

**Migración de un sistema de información para la administración de las relaciones con los clientes al
entorno Stack MERN**

Universidad Santiago de Cali

Facultad de Ingeniería

Programa Académico: Ingeniería de Sistemas

Título Profesional que se Otorga:

Ingeniero de Sistemas

Autor(es):

Daniel Alejandro Romero Gaitán.

Jhusef Alfonso López Parra.

Franciny Marín Romero.

Director del Trabajo:

Jair Enrique Sanclemente Castro

Localidad y Año:

Santiago de Cali, 2024

Resumen

Esta investigación tiene como objetivo realizar el diseño, construcción, despliegue y análisis de rendimiento de un sistema de información para la administración de las relaciones con los clientes, utilizando como base el Stack de desarrollo “MEAN” y actualizarlo al Stack de desarrollo “MERN”, debido a las altas tasas de procesamiento y de complejidad algorítmica que supone utilizar herramientas del primer Stack mencionado, como AngularJS, haciendo ver que existen alternativas igualmente llamativas, entrando en juego ReactJS. A su vez, utilizando como objetivo empresarial una cafetería para la construcción del software, se propone el uso de un prototipo web desarrollado mediante la metodología PSP (personal software process) y técnicas de análisis de rendimiento basado en pruebas del sistema en despliegue y local, lo cual genera como base un acercamiento más amplio y detallado sobre los cambios efectuados a la hora de intercambiar tecnologías en el desarrollo y optimizar la gestión de las relaciones con el cliente. De igual manera, permitirá la medición correspondiente y de control para cada requerimiento funcional del software. el prototipo permitirá medir el cambio que supone poner a contraposición dos tecnologías embarcadas en un mismo rubro tecnológico, pero con diferentes estructuras de codificación. Esto se traducirá en una diferenciación en cuanto a rendimiento computacional y de respuesta del software. El prototipo será desplegado en un entorno web para facilitar su conexión y permitirá la usabilidad de acuerdo con los roles brindados para los usuarios, optimizando espacio en cada uno de sus módulos. Los análisis generados por este sistema se enfocarán en brindar respuestas enfocadas al análisis de rendimiento y su diferenciación en cuanto al uso de las tecnologías antes mencionadas, con esto se busca obtener una visión completa del impacto y la efectividad de la elección en el desarrollo del sistema CRM, permitiendo así una visión más detallada y objetiva en cuanto a una mejora tecnológica y de satisfacción para los usuarios, teniendo como base un análisis enfocado en ventajas y desventajas de las herramientas iniciales y el sistema objetivo desarrollado en cuestión.

Palabras Clave: Gestión de relaciones con los clientes, AngularJS, ReactJS Personal software process, Entorno web, Eficiencia computacional, análisis de rendimiento, migración, despliegue.

Abstract

This research aims to design, construct, deploy, and analyze the performance of an information system for customer relationship management, using the "MEAN" development Stack as a basis and updating it to the "MERN" development Stack due to the high processing rates and algorithmic complexity that come with using tools from the first mentioned stack, such as AngularJS, thereby demonstrating that there are equally attractive alternatives, bringing ReactJS into play. Concurrently, with the business goal of using a coffee shop for the software construction, the use of a web prototype developed through the Personal Software Process (PSP) methodology and performance analysis techniques based on system testing in deployment and locally is proposed. This approach provides a broader and more detailed understanding of the changes made when exchanging technologies in development and optimizing customer relationship management. Similarly, it will allow for the appropriate measurement and control for each software functional requirement. The prototype will enable measurement of the change implied by contrasting two technologies within the same technological field but with different coding structures. This will result in a differentiation in terms of computational performance and software response. The prototype will be deployed in a web environment to facilitate its connection and will allow usability according to the roles provided for users, optimizing space in each of its modules. The analyses generated by this system will focus on providing answers aimed at performance analysis and its differentiation regarding the use of the aforementioned technologies. With this, the aim is to obtain a complete view of the impact and effectiveness of the choice in the development of the CRM system, thus allowing for a more detailed and objective view in terms of technological improvement and user satisfaction, based on an analysis focused on the advantages and disadvantages of the initial tools and the developed target system in question.

Keywords: Customer relationship management, AngularJS, ReactJS, Personal software process, Web environment, Computational efficiency, performance analysis, migration, deployment.

1. INTRODUCCIÓN

Un sistema de gestión de las relaciones con los clientes, es una herramienta globalmente utilizada, por millones de personas y empresas, ya que facilita administrar la información de los clientes, así mismo analizarla y utilizarla con un fin más práctico y específico dependiendo las reglas del negocio en cuestión, generando un valor agregado que le proporcione a las organizaciones una posición mucho mejor y efectiva en el mercado y así poder asegurar el máximo beneficio económico. (Bernal, 2021)

Debido a la alta demanda en cuanto al desarrollo de software, creciente y latente desde hace décadas, se ha podido evidenciar una gran necesidad en la innovación y evolución de nuevas tecnologías de desarrollo, combinado a su vez en conjunto con el objetivo de buscar una mayor simpleza y robustez en cuanto a procesamiento de información se refiere, es por ello, esencial, que en el desarrollo de software, se planteen y selecciones cuidadosamente herramientas que permitan afrontar de manera eficaz y efectiva las necesidades de construcción, dando como resultado un software completo y funcional, que no solo responda a las necesidades originales, sino, que sea escalable con el tiempo y de mantenimiento simple.

El proyecto, forma parte esencial del entendimiento en el arduo proceso que se denomina como la selección efectiva de las tecnologías de desarrollo, brindando una visión objetiva y sustancial sobre las diferenciaciones presentadas entre las diversas opciones cuando de desarrollo de software se habla, no solo a nivel lógico o de complejidad de negocio, sino a nivel computacional y de beneficencia en cuanto al desarrollador o desarrolladores se habla, sirviendo como el paso inicial y primordial para un resultado favorable en un proyecto.

AngularJS, es un framework de desarrollo todo en uno, que proporciona una grande cantidad de herramientas y funcionalidades para el desarrollo web, potenciado por la arquitectura MVC (modelo vista controlador), la cual servía como una guía efectiva en la aplicabilidad de conceptos claves para los desarrolladores y su facilidad para la creación de nuevo software, sin embargo, diferentes estudios han arrojado, que a pesar de que los desarrolladores puedan comprender y conocer conceptos de desarrollo efectivo gracias a AngularJS, uno de los mayores retos es su optimización y gestión del rendimiento. (Thilanka Kaushalya, 2021)

Seleccionar efectivamente las tecnologías de desarrollo proporciona una fácil percepción y procesamiento de la información, proporcionando una salida gratuita de información en una forma más familiar para los empleados de la empresa, como en formato de documento. La tarea común de la tarea es diseñar y desarrollar un proyecto que sea adecuado y cómodo para trabajar tanto con la pequeña cantidad de datos necesarios para un problema concreto como con una gran cantidad de datos. (S.A., 2021)

El proyecto proporcionado, analiza, desarrolla y evalúa las diferenciaciones en cuanto a la selección, ventajas y desventajas de tecnologías de desarrollo, así mismo, proporciona un panorama objetivo en cuánto al desarrollo de un mismo sistema en ambientes de desarrollo separados, rendimiento, optimización y mejores prácticas son solamente bases para tener un software de calidad, pero mayor a eso, tener el conocimiento y la perspectiva, en cómo nuevas tecnologías pueden influir en el rumbo futuro de un desarrollo más amigable, para el entorno virtual y sus desarrolladores.

