers: this.headers }).pipe(delay(1000));
  }

  saveAttention(dto: CreateAttentionDto): Observable<any> {
    return this._http.post(this.url + 'attention', dto).pipe(delay(2000));
  }

  deleteAttention(id: any): Observable<any> {
    return this._http.delete(this.url + 'attention/' + id, { headers: this.headers }).pipe(delay(2000));
  }

  updateAttention(dto: UpdateAttentionDto): Observable<any> {
    return this._http.patch(this.url + 'attention', dto, { headers: this.headers }).pipe(delay(2000));
  }

}
```

```typescript
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { delay, Observable } from 'rxjs';
import { Global } from 'src/app/config/global_url';

@Injectable({
  providedIn: 'root'
})
export class ControlCredService {

  public url: string;
  public headers: HttpHeaders;

  constructor(private _http: HttpClient,) {
    this.url = Global.url;
  }

  getControlCred(): Observable<any> {
    return this._http.get(this.url + 'control-cred', { headers: this.headers }).pipe(delay(1000));
  }

}

```

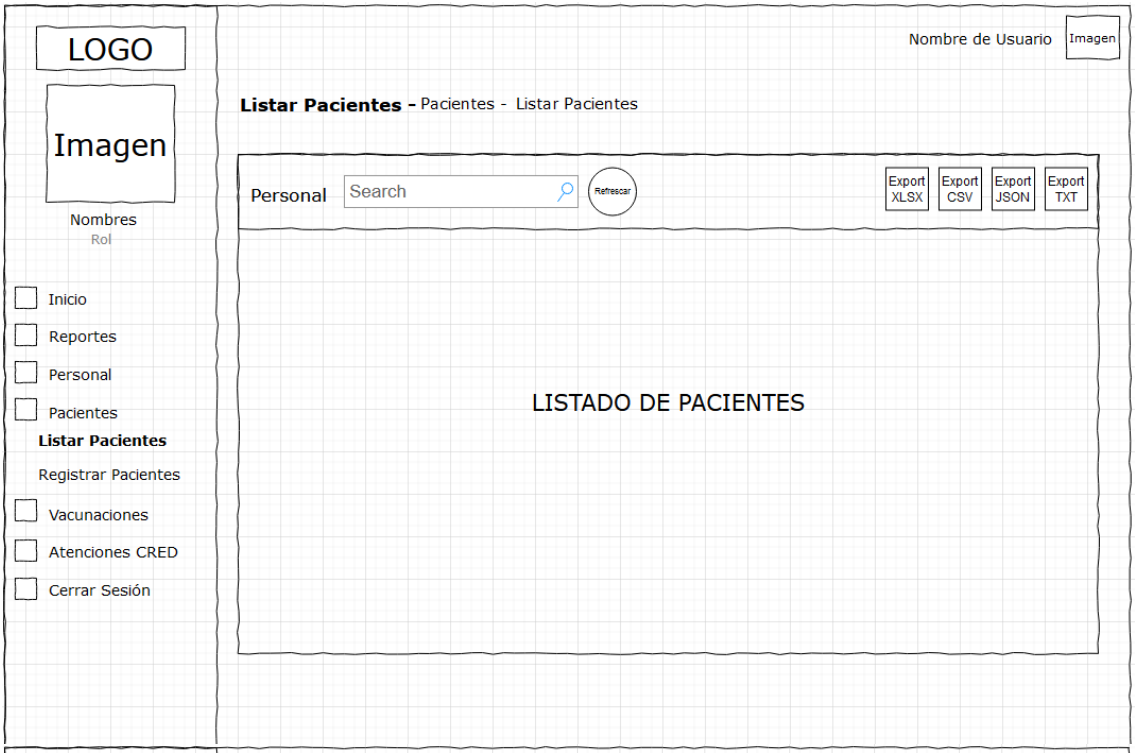
```typescript
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { delay, Observable } from 'rxjs';
import { Global } from 'src/app/config/global_url';

@Injectable({
  providedIn: 'root'
})
export class DangerSignsService {

  public url: string;
  public headers: HttpHeaders;

  constructor(private _http: HttpClient,) {
    this.url = Global.url;
  }

  getDangerSigns(): Observable<any> {
    return this._http.get(this.url + 'danger-signs', { headers: this.headers }).pipe(delay(1000));
  }

