FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Comparación del rendimiento entre angular4 y reactjs, basado en el modelo rail, en la progressive web app de Glup S.A.

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

AUTOR:

Tanta Diaz, Jordi Jersini

ASESOR:

Dr. Emigdio Antonio Alfaro Paredes

LÍNEA DE INVESTIGACIÓN:

Sistemas De Información Transaccionales

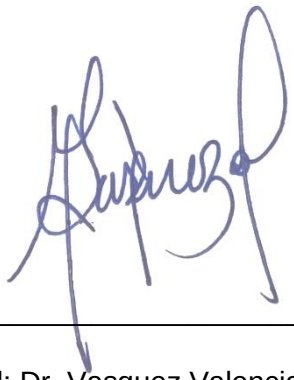
LIMA-PERÚ

2017

PÁGINA DEL JURADO

Presidente: Mg. Crispin Sanshez Ivan

Secretario: Mg. Rivera Crisostomo Renee



Vocal: Dr. Vasquez Valencia Yesenia

Dedicatoria

Este trabajo está dedicado a mis padres que durante tantos años trabajaron y lucharon para darme la mejor educación, a mi hermana que me cuidó y educó como a su propio hijo, a mi hermano que estuvo siempre a mi lado para arrancarme una sonrisa y tranquilizarme, y a mi enamorada que desde que llego a mi vida he conocido el valor del esfuerzo y el amor.

Agradecimiento

A mis padres Carmen Rosa Diaz Alania y Aureliano Castañeda por todo el apoyo y amor incondicional.

A mi hermana Mery Mercedes Bustamante Diaz que formo parte principal de mi formación como persona y profesional.

A mi hermano Abel Luis Tanta Diaz por ser mi compañero y amigo durante tantos años.

A mi enamorada Lucero Nataly Sierra Romero por ser mi fuente de equilibrio, tranquilidad y entendimiento en los momentos más difíciles y complicados de mi vida

A mis profesores de la Universidad Cesar Vallejo que me mostraron el valor del conocimiento y del respeto, y a mis compañeros de final de carrera que me dieron su apoyo en el desarrollo de mi investigación.

A mis jefes y amigos Kristofeer Moran y Jesús Ramírez que me apoyaron y financiaron parte de mi investigación

Declaratoria de autenticidad

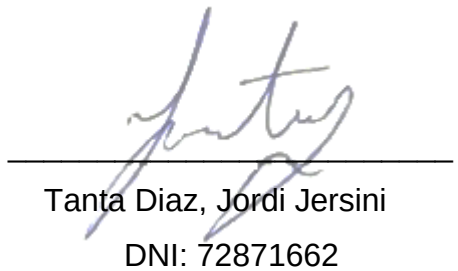
Yo Tanta Diaz Jordi Jersini, con DNI 72871662, a efecto de cumplir con las disposiciones vigentes consideradas en el Reglamento de Grados y Títulos de la Universidad César Vallejo, Facultad de Ingeniería, Escuela de Ingeniería de Sistemas, declaro bajo juramento que toda la documentación presentada es auténtica y veraz.

Asimismo, declaro también bajo juramento que todos los datos e información que se presenta en la esta tesis son auténticos y veraces y están debidamente referenciados.

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

Asimismo, autorizó a la Universidad Cesar Vallejo publicar la presente investigación, si así lo cree conveniente

Lima, 8 de julio del 2017



Tanta Diaz, Jordi Jersini

DNI: 72871662

Presentación

Señores miembros del Jurado Calificador:

En cumplimiento con los lineamientos técnicos que la Universidad Cesar Vallejo exige, pongo a su disposición el informe de tesis titulado “Comparación del rendimiento entre Angular4 y ReactJS, basado en el modelo RAIL, en la progressive web app de GLUP S.A.”, realizado para obtener el título profesional de Ingeniero de Sistemas.

La presente investigación tuvo como finalidad identificar las diferencias de rendimiento entre las tecnologías web Angular4 y React, haciendo uso del modelo RAIL, el cual posee un conjunto de métricas que permiten evaluar el rendimiento de una aplicación web.

El capítulo uno de este trabajo está enfocado en sustentar la importancia de analizar y medir el rendimiento de una aplicación web, y acentuar las bases teóricas necesarias para poder ejecutar la investigación.

En el capítulo dos se especifica las estrategias y herramientas metodológicas de la investigación.

En el capítulo tres se muestran los resultados obtenidos luego de 36 días de recolección de información, estos datos son procesados y posteriormente analizados en este capítulo.

En los capítulos finales, cuatro, cinco y seis, se compara los resultados con los antecedentes recolectados para la investigación, se generan las recomendaciones para futuros trabajos y se presentan las conclusiones obtenidas en función a los resultados obtenidos y los objetivos de la investigación.

Con la convicción de que se otorgará el valor justo y mostrando apertura a sus observaciones, agradezco por anticipado las sugerencias y apreciaciones que se brinden a la investigación

El autor.

ÍNDICE GENERAL

INDICE DE FIGURAS

ÍNDICE DE ANEXOS

ÍNDICE DE TABLAS

RESUMEN

El presente trabajo fue desarrollado con la finalidad de determinar que tecnología web, Angular4 o ReactJS, posee un mejor rendimiento. Para lo cual se usó como referencia al modelo RAIL, el cual define un conjunto de directrices que se deben cumplir para que una aplicación web pueda otorgar un buen rendimiento.

Este trabajo cuenta con una gran cantidad de información recopilada relacionada al funcionamiento de las aplicaciones web, las nuevas tecnologías, estrategias que han surgido en los últimos años orientados a mejorar el rendimiento de una aplicación web y el impacto económico del rendimiento de una aplicación web en una empresa.

Para lograr determinar la diferencia entre estas dos tecnologías, se desarrollaron dos aplicaciones software bajo la metodología XP, una con la tecnología ReactJS y otra con Angular4. Ambas aplicaciones fueron desplegadas en el servidor principal de la empresa GLUP S.A. durante un periodo de 36 días (18 días por tecnología) en los cuales se recopilo información de los dispositivos de los usuarios que accedieron a la página web de GLUP S.A.

La información que se recolecto aborda los 4 pilares principales del rendimiento definidos por RAIL, la velocidad de respuesta a una acción, cantidad de fotogramas perdidos en una animación, cantidad de tiempos inactivos del CPU y la velocidad de carga de la página web.

Todos los datos fueron procesados con la herramienta SPSS statistics 23, lo cual permitió concluir que Angular4 tiene una mejor velocidad de respuesta a acciones y permite animaciones más fluidez que React, sin embargo, React permite la aparición de mayores tiempos de inactividad del CPU y un mejor tiempo de carga.

Palabras clave: Angular4, React, rendimiento, RAIL, aplicaciones, empresa.

ABSTRACT

The present work was developed with the purpose of determining which web technology, Angular4 or ReactJS, has a better performance. For this purpose, the RAIL model was used as a reference, which defines a set of guidelines that must be met in order for a web application to perform well.

This work has a large amount of information collected related to the functioning of web applications, new technologies, strategies that have emerged in recent years aimed at improving the performance of a web application and the economic impact of the performance of a web application in a company.

In order to determine the difference between these two technologies, two software applications were developed under XP methodology, one with ReactJS technology and another with Angular4. Both applications were deployed on the main server of the company GLUP S.A. During a period of 36 days (18 days per technology) in which information was collected from the devices of the users who accessed the GLUP S.A. website.

The information collected addresses the 4 main performance pillars defined by RAIL, the speed of response to an action, the number of frames lost in an animation, the number of idle CPU times and the loading speed of the web page.

The information collected was processed using the tool SPSS statistics 23 which allowed to conclude that Angular4 has a better response speed to actions and allows animations more fluency than React, however, React allows the appearance of greater CPU idle times and A better loading time.

Keywords: Angular4, React, performance, RAIL, applications, company.

I. INTRODUCCIÓN

1.1 Realidad Problemática

En la actualidad existen muchas páginas web dedicadas a la venta por internet. Las pequeñas y grandes empresas han logrado encontrar en las plataformas web un medio directo para poder llegar a los clientes y estar presentes en el momento del nacimiento de la necesidad de compra debido al aumento de dispositivos móviles adquiridos. Más de 3 billones de dispositivos móviles están conectados a la web y se estima que para el 2020 llegue a los 50 billones Yuhao (2017). Esta gran demanda ocasiona que muchas empresas y científicos estén en la búsqueda de nuevas tecnologías y paradigmas que aseguren que la aplicación web para dispositivos móviles posea un alto grado de aceptación y preferencia por parte del usuario. Aunque en la actualidad no existe una tecnología que pueda dar una solución directa a esta situación, se sabe que el factor principal por el cual un usuario prefiere entrar y comprar en aplicación web para dispositivos móviles es el rendimiento, el 57% de los usuarios abandona la aplicación web si esta tarda más de 3 segundos en cargar y cerca de la mitad de la población espera que esta cargue en al menos 2 segundos Suresh (2017). El impacto que llega a tener los problemas de rendimiento en las ganancias de una empresa es muy alto. Se estima que por cada segundo de retraso en la carga de una página Amazon pierde 1.6 billones de dólares anuales Yuhao (2017). Este caótico escenario ha dado lugar a diferentes investigaciones enfocadas en la optimización del rendimiento de aplicaciones web para dispositivos móviles.

En el año 2015, la compañía Google realizo un congreso en el cual Join & Bovens (2015) mostraron el incremento en la preferencia, por parte de los usuarios, hacia una aplicación web móvil sobre aplicaciones Android nativas.

- Un promedio de 25 aplicaciones son usadas al mes por usuario mientras que más de 100 sitios web son navegados por usuario.
- Se pierde 20% de los usuarios objetivo por cada paso que se da en el proceso de instalación de una aplicación móvil.
- Las estadísticas mostradas por Fiksu (compañía que monitorea el ecosistema de las aplicaciones nativas), en la cual se logra apreciar un aumento del costo

invertido por consumidor de una aplicación nativa (Android y IOS), desde el año 2014.

Por lo cual Join & Bovens (2015) presentaron un nuevo concepto denominado progressive web app, el cual engloba un conjunto de técnicas y tecnologías que permitan al usuario percibir y usar una aplicación web como una aplicación nativa, aprovechando con ello la preferencia del usuario por la navegación en sitios web. Este nuevo concepto tiene como base principal la fluidez y rendimiento del sitio web, por ello Osmani (2015), menciona al modelo RAIL, modelo de rendimiento enfocado en el usuario, como un principio base para la valoración del rendimiento en una progressive web app.

Existen muchas tecnologías JavaScript que tienen la capacidad de implementar este nuevo concepto, dentro de los que más destacan, según Koetsier (2016), son React y Angular. La primera fue desarrollada por Facebook, esta tecnología permite trabajar bajo un desarrollo fluido y eficiente en una aplicación web, basada en componentes y la aplicación del virtual DOM para optimizar la manipulación y actualización de los elementos del árbol DOM. La segunda es un framework desarrollado por Google, ahora en su versión 4, maneja las capas de modelo, vista y controlador e incorpora el uso de componentes para la creación de aplicaciones web complejas.

La empresa GLUP S.A. ofrece un servicio de creación de un catálogo 3D para sitios web dedicadas a la venta de prendas de vestir y desea agregar el servicio de desarrollo de páginas web, y aun que en la actualidad existen empresas dedicadas a ello, GLUP S.A. apuesta por el nuevo concepto progressive web app. Este interés por parte de la empresa GLUP viene derivador de la existencia de e-comerce pioneras en el uso de este nuevo concepto, las cuales obtuvieron sorprendentes resultados tal y como es el caso de AliExpress con 104% de nuevos usuarios a través de todos los navegadores, Housing.com con un aumento del 38% de conversiones y 5miles que incremento en un 30% el tiempo de estadía en el sitio web (Google Developers, 2016). Ante ello la empresa GLUP S.A. tiene como meta determinar que tecnología web, Angular 4 o React, aborda mejor el concepto de

progressive web app y cuál de ellos provee un mejor rendimiento evaluándolos con el modelo RAIL, permitiendo así asegurar a la empresa que las progressive web app puestas en producción sean lo mejor posible y otorguen una buena experiencia al usuario. Para ello la empresa GLUP S.A. solicitó la creación de una página web con ambas tecnologías con el fin de monitorear y recopilar información sobre su rendimiento en dispositivos móviles.

1.2 Trabajos previos

En las consultas realizadas a diferentes revistas científicas y repositorios de universidades se ha podido hallar información relacionada a nuestro estudio.

- Molin (2016) realizó una comparación de las tecnologías JavaScript Angular y React con la intención de determinar que tecnología permite desarrollar una SPA de manera más eficiente, para lo cual se evaluó el tiempo de carga en una página web, el tiempo de respuesta ante una acción (modificar ítems de un registro) y la cantidad de memoria usada en la ejecución de tareas. Este trabajo concluyó que React es más rápido que Angular en cuanto al tiempo de carga en una página web y que Angular requiere usar una mayor cantidad de memoria para ejecutar una tarea.

  De este trabajo se rescatan los resultados obtenidos en la comparación de las tecnologías React y Angular, los cuales serán usados para contrastar los resultados en esta investigación.

- Tammy Everts (2013) hizo un resumen sobre las mejores estrategias de optimización que se puede seguir para mejorar el rendimiento de una aplicación móvil, la gran mayoría de estos enfocados al tiempo de carga de la página web ya que se muestra que en el año 2009 el 43% de los usuarios abandonan una página web si es que esta demora en cargar más de 3 segundos y en el 2011 un estudio realizado por Compuware indica que el 43% de los usuarios de dispositivos móviles afirma que los sitios web cargan más lento de lo que ellos esperaban .

De este trabajo se rescatan las técnicas de optimización de rendimiento las cuales serán usadas para el desarrollo de los dos softwares y los datos estadísticos relacionados al impacto del rendimiento de una aplicación web en la retención y obtención de nuevos clientes.

- Jan (2013) presentó como solución a los problemas de rendimientos de una aplicación web el uso de la técnica de paralelismo dinámico llamada especulación multi-hilo, la cual permite al motor JavaScript trabajar sobre los procesadores multi-núcleos del hardware, logrando así aprovechar al máximo los recursos de la computadora. La técnica fue aplicada al motor JavaScript Squirrelfish y posteriormente comparada con otros motores tales como Just-in-Time y Google V8 con lo cual se tuvo como resultado que el motor JavaScript Squirrelfish tuvo un rendimiento 8.5 veces más a comparación de los otros dos motores y que el motor Just-in-Time tiene el más bajo rendimiento.

  De este trabajo se rescata la valoración de los motores JavaScript ya que esto permite delimitar el escenario bajo el cual se recopilará la información para la comparación de las tecnologías.

- Ulan (2016) describió un enfoque, implementado en JavaScript, para el motor V8 usado por el navegador Chrome el cual permite programar el tiempo dedicado para la pausa de recolección de basura en los tiempos de inactividad del navegador Chrome, logrando así disminuir los tiempos de bajo rendimiento en una animación.

  Este trabajo permite consolidar los conocimientos referidos al tiempo de inactividad en una aplicación web, la cual es una de las dimensiones usadas en este trabajo de investigación.

- Koetsier (2016) evaluó un conjunto de tecnologías JavaScript con la finalidad de desarrollar una interfaz gráfica para un software llamado Vampires. Para esto se requería una tecnología moderna y una interfaz amigable. Para el proceso de discriminación entre las tecnologías JavaScript, se usó la ISO

25010:2011, el cual es un modelo que permite determinar la calidad del producto software.
Este trabajo tuvo como conclusión que ReactJS y AngularJS son las mejores opciones para el desarrollo de una aplicación web y que AngularJS tiene mayores ventajas sobre ReactJS debido a que AngularJS es un framework MVC y ReactJS solo trabaja en la capa de la vista.

De este antecedente, se rescata la valoración las tecnologías JavaScript AngularJS y ReactJS lo cual permite entender la importancia de su comparación.

- Nagele (2015) desarrolló una metodología para medir el rendimiento e identificar las secciones del código que tengan un impacto negativo en este. Este trabajo concluye con la generación de un método para la medición del rendimiento de una aplicación web y la creación de una herramienta para medir el rendimiento.

  De este antecedente se rescata el conocimiento generado con respecto al impacto de un determinado navegador web en una evaluación de rendimiento, lo cual será usado en este trabajo de investigación como referencia para delimitar y especificar el escenario para la recolección de información

- Phuc Tran (2016) evaluó diferentes frameworks JavaScript tales como ReactJS, AngularJS y BackBoneJS para poder determinar cuál de ellos usar para el desarrollo de la aplicación web. Así como una valoración de los diferentes navegadores y su soporte para las nuevas tecnologías. Este trabajo finiquitó con el desarrollo de una aplicación web escalable desarrollada bajo la tecnología AngularJS.

  De este trabajo se rescata el análisis y valoración de los diferentes navegadores y el soporte de estos para aplicaciones modernas, con lo cual se determinó que los navegadores Firefox y Chrome tienen mejores posibilidades de acoger una moderna aplicación web.

- Myeongjin (2015) habló sobre la aplicación de la programación paralela como solución para el rendimiento en una aplicación web ya que así se puede hacer uso de todos los recursos que puede otorgar el hardware del dispositivo. En este trabajo se plantea que para la creación de una aplicación web de debería usar el lenguaje OpenCL el cual está orientado al paralelismo y finalmente convertir el código a código JavaScript mediante la herramienta propuesta denominada OpenCL-to-WebCL con lo cual se logró mejorar el rendimiento en un 75%.

  De este trabajo se rescata el uso de un lenguaje de programación enfocado a paralelismo como solución a los problemas rendimiento de una aplicación web el cual será usado en la sección recomendaciones en este trabajo.

- Willian (2013) investigó las diferencias entre las aplicaciones nativas para dispositivos móviles y las aplicaciones web, planteando la posibilidad de que éste último pueda reemplazar a las aplicaciones nativas. Ello debido a la inclusión de HTML5 en aplicaciones web para dispositivos móviles y el alto rendimiento que estas pueden llegar a tener.
  Este trabajo permitió entender el rendimiento en ambas tecnologías al crear dos softwares similares y ponerlos a prueba durante 6 meses, con lo cual se dedujo que ante interacción con el hardware del dispositivo móvil una aplicación web posee un pobre rendimiento a comparación de una aplicación nativa.

  De este trabajo se rescata el resultado obtenido mediante la comparación de ambas tecnologías ya que permite entender las limitaciones que posee una aplicación web para móviles.

- Umar (2014) investigó el diseño de un nuevo patrón de diseño que permita mejorar el rendimiento de una aplicación web. Este patrón denominado framework de desarrollo de aplicaciones web está basado en el patrón MVC y tiene como prioridad mejorar el tiempo de respuesta ante la ejecución de una tarea transaccional que involucre una comunicación directa con la base de

datos. Para verificar el impacto de este nuevo patrón de diseño se realizó una comparación del tiempo de respuesta entre un sistema web convencional y el mismo sistema bajo el nuevo patrón, obteniendo con ello una pequeña mejora en la velocidad de respuesta.

Este trabajo muestra la importancia de usar los correctos patrones de diseño en la creación de una aplicación web, lo cual será tomado en cuenta en la fabricación de los dos softwares a evaluar en este trabajo de investigación.

- Pete (2016) estudió a la tecnología React con la finalidad de comprender su comportamiento y utilidad, para lo cual se realizó una entrevista a uno de los miembros creadores de esta tecnología, Pete Hunt, el cual mencionó que React surgió como solución a los problemas de rendimiento en la sincronización entre los elementos visuales de una página web y el conjunto de información obtenida del servidor, en función de ello se fueron evolucionando diferentes características de esta tecnología como el manejo de estados para una sincronización efectiva, el uso de accesorios para manejar la comunicación entre componentes padres e hijos y la creación de manejadores de eventos que permitan controlar el comportamiento de los componentes. Por lo cual se menciona que React posee como potencial característico el poder contar con un valor de entrada, tener un valor de salida, y computar la diferencia entre estos para recién ejecutar el renderizado de los elementos visuales, haciendo de una actualización de información una tarea fluida y de alto rendimiento.

  Este trabajo permite consolidar los conocimientos referidos a la tecnología React (bondades, características y funcionamiento interno).

- Rijwan (2016) investigó sobre las pruebas de rendimiento y su importancia en una aplicación web y el análisis de cuellos de botella de la aplicación basados en hardware, software y recursos utilizados, Asimismo también se hace un recorrido por las herramientas software más utilizadas para realizar este tipo de pruebas (Load Runner, HP ALM, Perform Logs, CA Wiley).

Este trabajo permite entender la importancia de un sistema de monitoreo, ya que ello permite tomar decisiones acertadas relacionadas a la solución de problemas de la aplicación web.

- Yuhao (2017) estudió el rendimiento de una aplicación web en un dispositivo móvil, desde la perspectiva del consumo de energía de la batería y el rendimiento del CPU. Para ello se propuso un núcleo general personalizado y especializado para aplicaciones móviles y el desarrollo de un acelerador de hardware para el kernel de resolución de estilos dentro del motor del navegador web. Este trabajo concluyó con la mejora de 22.2% en el rendimiento y una disminución en el consumo de energía de 18.6%.

  Este trabajo permite conocer la situación actual de las tecnologías móviles en relación a las aplicaciones web, Asimismo permite apreciar el interés del mercado en referencia a este tema debido a que fue financiado por la corporación Intel, AMD y Google.