1.1 Estado del Arte

1.1.1 Marco claro y eficiente para migrar Aplicación web de Angular a React.

El avance de los frameworks front-end hacia arquitecturas basadas en componentes ha sido notable en los últimos años, con React emergiendo como uno de los líderes indiscutibles en este espacio. Esta transición ha dejado obsoletos a muchos sistemas basados en AngularJS, tanto en términos de experiencia de usuario como eficiencia en el desarrollo. Sin embargo, migrar de una aplicación AngularJS a una arquitectura de componentes como React no es una tarea imposible, debido a las fundamentales diferencias tanto en conceptos como patrones y estructuras de la aplicación. En este sentido, esta investigación propone un marco claro y eficiente para llevar a cabo esta migración, con un enfoque paso a paso que aborda las complejidades inherentes. Además, se presenta una herramienta de asistencia llamada Ng-

React Copilot, diseñada para automatizar partes críticas de la migración y guiar a los desarrolladores en este proceso. Esta herramienta se desarrolló mediante la traducción de las refactorizaciones propuestas en el marco de migración en algoritmos específicos, lo que permite escanear y analizar la base de código existente. Tanto el marco de migración como la herramienta de asistencia se evaluaron exhaustivamente en una variedad de aplicaciones de AngularJS, desde pequeñas hasta empresas, demostrando la validez y eficacia de este enfoque de migración. (Kaushalya & Perera, 2021)

1.1.2 Análisis de rendimiento de aplicaciones web utilizando frameworks Angular, React y Vue.

A la hora de realizar desarrollo de software nos encontramos con muchas opciones, cada persona defiende el framework o herramientas las cuales les parezcan más cómodas o favorables, Elegir entre herramientas, herramientas como AngularJS, ReactJS y VueJS, muy conocidas en el campo de desarrollo front-end. (Konrad Bielak, Bartłomiej Borek, 2020)

Para llevar a cabo el análisis se usaron criterios estáticos los cuales son documentación favorable y generalmente disponible, desarrollo de aplicaciones, velocidad de desarrollo de aplicaciones, desarrollo y soporte por parte de los creadores, también criterios prácticos, los cuales son los tiempos medidos durante la ejecución, en las pruebas se realizaron con operaciones simples utilizando la función CRUD (Crear Leer Actualizar Eliminar), el análisis de rendimiento realizado de esta manera muestra las diferencias entre los marcos. Para el análisis comparativo se utilizaron las siguientes herramientas persas de desarrollador: Google Analytics, Google Chrome, Mozilla Firefox, Chrome DevTools.

1.1.3 Desarrollo de un prototipo CRM SaaS para PYMES del sector restaurantes.

En la actualidad JavaScript es uno de los lenguajes más usados en el mundo del desarrollo, no solo por su amplia gama de bibliotecas, herramientas o uso tanto en front-end como en backend, con tantas opciones en el mercado para framework surgen muchas preguntas y necesidades a la hora de elegir uno, curva d aprendizaje, consumo, rendimiento, etc. (Colares de Camargos et al., 2019)

Este estudio comparativo examina los framework JavaScript, ReactJS y AngularJS, el cual tiene como objetivo identificar cual es el que ofrece un valor superior o agregado en proyectos de desarrollo web, se concluye que, aunque similares, ReactJS emerge como la opción más completa y proporciona resultados superiores en los parámetros evaluados. (Konrad Bielak, Bartłomiej Borek, 2020)

1.1.4 Optimización del Proceso de Compra en la Renovación de la Aplicación de Tienda en Línea de Privalia.

En el sector de desarrollo para los e-commerce también se enfrentan desafíos como la correcta elección de framewroks o librerías basadas en el Modelo Vista Controlador (MVC), el uso de ReactJS con otras herramientas, dan una amplia gama de herramientas las cuales ofrecen todo lo necesario para llevar a cabo ya sea la creación de una página web entera o solo un módulo de esta.

En este artículo, encontramos que el objetivo de este trabajo es sustituir un antiguo formulario de pago por una aplicación que funcione independientemente, usando ReactJS y Redux la cual es una librería hace que la gestión de datos y los eventos que tienen que gestionar la aplicación sea más llevadero. (Pla & Fairén, 2019)

1.1.5 Una detallada encuesta sobre React.

Debido al crecimiento exponencial de consumidores y a su vez soluciones a estas, surge la necesidad de crear

aplicaciones las cuales sean escalables y rápidas para satisfacer no solo el mercado si no también el cliente, existen muchas soluciones y opciones disponibles para realizar a cabo un desarrollo web, pero se debe elegir el que sea mas adecuado y mas se acerque o cumpla con las expectativas de la herramienta que se necesita.

En esta estudio no solo se presentan la historia, ventajas y desventajas de ReactJS sino también un detallado análisis de la historia y las características que más destacan a ReactJS, no obstante, se presenta una discusión sobre React Native , que es una herramienta para realizar aplicaciones nativas , haciendo que este documento proporcione una idea general de la razón por la que ReactJS es una de las herramientas para desarrollo web más utilizadas.(Komperla et al., 2022)

1.2 Planteamiento del problema.

El arduo proceso de migrar de un Stack tecnológico a otro presenta desafíos significativos, especialmente cuando se busca optimizar y mejorar la experiencia del cliente. Particularmente hablando, la transición de una aplicación web basada en AngularJS a la arquitectura de componentes ReactJS, plantea interrogantes fundamentales que deben abordarse de manera integral. Por un lado, se enfrenta la necesidad de adaptar una estructura de aplicación AngularJS, con su enfoque más tradicional basado en MVC (Modelo-Vista-Controlador), a los principios de componentes reutilizables y unidireccionales de datos que caracterizan a ReactJS. Esta migración no se limita únicamente a la reescritura de código, sino que también implica una reconsideración profunda de la arquitectura y los patrones de diseño para aprovechar al máximo las ventajas que ofrece ReactJS en términos de rendimiento, mantenibilidad y escalabilidad.

Por otro lado, la migración entre Stacks tecnológicos involucra una serie de consideraciones prácticas que impactan directamente en la experiencia del cliente. Durante el proceso de migración, es muy importante minimizar cualquier interrupción en el servicio y se debe garantizar una transición fluida para los usuarios finales. Esto implica que se debe gestionar eficazmente la compatibilidad entre las dos tecnologías durante el período de transición, también mantener la coherencia en la interfaz de usuario para evitar confusión o frustración por parte de los usuarios. Además, se debe asegurar la preservación y mejora de las características y funcionalidades existentes, mientras se introducen nuevas capacidades y mejoras derivadas de la nueva tecnología.

Asimismo, el éxito de la migración depende en gran medida de la capacidad para abordar los desafíos específicos y complejidades inherentes a cada Stack tecnológico involucrado. En el caso de la transición de AngularJS a ReactJS, es crucial comprender las diferencias fundamentales en la forma en que estas tecnologías manejan el estado de la aplicación, el enrutamiento, la gestión de datos y otros aspectos clave del desarrollo web. Esta comprensión profunda es esencial para diseñar e implementar estrategias efectivas de migración que maximicen los beneficios de la nueva tecnología mientras se minimizan los riesgos y las complicaciones asociadas ¿Cómo afecta la elección de un Stack tecnológico frente a otro en un sistema, para optimizar y mejorar la experiencia del cliente?

1.3 Metas

El objetivo principal del proyecto es realizar la planeación, construcción, análisis y documentación de un sistema CRM basado en dos entornos de desarrollo en contraposición para determinar su diferenciación en cuanto a pruebas de rendimiento siguiendo la metodología PSP (personal software process), para ello se definieron las siguientes metas:

1. Planear y diseñar de manera óptima el comportamiento y los requerimientos del sistema CRM teniendo en cuenta la arquitectura escogida.
2. Codificar de manera completa las funcionalidades planteadas y a su vez realizar la compilación pertinente para la búsqueda de errores.
3. Realizar el análisis y pruebas de rendimiento, siguiendo la metodología escogida y realizar la documentación respectiva del proceso para su posterior evaluación.

