UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

Tema:

APLICACIÓN MULTIPLATAFORMA PARA LA GESTIÓN Y CONTROL DE PROCESOS DE MICROEMPRESAS DE DISEÑO Y CORTE LASER ORIENTADAS A LA TOMA DE PEDIDOS UTILIZANDO FULL STACK MERN.

Trabajo de Titulación Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Sistemas Computacionales e Informáticos.

ÁREA: Software

LÍNEA DE INVESTIGACIÓN: Desarrollo de Software.

AUTOR: Jonathan Josué Vélez Llerena

TUTOR: Ing. Edison Homero Álvarez Mayorga

Ambato - Ecuador

septiembre - 2021

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación con el tema: APLICACIÓN MULTIPLATAFORMA PARA LA GESTIÓN Y CONTROL DE PROCESOS DE MICROEMPRESAS DE DISEÑO Y CORTE LÁSER ORIENTADAS A LA TOMA DE PEDIDOS UTILIZANDO FULL STACK MERN, desarrollado bajo la modalidad Proyecto de Investigación por el señor Jonathan Josué Vélez Llerena, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, septiembre 2021

Firmado electrónicamente por:
EDISON HOMERO
ALVAREZ MAYORGA

Ing. Edison Homero Álvarez Mayorga
TUTOR

AUTORÍA

El presente Proyecto de Investigación titulado: APLICACIÓN MULTIPLATAFORMA PARA LA GESTIÓN Y CONTROL DE PROCESOS DE MICROEMPRESAS DE DISEÑO Y CORTE LÁSER ORIENTADAS A LA TOMA DE PEDIDOS UTILIZANDO FULL STACK MERN, es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, septiembre 2021

Jonathan Josué Vélez Llerena

C.C: 0706606167

AUTOR

APROBACIÓN TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Jonathan Josué Vélez Llerena, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado APLICACIÓN MULTIPLATAFORMA PARA LA GESTIÓN Y CONTROL DE PROCESOS DE MICROEMPRESAS DE DISEÑO Y CORTE LÁSER ORIENTADAS A LA TOMA DE PEDIDOS UTILIZANDO FULL STACK MERN, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidenta del Tribunal.

Ambato, septiembre 2021

Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Elsa Pilar Urrutia, Mg.
PRESIDENTA DEL TRIBUNAL

Firmado electrónicamente por:
HERNAN FABRICIO NARANJO AVALOS

Ing. Hernán Fabricio Naranjo
PROFESOR. CALIFICADOR

Firmado electrónicamente por:
OSCAR FERNANDO IBARRA TORRES

Ing. Oscar Fernando Ibarra
PROFESOR CALIFICADOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, septiembre 2021

Jonathan Josué Vélez Llerena

C.C: 0706606167

AUTOR

DEDICATORIA

Mi tesis la he dedicado a mi familia.

Jonathan Josué Vélez Llerena

AGRADECIMIENTO

Primeramente, a Dios por darme la vida y la inteligencia para cumplir con esta meta. A mis padres que siempre me han apoyado y aun en la distancia han buscado la forma de ayudarme.

Gracias a mis amigos en Ambato, cada uno ha formado en mi crecimiento personal, a la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, mi alma mater y a todos los docentes que no solo me brindaron conocimiento, sino que han corregido el rumbo de mi vida con valores y experiencia, al personal administrativo, en especial a Gabriela, quien siempre tuvo las palabras adecuadas en el momento preciso, que motivaban a seguir con mi camino universitario.

Gracias a mi tutor de Tesis, ing. Edison Álvarez, que en medio de la situación actual que atravesamos, con sabiduría y paciencia ha logrado asesorarme y guiarme durante el desarrollo de mi proyecto de investigación.

Jonathan Josué Vélez Llerena

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN EJECUTIVO

La implementación y control de procesos en una organización es fundamental y relevante en la definición del flujo de trabajo de sus tareas y actividades. , Estos procesos pueden ser manejados de manera manual o automática, sin embargo, los errores que se puedan presentar son directamente proporcionales al uso de la mano obrera, en otras palabras, por más que un sistema sea implementado de manera satisfactoria, siempre va a existir un riesgo en su manipulación por la intervención humana.

Una aplicación informática optima, se basa en aspectos fundamentales como ser: intuitivita, ergonomía, flexibilidad, portabilidad y sobretodo la usabilidad, donde el usuario no interactúe demasiado en el aplicativo, sea este móvil, CRM, web, entre otros. Esta tipología de desarrollo de software es muy costosa y requiere de mucho tiempo para que las microempresas opten por adquirir dicho producto. Siendo el fin de este trabajo investigativo ofrecer un sistema multiplataforma para el control de procesos de pedidos, bajo nuevas tecnologías de programación, fortaleciendo la refactorización y reutilización de código, con Full Stack.

La seguridad será una de las bases de este trabajo, garantizando el control de ingresos de los usuarios, codificando campos delicados y encriptando contraseñas, dándole un plus con la usabilidad de base de datos no relacionables como las es Mongo, evitando la inyección SQL.

El proyecto fue desarrollado en Visual Studio Code, alojando en un archivo la base de datos en Mongo, creando el BackEnd con Node.js, y el manejo de la aplicación y rutas con Express.js, devolviendo las apis necesarias las cuales se conectarán con el otro archivo con código fuente en React y React Native, explotando el uso de Hooks y reactividad.

Palabras clave: Full Stack, MERN, NoSQL, Callback Hell.

ABSTRACT

The control of processes within an organization is very important to solve and solve various problems that may be had, these can be done manually or automatically, however, the errors that may occur are directly proportional to the use of labor In other words, no matter how perfect a system is, it will always have a vulnerability and the user will do.

An optimal computer system is based on intuitiveness, ergonomics, flexibility, portability and above all usability, where the user does not interact too much in the application, be it mobile, CRM, web, among others. This type of software development is very expensive and time-consuming for micro-businesses to choose to purchase such a product. The purpose of this research work being to offer a multiplatform system for order process control, under new programming technologies, strengthening code refactoring and reuse, with Full Stack.

Security will be one of the bases of this work, guaranteeing the control of user income, coding sensitive fields and encrypting passwords, giving it a plus with the usability of unrelated databases such as Mongo, avoiding SQL injection.

The project was developed in Visual Studio Code, hosting the Mongo database in a file, creating the BackEnd with Node.js, and managing the application and routes with Express.js, returning the necessary apis which will connect with the other file with source code in React and React Native, exploiting the use of Hooks and reactivity.

Keywords: Full Stack, MERN, NoSQL, Callback Hell.

CAPÍTULO I:

MARCO TEÓRICO

1.1. Tema de Investigación

APLICACIÓN MULTIPLATAFORMA PARA LA GESTIÓN Y CONTROL DE PROCESOS DE MICROEMPRESAS DE DISEÑO Y CORTE LÁSER ORIENTADAS A LA TOMA DE PEDIDOS UTILIZANDO FULL STACK MERN.

1.2. Antecedentes Investigativos

1.2.1. Contextualización del problema

La variable tecnológica ha estado en constantes cambios evolutivos, acompañado de las nuevas generaciones, las cadenas televisivas, noticias circuladas en las redes sociales, hacen alusión de como los smarthpones han impactado la vida de los seres humanos. Hoy en día mas del 50% de la población mundial tiene acceso a internet y según el trabajo investigativo "JavaScript en aplicaciones móviles: React Native vs Ionic vs NativeScript vs desarrollo nativo", crea un estimado del 34% de las personas en todo el mundo, con al menos un celular inteligente [1].

Los autores Hugo Brito, Álvaro Santos, Jorge Bernardino y Anabela Gomes en su trabajo investigativo "Desarrollo móvil en Swif, Java y React Native: una evaluación experimental en audioguías" resaltan el inicio de los aplicativos móviles, segmentadas por su información y productividad . Los primeros celulares contaban con aplicaciones como la calculadora, calendario y en un paso innovador, correo electrónico, hoy en día se puede decir que hay herramientas para casi cada actividad. Todo este fundamento evolutivo en los aplicativos móviles han forjado una mejor calidad de vida para las personas, teniendo en claro que el usuario es el responsable de explotar dichas ventajas [2].

La escalabilidad de los aplicativos móviles representa una alteración representativa cada semestre [3],dando como resultado un incremento en el mercado móvil . Este crecimiento constante es directamente proporcional al aumento de desarrolladores [3] y por ende no es suficiente saber programar. Las viejas tecnologías están quedando

obsoletas, en la actualidad un software debe ser ergonómico, fluido, intuitivo, portable, escalable, multiplaforma, entre otros aspectos más, para competir frente a la demanda informativa [2].

La industria móvil ha generado una nueva cultura, creando nuevas necesidades y explotando el campo de ventas. Esto se debe a sus diversas herramientas o aplicaciones. En el continente asiático, el medio para cerrar ventas ha sido un portal web, pero el 90% de estos ingresos fueron ejecutados desde un aplicativo móvil [1].

El perfil del programador ha sufrido cambios constantes, con la llegada de los smathphones y dispositivos similares. Los conocimientos han tenido que ser actualizados y puestos en práctica de forma inmediata. Las habilidades de un desarrollador de software son muy significativas, en especial si el enfoque es, desarrollo nativo [4].

La diversidad de conocimiento y habilidades en la programación brinda mayores oportunidades en el campo laboral y en generación de ingresos. El top 10 de lenguajes de programación usados son C, Java, Python, C++, C#, Visual Basic, JavaScript, R, PHP (), SQL [5]. Cada uno de estos lenguajes se han mantenido tanto por la natividad como el hecho de ser multiplataforma.

Java se ha concentrado mucho en el lenguaje nativo, profundizado el área de sistemas operativos para maquinaria pesada, como un controlador de plotter. Mientras que C# ha ido evolucionando con el tiempo. Este lenguaje va desde el uso en consola, hasta ser parte vital para un video juego o ser multiplataforma con su reciente herramienta Xamarin.

La opción multiplataforma tiene diversas ventajas, la más resaltante e icónica es escribir una vez el código e implementarlo de distintas formas, introduciendo el tema de reutilización de código. Se realiza una base codificada una sola vez y se lo usa en distintas plataformas, no limitando el tipo de dispositivo a usar. Esto fue parte del concepto Write Once, Run Anywhere (WORA) [5].

Los entornos de desarrollo integrado (IDE) orientados a multiplataforma en su mayoría son de código abierto, abriendo un abanico de soporte y documentación para no perderse en el camino. En la investigación “Análisis de la aplicación móvil multiplataforma” categoriza a nivel de implementación, cuatro formas de aplicativos

móviles multiplataforma: aplicaciones web, híbridas, interpretadas y basadas en widgets [5].

Con respecto al párrafo anterior, en las tres primeras categorías existe un lenguaje de programación frecuente, JavaScript y sus frameworks que facilitan la comunicación entre capas, como lo es React. React es una herramienta muy poderosa e incluso con el lanzamiento de React Native, no solo se desarrollará un aplicativo web, sino que al ser un aplicativo interpretado, emula la natividad en un dispositivo móvil.

En sus inicios la programación multiplataforma tenía distintas falencias, una de ellas era el uso de los recursos nativos de cada hardware y SO. Java fue uno de los lenguajes innovadores, usando la virtualización como un medio de compatibilidad con distintos dispositivos, y fue de mayor impacto con el IDE de programación Android Studio, cubriendo el área móvil. A pesar de posicionarse como uno de los lenguajes mas rentables, no existía una compatibilidad con IOS móvil [4].

Android Studio y Xcode son lenguajes nativos para Android e IOS respectivamente, usando los recursos de cada dispositivo óptimamente. Los costos para generar una aplicación por SO son muy elevados. El esfuerzo y tiempo de productividad son variables importantes para calcular el costo de un aplicativo, usando el método Cocomo II.

Entendiendo este concepto de natividad exclusiva y su costo por SO, se presento una solución, opciones multiplataforma, como Xamarin, que usa C# como lenguaje base, desarrollando la lógica de negocios una sola ves y luego se especificaba el diseño de cada dispositivo. El esfuerzo y el tiempo de desarrollo disminuían parcialmente, sin embargo, no dejaba de ser costoso [9].

JavaScript con su framework React Native Expo soluciono en su mayoría el uso de recursos nativos y codificarlos una sola vez. Expo es una herramienta que virtualiza la natividad, acogiendo los recursos como suyos y traduce la codificación javascript en nativa. No todo lenguaje es perfecto, aun existen aspectos en los que se necesita natividad al momento del desarrollo, sin embargo Expo esta compuesto de una gran comunidad de desarrolladores que trabajan constantemente en el soporte e innovación [7].

En las formas de instalación de React Native en nuestro IDE preferido, hay un listado de dos opciones, la primera se basa en una instalación exhaustiva. Debe cumplir ciertos requisitos y en ciertas ocasiones es tedioso llegar al resultado final, sin embargo, al ser completa su implementación se puede detallar ciertas partes del código especificando el tipo de Sistema Operativo (SO) y emular un aplicativo nativo. Esta forma de instalación requiere juntamente con el uso de herramienta, el conocimiento de programación nativa.

Por otro lado, la instalación por medio de Expo, es más sencilla, con tan solo dos líneas de código se tiene el proyecto listo para usarse. Aún en sus limitaciones, esta herramienta sigue en constante crecimiento, permitiendo hacer aplicaciones muy potentes. Expo permite utilizar los recursos del dispositivo tanto android como el sistema operativo iPhone (iOS) y adaptarlo como si fuera el mismo. En la documentación de Expo se puede analizar que recursos se usan para la compatibilidad de plataformas, tanto para Andorid,iOS o Web, entre lo más representativos estan servicios de localización, acelerómetros, batería, cámara, FacebookAds, entre muchas más [6].

La codificación con frameworks como React facilitan el proceso del proyecto a desarrollar, lo que ha permitido a la industria del software tener un enorme crecimiento por su mayor demanda en el mundo empresarial. En el trabajo investigativo "Importancia del uso del software contable en pequeñas, medianas y grandes empresas del cantón Portoviejo" menciona el impacto económico de un sistema informático dentro de una organización, pues el manejo inteligente de información permite tomar las mejores estrategias y decisiones para la curva de crecimiento.

En dicha investigación cito "...el elemento diferenciador entre empresas supervivientes y sobrevivientes radica en el aprovechamiento de los recursos que la tecnología ofrece, y la manera en que dichos recursos son explotados..." [7], dando a entender que el correcto software y su manejo mantendrá en vigencia a la empresa.

Las micro, medianas o nuevas empresas en su mayoría no optan por la adquisición de un software a motivo de sus costos, aún cuando tengan presente los beneficios de las nuevas tecnologías. La natividad exclusiva exige un costo por cada SO o funcionalidad, y aunque la opción web ha sustituido parcialmente las aplicaciones de

escritorio, las investigaciones para su función en dispositivos móviles aun esta en progreso y en pruebas[2].

A su contraparte la opción multiplataforma en el sector interpretado, baja el costo de un sistema completo, dado que al reutilizar código, el esfuerzo, líneas de código y personal es menor.

La gestión de procesos es un punto clave en los antecedentes, la industria ha progresado con el objetivo de optimizar los procesos para la elaboración de un producto, dado que, si se formare un cuello de botella, la empresa comienza a perder dinero. Es aquí donde la frase “El tiempo es oro” da un inca-pie en la importancia de tener un registro regulado en la creación de un producto o la culminación de un servicio [8].

Otro aspecto para tratar es la importancia de la seguridad informática, los datos al igual que la tecnología, está en constante crecimiento a más de manifestarse su complejidad en las 3V (volumen, variedad y velocidad) [9].

Un gran problema de la big data ha sido el procesar las relaciones en una base de datos, y su gestión ha sido muy engorrosa, sin mencionar el hecho de la seguridad, dándose como alternativa las bases de datos NoSQL (No Structured Query Language). Al no tener un esquema fijo, el almacenamiento de datos puede ir en mayor escala y a su vez puede manejar estos datos tanto organizados y estructurados [9].

En la investigación “Prueba de liberación de datos espaciales en SQL y NoSQL” por Dany Laksono, concluye que las bases de datos NoSQL “poseen un gran potencial para aplicaciones geoespaciales en la web”

En el trabajo de investigación colaborativo de Vrinda Shachdeva y Sachin Gupta parte del Deparment of CSE titulado “Análisis básico de inyección NoSQL (no solo SQL) y detección en MongoDB” explora las vulnerabilidades en la base de datos, aclarando que la inyección SQL es la de mayor problema, y que también existe en NoSQL, lo que lleva al planteamiento de la solución por medio de codificación en la entrada de datos. Su estudio se basó en que varias entidades o empresas migraban datos SQL a

NoSQL, por su gran seguridad de datos, con el tiempo los hackers encontraron la forma de hacer inyection SQL por medio de JSON.

En la síntesis de la investigación de proyectos vinculados a la temática, los aplicativos informáticos cubren varias necesidades que a perspectiva de un microempresario puede resultar un gasto innecesario, sin embargo, al bajar costos, sin un deterioro de la calidad de software, manteniendo los estándares de optimización, ergonomía, portabilidad, uso intuitivo, fortalece y ofrece comodidades al mundo empresarial. Logrando potencializar ventas mediante un control de procesos y su respectivo análisis en cualquier parte del mundo con disponibilidad de conexión a internet.

Con todos los antecedentes enunciados, se considera pertinente la implementación de tecnologías como FullStack MERN, la industria de la programación está en constante progreso, las herramientas que alguna vez fueron la solución ahora se están volviendo el problema, el costo de un aplicativo móvil estándar puede ir desde los $1500 hasta los $15000 por sistema operativo, React Native, baja este presupuesto considerablemente por la oportunidad de realizar una sola codificación, a su agregado MERN nos facilita rapidez, eficiencia, seguridad y refactorización.