- Wonsun (2014) estudió el rendimiento del lenguaje JavaScript en el navegador Chrome, enfocándose principalmente en el compilador usado por este navegador llamado V8 el cual compila el código JavaScript para posteriormente ejecutarlo. SI bien es cierto el compilador optimizar el rendimiento del código a ejecutar, este posee un grave problema referente a la codificación, ya que V8 requiere una codificación estático mientras que JavaScript no incluye este concepto. Es por ello que finalmente este trabajo se modifica el comportamiento del compilador V8 para eliminar el requisito de codificación estático mejorando el tiempo de ejecución de una herramienta llamada JBench en un 36%.

  Esta investigación permite entender el comportamiento del navegador y funcionamiento e impacto de un compilador tiene en el rendimiento de una aplicación web.

- Sanchez (2016) comparó el rendimiento entre una aplicación web y una aplicación nativa para Android, para lo cual se desarrolló un software dedicado

a monitorear la radiación solar en ambas plataformas. La evaluación permitió observar el comportamiento de ambas plataformas con respecto al usuario y uso de recursos del dispositivo móvil, con lo cual se pudo determinar que muna aplicación nativa usa un 65% menos de memoria a comparación de una aplicación web, sin embargo, la aplicación web llega a tener mejor aceptación por parte del usuario final debido a que este puede ser ejecutada en diferentes plataformas y sistemas operativos.

De este trabajo se rescata la valoración por parte del usuario final hacia la aplicación web, a pesar de su alto uso de memoria, lo cual permite confirmar la importancia de conocer que tecnología, Angular4 o ReactJS, permite un mejor rendimiento. Asimismo, también se rescata la forma en la que se evaluaron ambas plataformas la cual será usada para el desarrollo de esta investigación.

- Troy (2014) estudió las diferencias entre un cliente web de escritorio y un cliente web móvil. Las principales contribuciones de este proyecto son relacionadas a las tendencias del uso de dispositivos móviles y sus dificultades en con respecto al manejo de la red, lo cual genera un problema en la experiencia del usuario.

  De este trabajo se rescata la observación referente a las dificultades de manejo de red que tienen los dispositivos móviles, por lo cual se optimizaran los recursos usados en el desarrollo de las aplicaciones web.

- David (2016) investigó sobre la importancia de tener un sitio web de diseño amigable y adaptable para mejorar el SEO. En este trabajo se aprecia el comportamiento y factores principales usados por el algoritmo de google para poder posicionar a una página web ante una búsqueda, dentro de los cuales se destacan el impacto de ocultar elementos de una página web, los beneficios de un diseño adaptable y la unión de las URLs de la versión móvil y de escritorio. Resaltando en esta investigación la importancia de una web adaptativa, pues el algoritmo de google toma a esta con una valoración más alta para el SEO, que el contar con una versión dedicada a dispositivos móviles.

De esta investigación se recogen las recomendaciones relativas al diseño de una aplicación web enfocadas a mejorar la experiencia del usuario.

- Alex (2013) investigó sobre las buenas prácticas a seguir para poder crear una eficiente aplicación web para dispositivos móviles. Debido a la poca capacidad que posee un dispositivo móvil es necesario establecer estrategias que permitan un funcionamiento similar al de una aplicación nativa. Estas estrategias fueron enfocadas principalmente en el tiempo de carga de la aplicación, abordando por ello temas como conexiones paralelas para optimizar las descargas, comprimir recursos, administrar el tiempo de análisis del código JavaScript, evitar el proceso de diseño y cálculo de estilos por parte del navegador, evitar selectores ineficientes de CSS, comprimir recursos y optimizar imágenes.

  De este artículo se rescató todo el conjunto de prácticas orientadas al desarrollo de una aplicación web, permitiendo así que las aplicaciones usadas en el proceso de comparación posean el mejor rendimiento posible.

- Patrick (2013) estudió el rendimiento y monitoreo web, describiendo los dos tipos de monitoreo que pueden ser usados para recopilar información, el monitoreo activo y monitoreo pasivo. Ambos métodos son usados para diferentes fines y la información obtenida es valorada de manera distinta, según se menciona en el artículo el monitoreo activo sirve para formar un entendimiento base sobre el rendimiento de una página web en un entorno controlado y el monitoreo pasivo sirve para entender el rendimiento de la página web desde la perspectiva real de un usuario haciendo uso de un conjunto de tecnologías como los APIs creados por la W3C.

  De este artículo se rescató las técnicas y APIs usados para la recolección de información en un monitoreo pasivo.

- Nygard (2016) realizó una revisión de la literatura de las nuevas tecnologías y conceptos surgidos con respecto a las aplicaciones web modernas. Asimismo, también se analiza los problemas de rendimiento que pueden ocasionar estas tecnologías y cómo resolverlos.

  De este trabajo se rescata la información obtenida acerca el estado actual de las tecnologías web, lo cual permite consolidar los conocimientos acerca de las tecnologías y los frameworks para el desarrollo de aplicaciones web modernas.

1.3 Teorías relacionadas al tema

1.3.1 Progressive web app

LePage (2016) mencionó que una progressive web app une lo mejor de las tecnologías web y lo mejor de las aplicaciones nativas, permite construir una continua relación con el usuario mediante el uso de push notifications, rápida carga de la página web en conexiones lentas (3G) y acceso mediante un ícono en la pantalla de inicio del dispositivo móvil.
Según LePage (2016), existen algunos conceptos claves que involucran el término progressive web app, estos son:

- **Progresivo**

  Funciona para cualquier usuario independientemente del navegador que se use.

- **Adaptable**

  Es adaptable a cualquier tipo de dispositivo o pantalla.

- **Independiente de la conectividad**

  Se apoya en el service worker para poder funcionar en escenarios sin conexión y con baja calidad de conexión de internet.

- **Como una app**

  Se siente como una aplicación móvil, ya que tiene el estilo de interacción y navegación de una aplicación nativa.

- **Fresco**

  Puede recibir paquetes de actualización de manera continua.

- **Seguro**

  Usa certificados SSL para mejorar la seguridad.

- **Descubrible**

  Con la ayuda del archivo manifiesto liberado por la W3C y su registro mediante el service worker, permite ser identificado como una aplicación y que los motores de búsqueda puedan indexarlo.

- **Re-enganchable**

  Permite la fácil fidelización del usuario ya que puede recibir notificaciones.

- **Instalable**

  Permite al usuario agregar la aplicación web a la pantalla de inicio sin dirigirse a la tienda de aplicaciones.

- **Enlazable**

  Fácil de compartir ya que es solo un enlace web y no requiere instalación.

- **App shell**

  El app shell se refiere al uso mínimo de HTML, CSS y JavaScript para cargar el núcleo principal de la aplicación y la interfaz de usuario (UI) independientemente de la información. Toda la UI y la infraestructura es almacenado en caché o mediante el API IndexedDB y luego utilizado por el Service Worker. Todo ello permite enfocarse en la velocidad de carga de la progressive web app, otorgándole así propiedades similares a una aplicación nativa.

  Dentro de este conjunto mínimo de componentes deben estar presentes:

- El header y un botón para agregar o actualizar.
- Un contenedor para las tarjetas (paneles de información)
- Tarjetas de contenido
- Un diálogo para agregar nuevas localidades
- Un indicador de carga

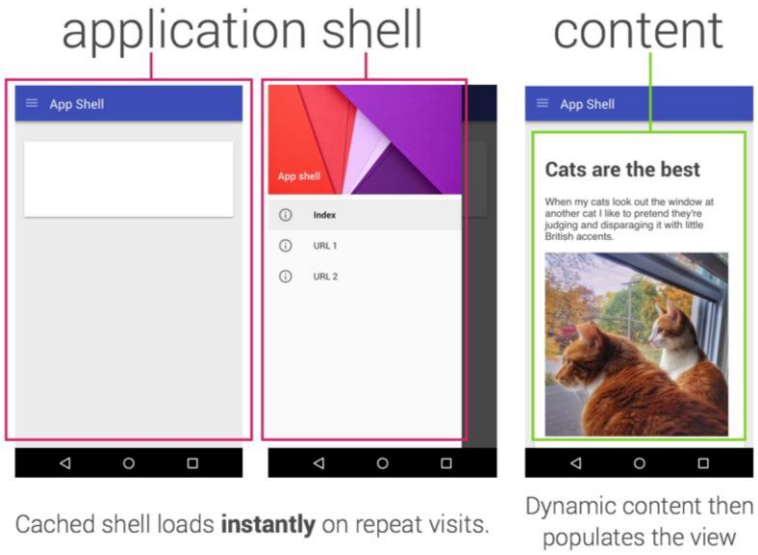


Figura 1. App Shell. (LePage, 2016)

Osmani (2016) mencionaron que el app shell incluye estilos CSS en línea, dentro del tag <style>. Ello es necesario para aumentar la velocidad del proceso de pintura de la app shell ya que solicitarlo de red implicaría un retraso en la construcción del árbol DOM.

- **Service Worker**

  Osmani (2016) se refirió al service worker como un script que se ejecuta en un hilo paralelo al principal, el cual puede responder a solicitudes de red creadas por la página web y a push notifications del server. El service worker no puede interactuar directamente al árbol DOM, pero puede acceder a complementos como API Cache, API Fetch, API IndexedDB y postMessage para comunicación con el hilo principal.

  El service worker funciona como un proxy en el navegador, el cual puede interceptar solicitudes de red creadas por la página web, y retornar la respuesta de la red o de un almacenamiento local es por ello que el service worker provee la capacidad a la progressive web app de imitar la carga instantánea y actualizaciones regulares de una aplicación nativa. Así como otorgarle funcionamiento en un escenario sin conexión, al almacenar localmente el app shell y otros recursos después de la primera carga de la página.

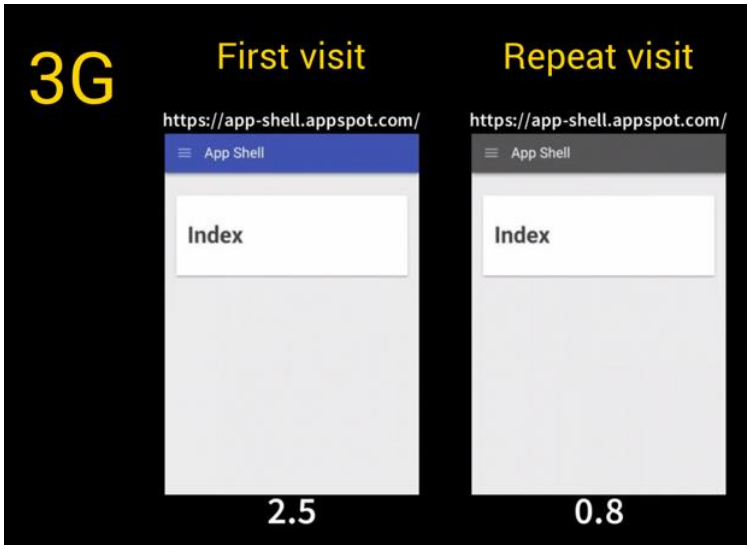


Figura 2. Tiempo para el primer evento paint. (Osmani, 2016)

- **IndexedDB**

  W3C (2016) definió a un IndexedDB como un API que permite manejar el almacenamiento web basado en pares de llaves y valores. El origen de la base de datos es el mismo que el de la aplicación web, la cual no es alterada si se cambia el dominio, ello permite que cada aplicación web maneje su propia e independiente base de datos y que esta no pueda ser alterada por otras aplicaciones web. La información almacenada puede ser manejada mediante una llamada asíncrona a una transacción, la cual puede ser eliminar, actualizar u obtener un registro.

1.3.2 Ruta crítica de reproducción

Grigorik (2015) indicó que hablar sobre el rendimiento de una aplicación web y su optimización, implica conocer y entender la ruta crítica de reproducción, el cual es el proceso que se ejecuta desde la recepción de los recursos HTML, CSS y JavaScript hasta la visualización de los pixeles en la pantalla.

Para que el navegador pueda mostrar la aplicación en la pantalla, requiere primero construir el árbol DOM y CSSOM.

- **Modelo de objetos del documento (DOM)**

  El DOM es un objeto jerárquico en forma de árbol, en el cual existen relaciones de padre e hijo. Asimismo, cuenta con un proceso de construcción secuencial que va desde la recepción de la respuesta de red hasta la construcción de todo el árbol DOM.

- Conversión: El navegador lee los bytes del archivo HTML desde el almacenamiento local o de la red y lo traslada en caracteres individuales basado en la codificación del archivo.
- Convertidor de token: El navegador convierte cada cadena de caracteres en distintos tokens con sus propias tales como “<html>”.
- Lexing: Los tokens son transformados en objetos cada uno con sus propias propiedades y reglas.
- Construcción del DOM: Al final el árbol DOM se construye definiendo relaciones padre-hijo entre cada objeto.

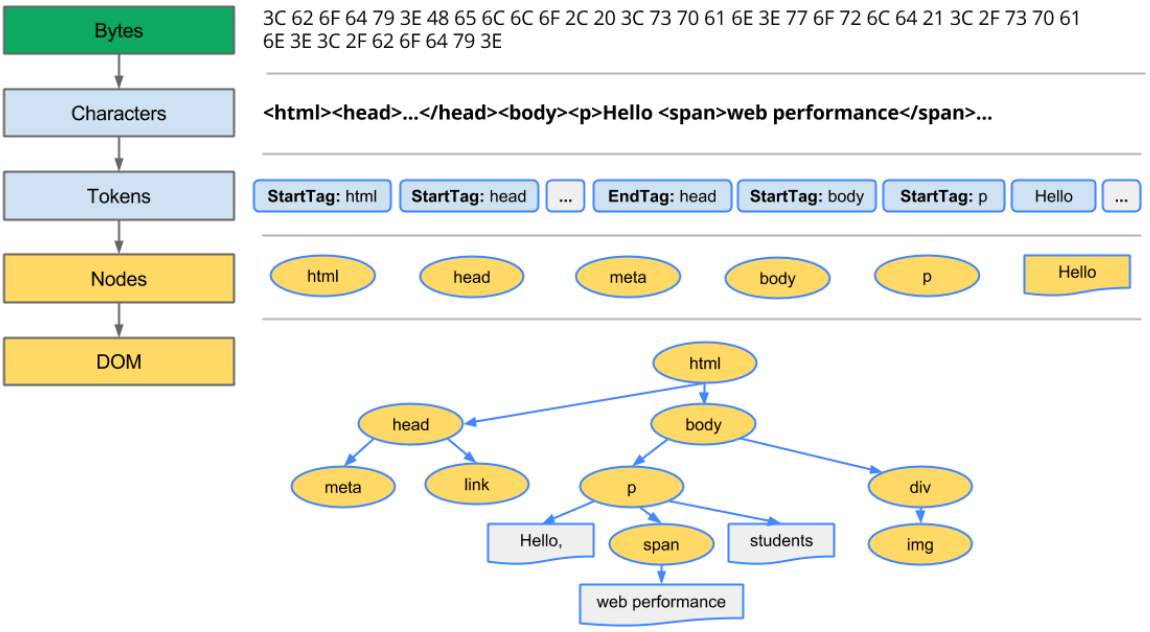


Figura 3. Creación del DOM. (Grigorik, 2015)

- **Modelo de objeto CSS (CSSOM)**

  Mientras el árbol DOM está construyéndose, este encontrará el tag “link” en la sección “head”, el cual hace referencia a un archivo externo CSS. Ya que este es un recurso necesario para mostrar el contenido en la pantalla, inmediatamente el navegador ejecuta una solicitud a la red para obtener este recurso y ejecutará un proceso similar al del árbol DOM para construir el árbol CSSOM.

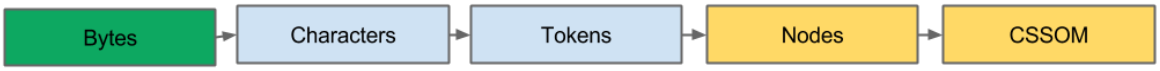


Figura 4. Creación del CSSOM (Grigorik, 2015)

Los estilos son aplicados de manera jerárquica, permitiendo así que las reglas CSS se apliquen de manera recursiva bajando por cada nodo del árbol CSSOM.

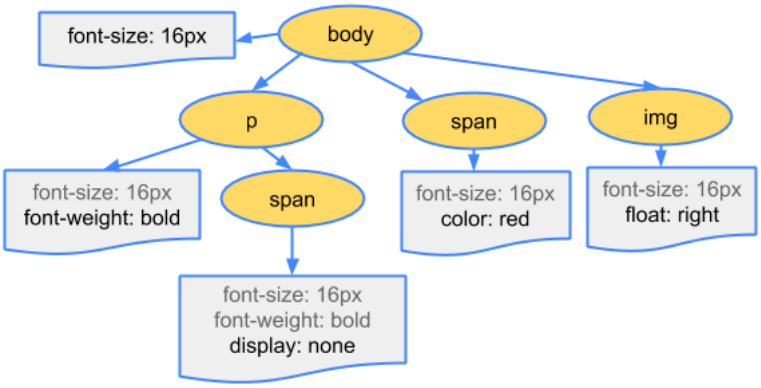


Figura 5. Aplicación de reglas CSS (Grigorik, 2015)

Luego de la construcción del árbol DOM y el árbol CSSOM, estos se combinan y generan el árbol de visualización.

- **Árbol de visualización**

  El árbol de visualización permite calcular el diseño de cada elemento visible y se usa como base para el proceso de pintura que permite visualizar los pixeles en la pantalla. Ello se logra mediante la unión del árbol DOM y el árbol CSSOM.

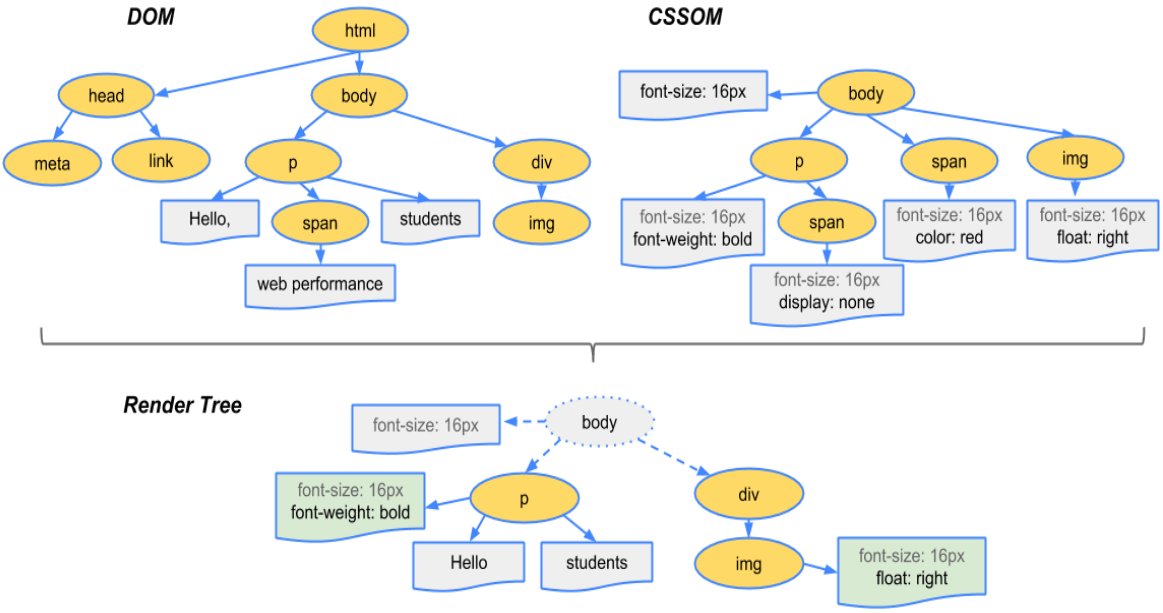


Figura 6. Árbol de visualización (Grigorik, 2015)

Los pasos que ejecuta el navegador para la construcción del árbol de visualización son los siguientes.

- Inicia en la raíz del árbol DOM a través de cada nodo visible.
- Por cada nodo visible, se le asigna su correspondiente regla CSSOM.
- Se obtienen los nodos visibles con sus respectivos estilos.

- **Diseño (*Layout*)**

  Después de generar el árbol de visualización se necesita calcular la posición exacta y tamaño de cada objeto basado en la ventana de visualización del dispositivo.

  El navegador web inicia este proceso desde el objeto principal del árbol y baja por cada nodo, permitiendo con ello que el objeto padre asigne una dimensión de referencia al objeto hijo.

Este proceso genera un “box model”, el cual captura la posición y tamaño exacto de cada elemento.

- **Pintura (*Painting*)**

  El último paso de este proceso es el Painting, el cual se encarga de plasmar el árbol de visualización en pixeles en la pantalla.

1.3.3 Animaciones

Lewis (2016) indicó que la mayoría de los dispositivos actualizan sus pantallas 60 veces por segundo, si es que se presenta algún tipo de animación o transición dentro el navegador, este debe coincidir con la velocidad de actualización del dispositivo para permitir una animación sin retrasos.

Cada uno de estos fotogramas se ejecuta en al menos 16 milisegundos (1 segundo / 60 = 16.66ms), pero solo se tiene 10ms para la ejecución de la animación, ya que dentro de estos 16ms también se cuenta los trabajos internos que debe realizar el navegador. Es por ello que es importante conocer y entender el proceso que implica la ejecución de alguna animación o efecto dentro de una aplicación web. Lewis (2016) se refirió a este proceso como pixel pipeline.

- **Pixel Pipeline**

  Según Lewis (2016) existen 5 procesos claves que forman parte de la ejecución de una animación o efecto.