4. Concluir de manera efectiva el proyecto y su documentación detallada de los resultados y procesos efectuados.

2. MATERIALES Y MÉTODOS/METODOLOGÍA

Materiales y Métodos:

2.1.1 Customer relationship management system (CRM):

Un sistema CRM, más conocido como “Customer relationship management system” es, como su nombre lo indica, un sistema de apoyo para la administración de las relaciones con los clientes, siendo más específicos, es un conjunto de herramientas y procesos, que comprende una gran cantidad de prácticas y metodologías, que se ayudan y se soportan en un entorno más grande que lo cobija, siendo una organización, la responsable de darle la estructura y los lineamientos según el objetivo principal, es así, la alineación de la estrategia empresarial, la estructura de la misma, y su cultura organizacional, llevada de la mano con la tecnología con el fin de que cada interacción con el o los clientes, sea satisfactoria y añada valor a quien haga uso de esta herramienta, aprovechando al máximo el rendimiento de la organización. (BAIYEWU, 2022)

2.1.2 AngularJS:

AngularJS, es conocido como un framework mantenido y diseñado por google, que está escrito en base a JavaScript, pero más concretamente realizado en typescript, el cual es un superset del anterior nombrado, proporciona controladores y directivas para dividir el código, basándose principalmente en arquitectura MVC (modelo vista controlador) sirviendo como guía para codificar de manera limpia y con buena taza de respuesta. (Vazquez, 2020)

2.1.3 ReactJS:

ReactJS, es una librería de código abierto, diseñada y mantenida por Facebook, a diferencia de AngularJS que es un framework (marco de trabajo), no ejerce un patrón de diseño como el MVC, por ejemplo, así mismo, depende del desarrollador poder utilizar y unir todos los componentes de ReactJS, puesto que es más liberal y permisivo. (Subramanian, 2019)

2.1.4 NodeJS:

NodeJS, es un entorno de ejecución del lenguaje JavaScript, pero, que se ejecuta fuera del navegador, los creadores utilizaron el motor V8 diseñado por Google para JavaScript y lo implementaron de tal forma que fuera posible ejecutarlo independientemente, naciendo así NodeJS, famoso por ser utilizado en diferentes plataformas como Netflix o LinkedIn. (Subramanian, 2019)

2.1.5 MongoDB:

MongoDB es una base de datos no SQL (structured query language), orientada a documentos, posee un esquema flexible y su forma de ser consultada es mediante JSON (javascript object notation), al ser una base de datos no relacional, permite ser de gran utilidad al utilizar gran volumen de datos. (Subramanian, 2019)

2.1.6 ExpressJS:

Similar a AngularJS, ExpressJS es un framework o marco de trabajo, pero, basado en NodeJS, que sirve para realizar la codificación y estructuración de la aplicación del lado del servidor (backend), conexiones a bases de datos, rutas, peticiones HTTP, la mayor diferencia es que, a diferencia el anteriormente nombrado, framework, este no maneja una arquitectura obligatoria. (Subramanian, 2019)

2.1.7 Pruebas de rendimiento:

Las pruebas de rendimiento en software son fundamentales para determinar la calidad del producto o mejorarlo, permiten identificar aspectos que se deben pulir, para poder alcanzar la satisfacción del o los clientes, pueden constar en el uso de diferentes herramientas o técnicas, que planteen las debilidades más notorias de un software en su etapa final o de producción. (Gil-Vera, 2018)

2.1.8 Páginas pruebas de rendimiento online:

2.1.8.1 PageSpeed Insights:

Página web diseñada por google, que proporciona una visión general de la velocidad, genera reportes rápidos y puntuaciones adaptadas al lenguaje natural, que se entienden con el uso de gráficas en sus respuestas, una herramienta fácil y versátil.

2.1.8.2 WebPageTest:

Similar a la anterior herramienta, es una página web que realiza pruebas de rendimiento, mediante el URL, tiene diversos modos de configuración, para el host de conexión, el navegador a usar, entre otras opciones, cuenta con un modo gratuito y uno de pago que ofrece mayor personalización.

2.1.9 Extensiones de Google:

2.1.9.1 Angular DevTools:

Una extensión que permite a los desarrolladores, realizar pruebas de rendimiento, observar los componentes de AngularJS, tiempos de ejecución, código, entre otras herramientas, para comprender la estructura de una app o página web.

2.1.9.2 React DevTools:

Extensión que permite realizar análisis al árbol de componentes, configuraciones y pruebas de rendimiento, sirve para hacer “debugging”, además de tener una interfaz entendible y clara para el usuario.

2.1.9.3 Lighthouse:

Extensión proporcionada por google, para ejecutar pruebas de rendimiento en páginas web, el objetivo es entender las fallas, buenas prácticas y problemáticas que se puedan presentar en el sitio en cuestión, fácil de usar y muy intuitiva, se posiciona como una opción completa y accesible a los desarrolladores.

2.1.10 Módulos:

2.1.10.1 Módulo Logueo o Registro:

El Módulo de logueo o registro, bien sea el caso, permite al usuario, o en su defecto el consumidor inicial del sistema, realizar su primer contacto con el sitio, registrando en un primer paso sus datos, o si ya se realizó, permitiéndole ingresar de manera directa, con persistencia en la información y evitando errores de dobles registros.

2.1.10.2 Módulo de pedido:

El módulo de pedidos permite al usuario, la creación de una nueva orden para el sistema, utilizando las bases de datos permitidas y la información previamente almacenada, para su consulta.

2.1.10.3 Módulo categorías:

Este módulo permite a los administradores la creación, eliminación, edición de categorías para los productos de la aplicación, estas opciones solo están habilitadas para el rol de administrador, debido a que los pedidos dependen de este módulo y es propenso a alterar el flujo de datos.

2.1.10.4 Módulo correos:

Este módulo permite a los administradores enviar correos personalizados a los usuarios, basándose en una

consulta y filtro, por fechas, pedidos, nombres, etc, también admite un formato HTML para crear un correo más visible y el uso de imágenes, a su vez, permite personalizar la fecha de envío de los antes descritos.

2.1.10.5 Módulo de configuración de perfil:

Este módulo permite configurar el perfil del usuario o administrador, su información de contacto, o errores ortográficos, módulo interactivo y básico de cualquier sistema de manejo de datos.

2.1.10.6 Módulo de historial:

Este módulo permite a los usuarios y administradores previsualizar su historial de pedidos, ver gráficas relacionadas y poder tener conciencia de la cantidad de pedidos y sus motivantes, si es un administrador, es capaz de visualizar más a fondo el historial de otros usuarios no administradores, haciendo una diferencia entre las posibilidades de un rol y otro.

2.1.10.7 Módulo de estadísticas:

Este módulo permite a los administradores, poder visualizar gráficas para entender la cantidad de pedidos, ingresos y egresos del sistema, estadísticas básicas en base al día anterior a la visualización y poner filtros a las gráficas para ver información específica

2.2 Explicación de la Metodología

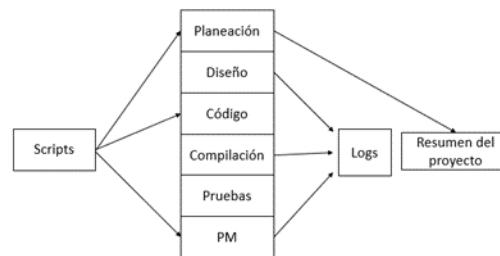


Figura 01: Metodología PSP (Elaboración propia)

La metodología PSP (Personal Software Process) surge de la necesidad de entregar software con un nivel de calidad alto, por lo tanto, se tiene que garantizar que cada uno de los procesos que conlleva a estos desarrollos surgen de un proceso maduro a nivel personal como es la planificación y que todo se ejecute de acuerdo con el plan establecido.

El Proceso de Software Personal (PSP) es un marco trabajos que tienen como objetivo enseñar los programadores hacen mejor su trabajo. Muestra cómo estimar y planificar el trabajo, monitorear el desempeño en comparación con estos planes y cómo mejorar la calidad del programa.