}

```

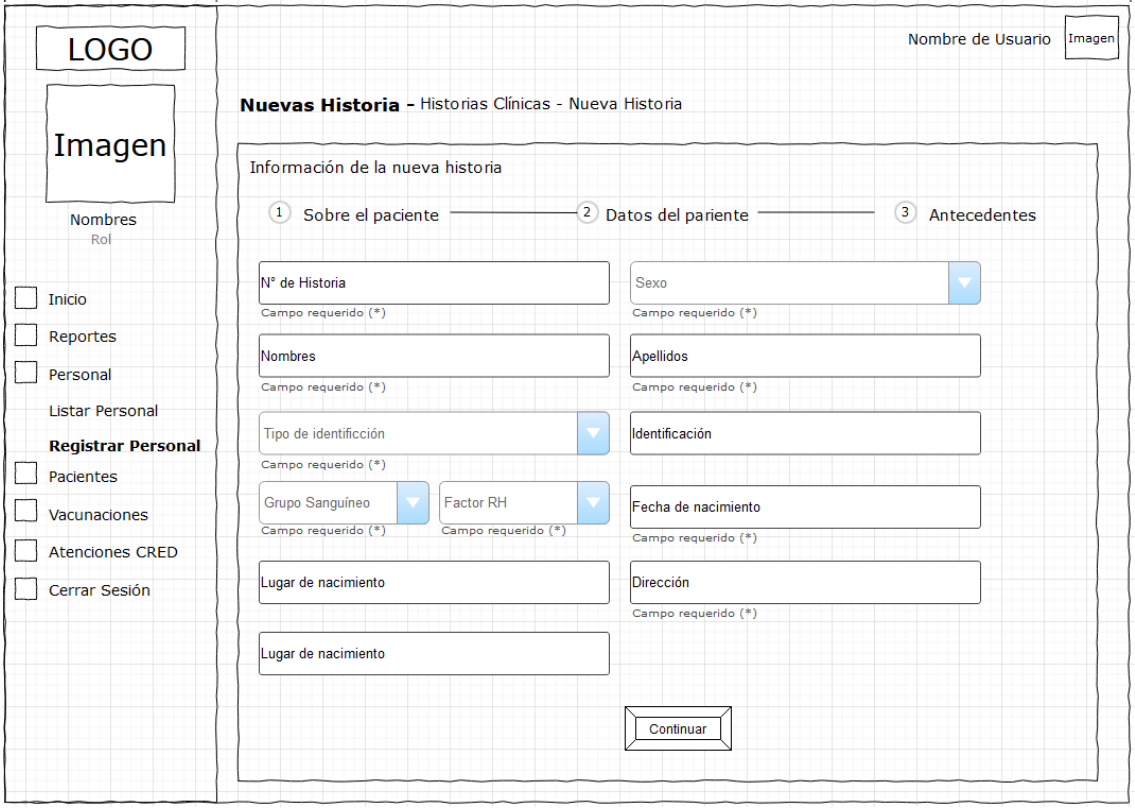
- **Sprint Review**

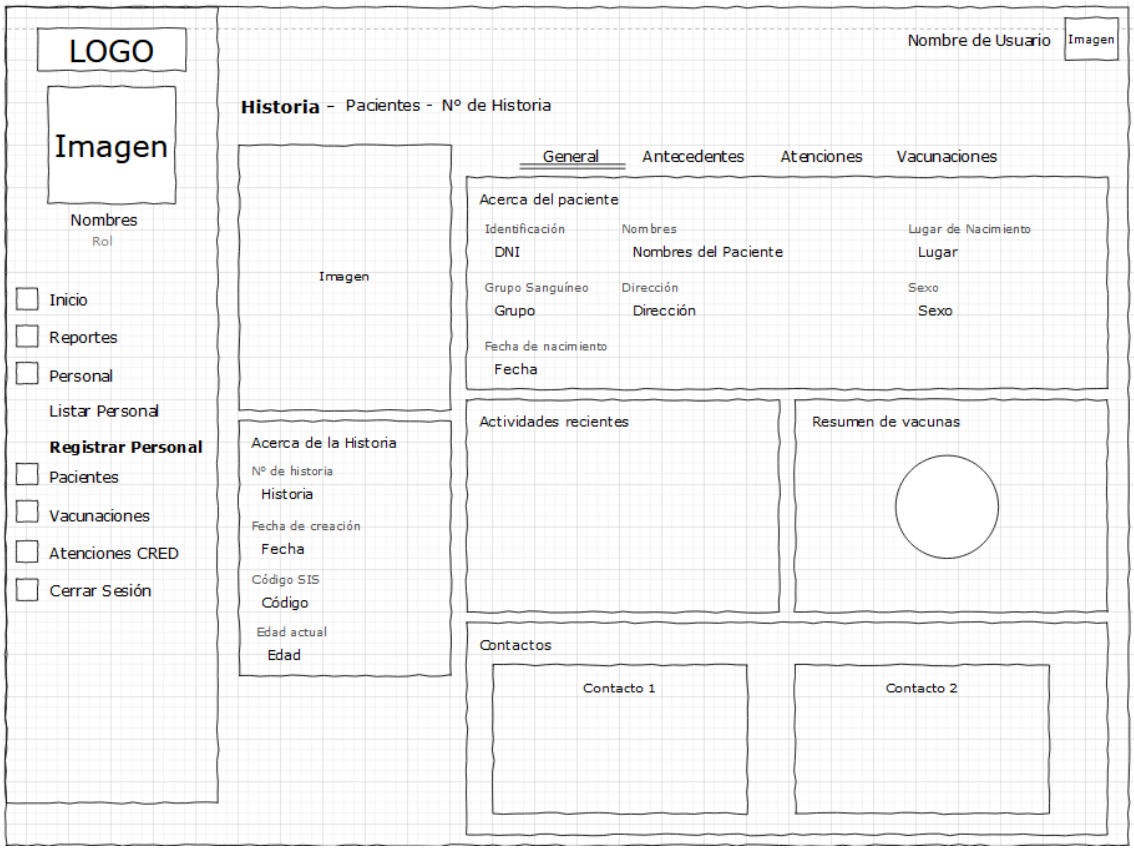
N° Historia	Sprint	Descripción de tarea	Estado	Estimación (dias)
	Sprint 1: Diseño de base de datos	Diseño de base de datos: Identificación de entidades	Concluido	2
		Diseño de base de datos: Diseño del modelo lógico	Concluido	2
		Diseño de base de datos: Diseño del modelo físico	Concluido	1
1 y 2	**Sprint 2:** Módulo de usuarios	Mockups	Concluido	1
		Creación de entidades (Backend)	Concluido	1
		Creación de servicios (Backend)	Concluido	2
		Creación de controladores (Backend)	Concluido	1
		Creación de DTOs y validaciones (Backend)	Concluido	1
		Prueba de APIs	Concluido	1
		Creación de módulos (Frontend)	Concluido	1
		Creación de interfaces de usuario (Frontend)	Concluido	3
		Comunicación entre backend y frontend	Concluido	1
		Sprint Review	Concluido	1
5	**Sprint 3:** Módulo de vacunaciones	Mockups	Concluido	1
		Creación de entidades (Backend)	Concluido	1
		Creación de servicios (Backend)	Concluido	2
		Creación de controladores (Backend)	Concluido	1
		Creación de DTOs y validaciones (Backend)	Concluido	1
		Prueba de APIs	Concluido	1
		Creación de módulos (Frontend)	Concluido	1
		Creación de interfaces de usuario (Frontend)	Concluido	3
		Comunicación entre backend y frontend	Concluido	1
		Sprint Review	Concluido	1
4	**Sprint 4:** Módulo de atenciones	Mockups	Concluido	2
		Creación de entidades (Backend)	Concluido	1
		Creación de servicios (Backend)	Concluido	3
		Creación de controladores (Backend)	Concluido	1
		Creación de DTOs y validaciones (Backend)	Concluido	2
		Prueba de APIs	Concluido	1
		Creación de módulos (Frontend)	Concluido	1
		Creación de interfaces de usuario (Frontend)	Concluido	4
		Comunicación entre backend y frontend	Concluido	1
		Sprint Review	Concluido	1
3, 8	**Sprint 5:** Módulo de pacientes	Mockups	Pendiente	1
		Creación de entidades (Backend)	Pendiente	1
		Creación de servicios (Backend)	Pendiente	2
		Creación de controladores (Backend)	Pendiente	1
		Creación de DTOs y validaciones (Backend)	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	3
		Comunicación entre backend y frontend	Pendiente	1
		Sprint Review	Pendiente	1
6 y 7	**Sprint 6:** Módulo de reportes	Mockups	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	1
		Comunicación entre backend y frontend	Pendiente	1
	Sprint 7: Producción y capacitación	Llenado de la base de datos	Pendiente	1
		Mandar la aplicación a producción	Pendiente	3
		Capacitación de los usuarios finales	Pendiente	1

SPRINT 5: Módulo Pacientes

- **Mockups**

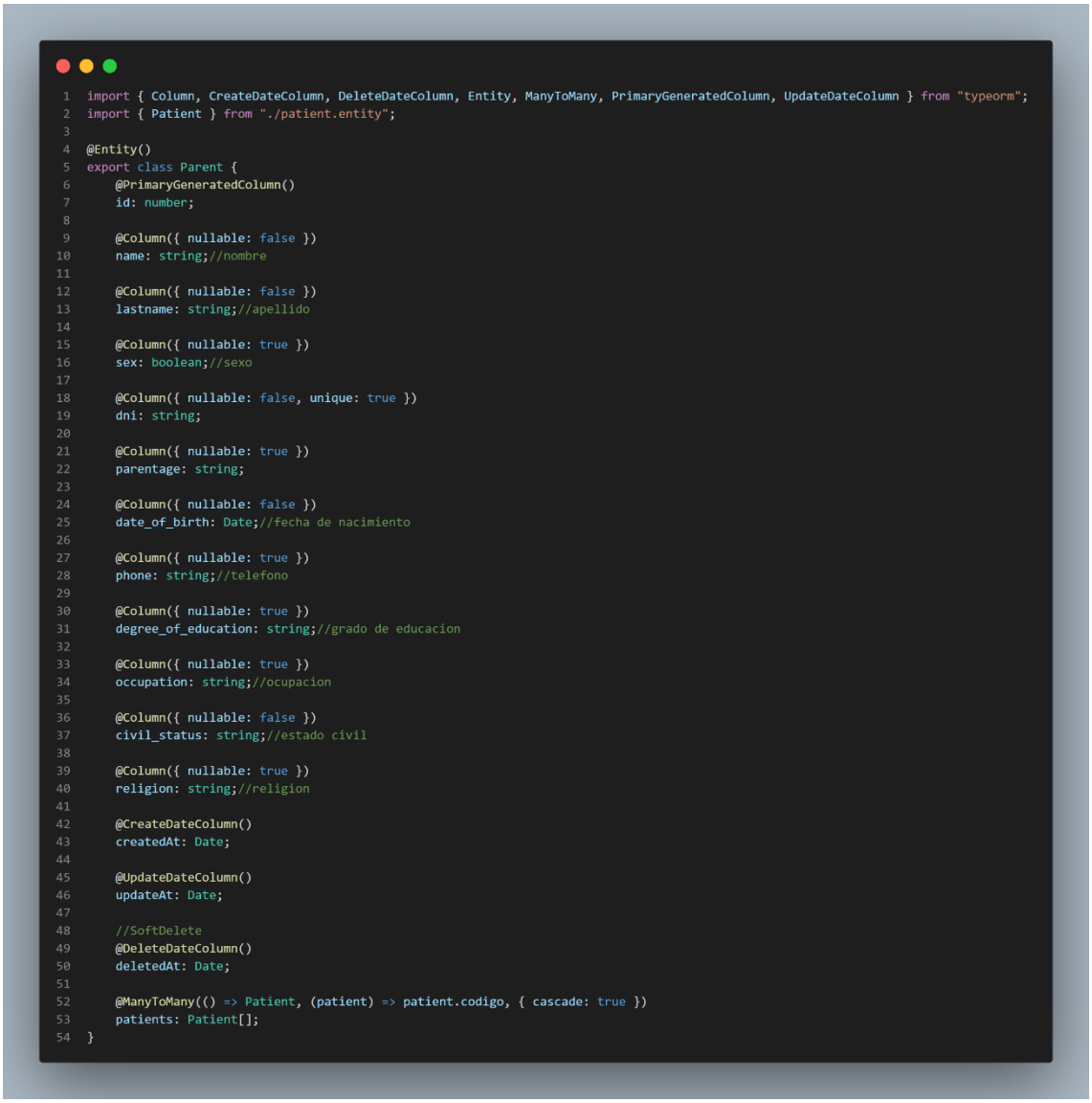





LOGO
Imagen
Nombres
Rol
Inicio
Reportes
Personal
Listar Personal
Registrar Personal
Pacientes
Vacunaciones
Atenciones CRED
Cerrar Sesión
Nombre de Usuario
Imagen
Historia - Pacientes - N° de Historia
Imagen
General
Antecedentes
Atenciones
Vacunaciones
Acerca del paciente
Identificación
DNI
Nombres
Nombres del Paciente
Lugar de Nacimiento
Lugar
Grupo Sanguíneo
Grupo
Dirección
Dirección
Sexo
Sexo
Fecha de nacimiento
Fecha
Acerca de la Historia
N° de historia
Historia
Fecha de creación
Fecha
Código SIS
Código
Edad actual
Edad
Actividades recientes
Resumen de vacunas
Contactos
Contacto 1
Contacto 2

- **Creación de entidades**

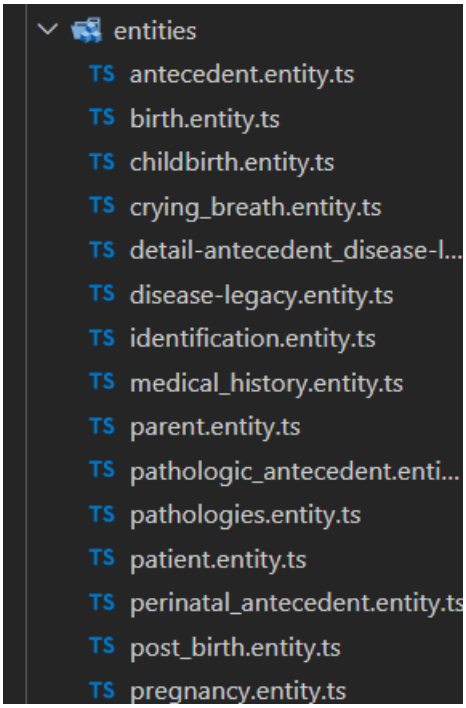
Código fuente que muestra a las entidades de la base de datos pertenecientes al módulo de usuarios; las cuales son: user y role.



```typescript
import { Column, CreateDateColumn, DeleteDateColumn, Entity, ManyToMany, PrimaryGeneratedColumn, UpdateDateColumn } from "typeorm";
import { Patient } from "./patient.entity";

@Entity()
export class Parent {
    @PrimaryGeneratedColumn()
    id: number;

    @Column({ nullable: false })
    name: string;//nombre

    @Column({ nullable: false })
    lastname: string;//apellido

    @Column({ nullable: true })
    sex: boolean;//sexo

    @Column({ nullable: false, unique: true })
    dni: string;

    @Column({ nullable: true })
    parentage: string;

    @Column({ nullable: false })
    date_of_birth: Date;//fecha de nacimiento

    @Column({ nullable: true })
    phone: string;//telefono

    @Column({ nullable: true })
    degree_of_education: string;//grado de educacion

    @Column({ nullable: true })
    occupation: string;//ocupacion

    @Column({ nullable: false })
    civil_status: string;//estado civil

    @Column({ nullable: true })
    religion: string;//religion

    @CreateDateColumn()
    createdAt: Date;

    @UpdateDateColumn()
    updateAt: Date;

    //SoftDelete
    @DeleteDateColumn()
    deletedAt: Date;

    @ManyToMany(() => Patient, (patient) => patient.codigo, { cascade: true })
    patients: Patient[];
}
```

```ts
import { Column, CreateDateColumn, DeleteDateColumn, Entity, JoinColumn, JoinTable, ManyToMany, OneToOne, PrimaryGeneratedColumn, UpdateDateColumn } from "typeorm";
import { Identification } from "./identification.entity";
import { Parent } from "./parent.entity";

@Entity()
export class Patient {
    @PrimaryGeneratedColumn()
    codigo: number;

    @Column({ nullable: false })
    name: string;//nombre

    @Column({ nullable: false })
    lastname: string;//apellido

    @Column({ nullable: false })
    sex: string;//sexo

    @Column({ nullable: true })
    img: string;//imagenes

    @Column({ nullable: false })
    date_of_birth: Date;//fecha de nacimiento

    @Column({ nullable: false })
    place_of_birth: string;//lugar de nacimiento

    @Column({ nullable: false })
    address: string;//direccion

    @Column({ nullable: false })
    blood_group: string;//grupo sanguineo

    @Column({ nullable: true })
    factor_rhesus: string;//factor RH

    @CreateDateColumn()
    createdAt: Date;

    @UpdateDateColumn()
    updateAt: Date;

    //SoftDelete
    @DeleteDateColumn()
    deletedAt: Date;

    @ManyToMany(() => Parent, (parent) => parent.id, { cascade: true })
    @JoinTable()
    parents: Parent[];

    @OneToOne(() => Identification, { cascade: true })
    @JoinColumn()
    identification: Identification
}
```

```ts
import { Column, CreateDateColumn, DeleteDateColumn, Entity, JoinColumn, OneToOne, PrimaryColumn, UpdateDateColumn } from "typeorm";
import { Antecedent } from "./antecedent.entity";
import { Patient } from "./patient.entity";

@Entity()
export class MedicalHistory {
    @PrimaryColumn()
    num_hc: string;//numero de historia

    @Column({ nullable: true })
    cod_sis_other: number;//codigo sis

    @CreateDateColumn()
    createdAt: Date;

    @UpdateDateColumn()
    updateAt: Date;

    //SoftDelete
    @DeleteDateColumn()
    deletedAt: Date;

    @OneToOne(() => Patient, { cascade: true })
    @JoinColumn()
    patient: Patient