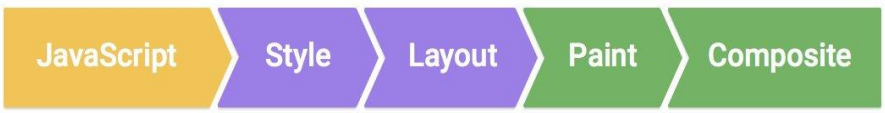


Figura 7. Pixel Pipeline (Lewis, 2016)

- **JavaScript:**

  Hace referencia al iniciador de los cambios visuales dentro de la aplicación, el cual puede implicar ordenar un conjunto de datos y agregar elementos DOM a la página.
- **Estilo (*Style*):**

Es el proceso en el cual el navegador averigua que selector (clase, id, pseudo-selector) es aplicado a un elemento y luego se recalculan y obtienen los nuevos estilos de los elementos.

En el caso de los componentes web, el proceso de cálculo de estilos es diferente, ya que estos poseen una característica denominada shadow DOM, y los estilos no atraviesan el alcance de un componente.

- **Diseño (*Layout*):**

  Si estos estilos tienen alguna implicancia en la geometría de algún elemento del árbol DOM, el navegador debe calcular el espacio necesario para colocar todos los componentes en la pantalla debido a dicho cambio.
- **Pintura (*Paint*):**

  Es el proceso de rellenar los píxeles en la pantalla, para ello usualmente se generan muchas superficies, también conocidas como capas
- **Composición (*Composite*):**

  Es el proceso de ordenar y unir de todas las capas creadas en la pantalla en su orden correcto.

Existen 3 maneras en las cuales se puede ejecutar el pixel pipeline, cada una de ellas depende del escenario y necesidades de la animación o efecto.

- **JS / CSS > Estilo > Diseño> Pintura> Composición**

  Cuando se cambia alguna propiedad diseño. El navegador necesita verificar todos los elementos y hacer un reflujo en toda la página. Cualquier área afectada necesita ser repintada y luego unida con las otras capas.

- **JS / CSS > Estilo > Pintura > Composición**

  Cuando se cambia una propiedad de pintura como por ejemplo color de fondo o texto se omite el proceso layout.

- **JS / CSS > Estilo > Composición**

  Cuando se cambia alguna propiedad que no sea de pintura o diseño, tal y como cursor o transform.

1.3.4 Webgl

Dirksen (2015) mencionó que es una tecnología que une los gráficos 3D con las animaciones web para crear escenarios tridimensionales que se ejecutan directamente en el navegador web.

Los escenarios tridimensionales cuentan con un conjunto de artefactos que permiten que el usuario pueda interactuar y manipular el escenario.

Geometría: Todo objeto que está presente en el escenario tridimensional está basado en un conjunto de vértices que permiten estructurarlo de manera geométrica.

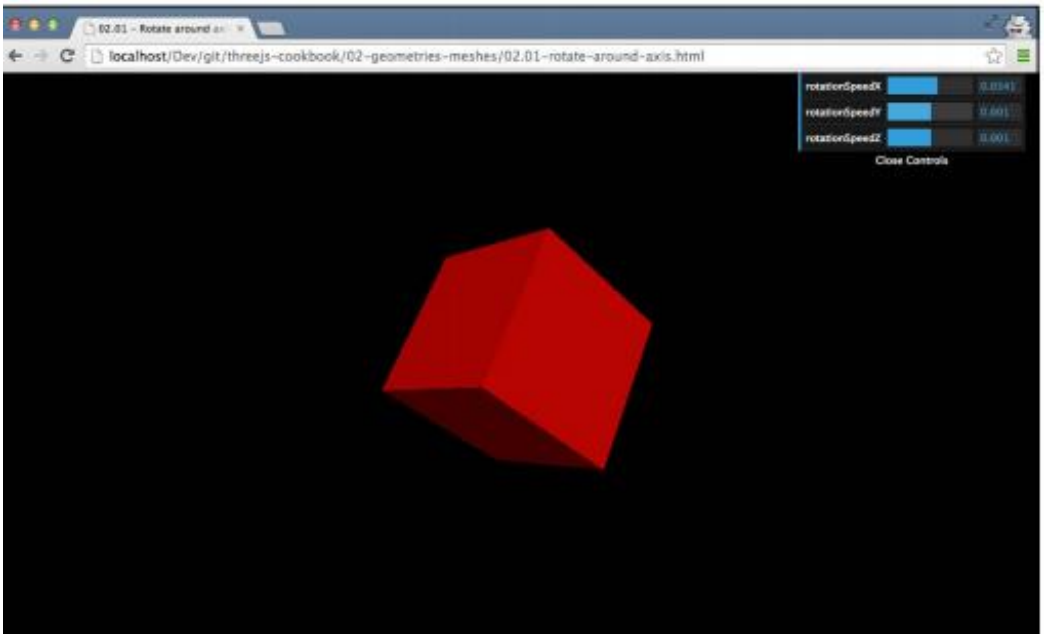


Figura 8. Cubo 3D *(Dirksen, 2015)*

Mientras más vértices tenga una geometría esta será más lisa, fina y detallada, sin embargo, también implica un mayor uso del GPU lo cual se refleja en un impacto en el rendimiento de la aplicación web.

Cámara: La cámara define que sección de la escena será reproducida y como la información será proyectada en la pantalla.

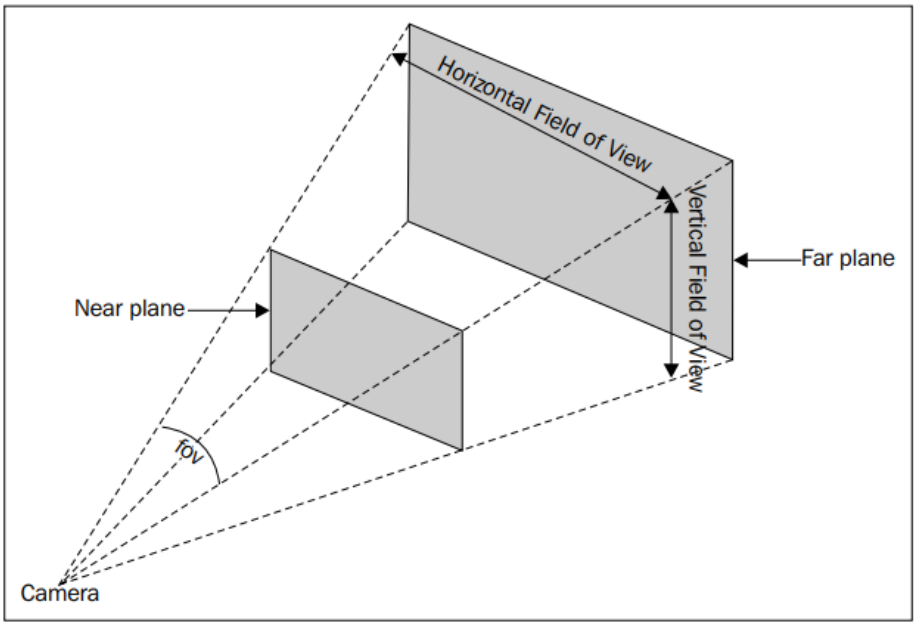


Figura 9. Cámara *(Dirksen, 2015)*

Materiales y textura: Los materiales permiten otorgarle propiedades al objeto 3D tales como brillo, reflexión de luces, color y texturas.

Las texturas son imágenes que recubren a la geometría y permiten aumentar la apercepción de realismo.

Figura 10. Materiales y texturas *(Dirksen, 2015)*

Luces: Las luces permiten agregar un juego de iluminación y sombras que otorgan a la escena tridimensional un mayor realismo.

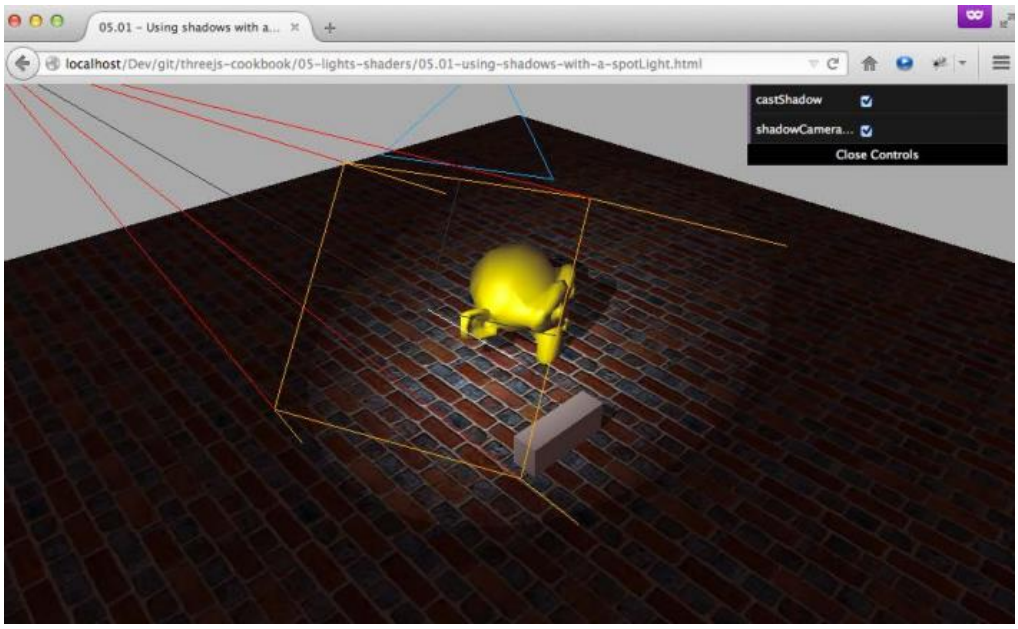


Figura 11. Luces *(Dirksen, 2015)*

1.3.5 RAIL

Lewis (2015) mencionó que RAIL es un modelo de rendimiento centrado en el usuario, el cual permite entender como el usuario percibe el retraso en el rendimiento, para ello se puede ver una vista general de la percepción del usuario y el retraso en las acciones.

Tabla 1

Retraso y reacción del usuario. (Lewis P. I., 2015)

Tiempo milisegundos	Descripción
0 – 16	Dado que la pantalla se actualiza 60 veces por segundo, este fragmento de tiempo representa el tiempo para obtener una imagen individual en la pantalla.
0 – 100	Responder a la acción del usuario en este fragmento de tiempo permite darle la sensación al usuario de un resultado inmediato. Si durara más tiempo la conexión entre la acción y la reacción se rompe.
100 – 300	Los usuarios experimentan un ligero retraso.

300 -1000	En este fragmento de tiempo, las cosas se sienten naturales y fluidas en el proceso de ejecución de la tarea. Para la mayoría de los usuarios en la web, cargar una página o cambiar de vista representa una tarea.
1000+	Luego de 1 segundo, el usuario pierde la concentración en la tarea que ejecuto.
10,000+	El usuario se siente frustrado y es probable que abandone la tarea; puede o no volver más tarde.

RAIL clasifica estos tiempos en 4 categorías diferentes

- **Respuesta**

  Se cuenta con al rededor 100 milisegundos para responder a la interacción del usuario, antes que este sienta un retraso.

- **Animación**

  Toda animación ejecutada en la aplicación debe ser ejecutada a 60 fotogramas por segundo, ello debido a que en la actualidad la mayoría de pantallas se actualiza cada 60 segundos, lo cual deriva a 16 milisegundos para el procesamiento del navegador web y la ejecución del código JavaScript.

- **Inhabilitado**

  Es el tiempo de no interacción por parte del usuario luego de cargar la página, el cual oscila alrededor de los 50 milisegundos en los cuales se puede cargar recursos de manera asíncrona o ejecutar tareas no críticas.

- **Carga**

  El contenido debe ser cargado en menos de 1000 milisegundos, si se sobrepasa dicho tiempo el usuario se distraerá y perderá la intención por ejecutar la tarea.

Tabla 2

Resumen de las métricas principales de RAIL.

RAIL	Métricas clave	Acciones del usuario
Respuesta	Tiempo de respuesta (desde que se presiona hasta que se pinta) < 100ms.	El usuario presiona un icono o botón (por ejemplo, abriendo el menú de navegación).
Respuesta	Tiempo de respuesta (desde que se presiona hasta que se pinta) < 16ms.	El usuario arrastra el dedo y la respuesta de la aplicación está vinculada a la posición del dedo (por ejemplo, tirar para actualizar, deslizar un carrusel).
Animación	Tiempo de respuesta (desde que se presiona hasta que se pinta) < 100ms para la respuesta inicial.	El usuario inicia el desplazamiento de la página o comienza la animación.
Animación	El trabajo de cada fotograma (desde el evento JavaScript al evento Paint) se completa en < 16ms.	El usuario desplaza la página o ve una animación.
Inhabilitado	El hilo principal del código JavaScript trabaja en tiempos menores a 50ms.	El usuario no está interactuando con la página, pero el hilo principal debe estar disponible lo suficiente como para manejar la siguiente entrada del usuario.
Carga	Página considerada lista para usar en 1000ms.	El usuario carga la página y ve el contenido crítico.
	Satisfacer los objetivos de respuesta durante el proceso de carga de página completa.	El usuario carga la página y comienza a interactuar (por ejemplo, desplaza o abre la navegación).

Grigorik (2015), menciona que RAIL no puede basarse en análisis en laboratorios bajo entornos controlados ya que existen muchos factores en el mundo real que pueden llegar a afectar el rendimiento de una aplicación, por ello recomienda recolectar información RUM (Monitoreo de usuarios reales) con la ayuda de las APIs creadas y publicadas por el W3C (Performance Timing, High Resolution Time y Cooperative Scheduling of Background Tasks)

1.3.6 Rendimiento

Firtman (2015) mencionó que antiguamente al hablar de rendimiento web, se referían al tiempo de carga de los recursos y la visualización de estos en la pantalla. Sin embargo, en la actualidad ello ha cambiado y ahora se ve el rendimiento desde la perspectiva del usuario, tal y como lo plantea el modelo RAIL.

Existen muchos factores del dispositivo móvil que suelen relacionarse de manera directa con el rendimiento de una aplicación web, sin embargo, existen dos puntos principales en los cuales estos pueden agruparse:

Hardware

- CPU: Se encarga de analizar, procesar y ejecutar acciones.
- Memoria: Almacena el árbol DOM, las imágenes y datos de descompresión.
- GPU: Se encarga de reproducir las interfaces visuales (también conocido como aceleración de hardware)
- Memoria de GPU: Algunas imágenes y capas son almacenadas cuando el GPU está disponible.

Conexiones a red – Ancho de banda

Tabla 3

Conexiones a red

Velocidades	Mínimo (Megabytes por segundo)	Máximo (Megabytes por segundo)
2G	0.1	0.4
3G	0.5	5
4G	1	50
Wifi	1	100

En la actualidad existen diferentes maneras de analizar el rendimiento e impacto de una aplicación web en un usuario, por ejemplo, herramientas como Chrome Dev Tools y Firefox Dev Tools que permiten dar un seguimiento continuo y detallado de una aplicación web, sin embargo, estos datos no son reflejados necesariamente en

la realidad debido a que los usuarios cuentan con una amplia variedad de dispositivos móviles, con diferentes características de hardware y conexiones de red, por lo cual existe un método de monitoreo de rendimiento denominado RUM.

- **Monitoreo de usuario real (RUM)**

  Molyneaux (2015) se refirió al RUM como un monitoreo pasivo el cual analiza las actividades ejecutadas por el usuario desde el navegador.

  Muchas de las herramientas para RUM son basados en estándares de la W3C, los cuales permiten registrar el ingreso de usuarios, capturar y reportar sesiones, registrar la línea de rendimiento

- **Enfoque para RUM**

a) Iniciar con el cuarto limpio
   Analizar en busca de periodos de pobre rendimiento.
   Comparar los resultados en diferentes navegadores.
b) Enfocarse en áreas con degradación de rendimiento
   Dividir el análisis por página.
   Dividirlo por componentes.
   Comparar el rendimiento relativo a la información base.
c) Revisar la calidad del servicio
   Generar un chat de reporte de errores para el usuario.
   Revisar el reporte generado por el RUM.
d) Capturar, resumir y reportar lo encontrado para futuras comparaciones
   Asegurarse que la información obtenida está guardada y accesible para usarlo como línea base o para propósitos de comparación.

1.3.7 React

Horton (2016) definió a React como un framework basado en componentes autónomos que permite crear vistas compuestas.

Un componente es la representación visual y lógica de una sección de una aplicación web.

```
var HelloReact = React.createClass({
  render: function() {

    var localMessage = this.props.message;

    return (
      <div>
        <div>Hello React</div>
        <div>{localMessage}</div>
      </div>
      );
  }
});
```

Hello React
Message from props and from JSX

Figura 12. Componente

Todos los componentes poseen características especiales que le permiten recibir y enviar el flujo de datos entre diferentes componentes, entre ellos se encuentran:

- **Props**

  Los componentes poseen un conjunto de props, los cuales son los valores recibidos como parámetro al instanciar el componente, estos parámetros son enviados como atributos HTML y son inmutables**.**

- **PropTypes**

  Los proptypes sirven para validar los parámetros enviados a los componentes, si no se cumple la restricción indicada por el proptypes, el navegador web generará una advertencia.

- **Getdefaultprops**

  Los getDefaultProps permiten definir props por defecto dentro de un componente.

- **State**

  Los componentes tienen un concepto denominado state, este permite darle al componente la flexibilidad y dinamismo requerido en toda aplicación web. Cuando un valor del diccionario del state es actualizado el componente entero es producido nuevamente, lo cual es necesario para la carga de registros y

ejecución de animaciones. El state puede instanciarse invocando a la función getInitialState, en ella se inicializa como un tipo de variable diccionario. Los valores definidos en el state son mutables, pueden modificarse y agregar nuevos valores, para ello se puede utilizar la función setState, la cual recibe como parámetro un diccionario, si el dato ya está presente en el state, se actualiza y caso contrario se crea. Los valores del state son de tipo diccionario y se puede acceder a ellos mediante el llamado a su respectiva llave this.state.llave.

Cada componente cuenta con un ciclo de vida que va desde la creación del componente hasta su destrucción, este ciclo de vida está conformado por un conjunto de funciones que permiten la alteración de cualquiera de las fases del ciclo de vida del componente.

- componentWillMount: Este evento es llamado antes que el componente sea montado en el virtual DOM.
- componentDidMount: Este evento es llamado luego que el componente es montado en el virtual DOM.
- componentWillUnmount: Este evento es llamado antes que el componente sea desmontado.
- shouldComponentUpdate: Este evento es llamado antes de re-reproducir el componente, recibe como parámetro los nuevos valores de los props y states y si retorna un valor verdadero el componente será re-reproducido, caso contrario los cambios en los props y states nunca podrán ser reflejados visualmente.

1.3.8 Angular4

Koetsier (2016) definió a Angular4 como un framework MVC de JavaScript que provee un nivel de abstracción que permite el dinamismo en una página web, Angular4 permite extender el código HTML con tags específicos (directivas) para luego manejarlos mediante controladores con el lenguaje Typescript. Asimismo, menciona que la arquitectura del framework está conformada por 8 principales bloques de construcción para cualquier aplicación.

- ***Modules***

  Toda aplicación en Angular4 es un framework modular, por lo cual una aplicación posee al menos un módulo, el módulo principal, aunque usualmente poseen más, tales como los dedicados al flujo de datos, dominio de la aplicación o conjunto de propiedades.

- ***Components***

  Los components plasman la lógica relacionada a la capa vista, otorgándole mayor dinamismo ante la interacción del usuario.

- ***Templates***

  Un template es un tipo de HTML, que le permite a Angular4 describir y definir como se visualizará un componente.

- ***Metadata***

  La metadata permite incluir funcionalidades extras en las clases, permitiendo con ello que una clase se vuelva un componente.

- ***Data binding***

  Es un mecanismo para coordinar partes de un template con partes de un componente.

- ***Directives***

  Las directivas es una clase en forma de atributo HTML, que da las instrucciones de como un template debe visualizar y modificar los elementos del árbol DOM. Existen dos tipos de directivas.

- **Estructuras**: Son aquellas directivas que alteran el diseño del árbol DOM.
- **Atributos**: Altera el comportamiento y apariencia de un elemento existente.

- ***Services***

  Los servicios son cualquier valor, función o característica que requiere una aplicación**.**

- ***Dependency injection***

  La dependency injection permite al framework decidir qué servicio necesita un componente al ser este especificado en su constructor.

1.4 Formulación del problema

1.4.1 Problema Principal

- ¿Cuál es la diferencia de rendimiento, basado en el modelo rail, entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.?

1.4.2 Problemas Específicos

- ¿Cuál es la diferencia en el aspecto “Respuesta” entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.?
- ¿Cuál es la diferencia en el aspecto “Animación” entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.?
- ¿Cuál es la diferencia en el aspecto “inhabilitado” entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.?
- ¿Cuál es la diferencia en el aspecto “carga” entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.?

1.5 Justificación del estudio

Desde la perspectiva tecnológica, este trabajo abarca la evaluación de dos tecnologías relativamente nuevas, ReactJS y Angular4, ambas evaluadas, bajo el modelo RAIL, en una progressive web app. Ello permite a este trabajo aportar al conocimiento de nuevas tecnologías y las diferencias entre estas.

Desde la perspectiva económica, este trabajo permite entender y conocer el comportamiento de las tecnologías ReactJS y Angular4 antes de su aplicación en

futuros proyectos, permitiendo así una posible reducción en los costos de no-calidad del producto software.

Desde la perspectiva teórica, la aplicación del flujo de trabajo para el diagnóstico del rendimiento y su contraste con el modelo de rendimiento RAIL, permite que el presente trabajo se pueda convertir en un marco de referencia para futuras investigaciones relacionadas al tema.

Desde la perspectiva de la implicancia práctica, el trabajo permitirá a los desarrolladores front-end discernir cual tecnologías, ReactJS o Angular4, usar en la creación de una progressive web app bajo un escenario similar.