El surgimiento de nuevas tecnologías es directamente proporcional a nuevos errores. En la era de los ordenadores analógicos, Grace Hopper, informática, de los años 40 encontró el primer bug de la historia, literalmente era un insecto que impedía el funcionamiento de la gran computadora.

Los aplicativos tanto móvil como web no están exentos de estos problemas. La programación ha tenido muchos cambios con el paso del tiempo, algunos lenguajes han dejado de existir o se han vuelto obsoletos por dejar de dar soporte. Hace varios años la programación web era muy complicada y difícil de implementar, dentro del código se debía especificar la resolución de la pantalla para que la página se adapte, sin embargo, no existía una sola resolución de pantalla. Con el tiempo se fue solucionando esta problemática hasta la llegada de los smathphones y tabletas.

Actualmente, el mundo está viviendo un encierro provocado por la pandemia COVID19, muchos negocios han dejado de funcionar, la tasa de empleos ha bajado, al igual que la economía mundial. La bolsa de valores ha estado apuntando a las aplicaciones móviles, tiendas en línea como Amazon, entre otras. Las empresas distan

en la adquisición de un software, llegando a generar en su mayoría una muerte paulatina, TultiPrint de manera general en micro y medianas empresas.

La investigación de este trabajo se enfoca en la realización de software multiplataforma a bajo costo sin perder los estándares de calidad, lo cual se cumplirá usando la modalidad de programación Full Stack.

La problemática relacionada a la resolución de pantalla de los distintos dispositivos fue solucionada con la llegada de la programación nativa o multiplataforma con renderización automática usando diversos Frameworks.

El sentido de reutilización es uno de los métodos más usados para la optimización de código y es lo que nos aporta las nuevas tecnologías. El Full Stack que se usará en este trabajo es el MERN (Mongo, Express, React y Node). Este tipo de programación fragmenta el código en Back End, donde se aplicará toda la lógica de programación, el modelo de negocio, validaciones, reutilización de código, creación y consumo de servicio REST (Transferencia de Estado Representacional), se aplica el termino WORA.

El Back End es la base para cualquier tipo de dispositivo, creando capas estratégicas para una fácil comunicación. La otra parte es el Front End, que es la parte visual que el cliente aprecia.

En el Front End, existen distintos lenguajes basados en JavaScript que permite que un software sea multiplataforma, este es el caso de React, que crea páginas web tanto informativas e interactivas que no se limitan con la resolución de la pantalla o React Native, que es enfocada a los dispositivos móviles. Gracias al proyecto Expo, la codificación de la parte visual del aplicativo solo se la realizará una vez, exportando el aplicativo tanto para IOS o Android. El uso de Hooks evita la renderización constante de las aplicaciones, y el famoso callback hell.

Actualmente existen lenguajes de programación multiplataforma, una de ellas de la empresa de Microsoft es Xamarin, de Oracle es Java entre otras, sin embargo, está en vigencia el problema de refactorización de código. A pesar de que Xamarin ofrece varias soluciones de multiplataforma, se debe tanto adaptar el código para IOS, Android o Windows, dejando a un lado la navegación Web.

La acumulación de código en JavaScript es muy frecuente, en especial cuando el programador crea varias funciones una tras otra, en este entorno este lenguaje a simple vista puede ser más un problema que una solución, especialmente a aquellos desarrolladores que cambian de una programación orientada a objetos a Full Stack. Esta intercalación de funciones es conocida como callback hell, pues su resultado satura la lectura de data. La solución más viable fue la implementación de funciones asíncronas o el uso de Then. Ambos casos facilitan el flujo continuo de comunicación.

Hoy en día el mundo desea velocidad, eficiencia, calidad, funcionalidad, tener un buen ROI (Retorno de la Inversión) con respecto al crecimiento de empresas, las bases de datos relacionales afectan la velocidad de lectura de datos, por las múltiples llamadas al servidor. La mala distribución de código y servidores es una de las principales fuentes de bugs. Es bien conocido que los programadores necesitan del apoyo de un diseñador para la parte visual, lo que determina más egreso de una entidad financiera [10].

Para agilizar la ejecución de peticiones al servidor es necesario usar la estructura Serverless, delegando las operaciones a distintos servidores, el problema de aumento de personal que controle cada servidor se soluciona con FAAS (Function as a Service). La combinación entre ejecución en frío y caliente, ahorrará el uso de la memoria de nuestro servidor.

MongoDB al ser una base de datos no relacional, permite búsquedas más rápidas, posee funciones de CRUD (Crear, Leer, Actualizar, Eliminar) optimas, al combinar los conocimientos adquiridos en la carrera con la documentación que ofrece la página oficial de MongoDB, se crearían funciones complejas.

La seguridad es uno de los principios fundamentales para la adquisición de un sistema [5]. A pesar de la inexistencia de Inject SQL, MongoDB no está extenso a vulnerabilidades de tipo inject JSON, las cuales se han ido solucionando por medios de autenticación. La tokenización y encriptación de datos cuidan el acceso por rutas a la base de datos, creando barreras ante ataques.

1.2.2. Fundamentación Teórica

Aplicativos Web

La WWW(World Wide Web), es el principal pilar en el mundo del internet, teniendo como como origen acceder sin complicaciones a distinta información. Tim Berners-Lee diseñó este tipo de sistema para la empresa que trabajaba a finales de los años 80 usando como base el hipertexto para interconectar los documentos, fue así como nació el primer navegador web [11].

La Web en su traducción al español red o malla hace referencia al internet, mientras que un aplicativo web es la agrupación de páginas digitales, facilitando el almacenamiento de la data, con un conjunto de componentes como textos, audio, imágenes, videos, mapas, calendarios entre otros, donde un usuario puede ver por medio de un navegador web.

Estos aplicativos son codificados en lenguajes dedicados al mundo web, teniendo como principal objetivo hacer de una página web dinámica y exploratoria, por medio de enlaces, hipervinculizando distintos documentos, es decir, una fácil navegación entre páginas web.

Peticiones y Respuestas

Parte fundamental en la web se basa en dos aspectos, el protocolo HTTP(Hypertext Transer Protocol) cual función principal es la implementación de sistemas de comunicación de manera asequible, en otras palabras, este protocolo permite la emisión de distintos ficheros sin ninguna complicación y no se limita al tipo de servidor, dado que un servidor básico puede atender miles de peticiones. El otro fundamento es el lenguaje HTML, quien se encarga de los enlaces o interconectar páginas web con gran eficiencia y efectividad, facilitando su uso.

El protocolo HTTP se maneja por sus conexiones basadas en TCP (Transport control protocol) creando un canal de comunicaciones entre el cliente y el servidor. Su variante HTTPS (Hypertext Transer Protocol Secure) utiliza un protocolo de seguridad SSL (Secure Socket Laker) cifrando y autenticando el flujo de data, es decir, controla que los datos enviados y receptados fluyan por un canal seguro [11].

El proceso de HTTP inicia en el momento que el cliente se interconecta con el servidor vía TCP, se envía un comando de petición y en esa conexión se recepta la respuesta

con la información postulada. Los comandos más utilizados son GET, POST, PUT, DELETE, los dos primeros son peticiones de recurso con diferencia que en POST se pasa parámetros, PUT crea o a su vez envía los recursos y DELETE su respectiva eliminación.

El esquema global del uso de este protocolo comienza con la línea de petición, donde abarca el método, la identificación del recurso y la versión del protocolo, seguido de la cabecera de petición, con información extra para el fácil procesamiento de las peticiones en el servidor, por ejemplo, el Host, User-Agent, Accept, entre otros. Por último, los parámetros de petición en el caso de POST, PUT, DELETE, y similares. Al igual que las peticiones, las respuestas poseen el mismo mecanismo, a diferencia que lanza los resultados solicitados en la esquematización del protocolo.

Servidor Web

El servidor web es el canal que recepta y emite el sin fin de peticiones en los navegadores web, es decir, lo que se solicita por medio del protocolo HTTP o su variante HTTPS el servidor web se lo da. Un ejemplo estándar de esta comunicación comienza con la espera de peticiones normalmente en el puerto 80, al recibir lo solicitado, el servidor busca el recurso, lo envía por la misma conexión y regresa a la recepción de peticiones, formando un bucle infinito.

En sus funciones básicas un servidor web debe tener el servicio de ficheros estáticos, en otras palabras, poder acceder a los archivos que se encuentran en una parte específica del disco, teniendo como requisito la especificación del disco para su uso. En su mayoría, los servidores web vienen con la capacidad de implementar directorios seguros, o especificar el índice del directorio, entre algunos servicios más.

La seguridad es uno de los campos más importantes dentro de un servidor web, dado que la prioridad es la información y su correcto manejo, se pensó en varias formas de cuidar dicha información. .htacces es uno de los métodos más sencillos, pues este sistema de seguridad agrega un fichero de este tipo con el fin que usuarios, archivos, máquinas puedan tener acceso a los directorios y sus derivados. Otra forma es declarando reglas en los directorios y ficheros, de igual manera limitando el acceso a las diferentes entidades o clientes.

Sin embargo, el método más usado y viable es la autenticación, validando la entrada de un cliente por medio de un usuario y contraseña. A pesar de las alternativas para poder hackear estas cuentas, la seguridad ha incrementado rigurosamente, entre complicar la forma de la contraseña con la combinación de mayúsculas, minúsculas, números y caracteres hasta la forma de encriptar dicha contraseña.

Plataforma como servicio (PaaS)

Una plataforma como servicio (PaaS) es una plataforma en la nube en la que se puede desarrollar, ejecutar y gestionar casi sin tener que realizar configuraciones ya que la plataforma proporciona la infraestructura de la aplicación, además de proporcionar almacenamiento, redes, sistemas operativos, middleware en tiempo de ejecución, base de datos y otros servicios. Kubernet es el administrador de clúster de contenedores más utilizados y puede usarse como base para poder desarrollar una plataforma como servicio (PaaS) [1].

Es una plataforma para el desarrollo y alojamiento de las aplicaciones de una empresa. La plataforma es un servicio en la nube que se combina con la infraestructura (IaaS, Infraestructura como Servicio), un software para el desarrollo de aplicaciones y seguridad. Esto hace que sea más fácil y barato desarrollar aplicaciones propias para las empresas, así coma hospedar aplicaciones ya personalizadas. PaaS incluye infraestructura: servidores, almacenamiento y redes" [2].

Esta plataforma proporciona un entorno de desarrollo ágil que facilita al usuario desarrollar aplicaciones rápidamente y adoptarlas al instante. Se elimina la espera para la implementación de hardware y software adecuados para la aplicación. Los usuarios pueden usar la plataforma para crear aplicaciones utilizando idiomas, bibliotecas, servicios o herramientas compatibles con el proveedor [3].

Infraestructura como servicio (IaaS)

Una infraestructura como servicio permite alquilar la infraestructura de TI (servidores o máquinas virtuales) de un proveedor de la nube en forma de pago por el uso. El consumidor necesita recursos informáticos como servidores, redes y almacenamiento, los cuales se requieren para implementar y ejecutar aplicaciones desarrolladas por un especialista [4].

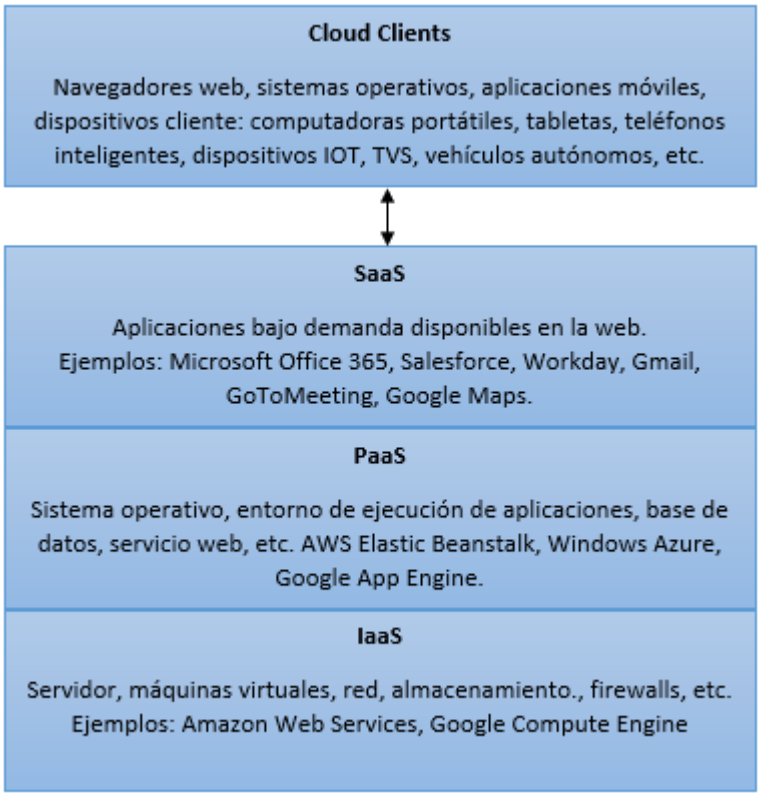


Figura 1: Modelos de servicios de computación en la nube

Fuente: [12]

Estándares de calidad de software

La calidad de software se ha convertido una base importante dentro de las empresas, el control del diseño, codificación, fase de pruebas y su análisis otorgan mayor viabilidad, es decir, con estos estándares, un software es confiable, con un gran índice de manutención, escalable, entre otros aspectos más, elevando la productividad, eficacia y eficiencia.

Ciclo de Vida del Software: La secuencia definida de las distintas etapas del software y su organización desde la concepción de la idea hasta la obsolescencia del software es conocido como el ciclo de vida del software.

Así como el ciclo de vida del ser vivo es nacer, crecer, reproducirse y morir, el software tiene de igual manera un esquema estandarizado por la iso/iec 12207.

- Proceso de acuerdo, donde se aclara la adquisición y suministro del software, en la primera se busca satisfacer las necesidades del cliente, identificando y especificando lo que el usuario final desea, aceptando el producto o servicio. El proceso de suministros se encarga de la compra de los servicios en base a los requisitos definidos con anterioridad
- Procesos Organizacionales, donde se gestiona el modelo de ciclo de vida, es decir las políticas de proceso, definición, objetivos, una mejora continua, a su vez gestionar la infraestructura, la cartera de proyectos, los recursos humanos y la calidad
- Proceso del proyecto, ciclo en el cual se planifica, estableciendo los requisitos, identificando el alcance, tareas, salidas, planes y recursos. A demás de poder evaluar y controlar el proyecto, gestionando decisiones, riesgos, información, culminando con la medición de la recolección de datos.

- Procesos Técnicos, parte final del ciclo, donde se define los requisitos de las partes interesadas, analizando los requisitos de sistema, la implementación, integración del sistema. Se comprueba todos los requisitos dados por el usuario, con su posterior instalación, permitiendo al usuario el uso de este. Con el pasar del tiempo es necesario el mantenimiento del software, sin embargo, es importante recalcar el proceso de retirada del mismo.

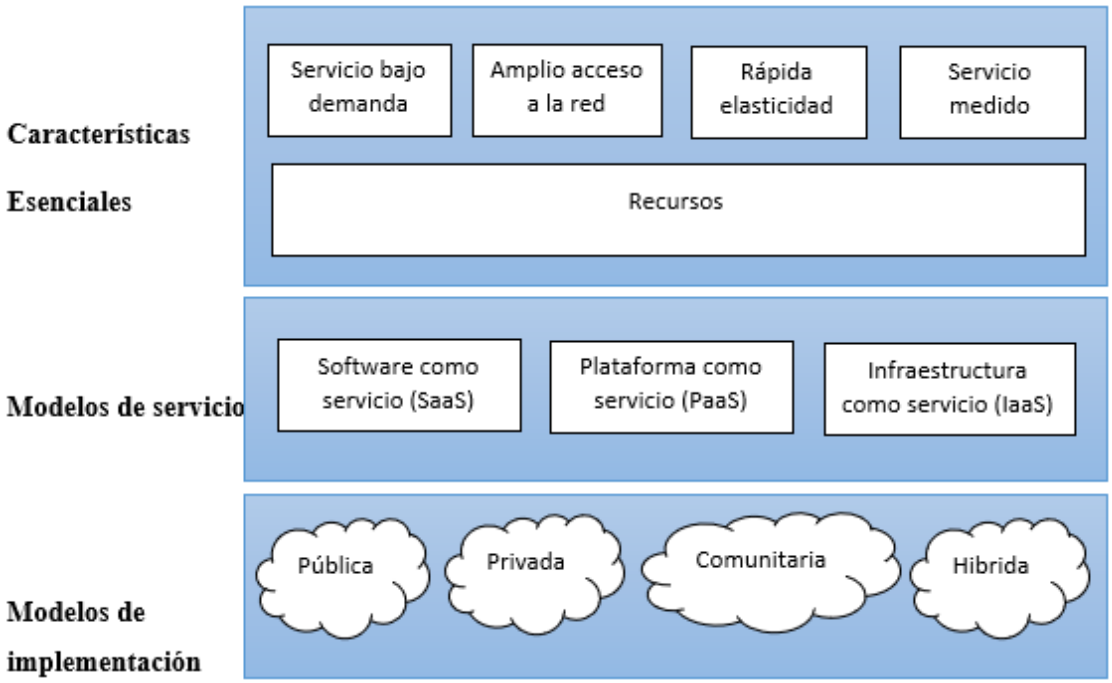


Figura 2: Características de la nube, modelos de servicio e implementación

Fuente: [14]

Calidad externa e interna

La ISO/IEC 9126 nos permite evaluar los productos de software mediante características internas y externas, estandarizando la usabilidad y calidad de un software. Estas características son:

- Funcionalidad, donde se evalúa la aplicabilidad, precisión, interoperabilidad, seguridad y la conformidad de la funcionalidad.
- Fiabilidad, capacidad del software para mantener un balance del funcionamiento, basado en la madurez, tolerancia a fallos, recuperable y la capacidad del software para adaptarse a las normas, convenciones o regulaciones relacionadas con la fiabilidad.
- Usabilidad, estándar de calidad, donde el usuario final aprecie los siguientes parámetros: el software debe ser entendible, operable, con una buena interfaz gráfica, fácil de aprender para una conformidad en su uso.
- Eficiencia, capacidad del producto en el comportamiento del tiempo a respuestas y procesos, optimización de las cantidades y tipos de recursos a usarse.
- Factibilidad de mantenimiento, el código de un software debe tener respuestas rápidas ante los cambios o bugs. Cuando se presente alguna vulnerabilidad o el sistema este creciendo, previo a un análisis, se debe alterar el código sin perjudicar los procesos adicionales. Gracias al testeo, es posible mantener un código sin perjudicar a la organización.
- Portabilidad, capacidad del software para adaptarse a distintos entornos, fácil instalación independientemente de los recursos que lo rodean recordando que en un futuro pueda ser remplazado sin la perdida de datos.