2.3 Etapas o Fases de la Metodología

Debemos tener en cuenta que en esta metodología los llamados “scripts”, son documentos guías o documentos que contienen la información detallada de sobre cómo realizar actividades durante cada etapa del desarrollo de software, están diseñados para ayudar a los desarrolladores a cumplir un proceso bien estructurado, lo que puede mejorar la calidad y la productividad del trabajo.

2.3.1 Planeación.

En nuestro proyecto teniendo en cuenta que contábamos inicialmente con un código open source de un CRM y la documentación de las herramientas de las pruebas que vamos a realizar en el proyecto, procedemos con definir los objetivos del proyecto de desarrollo:

1. Funcionalidad
2. Estándares de calidad

3. Asignación de recursos tecnológicos como herramientas de desarrollo de software.

2.3.2 Diseño.

En esta etapa se revisa y comprende cuales son los requisitos del sistema tales como funcionalidades y los requisitos del cliente y el usuario final, el diseño en base a la arquitectura, tal como la división de módulos y como se comunican entre ellos, con su asignación de responsabilidades.

2.3.3 Código.

En esta etapa construimos el código que se necesita para poder implementar las funcionalidades definidas en la etapa anterior(diseño), teniendo documentado el tiempo total gastado en la codificación.

2.3.4 Compilación.

Antes de iniciar con esta etapa de la metodología se deben preparar y configurar el entorno de desarrollo con las herramientas que se van a necesitar, como el compilador o editor de código, las bibliotecas, configurar las variables de entorno, etc.

Se registra cualquier defecto encontrado frente al código, también se lleva un script del tiempo total gastado en la compilación.

2.3.5 Pruebas.

En esta fase de la metodología se debe crear un plan en el cual se defina el alcance, estrategias, casos de pruebas que abarquen aspectos y funcionalidades, recursos y criterios para realizarles las pruebas al software, luego de conocer la gravedad, impacto en el código de los errores, se realizan pruebas hasta que no se presenten más errores y reparar los errores que han surgido

2.3.6 PM(PostMortem).

En esta fase también conocida como revisión posterior, se revisan todos los resultados del proyecto, tiempos de entrega, se verifica si se cumplieron los requisitos, el material o presupuesto utilizado, áreas de fortaleza o áreas de mejora, la calidad del código, se presenta toda la documentación resultante de todas las fases de la metodología y se proponen correcciones en caso de que se realicen proyectos futuros.

3. DESARROLLO

3.1 Introducción al Proceso de Desarrollo

Una vez se haya completado la fase de la planeación, que, en el caso del proyecto presentado, viene siendo el funcionamiento y requerimientos del sistema CRM y las herramientas a utilizar en sus fases posteriores, se procede a realizar el diseño del mismo en base a ambos Stacks de desarrollo presentados, la fase de codificación y compilación, representan el paseo intermedio y final en la construcción del software en ambos entornos, pasando a las pruebas, en esta etapa se hace énfasis en el estilo de pruebas, puesto que se verán testeados ambos sistemas, para determinar su mayor diferenciación y poder llegar a la fase del post mortem, que se verá evidenciada en las conclusiones del proyecto en cuestión y en su afección en el contexto planteado anteriormente.

3.1.1 Ejecución del Desarrollo

3.1.1.1 Etapa Planeación:

Durante la etapa de planeación, se empezó a definir las ideas de requerimientos funcionales y no funcionales del sistema CRM, se debía entender el funcionamiento de este, las herramientas de desarrollo escogidas (Stacks de desarrollo) y herramientas de apoyo para la gestión del tiempo, a su vez, se realizó la base teórica para la distribución de actividades en el grupo de desarrollo, y poder realizar las demás fases de la metodología.

3.1.1.2 Etapa Diseño:

Se utilizaron herramientas como Lucidshart o draw.io para la creación y diseño inicial de los requerimientos y

casos de uso del sistema objetivo, entendiendo más a fondo el funcionamiento deseado.

Se definieron el comportamiento lógico que debe tener el sistema, los participantes y sus actores en cada caso de uso, así mismo, una visión general de cada módulo y su funcionalidad, como las ventas, historiales, inicios de sesión, categorías, pedidos, correos, etc.

Adicional a ello, se definió el uso de patrones de diseño para administrar mejor el orden de codificación y llevar de la mano la metodología escogida (personal software process), siendo este patrón de arquitectura el modelo-vista-controlador, puesto que iba de la mano con frameworks como AngularJS, que está presente en uno de los 2 Stacks tecnológicos objetivo, NodeJS, siendo el entorno backend escogido para la lógica del lado del servidor, complementado con ExpressJS y MongoDB, sirviendo como framework de backend y lógica de conexión para los llamados “end-points” o rutas que tendría el sistema, y el segundo como la base de datos NoSQL elegida, para un mejor manejo de datos, y mayor volumen de los mismos.

3.1.1.3 Etapa Código:

En esta etapa se encuentra el corazón del desarrollo del sistema en ambos entornos, aquí se procede a la codificación neta, sirviendo de ayuda herramientas como Visual Studio Code (VScode), y soporte en inteligencias artificiales como GitHub Copilot y Bard.

En este proceso se utilizó la documentación previamente creada, para realizar la codificación perfectamente alineada a los requerimientos, siendo esta la etapa más larga del proceso y la que más énfasis debería tener para el correcto funcionamiento del CRM, implementando cada funcionalidad y anidando cada necesidad planteada.

Contraponiendo ambos Stacks tecnológicos se evidenció las dificultades de sus herramientas diferenciadoras, siendo AngularJS un reto más grande que ReactJS en cuanto a comprensión de funcionalidades y ejecución de las mismas.

3.1.1.4 Etapa Compilación:

La etapa de compilación se dividió en dos sub-partes para la satisfacción y optimización del tiempo.

En la primera parte se realizó la compilación del código generado en ambos stacks, se detectaron errores de codificación y fallas lógicas, se realizó una pequeña documentación temporal para la corrección de los mismos.

En la segunda parte se realizaron chequeos de errores no documentados, se hizo el inicio del montaje de ambos sistemas a plataformas de hosting como “Railway app” y “Netlify”, y adicional se utilizaron contenedores virtuales con Docker para optimizar espacio en disco y el re uso de tecnologías instalables como npm (Node Package Manager) y módulos internos de AngularJS y ReactJS, como también el montaje de la base de datos de MongoDB para su reutilización.

3.1.1.5 Etapa Pruebas:

La etapa de pruebas, es la etapa previa al final de la realización del proyecto, en esta fase se tuvieron en cuenta ambos entornos de desarrollo, sus componentes y el funcionamiento del sistema en general, en esta fase se realizaron las pruebas de rendimiento, enfocadas en conocer y evidenciar las diferencias no visibles de ambas tecnologías en contraposición (AngularJS y ReactJS), se utilizaron páginas web como “PageSpeed Insights” y “WebPageTest” para conocer las diferencias de rendimiento más visibles y fáciles de detectar, complementando el criterio con extensiones específicas para AngularJS como “Angular DevTool” y para ReactJS “React DevTools” sirviendo de jueces más profundos.

También se utilizaron herramientas como “Lighthouse” proporcionada por google, y Apache JMeter, sirviendo de aplicación para un resultado más completo y diferenciador de los demás implantados anteriormente, se pusieron a juicio y se notaron diferencias entre cada veredicto.

3.1.1.6 Etapa Post Mortem:

Etapla final del proyecto, siendo la etapa de documentación y análisis de los resultados finales, en esta etapa se realizó la discusión de los resultados proporcionados en la etapa anterior, se hicieron contraposición de todos, se determinó el veredicto y respuesta a la pregunta problema planteada al inicio del proyecto y se efectuó una debida descripción del paso a paso en cada fase anterior para el orden de los resultados.