1.6 Objetivos

1.6.1 Objetivo general

- Determinar la diferencia de rendimiento, basado en el modelo rail, entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A

1.6.2 Objetivos específicos

- Determinar la diferencia en el aspecto “Respuesta” entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.
- Determinar la diferencia en el aspecto “Animación” entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.
- Determinar la diferencia en el aspecto “Inhabilitado” entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.
- Determinar la diferencia en el aspecto “Carga” entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.

II. MÉTODO

2.1 Diseño de investigación.

El diseño de investigación del presente trabajo es de tipo no experimental, ya que como mencionó Hernández (2014) el diseño no experimental es aquel en el cual no existe manipulación de algún tipo por parte del investigador hacia una o más variables de investigación.

Asimismo, ya que el alcance de la investigación es descriptivo, el tipo de diseño no experimental es transeccional descriptivo, ya que como mencionó Hernández (2014) este diseño pretende ubicar en una o más variables un grupo de objetos para con ello proporcionar su descripción.

2.2 Variables, operacionalización

Variable 1: Rendimiento basado en el modelo RAIL.

Tabla 4

Operacionalización de variable.

Variable	Definición conceptual	Definición operacional	Dimensión	Indicadores	Escala de medición
Rendimiento basado en el modelo RAIL	Modelo de rendimiento centrado en el usuario (Lewis P. I., 2015)	Repositorio de datos pertenecientes al monitoreo de los eventos de los usuarios al interactuar con la progressive web app	“Respuesta” (Lewis P. I., 2015)	Tiempo de respuesta ante la acción "Ir a demo" (Lewis P. I., 2015)	De razón
				Tiempo de respuesta ante la acción "Ir a probador" (Lewis P. I., 2015)	De razón
				Tiempo de respuesta ante la acción "Ir a Home desde Probador principal" (Lewis P. I., 2015)	De razón
				Tiempo de respuesta ante la acción "Ir a Probador principal desde probador mujer" (Lewis P. I., 2015)	De razón
				Tiempo de respuesta ante la acción "Ir al probador detalle" (Lewis P. I., 2015)	De razón
				Tiempo de respuesta ante la acción "Seleccionar prenda" (Lewis P. I., 2015)	De razón
				Tiempo de respuesta ante la acción "Ver detalle de la prenda" (Lewis P. I., 2015)	De razón
			“Animación” (Lewis P. I., 2015)	Fotogramas perdidos en la página probador (Lewis P. I., 2015)	De razón
				Fotogramas perdidos en la página demo (Lewis P. I., 2015)	De razón
			“Inhabilitado” (Lewis P. I., 2015)	Cantidad de tiempos Inhabilitados disponibles en Catalogo 3D (Lewis P. I., 2015)	De razón
				Cantidad de tiempos Inhabilitado disponibles en Demo (Lewis P. I., 2015)	De razón
				Cantidad de tiempos Inhabilitado disponibles en Descripción (Lewis P. I., 2015)	De razón

				Cantidad de tiempos Inhabilitado disponibles en Home (Lewis P. I., 2015)	De razón
				Cantidad de tiempos Inhabilitado disponibles en Probador/Maniquí (Lewis P. I., 2015)	De razón
			“Carga” (Lewis P. I., 2015)	Tiempo de carga de la página Home (Lewis P. I., 2015)	De razón

2.3 Población y muestra

2.3.1 Población

Para esta investigación, la población estará conformada por los registros de interacción de los usuarios en las páginas web desarrolladas en Angular4 y ReactJS.

2.3.2 Muestra

Para esta investigación, la población estará conformada por los registros de interacción de los usuarios en las páginas web, desarrolladas en Angular4 y ReactJS, generados entre de 26 mayo del 2017 al 30 de junio del 2017.

2.3.3 Muestreo

En esta investigación se usará un muestreo por conveniencia ya que como mencionó Hernández (2014) es una elección cuidadosa y controlada de casos que poseen ciertas características determinadas en el planteamiento del problema.

2.4 Técnicas e instrumentos de recolección de datos, validez y confiabilidad

En esta investigación la técnica usada será la de la observación, ya que según Hernández (2014) permite un registro sistemático, válido y confiable respecto a un comportamiento o situaciones observables, con lo cual se requiere usar como instrumento una guía / ficha de observación.

Los registros de monitoreo de eventos fueron obtenidos desde la interacción del usuario con la progressive web app y fueron procesados con la herramienta SPSS statisics 23.

2.5 Aspectos éticos

Todos los datos fueron recopilados de manera científica, haciendo uso de las herramientas proporcionadas por el consorcio internacional W3C. Asimismo el investigador se compromete a respetar la veracidad de los datos registrados y sus respectivos resultados.

Este trabajo contó con la autorización de la empresa Glup S.A. para poder hacer uso de su aplicación web como la base principal para esta investigación.

III. RESULTADOS

Los datos fueron recopilados usando la herramienta ficha de observación, la cual posee información relacionada al rendimiento de los dispositivos móviles que accedieron a la página m.glup.com.pe durante un periodo de tiempo: Del 26 mayo del 2017 al 30 de junio de 2017 (18 días en Angular4 y 18 días en React).

3.1 Dimensión - Respuesta

Tabla 5

Estadísticas descriptivas para la respuesta.

	Angular4	React
N	749	749
Media	74.38791055	154.77720206
Desv. Tip.	371.635527799	1295.367600843
Asimetría	8.994	18.965
Curtosis	94.188	409.494

La tabla 5 muestra que la tecnología React demora en promedio 80.38929151 milisegundos más que Angular4 en responder a una acción ejecutada por un usuario, y que Angular4 logar cumplir con las métricas indicadas por el modelo RAIL al tener un valor promedio menor a 100 milisegundos.

La tabla 5 también muestra que los tiempos de respuesta de la tecnología React tienen una mayor concentración de valores en la región central de la distribución (Leptocúrtica) a comparación de Angular4, sin embargo, el valor asimétrico de Angular4 indica que existen más registros cercanos a la media que en React.

Tabla 6

Promedio duración de eventos por páginas

Acciones	Angular4	React
Ir a demo	29.41328829	20.998018
Ir a Home desde Probador principal	9.17880597	88.8514179
Ir a probador principal desde Home	9.45628788	11.4983992
Ir a Probador principal desde probador mujer	44.35637363	87.4502198
Ir al probador detalle	8.52960396	11.3901485
Ocultar prendas	213.165	511.057978
Seleccionar prenda	307.7079518	324.096687
Ver detalle de la prenda	12.7887234	180.091099

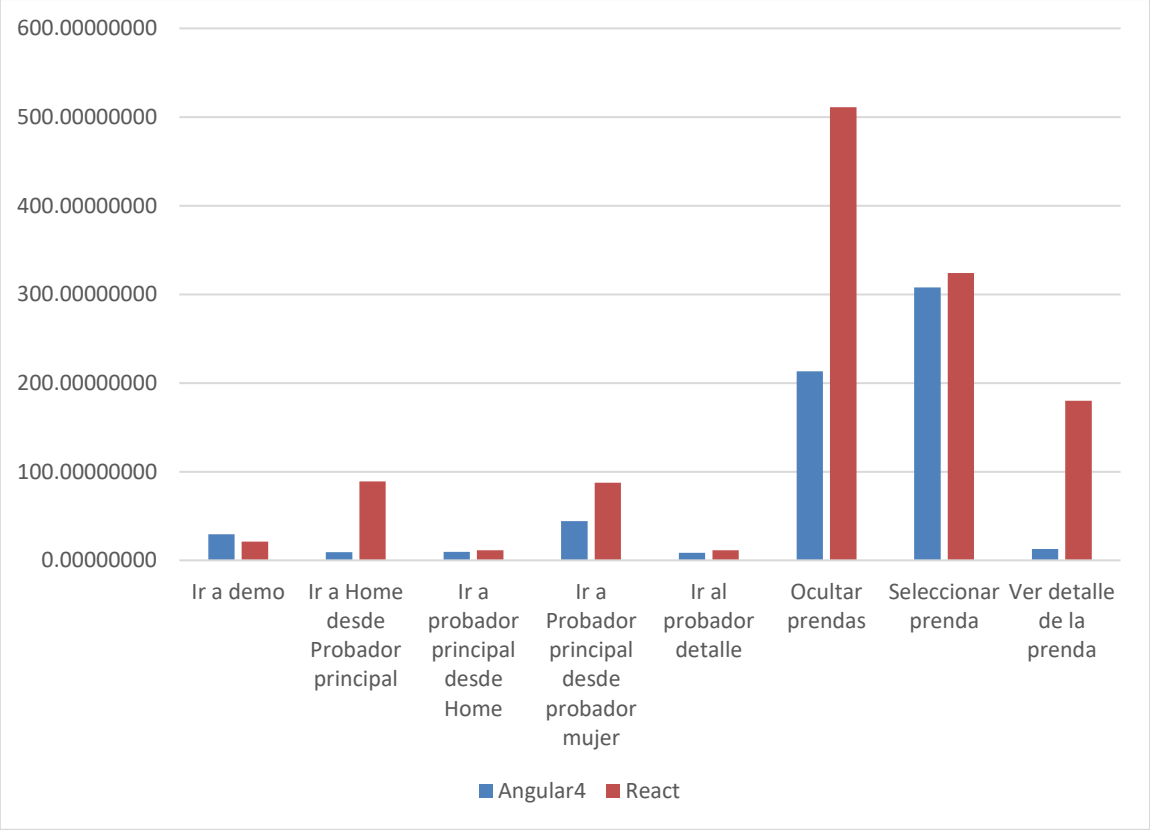


Figura 13. Tiempos promedio de respuesta de acciones

La tabla 6 junto con la figura 13 muestra que React y Angular4 sobrepasan los 100 milisegundos de tiempo de respuesta en las acciones “Seleccionar prenda” y

“Ocultar prendas” de la página de “Probador”, lo cual implica que en estas acciones ninguno de las dos tecnologías logra mantener la percepción de una respuesta instantánea ante una acción y esto es debido a que la página en la cual estas acciones son ejecutadas, “Probador”, reproduce un escenario tridimensional con varias geometrías, para lo cual requiere el consumo de muchos recursos del dispositivo móvil. De la misma manera se aprecia que la principal diferencia entre React y Angular4 se da en la ejecución del evento “Ver detalle de la prenda” en la página “Demo”, en la cual se reproduce un escenario tridimensional con una sola geometría.

3.2 Dimensión - Animación

Tabla 7

Estadísticas descriptivas para la animación.

	Angular4	React
N	585	585
Media	181.24	295.54
Desv. Tip.	457.105	1399.435
Asimetría	4.542	14.935
Curtosis	30.427	284.275

La tabla 7 muestra que React pierde un promedio de 114.3 fotogramas más que Angular4, lo cual implica que Angular4 permite ejecutar animaciones con menos retardo visual que React.

La tabla 7 también muestra que los fotogramas perdidos por la tecnología React tienen una mayor concentración de valores en la región central de la distribución (Leptocúrtica) a comparación de Angular4, sin embargo, el valor asimétrico de Angular4 indica que existen más registros cercanos a la media que en React.

Tabla 8

Promedio de Fotogramas perdidos por página

Páginas	Angular4	React
Demo	605.58	529.41
Probador /Maniquí	291.25	622.64

La tabla 8 muestra que el déficit en la animación por parte de React ocurre en mayor cantidad en la página de “Probador”.

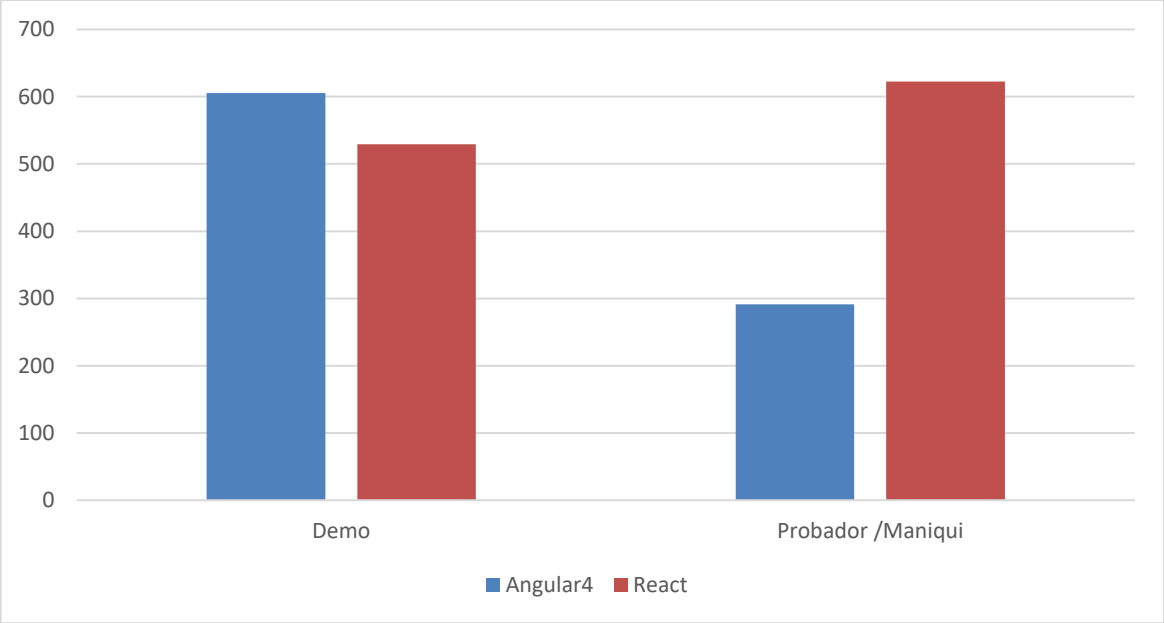


Figura 14. Promedio de Fotogramas perdidos en Angular4 y React

La figura 14 muestra que React, a comparación de Angular4, pierde más fotogramas en la página “Probador/Maniquí”.

3.3 Dimensión - Inhabilitado

Tabla 9

Estadísticas descriptivas para el tiempo de inactividad

	Angular4	React
N	894	894
Media	160623.66	200496.15
Desv. Tip.	1500863.008	3535989.747
Asimetría	13.343	29.127
Curtosis	197.721	862.281

La tabla 9 muestra que React permite, en promedio, 39872.49 tiempos de inactividad más que la tecnología Angular4, lo cual implica que React, a comparación de Angular4, puede ejecutar más tareas en paralelo que no perjudiquen la fluidez de la navegación en el sitio web.

La tabla 9 también muestra que los tiempos de inactividad de la tecnología React tienen una mayor concentración de valores en la región central de la distribución (Leptocúrtica) a comparación de Angular4, sin embargo, el valor asimétrico de Angular4 indica que existen más registros cercanos a la media que en React.

Tabla 10

Promedio de cantidad de tiempos de inactividad por página

Páginas	Angular4	React
Catalogo 3D	35258.61	628438.20
Demo	349104.16	160469.63
Home	262565.89	89489.97
Probador/Maniquí	44611.55	85075.53

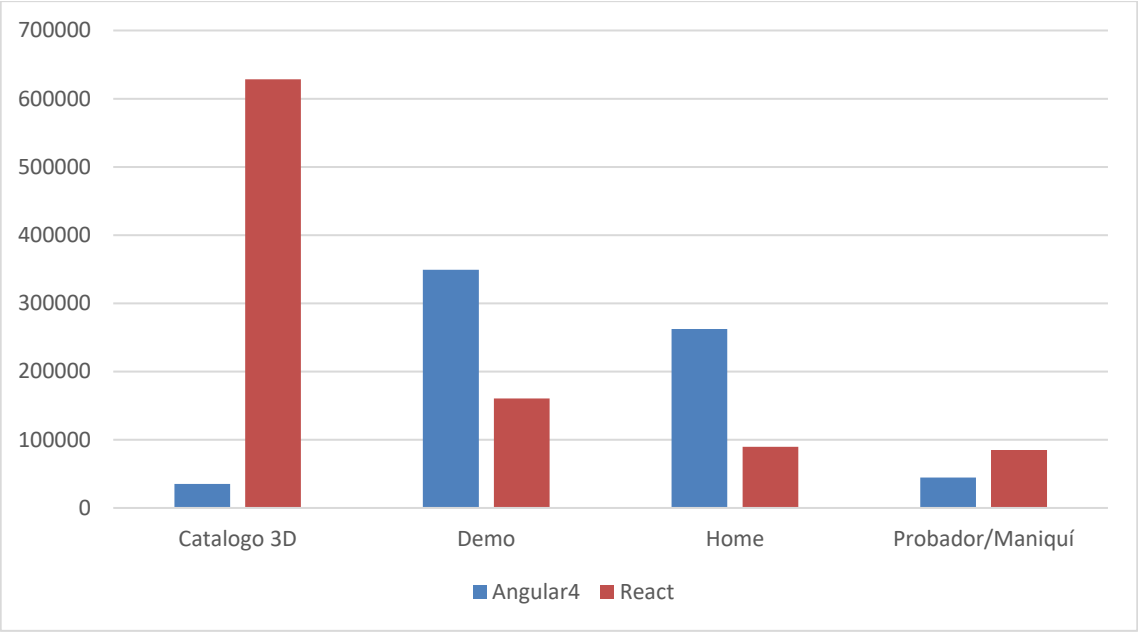


Figura 15. Promedio de cantidad de tiempos de inactividad por página

La figura 15 muestra que la principal diferencia de tiempos de inactividad entre React y Angular4 se encuentra en la página “Catálogo 3D” y que en las páginas “Demo” y “Home” Angular4 logra superar a React por más del doble.

3.4 Dimensión - Carga

Tabla 11

Estadísticas descriptivas para el tiempo de carga

	Angular4	React
N	190	190
Media	11289.23	4373.53
Desv. Tip.	40769.515	6697.070
Asimetría	9.333	4.890
Curtosis	97.785	29.508

La tabla 11 muestra que la tecnología React demora en promedio 6915.7 milisegundos menos que Angular4. Lo cual implica que Angular4 aumenta las

posibilidades de que el usuario se retire de la página web antes de que esta termine de cargar por completo.

La tabla 11 también muestra que los tiempos de carga de la tecnología Angular4 tienen una mayor concentración de valores en la región central de la distribución (Leptocúrtica) a comparación de React, sin embargo, el valor asimétrico de React indica que existen más registros cercanos a la media que en Angular4.

Tabla 12

Promedio de tiempo de carga por página

Páginas	Angular4	React
Home	11289.23	4373.53

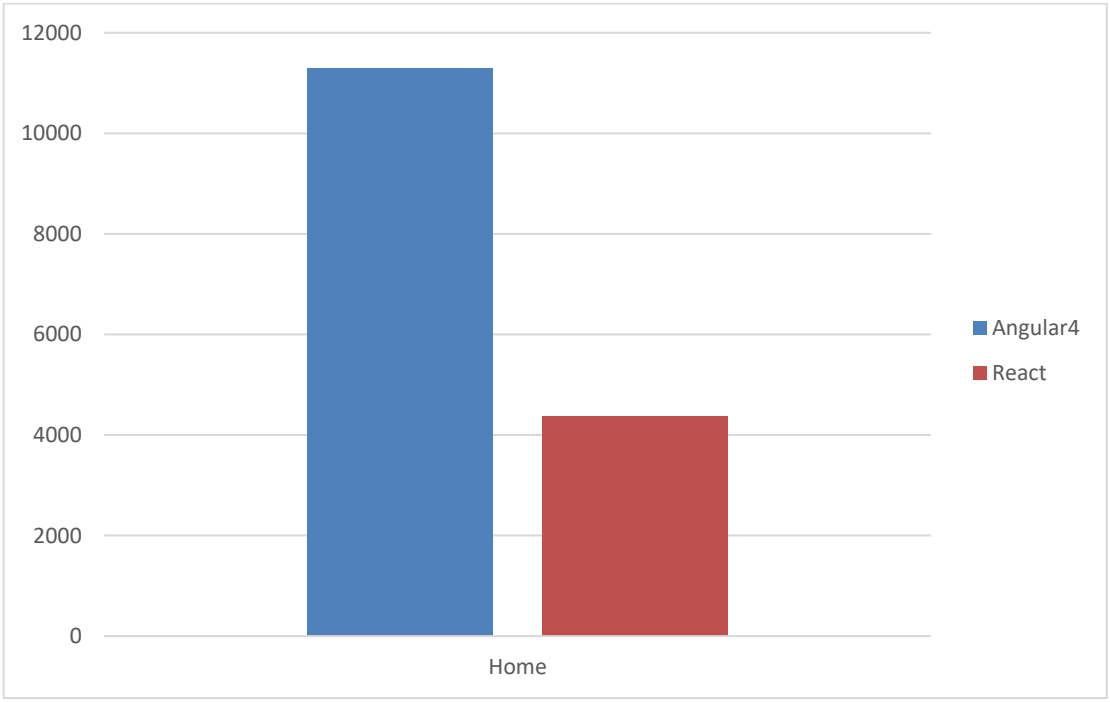


Figura 16. Promedio de tiempo de carga por página

La tabla 12 y figura 16 muestra que Angular4 y React sobrepasan los 1000 milisegundos indicados por RAIL para la carga de una página web, sin embargo, Firtman (2015) mencionó que la velocidad de carga depende mucho del tipo de

conexión de red, por lo cual no se puede determinar con veracidad la diferencia entre Angular4 y React en el tiempo de carga.

IV. DISCUSIÓN

La dimensión respuesta fue evaluada al comparar el tiempo de respuesta ante las acciones ejecutadas por el usuario en la página web. Los resultados de la investigación referente a esta dimensión, muestran que Angular4 permite dar una respuesta más rápida que React.