Refactorización

La refactorización se basa en el cambio estructural del código de un software sin alterar el funcionamiento externo, en otras palabras, limpiar el código del sistema para prevenir futuros bugs, mejorando el diseño de la codificación.

Por el código se puede conocer al programador, la programación es clara, si un código tiene comentarios, declaración de variables y funciones con nombres entendible, está en el primer paso de un condigo refactorizado, en caso contrario es buena la comparación de código, para comprender que está mal.

GRASP (General Responsibility Assgnment Software Patterns), se encarga de distintas pautas dando responsabilidades a objetos y clases, por medio de controladores, alta cohesión, in direcciones, entre otros. GRASP de la mano con SoC (Separation of Concerns) que significa separación de preocupaciones, el cual consiste en separar un sistema en secciones distintas, analiza si una clase tiene varias responsabilidades, para su posterior separación en tantas clases como sean necesarias.

Gran similitud se presenta en la función de Soc/KISS (Keep It Simple, Stupid), dividiendo tareas independientes de un método muy extenso. Un ejemplo simple de este concepto es al tener tareas de operaciones básicas, donde se suma, se multiplica, se resta se divide en un solo método, a pesar de un funcionamiento correcto de la función, la refactorización me obliga a dividir estas tareas en métodos independientes. Lo importante en estos casos es poder analizar las tareas que puede representar una acción.

Esta separación de clases o funciones, van de la mano con una buena estructura y organización, separando por carpetas o capas el código.

Reutilización

En términos ambientales la reutilización es parte de las 3R (Reducir, Reutilizar y Reciclar) para el manejo de residuos sólidos para evitar una contaminación masiva, donde se afirma si un residuo solido puede volverse a usar o darle un uso. Dentro de la codificación de un software se usa este mismo principio.

Full Stack Developer

En la estructura de datos es conocido el término stack o pila, un full stack es una persona que trabaja en pila completa, desde el Backend hasta el Frontend, dando vida al concepto lo último que entra es lo primero que sale.

Existen distintas herramientas u opciones para ser un full stack developer, como MEAN (Mongo, Express, Angular, Node) o MERN (Mongo, Express, React, Node). Dentro de los lenguajes que se usa para Back End, está Java, Node, Mongo, C# entro otros, mientras que su contraparte complementaria el frontend se respalda en Bootstrap, Angular, Backbone, React, entre otras.

MongoDB es una base de datos no relación o No SQL, ofreciendo gran escalabilidad, flexibilidad y un modelo de consultas avanzado.

Su porcentaje de aprendizaje es muy alto, y aporta a los desarrolladores las funcionalidades necesarias para cumplir con los requisitos más complejos en el manejo de la Data. Existe una gran comunidad que aporta conocimiento y respuestas ante inquietudes que surgen en el camino.

Al igual que JSON (Notación de Objetos de JavaScript), Mongo maneja el almacenamiento de datos flexibles, variando los campos a documentos y la estructura de la data puede variar.

Express se basa en IAAS, dado que es una infraestructura vinculada con aplicativos webs Node, otorgando un grupo fuerte de características para los aplicativos móviles y web. Gracias a la comunidad de desarrolladores y su helpdesk constante, la creación de servicios Web son sencillos e inmediatos, siendo clave para la comunicación con el servidor.

React es una biblioteca perteneciente a Javascript para la creación de interfaces de usuario. Desde la opción del diseño de una página web informativa, hasta la elaboración compleja de un aplicativo móvil, React es la herramienta correcta para el desarrollo multiplataforma.

Node.js, es un entorno de ejecución JavaScript multiplataforma, se puede realizar varios proyectos, entre una consola, un servidor web, un aplicativo de escritorio y mucho más.

Node se ejecuta en un solo proceso, no es necesario la creación de nuevos hilos por cada solicitud, mediante un conjunto de E/S (Entradas y Salidas) asíncronas, evitando el bloqueo del código.

El momento en el que Node realiza una operación E/S que puede ser el acceso a una base de datos, a un sistema de archivos o la lectura desde la red, no bloquea el hilo lo que desperdicia los ciclos de CPU, sino que reanudará las funcionales al memento que regrese la respuesta.

1.3. Objetivos

1.3.1 Objetivo General

Desarrollar un sistema de gestión y control de procesos de bajo costo para microempresas que ofrecen servicios de publicidad y corte láser

1.3.2. Objetivos Específicos

- Analizas los distintos procesos de una microempresa de publicidad y corte láser.
- Diseñar la propuesta de solución, aplicando la refactorización y reutilización.
- Automatizar la gestión y control de procesos en microempresas orientadas a la publicidad y corte láser.

CAPÍTULO II:

METODOLOGÍA

2.1 Materiales

Para la presenta investigación es necesario el uso de libros, artículos, revistas, por el hecho de ser una investigación bibliográfica, como agregado se aplicará los conocimientos adquiridos con el pasar de los años dentro de la carrera.

La herramienta fundamental entre las técnicas de investigación a usar es la observación, evitando datos dudosos o incongruentes al realizar una encuesta o una entrevista. La observación permite analizar los problemas de manera interna y externa, sistematizando la resolución de datos.

En el caso investigativo, se observó directamente el funcionamiento de una empresa de diseño y corte láser, con su respectivo control de procesos en el área de producción.

Tabla 1: Lista de Cotejo sobre la secuencia de operaciones para la toma de pedidos

Elaborado por: El investigador

Lista de cotejo sobre secuencia de operaciones a la toma de pedidos El proceso de observación a los procesos a la toma de pedidos se ha puesto en desarrollo para el conocimiento de la secuencia inicial para la elaboración de un pedido.		
Proceso	Si	No
Toma de requerimientos del cliente		
Cancelación monetaria parcial o total de la Orden		
Toma de datos informativos del cliente		
Entrega de un recibo de la orden o pedido		

Tabla 2: Escala de apreciación numérica acerca de la secuencia de operaciones de diseño

Elaborado por: El investigador

Escala de apreciación numérica sobre la elaboración del diseño			
4. Siempre	3. Generalmente	2. Ocasionalmente	1. Nunca

Proceso	4	3	2	1
Elaboración del diseño				
Aprobación de diseño				
Cambio de diseño				
Entrega de diseño				

En la Tabla 2 se aprecia una escala de apreciación numérica sobre el proceso de un diseño a base de los requerimientos del cliente, donde 1 es nunca, 2 ocasionalmente, 3 generalmente y 4 siempre, relacionado al proceso descrito en la tabla.

Tabla 3: Lista de Cotejo sobre la secuencia de operaciones para el corte o impresión

Elaborado por: El investigador

Lista de cotejo sobre secuencia de operaciones de impresión El proceso de observación a los procesos de impresión se ha puesto en desarrollo para el conocimiento del proceso de producción final del producto.		
Proceso	Si	No
Recepción de archivos compatibles con la maquinaria (Plotter de corte, Plotter de impresión, cortadora CNC, impresoras, entre otras)		
Impresión de archivos en las respectivas maquinarias		
Armado del producto		
Añadir detalle al producto		
Empaquetar		

Tabla 4: Escala de apreciación numérica acerca de la secuencia final del producto con el Cliente

Elaborado por: El investigador

Escala de apreciación numérica sobre la entrega del Producto 4. Siempre 3. Generalmente 2. Ocasionalmente 1. Nunca				
Proceso	4	3	2	1
Notificar al cliente del producto terminado				

Inspeccionar requerimientos y calidad del producto				
Pago final o total de la Orden				
Entrega del producto				

Tabla 5 : Escala de apreciación gráfica sobre los procesos de una orden.

Elaborado por: El investigador

Escala de apreciación gráfica en el proceso de una orden				
La escala de apreciación tiene como fin representar aspectos efectivos en el momento de elaborar un producto.				
1. Proceso de toma de pedidos Manual				Automático
2. Cambio de Requerimientos Frecuente				Pocos
1. Conocer el estado del pedido Desconocido				Tiempo Real

2.2. Métodos

2.2.1. Modalidad de la Investigación

En el presente proyecto se aplicó las siguientes modalidades**:**

Investigación bibliográfica-documental

Este tipo de investigación tendrá la funcionalidad de obtener referencias necesarias para el correcto proceso obteniendo los resultados pertinentes.

Investigación de campo

La naturalidad de este trabajo se basa en la manipulación y control de variables dada la problemática descrita en el inicio de este documento. Se usarán parte del método científico, como la observación, experimentación y comprobación, es decir, provocar una determinada situación.

2.2.3 Recolección de información

La recolección de datos o información tiene como fundamento la observación directa, la cual brindó información real y concisa sobre el estado actual de los procesos por el que pasa un producto antes de entregarse.

La observación se llevó a cabo en la empresa El Taller Fábrica de Ideas, Banovo, Virtual Design y Magia y Láser. En la primera microempresa, el investigador trabajo directamente en el área de administración, validando todo tipo de información y adquiriendo conocimiento total de los procesos, generando una base para un sistema general para microempresas de diseño y corte láser.

2.2.3.2. Resultados de la observación hecha a las distintas microempresas de diseño y corte láser.

Objetivo: Analizar los distintos procesos de una microempresa de publicidad y corte láser.

Dado los resultados en el Anexo 1, 2, 3, 4 y 5 sobre los procesos de observación aplicados a las 4 microempresas, se obtuvieron las siguientes gráficas.

1. Secuencia de operaciones a la toma del pedido.

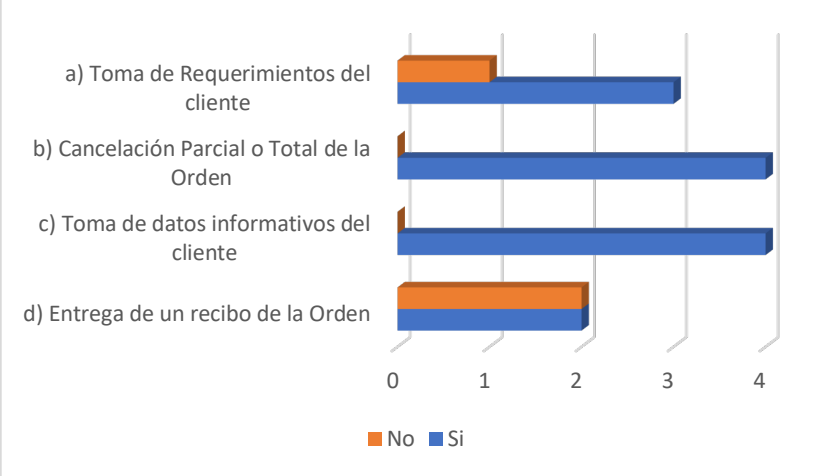


Figura 3: Lista de Cotejo, toma de pedidos

Elaborado por: El investigador

Análisis e interpretación: En el literal a) el 75% de las microempresas apuntan los requerimientos del cliente, en el literal b) el 100% recepta un anticipo o el pago total de la orden, así como en el literal c) el 100% apunta datos informativos del cliente, cómo números de teléfono para poderse contactar, sin embargo, solo el 50% en el

literal d) entregan un recibo a los clientes con respecto a su orden. Todos estos datos dan un punto de partida a los procesos que se manejan.

2. Secuencia de operaciones en la elaboración del diseño.

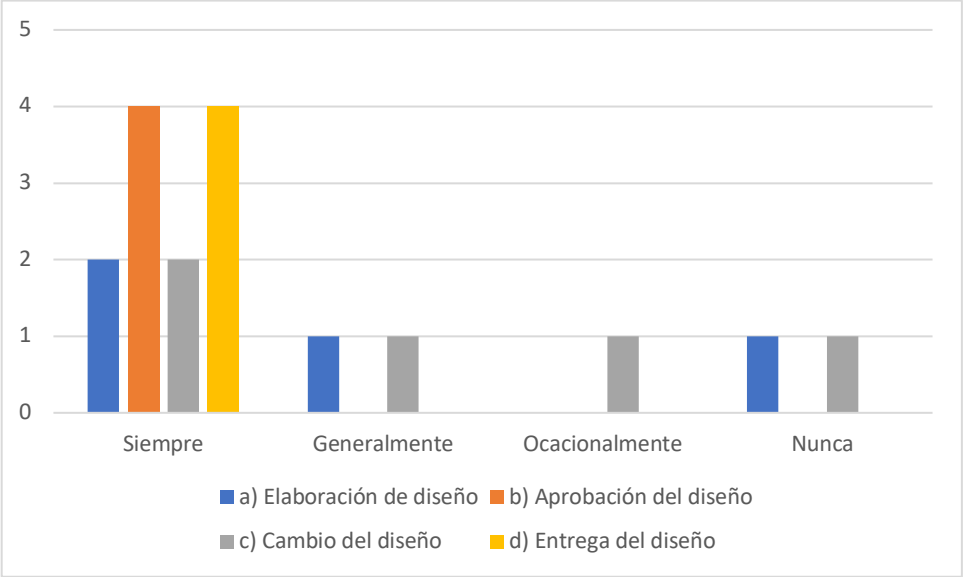


Figura 4: Escala numérica, Elaboración de diseño

Elaborado por: El investigador

Análisis e interpretación: El 75% de las organizaciones elaboran sus diseños, sin embargo, solo el 66.66% lo hace siempre. El otro 25% terceriza los diseños, razón por la cual el 100% manda a aprobación de los diseños. El 50% realiza siempre cambios en sus diseños, mientras que el resto se divide equitativamente en generalmente, ocasionalmente o nunca, el ultimo caso se da a motivo de la tercerización. El 100% entrega sus diseños, sean a una empresa externa, al cliente, o al encargado de la maquinaria a utilizar.

3. Secuencia de operaciones de Corte o impresión.

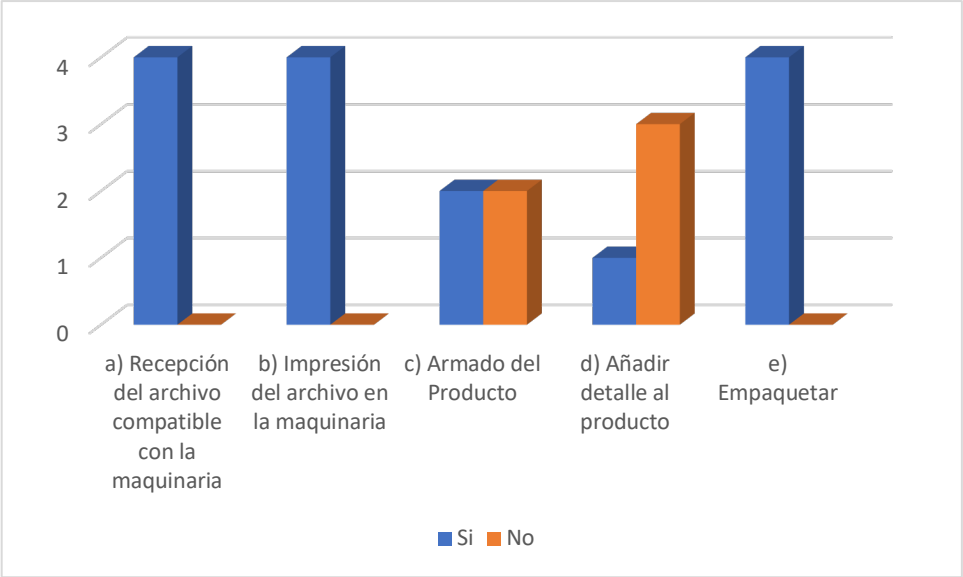


Figura 5: Lista de cotejo, corte y/o impresión

Elaborado por: El investigador

Análisis de interpretación: el 100% recepta el archivo, sea para impresión o corte, imprime el archivo, entendiendo que el proceso de imprimir representa el uso de la maquinaria de cada empresa, así mismo el 100% empaqueta el producto final, sin embargo, solo el 50% arma el producto, dado que son las empresas de corte y el 25% realiza un toque final a la orden, siendo un plus de la empresa.