4. RESULTADOS

4.1 Introducción a los Resultados

Como resultado de la investigación, se puede realizar un seguimiento exhaustivo a cada cliente que se registre en la base de datos del establecimiento sector servicios en Santiago de Cali. Esto permitió mejorar la retención de los clientes, ya que se conocen sus gustos y preferencias, Con esto el sistema será capaz de adaptar las experiencias y las ofertas de manera individualizada para cada uno de los clientes. Además de esto, se espera un incremento en las ventas debido a la mejora en la satisfacción del cliente y a la capacidad de ofrecer promociones y ofertas relevantes para cada uno de ellos, de acuerdo con los objetivos planteados, se espera obtener los siguientes resultados:

4.1.1 Resultados de la Etapa Planeación y Diseño:

Durante esta etapa de planeación, se realiza una lista de requerimientos funcionales y no funcionales de la aplicación web los cuales se verán reflejados en los siguientes mockups [Anexo](#), también se definen requisitos mínimos de los componentes hardware para un correcto funcionamiento.

4.1.1.1 Requerimientos Funcionales:

- El software posee un inicio de sesión para administradores y usuarios.
- El software debe verificar que el ingreso sea correcto.
- El software debe permitir al usuario crear, editar y eliminar pedidos.
- El software debe permitir a los administradores crear, editar y eliminar categorías de productos de la aplicación.
- El software debe permitir a los administradores enviar correos personalizados a los usuarios.
- El software debe permitir a los administradores personalizar la fecha de envío de los correos personalizados a los usuarios.
- El software debe permitir consultar, editar la información del perfil de administradores o usuarios.
- El software debe permitir a los administradores o usuarios previsualizar su historial de pedidos.
- El software debe permitir a los administradores o usuarios ver gráficas relacionadas con la cantidad de pedidos y sus motivantes
- El software debe permitir a los administradores visualizar más a fondo el historial de otros usuarios no administradores.
- El software debe permitir a los administradores visualizar gráficas de ingresos y egresos.
- El software debe permitir a los administradores poner filtros a las gráficas para ver información específica

4.1.1.2 Requerimientos No Funcionales:

- El software debe ser escalable para futuros cambios.
- La longitud de caracteres en el número de celular debe ser = 10 (Colombia)
- La contraseña debe tener un carácter especial o mayúscula.

4.1.1.3 Casos de Uso

Los escenarios de caso de uso que se eligieron para la creación del sistema son los que se detallan a continuación:

Registro	Actores	Usuarios nuevos
	Descripción	Permite realizar el registro de los usuarios nuevos, con información personal y de forma segura
Inicio de sesión	Actores	Usuarios, Administradores
	Descripción	Permite ingresar de manera segura al sistema con las credenciales creadas en el registro.
Modificar perfil	Actores	Usuarios, Administradores
	Descripción	Permite realizar cambios en el perfil, con la información registrada en el sistema y en su base de datos
Crear, Eliminar, Editar pedido	Actores	Administradores, Usuarios
	Descripción	Permite realizar un pedido, adicionar productos, eliminarlos, modificar cantidad y enviar pedido.
Cambiar pedidos de estado	Actores	Administradores
	Descripción	Permite cambiar el estado de un pedido (en curso-entregado), de cualquier usuario.
Crear, editar, eliminar categorías	Actores	Administradores
	Descripción	Permite realizar acciones básicas (CRUD) de las categorías en el sistema, visibles ante todos los usuarios y administradores.
Ver historial	Actores	Administradores, Usuarios
	Descripción	Permite conocer el historial de los pedidos realizados por la cuenta actual en el sistema.
Ver historial por usuario	Actores	Administradores
	Descripción	Permite conocer el historial de los pedidos realizados por un usuario en específico del sistema (dependiendo consulta personalizada)
Ver resumen de activos	Actores	Portal administrativo
	Descripción	Permite a los administradores revisar el resumen de activos del sistema en base al día anterior a la consulta.
Ver Análisis de interacciones	Actores	Administradores
	Descripción	Permite a los administradores del sistema poder ver las gráficas generadas en base a las interacciones de los demás administradores y usuarios (pedidos) y su balance mensual.
Ver, Crear, Editar, Eliminar correos	Actores	Administradores

personalizados	Descripción	Permite a los administradores realizar acciones básicas (CRUD) en base a correos electrónicos para los demás usuarios del sistema, adicional, permite configurar fecha para el envío de los mismos.
Cierre de sesión	Actores	Administradores, Usuario
	Descripción	Permite a los usuarios y administradores del sistema, realizar el cierre de sesión de manera segura.

Tabla 01: Escenarios de Caso de Uso - Elaboración propia

En la **Tabla 1.** Se describen los escenarios de caso de uso descritos en materiales y métodos (punto 2), y su funcionamiento, así como una correcta descripción de actores y funcionalidades.

A continuación, se presenta el diagrama de caso de uso:

Diagrama caso de uso

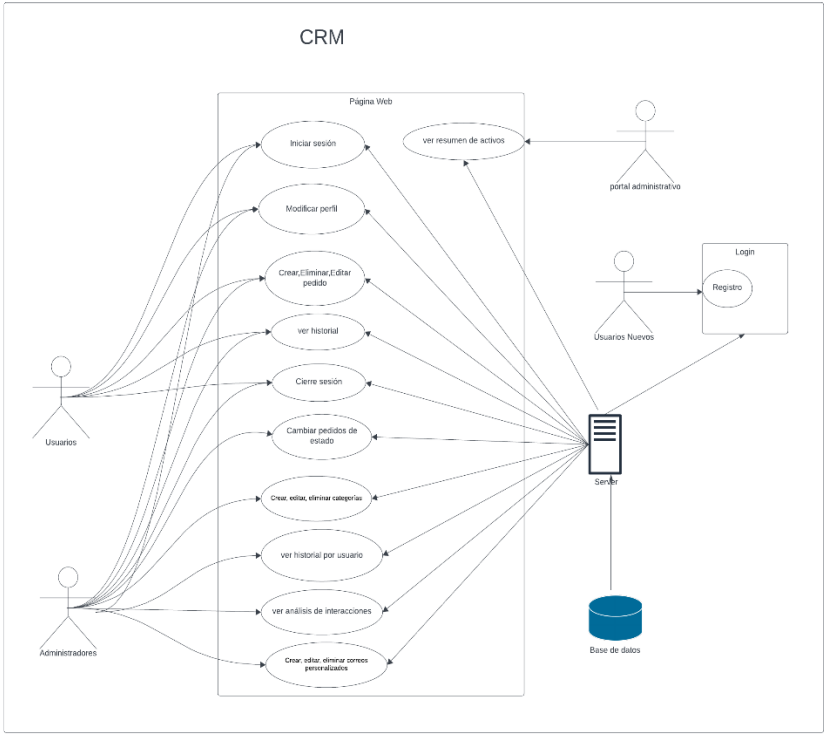


Figura 02: Diagrama de casos de uso - Elaboración propia

Es fundamental mantener el diseño acordado previamente para asegurar que el software pueda ser escalable con el tiempo, mantenerse fácilmente. En el diagrama de actividades correspondiente a la vista lógica, que muestra el proceso de la aplicación de forma visual, en el siguiente enlace: [Anexo1](#)

4.1.2 Resultado de la Etapa Código:

Durante la etapa de codificación, se realizó la construcción en materia del sistema en ambos ambientes de desarrollo, siguiendo las métricas establecidas en las etapas anteriores, sirviendo como ayuda práctica herramientas de codificación como Visual Studio Code, GitHub Copilot, IA Bard, entre otras documentaciones.