    @OneToOne(() => Antecedent, { cascade: true })
    @JoinColumn()
    antecedent: Antecedent
}
```


entities
antecedent.entity.ts
birth.entity.ts
childbirth.entity.ts
crying_breath.entity.ts
detail-antecedent_disease-l...
disease-legacy.entity.ts
identification.entity.ts
medical_history.entity.ts
parent.entity.ts
pathologic_antecedent.enti...
pathologies.entity.ts
patient.entity.ts
perinatal_antecedent.entity.ts
post_birth.entity.ts
pregnancy.entity.ts

- **Creación de servicios**
  Código fuente que muestra a los servicios utilizados para consultar a las entidades mencionadas anteriormente.

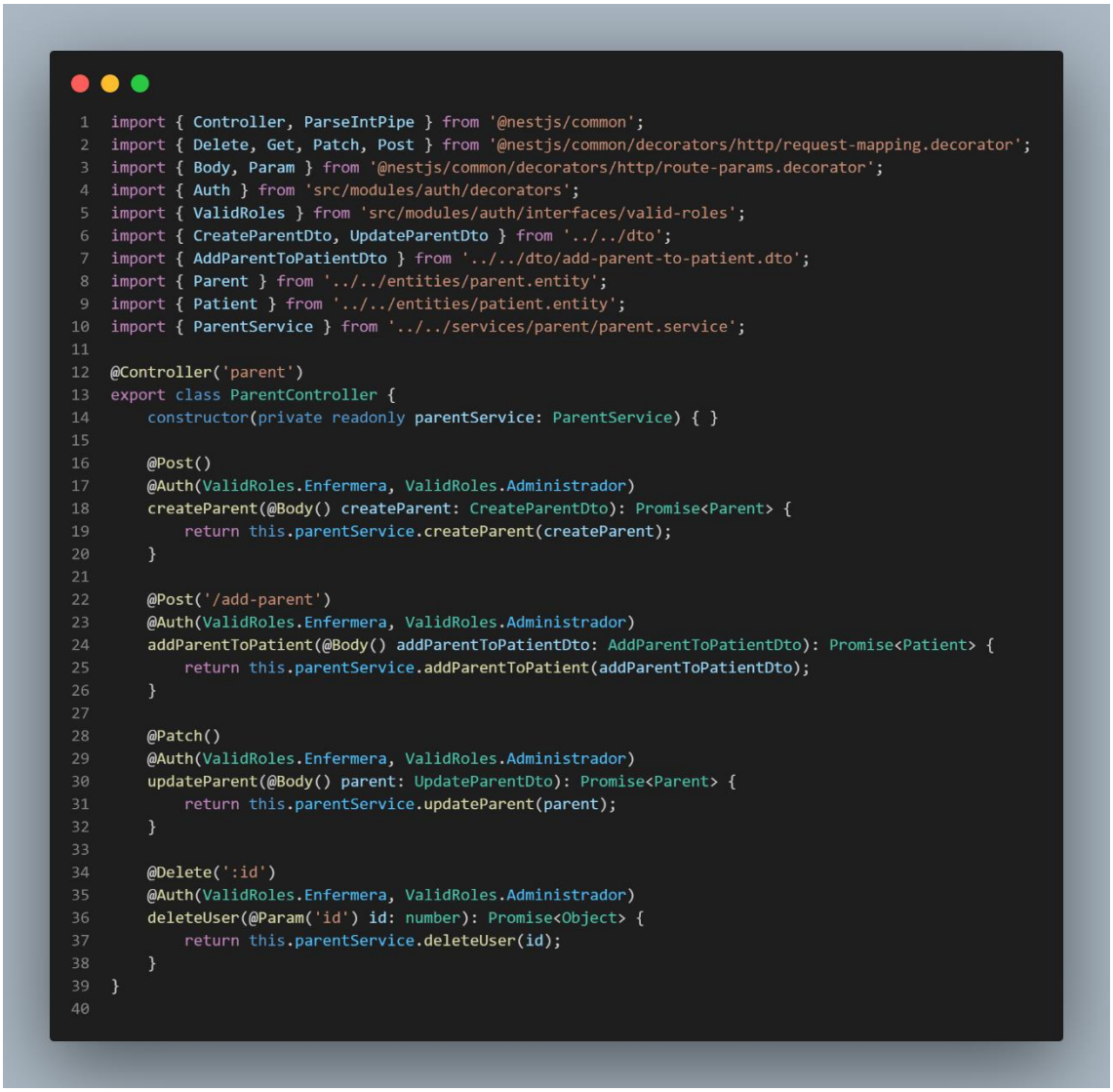
```ts
import { HttpException, HttpStatus, Injectable, NotFoundException } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { DiseaseLegacy } from '../../entities/disease-legacy.entity';
import { Pathologies } from '../../entities/pathologies.entity';

@Injectable()
export class DiseaseLegacyService {
    constructor(
        @InjectRepository(DiseaseLegacy) private readonly diseaseLegacyRepository: Repository<DiseaseLegacy>,
    ) { }

    async getDiseasesLegacy(): Promise<DiseaseLegacy[]> {
        const diseases = await this.diseaseLegacyRepository.find({
            select: {
                id: true, name: true
            },
            order: {
                name: 'asc',
            },
        });

        //Mensaje para cuando salga mal la consulta
        if (!diseases) throw new NotFoundException("Algo salió mal.");

        //Mensaje para cuando no se encuentren registros
        if (diseases && diseases.length == 0) throw new HttpException({
            status: HttpStatus.NOT_FOUND,
            error: 'No hay enfermedades hereditarias para mostrar.',
        }, HttpStatus.NOT_FOUND)

        return diseases;
    }
}

```

```ts
import { BadRequestException, HttpException, HttpStatus, Injectable, InternalServerErrorException, NotFoundException } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { Pathologies } from '../../entities/pathologies.entity';

@Injectable()
export class PathologiesService {
    constructor(
        @InjectRepository(Pathologies) private readonly pathologiesRepository: Repository<Pathologies>,
    ) { }

    async getPathologies(): Promise<Pathologies[]> {
        const pathologies = await this.pathologiesRepository.find({
            select: {
                id: true, name: true, category: true
            },
            order: {
                category: 'asc',
                name: 'asc',
            },
        });

        //Mensaje para cuando salga mal la consulta
        if (!pathologies) throw new NotFoundException("Algo salió mal.");

        //Mensaje para cuando no se encuentren registros
        if (pathologies && pathologies.length == 0) throw new HttpException({
            status: HttpStatus.NOT_FOUND,
            error: 'No hay patologías para mostrar.',
        }, HttpStatus.NOT_FOUND)

        return pathologies;
    }
}

```

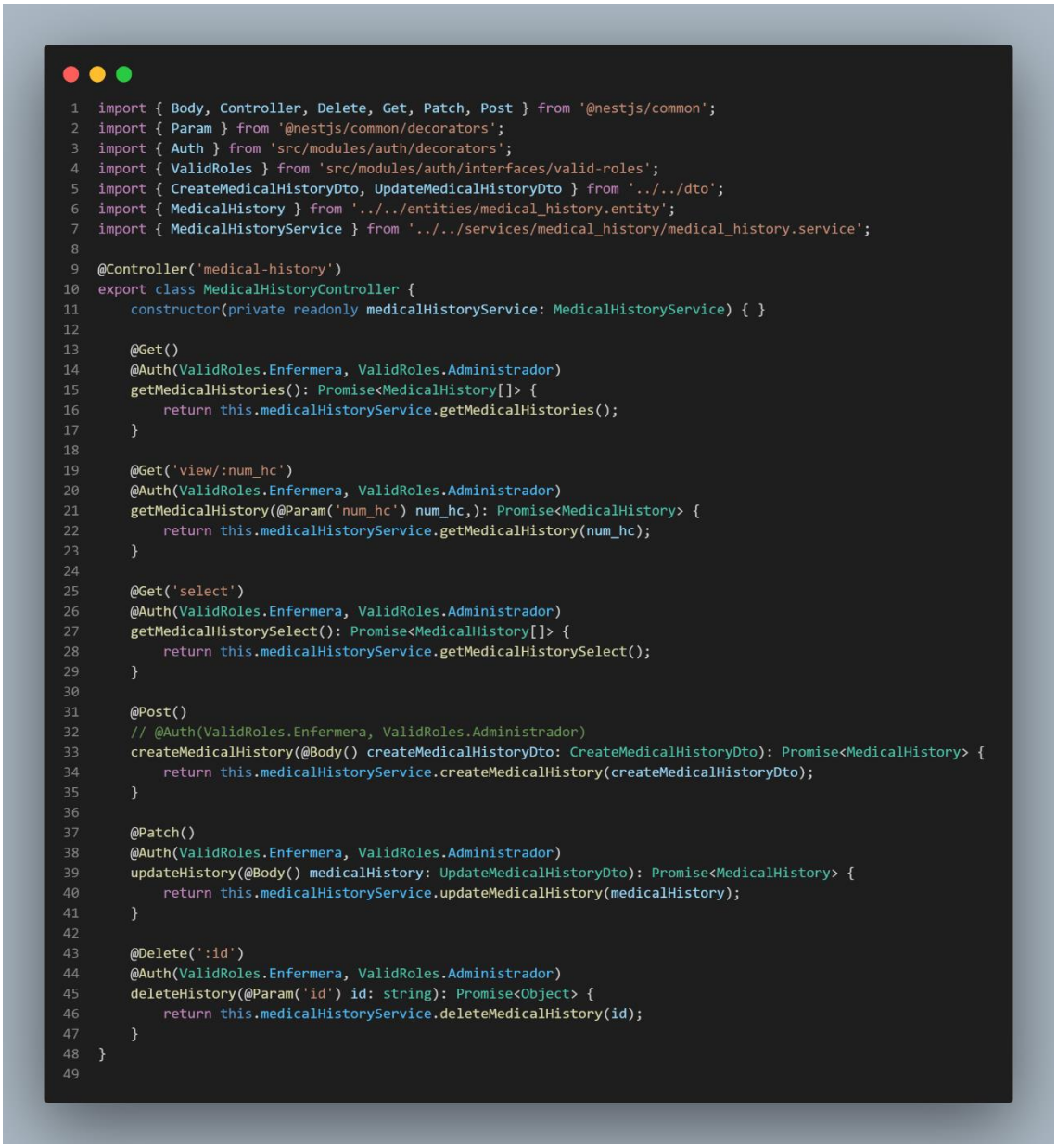
```ts
import { BadRequestException, HttpException, HttpStatus, Injectable, InternalServerErrorException, NotFoundException } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { CreateParentDto, UpdateParentDto } from '../../dto';
import { AddParentToPatientDto } from '../../dto/add-parent-to-patient.dto';
import { MedicalHistory } from '../../entities/medical_history.entity';
import { Parent } from '../../entities/parent.entity';
import { Patient } from '../../entities/patient.entity';