4. Secuencia de operaciones final de entrega.

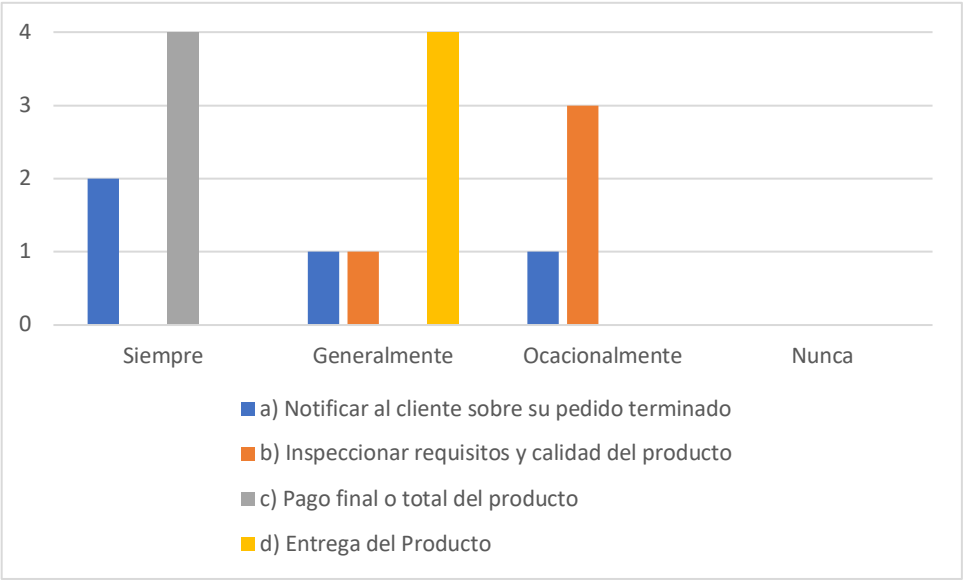


Figura 6: Escala numérica, Entrega de Producto

Elaborado por: El investigador

Análisis e interpretación: El 50% notifica al cliente que su pedido ya esta listo para entregar, mientras que el 25% lo hace general u ocasionalmente, a su vez el 75% ocasionalmente hacen un control de calidad. El 100% cobra el total o saldo restante y generalmente hacen la entrega del producto, dado que a veces no hay satisfacción del cliente en su producto o se pasa el plazo de espera de garantía del producto.

5. Escala gráfica de proceso de producción.

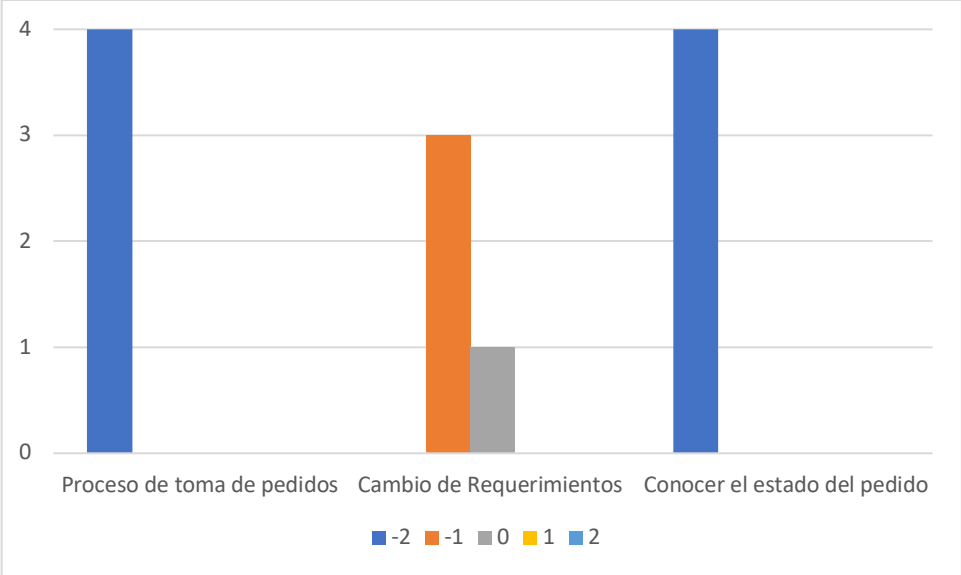


Figura 7: Escala gráfica, procesos de producción

Elaborado por: El investigador

Análisis e interpretación: Las 4 empresas manejan un estado manual para la recepción de pedidos, dando énfasis en la importancia de aplicar un sistema para este tipo de empresas, el 75% de las empresas tienen cambios frecuentes, sin embargo, el 100% de las organizaciones no conoce a tiempo real en que estado se encuentra la orden de pedido.

2.2.3 Procesamiento y análisis de datos

En base a la observación y su respectivo proceso de análisis, los datos recopilados ayudaran a tener un amplio espectro de los procesos manejados dentro de una empresa de diseño y corte láser, dichos resultados serán respaldados con proyectos investigativos relacionados a micrompresas y sus procesos, lo que ayudara a la resolución y necesidad de un sistema multiplataforma para el control y gestión de procesos de elaboración de productos.

En el Taller Fábrica de Ideas, donde se realizo una observación participativa, se manejan bajo procesos y subprocesos para mejorar la calidad del producto final, sin embargo, al hacer todo netamente manual, algunos procesos de producción e incluso lo ingresos al final del día resultan truncados o inconsistentes. La importancia de un software de control de procesos es importante.

Es claro que un sistema multiplataforma puede resultar muy costoso, en especial para organizaciones que recién están empezando o sus ingresos no representan esta inversión. La idea de utilizar Full Stack Mern es restar el tiempo, disminuyendo el esfuerzo. En Cocomo2 los resultados de precios son directamente proporcional al esfuerzo.

Desarrollo del Proyecto

- Análisis de lineamientos generales de una microempresa de corte láser.
- Análisis del ciclo de producción de una microempresa de corte láser y diseño.
- Diseño de la Arquitectura de la aplicación
- Diseño de la base de datos del aplicativo.
- Diseño del backend del aplicativo.
- Diseño de Hooks para el FrondEnd.
- Diseño arquitectural FrondEnd para el sistema web y móvil.
- Desarrollo del esquema BackEnd y FrondEnd.
- Prueba en los distintos dispositivos (Web, Android, iOS) de los componentes planteados.

CAPÍTULO III:

RESULTADOS Y DISCUSIÓN

3.1. Análisis y discusión de los resultados

Análisis de lineamientos generales de una microempresa de corte láser y diseño.

Las microempresas orientadas al diseño y corte láser tienen como fundamento la realización de productos en base a requerimientos del consumidor final, ofreciendo diferentes servicios de visualización física o virtual de un producto terminado, el organigrama básico de dichos establecimientos es de un gerente general, un gerente de marketing y un diseñador, dependiendo de las necesidades se puede extender a ventas, ingeniería o innovación.

La obtención de materia prima y el equipo correcto es fundamental dentro de este tipo de empresas, como un plotter de impresión, un plotter de corte, una cortadora láser o CNC, computadoras, entre otros. En el caso de que la microempresa no disponga de dichos recursos acuden a la tercerización.

Análisis del ciclo de producción de una microempresa de corte láser y diseño.

Con el conjunto de datos obtenidos se puede concluir que la elaboración de un software de control de elaboración de productos debe estar orientado a cada microempresa, es decir, el módulo procesos dentro del sistema, no se puede realizar de manera general. Es importante implementar un CRUD de procesos dentro del modelado de la base de datos, para que el usuario pueda administrar la creación, edición y eliminación de los procesos que maneja su microempresa.

Estudios han representado de manera general los procesos bases en la producción de diseños y corte láser, dichos procedimientos se han fundamentado en la investigación, teniendo como fin un control de calidad [15]. Un resumen de estos procesos se puede observar en la Tabla 6, es importante enfatizar que, estos procesos no serán parte del sistema multiplataforma.

Tabla 6: Procesos base de una microempresa de corte láser y diseño.

Elaborado por: El investigador

Procesos base de una microempresa de corte láser y diseño		
Proceso columna	Forma de uso	Estado
Información dada por el Cliente	Recepción de los requerimientos del cliente, escribiendo a detalle cada aspecto del producto final, evitando problemas en un futuro y creando satisfacción y lealtad al cliente	1
Elaboración del Diseño	Con los datos específicos del producto final, el diseñador se encarga de realizar un bosquejo o simulación del trabajo terminado, cumpliendo a detalle los requerimientos del cliente	2
Aprobación del Diseño	Se envía el bosquejo al cliente para su respectiva aprobación o realizar cambios conforme al acuerdo llegado	3
Producción del Diseño	Con la aprobación del bosquejo o diseño, se realiza la producción del trabajo final, este puede ser elaborado en las distintas maquinarias de la empresa o la entrega del archivo digital.	4
Entrega de producto Final	Ya terminado el producto, se hace la entrega al cliente y finaliza el proceso de aquel servicio.	5

Análisis de la arquitectura a utilizar.

Existen distintas arquitecturas de programación como se puede apreciar en la Tabla 7, cada una tiene distintas características, sin embargo, la arquitectura Full Stack es un conjunto de las mejores propiedades de distintas arquitecturas. Desde el inicio de su funcionamiento Modelo, Vista, Rutas (MVR), hasta la disminución de tiempo de desarrollo.

A pesar de tener grandes características, Full Stack puede ser desarrollado en distintos lenguajes en un contexto de FrontEnd, y cada uno aporta o limita la explotación de dicha arquitectura. La Tabla 8 desglosa las ventajas y desventajas de frameworks Front, en el caso de este trabajo investigativo se opto por React, por su facilidad de comunicación entre componentes, seguridad y su valiosa herramienta Expo para desarrollo móvil, cumpliendo la refactorización, fragmentación y reutilización de código.

El framework React se optimiza con el uso de Hooks (Ganchos), permitiendo la globalización de variables de forma segura con useContext, interacción de rutas y procesos, con useMemo y useReducer, cambios de renderización con useEffect, asignación y uso de variables con useState, entre algunos más. El uso correcto de estas librerías da como resultado un código refactorizado, fragmentado y reutilizado, en otras palabras, buenas prácticas de programación.

3.1.1. Tabla comparativa de distintas arquitecturas de programación.

Tabla 7: Tabla comparativa de distintas arquitecturas de programación

Elaborado por: El investigador

Parámetros	MicroServicios	Diseño Estructurado	Arquitectura Monolítica	N-Capas	MVC	SOA	Full Stack
Funcionamiento	Uso de metodología inductiva, donde se manejan servicios para la construcción de una aplicación.	Se manejan a partir de diagramas de flujo de datos (DFD), describiendo el constante movimiento de información.	Agrupación de todas las funciones y servicios en un solo proyecto.	Una arquitectura cliente-servidor más centralizada, donde se divide la lógica de negocios y la lógica de diseño.	El modelo, vista, controlador, separa el código en estas 3 responsabilidades, manejándose en distintas capas concretas.	La arquitectura orientada a servicios (SOA) une las metas del modelo de negocio, con el sistema de software.	En un conjunto de arquitecturas, entre microservicios, MVC y SOA
Escalabilidad	Al trabajar cada servicio con un proyecto independiente, para ser parte de algo más grande, posee un gran porcentaje de escalabilidad.	Al basarse inicialmente en el negocio clave y en diseños predefinidos, sus planteamientos no son tan escalables,	Al estar todo el código en un solo proyecto la arquitectura monolítica no tiene crecimiento alguno.	Su comunicación da accesibilidad completa a que el software crezca, sin embargo, hay que tomar en	El manejo de responsabilidades de cada capa da el reflejo de microservicios, e incluso, los microservicios nacen del MVC	Su arquitectura pensando en el modelo de negocios ayuda a garantizar un crecimiento aplicativo.	Al ser el conjunto de arquitecturas escalables, Full Stack ofrece el crecimiento continuo fácil y rápido sin detener el

		permite un crecimiento limitado de la aplicación.		cuenta que todo este relacionado para su funcionamiento.			proceso de la aplicación.
Reutilización	SI	NO	NO	SI	SI	SI	SI
Forma de conexión	Cada servicio se maneja independientemente alojados en un servidor, y para acceder a los micro servicios se manejan APIS.	Comunicación directa	Comunicación en un mismo entorno.	Cada capa se comunica entre si	Las vistas se comunican con los controladores, mientras ellos se conectan con el modelo y este último con la base de datos.	Comunicación Distribuida	Dentro del Backend se manejan los micro servicios con modelos y controladores, que se conectan con la base de datos. Al devolver APIS de forma distribuida son consumidas por el Frontend.
Seguridad	Manejo de Tokens por micro servicio, otorgando gran protección ante	Ataques directo a la base de datos	Gran vulnerabilidad ante ataques.	Si no se manejan bien las capas y se facilitan encriptaciones se puede	Cada capa tiene un distinto nivel de acceso a un cliente final, sin embargo, dependiendo	Su arquitectura ofrece una tranquilidad al cliente	Además de el manejo de tokens, encriptación de datos, el uso de

	ataques cibernéticos.			atacar con facilidad.	del tipo de base de datos, puede ser vulnerable a ataques	final ante ataques.	middlewares añade un adicional de seguridad a este tipo de arquitectura. Esto permite acceso a diferentes tipos de clientes, y usualmente usan base de datos NoSQL, evitando el inyect SQL y ofreciendo seguridad de datos.
Despliegue	Se pueden agregar en contenedores, kubernets, mallas o PAAS	En un solo entorno de programación	Aplicación netamente Nativa	La lógica de negocios se almacena en un servidor mientras el cliente se aloja nativamente.	Funciona en PAAS, IAAS o SAAS	De igual manera se añaden a contenedores y servicios de nube	Multi servidores, facilitando la libración de data y velocidad en lectura y escritura.
Costo de Desarrollo	El costo es por microservicio, sin contar la parte del	Su aproximación de coso va	Es la opción más económica,	Los lenguajes de programación	Igual que en n-capas, sus costos varían	Al ser una arquitectura compleja y	Se puede decir a primeras instancias que

	cliente. Cada microservicio por lo bajo llega a costar $20.	desde el planteamiento del diseño y de sus organigramas, siendo una de las arquitecturas más costosas	sin embargo, si la empresa crece debe adquirir otro sistema representando gastos adicionales.	en su mayoría son con licencia, representando un costo adicional del sistema, sus precios son rentables, pero con respecto a tiempo los costos se elevan demasiado.	por el tiempo y esfuerzo, según COCOMO2, la aplicación a realizar puede variar de $15000 en adelante.	eficiente, sus precios son muy altos, dado que empresas grandes optan por dicha arquitectura.	es una arquitectura muy costosa, la hora de los programadores varían en su funcionalidad, es decir, en micro servicios, manejo de servidores, uso de IAAS o PAAS, personal de QA (control de calidad de código), desarrolladores FRONTEND, asumiendo grandes costos, sin embargo, su tiempo de programación se reduce casi a la mitad, en especial al ser

							sistemas multiplataforma.
Tiempo de Desarrollo	En lo mínimo un micro servicio puede demorar 1 hora de realización y dependiendo del proyecto y su complejidad el tiempo se puede ir incrementando.	Un sistema básico cumple el periodo de un semestre en adelante	Pueden llevar años la implementación de un sistema en esta arquitectura.	De un semestre en adelante	De un semestre en adelante	De un año en adelante	De un mes en adelante, dependiendo la complejidad de la aplicación y al reutilizar micro servicios, ventanas, clases, su tiempo se acorta.

3.1.2. Tabla ventajas y desventajas de algunas arquitecturas Full Stack

Tabla 8: Ventajas y desventajas de algunas arquitecturas Full Stack

Elaborado por: El investigador

	Ventajas	desventajas
React	**Documentación** Al ser una freamwork desarrollado por Facebook, su índice de aprendizaje es favorable para quienes se sumergen a este nuevo lenguaje, su documentación en general es muy completa y está en constante mantenimiento.	**Documentación** El mismo hecho de ser una documentación completa, también es grande y puede ser irritante al programador encontrar la mejor solución.
	React Native Expo Aparte del desarrollo web responsive, el cual se puede utilizar para aplicaciones progresivas, React cuenta con React Native, que junto a la librería expo crea aplicaciones para IOs y Android a partir de un solo código de programación. Es importante resaltar que no son aplicaciones Hibridas o Progresivas son aplicaciones Nativas a precio de un solo código.	**Curva de aprendizaje** La curva de aprendizaje puede llegar a ser empinada si no se tiene conocimientos fuertes sobre las diferentes formas de programación en especial la POO.
	Hooks El uso de Hooks facilita el uso de este lenguaje FrontEnd, manejo de estados, efectos, contextos, entre mucho mas hacen que los sistemas aplicativos funcionen a tiempo real.	
	Componentes Puedes encontrar desde un componente de texto, hasta un componente de realidad aumentada, y cada uno con su respectiva documentación.	
Angular	**Databinding** Es complejo y posee un gran potencial, permitiendo crear minicomponentes prestando facilidad a la capa de presentación a un nivel organizacional.	**Documentación** Falta de contenido sobre componentes o su uso,
	Testeo Karma ayuda a crear test unitarios, de extremo a extremo, creando estabilidad en la aplicación	**Solo Angular** Aunque se programa con typeScript y POO (Programación Orientada a Objetos), no es compatible con otros frameworks.
Vue	**HTML empoderado** Optimización de bloques html al usar diferentes componentes igual a sus predecesores.	**Nuevo** Documentación escasa.
	Integración Su uso puede ir desde páginas simples o únicas hasta complejas estructuras web, conocidos como sitios web.	**Excesiva Flexibilidad** Si no se juegan bien las cartas, su integración a proyectos macros puede terminar en problemas y no existe soluciones pertinentes con respecto a la temática.