Estos resultados son similares a los resultados del estudio de Molin (2016) en el cual mostró que Angular toma una ventaja de más del 300% sobre React en cuanto al tiempo de carga. Además, los resultados difieren a los obtenidos por Koetsier (2016) en el cual indicó que Angular otorga un mejor rendimiento en una aplicación web, sin embargo, los resultados muestran que el tiempo de respuesta promedio de Angular4 sobrepasa el umbral de 50 milisegundos que indica RAIL es necesario para otorgar un buen rendimiento al usuario.

La dimensión animación fue evaluada al comparar la cantidad de fotogramas perdidos en las animaciones creadas para en WebGL (un fotograma se considera perdido si la actualización del árbol DOM demora más de 16 milisegundos). Los resultados de la investigación referente a esta dimensión, muestran que Angular4 pierde menos fotogramas que React.

Estos resultados son similares a los resultados del estudio de Molin (2016) en el cual indicó que Angular permite una actualización del árbol DOM más rápida que React.

La dimensión inhabilitado fue evaluada al comparar la cantidad de tiempos inhabilitados disponibles en las páginas web. Los resultados de la investigación referente a esta dimensión, muestran que React permite más tiempos de inhabilitado en una página web que Angular4, lo cual implica que React requiere una menor cantidad de memoria en la ejecución de tareas.

Estos resultados son similares a los resultados del estudio de Molin (2016) en el cual se concluyó que Angular requiere un uso más alto de memoria para ejecutar las tareas, dejando poco espacio para la existencia de tiempos inhabilitados.

La dimensión carga fue evaluada al comparar la duración del tiempo de carga de las páginas web. Los resultados de la investigación referente a la dimensión tiempo de carga, muestran que React permite cargar una página web más rápido que Angular4.

Estos resultados son similares a los resultados del estudio de Molin (2016) el cual concluyó que Angular demora más tiempo que React en cargar una página web, debido a que Angular se compila en un paquete más pesado. Asimismo, los resultados son diferentes a los obtenido por Koetsier (2016) en el cual indicó que Angular otorga un mejor rendimiento en una aplicación web, ya que esta investigación hace hincapié en el cumplimiento de las cuatro dimensiones RAIL para poder asegurar que una tecnología JavaScript permite un rendimiento ideal.

V. CONCLUSIONES

Las conclusiones de esta investigación fueron las siguientes:

1. Angular 4 posee un mejor tiempo de respuesta ante un evento a comparación de React, con una diferencia promedio de 51.834 milisegundos. Lo cual según Lewis (2015) nos permite concluir que Angular4 permitirá mantener enfocado al usuario en la acción que ha ejecutado, a comparación de React el cual llega a demorar hasta 1000 milisegundos en promedio. Por lo cual es recomendable usar Angular4 en proyectos web en los cuales se requiera una respuesta rápida, por ejemplo, videojuegos. Asimismo, los registros indican que los registros de tiempos de respuesta obtenidos de Angular4 son más homogéneos que en React y por lo tanto permitirá realizar un análisis estadístico más eficiente.
2. Angular 4 posee una mejor fluidez y desempeño en las animaciones que React, con una diferencia promedio de 205.74 fotogramas que no lograron ejecutarse en el umbral de 16 milisegundos. Esto según Lewis (2015) nos permite concluir que las animaciones creadas por Angular4 poseerán menos retardo que las creadas con React. Por lo cual es recomendable usar Angular4 en proyectos web en los cuales se haga uso constante de animaciones, por ejemplo, webGL o videojuegos. Asimismo, los registros indican que los registros de la cantidad de fotogramas perdidos obtenidos de Angular4 son más homogéneos que en React y por lo tanto permitirá realizar un análisis estadístico más eficiente.
3. React permite mayores tiempos de inactividad a comparación de Angular 4, con una diferencia promedio de 39872.49. Esto según Lewis (2015) permite concluir que React tiene mejor capacidad para ejecutar trabajos en paralelo sin afectar el rendimiento de la aplicación web. Por lo cual es recomendable usar React en proyectos web que requieran ejecución de múltiples tareas en corto tiempo, por ejemplo, sistemas transaccionales, paneles de control, CRMs, ERPs, etc. Asimismo, los registros indican que los registros de cantidad de tiempos de inactividad obtenidos de Angular4 son más homogéneos que en React y por lo tanto permitirá realizar un análisis estadístico más eficiente.

4. React permite una rápida carga de la página web a comparación de Angular4, con una diferencia promedio de 7266.62 milisegundos. Esto Lewis (2015) permite concluir que React genera que el usuario no pierda el interés por ejecutar una tarea y según Suresh (2017) existe una alta posibilidad de que el usuario permanezca y regrese a la página web. Sin embargo, Firtman (2015) mencionó que la velocidad de carga depende mucho del tipo de conexión de red, por lo cual no se puede determinar con veracidad la diferencia entre Angular4 y React en el tiempo de carga. Asimismo, los registros indican que los registros de tiempos de carga obtenidos de React son más homogéneos que en Anguar4 y por lo tanto permitirá realizar un análisis estadístico más eficiente.

VI. RECOMENDACIONES

Luego del desarrollo del trabajo de investigación, se recomienda para futuras investigaciones:

- En la actualidad la tecnología web puede usarse para la creación de aplicaciones móviles, con ayuda de herramientas como Ionic o Phonegap, por lo cual se recomienda evaluar las diferencias de rendimiento bajo el modelo RAIL de estas tecnologías en aplicaciones hibridas para Android y IOS.

- Wonsun (2014) indicó que el uso de un determinado compilador JavaScript altera el rendimiento final de una aplicación web, por lo cual se recomienda para futuras investigaciones evaluar diferentes compiladores para cada tecnología y ver qué efecto tienen estos en su rendimiento y posteriormente compararlos.

- Recolectar información durante un periodo de tiempo más largo lo cual permita hacer una valoración más acertada referente a las tecnologías Angular4 y React.

- Hacer un seguimiento de las regiones de las cuales provienen las métricas de rendimiento, esto es debido a que cada región o país tiene una economía y poder adquisitivo diferente del otro, lo cual implica que la población posea dispositivos con un hardware más avanzado y con mejor rendimiento.

- Firtman (2015) indicó que existen muchos factores que pueden alterar los resultados obtenidos referentes al tiempo de carga, por lo cual en este trabajo no se pudo obtener un resultado concluyente referente a ello, por lo cual se recomienda fabricar estrategias que permitan determinar la diferencia en el tiempo de carga ignorando las posibles dificultades en las conexiones de red.

REFERENCIAS

Alex, N. (2013). Best Practices on the Move: Building Web Apps for Mobile Devices. *ACM Queue, 11*(6). Recuperado el 20 de diciembre de 2016

Andrew S., J. G. (2015). *Learning Agile.* Estados Unidos: O'Reilly Media. Obtenido de https://goo.gl/fXJq4S

Angular. (s.f.). *Arquitectura*. Recuperado el 14 de noviembre de 2016, de https://goo.gl/LEBRNj

Beck, K. (2004). Extreme Programming Explained. Nueva York, Estados Unidos. Obtenido de https://goo.gl/YUTRUr

Behar, D. (2008). *Metodología de la Investigación* (1 ed.). Barcelona: Shalom.

David, S. (31 de mayo de 2016). Influence of Mobile-friendly Design to Search Results on Google Search. *Procedia - Social and Behavioral Sciences, 220*(1), 424-433. Recuperado el 11 de abril de 2017, de https://goo.gl/bsG8Rd

Dirksen, J. (2015). *Three.js Cookbook.* Birmingham, Reino Unido: Packt Publishing. Obtenido de https://www.packtpub.com/web-development/threejs-cookbook

EMMIT A. SCOTT, J. (2016). *SPA Design and Architecture* (1 ed.). Nueva York, Estados Unidos: Manning. Obtenido de https://goo.gl/XAtpmX

Firtman, M. (2015). *High Performance Mobile Web* (1 ed.). Estados Unidos: O'Reilly Media. Obtenido de https://goo.gl/cXg6tm

Google Developers. (2016). *Case Studies*. Recuperado el 6 de septiembre de 2016, de https://goo.gl/rUR94K

Grigorik, I. (2015). Quantify and improve real-world RAIL. California. Obtenido de https://goo.gl/DRcxQi

Grigorik, I. (s.f.). *Critical rendering path.* (G. Inc, Ed.) Recuperado el 18 de noviembre de 2016, de https://goo.gl/hq9mxr

Hernández S., F. C. (2014). *Metodología de la investigación científica* (6 ed.). México DF, México: McGraw-Hill.

Horton A., V. R. (2016). *Mastering React.* Birmingham, Reino Unido: Packt Publishing. Obtenido de https://goo.gl/4nbG7F

Hunt, P. (2016). *ReactJS Blueprints.* Birmingham, Reino Unido: Publishing, Packt. Recuperado el 1 de julio de 2016, de https://goo.gl/e729oa

INRA. (s.f.). Recuperado el 22 de abril de 2008, de https://goo.gl/jEZeJH

James S., S. W. (2008). *The Art of Agile Development.* Nueva York, Estados Unidos: O'Reilly Media. Obtenido de https://goo.gl/VAhC4n

Jan, K. (2013). Using Speculation to Enhance JavaScript Performance in Web Applications. *IEEE Communications Magazine, 17*(2). Recuperado el 17 de marzo de 2017, de http://ieeexplore.ieee.org/document/6383150/

Join Alex R. & Bovens, A. (2015). Progressive Web Apps. *Chrome Dev Summit 2015*. California. Obtenido de https://goo.gl/18ERXE

Koetsier, J. (2016). Evaluation of JavaScript frameworks for the development of a web-based user interface for Vampires. *Universidad de Van Amsterdam*. Recuperado el 23 de octubre de 2016, de https://goo.gl/XC4BHB

LePage, P. (2016). *Your First Progressive Web App.* (G. Inc., Ed.) Recuperado el 20 de noviembre de 2016, de https://goo.gl/bs5am8

Lewis, P. (2016). *RAIL in the real world.* (G. Inc., Ed.) Recuperado el 29 de noviembre de 2016, de https://goo.gl/YCPofV

Lewis, P. (2016). *Rendering Performance.* (G. Inc., Ed.) Recuperado el 17 de agosto de 2016, de https://goo.gl/AF2mak

Lewis, P. I. (2015). Introduction to RAIL. (G. Inc., Ed.) Recuperado el 28 de agosto de 2015, de https://www.youtube.com/watch?v=wO9GGY17NXY&t=299s

Love, R. (2007). Linux System Programming. Nueva York, Estados Unidos: O'Reilly Media. Obtenido de https://goo.gl/PgLsNq

Mariano, C. (2017). Benchmarking JavaScript Frameworks. *Instituto tecnológico de Dublin*.

Molin, E. (2016). Comparison of Single-Page Application Frameworks. *Instituto de tecnología ROYAL KTH*.

Molyneaux, I. (2015). The Art of Application Performance Testing: From Strategy to Tools. Manchester, Reino Unido. Obtenido de https://goo.gl/GfKiHQ

Myeongjin, C. (2015). O2WebCL: an automatic OpenCL-to-WebCL translator for high performance web computing. *Journal of Supercomputing, 71*(6). Recuperado el 27 de noviembre de 2016, de https://link.springer.com/article/10.1007/s11227-014-1260-4

Nagele, T. (2015). Client-side performance profiling of JavaScript for web applications. *Universidad de Radbound*. Recuperado el 10 de noviembre de 2016, de https://goo.gl/xgTjbd

Nygard, K. (2016). Single page architecture as basis for web applications. *Universidad Aalto*. Recuperado el 1 de diciembre de 2016, de https://goo.gl/dRaKhW

Osmani, A. (2016). *Getting Started with Progressive Web Apps.* (G. Inc., Ed.) Recuperado el 1 de noviembre de 2016, de https://goo.gl/XQUi92

Osmani, A. (2016). *Instant Loading Web Apps with an Application Shell Architecture.* (G. Inc, Ed.) Recuperado el 18 de noviembre de 2016, de https://goo.gl/qVWVLw

Patrick. (2013). Web site performance data has never been more readily available. *ACM Queue, 11*(2). Recuperado el 17 de junio de 2017, de http://queue.acm.org/detail.cfm?id=2446236

Pete, H. (5 de septiembre de 2016). React: Facebook's Functional Turn on Writing JavaScript. *ACM Queue, 14*(4). Obtenido de http://queue.acm.org/detail.cfm?id=2994373

Phuc Tran, D. (2016). Design and Implement Scalable Robust Modern Web Application. *Universidad Helsinki Metropolia of Applied Sciences*. Recuperado el 15 de noviembre de 2016, de https://goo.gl/9DvAwg

React. (s.f.). Recuperado el 1 de julio de 2016, de https://goo.gl/e729oa

Rijwan, K., & Mohd, A. (2016). Performance testing (load) of web applications based on test case management. *In Recent Trends in Engineering and Material Sciences, 8*(1). Recuperado el 4 de abril de 2017, de http://www.sciencedirect.com/science/article/pii/S2213020916300957

Sanchez, S. (10 de noviembre de 2016). Comparison of performance between a native app and a mobile web application for monitoring a photovoltaic system. *Sistemas & Telemática, 14*(39), 29-40. doi:10.18046/syt.v14i39.2347

Simpson, K. (2015). You Don't Know JS: Async & Performance. Nueva York, Estados Unidos: O'Reilly Media. Obtenido de https://goo.gl/s29p8T

Sousana, C. d. (2015). Pro React. Nueva York, Estados Unidos: Apress. Obtenido de https://goo.gl/3wiiBT

Suresh, K. S. (2017). Performance Driven Development Framework for Web Applications. *Global Journal of Enterprise Information System*(19), 75-84. Recuperado el 20 de junio de 2017, de https://goo.gl/pMiWSJ

Tammy Everts, R. (2013). Rules for Mobile Performance Optimization. *ACM Queue, 11*(6). Recuperado el 12 de mayo de 2017, de http://queue.acm.org/detail.cfm?id=2510122

Terukina, I. (2005). El Fotograma es una Imagen Estática... sin embargo, se Mueve. *Razón y Palabra*(16). Recuperado el 19 de diciembre de 2016, de https://goo.gl/TV95GR

Troy, J. (2014). Desktop and mobile web page comparison: characteristics, trends, and implications. *IEEE Communications Magazine, 52*(9). Recuperado el 22 de mayo de 2017

Ulan, G. (2016). Idle-Time Garbage-Collection Scheduling. *ACM Queue, 14*(3). Recuperado el 30 de abril de 2017, de http://queue.acm.org/detail.cfm?id=2977741

Umar, K. (2014). Architectural Pattern for Improving Performance of Web Applications. *International Journal of Computer Science, 11*(2). Recuperado el 15 de julio de 2017, de http://airccse.org/journal/ijwest/papers/5314ijwest01.pdf

Victor, S. (2017). *Angular Router.* Birmingham, Reino Unido: Packt Publishing.

Vipul, A. (2016). ReactJS by Example – Building Modern Web Applications. Birmingham, Reino Unido: Packt Publishing. Obtenido de https://goo.gl/fY4usG

W3C. (2017). *IndexedDB*. Recuperado el septiembre de seiptiembre de 2016, de https://goo.gl/f8Gw3g

Willian, J. (2013). Native Apps vs. Mobile Web Apps. *International Journal of Interactive Mobile Technologies, 7*(4). Recuperado el 11 de mayo de 2017, de http://online-journals.org/i-jim/article/view/3226

Wonsun, A. (2014). Improving JavaScript performance by deconstructing the type system. *ACM / SIGPLAN Notices, 49*(6). Recuperado el 2 de diciembre de 2016

World Wide Web Consortium. (s.f.). *Introduction.* Recuperado el 3 de febrero de 2017, de https://goo.gl/ae2ULo

Yuhao, Z. (2017). Optimizing General-Purpose CPUs for Energy-Efficient Mobile Web Computing. *ACM, 35*(1). Recuperado el 12 de junio de 2017, de http://dl.acm.org/citation.cfm?id=3041024

Yuhao, Z. (2017). The Red Future of Mobile Web Computing. *ACM Queue*(19), 75-84. Recuperado el 20 de junio de 2017, de https://goo.gl/pMiWSJ

ANEXOS

ANEXO “A” GLOSARIO DE TÉRMINOS

Aplicación JavaScript Isomorfa

Sousana (2015) mencionó que una aplicación JavaScript isomorfa es aquella que ejecuta su código JavaScript en el servidor para que cuando la aplicación se cargue en el navegador el contenido sea visible de manera casi instantánea y no deba esperar por la descarga de archivos y ejecución de solicitudes asíncronas al servidor.

Carga lenta

La carga lenta, también conocido por su versión en inglés Lazy Loading es una técnica la cual según Victor (2017) permite mejorar el tiempo de carga de una aplicación al dividir el paquete principal JavaScript que este posee en múltiples paquetes los cuales son descargados según la página a la cual se acceda.

Virtual DOM

Vipul (2016) se refirió al virtual DOM como la tecnología usada por ReactJS para manipular el árbol DOM. React mantiene una copia del árbol DOM en memoria, cuando se debe realizar alguna manipulación de sus componentes, ReactJS genera una nueva copia con los nuevos cambios ya aplicados, luego ReactJS busca las diferencias entre el virtual DOM y la copia, y son esas mínimas diferencias obtenidas las que finalmente se aplican al árbol DOM original.

Framework

EMMIT A. SCOTT (2016) mencionó que un framework es un conjunto de utilidades que permiten a una aplicación, manejar y administrar la comunicación entre las diferentes capas (secciones) que involucran una aplicación, las cuales son:

- La capa relacionada a la vista
- La capa relacionada a la información y su manejo
- La capa relacionada a la lógica de funcionamiento de los componentes de la UI.

MVC

EMMIT A. SCOTT (2016) se refirió a MVC como una arquitectura que permite realizar una separación de los componentes principales en diferentes capas.

- Modelo

  El modelo usualmente contiene la información, la lógica de negocio y la validación lógica de una aplicación web.
- Vista

  La vista es con lo que el usuario ve e interactuará. Es la representación visual de la información del modelo.
- Controlador

  Se encarga de recibir la señal emitida desde la UI al ejecutar una acción, ejecuta algún proceso relacionado al evento, y manda la acción hacia la capa modelo.

Fotograma

Terukina (2005) mencionó que un fotograma es una imagen visual estática, que, en una rápida sucesión, genera la ilusión de movimiento o animación.

API

Love (2007) mencionó que un API es una pieza de software que define la conexión de comunicación con otra pieza de software y le proporciona un funcionamiento extra.

HTML

Nygard (2016) se refirió a HTML como un lenguaje de marcado usado en toda página web, la cual permite definir y describir el documento mostrado por el navegador web.

CSS

Nygard (2016) mencionó que CSS (Cascading Style Sheets) es un lenguaje que permite estilizar una página web, dándole el diseño a la estructura y elementos de definidos en el lenguaje HTML.

JavaScript

Simpson (2015) se refirió a JavaScript como la tecnología (lenguaje de programación) que se encarga de la interacción y dinamismo en una página web.

Caso de uso

Molyneaux (2015) mencionó que un caso de uso es un conjunto de transacciones que representan la actividad típica de una aplicación, tales como pulsar sobre un botón, ingresar un texto para realizar una búsqueda, iniciar sesión, cerrar sesión, etc.

Captura de eventos de usuario

Según Molyneaux (2015), capturar eventos de usuario es la capacidad de registrar el tiempo entre dos eventos, ejemplo (evento click del mouse, scrolling, etc).

World Wide Consortium (W3C)

En Consortium (2016) se refirió al W3C como una comunidad internacional que trabaja en conjunto para crear estándares web.

Velocidad

James S. (2008) mencionó que la velocidad es el total de estimaciones (duración) para la culminación de un conjunto de historias en una iteración.

Historia de usuario

Según Andrew S. (2015), una historia de usuario es una herramienta rápida que permite una descripción específica y simple de lo que el usuario quiere del software. La mayoría de las historias tienen entre una y cuatro oraciones de largo.

Plantilla: Como un <tipo de usuario>, quiero <especificar acción> así que <que es lo que se quiere como resultado>

Ilustración 1 - Historia de usuario (Andrew S., 2015)

Cada historia puede ser contrastada con ayuda de las condiciones de satisfacción, las cuales son plasmadas en la parte posterior de la tarjeta de la historia de usuario.

Aplicación de una sola página

Una aplicación de una sola página o conocida también en inglés como Single Page Application (SPA), es según EMMIT A. SCOTT (2016), un nuevo concepto que tiene como objetivo hacer que una aplicación web tenga la apariencia y comportamiento de una aplicación nativa usando solo JavaScript, HTML y CSS. Este tipo de aplicación cuenta con una arquitectura diferente a una aplicación tradicional, ya que en esta se traslada el manejo de la capa de presentación al lado del cliente, permitiendo así que el servidor se enfoque únicamente en distribuir información.