@Injectable()
export class ParentService {
    constructor(
        @InjectRepository(MedicalHistory) private readonly medicalHistoryRepository: Repository<MedicalHistory>,
        @InjectRepository(Parent) private readonly parentRepository: Repository<Parent>,
        @InjectRepository(Patient) private readonly patientRepository: Repository<Patient>,
    ) { }

    async createParent({ num_hc, ...dto }: CreateParentDto): Promise<Parent> {
        const parent = this.parentRepository.create(dto);

        const saveParent = await this.parentRepository.save(parent).catch((e) => { this.handleDBErrors(e) });
        if (num_hc) {
            this.addParentToPatient({ number_hc: num_hc, id_parent: saveParent.id })
        }
        return saveParent;
    }

    async addParentToPatient({ number_hc, id_parent }: AddParentToPatientDto): Promise<Patient> {
        const medical_history = await this.medicalHistoryRepository.findOne({
            where: { num_hc: number_hc },
            relations: { patient: { parents: true } }
        });

        if (!medical_history) {
            throw new HttpException({
                status: HttpStatus.NOT_FOUND,
                message: ['La historia médica no existe.'],
            }, HttpStatus.NOT_FOUND)
        }

        const parent = await this.parentRepository.findOne({
            where: { id: id_parent },
        });

        if (!parent) {
            throw new HttpException({
                status: HttpStatus.NOT_FOUND,
                message: ['El pariente no existe.'],
            }, HttpStatus.NOT_FOUND)
        }

        const patient = await this.patientRepository.preload({ codigo: medical_history.patient.codigo, parents: [...medical_history.patient.parents, parent] });

        if (!patient) {
            throw new HttpException({
                status: HttpStatus.NOT_FOUND,
                message: ['El paciente no existe.'],
            }, HttpStatus.NOT_FOUND)
        }

        console.log(patient)

        return await this.patientRepository.save(patient);
    }

    async updateParent({ id_parent, ...data }: UpdateParentDto): Promise<Parent> {
        const parent: Parent = await this.parentRepository.preload({
            id: id_parent, ...data
        });

        if (!parent) {
            throw new NotFoundException('Pariente no encontrado.')
        }

        return this.parentRepository.save(parent).catch((e) => {
            const errors = [];
            if (/(identification)[\s\S]+(already exists)/.test(e.detail)) errors.push('La identificación ingresada ya pertenece a un pariente.')

            if (errors.length > 0) throw new BadRequestException(errors);

            return e;
        });
    }


    async deleteUser(id: number): Promise<Object> {
        const deleteResponse = await this.parentRepository.delete(id);
        if (!deleteResponse.affected) {
            throw new NotFoundException('Pariente no encontrado.');
        }

        return {
            status: HttpStatus.ACCEPTED,
            message: 'Pariente eliminado exitosamente.',
        }
    }

    private handleDBErrors(error: any): never {
        if (error.code === '23505')
            throw new BadRequestException(error.detail);

        console.log(error)

        throw new InternalServerErrorException('Please check server logs');

    }
}

```

- **Creación de controladores**

Código fuente que contiene las APIs a las cuales se debe consultar para llamar a un determinado servicio.



```
import { Controller, ParseIntPipe } from '@nestjs/common';
import { Delete, Get, Patch, Post } from '@nestjs/common/decorators/http/request-mapping.decorator';
import { Body, Param } from '@nestjs/common/decorators/http/route-params.decorator';
import { Auth } from 'src/modules/auth/decorators';
import { ValidRoles } from 'src/modules/auth/interfaces/valid-roles';
import { CreateParentDto, UpdateParentDto } from '../../dto';
import { AddParentToPatientDto } from '../../dto/add-parent-to-patient.dto';
import { Parent } from '../../entities/parent.entity';
import { Patient } from '../../entities/patient.entity';
import { ParentService } from '../../services/parent/parent.service';

@Controller('parent')
export class ParentController {
    constructor(private readonly parentService: ParentService) { }

    @Post()
    @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    createParent(@Body() createParent: CreateParentDto): Promise<Parent> {
        return this.parentService.createParent(createParent);
    }

    @Post('/add-parent')
    @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    addParentToPatient(@Body() addParentToPatientDto: AddParentToPatientDto): Promise<Patient> {
        return this.parentService.addParentToPatient(addParentToPatientDto);
    }

    @Patch()
    @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    updateParent(@Body() parent: UpdateParentDto): Promise<Parent> {
        return this.parentService.updateParent(parent);
    }

    @Delete(':id')
    @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    deleteUser(@Param('id') id: number): Promise<Object> {
        return this.parentService.deleteUser(id);
    }
}
```



```typescript
import { Body, Controller, Delete, Get, Patch, Post } from '@nestjs/common';
import { Param } from '@nestjs/common/decorators';
import { Auth } from 'src/modules/auth/decorators';
import { ValidRoles } from 'src/modules/auth/interfaces/valid-roles';
import { CreateMedicalHistoryDto, UpdateMedicalHistoryDto } from '../../dto';
import { MedicalHistory } from '../../entities/medical_history.entity';
import { MedicalHistoryService } from '../../services/medical_history/medical_history.service';

@Controller('medical-history')
export class MedicalHistoryController {
    constructor(private readonly medicalHistoryService: MedicalHistoryService) { }

    @Get()
    @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    getMedicalHistories(): Promise<MedicalHistory[]> {
        return this.medicalHistoryService.getMedicalHistories();
    }

    @Get('view/:num_hc')
    @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    getMedicalHistory(@Param('num_hc') num_hc,): Promise<MedicalHistory> {
        return this.medicalHistoryService.getMedicalHistory(num_hc);
    }

    @Get('select')
    @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    getMedicalHistorySelect(): Promise<MedicalHistory[]> {
        return this.medicalHistoryService.getMedicalHistorySelect();
    }

    @Post()
    // @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    createMedicalHistory(@Body() createMedicalHistoryDto: CreateMedicalHistoryDto): Promise<MedicalHistory> {
        return this.medicalHistoryService.createMedicalHistory(createMedicalHistoryDto);
    }

    @Patch()
    @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    updateHistory(@Body() medicalHistory: UpdateMedicalHistoryDto): Promise<MedicalHistory> {
        return this.medicalHistoryService.updateMedicalHistory(medicalHistory);
    }

    @Delete(':id')
    @Auth(ValidRoles.Enfermera, ValidRoles.Administrador)
    deleteHistory(@Param('id') id: string): Promise<Object> {
        return this.medicalHistoryService.deleteMedicalHistory(id);
    }
}

```

- **Creación de DTOs y validaciones**
  Objeto de transferencia de datos que se utiliza al momento de enviar datos al backend para los diferentes casos de usos. Además, las validaciones personalizadas utilizadas para verificar si el ID de un tipo de usuario existe.

```typescript
import { IsAlpha, IsAlphanumeric, IsDate, IsNotEmpty, IsNumber, IsNumberString, IsOptional, Length, Matches, MaxLength, MinLength } from "class-validator";
import { CodSisExists } from "./custom-validations/cod-sis-exists.validate";
import { IdentificationExists } from "./custom-validations/identification-exists.validate";
import { NumHCExists } from "./custom-validations/num-hc-exists.validate";

export class CreateMedicalHistoryDto {
    //Validando el campo numero de historia
    @Matches(RegExp("^[A-Za-zñÑáéíóúÁÉÍÓÚ/.-0-9 ]"), { message: 'El número de historia solo puede contener letras, números y algunos caracteres especiales.' })
    @NumHCExists(true)
    readonly num_hc: string;

    //validando codigo de sis
    @IsNumber({}, {
        message: 'El código SIS debe ser numérico.'
    })
    @CodSisExists(true)
    @IsOptional()
    readonly cod_sis_other: number; //codigo SIS

    //validando nombre
    @MinLength(3, {
        message: 'El campo nombre debe contener 3 caracteres como mínimo.',
    })
    @MaxLength(60, {
        message: 'El campo nombre solo puede contener 60 caracteres como máximo.',
    })
    @Matches(RegExp("^[A-Za-zıöüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ/. ]"), { message: 'El campo nombre solo puede contener letras.' })
    @IsNotEmpty({
        message: 'El campo nombre es requerido.',
    })
    readonly name: string;

    //validando apellidos
    @MinLength(6, {
        message: 'El campo apellido debe contener 3 caracteres como mínimo.',
    })
    @MaxLength(60, {
        message: 'El campo apellido solo puede contener 60 caracteres como máximo.',
    })
    @Matches(RegExp("^[A-Za-zıöüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ/. ]"), { message: 'El campo apellido solo puede contener letras.' })
    @IsNotEmpty({
        message: 'El campo apellido es requerido.',
    })
    readonly lastname: string;

    //validando sexo
    @MaxLength(9, {
        message: 'El campo sexo solo puede contener 9 caracteres como máximo.',
    })
    @Matches(RegExp("^[A-Za-z/. ]"), { message: 'El campo sexo solo puede contener letras.' })
    @IsNotEmpty({
        message: 'El campo sexo es requerido.',
    })
    readonly sex: string;

    //validando lugar de nacimiento
    @MaxLength(15, {
        message: 'El campo lugar de nacimiento solo puede contener 15 caracteres como máximo.',
    })
    @Matches(RegExp("^[A-Za-záéíóú0-9-/._ ]"), { message: 'El campo lugar de nacimiento solo puede contener letras, números y algunos caracteres especiales.' })
    @IsNotEmpty({
        message: 'El campo lugar de nacimiento es requerido.',
    })
    readonly place_of_birth: string;

    //validando direccion
    @MaxLength(100, {
        message: 'La dirección solo puede contener 10 caracteres como máximo.',
    })
    @Matches(RegExp('^[A-Za-zıöüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ0-9-/,; ]+$'), { message: 'La dirección solo puede contener letras' })
    @IsNotEmpty({
        message: 'La dirección es requerida.',
    })
    readonly address: string;

    //validando grupo sanguineo
    @MinLength(1, {
        message: 'El campo grupo sanguineo debe contener 1 caracteres como mínimo.',
    })
    @MaxLength(2, {
        message: 'El campo grupo sanguineo solo puede contener 2 caracteres como máximo.',
    })
    @IsAlpha('es-ES', {
        message: 'El campo grupo sanguineo solo puede contener letras.',
    })
    @IsNotEmpty({
        message: 'El campo grupo sanguineo es requerido.',
    })
    readonly blood_group: string;

    //validando factor RH
    @Length(1, 1, {
        message: 'El campo factor rh solo puede contener 1 caracter como máximo.',
    })
    @Matches(RegExp("^[+-/]"), { message: 'El campo factor rh solo puede contener "+" o "-"' })
    @IsNotEmpty({
        message: 'El campo factor rh es requerido.',
    })
    readonly factor_rhesus: string;

    //Validando el campo para fecha de nacimiento
    @IsDate({
        message: 'Ingresar una fecha de nacimiento válida.'
    })
    @IsNotEmpty({
        message: 'La fecha de nacimiento es requerida.',
    })
    readonly date_of_birth: Date;

    //validando valor de identificacion
    @MinLength(8, {
        message: 'La identificación debe contener 8 caracteres como mínimo',
    })
    @IsNumberString({}, {
        message: 'La identificación es inválida.'
    })
    @IdentificationExists(true)
    @IsOptional({
        message: 'La identificación es opcional',
    })
    readonly value_identification: string;

    //Validando el campo tipo de identificción
    @Matches(RegExp('^[A-Za-zıöüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ ]+$'), { message: 'El tipo de identificación debe contener solo letras' })
    @IsNotEmpty({
        message: 'El tipo de identificación es requerido',
    })
    readonly type_identification: string;
}
```

```ts
import { IsAlpha, IsDate, IsNotEmpty, IsNumber, IsNumberString, IsOptional, Length, Matches, MaxLength, MinLength } from "class-validator";
import { NumHCExists } from "./custom-validations/num-hc-exists.validate";

export class UpdateMedicalHistoryDto {
    //Validando el campo numero de historia
    @Matches(RegExp("^[A-Za-zñÑáéíóúÁÉÍÓÚ/. 0-9 ]"), { message: 'El número de historia solo puede contener letras, números y algunos caracteres especiales.' })
    @NumHCExists(false)
    readonly num_hc: string;