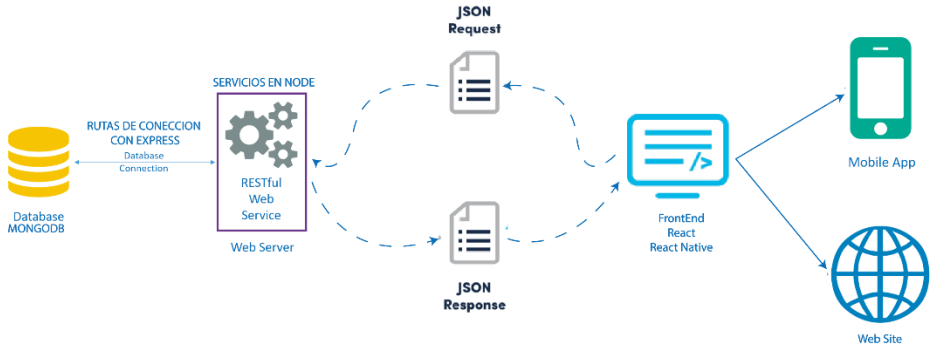


Figura 8: Arquitectura Full Stack MERN

Elaborado por: El investigador

En la figura 8, la arquitectura desarrollada enfatiza los componentes fundamentas a considerar, relacionándolos con las bases arquitectónicas de los aplicativos móviles o web.

Como se menciono con anterioridad, esta arquitectura representa la combinación de los modelamientos actuales y que han representado un impacto en el desarrollo del software. Donde antes todo estaba encapsulado en un solo entorno, ahora se habla de servicios y micro servicios. IAAS era la única forma de trabajo, donde las empresas adquirían servidores, invertían grandes cantidades en hardware, y en personal específico para el manejo del software, sin contar de un sistema intranet e internet. Con la llegada de los nuevos servicios y las bases de datos no relacionables, la inversión se ve menguada comparada con los administradores de bases de datos comerciales como lo es MSSQL (Microsoft Server SQL).

Dentro del backend se maneja una subestructura, siendo esta la fusión entre MVC, SOA y micro servicios. Cada módulo se lo va a realizar en un modelamiento, comunicado con su respectivo controlador, dentro de los controladores se desarrollarán los triggers o disparadores, explotando la habilidad rápida de escritura y lectura en MongoDB. Estos micro servicios se interconectan entre si por medio de Express, creando una App dando como respuesta un JSON a modo API REST.

Contando con los servicios subidos a una plataforma CLOUD o en este caso para pruebas a nivel local, funciona en un puerto específico y si es necesidad de la empresa se la puede distribuir en distintos servidores. Al comprobar las Apis, por medio de

Postman, está preparado el entorno para la comunicación con el FronEnd, cuyo resultado favorable viene siendo React y ReactNative, para la reutilización de código.

3.2. Metodología de desarrollo

3.2.1. Planificación

Cuando se trata de metodologías ágiles, Scrum encabeza el listado con el 56% de aceptabilidad [28], comprendido por historias de usuario, roles, seguimiento de cada actividad, entre otros aspectos. Con todos los beneficios que ofrece Scrum, no se adapta al presente trabajo investigativo. La segunda opción mas usada es la metodología híbrida, donde el investigador seleccionara características de diferentes metodologías que se adapten al proyecto a realizar.

3.2.1.1. Levantamiento de Historias de Usuario

Tabla 9: Historia de Usuario 1: Diseño de la Base de Datos

Elaborado por: El investigador

<table>
<tr><th colspan="3">Historia de usuario</th></tr>
<tr><td>Número: 1</td><td colspan="2">Usuario: Virtual Design, El Taller, Banovo, Magia y Láser</td></tr>
<tr><td colspan="3">Nombre historia: Diseño conceptual de la Base de Datos</td></tr>
<tr><td>Prioridad en negocio: Alta</td><td colspan="2">Riesgo en desarrollo: Alto</td></tr>
<tr><td>Programador responsable:
Josué Vélez</td><td colspan="2">Iteración asignada: 1</td></tr>
<tr><td colspan="3">Descripción:
Como microempresa con una gran trayectoria en el diseño y corte láser
Deseamos un sistema de control de procesos de la elaboración de nuestros servicios, gestionando los mismos.
Para llevar un control y registro de estado actual de una orden de pedido.</td></tr>
<tr><th colspan="3">Criterios de aceptación</th></tr>
<tr><td colspan="3">Dado que el sistema manejara las ordenes de pedido
Cuando los clientes llegan a un acuerdo
Entonces diseñar una base de datos que incluya las tablas y campos que permitan almacenar la información correspondiente a cada documento.</td></tr>
</table>

Historia de Usuario 1

Figura 9: Tablas básicas para la base de datos.

Tabla 10: Historia de Usuario 2: UX del CRUD de Usuarios

Elaborado por: El investigador

Historia de usuario	
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Diseño del CRUD de los Usuarios	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** un registro, actualización, activación, y eliminación de usuarios. **Para** que gestionen los procesos del sistema	

Tabla 11: Historia de Usuario 2.1: Guardar los Datos del Usuario

Elaborado por: El investigador

Historia de usuario	
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Guardar los datos de los colaboradores	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que el usuario se registre con los siguientes datos cedula, nombres completos, celular, dirección, email y una contraseña **Para** que puedan iniciar sesión en el sistema	
Criterios de aceptación	
Dado que el usuario se registrará en el sistema **Cuando** de click en guardar **Entonces** los datos se guardarán en el sistema, pero aun no podrá iniciar sesión hasta que se active al usuario. **Criterio 2** **Dado** que el usuario se registrará en el sistema **Cuando** de click en guardar y no halla registrado todos los campos **Entonces** se manejarán alertas para controlar estos errores de campos vacíos.	

Historia de Usuario 2

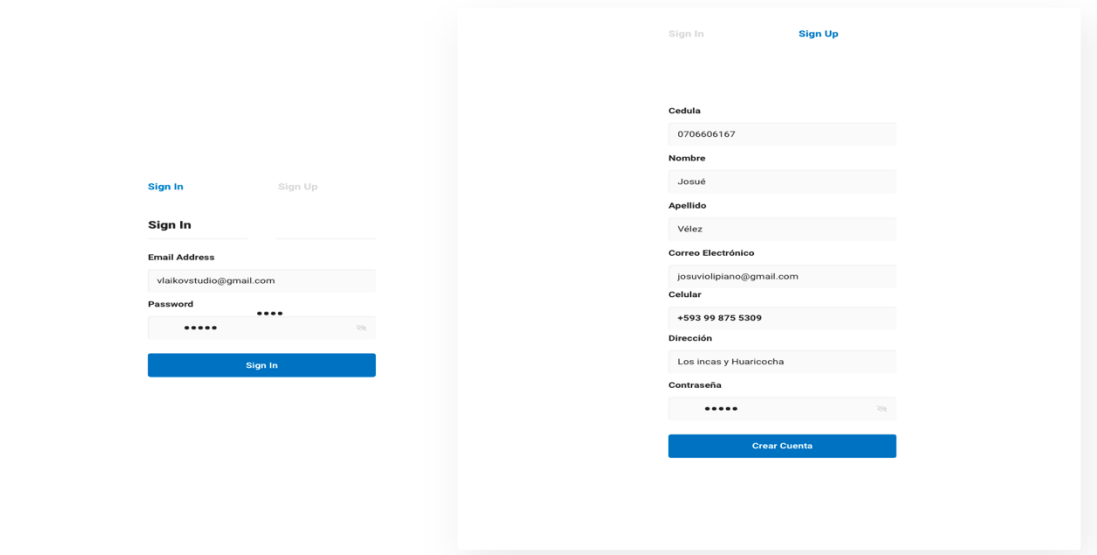


Figura 10: Login y Registro de Usuarios.

Tabla 12: Historia de Usuario 2.2: Modificar los Datos del Usuario

Elaborado por: El investigador

	Historia de usuario
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Modificar los datos de los colaboradores	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que se puedan editar el nombre completo, correo, teléfono, Avatar, dirección y contraseña **Para** tener datos reales del usuario	
	Criterios de aceptación
Dado que un usuario loggueado en el sistema y desee actualizar sus campos **Cuando** de click en actualizar **Entonces** los datos se modifiquen en el sistema y se reflejen esos cambios.	

Tabla 13: Historia de Usuario 2.3: Eliminar un Usuario

Elaborado por: El investigador

	Historia de usuario
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser

Nombre historia: Eliminar un colaborador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que un administrador pueda eliminar algún colaborador que salga de la empresa. **Para** cuidar la integridad de la empresa	

Historia de Usuario 2

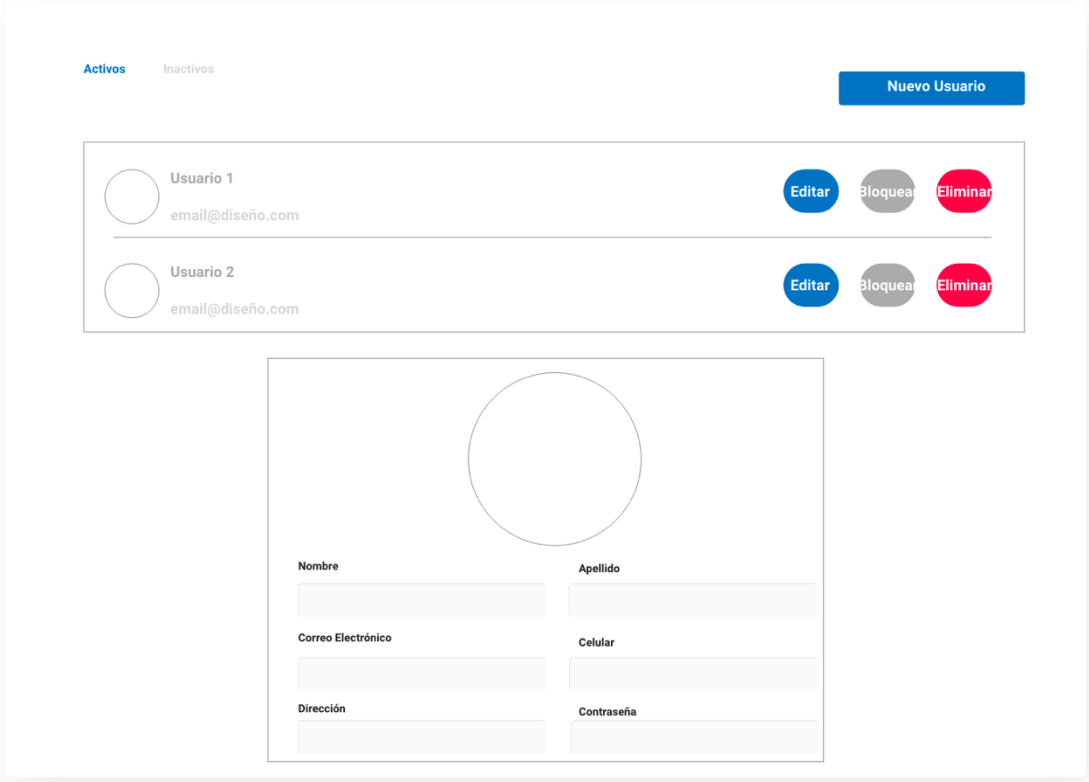


Figura 11: Administración de Usuarios

Tabla 14: Historia de Usuario 3: Administración de Procesos personalizados

Elaborado por: El investigador

		Historia de usuario
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser	
Nombre historia: Guardar los procesos de cada microempresa		
Prioridad en negocio: Alta	**Riesgo en desarrollo:** alta	
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1	
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser		

Deseamos que exista una administración de procesos independientemente de la organización, siendo procesos personalizados **Para** que se puedan gestionar y controlar la elaboración de una orden de pedido

Tabla 15: Historia de Usuario 3.1: Crear los procesos

Elaborado por: El investigador

	Historia de usuario
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Guardar los procesos personalizados	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que se puedan guardar el nombre y la descripción de cada proceso para la elaboración de una orden. **Para** el control y gestión de los procesos	
	Criterios de aceptación
Dado que se guardarán los procesos **Cuando** de click en guardar **Entonces** los datos se guardarán en el sistema, controlando que se ingresen los datos pertinentes.	

Tabla 16: Historia de Usuario 3.2: Modificar los procesos

Elaborado por: El investigador

	Historia de usuario
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Modificar los procesos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que se modifiquen los procesos **Para** que se reflejen datos reales de procesos.	
	Criterios de aceptación
Dado que se modificarán los datos **Cuando** de click en actualizar **Entonces** los datos se modifiquen en el sistema	

Tabla 17: Historia de Usuario 2.3: Eliminar un Proceso

Elaborado por: El investigador

Historia de usuario	
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Eliminar un proceso	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que un administrador pueda eliminar algún proceso infructuoso de la elaboración de una orden de pedido **Para** evitar cuellos de botella o una mejor gestión de procesos.	

Historia de Usuario 3

Figura 12: Administración de Procesos

Tabla 18: Historia de Usuario 4: Administración de Ordenes de Pedido

Elaborado por: El investigador

Historia de usuario	
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Guardar las ordenes de Pedido	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser	

Deseamos que se ingresen las ordenes de pedido con su respectivo detalle, y los abonos respectivos **Para** conocer los requerimientos del cliente con respecto a su orden.

Tabla 19: Historia de Usuario 4.1: Guardar los Clientes

Elaborado por: El investigador

		Historia de usuario
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser	
Nombre historia: Guardar Clientes		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alto
Programador responsable: **Josué Vélez**		**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que se ingresen los clientes nuevos al crear la orden de pedidos **Para** contar con una base de datos de clientes		
		Criterios de aceptación
Dado que se ingresará una orden de pedido **Cuando** de ingrese la C.I. del cliente, si el cliente ya esta ingresado se carguen los datos del usuario o si no se abra una ventana para ingresar un nuevo cliente, con los mismos campos que un usuario **Entonces** se ingresará a la base de datos el nuevo cliente y se refrescará la cabecera del orden de Pedido		

Tabla 20: Historia de Usuario 4.2: Ingresar Cabecera de Pedido

Elaborado por: El investigador

		Historia de usuario
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser	
Nombre historia: Ingresar Cabecera de Pedido		
Prioridad en negocio: Alta		**Riesgo en desarrollo:** Alto
Programador responsable: **Josué Vélez**		**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que se ingrese la cabecera del Pedido **Para** generar un número de orden y vincularlo a los detalles del Pedido.		
		Criterios de aceptación
Dado que se ingresará una orden de pedido **Cuando** se de click en el check **Entonces** se guardará la cabecera en la base de datos.		

Tabla 21: Historia de Usuario 4.3: Ingresar Detalle del Pedido

Elaborado por: El investigador

Historia de usuario	
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Ingresar Detalle del Pedido	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que se ingrese el detalle de cada orden de pedido **Para** generar datos contables de la orden de pedido	
Criterios de aceptación	
Dado que se ingresará los detalles **Cuando** se ingrese cada detalle **Entonces** se actualizarán los datos contables como el subtotal y total de la orden	

Tabla 22: Historia de Usuario 4.3: Ingresar Abono del Pedido

Elaborado por: El investigador

Historia de usuario	
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Ingresar Abono del Pedido	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que se ingrese el abono de cada orden de pedido **Para** llevar un control de entrega del pedido	
Criterios de aceptación	
Dado que se ingrese el abono de la orden **Cuando** se de click en guardar **Entonces** se actualizarán los datos contables y guardar el abono de la orden.	

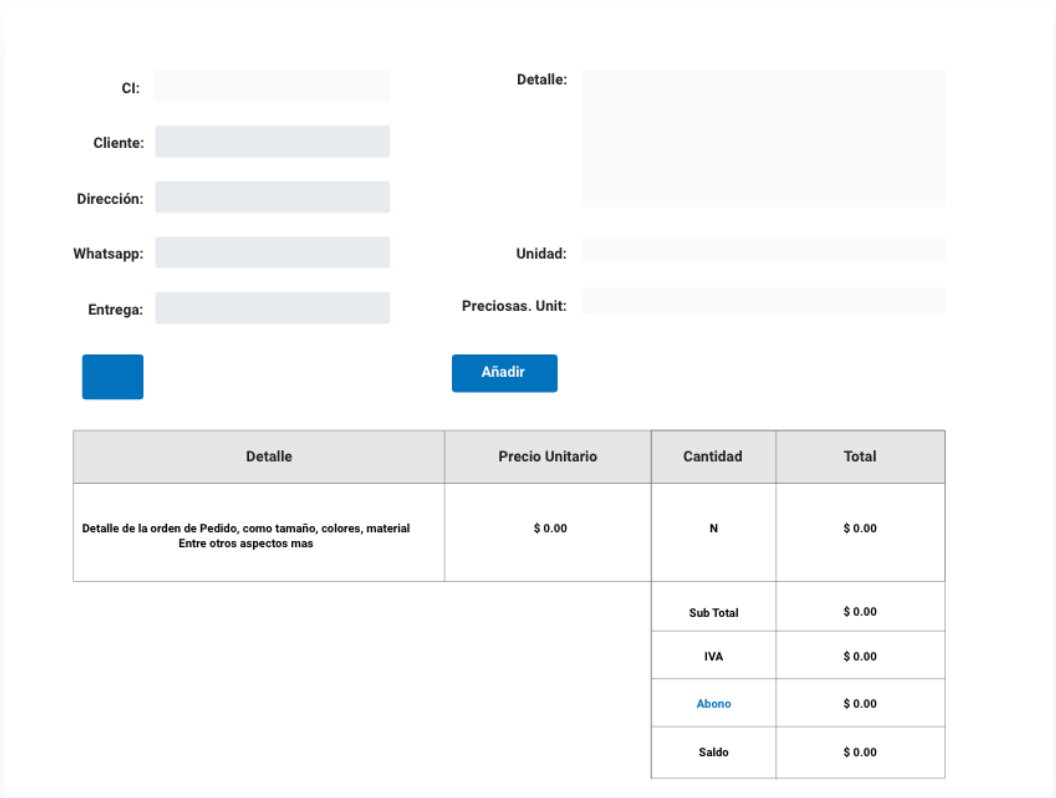


Figura 13: Administración de Pedidos

Tabla 23: Historia de Usuario 5: Control de Procesos

Elaborado por: El investigador

Historia de usuario	
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Control de Procesos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** que se pueda ver el estado actual de una orden de pedido y actualizar el mismo **Para** gestionar los pedidos correctamente.	
Criterios de aceptación	
Dado que se ha ingresado una orden de pedido, se genera un estado inicial **Cuando** se de click en el proceso en el que se encuentra **Entonces** se actualizarán el proceso de la orden de pedido.	