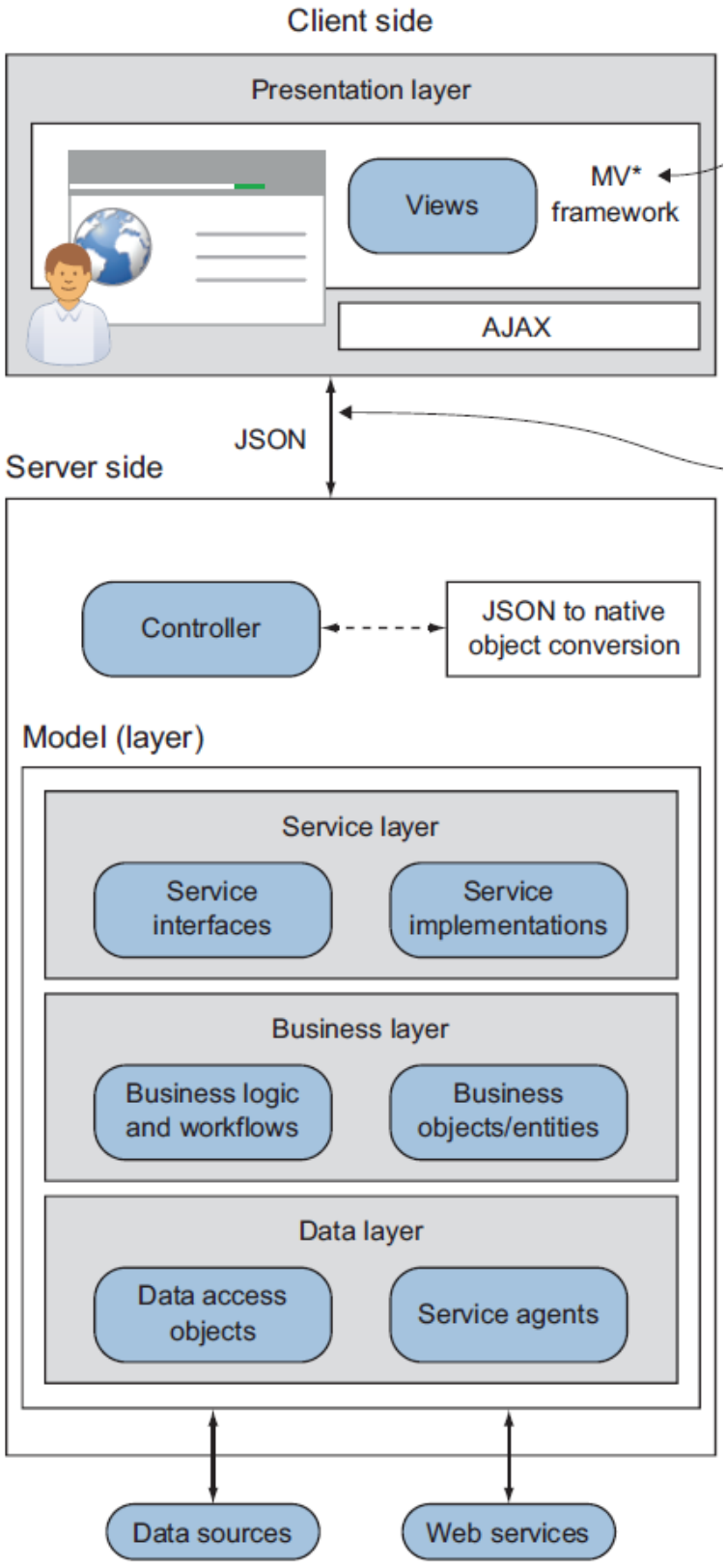


Ilustración 2 - SPA (EMMIT A. SCOTT, 2016)

La capa de presentación es manejada mediante código JavaScript y técnicas Ajax, por lo cual cuando un usuario navega entre los links internos de la aplicación no se requiere recargar la página, tan solo se cambia el contenido principal de la aplicación y se mantiene el código HTML inicial.

Las aplicaciones de una sola página aplican el concepto 'escudo' en la capa de presentación, el cual se refiere al contenido base inicial de código HTML que requiere una página para cargar, por ejemplo, la cabecera de la página es un escudo, ya que es una porción de HTML que estará presente en todas las páginas que posee la aplicación y ante el cambia de página a página tan solo el contenido principal es redibujado y la cabecera siempre se mantiene.

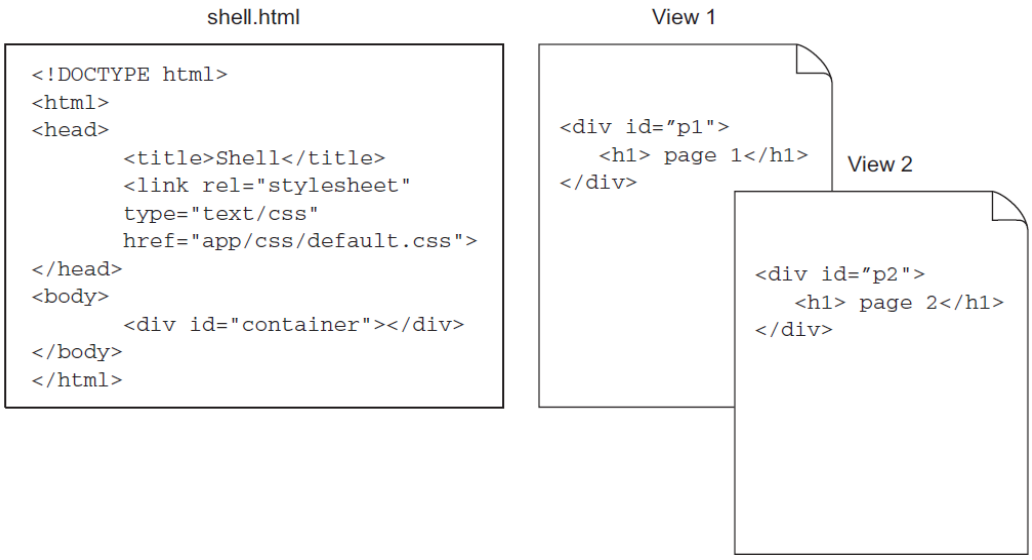


Ilustración 3 - Shell (EMMIT A. SCOTT, 2016)

Beneficios de una aplicación de una sola página sobre aplicaciones web tradicionales:

- **Se renderiza como una aplicación nativa de escritorio**

Una aplicación de una sola página tiene la capacidad de realizar las transiciones de página a página tan solo redibujando una sección de la pantalla, dando la sensación al usuario de una actualización inmediata del contenido.

- **Capa de presentación desacoplada**

El código que gobierna el como la interfaz gráfica aparece y se comporta es manejado del lado del cliente, lo cual logra desacoplar lo más posible al cliente y al servidor y por lo tanto que su mantenimiento y actualización pueda manejarse de manera separada.

- **Cargas útiles de transacciones más rápidas y ligeras**

Las transacciones con el servidor son muy ligeras, ya que luego de la primera entrega de recursos por parte del servidor el cliente solo solicita y envía información al servidor.

- **Menos tiempo de espera para el usuario**

El usuario tendrá un tiempo de espera menor en la carga inicial de la página ya que uso del shell permite dar una percepción de carga rápida. Asimismo, el tiempo de espera al navegar entre los links internos de la aplicación disminuye ya que para ello solo el contenido principal es redibujado y no se requiere refrescar toda la página.

- **Fácil mantenimiento del código**

Al separar la capa de presentación del servidor y moverlo al lado del cliente el mantenimiento y actualización del código también se separa, facilitando de esta manera que los programadores especializados en JavaScript, CSS y HTML no tengan la necesidad de manipular código del lado del servidor.

ANEXO 1 MATRIZ DE CONSISTENCIA

PROBLEMAS	OBJETIVOS	VARIABLE	DIMENSIÓN	INDICADORES
General	General	Rendimiento basado en el modelo RAIL (Lewis P. I., 2015)	Respuesta (Lewis P. I., 2015)	Tiempo de respuesta ante la acción "Ir a demo" (Lewis P. I., 2015)
¿Cuál es la diferencia de rendimiento basado en el modelo rail entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.?	Determinar la diferencia de rendimiento basado en el modelo rail entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.			Tiempo de respuesta ante la acción "Ir a probador" (Lewis P. I., 2015)
				Tiempo de respuesta ante la acción "Ir a Home desde Probador principal" (Lewis P. I., 2015)
				Tiempo de respuesta ante la acción "Ir a Probador principal desde probador mujer" (Lewis P. I., 2015)
Específico	Específico			Tiempo de respuesta ante la acción "Ir al probador detalle" (Lewis P. I., 2015)
				Tiempo de respuesta ante la acción "Seleccionar prenda" (Lewis P. I., 2015)
¿Cuál es la diferencia de Respuesta entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.?	Determinar cuál es la diferencia de Respuesta entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.		Animación (Lewis P. I., 2015)	Fotogramas perdidos en la página probador (Lewis P. I., 2015)
¿Cuál es la diferencia de Animación entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.?	Determinar cuál es la diferencia de Animación entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.			

¿Cuál es la diferencia de Inhabilitado time entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.?	Determinar cuál es la diferencia de Inhabilitado time entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A..			Fotogramas perdidos en la página demo (Lewis P. I., 2015)
			Inhabilitado (Lewis P. I., 2015)	Cantidad de tiempos Inhabilitados disponibles en Catalogo 3D (Lewis P. I., 2015)
				Cantidad de tiempos Inhabilitados disponibles en Demo (Lewis P. I., 2015)
				Cantidad de tiempos Inhabilitados disponibles en Descripción (Lewis P. I., 2015)
				Cantidad de tiempos Inhabilitados disponibles en Home (Lewis P. I., 2015)
				Cantidad de tiempos Inhabilitados disponibles en Probador/Maniquí (Lewis P. I., 2015)
¿Cuál es la diferencia de carga entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.?	Determinar cuál es la diferencia de carga entre las tecnologías Angular4 y ReactJS en la progressive web app de la empresa GLUP S.A.		Carga (Lewis P. I., 2015)	Tiempo de carga de la página Home (Lewis P. I., 2015)

ANEXO “B” HERRAMIENTAS APIS DE RENDIMIENTO

- **Performance Timing**

  Consortium (2016) lo definió como una interfaz que le otorga los mecanismos suficientes a Javascript para poder proveer una medición completa sobre los retardos involucrados en el cliente. La aplicación principal que tiene este API es el calcular los tiempos de carga de una página web desde que el usuario solicita el documento html al servidor, hasta que la página está totalmente cargada y lista para navegar.

  var page_load_time = now - performance.timing.navigationStart;

  Esta información permite conocer la experiencia que se ha llevado el usuario con respecto a la carga de la página web.

- **High Resolution Time Level 3**

  Consortium (2016) lo definió como una API que provee el tiempo actual en sub-milisegundos y no está sometida a los ajustes del reloj del sistema.

  performance.now()

- **Cooperative Scheduling of Background Tasks**

  Consortium (2016) lo definió como un API que detecta la existencia de un periodo de tiempo inhabilitado, y permite ejecutar tareas en función a su duración, si es la tarea sobrepasa este tiempo, esta será cancelada y ejecutada en el siguiente Inhabilitado time. El objetivo principal de este API es lograr ejecutar tareas secundarias sin afectar el rendimiento de a aplicación.

  Para ello el api requestIdleCallback recibe como parámetro una función, la cual será ejecutada cuando el API detecte un Inhabilitado time, y al mismo tiempo esta función recibirá un dato de entrada el cual es el tiempo límite para ejecutar la tarea.

ANEXO “C” METODOLOGÍA XP

Beck (2004) mencionó que la programación extrema es una metodología de desarrollo de software flexible, predecible y eficiente que permite manejar los cambios continuos de requerimientos.

La metodología XP posee 4 valores fundamentales, que deben estar presentes en todos los miembros del equipo.

- **Comunicación:**

  Al ser la comunicación uno de los principales factores por la cual tiene problemas un proyecto, XP maneja un conjunto de técnicas tales como las pruebas unitarias, par de programadores y estimaciones, las cuales permiten mantener una comunicación efectiva.

- **Simplicidad:**

  La simplicidad se refiere a la capacidad que se posee para poder eliminar el trabajo innecesario, y solo limitarlo a lo más simple posible para su correcto funcionamiento.

- **Retroalimentación:**

  La retroalimentación constante asegura que el desarrollo del software aborde los temas de estimaciones, tiempos y requerimientos lo más preciso posible, de esta manera si existiese algún cambio en un requerimiento, el programador puede entregar una retroalimentación directa al cliente, y este a su vez ajustar su tiempo en función a las nuevas estimaciones del programador.

- **Coraje:**

  El coraje es un valor muy importante dentro de XP, y engloba en parte a las tres anteriores, ya que este te permite realizar las cosas de manera correcta en situaciones complicadas.

  Asimismo, la metodología XP posee conjunto de prácticas que se recomienda debe ser usada en todo proyecto XP.

- **Planificando el juego:**

Determina rápidamente el alcance del siguiente entregable combinando las prioridades del negocio y las estimaciones técnicas.

- **Pequeños entregables:**

  Colocar el sistema base en producción lo más rápido posible, luego actualizarlo con las nuevas versiones obtenidas en cada iteración.

- **Metáfora:**

  Guía el desarrollo entero del sistema con una simple historia compartida sobre cómo funciona todo el sistema.

- **Diseño simple:**

  El sistema debe ser diseñado tan simple como es posible, cualquier descubierta complejidad debe ser removida lo antes posible.

- **Pruebas:**

  Los programadores crean pruebas unitarias para poder probar el funcionamiento del código. Los clientes crean pruebas para probar las características.

- **Refactorización:**

  Los programadores reestructuran el código sin cambiar el comportamiento, ello con la finalidad de remover duplicados, mejorar la comunicación, simplificar código, etc.

- **Par de programadores:**

  Todo el código de producción es escrito por dos programadores en una misma computadora.

- **Propiedad colectiva:**

  Cualquier miembro del equipo puede cambiar cualquier línea de código del sistema.

- **Integración continúa:**

Se integra y construye el sistema muchas veces en el día (cada vez que se finaliza una tarea).

- **40 horas a la semana:**

  Nunca trabajar más de 40 horas

  Cliente en el sitio: Incluir un usuario real en el equipo, esto permite que el equipo pueda realizar preguntas continuamente en cualquier momento.

- **Normas de codificación:**

  Los programadores escriben el código de acuerdo con reglas que enfaticen la comunicación.

CICLO DE VIDA DE LA METODOLOGÍA XP

a) Exploración

- En la etapa de exploración los programadores se familiarizan con las tecnologías a utilizar en la producción del sistema.
- Se usan una o dos semanas en construir las diferentes posibles arquitecturas del sistema.
- Se experimentan con los límites del rendimiento.
- Los programadores empiezan a realizar las primeras estimaciones de tiempo con respecto a las tareas presentadas por el cliente.
- Se educa al cliente en la creación de las historias, ello se logra con muchos pequeños feedbacks sobre las primeras historias, esto permite que el cliente pueda aprender la manera en la cual especificar y escribir la historia para que el programador pueda realizar una correcta estimación.

b) Planificación

- El principal objetivo de esta fase es que el cliente y el programador se pongan de acuerdo con respecto a las fechas (las más pequeñas) y el conjunto de historias más importantes.
- Combina las prioridades del negocio con las estimaciones técnicas.

c) Iteraciones para el primer entregable

- El cronograma de compromiso se divide en iteraciones de uno o cuatro semanas.
- Cada iteración produce un conjunto de casos de prueba funcional para cada una de las historias programadas.
- Las siguientes iteraciones son decididas por el cliente.
- Idealmente en el final de cada iteración el cliente tiene un completo test funcional.
- Al final de cada iteración se esa listo para poner el sistema en producción.

d) Producción

- Normalmente existen algunos procesos a ejecutar para ponerlo a producción.
- Esta fase es un buen momento para mejorar el rendimiento, ya que se cuenta con un conocimiento entero de la arquitectura del sistema.

e) Mantenimiento

- Se puede considerar como el estado natural de un proyecto en XP, ya que se producen nuevas funcionalidades, se mantiene el sistema corriendo y se incorporan nuevas personas al equipo.
- Se debe ser muy cuidadoso al momento de actualizar el sistema.
- Se cambia la estructura del equipo, ya que al trabajar con el sistema ya en producción se requiere mayor participación del service desk que de los programadores.

f) Entropía

- Se da cuando el cliente ya no puede escribir más historias.
- Se escribe la documentación entera del sistema, asegurando que sea lo suficiente específica para cuando se genere una nueva historia.
- Es el inicio de la muerte del sistema.

ROLES

a) Programador

- Son el corazón de XP.
- Su valor principal es la comunicación con las personas.
- Escriben test que permiten demostrar el funcionamiento de los aspectos importantes del software.
- Separa el programa de piezas pequeñas de código.
- Tiene el hábito de a simplicidad.

b) Cliente

- Son la otra mitad esencial en XP.
- Conocen que es lo que se va a programar.
- Debe indicar el orden de importancia de cada tarea.
- Saben escribir test funcionales.
- El mejor cliente es el que maneja el sistema o e que aprecia directamente la necesidad a satisfacer por el software a desarrollar.

c) Tester

- Rol enfocado al cliente.
- Ayuda al cliente a escoger y escribir los test funcionales.
- Ejecutan los test periódicamente.

d) Tracker

- Es la conciencia del equipo.
- Ejecuta las estimaciones con precisión basándose en los feedbacks.
- Mantiene el registro de la ejecución de las pruebas y defectos reportados.
- Su habilidad principal es la recolección de información.

e) Coach

- Responsable del proceso entero.

- Identificar y notifica cuando algún miembro del equipo se desvía del plan.
- Mayor conocimiento sobre XP, esto le permite dar soluciones eficientes al equipo.
- Enseña sobre la metodología XP.
- El rol del coach disminuye en equipos maduros y se distribuye.

f) Consultor

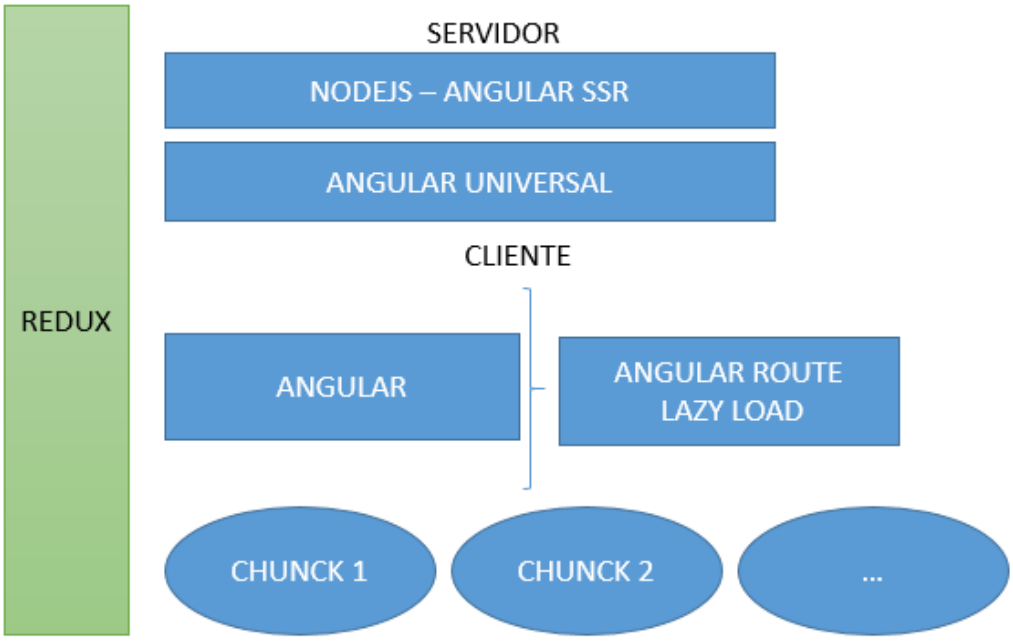
- Posee conocimientos técnicos sobre temas desconocidos por el equipo.
- Enseña al equipo como solucionar el problema.

g) Big Boss

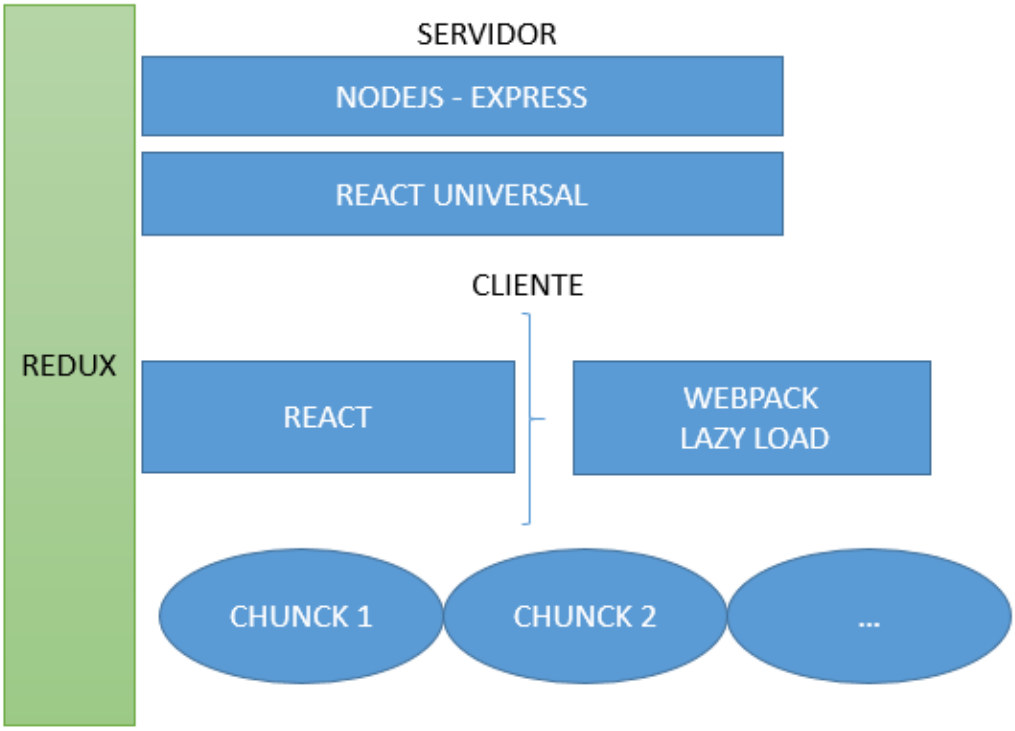
- Es el encargado del equipo.
- Gestiona las tareas y se encarga de que estas se ejecuten.
- Intermediario entre el cliente y el programador.