    //validando codigo de sis
    @IsNumber({}, {
        message: 'El código SIS debe ser numérico.'
    })
    @IsOptional()
    readonly cod_sis_other: number; //codigo SIS

    //validando nombre
    @MinLength(3, {
        message: 'El campo nombre debe contener 3 caracteres como mínimo.',
    })
    @MaxLength(60, {
        message: 'El campo nombre solo puede contener 60 caracteres como máximo.',
    })
    @Matches(RegExp("^[A-Za-z1öüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ/. ]"), { message: 'El campo nombre solo puede contener letras.' })
    @IsNotEmpty({
        message: 'El campo nombre es requerido.',
    })
    readonly name: string;

    //validando apellidos
    @MinLength(6, {
        message: 'El campo apellido debe contener 3 caracteres como mínimo.',
    })
    @MaxLength(60, {
        message: 'El campo apellido solo puede contener 60 caracteres como máximo.',
    })
    @Matches(RegExp("^[A-Za-z1öüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ/. ]"), { message: 'El campo apellido solo puede contener letras.' })
    @IsNotEmpty({
        message: 'El campo apellido es requerido.',
    })
    readonly lastname: string;

    //validando sexo
    @MaxLength(9, {
        message: 'El campo sexo solo puede contener 9 caracteres como máximo.',
    })
    @Matches(RegExp("^[A-Za-z/. ]"), { message: 'El campo sexo solo puede contener letras.' })
    @IsNotEmpty({
        message: 'El campo sexo es requerido.',
    })
    readonly sex: string;

    //validando lugar de nacimiento
    @MaxLength(15, {
        message: 'El campo lugar de nacimiento solo puede contener 15 caracteres como máximo.',
    })
    @Matches(RegExp("^[A-Za-z/. ]"), { message: 'El campo lugar de nacimiento solo puede contener letras.' })
    @IsNotEmpty({
        message: 'El campo lugar de nacimiento es requerido.',
    })
    readonly place_of_birth: string;

    //validando direccion
    @MaxLength(100, {
        message: 'La dirección solo puede contener 10 caracteres como máximo.',
    })
    @Matches(RegExp('^[A-Za-z1öüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ0-9-/,; ]+$'), { message: 'La dirección solo puede contener letras' })
    @IsNotEmpty({
        message: 'La dirección es requerida.',
    })
    readonly address: string;

    //validando grupo sanguineo
    @MinLength(1, {
        message: 'El campo grupo sanguíneo debe contener 1 caracteres como mínimo.',
    })
    @MaxLength(2, {
        message: 'El campo grupo sanguíneo solo puede contener 2 caracteres como máximo.',
    })
    @IsAlpha('es-ES', {
        message: 'El campo grupo sanguíneo solo puede contener letras.',
    })
    @IsNotEmpty({
        message: 'El campo grupo sanguíneo es requerido.',
    })
    readonly blood_group: string;

    //validando factor RH
    @Length(1, 1, {
        message: 'El campo factor rh solo puede contener 1 caracter como máximo.',
    })
    @Matches(RegExp("^[+-/]"), { message: 'El campo factor rh solo puede contener "+" o "-"' })
    @IsNotEmpty({
        message: 'El campo factor rh es requerido.',
    })
    readonly factor_rhesus: string;

    //Validando el campo para fecha de nacimiento
    @IsDate({
        message: 'Ingresar una fecha de nacimiento válida.'
    })
    @IsNotEmpty({
        message: 'La fecha de nacimiento es requerida.',
    })
    readonly date_of_birth: Date;

    //validando valor de identificacion
    @MinLength(8, {
        message: 'La identificación debe contener 8 caracteres como mínimo',
    })
    @IsNumberString({}, {
        message: 'La identificación es invalida.'
    })
    @IsOptional({
        message: 'La identificación es opcional',
    })
    readonly value_identification: string;

    //Validando el campo tipo de identificcion
    @Matches(RegExp('^[A-Za-z1öüçğşİÖÜÇĞŞñÑáéíóúÁÉÍÓÚ ]+$'), { message: 'El tipo de identificación debe contener solo letras' })
    @IsNotEmpty({
        message: 'El tipo de identificación es requerido',
    })
    readonly type_identification: string;
}
```

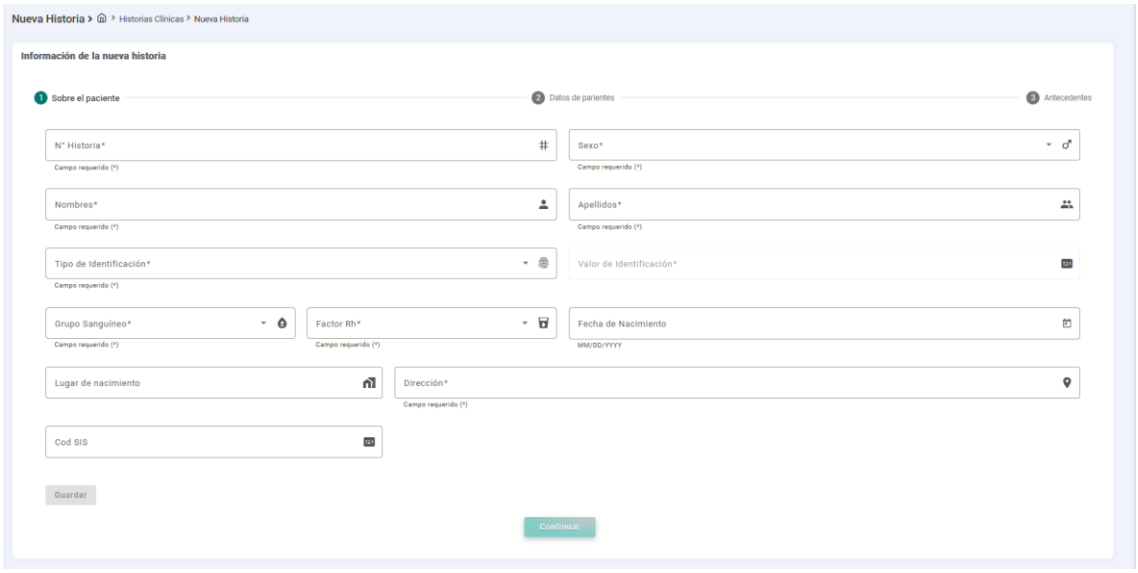
- **Creación de módulos**

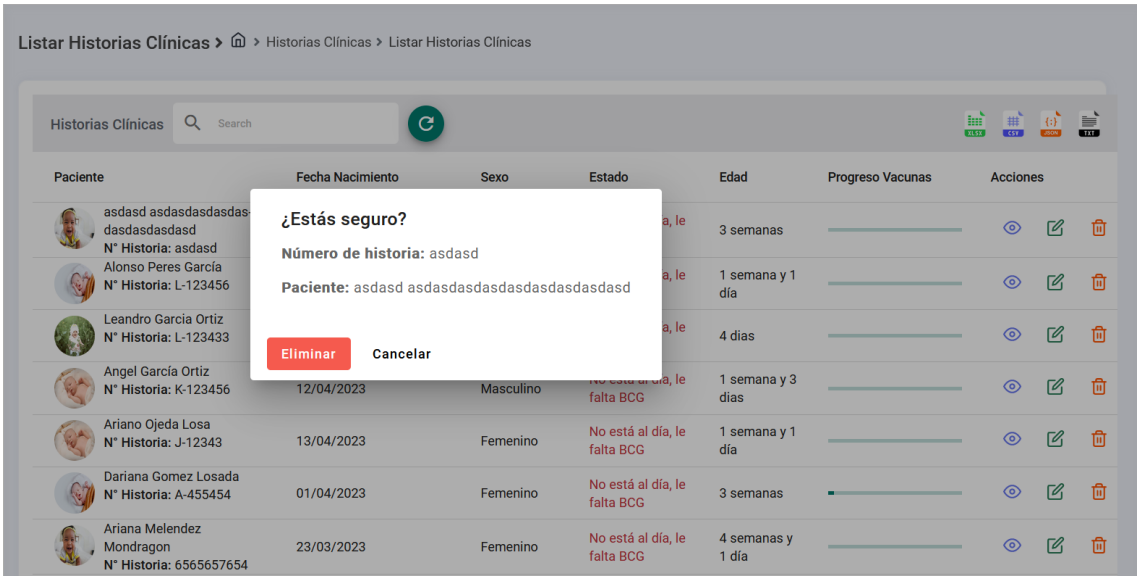
Creación del módulo en el frontend que permite administrar todas las vistas, importaciones y módulos externos correspondientes a este.

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { PatientsComponent } from './patients.component';
import { PatientsRoutingModule } from './patients-routing.module';
import { AllMedicalHistoriesComponent } from './pages/all-medical-histories/all-medical-histories.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { MatPaginatorModule } from '@angular/material/paginator';
import { MatTableModule } from '@angular/material/table';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import { MatButtonModule } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';
import { MatDialogModule } from '@angular/material/dialog';
import { MatSortModule } from '@angular/material/sort';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatSelectModule } from '@angular/material/select';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatTabsModule } from '@angular/material/tabs';
import { MatTooltipModule } from '@angular/material/tooltip';
import { MatTableExporterModule } from 'mat-table-exporter';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { MatStepperModule } from '@angular/material/stepper';
import { StaffRoutingModule } from '../staff/staff-routing.module';
import { ComponentsModule } from 'src/app/shared/components/components.module';
import { SharedModule } from 'src/app/shared/shared.module';
import { StoreModule } from '@ngrx/store';
import { EffectsModule } from '@ngrx/effects';
import { historiesReducer } from './state/reducers/medical-history.reducers';
import { HistoriesEffects } from './state/effects/medical-history.effects';
import { MatChipsModule } from '@angular/material/chips';
import { ViewMedicalHistoryComponent } from './pages/view-medical-history/view-medical-history.component';
import { NgApexchartsModule } from 'ng-apexcharts';
import { NgxSkeletonLoaderModule } from 'ngx-skeleton-loader';
import { TableVaccinationsOfHistoryComponent } from './pages/view-medical-history/components/molecules/table-vaccinations-of-history/table-vaccinations-of-history.component';
import { TableAttentionsOfHistoryComponent } from './pages/view-medical-history/components/molecules/table-attentions-of-history/table-attentions-of-history.component';
import { AddMedicalHistoryComponent } from './pages/add-medical-history/add-medical-history.component';
import { UpdateMedicalHistoryComponent } from './pages/update-medical-history/update-medical-history.component';
import { UpdateFormAntecedentComponent } from './pages/update-medical-history/components/update-form-antecedent/update-form-antecedent.component';
import { DeleteDialogComponent } from './pages/all-medical-histories/dialog/delete/delete.component';

@NgModule({
  imports: [
    PatientsRoutingModule,
    CommonModule,
    FormsModule,
    ReactiveFormsModule,
    MatTableModule,
    MatPaginatorModule,
    MatFormFieldModule,
    MatInputModule,
    MatSnackBarModule,
    MatButtonModule,
    MatIconModule,
    MatDialogModule,
    MatStepperModule,
    MatSortModule,
    MatToolbarModule,
    MatSelectModule,
    MatCheckboxModule,
    MatDatepickerModule,
    MatTabsModule,
    MatChipsModule,
    MatTooltipModule,
    MatTableExporterModule,
    MatProgressSpinnerModule,
    NgApexchartsModule,
    StaffRoutingModule,
    ComponentsModule,
    SharedModule,
    NgxSkeletonLoaderModule,
    StoreModule.forFeature('historias', historiesReducer),
    EffectsModule.forFeature([HistoriesEffects]),
  ],
  declarations: [
    PatientsComponent,
    AllMedicalHistoriesComponent,
    ViewMedicalHistoryComponent,
    TableVaccinationsOfHistoryComponent,
    TableAttentionsOfHistoryComponent,
    AddMedicalHistoryComponent,
    UpdateMedicalHistoryComponent,
    UpdateFormAntecedentComponent,
    DeleteDialogComponent
  ]
})
export class PatientsModule { }
```

- **Creación de interfaces de usuario**

Creación de interfaces de usuario del Frontend.