Estado de la orden de Pedido

Oredn	Detalle	Cant.	Inicio	Proc.1	Proc.2	Proc.3	Proc.4	Proc.5	Proc.n	Fin
1		n	☐	☐	☐	☐	☐	☐	☐	☐
2		n	☐	☐	☐	☐	☐	☐	☐	☐
3		n	☐	☐	☐	☐	☐	☐	☐	☐
4		n	☐	☐	☐	☐	☐	☐	☐	☐
5		n	☐	☐	☐	☐	☐	☐	☐	☐

Figura 14: Estado de la orden de Pedido

Tabla 24: Historia de Usuario 6: Informes

Elaborado por: El investigador

	Historia de usuario
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Informes	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador responsable: Josué Vélez	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** ver los ingresos Diarios, ordenes pendientes por fecha y la orden de pedido **Para** gestionar procesos básicos de la organización	

Tabla 25: Historia de Usuario 6.1: Ingresos Diarios

Elaborado por: El investigador

	Historia de usuario
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Ingresos Diarios	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto

Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** ver los ingresos diarios de las ordenes emitidas o abonos ingresados **Para** analizar el estado contable de la empresa	
Criterios de aceptación	
Dado que se han ingresado abonos de las ordenes de pedido **Cuando** se de click en la opción de ingresos **Entonces** se genera un reporte de los ingresos diarios.	

Tabla 26: Historia de Usuario 6.2: Ordenes Pendientes

Elaborado por: El investigador

Historia de usuario	
Número: 1	**Usuario:** Virtual Design, El Taller, Banovo, Magia y Láser
Nombre historia: Ingresos Diarios	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador responsable: **Josué Vélez**	**Iteración asignada:** 1
Descripción: **Como** microempresa con una gran trayectoria en el diseño y corte láser **Deseamos** ver las ordenes pendientes **Para** gestionar el proceso de las mismas	
Criterios de aceptación	
Dado que en el día a día se ingresan ordenes de pedido **Cuando** se de click en la opción de ordenes pendientes **Entonces** se genera un listado con las ordenes pendientes del día o una búsqueda por fecha.	

3.2.2. Modelamiento de Micro servicios

Los micro servicios son parte fundamental de la arquitectura planteada, al desarrollarse de manera independiente, se logra una mejor comunicación con la base de datos y el frontend. En las tablas 9, 10 y 11 se visualiza los micro servicios categorizados por servicios de ingreso, egreso y contables.

Tabla 27: Micro servicios de ingresos

Elaborado por: El investigador

Ingresos
Clientes
Ordenes de Pedido
Procesos

Tabla 28: Micro servicios de egresos

Elaborado por: El investigador

Egresos
Ordenes de Pedido
Estado del Producto

Tabla 29 : Micro servicios de contabilidad

Elaborado por: El investigador

Contabilidad
Ingresos diarios en ordenes de Pedido
Abono

3.2.2.1. Establecer actividades en cada módulo.

En cada micro servicio se encuentras distintos módulos que se pueden conectar entre sí. Gracias a la observación participativa se pudo llevar a cabo un modelamiento de actividades, creando los casos de usos.

El principal actor en los micro servicios es el administrador quien, crea, actualiza, lista clientes, ordenes y procesos (Figura 9), de igual forma podrá ver el estado de cada pedido con su respectivo detalle, dependiendo de los procesos administrados (Figura 10). Al llevar el ingreso de ordenes y su abono consecutivo, el actor visualizara los ingresos diarios categorizados por forma de pago (Figura 11).

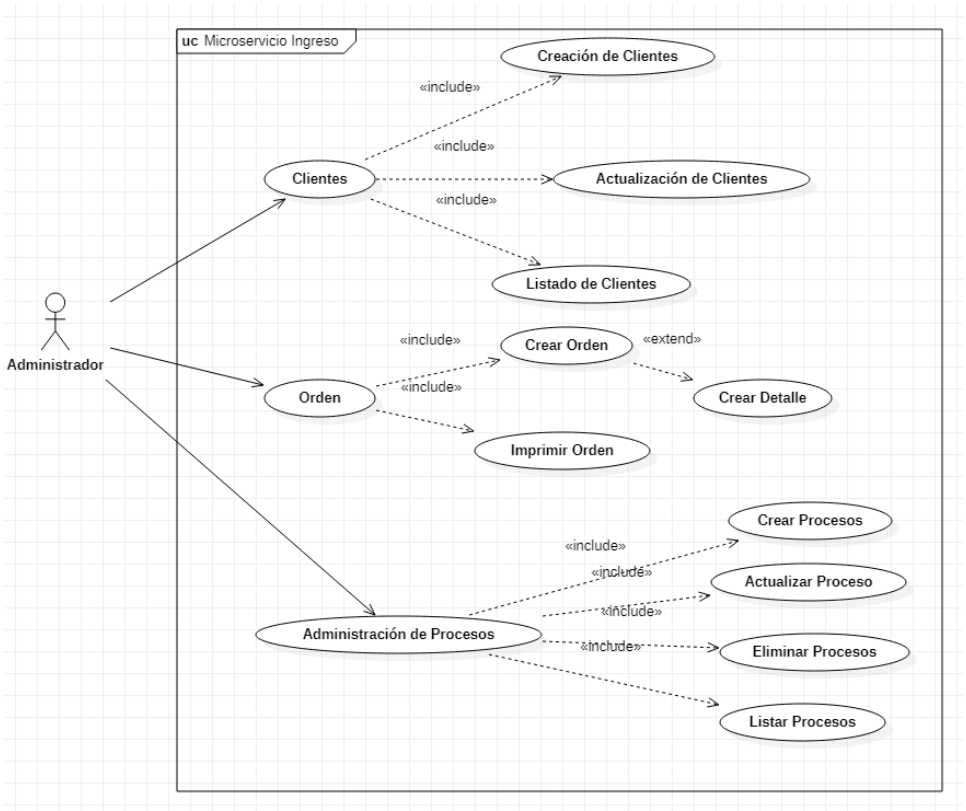


Figura 15: Caso de Usos Micro servicio de Ingreso

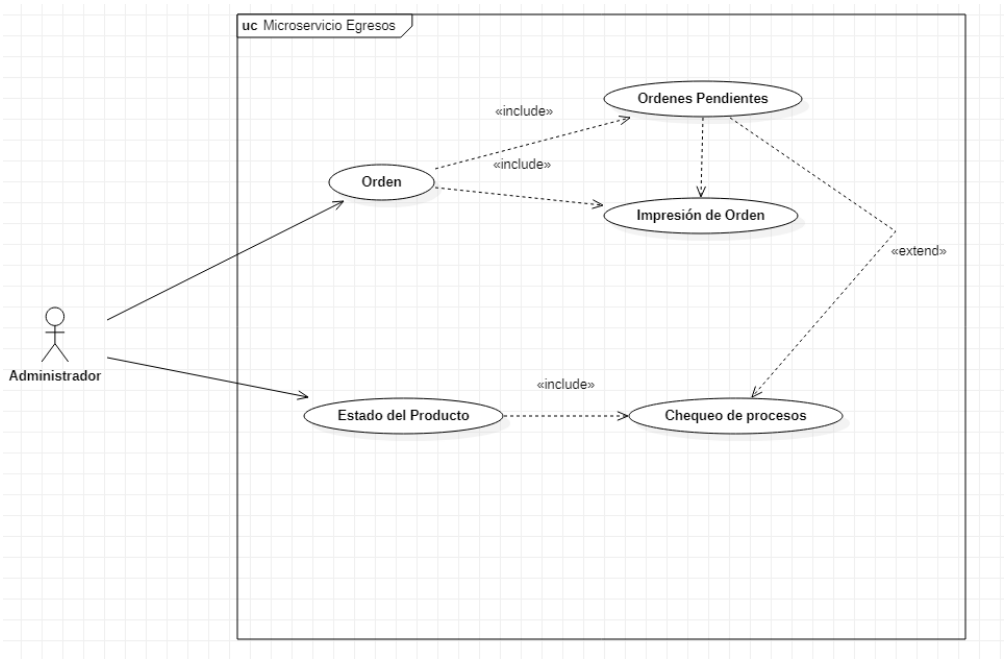


Figura 16: Caso de Usos Micro servicio de Egreso

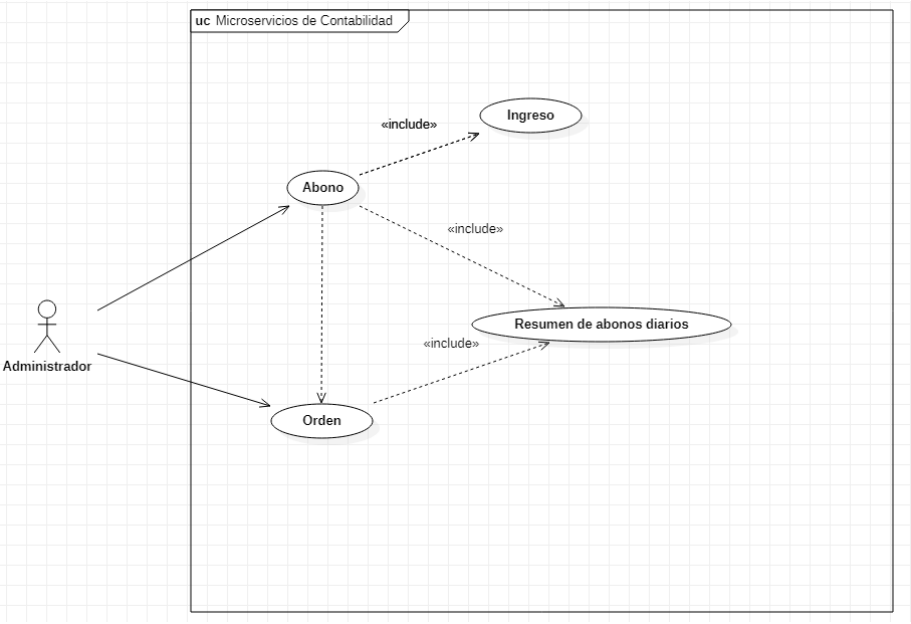


Figura 17: Caso de Usos Micro servicio de Contabilidad

- **Diagrama de Actividades**

La validación de datos es parte fundamental para el aplicativo, para aquello es importante encriptar los datos y crear un token de inicio de sesión. Todo esto inicia al ingreso de datos de acceso, puede ser un correo y contraseña, autenticación local o autenticación externa, cualquiera que sea el método de acceso debe haber una validación de datos. Al estar todo en orden se conecta con la base de datos y devuelve un resultado favorable (Figura 12)

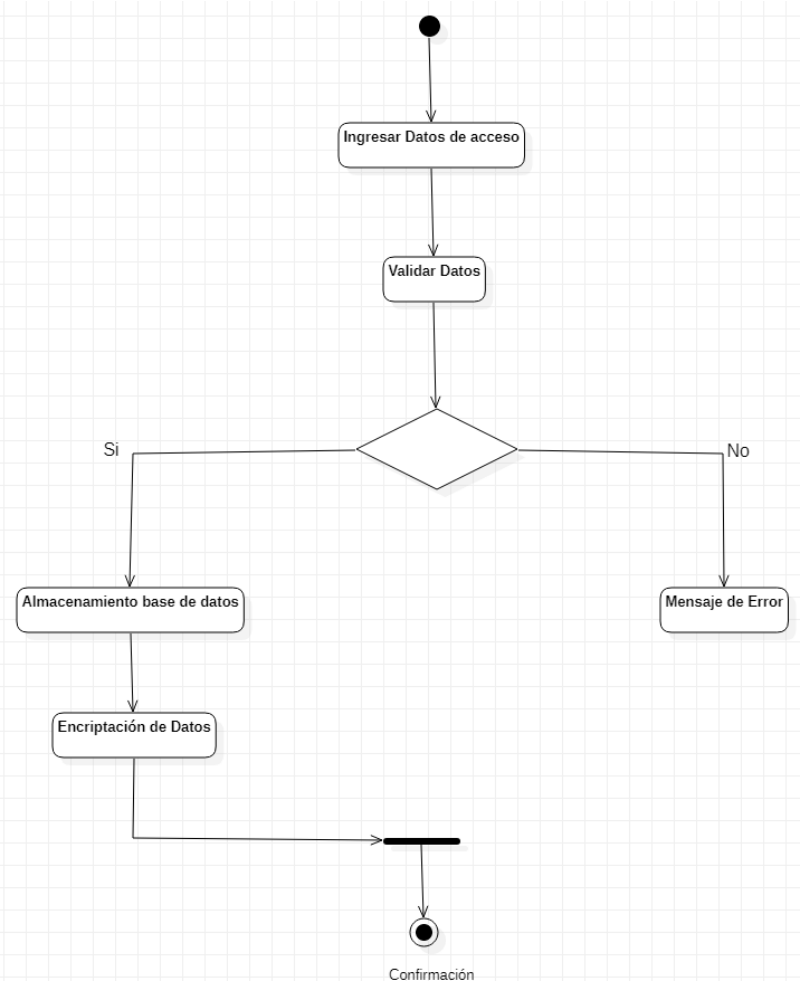


Figura 18: Diagrama de Actividades del Login

Con el planteamiento de los casos de uso y diagramas de actividades, se generará un modelamiento de base de datos NoSQL, como se puede visualizar en la Figura 15.

- Desarrollo de tareas

El código del trabajo investigativo esta desarrollado en el IDE de programación Microsoft Visual Studio Code, dado que cuenta con la integración de diversos lenguajes, herramientas, librerías, entre muchas opciones más. Para un mejor entorno de programación es necesario instalar distintas extensiones, tanto para el acceso rápido a las librerías, como para la comunicación arquitectónica (Figura 13)

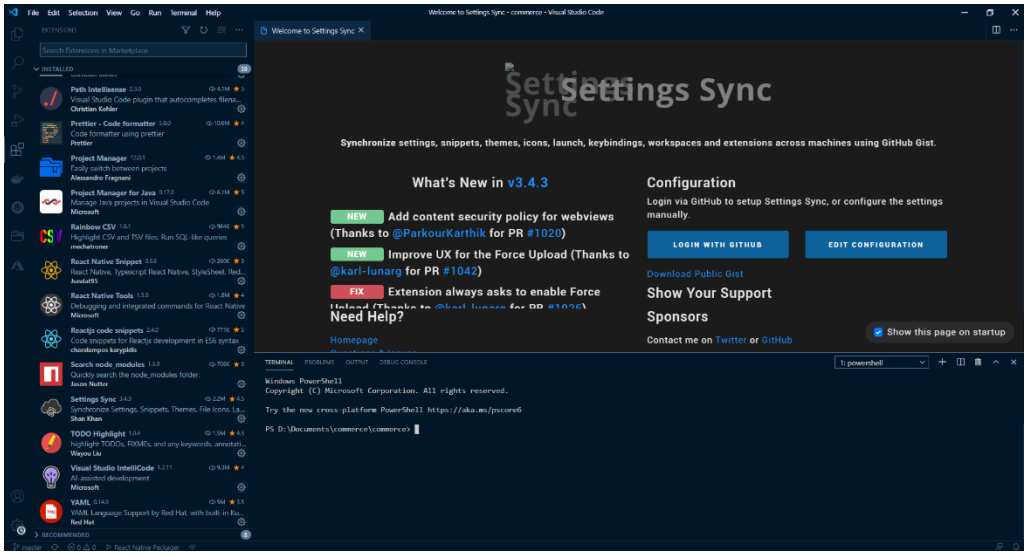


Figura 19 : Visual Studio Code y extensiones necesarias.

La base de datos no relacional MongoDB esta alojada en Atlas, un servicio cloud propio de MongoDB. En la Figura 14 se puede apreciar las colecciones o documentos de la base de datos a usar, a diferencia del manejo de datos SQL, como lo es Microsoft SQL Server, al crear las tablas y relaciona automáticamente lanza un modelo de datos. En el caso de Mongo al manejarse por documentos, el esquema se usa para representación de las relaciones en softwares externos.

Cada colección almacena documentos en formato JSON. Esto no solo permite una rápida consulta o escritura de datos, su sistema de almacenamiento esta optimizado, permitiendo que los costos de consultas sean económicos. Además, Atlas permite consultas rápidas en base a parámetros, teniendo en cuenta la arquitectura utiliza, JSON Request, JSON Response, en otras palabras, se requiere un formato JSON para las consultas, y regresa una colección JSON.