EXPLORACÓN

Esquema de la arquitectura de la aplicación – Angular 4



Esquema de la arquitectura de la aplicación - ReactJS



Durante la etapa de exploración se investigó sobre las tecnologías que participarían en el proyecto y su respectiva versión.

Angular 4	React JS
angular 4.4.2	react 15.5.4
ngrx/store 2.2.2	express 4.14.0
angular-universal 4.1.0-rc.1	redux 3.6.0
three 0.84.0	three 0.84.0
three-orbitcontrols-t 0.1.2	three-orbitcontrols-t ^0.1.2

PLANIFICACIÓN

Historia de usuarios / Tarjetas recopiladas de la herramienta trello.

HOME

Historía 1: Secciones del negocio **Descripción:** Como posible cliente de GLUP, debo poder visualizar a grandes rasgos todos los servicios que GLUP ofrece. **Condición de aceptación:** Mostrar multimedia relacionada a cada rubro del negocio. Mostrar una descripción breve de GLUP.

Historía 2: Sección de contacto **Descripción:** Como posible cliente de GLUP, debo poder visualizar la información de contacto de la empresa. **Condición de aceptación:** Mostrar información de la empresa / Número telefónico y dirección. Debe mostrarse como una sección oculta en el home.

Historía 3: Formulario de contacto **Descripción:** Como posible cliente de GLUP debo poder enviar un mensaje a GLUP para solicitar mayor información sobre el servicio ofrecido. **Condición de aceptación:** Validar los datos ingresados Registrar nombre, teléfono y motivos de contacto

Historía 4: Diseño del home **Descripción:** Como posible cliente de GLUP debo poder enviar un mensaje a GLUP para solicitar mayor información sobre el servicio ofrecido. **Condición de aceptación:** La paleta de colores debe ser negro y morado. Debe permitir apreciar cada multimedia relacionada a cada rubro del negocio.

AGENCIA DIGITAL

Historía 5: Publicaciones de las redes sociales **Descripción:** Como posible cliente debo poder visualizar las publicaciones que GLUP generará para mis redes sociales y como lograrán. **Condición de aceptación:** Se debe separar las publicaciones por tipo de redes sociales. Debe contar con una animación de entrada y salida de la página. Cada sección separada debe tener un texto de referencia al multimedia de la aplicación.

Historía 6: Diseño de agencia digital **Condición de aceptación:** Debe contener multimedia relacionada a las publicaciones de las redes sociales. Debe explicar de manera resumida la fabricación del recurso multimedia. Paleta de colores: Blanco y negro.

CATALOGO 3D

Historía 7: Diseño Catalogo 3D **Condición de aceptación:** Paleta de colores: Blanco y negro. Debe tener un multimedia relacionado al catálogo de productos que se ofrece. Paleta de colores: Blanco y negro.

Historía 8: Servicio de catálogo 3D **Descripción:** Como posible cliente debo poder visualizar un ejemplo del catalogo 3D que se ofrece y su respectiva descripción **Condición de aceptación:** Debe tener una animación al entrar y salir de la página Se debe poder acceder a un demo relacionado al 3D desde esta página

CATALOO 3D – 2

Historía 9: Servicio de creación de páginas web. **Descripción:** Como posible cliente de GLUP debo saber las tecnologías que se usarían si se contrato del servicio de creación de páginas web. **Condición de aceptación:** Mostrar descripción del servicio de creación de páginas web. Debe tener un botón par acceder a un demo relacionado al 3D. Debe contar con una animación de entrada y salida de la página.

Historía 10: Diseño de Catalogo 3D - 2 **Condición de aceptación:** Debe tener un multimedia relacionado al servicio de creación de páginas web Paleta de colores: Blanco y negro

CATALOGO 3D – DETALLE

Historía 11: Detalle del servicio de catalogo de productos **Descripción:** Como posible cliente debo poder saber cuales son las tareas que ejecutará GLUP para generar el catálogo de produtos. **Condición de aceptación:** La visualización entre paso y paso debe tener una animación. Debe contar con una animación de entrada y salida de la página.

Historía 12: Diseño de Catalogo 3D - Detalle **Condición de aceptación:** Debe tener multimedias relacionados a los pasos a seguir para la creación del catalogo 3D Paleta de colores: Blanco y negro

PROBADOR

Historía 13: Servicio de probador 3D **Descripción:** Como posible cliente debo poder entender el servicio del probador 3D. **Condición de aceptación:** Debe contar con una animación de entrada y salida de la página. Al seleccionar a uno de las imágenes (Maniquí Homre o Mujer), debe enviarte a la página probador detalle mostrandote el maniquí correspondiente en 3D.

Historía 14: Diseño de probador 3D **Condición de aceptación:** Mostrar una descripción sobre el probador. Mostrar opciones de maniquí 3D (Hombre - Mujer). Paleta de colores: Blanco y negro.

PROBADOR DETALLLE

Historía 15: Servicio de Probador 3D - Demo **Descripción:** Como posible cliente del servicio de Probador 3D, debo poder interactuar con el maniquí 3D y las prenda en 3D. **Condición de aceptación:** Se debe poder vestir al maniquí. Mostrar y ocultar lista de prendas con una animación. Mostrar y manipular al maniquí 3D. Las prendas seleccionadas (usadas para vestir al maniquí), deben distinguirse de las no seleccionadas.

Historía 16: Diseño de probador demo 3D **Condición de aceptación:** Paleta de colores: Blanco y negro Se debe dar la mayor parte del espacio al maniquí. Paleta de colores: Blanco y negro.Las prendas deben mostrarse ordenadas y ser seleccionables para vestir al maniquí

DEMO

Historía 17: Demo para el catálogo 3D **Descripción:** Como posible cliente del servicio de Cataogo 3D, debo poder interactuar con las prenda en 3D y su respectivo detalle. **Condición de aceptación:** Mostrar lista de prendas. Mostrar detalle de prenda seleccionada y su respectivo 3D.

Historía 18: Diseño del demo **Condición de aceptación:** Paleta de colores: Blanco y negro Deben mostrar las prendas de manera ordenada y fácil de apreciar. Del detalle de las prendas debe permitir visualizar las descripciones, composición y recomendaciones de las prendas así como la posibilidad de cambiar de imagen preview o verlo en 3D Las prendas deben mostrarse con una imagen y su respectivo nombre.

Progressive web app

Historía 19: Convertir aplicación en una progressive web app **Descripción:** Como posible usuario deseo que la aplicación web funcione offline, pueda agregarse en la pantalla de inicio de mi celular e iniciarse como una aplicación independiente del navegador, debe poseer un funcionamiento fluido y estable. **Condición de aceptación:** Terminado (Pendiente de SSL) - Funcionar offline Agregar a Home Splash Screen Carga de página en menos de 1.5s Animaciones fluidas < 16ms por frame Optimización de los recursos (webp y minificadores) Optimización de código (Rendimiento)

Iteraciones

Cada iteración tendrá una duración de una semana, para lo cual se estimaron las siguientes velocidades de programación.

	ITERACIONES			
HISTORIAS	Iteración 1	Iteración 2	Iteración 3	Iteración 4
1	x			
2	x			
3	x			
4	x			
5			x	
6			x	
7	x			
8	x			
9	x			
10	x			
11		x		
12		x		
13		X		
14		X		
15			X	

16			X	
17				x
18				x
19				x

Cronograma de iteraciones

Angular 4

ITERACIONES	F INICIO	F FIN
1	3/04/2017	10/04/2017
2	10/04/2017	17/04/2017
3	17/04/2017	24/04/2017
4	24/04/2017	1/05/2017

ReactJS

ITERACIONES	F INICIO	F FIN
1	1/05/2017	8/05/2017
2	8/05/2017	15/04/2017
3	15/04/2017	22/04/2017
4	22/04/2017	29/05/2017

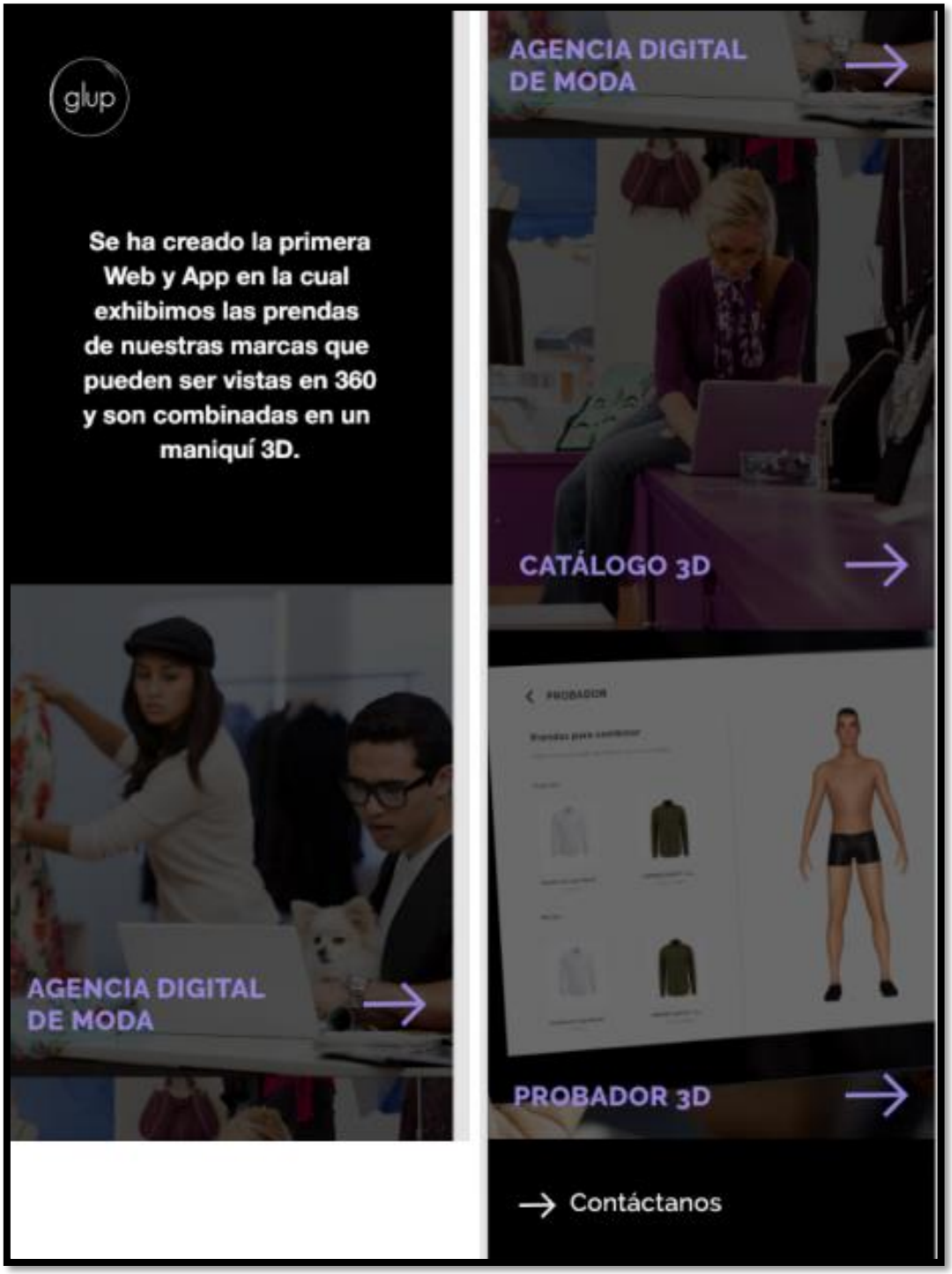
ITERACIONES

1. Iteración 1

H1 Secciones del negocio

Objetivos:

- Mostrar multimedia relacionada a cada rubro del negocio.
- Mostrar una descripción breve de GLUP.



H2 Sección de contacto

Objetivos:

- Mostrar información de la empresa / Número telefónico y dirección
- Debe mostrarse como una sección oculta en el home

H3 Formulario de contacto

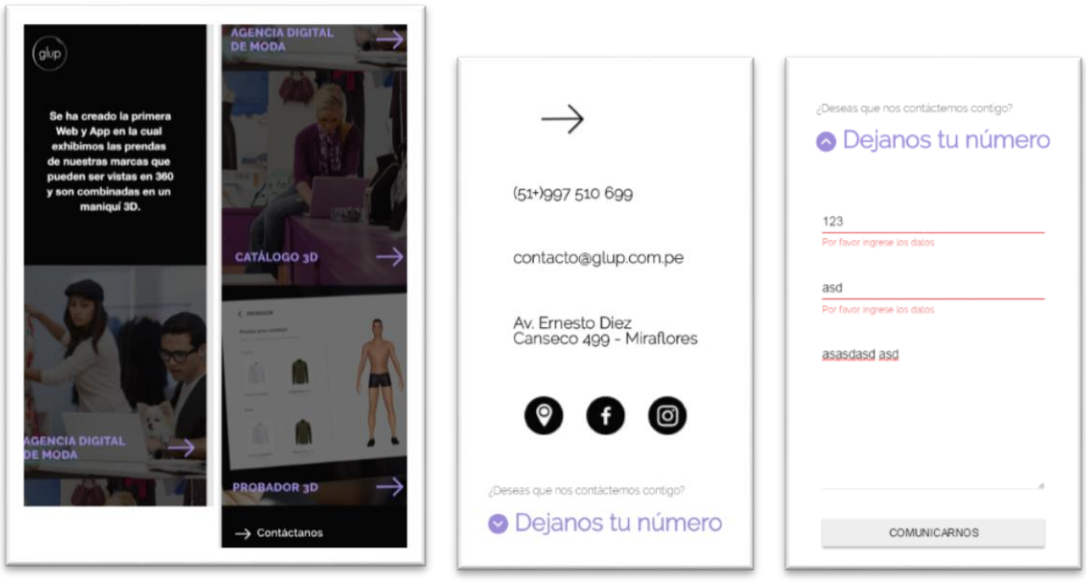
Objetivos:

- Validar los datos ingresados.
- Registrar nombre, teléfono y motivos de contacto.

H4 Diseño de home

Objetivos:

- La paleta de colores debe ser negro y morado.
- Debe permitir apreciar cada multimedia relacionada a cada rubro del negocio.



H7 Servicio de catálogo 3D - H8 Diseño catálogo 3D

Objetivos H7

- Debe tener una animación al entrar y salir de la página
- Se debe poder acceder a un demo relacionado al 3D desde esta página
- Debe contar con una animación de entrada y salida de la página

Objetivos H8

- Paleta de colores: Blanco y negro
- Debe tener un multimedia relacionado al catálogo de productos que se ofrece

H9 Servicio de creación de páginas web - H10 Diseño catálogo 3D-2

Objetivos H9

- Mostrar descripción del servicio de creación de páginas web
- Debe tener un botón para acceder a un demo relacionado al 3D
- Debe contar con una animación de entrada y salida de la página

Objetivos H10

- Debe tener un multimedia relacionado al servicio de creación de páginas web
- Paleta de colores: Blanco y negro

2. Iteración 2

H11 Detalle del servicio de catálogo de productos – H12 Diseño de Catalogo 3D-Detalle

Objetivos H11

- La visualización entre paso y paso debe tener una animación
- Debe contar con una animación de entrada y salida de la página

Objetivo H12

- Paleta de colores: Blanco y negro
- Debe tener multimedia relacionados a los pasos a seguir para la creación del catálogo 3D

H13 Servicio de probador 3D – H14 Diseño de probador 3D

Objetivos H13

- Debe contar con una animación de entrada y salida de la página
- Al seleccionar a uno de las imágenes (Maniquí Homre o Mujer), debe enviarte a la página probador detalle, mostrándote el maniquí correspondiente en 3D.

Objetivos H14

- Mostrar una descripción sobre el probador
- Mostrar opciones de maniquí 3D (Hombre - Mujer)
- Paleta de colores: Blanco y negro

3. Iteración 3

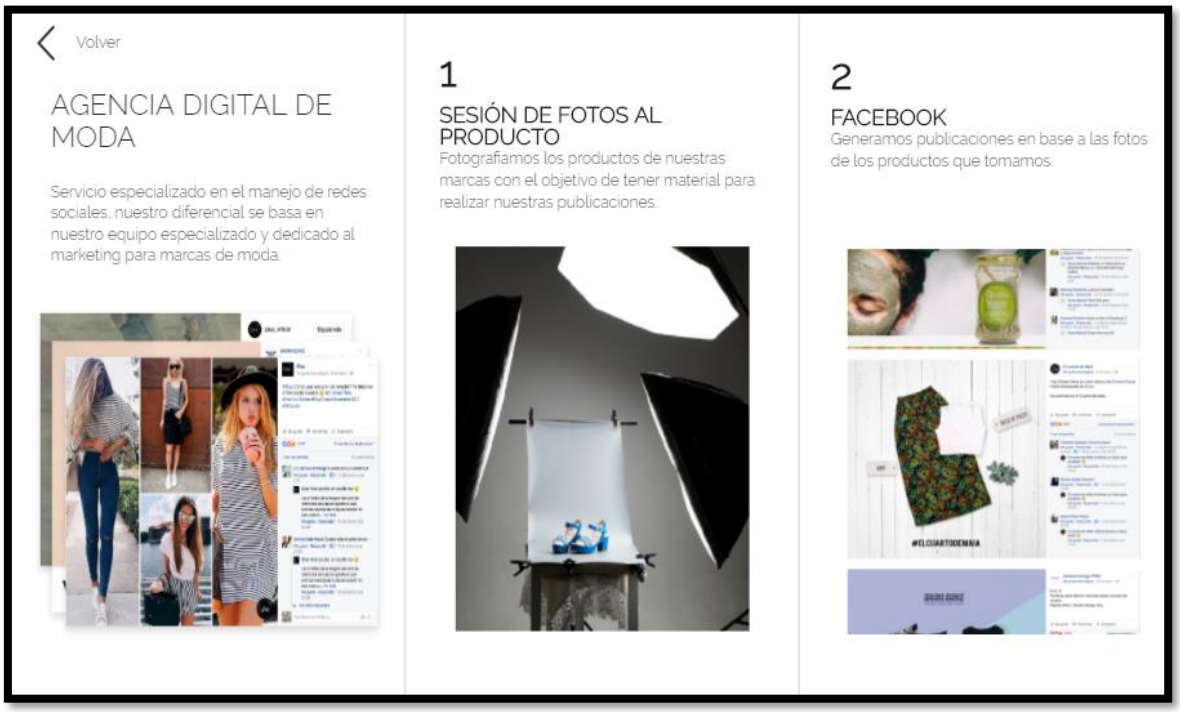
H5 Publicaciones de las redes sociales– H6 Diseño de agencia digital

Objetivos H5

- Se debe separar las publicaciones por tipo de redes sociales.
- Cada sección separada debe tener un texto de referencia al multimedia de la aplicación.
- Debe contar con una animación de entrada y salida de la página.

Objetivos H6

- Debe contener multimedia relacionada a las publicaciones de las redes sociales
- Debe explicar de manera resumida la fabricación del recurso multimedia
- Paleta de colores: Blanco y negro



H15 Servicio de Probador 3D-Demo – H16 Diseño de probador 3D

Objetivos H15

- Se debe poder vestir al maniquí.
- Mostrar y ocultar lista de prendas con una animación
- Mostrar y manipular al maniquí 3D
- Las prendas seleccionadas (usadas para vestir al maniquí), deben distinguirse de las no seleccionadas.

Objetivos H16

- Paleta de colores: Blanco y negro

- Se debe dar la mayor parte del espacio al maniquí
- Las prendas deben mostrarse ordenadas y ser seleccionables para vestir al maniquí

4. Iteración 4

H17 Demo para el catálogo 3D– H18 Diseño del demo

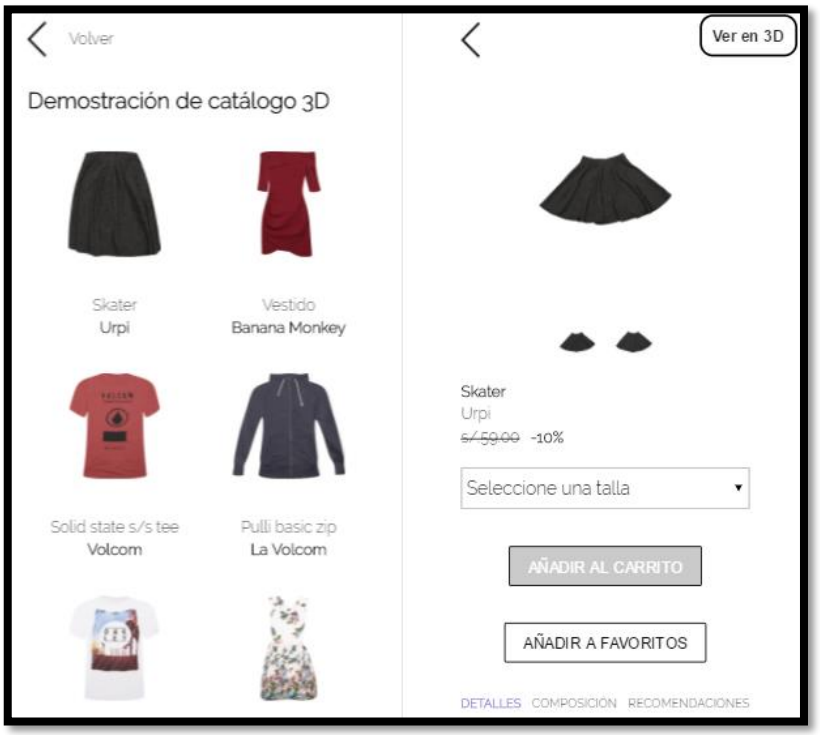
Objetivos H17

- Mostrar lista de prendas
- Mostrar detalle de prenda seleccionada y su respectivo 3D

Objetivos H18

- Paleta de colores: Blanco y negro
- Deben mostrar las prendas de manera ordenada y fácil de apreciar.
- Las prendas deben mostrarse con una imagen y su respectivo nombre.