Listar Historias Clínicas > Historias Clínicas > Listar Historias Clínicas

Historias Clínicas

Search

Paciente	Fecha Nacimiento	Sexo	Estado	Edad	Progreso Vacunas	Acciones
 Alonso Peres García **N° Historia:** L-123456	13/04/2023	Masculino	No está al día, le falta BCG	1 semana y 1 día		
 Leandro Garcia Ortiz **N° Historia:** L-123433	18/04/2023	Masculino	No está al día, le falta BCG	4 dias		
 Angel García Ortiz **N° Historia:** K-123456	12/04/2023	Masculino	No está al día, le falta BCG	1 semana y 3 dias		
 Ariano Ojeda Losa **N° Historia:** J-12343	13/04/2023	Femenino	No está al día, le falta BCG	1 semana y 1 día		
 Dariana Gomez Losada **N° Historia:** A-455454	01/04/2023	Femenino	No está al día, le falta BCG	3 semanas		
 Ariana Melendez Mondragon **N° Historia:** 6565657654	23/03/2023	Femenino	No está al día, le falta BCG	4 semanas y 1 día		
 Alonsoza Ortega Dias **N° Historia:** 6565656511	15/03/2023	Masculino	No está al día, le falta BCG	1 mes y 1 se-mana		

- **Comunicación entre backend y frontend**
  Código fuente que muestra los métodos utilizados para consultar al backend desde el frontend.

```typescript
import { Injectable } from '@angular/core';
import { HttpClient, HttpErrorResponse, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';
import { delay } from 'rxjs/operators';
import { Global } from 'src/app/config/global_url';
import { CreateMedicalHistoryDto, UpdateMedicalHistoryDto } from '../dto';

@Injectable({
  providedIn: 'root'
})
export class MedicalHistoryService {

  public url: string;
  public headers: HttpHeaders;

  constructor(
    private _http: HttpClient,
  ) {
    this.url = Global.url;
  }


  /** CRUD METHODS */
  getMedicalHistories(): Observable<any> {
    return this._http.get(this.url + 'medical-history', { headers: this.headers }).pipe(delay(1000));
  }

  getMedicalHistoriesSelect(): Observable<any> {
    return this._http.get(this.url + 'medical-history/select', { headers: this.headers }).pipe(delay(2000));
  }

  getMedicalHistory(num_hc: string): Observable<any> {
    return this._http.get(this.url + 'medical-history/view/' + num_hc, { headers: this.headers }).pipe(delay(1000));
  }

  saveMedicalHistory(dto: CreateMedicalHistoryDto): Observable<any> {
    return this._http.post(this.url + 'medical-history', dto).pipe(delay(2000));
  }

  deleteHistory(id: any): Observable<any> {
    return this._http.delete(this.url + 'medical-history/' + id, { headers: this.headers }).pipe(delay(2000));
  }

  updateHistory(dto: UpdateMedicalHistoryDto): Observable<any> {
    return this._http.patch(this.url + 'medical-history/', dto, { headers: this.headers }).pipe(delay(2000));
  }

}

```

```typescript
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';
import { delay } from 'rxjs/operators';
import { Global } from 'src/app/config/global_url';
import { AddParentToPatientDto, CreateParentDto, UpdateParentDto } from '../dto';

@Injectable({
  providedIn: 'root'
})
export class ParentService {

  public url: string;
  public headers: HttpHeaders;

  constructor(
    private _http: HttpClient,
  ) {
    this.url = Global.url;
  }

  getParents(): Observable<any> {
    return this._http.get(this.url + 'parent', { headers: this.headers }).pipe(delay(1000));
  }

  saveParent(dto: CreateParentDto): Observable<any> {
    return this._http.post(this.url + 'parent', dto).pipe(delay(2000));
  }

  addParentToPatient(dto: AddParentToPatientDto): Observable<any> {
    return this._http.post(this.url + 'parent/add-parent', dto).pipe(delay(2000));
  }

  deleteParent(id: any): Observable<any> {
    return this._http.delete(this.url + 'parent/' + id, { headers: this.headers }).pipe(delay(2000));
  }

  updateParent(dto: UpdateParentDto): Observable<any> {
    return this._http.patch(this.url + 'parent/', dto, { headers: this.headers }).pipe(delay(2000));
  }
}

```

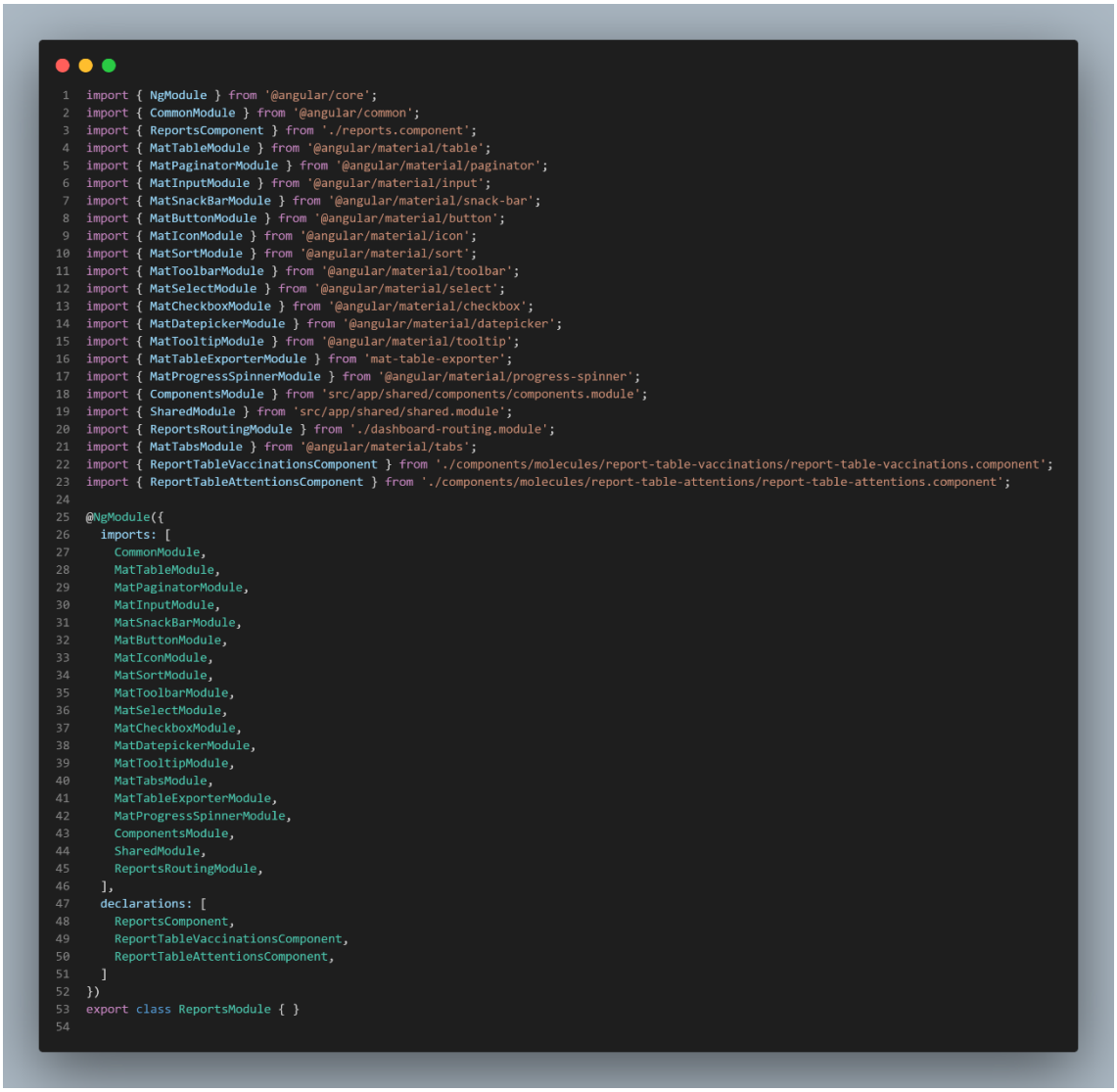
- **Sprint Review**

N° Historia	Sprint	Descripción de tarea	Estado	Estimación (dias)
	Sprint 1: Diseño de base de datos	Diseño de base de datos: Identificación de entidades	Concluido	2
		Diseño de base de datos: Diseño del modelo lógico	Concluido	2
		Diseño de base de datos: Diseño del modelo físico	Concluido	1
1 y 2	**Sprint 2:** Módulo de usuarios	Mockups	Concluido	1
		Creación de entidades (Backend)	Concluido	1
		Creación de servicios (Backend)	Concluido	2
		Creación de controladores (Backend)	Concluido	1
		Creación de DTOs y validaciones (Backend)	Concluido	1
		Prueba de APIs	Concluido	1
		Creación de módulos (Frontend)	Concluido	1
		Creación de interfaces de usuario (Frontend)	Concluido	3
		Comunicación entre backend y frontend	Concluido	1
		Sprint Review	Concluido	1
5	**Sprint 3:** Módulo de vacunaciones	Mockups	Concluido	1
		Creación de entidades (Backend)	Concluido	1
		Creación de servicios (Backend)	Concluido	2
		Creación de controladores (Backend)	Concluido	1
		Creación de DTOs y validaciones (Backend)	Concluido	1
		Prueba de APIs	Concluido	1
		Creación de módulos (Frontend)	Concluido	1
		Creación de interfaces de usuario (Frontend)	Concluido	3
		Comunicación entre backend y frontend	Concluido	1
		Sprint Review	Concluido	1
4	**Sprint 4:** Módulo de atenciones	Mockups	Concluido	2
		Creación de entidades (Backend)	Concluido	1
		Creación de servicios (Backend)	Concluido	3
		Creación de controladores (Backend)	Concluido	1
		Creación de DTOs y validaciones (Backend)	Concluido	2
		Prueba de APIs	Concluido	1
		Creación de módulos (Frontend)	Concluido	1
		Creación de interfaces de usuario (Frontend)	Concluido	4
		Comunicación entre backend y frontend	Concluido	1
		Sprint Review	Concluido	1
3, 8	**Sprint 5:** Módulo de pacientes	Mockups	Concluido	1
		Creación de entidades (Backend)	Concluido	1
		Creación de servicios (Backend)	Concluido	2
		Creación de controladores (Backend)	Concluido	1
		Creación de DTOs y validaciones (Backend)	Concluido	1
		Prueba de APIs	Concluido	1
		Creación de módulos (Frontend)	Concluido	1
		Creación de interfaces de usuario (Frontend)	Concluido	3
		Comunicación entre backend y frontend	Concluido	1
		Sprint Review	Concluido	1
6 y 7	**Sprint 6:** Módulo de reportes	Mockups	Pendiente	1
		Prueba de APIs	Pendiente	1
		Creación de módulos (Frontend)	Pendiente	1
		Creación de interfaces de usuario (Frontend)	Pendiente	1
		Comunicación entre backend y frontend	Pendiente	1
	Sprint 7: Producción y capacitación	Llenado de la base de datos	Pendiente	1
		Mandar la aplicación a producción	Pendiente	3
		Capacitación de los usuarios finales	Pendiente	1