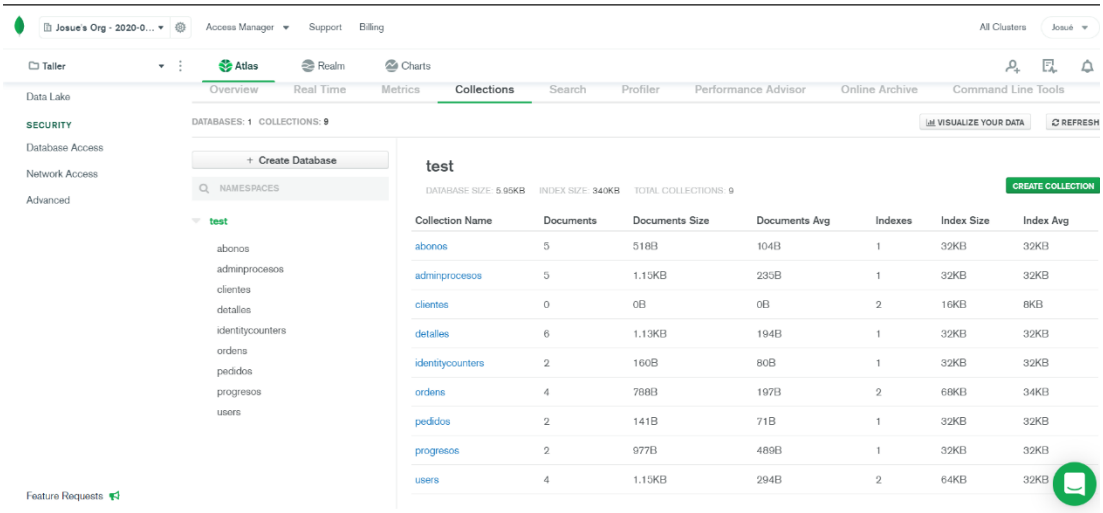


Figura 20: Interfaz Atlas

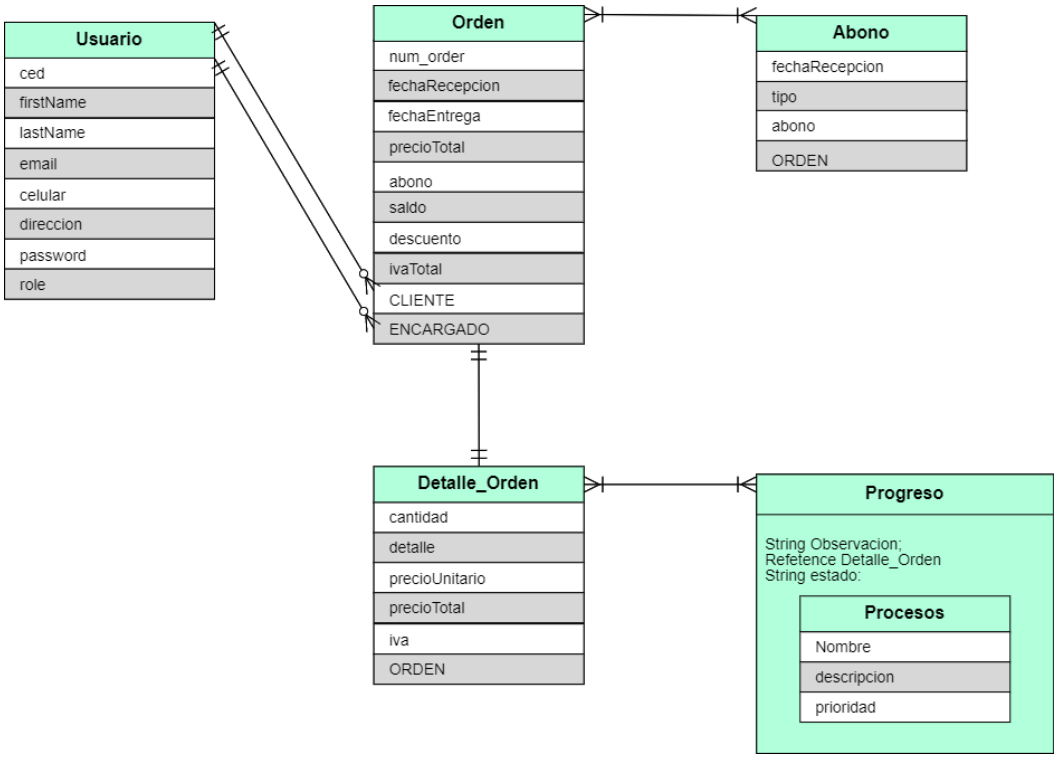


Figura 21: Esquema de base de Datos NoSQL

Para datos de ejemplo, se ingresarán datos pertinentes, sin embargo, al ser un sistema general, el usuario final ingresara los clientes, procesos y documentos necesarios para dar vida a su software.

- Desarrollo del BackEnd

La lógica de negocio, validaciones, manejo de base de datos, triggers, funciones complejas, entre algunos aspectos similares, son lo que engloban la terminología BackEnd. Este se desarrolla en base a modelos, controladores, middleware, rutas y encriptación de datos, en los lenguajes de programación Node y Express.

Models: Los modelos detallan la creación de ítems de cada documento, es decir, crean los esquemas de la base de datos no relacional, de igual manera se implementan validaciones, campos auto incrementables, datos por default, reglas, enumeraciones, y muchas mas características para crear un modelo adecuado y optimo.
Controllers: Los controladores en base a su respectivo modelo, crean los ingresos, consultas, proyectos, agregaciones, triggers, actualizaciones o eliminaciones que se gestionan dentro de la base de datos. En este segmento de la arquitectura es donde se generan los Request, Response JSON.
Routes: La librería express gestiona las funciones creadas en cada controlador, creando las rutas de cada micro servicio. Aquí se controlará la autenticación, autorización o cualquier middleware que deseemos usar.
Middleware: En su contenido se encuentra el fichero de autenticación de usuario y su respectivo control para evitar ataques a la base de datos.
SRC: Su archivo permite crear y refrescar los tokens para el login.

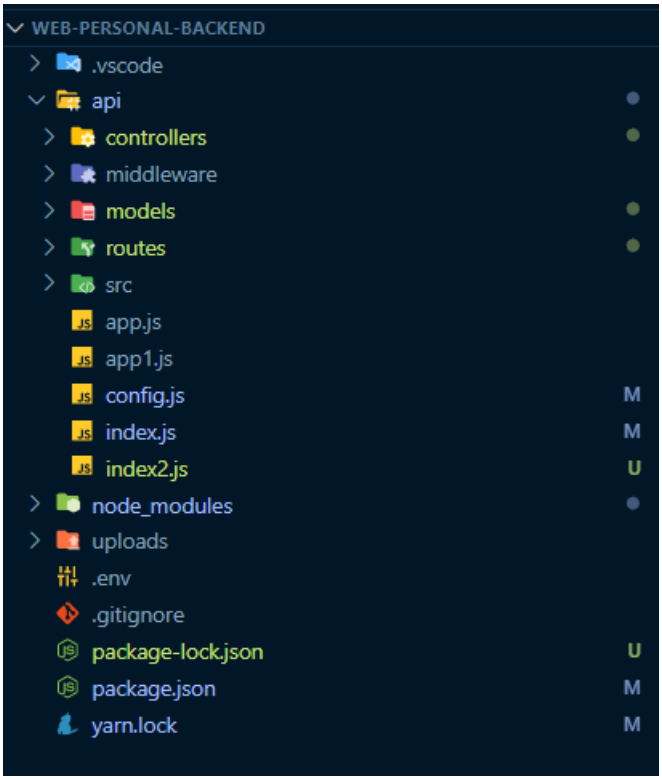


Figura 22 : Estructura BackEnd

Creando conexión con la base de Datos.

A diferencia con SQL Server, las conexiones a mongoDB son más sencillas y fáciles implementar (Figura 17), dado que desde la plataforma MongoDB Atlas al crear un usuario y contraseña, nos otorga un link de conexión al crear dicha base. En la Figura 16 se aprecia las constantes que se usarán dentro de la aplicación, se manejan bajo esa estructura para evitar ataques directo a las APIs por medio de las opciones de desarrollador que cada navegador de internet pose.

Figura 23 : Archivo de configuración

```
const express = require("express");
var cors = require("cors");
const bodyParser = require("body-parser");
const mongoose = require("mongoose");
const app = express(); //inicialixzzar express
const http = require("http");

const { PORT, IP_SERVER ,URI} = require("./config");

mongoose.connect(
  URI,
  {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  },
  (err, res) => {
    app.listen(PORT, () => {
      console.log("####################");
      console.log("######API REST######");
      console.log("####################");
      console.log(`http://${IP_SERVER}:${PORT}/api/`);
    });
  }
);
```

Figura 24 : Conexión con MongoDB

Seguridad en la Autenticación de usuario.

A pesar de ser un gestor de datos seguro, mongo no está extenso a ataques informáticos, la solución en bases de datos relacionales para el inject SQL es crear procedures y distintos procesos más, lo que no cuenta las bases No SQL, abriendo la posibilidad de ataques inject JSON. La alternativa en este caso no relacional, es el uso de autenticación y autorización por medio de token y encriptación de datos.

La base para la creación de tokens es JWT (Json Web Tokens), la cual permite decodificar y codificar documentos, en la Figura 19 se ve con claridad los métodos de creación de un Token, y que pasa cuando este expira, dependiendo el tipo de aplicación, la sesión puede durar unas horas o días. Cuando un método activa el JWT retorna un json con los datos especificados, en este caso, un accessToken y un refreshToken, en caso que el Token original haya expedido (Figura 20)

Figura 25 : Controlador Auth, creando y refrescando Tokens

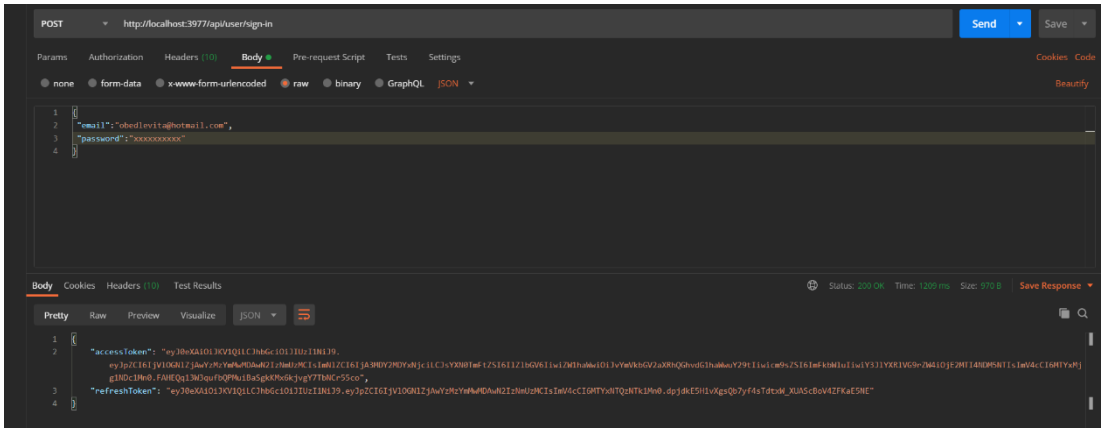


Figura 26 : Prueba Api encriptación de Usuario

El trabajo de seguridad no queda solo en la creación de tokens, sino, en la encriptación unidireccional de datos sensibles, como lo es la contraseña. Gracias a la librería bycript, se encriptará los datos necesarios y al momento de usar una UI se comparará la encriptación con el dato ingresado por UI, es decir, estos datos no se pueden des encriptar (Figura 21).

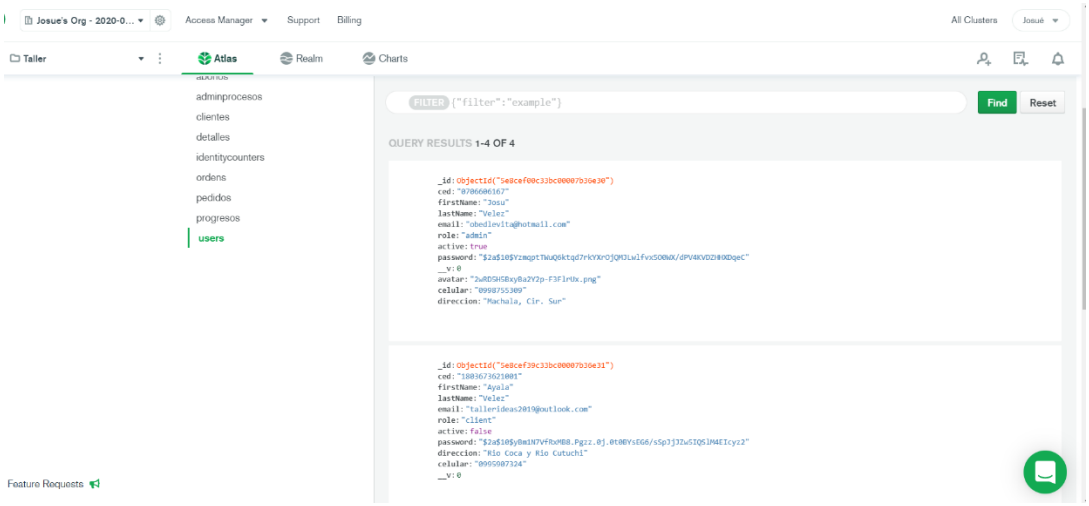


Figura 27 : Contraseñas encriptadas

El middleware permite que usuarios autentificados ingresen a rutas específicas, esto se maneja dentro del esquema routes.

En la Figura 22, en la línea de código 4 se llama a la clase authenticated, creando un middleware md_auth. Se aprecia claramente en el método get, para obtener los usuarios el uso del middleware, si no tiene una cabecera de autentificación, no se podrá acceder a dicho servicio.

Cuando una ruta cuenta con el middleware de autenticación, dentro de un navegador de Apis, es necesario agregar en la cabecera el token correcto, para acceder al micro servicio, caso contrario dará un error como se visualiza en la Figura 23 y 24.

User.routes.js

api > routes > User.routes.js > ...

```
1  const express = require("express");
2  const UserController = require("../controllers/user.controller");
3  const multipart = require("connect-multiparty");
4  const md_auth = require("../middleware/authenticated");
5  const md_upload_avatar = multipart({ uploadDir: "./uploads/avatar" });
6
7  const api = express.Router();
8
9  api.post("/sign-up", UserController.signUp);
10 api.post("/sign-up-admin", UserController.signUpAdmin);
11 api.post("/sign-in", UserController.signIn);
12 api.get("/users", [md_auth.ensureAuth], UserController.getUsers);
13 api.get("/users-active", [md_auth.ensureAuth], UserController.getUsersActive);
14 api.put(
15   "/upload-avatar/:id",
16   [md_auth.ensureAuth, md_upload_avatar],
17   UserController.uploadAvatar
18 );
19 api.get("/avatar/:avatarName", UserController.getAvatar);
20 api.put("/update-user/:id", [md_auth.ensureAuth], UserController.updateUser);
21 api.put(
22   "/activate-user/:id",
23
24   UserController.activateUser
25 );
26 api.delete("/delete-user/:id", [md_auth.ensureAuth], UserController.deleteUser);
27 api.post("/get-client/:CI", [md_auth.ensureAuth], UserController.getClientes);
28
29 module.exports = api;
```

Figura 28 : Rutas Express del Usuario.

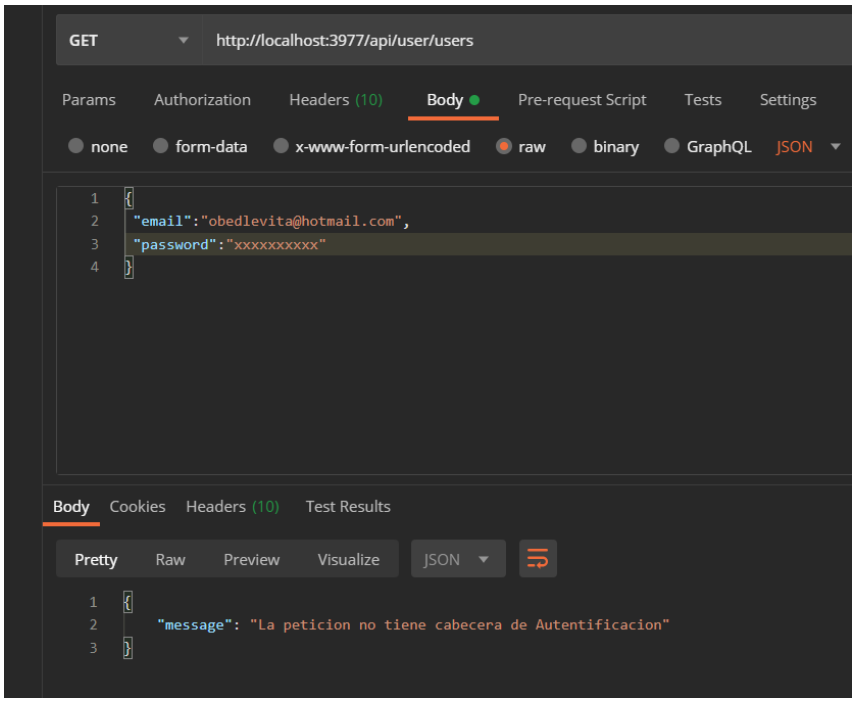


Figura 29 : Falla de uso de servicio por falta de Token

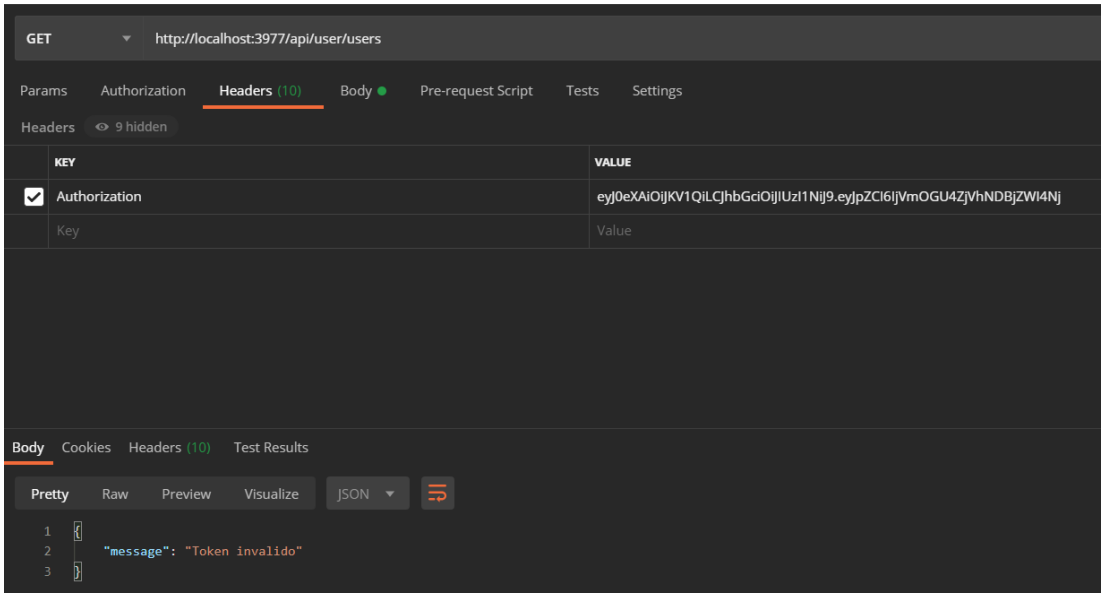


Figura 30 :Falla de uso de servicio por Token invalido.