- Del detalle de las prendas debe permitir visualizar las descripciones, composición y recomendaciones de las prendas así como la posibilidad de cambiar de imagen preview o verlo en 3D.



H19 Convertir aplicación en una progressive web app

Objetivos H19

- Terminado (Pendiente de SSL) - Funcionar offline
- Agregar a Home
- Splash Screen

```
{
  "short_name": "Glup Landing",
  "name": "Glup Landing",
  "background_color": "#000000",
  "display": "standalone",
  "theme_color": "#000000",
  "icons": [
    {
      "src": "launcher-icon-1x.png",
      "type": "image/png",
      "sizes": "48x48"
    },
    {
      "src": "launcher-icon-2x.png",
      "type": "image/png",
      "sizes": "96x96"
    },
    {
      "src": "launcher-icon-4x.png",
      "type": "image/png",
      "sizes": "192x192"
    }
  ],
  "start_url": "/?utm_source=homescreen"
}
```


- Carga de página en menos de 1.5s

  Los componentes son renderizados en el servidor, lo cual permite una carga casi instantánea.

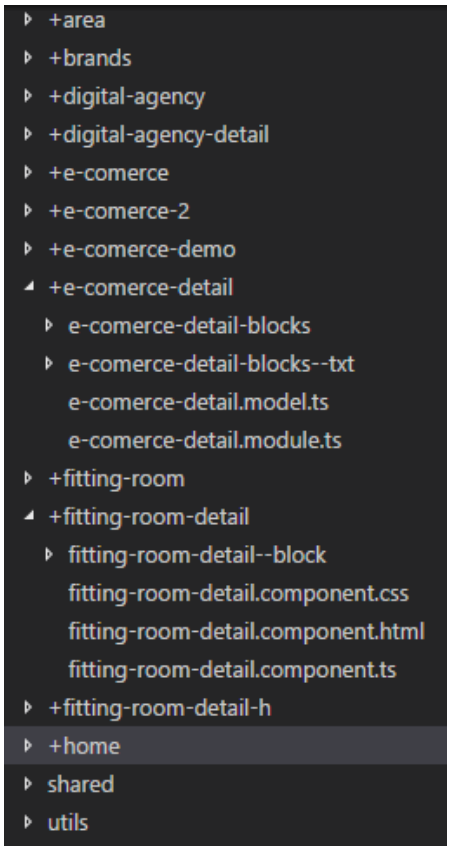
```
<!DOCTYPE html><html lang="en"><head><title data-react-helmet="true">Glup</title><meta data-react-helmet="true" name="application-name" content="Glup"/><meta data-react-helmet="true" name="description" content="3D Glup"/><meta data-react-helmet="true" charset="utf-8"/><meta data-react-helmet="true" name="viewport" content="width=device-width, initial-scale=1"/><meta data-react-helmet="true" name="msapplication-TileColor" content="#2b2b2b"/><meta data-react-helmet="true" name="msapplication-TileImage" content="/favicons/mstile-144x144.png"/><meta data-react-helmet="true" name="theme-color" content="#2b2b2b"/><meta data-react-helmet="true" name="msapplication-TileColor" content="#2b2b2b"/><link data-react-helmet="true" rel="stylesheet" href="/assets/vendor/material.min.css"/><link data-react-helmet="true" rel="shortcut icon" href="/favicon.ico"/><link data-react-helmet="true" rel="manifest" href="/assets/manifest.json"/><link href="/client/index-5ca054399a94bb3a492699381d48dd8d.css" media="screen, projection" rel="stylesheet" type="text/css"/></head><body><div id="app"><div style="overflow:initial;" data-reactroot="" data-reactid="1" data-react-checksum="-877786970"><!-- react-empty: 2 --><div class="home-animation-host " data-reactid="3"><div style="background-color:white;" class="home-before " data-reactid="4"><div class="about-component-1" data-reactid="5"><div class="about-component--action" data-reactid="6"><img src="./assets/img/icon_arrow_right.png" style="position:absolute;left:27%;top:44%;cursor:pointer;" data-reactid="7"/></div><div class="about-component--detail" data-reactid="8"><div class="about-component--telf" data-reactid="9">(51+)997 510 699</div><div class="about-component--email" data-reactid="10">contacto@glup.com.pe</div><div class="about-component--direction" data-reactid="11"><!-- react-text: 12 -->Av. Ernesto Diez<!-- /react-text --><br data-reactid="13"/><!-- react-text: 14 --> Canseco 499 - Miraflores<!-- /react-text --></div><div class="about-component--icons" data-reactid="15"><img src="./assets/img/Icons/pl.png" data-reactid="16"/><img src="./assets/img/Icons/fb.png" data-reactid="17"/><img src="./assets/img/Icons/ins.png" data-reactid="18"/></div></div><div class="about-component--footer" data-reactid="19"><div class="about-component--footer-1" data-reactid="20">¿Deseas que nos contáctemos contigo?</div><div class="about-component--footer-2" data-reactid="21"><img src="./assets/img/under.svg" style="height:26px;" data-reactid="22"/><!-- react-text: 23 --> <!-- /react-text --><!-- react-text: 24 -->Déjanos tu número<!-- /react-text --></div></div></div><div class="about-component-2" data-reactid="25"><div class="about-form-finish " data-reactid="26"><div data-reactid="27">Gracias por escribirnos, estaremos contactándonos pronto.</div></div></div><div class="about-component-2-form" data-reactid="28"><div style="width:100%;" class="mdl-textfield mdl-js-textfield" data-reactid="29"><input type="text" class="mdl-textfield__input" id="name" pattern="[A-Z,a-z, ]*" data-reactid="30"/><label class="mdl-textfield__label" for="name" data-reactid="31">Nombre</label><span class="mdl-textfield__error " data-reactid="32">Por favor ingrese los datos</span></div><div style="width:100%;" class="mdl-textfield mdl-js-textfield" data-reactid="33"><input type="text" class="mdl-textfield__input" pattern="[0-9]*" id="number" data-reactid="34"/><label class="mdl-textfield__label" for="number" data-reactid="35">Número</label><span class="mdl-textfield__error " data-reactid="36">Por favor ingrese los datos</span></div><div style="width:100%;" class="mdl-textfield mdl-js-textfield" data-reactid="37"><textarea class="mdl-textfield__input" type="text" rows="10" id="reason" data-reactid="38"></textarea><label class="mdl-textfield__label" for="reason" data-reactid="39">Schools attended</label><span class="mdl-textfield__error " data-reactid="40">Por favor ingrese los datos</span></div><button class="mdl-button mdl-js-button mdl-button--raised mdl-js-ripple-effect" data-reactid="41">Comunicarnos</button></div></div></div><div class="home-container" style="height:100%;" data-reactid="42"><div class="home-header" data-reactid="43"><div class="home-header-logo" data-reactid="44"><img src="./assets/img/logo.png" data-reactid="45"/></div><div class="home-header-description" data-reactid="46">Se ha creado la primera Web y App en la cual exhibimos las prendas de nuestras marcas que pueden ser vistas en 360 y son combinadas en un maniquí 3D.</div></div><div class="home-sections" data-reactid="47"><div class="home-block" data-reactid="48"><div class="img-container" data-reactid="49"><img alt="" class="img-container__img hight" src="./assets/img/Agencia.png" data-reactid="50"/></div><div class="home-block-description" data-reactid="51"><div class="home-block-description__text" data-reactid="52">AGENCIA DIGITIAL DE MODA</div><div class="home-block-description__arrow" data-reactid="53"><img class="arrow" src="./assets/img/icon_arrow_right_aux.png" data-reactid="54"/></div></div></div><div class="home-block" data-reactid="55"><div class="img-container" data-reactid="56"><img alt="" class="img-container__img hight" src="./assets/img/catalogo.png" data-reactid="57"/></div><div class="home-block-description" data-reactid="58"><div class="home-block-description__text" data-reactid="59">CATÁLOGO 3D</div><div class="home-block-description__arrow" data-reactid="60"><img class="arrow" src="./assets/img/icon_arrow_right_aux.png" data-reactid="61"/></div></div></div><div class="home-block" data-reactid="62"><div class="img-container" data-reactid="63"><img alt="" class="img-container__img hight" src="./assets/img/probador.png" data-reactid="64"/></div><div class="home-block-description" data-reactid="65"><div class="home-block-description__text" data-reactid="66">PROBADOR 3D</div><div class="home-block-description__arrow" data-reactid="67"><img class="arrow" src="./assets/img/icon_arrow_right_aux.png" data-reactid="68"/></div></div></div></div><div class="contact-us" data-reactid="69"><div class="contact-us__img" data-reactid="70"><img class="contact-us__arrow" src="./assets/img/icon_arrow_right_white.png" data-reactid="71"/></div><div class="contact-us__text" data-reactid="72">Contáctanos</div></div></div><div style="background-color:;" class="home-next" data-reactid="73"></div></div></div></div><script type="text/javascript" nonce="5316785b-aadf-401d-8bbb-ae47b708161a">window.__CLIENT_CONFIG__={"welcomeMessage":"Hello world!","serviceWorker":{},"polyfillIO":{"enabled":true,"url":"\u002F\u002Fcdn.polyfill.io\u002Fv2\u002Fpolyfill.min.js","features":["default","es6"]},"htmlPage":{"titleTemplate":"Glup","defaultTitle":"Glup","description":"3D Glup"}}</script><script nonce="5316785b-aadf-401d-8bbb-ae47b708161a" type="text/javascript">window.__ASYNC_COMPONENTS_REHYDRATE_STATE__={"resolved":{"1":true}};</script><script type="text/javascript" src="//cdn.polyfill.io/v2/polyfill.min.js?features=default,es6"></script><script type="text/javascript" src="/client/index-b0dcb802ece5d157d5bf.js"></script><script data-react-helmet="true" type="text/javascript">
```


- Animaciones fluidas < 16ms por Frame
- Optimización de los recursos (minificadores)

- Optimización de código (Rendimiento)

  El proyecto es modulado por concepto de componentes, esto permite que ante la actualización de un componente del árbol DOM, la actualización pueda encapsularse.

  Loops ejecutados fuera de las zonas de actualización de angular.



```
this._ngZone.runOutsideAngular(() => {
  let render = (time_0)=>{
    let time_1 = this.window.nativeWindow.performance.now();
    if( (time_0!=0) && (time_1-time_0)>16 ){
      if(this.isChrome){
        this.window.nativeWindow.worker_rum.postMessage(
          {
            "duracion":time_1-time_0,"event_name":"Animacion 3D"
          });

      }
    }
    this.controls.update()
    this.renderer.render(this.scene,this.camera)
    if(!this.cancel_requestanimation){this.requestAnimationFrame(()=>render(time_1))}
  }
  render(0)
})
```

PRODUCCIÓN

Cada iteración tiene como resultado un conjunto de entregables, los cuales son probados al validar las condiciones de aceptación de cada historia.

Historias	Resultado de las pruebas funcionales
1	100%
2	100%
3	100%
4	100%
5	100%
6	100%

7	100%
8	100%
9	100%
10	100%
11	100%
12	100%
13	100%
14	100%
15	100%
16	100%
17	100%
18	100% - Pendiente de implementación SSL

Compiladores:

- Webpack - React
- NgCompiler - Angular 4

Servidor:

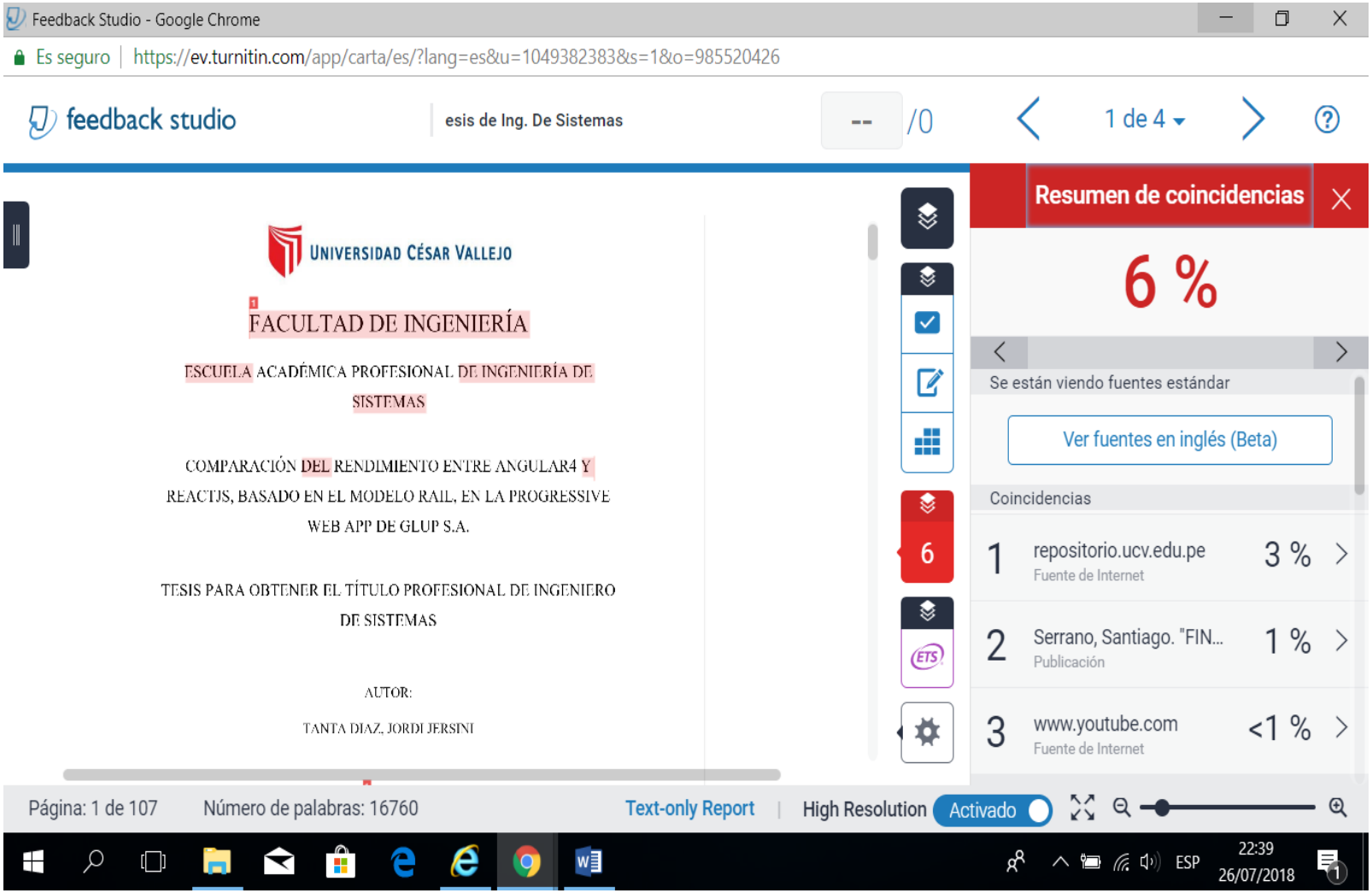
- Desplegado en un servidor NGINX
- Back End – NodeJS y Express

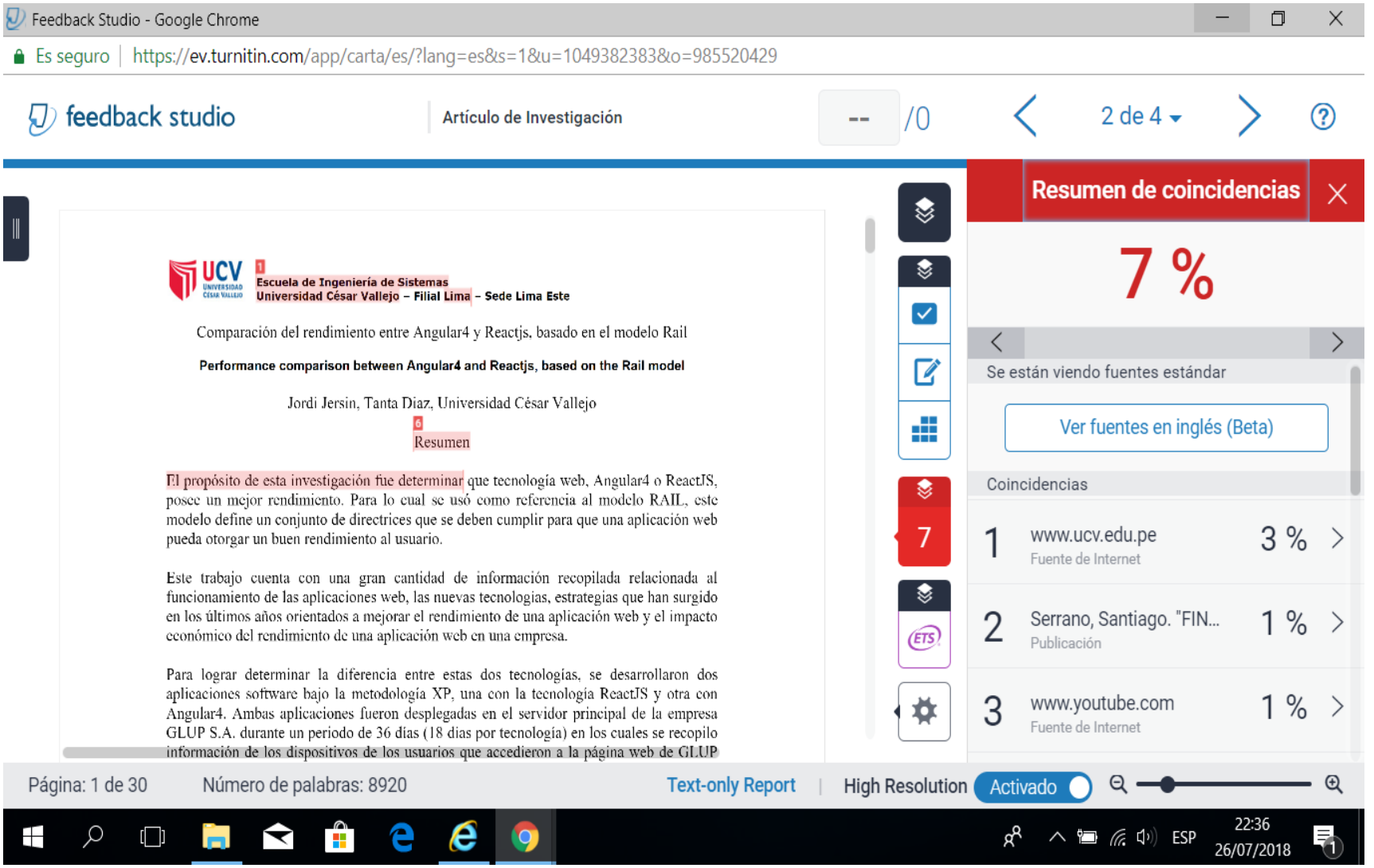
Dominio: http://www.m.glup.com.pe

MANTENIMIENTO

Aún no existen nuevas historias.

ANEXO “D” RESULTADO TURNITIN




Feedback Studio - Google Chrome
Es seguro | https://ev.turnitin.com/app/carta/es/?lang=es&s=1&u=1049382383&o=985520429
feedback studio
Artículo de Investigación
-- /0
2 de 4
Escuela de Ingeniería de Sistemas
Universidad César Vallejo – Filial Lima – Sede Lima Este
Comparación del rendimiento entre Angular4 y Reactjs, basado en el modelo Rail
Performance comparison between Angular4 and Reactjs, based on the Rail model
Jordi Jersin, Tanta Díaz, Universidad César Vallejo
Resumen
El propósito de esta investigación fue determinar que tecnología web, Angular4 o ReactJS, posee un mejor rendimiento. Para lo cual se usó como referencia al modelo RAIL, este modelo define un conjunto de directrices que se deben cumplir para que una aplicación web pueda otorgar un buen rendimiento al usuario.
Este trabajo cuenta con una gran cantidad de información recopilada relacionada al funcionamiento de las aplicaciones web, las nuevas tecnologías, estrategias que han surgido en los últimos años orientados a mejorar el rendimiento de una aplicación web y el impacto económico del rendimiento de una aplicación web en una empresa.
Para lograr determinar la diferencia entre estas dos tecnologías, se desarrollaron dos aplicaciones software bajo la metodología XP, una con la tecnología ReactJS y otra con Angular4. Ambas aplicaciones fueron desplegadas en el servidor principal de la empresa GLUP S.A. durante un periodo de 36 días (18 días por tecnología) en los cuales se recopilo información de los dispositivos de los usuarios que accedieron a la página web de GLUP
Resumen de coincidencias
7 %
Se están viendo fuentes estándar
Ver fuentes en inglés (Beta)
Coincidencias
1 www.ucv.edu.pe 3 %
Fuente de Internet
2 Serrano, Santiago. "FIN... 1 %
Publicación
3 www.youtube.com 1 %
Fuente de Internet
Página: 1 de 30
Número de palabras: 8920
Text-only Report
High Resolution
Activado
ESP
22:36
26/07/2018