El proyecto está planteado en 2 entornos de desarrollo “stack MEAN y MERN”, sirviendo como tecnologías principales y su descripción de uso las siguientes:

1. AngularJS para la codificación del fron-end o vista de usuarios, en palabras resumidas es el aspecto visual que tendría el sistema en su ambiente propio.
2. ReactJS al igual que la anterior, sirve para la codificación visual en el segundo ambiente de desarrollo a contraposición para las pruebas pertinentes en las futuras etapas.
3. MongoDB sirviendo de base de datos principal del proyecto, para ambos entornos de desarrollo y siendo parte esencial de la pertinencia de datos y su reutilización continúa.
4. NodeJS entorno de ejecución del lenguaje JavaScript, del lado del servidor, para la lógica de las funcionalidades especificadas en las anteriores fases.
5. ExpressJS framework para la codificación de la lógica del servidor (back-end) y sirviendo para la utilización efectiva de la base de datos MongoDB y sus conexiones con AngularJS y ReactJS en sus entornos de desarrollo.
6. Docker siendo la herramienta para la automatización, creación y despliegue de contenedores para el entorno de pruebas en la codificación antes de subir el proyecto a las plataformas de hosting final.
7. GitHub siendo el portal para guardar el progreso del proyecto en cada etapa de la codificación y sirviendo para el control de versiones que se presenten en este mismo desarrollo, parte esencial para evitar la pérdida de progreso y datos.

Para evidenciar la codificación pertinente y realizada en esta fase se precisa del siguiente [Anexo](#)

También se presenta el diagrama de la arquitectura del sistema, para entender todo el comportamiento de las transacciones:

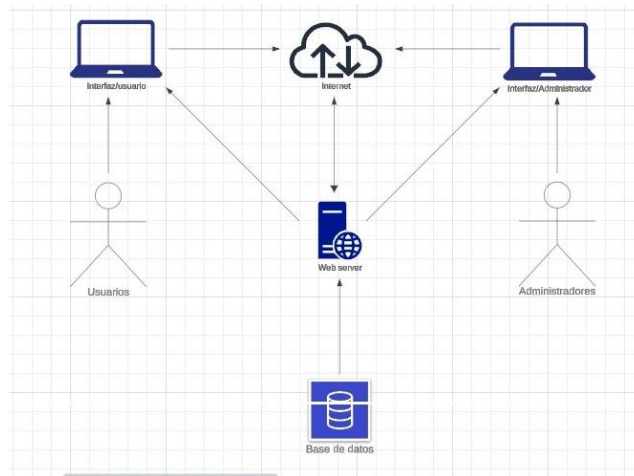


Figura 03: Diseño Arquitectura de Sistemas- Elaboración propia

4.1.3 Resultado de la Etapa Compilación:

Durante el proceso de compilación, se contemplaron diferentes fallas y errores en la ejecución del sistema, por ejemplo, en los módulos de registro e inicio de sesión de AngularJS, se presentaron fallas de cookies en el navegador y errores en el uso de expresiones regulares para manejar la información usada y poder clasificarla como segura o no segura, en ReactJS, se precisaba errores de inicio de sesión y en las peticiones HTTPS.

En los módulos de creación de pedidos, se presentaba errores en manejar información previa, y se veía duplicada al intentar hacer nuevas transacciones, estos errores fueron claves para la optimización del uso del sistema, mejorando la calidad de la arquitectura y posibles conflictos futuros con las pruebas de rendimiento en etapas posteriores.

Esta etapa es clave para la corrección de errores no previstos, manejo de la calidad y efectiva optimización de acciones en los casos de uso antes planteados.

4.1.4 Resultado de la Etapa Pruebas:

Durante la etapa de pruebas, se realizaron diferentes ejecuciones en entornos divididos como lo son “local” y “producción”, esto con el fin de poder diferenciar los comportamientos del sistema y sus variables de rendimiento, utilizando diferentes servidores de alojamiento contra un entorno local, ya que, para el entendimiento de los resultados se disponen de tiempos de respuesta, que pueden variar, dependiendo del entorno de ejecución, es por ello, que se decidió dividir la etapa de pruebas, para encontrar las diferencias en estos entornos y así facilitar los resultados siguiendo los lineamientos que las herramientas utilizadas exigen para un correcto funcionamiento de sus procesos.

Para ambos entornos de pruebas, se utilizaron las siguientes configuraciones de hardware:

PC 1:

- a. Procesador Intel i5-8250
- b. 16GB RAM 2100mhz
- c. HDD 1tb
- d. Intel uhd 620
- e. Windows 10 home 64 bits

PC 2:

- a. Procesador Intel i3-8100
- b. 16 GB RAM 2400mhz
- c. HDD 1tb
- d. SSD 250GB
- e. Windows 10 Pro 64 bits

4.1.4.1 Resultados en entorno local:

Para el entorno de desarrollo, se dispone de los siguientes componentes para la ejecución de ambos CRM en sus respectivos Stacks tecnológicos:

1. **Servidor Backend:** El servicio backend para las funcionalidades del lado del servidor están habilitadas para su funcionamiento, mediante un servicio de alojamiento en la plataforma web [Render](#) esto con el fin de contar con un servicio separado de los demás en casos de errores y poder manejarlo de forma más segura.
2. **Base de datos:** El servicio de base datos MongoDB, para la administración de la información guardada y de consulta de ambos sistemas CRM, está alojada mediante la plataforma web [MongoDB Atlas](#) con el fin de asegurar la integridad de la información y tener monitoreo permanente de las consultas y conexiones realizadas tanto en la codificación como las pruebas.
3. **Cliente React:** El montaje de la plataforma CRM en su apartado visual usando la librería ReactJS para consumo final, está alojado localmente en el pc1 en lugar de depender de un servidor remoto, de esta manera se puede acceder y trabajar en el software desde el entorno de desarrollo local
4. **Cliente Angular:** El montaje de la plataforma CRM en su apartado visual usando el framework AngularJS para consumo final, está alojado localmente en el pc1 en lugar de depender de un servidor remoto, de esta manera se puede acceder y trabajar en el software desde el entorno de desarrollo local.

Siguiendo con la realización de las pruebas en el entorno local, se utilizaron diferentes herramientas para la medición de rendimiento y parámetros de buenas prácticas para aplicativos webs, bien sean:

1. Lighthouse: extensión para navegadores basada en Chrome DevTools sirviendo como herramienta para la medición de rendimiento en páginas web utilizando estándares de calidad actuales.
2. Angular Dev Tools: extensión para navegadores basada en Chrome DevTools sirviendo como herramienta

para perfilar, visualizar un árbol con la estructura de componentes. editar las propiedades de sus componentes y depurar aplicaciones hechas con AngularJS.

3. React Dev Tools: extensión para navegadores basada en Chrome DevTools sirviendo como herramienta para inspeccionar elementos de ReactJS, editar los props y estado, a su vez dándonos el tiempo en (ms) que demora cada componente e identificar problemas de rendimiento.

A continuación, se presentarán diferentes tablas de resultados denotando los parámetros de la medición, su respectivo ambiente probado y los resultados arrojados en un resumen realizado en conjunto con todas las herramientas utilizadas, adicional se realiza del siguiente [Anexo](#) para el entendimiento de las pruebas realizadas y su ejemplificación.

Interfaz de usuario			
		AngularJS	ReactJS
Módulo	Parámetro	Tiempo (ms)	
Historial	Renderizado componentes	±40	±30
	Bloqueo precarga de componentes	0	0
	Carga de diseños	0	0
Pedidos	Renderizado componentes	±60	±50
	Bloqueo precarga de componentes	±30	±30
	Carga de diseños	±0.064	±0.012
Perfil	Renderizado componentes	±40	±30
	Bloqueo precarga de componentes	0	0
	Carga de diseños	0	0

Tabla 02: Análisis módulos de interfaz de usuario (Entorno Desarrollo)-Elaboración propia

Interfaz de administrador			
		AngularJS	ReactJS
Módulo	Parámetro	Tiempo (ms)	
Estadísticas	Renderizado componentes	±60	±20
	Bloqueo precarga de componentes	0	0
	Carga de diseños	0	0
Email	Renderizado componentes	±70	±100
	Bloqueo precarga de componentes	0	0
	Carga de diseños	±0.002	0
Categorías	Renderizado componentes	±90	±60
	Bloqueo precarga de componentes	±100	0
	Carga de diseños	±0.06	±0.005

Tabla 03: Análisis módulos de interfaz de Adminstrador (Entorno Desarrollo)-Elaboración propia

Para realizar la siguiente tabla, se presentó un desafío a la hora de llevar a cabo el análisis , ya que estas herramientas no están enfocadas al análisis de rendimiento, pero brindan una herramienta bastante interesante , la cual nos permite conocer el tiempo de ejecución que lleva a cabo cada componente, en base a esto , podemos analizar el comportamiento de cada módulo, escogiendo el componente que demora mas tiempo , para luego compararlo con el mayor tiempo que tomó el componente del otro Stack tecnológico.