SPRINT 6: Módulo Reportes

- **Creación de módulos**

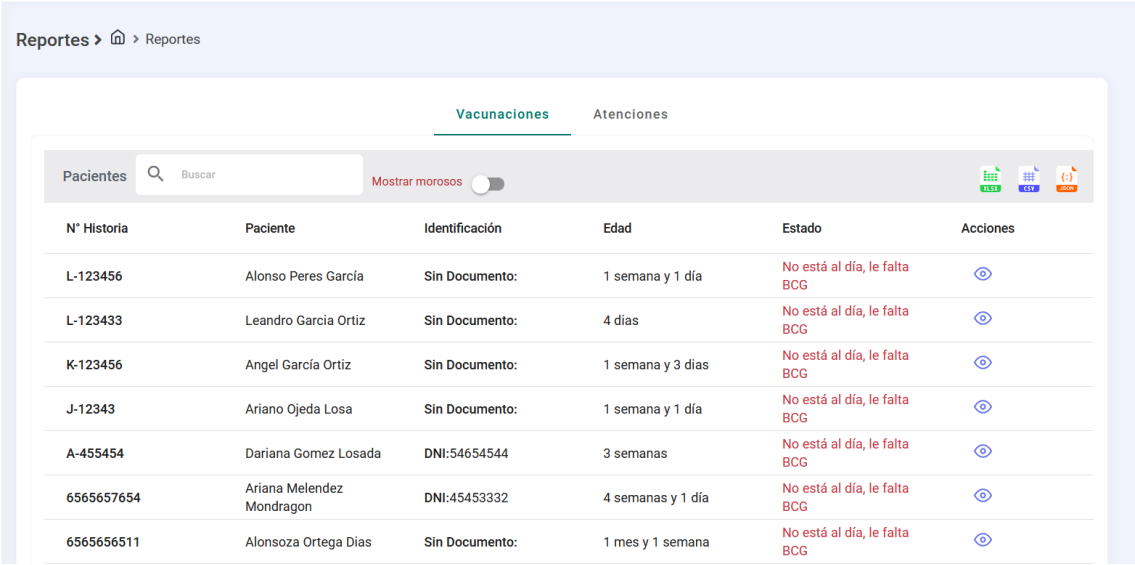
Creación del módulo en el frontend que permite administrar todas las vistas, importaciones y módulos externos correspondientes a este.

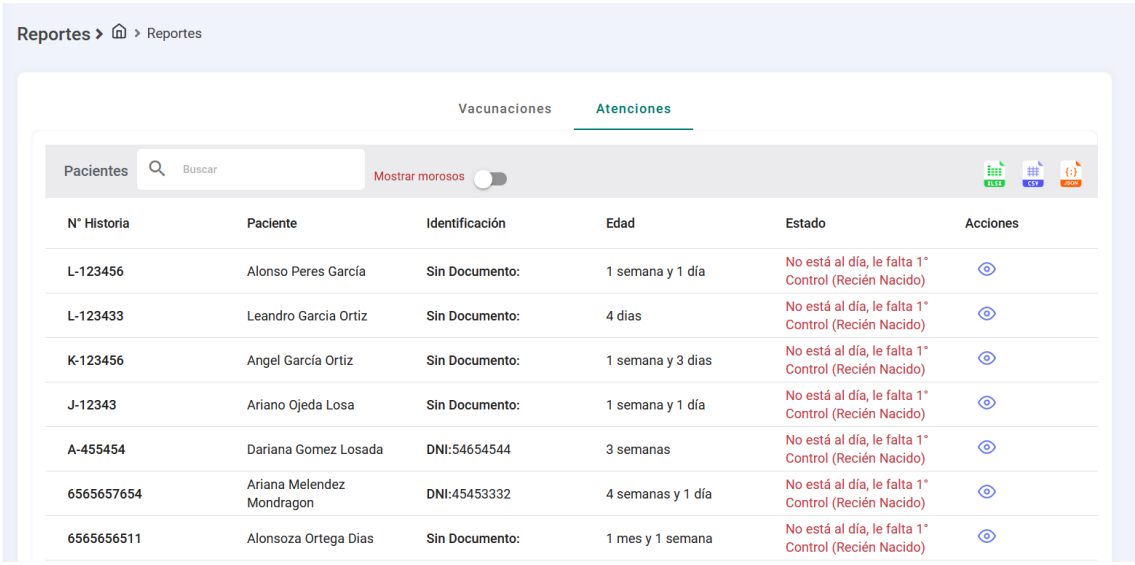


```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ReportsComponent } from './reports.component';
import { MatTableModule } from '@angular/material/table';
import { MatPaginatorModule } from '@angular/material/paginator';
import { MatInputModule } from '@angular/material/input';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import { MatButtonModule } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';
import { MatSortModule } from '@angular/material/sort';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatSelectModule } from '@angular/material/select';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatTooltipModule } from '@angular/material/tooltip';
import { MatTableExporterModule } from 'mat-table-exporter';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { ComponentsModule } from 'src/app/shared/components/components.module';
import { SharedModule } from 'src/app/shared/shared.module';
import { ReportsRoutingModule } from './dashboard-routing.module';
import { MatTabsModule } from '@angular/material/tabs';
import { ReportTableVaccinationsComponent } from './components/molecules/report-table-vaccinations/report-table-vaccinations.component';
import { ReportTableAttentionsComponent } from './components/molecules/report-table-attentions/report-table-attentions.component';

@NgModule({
  imports: [
    CommonModule,
    MatTableModule,
    MatPaginatorModule,
    MatInputModule,
    MatSnackBarModule,
    MatButtonModule,
    MatIconModule,
    MatSortModule,
    MatToolbarModule,
    MatSelectModule,
    MatCheckboxModule,
    MatDatepickerModule,
    MatTooltipModule,
    MatTabsModule,
    MatTableExporterModule,
    MatProgressSpinnerModule,
    ComponentsModule,
    SharedModule,
    ReportsRoutingModule,
  ],
  declarations: [
    ReportsComponent,
    ReportTableVaccinationsComponent,
    ReportTableAttentionsComponent,
  ]
})
export class ReportsModule { }

```

- **Creación de interfaces de usuario**

Creación de interfaces de usuario del Frontend.





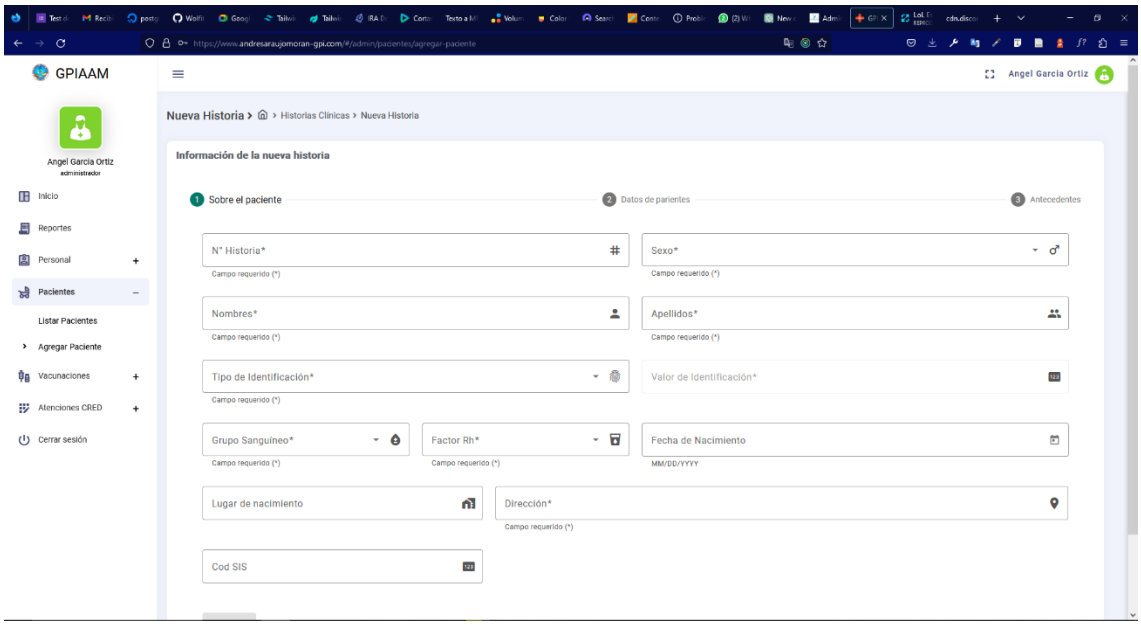
- **Comunicación entre backend y frontend**

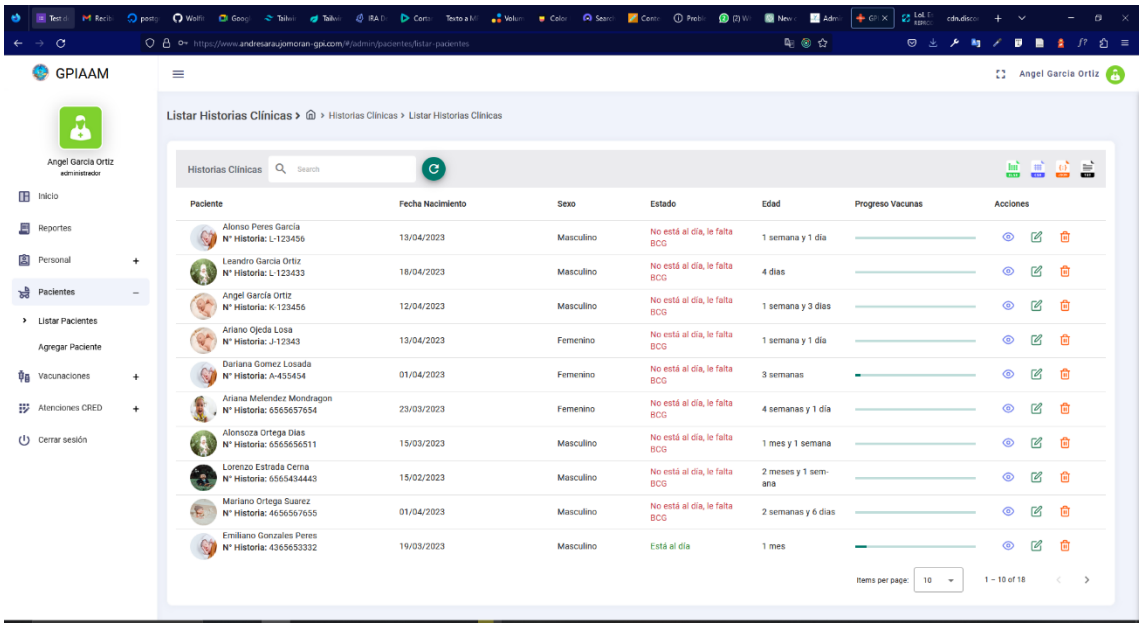
Código fuente que muestra los métodos utilizados para consultar al backend desde el frontend.

```
import { DashboardComponent as doctorDashboard } from "./../../doctor/dashboard/dashboard.component";
import { Page404Component } from "./../../authentication/page404/page404.component";
import { NgModule } from "@angular/core";
import { Routes, RouterModule } from "@angular/router";
import { ReportsComponent } from "./reports.component";

const routes: Routes = [
  {
    path: "",
    component: ReportsComponent,
  },
  { path: "**", component: Page404Component },
];
@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class ReportsRoutingModule { }