AngularJS	ReactJS
-----------	---------

Módulo	Mayor tiempo consumido por comp. (ms)	Mayor tiempo consumido por comp. (ms)
Login	±24	±12
Resumen	±8	±4
Análisis	±13	±9
Emails	±18	±13
Hsitorial	±36	±9
Pedidos	±27	±7
Categorías	±34	±11
Perfil	±22	±4

Tabla 04: Análisis módulos de interfaz de Administrador y usuario (Entorno Desarrollo)-Elaboración propia

4.1.4.2 Resultados en entorno de producción:

Para el entorno de producción, se dispone de los siguientes componentes para la ejecución de ambos CRM en sus respectivos Stacks tecnológicos:

1. **Servidor Backend:** El servicio backend para las funcionalidades del lado del servidor están habilitadas para su funcionamiento, mediante un servicio de alojamiento en la plataforma web [Render](#) esto con el fin de contar con un servicio separado de los demás en casos de errores y poder manejarlo de forma más segura.
2. **Base de datos:** El servicio de base datos MongoDB, para la administración de la información guardada y de consulta de ambos sistemas CRM, está alojada mediante la plataforma web [MongoDB Atlas](#) con el fin de asegurar la integridad de la información y tener monitoreo permanente de las consultas y conexiones realizadas tanto en la codificación como las pruebas.
3. **Cliente React:** El montaje de la plataforma CRM en su apartado visual usando la librería ReactJS para consumo final, está alojado mediante la plataforma web [Netlify](#) en el siguiente [link](#) para su uso pertinente tanto en pruebas como uso de usuarios finales.
4. **Cliente Angular:** El montaje de la plataforma CRM en su apartado visual usando el framework AngularJS para consumo final, está alojado mediante la plataforma web [Netlify](#) en el siguiente [link](#) para su uso pertinente tanto en pruebas como uso de usuarios finales.

Siguiendo con la realización de las pruebas en el entorno de producción, se utilizaron diferentes herramientas para la medición de rendimiento y parámetros de buenas prácticas para aplicativos webs, bien sean:

1. **Lighthouse:** extensión para navegadores basada en Chrome DevTools sirviendo como herramienta para la medición de rendimiento en páginas web utilizando estándares de calidad actuales.
2. **PageSpeed Insights:** Página web que proporciona información sobre tiempos de respuesta, manejo de peticiones, rendimiento general, configuraciones y accesibilidad del sitio.
3. **WebPageTest:** como su nombre lo indica, es una página web, para la medición de rendimiento, accesibilidad, buenas prácticas y tiempos de renderizado.

A continuación, se presentarán diferentes tablas de resultados denotando los parámetros de la medición, su respectivo ambiente probado y los resultados arrojados en un resumen realizado en conjunto con todas las herramientas utilizadas, adicional se realiza del siguiente [anexo](#) para el entendimiento de las pruebas realizadas y su ejemplificación.

Interfaz de usuario			
		AngularJS	ReactJS
Módulo	Parámetro	Tiempo (ms)	
Historial	Renderizado componentes	±40	±20
	Bloqueo precarga de componentes	±30	30
	Carga de diseños	±0.04	±0.026

Pedidos	Renderizado componentes	± 60	± 60
	Bloqueo precarga de componentes	± 30	± 20
	Carga de diseños	± 0.064	± 0
Perfil	Renderizado componentes	± 0.1	± 0.1
	Bloqueo precarga de componentes	± 30	± 40
	Carga de diseños	± 0.001	± 0

Tabla 05: Análisis módulos de interfaz de usuario (Entorno Producción)-Elaboración propia

Página de inicio			
		AngularJS	ReactJS
Vista	Parámetro	Tiempo (ms)	
Escritorio	Renderizado de componentes	± 220	± 0
	Velocidad de carga del index	± 54	± 42
	Carga de diseños	± 42	± 42
	Retraso de renderizado header	± 420	± 560
Móvil	Renderizado de componentes	± 530	± 70
	Velocidad de carga del index	± 186	± 120
	Carga de diseños	± 100	± 100
	Retraso de renderizado header	± 2400	± 2160

Tabla 06: Análisis módulo de inicio vistas móvil y escritorio (Entorno Producción) - Elaboración propia

Interfaz de administrador			
		AngularJS	ReactJS
Módulo	Parámetro	Tiempo (ms)	
Estadísticas	Renderizado componentes	± 60	± 20
	Bloqueo precarga de componentes	± 0.1	± 0.1
	Carga de diseños	± 0.001	± 0.009
Email	Renderizado componentes	± 60	± 50
	Bloqueo precarga de componentes	± 0.1	± 0.01
	Carga de diseños	± 0.003	± 0.004
Categorías	Renderizado componentes	± 60	± 30
	Bloqueo precarga de componentes	± 110	± 0.1
	Carga de diseños	± 0.0046	± 0.0026

Tabla 07: Análisis módulos interfaz de administrador (Entorno Producción)- Elaboración propia

5. DISCUSIÓN

5.1 Introducción a la Discusión:

A partir de los resultados obtenidos, se puede concluir que la metodología Personal Software Process (PSP), demostró ser eficaz para llevar a cabo La investigación. Además, los datos recopilados y tratados abordan de manera óptima la conclusión en favor al uso de esta metodología para procesos de investigación y desarrollo de software, así mismo, se efectuó de manera esperada, la migración y actualización del sistema con sus pruebas finales respectivas.

5.1.1 Discusión de los Resultados de la Etapa Planeación:

En la fase inicial del proyecto, se encontraron diferentes formas de realizar la investigación y el desarrollo del sistema, pero, es fundamental comprender los requerimientos funcionales y no funcionales que se requieren alcanzar, para ello se tuvieron que realizar reuniones y discusiones del objetivo final del sistema y el rumbo de la investigación, a pesar que hubo inconvenientes como en cualquier trabajo investigativo, se realizó de manera óptima y confiable la estructuración pertinente de dichos requerimientos, donde se dejó constancia y documentación para ello.

5.1.2 Discusión de los Resultados de la Etapa Diseño:

La etapa de diseño comprende un paso a paso fundamental para la investigación y desarrollo, puesto que con esta fase se posee una idea general pero ilustrativa del resultado final a alcanzar, en esta etapa se tuvieron inconvenientes en el manejo y administración de las herramientas de diseño utilizadas, pero la documentación encontrada en la web fue de vital importancia para el uso correcto de las mismas, un aprendizaje profundo y un entendimiento de cómo realizar una abstracción es el resultado de un arduo trabajo para documentar e ilustrar las funcionalidades más importantes y el comportamiento de un sistema, esta etapa deja como resultados las habilidades para diseñar y elaborar documentación ilustrativa sobre la arquitectura de un sistema, más allá de su tipología, sus parámetros y necesidades primordiales.

5.1.3 Discusión de los Resultados de la Etapa Código:

La etapa de Código marca el rumbo más importante del proyecto, siendo la etapa donde más recursos se deben utilizar, es la etapa donde más problemas, tanto de desarrollo como de tiempo, se pueden presentar, en esta etapa, se encontraron diversos inconvenientes, desde la compatibilidad de versionamiento, como el aprendizaje y uso de buenas prácticas de desarrollo para la codificación respectiva de todas las funcionalidades planteadas anteriormente, además se pudo evidenciar las curvas de aprendizaje presentadas en el uso de tecnologías contrapuestas como ReactJS y AngularJS, donde se dejó en claro cómo un mismo fin es posiblemente alcanzado por más de una forma en el ámbito TI.