```

SPRINT 7: Producción

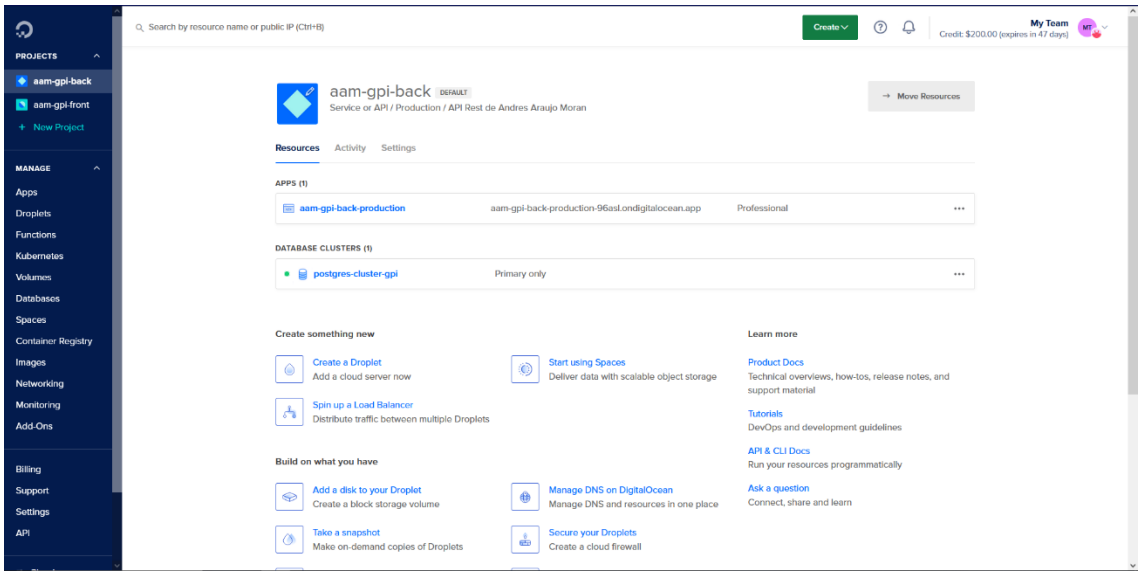
- **Llenado de base de datos**

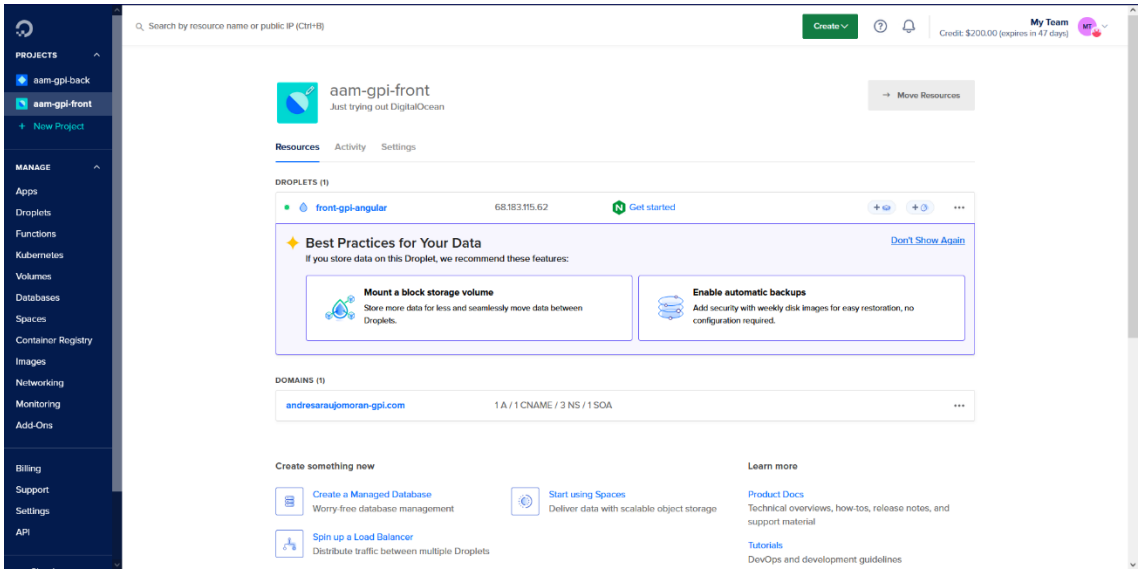




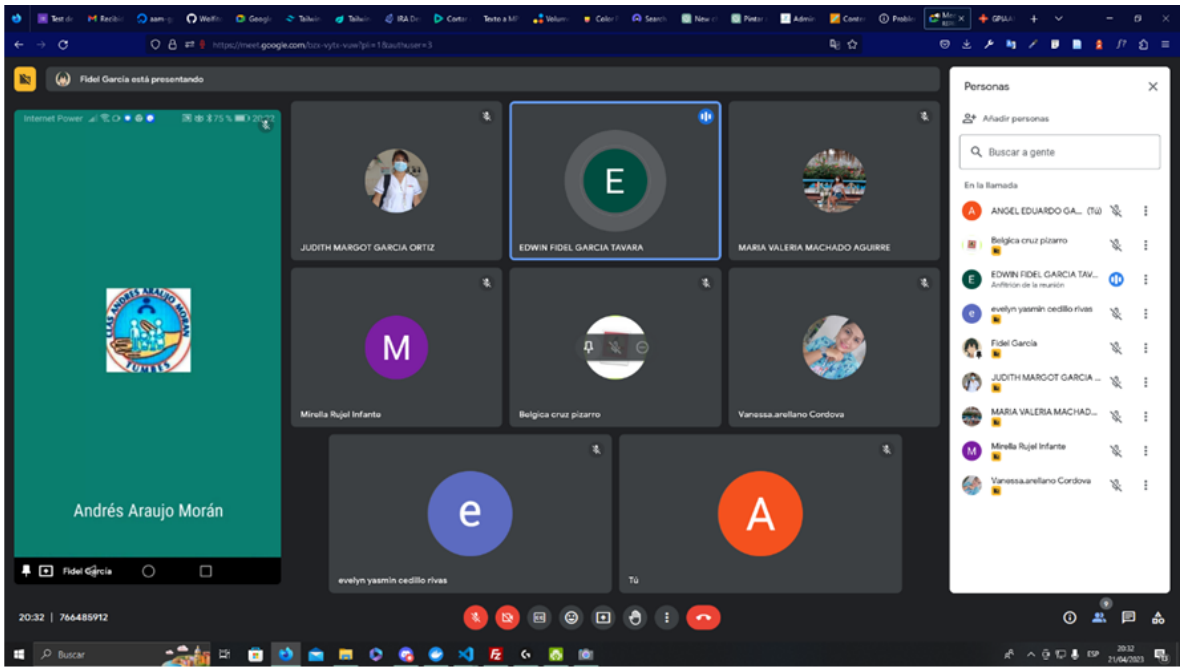
- **Mandar la aplicación a producción**

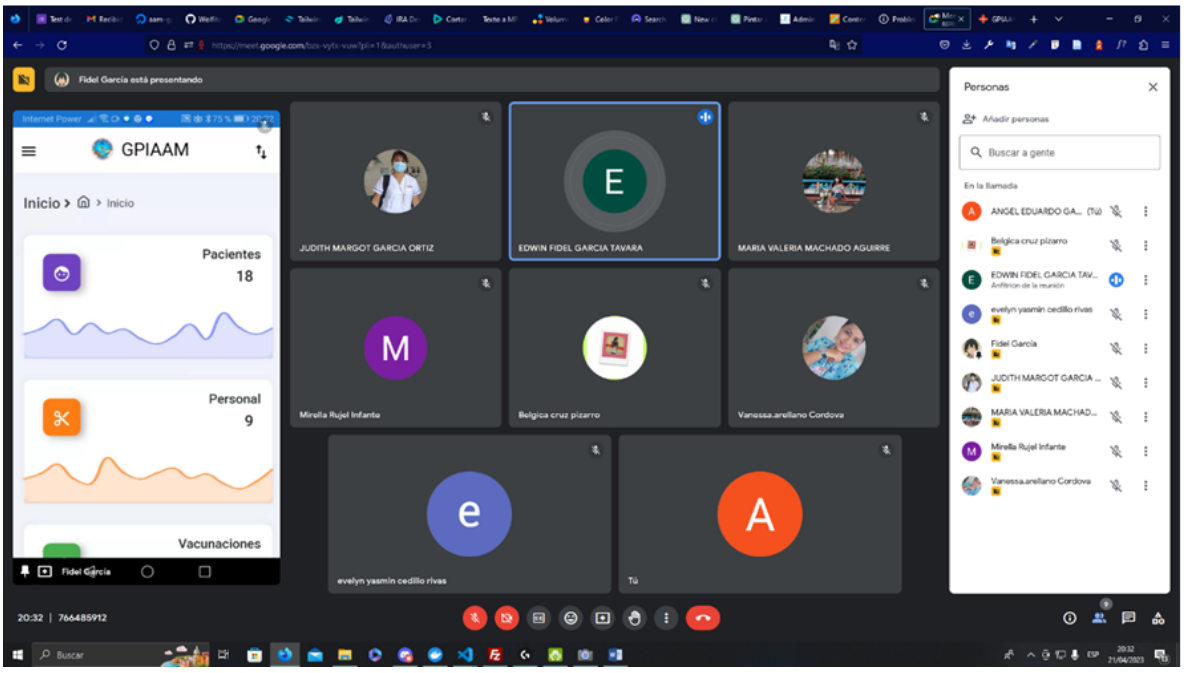
Despliegue en Digital Ocean





- **Capacitación de los usuarios finales**





UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Declaratoria de Autenticidad del Asesor

Yo, AGURTO MARCHAN WINNER, docente de la FACULTAD DE INGENIERÍA Y ARQUITECTURA de la escuela profesional de INGENIERÍA DE SISTEMAS de la UNIVERSIDAD CÉSAR VALLEJO SAC - PIURA, asesor de Tesis titulada: "Aplicación web progresiva con arquitectura monolítica modular para la gestión de pacientes infantes en un centro de salud, Tumbes 2023", cuyos autores son GARCIA TAVARA EDWIN FIDEL, GARCIA ORTIZ ANGEL EDUARDO, constato que la investigación tiene un índice de similitud de 11.00%, verificable en el reporte de originalidad del programa Turnitin, el cual ha sido realizado sin filtros, ni exclusiones.

He revisado dicho reporte y concluyo que cada una de las coincidencias detectadas no constituyen plagio. A mi leal saber y entender la Tesis cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad César Vallejo.

En tal sentido, asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

PIURA, 09 de Julio del 2023

Apellidos y Nombres del Asesor:	**Firma**
AGURTO MARCHAN WINNER **DNI:** 40673760 **ORCID:** 0000-0002-0396-9349	Firmado electrónicamente por: WAGURTOM el 09-07-2023 16:46:05

Código documento Trilce: TRI - 0581986

INVESTIGA UCV