5.1.4 Discusión de los Resultados de la etapa Compilación:

Siendo aliada de la etapa anterior, la etapa de compilación es una etapa en donde no solo, se soporta de la codificación efectiva, sino, se hace uso del software en mayor medida, en esta etapa se presentaron y corrigieron errores de codificación, se realizaron reuniones en su mayoría del tiempo, donde la colaboración verbal fue clave para el desarrollo de esta etapa, además de utilizar el sistema de formas tanto como usuario y desarrolladores, se evaluaron nuevas formas para optimizar y mejorar el mismo, en esta etapa se hizo un exhaustivo análisis de todos los requerimientos planteados en etapas anteriores, se evaluó su desarrollo y se concluyó con el sistema en ambos entornos totalmente funcional y listo para realizar las pruebas finales y de conclusión.

5.1.5 Discusión de los resultados de la etapa Pruebas:

En esta etapa ya casi final, a pesar de tener muchos pasos planeados y herramientas a la mano, se presentaron desafíos, los cuales se llegó a profundizar y seleccionar con cuidado las herramientas a usar, ya que cada una da diferentes resultados, se tenía que elegir la herramienta que nos diera la mejor información, la mas precisa y completa, claramente que estuviesen de la mano de los criterios los cuales se habían decidido medir, después de recopilar tanta información de las diferentes herramientas, se clasifica y organiza la información de tal manera que a simple vista sea fácil notar e interpretar las diferencias de rendimiento que poseen los diferentes ambientes desarrollados para el CRM.

6. CONCLUSIONES

6.1 Conclusiones de la Etapa Planeación:

Durante esta etapa, se lograron plantear los objetivos principales de la investigación, los requerimientos funcionales y no funcionales y adicional el rumbo del desarrollo del sistema en ambos entornos, así mismo se realizó el inicio efectivo para el proyecto, concluyendo con las bases para la realización del proyecto.

6.2 Conclusiones de la etapa Diseño:

La etapa de diseño, consolidó efectivamente la elección de patrones de diseño a utilizar, se crearon visualmente

las funcionalidades anteriormente planteadas y además se utilizaron herramientas de diagramación para el correcto planteamiento del comportamiento final del sistema, finalizando esta etapa con el diseño completo necesario para el entendimiento del sistema.

6.3 Conclusiones de la Etapa Código:

La codificación efectiva de un sistema es el corazón de un buen desarrollo y una investigación efectiva, esta etapa demuestra que tener en conjunto de buenos lineamientos y objetivos estructurados, de la mano con las funcionalidades de las anteriores etapas deja como consecuencia un código con buenas prácticas y tecnologías anidadas de forma efectiva para el funcionamiento del sistema en ambos entornos.

6.4 Conclusiones de la Etapa Compilación:

Realizar de forma exhaustiva pruebas de calidad de software es fundamental en el momento del desarrollo, antes de poder ejecutar un sistema en forma final, se deben solucionar problemas que solo son evidentes en la utilización por usuarios y desarrolladores, es así como esta etapa concluye con la ejecución del sistema de forma efectiva y solucionando todos los problemas pertinentes y no contemplados, así mismo con su montaje completo a los entornos de pruebas que se realizaron en las etapas posteriores.

6.5 Conclusiones de la Etapa Pruebas:

En esta etapa sucede toda la magia, es lo que afirma todo lo planeado, pensado y analizado, aclara las dudas sobre cual entorno de desarrollo posee ciertas ventajas sobre el otro, cabe resaltar que en ambos entornos se ha venido trabajando en cambios y mejoras, buscando que su objetivo final el cual es el cliente, encuentre una alternativa la cual le haga sentirse satisfecho y con el mejor rendimiento posible, una vez ya analizado e interpretado los resultados nos damos cuenta de las diferencias del tiempo de carga, el tiempo de renderización de componentes, carga de diseños e impacto en la experiencia de usuario, encontrando campos o variables a mejorar, cual entorno es el más óptimo en ciertos procesos, etc.

6.6 Conclusiones de la Etapa Post-Mortem(final):

Como conclusión tenemos que esta migración fue motivada por algunas de las altas tasas de renderizado y poco rendimiento asociado con el Stack mean, especialmente con el uso de AngularJS, el CRM permitió medir el cambio que supone contrastar dos entornos de desarrollo de un mismo sector tecnológico, los cuales tienen diferentes estructuras de codificación, esto se tradujo en una notable diferenciación frente a los temas de rendimiento y respuesta del CRM, luego de interpretar cada resultado obtenido, podemos ver que en este análisis el entorno de desarrollo MERN el cual usa ReactJS, presenta un mejor resultado comparado a AngularJS, con el rendimiento y renderizado de componentes y carga de diseños, se termina de confirmar cuando se realiza la medición componente a componente de cada stack, donde el componente que más tiempo demora de AngularJS es 36ms en renderizarse mientras que el más alto de ReactJS es 13ms, dando una clara respuesta de cual entorno de desarrollo es más óptimo y traerá mayor satisfacción y un mejor impacto en la experiencia del cliente, teniendo en cuenta que el cliente prefiere un entorno de desarrollo el cual tenga un buen rendimiento de la mano con unos buenos tiempos de respuesta.

7. REFERENCIA

- BAIYEWU, O. (2022). *THE IMPACT OF CUSTOMER RELATIONSHIP MANAGEMENT ON ORGANIZATIONAL PERFORMANCE*. Degree Programme and place of study: Master of Business Administration, Vaasa.
- Bernal, L. B. (2021). Gestión de las relaciones con el cliente:. *Gestión en el Tercer Milenio*, 8.
- Czuper, M. (2022). *Applying automated performance testing with Apache JMeter*. Espoo: Aalto University School of Science.
- Gil-Vera, V. D. (2018). MACHINE LEARNING APLICADO AL ANÁLISIS DEL. *Revista Politécnica ISSN*, 13.

- S.A., K. (2021). OVERVIEW AND JUSTIFICATION FOR CHOOSING TECHNOLOGY. *National Aviation University*, 10.
- Subramanian, V. (2019). *Pro MERN Stack Second Edition*. Bangalore, Karnataka, India : Apress Media.
- Thilanka Kaushalya, I. P. (2021). Framework to migrate AngularJS based legacy web application to react component architecture. *University of Moratuwa*, 7.
- Vazquez, Z. (2020). Desarrollo de aplicaciones web utilizando. *UNIVERSIDAD POLITÉCNICA DE SINALOA*, 56.
- Colares de Camargos, J. G., Coelho, J. F., Aramuni, J. P., & Fernandes Villela, H. (2019). Uma Análise Comparativa entre os Frameworks Javascript Angular e React. *Computação e Sociedade*, 1, 101–113.
- Kaushalya, T., & Perera, I. (2021). Framework to migrate AngularJS based legacy web application to react component architecture. *MERCon 2021 - 7th International Multidisciplinary Moratuwa Engineering Research Conference, Proceedings, August*, 693–698. <https://doi.org/10.1109/MERCon52712.2021.9525659>
- Komperla, V., Pratiba, D., Ghuli, P., & Pattar, R. (2022). React: A detailed survey. *Indonesian Journal of Electrical Engineering and Computer Science*, 26(3), 1710. <https://doi.org/10.11591/ijeecs.v26.i3.pp1710-1717>
- Konrad Bielak, Bartłomiej Borek, M. P.-W. (2020). *Comparación del rendimiento de las herramientas para crear una interfaz de SPA usando el ejemplo de React y Vue . js*. 14, 73–77.
- Pla, A., & Fairén, P. M. (2019). *a React y estudio de algoritmos de ES6 Memoria Índice general*.