Validación de datos

Full Stack Mern permite que sus validaciones de datos se manejen desde la lógica de negocio a un nivel más profundo. Los esquemas dentro de MongoDB poseen opciones para definir el tipo de dato, límites de cadena, y en lo más resaltante sus validaciones. En la Figura 25 se aprecia distintas validaciones como que sea de tipo único, la transformación del texto a minúsculas. La validación compleja en este caso es dada por una librería que comprueba que el JSON Request email cumpla con las normas de formato.

```
email: {
  type: String,
  required: true,
  unique: true,
  trim: true,
  lowercase: true,
  validate: {
    validator(value) {
      return validator.isEmail(value);
    },
    message: (props) => `${props.value} no es un correo valido`,
  },
},
```

Figura 31 : Validación de Correo Electrónico

En el modelo Abono, se a creado una función con 3 datos constantes (Figura 26), los cuales son validados por una enumeración, lo que se conoce como columnas calculadas. Su rango de validaciones es muy extenso, gracias a la comunidad de programadores que mantienen estos lenguajes a flote.

Figura 32 : Columnas calculadas con enumeración.

Comunicación en Frío y Caliente

Las rutas se comunican por medio de Express, una potente herramienta para crear servicios web REST, sin embargo, es cuestión del desarrollador realizar una estructura de comunicación mixta. La programación en Frío y Caliente tienen sus grandes falencias al trabajar por separado, es igual que el sistema de respaldo de base de datos en discos.

Si no se trabaja con una arquitectura mixta en el aspecto de conexión, al crecer la aplicación, el exceso de entradas a los servicios satura las respuestas, considerando que la base de datos a trabajar entrega documentos inmediatos. Por ello en el proyecto de investigación se ha planteado una conexión general conocida como comunicación en Frío, y a partir de ellas pequeñas conexiones (Figura 27), en otras palabras, comunicación en Caliente.

```
app.use(`/api/user`, authRoutes);
app.use(`/api/user`, userRoutes);
app.use(`/api/orden`, ordenRoutes);
app.use(`/api/material`, materialRoutes);
app.use(`/api/orden/detalle`, detalleOrdenRoutes);
app.use(`/api/orden/detalle/subdetalle`, subDetalleRoutes);
app.use(`/api/orden/abono`, abonoRoutes);
app.use(`/api/prueba`, prueba);
app.use(`/api/progreso`, progresoRoutes);
app.use(`/api/proceso`, procesoRoutes);
```

Figura 33 : Comunicación mixta

Funciones complejas o disparadores.

Los triggers son una desventaja dentro de las bases de datos NoSQL, sin embargo, existen alternativas para ejecutar disparadores, estas permiten disparar funciones ante una acción. (Figura 28)

```
    //Inicializacion del Trigger, se llama a la clase Orden para el ingreso del total de la Orden, disparando al documento Orden desde el
    //Extracción de ivaTotal, precioTotal
Orden.findById(
  { _id: orden },
  { ivaTotal: 1, precioTotal: 1 },
  (err, ordenP) => {
    if (err) {
      console.log(err);
    } else {
      // Busqueda de la orden y actualización de la misma
      Orden.findByIdAndUpdate(
        { _id: ordenP.id },
        {
          //disparando la actualización del iva, precio y saldo
          $set: {
            ivaTotal: ordenP.ivaTotal + detalleO.iva,
            precioTotal:
              ordenP.precioTotal + detalleO.precioTotal + detalleO.iva,
            saldo:
              ordenP.precioTotal + detalleO.precioTotal + detalleO.iva,
          },
        },
        (err, res) => {
          if (err) {
            throw err;
          } else {
            let dias = 0;
            const start = moment(res.fechaEntrega).format("YYYY-MM-DD");
            const end = moment().format("YYYY-MM-DD");
            const fecha1 = moment(start);
            // console.log(fecha1);
            const fecha2 = moment(end);
            // console.log(fecha2);
            dias = fecha1.diff(fecha2, "days");
            //// Segundo Trigger de guardado, al crear el detalle, se crea automaticamente el progreso, por medio de la clase Progreso
            const progreso = new Progreso();
            progreso.Observacion = `Le quedan ${dias} dias`;
            progreso.detalleOrden = detalleStored._id;
            progreso.num_order = res.num_order;
            progreso.save();
```

Figura 34 : Función compleja o Disparador.

Librerías BackEnd

Para el uso de este proyecto de investigación se utilizaron diversas librerías generadas en el node_modules. La Figura 29 detalla todas las dependencias externas instaladas en el proyecto.

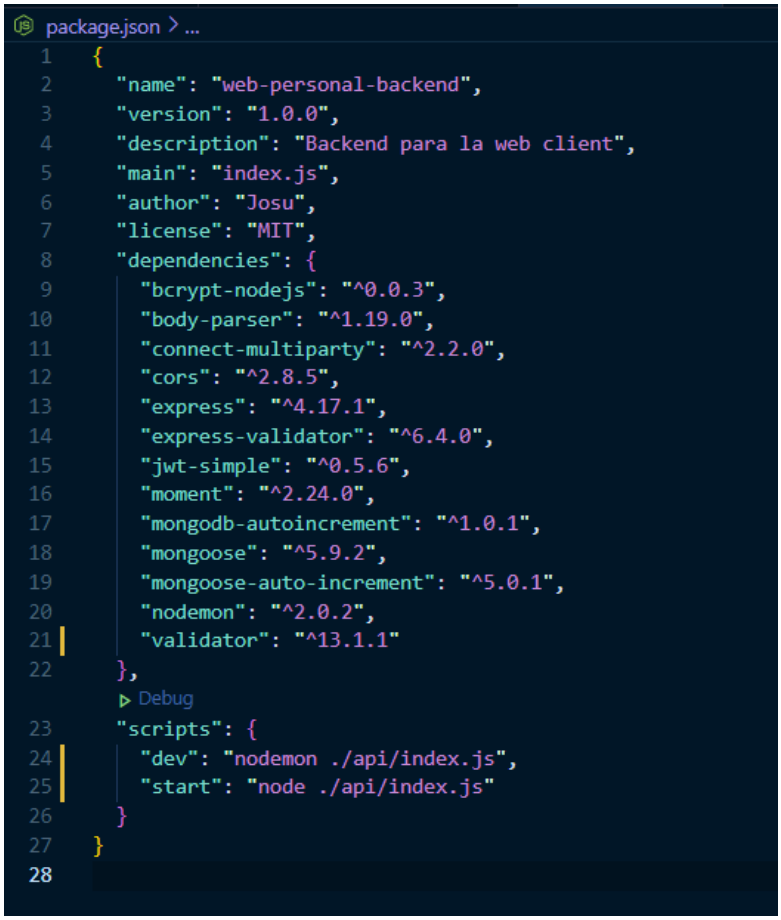


Figura 35 : Package.json

- **Desarrollo del FrontEnd**

El diseño del aplicativo móvil o web en React se maneja por jerarquías, su desarrollo se concentra en hook, reactividad, funcionalidad, visualización, ventanas, entre otras. Las carpetas dentro del proyecto React constan de: api, assets, components, config, hooks, layouts, pages, providers, scss, utils (Figura 30).

Api: Almacena la comunicación con el servidor web rest por medio de fetch, usando la url del servicio, mandando un json como requisito y un json de respuesta. esta carpeta es reutilizada para su uso en React Native y React, sin contar el hecho que los servicios web son el mejor ejemplo de reutilización de código.

Assets: Conjunto de multimedia como fotos o archivos que gestiona el aplicativo.

Components: Creación de componentes específicos para cada página, esto es parte de la refactorización de código, cada componente es utilizado una y otra vez en diferentes páginas, y dada su estructura facilita la edición de código.

Config: Rutas de comunicación de página a página.

Hooks: Contexto para la generalización de programación en frio por medio de un provider.

Layouts: Master del administrador y del sitio web.

Pages: Contiene todas las páginas organizadas de forma jerárquica y arquitectónico correcta.

Scss: Sistema de diseño del aplicativo, contando con el hecho de ser una potencial herramienta de diseño.

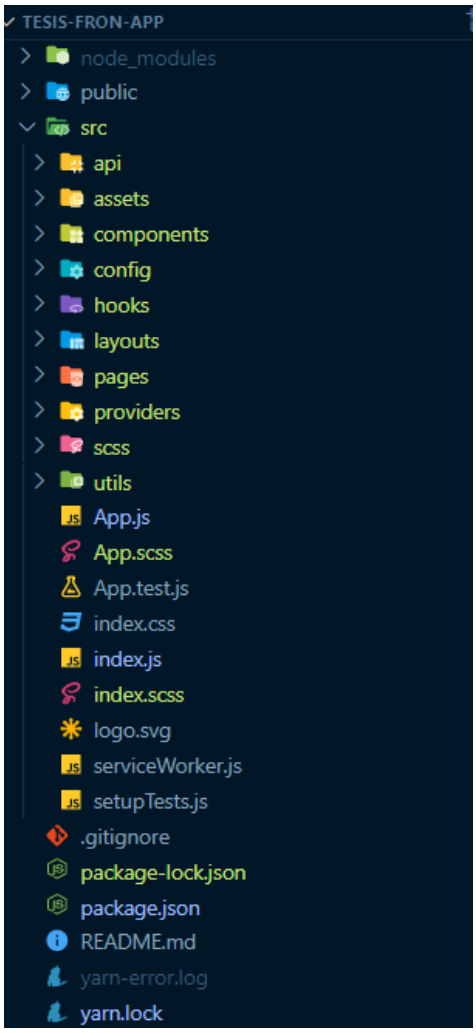


Figura 36 : Estructura FrontEnd

La refactorización y reutilización es vital dentro de este trabajo investigativo, en la Figura 31 y 32, se visualiza una descomposición de componentes y páginas que servirán para la UI, e incluso cada carpeta puede constar de sub carpetas, reflejando una correcta organización de programación. Esta reestructuración de código no altera en ningún aspecto el comportamiento externo del aplicativo, en otras palabras, la codificación puede pasar a terceras manos evitando los code smells, que no es más que elementos o componentes defectuosos.

Para evitar las estructuras complicadas o demasiado largas, cada clase tiene su propio archivo, controlando de igual manera la existencia de Middle man. Cada clase esta equilibrada en relación a sus funciones o métodos, especificando características principales.

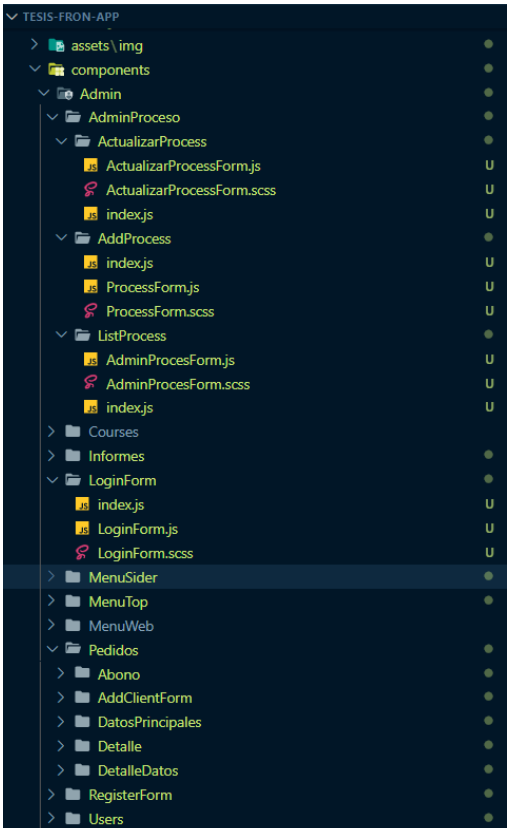


Figura 37 : Esquema de los componentes

Figura 38 : Esquema de páginas

Una de las soluciones que presenta la refactorización, es la lista de parámetros demasiado largas, esto se visualiza claramente en los archivos con extensión scss. (Figura 33 y 34)

Los .scss tienen un layout principal, el cual, con un solo cambio en la clase padre, cambia todas las clases heredadas, los siguientes algoritmos guardan las variables segmentadas por tipo.

Cada página se maneja a partir de un Scss, dando una percepción agradable al usuario final, sin embargo, todas ellas se comunicación con el Scss principal, manteniendo la refactorización desde un bajo nivel.

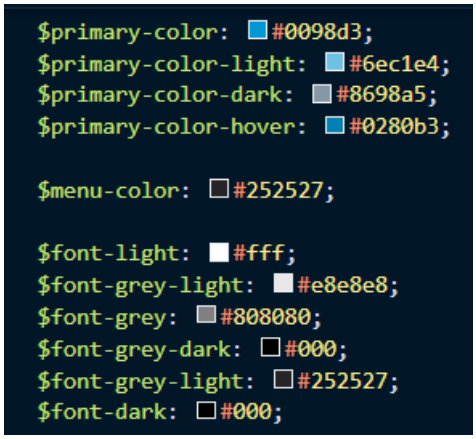


Figura 39 : Colores primarios y estilos de fuente

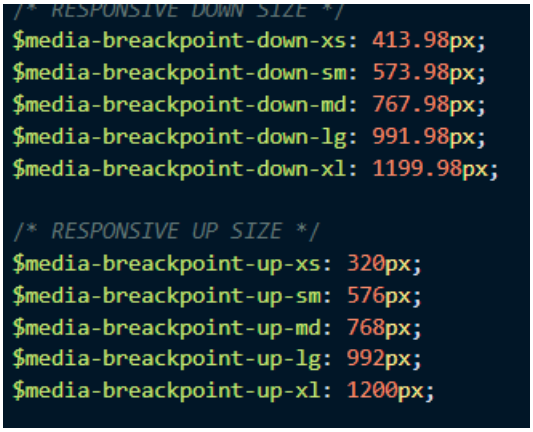


Figura 40 : Cambios de tamaño responsive

La fragmentación de código tiene como principio globalizar el proyecto en un archivo con la menor líneas de código. El App.js (Figura 35-36) contiene pocas líneas de código, donde el componente AuthProvider envuelve a todas las rutas de navegación del aplicativo.

Esta validación de componentes es resumida en dos líneas de código ejecutables, dado que el resto de la codificación son importaciones de clases y librerías necesarias.

Todo el código esta refactorizado, cumpliendo normativas en los nombres de variables y logrando una comunicación mixta. La estructura de componentes se maneja a tal grado que cada módulo trabaja con subcarpetas.

```
App.js > ...
import React from "react";
import "./App.scss";
import { BrowserRouter as Router, Route, Switch } from "react-router-dom";
import routes from "./config/routes";
import AuthProvider from "./providers/AuthProvider";

function App() {
  return (
    <AuthProvider>
      <Router>
        <Switch>
          {routes.map((route, index) => (
            <RouteWidthSubRoutes key={index} {...route} />
          ))}
        </Switch>
      </Router>
    </AuthProvider>
  );
}

function RouteWidthSubRoutes(route) {
  return (
    <Route
      path={route.path}
      exact={route.exact}
      render={(props) => <route.component routes={route.routes} {...props} />}
    />
  );
}

export default App;
```

Figura 41 : App.js

```
App.js
Progrssive > App.js > ...
1   import * as React from "react";
2   import { WebView } from "react-native-webview";
3
4   export default class App extends React.Component {
5     render() {
6       return (
7         <WebView
8           source={{ uri: URI }}
9           style={{ marginTop: 20 }}
10        />
11      );
12    }
13  }
14
```

Figura 42 : App.js de la aplicación móvil.

Cliente

Luego de un control de errores, refactorización, reutilización, se ejecuta el código, teniendo como resultado la UI. Los controles son importantes en esta parte del trabajo, dado que si en el Login, ingresan datos incorrectos y no existe una forma de saber si lo que hizo el usuario es correcto, el sistema es obsoleto.

Dentro de la ingeniería de software el termino usabilidad, representa la base de un sistema, mientras más fácil su uso, mayor alcance posee.

En la Figura 37, se visualiza el Login.js donde la contraseña fue incorrecta, a su vez, la Figura 38, refleja el error cuando el correo es incorrecto. Todas estas validaciones facilitan el uso del software.

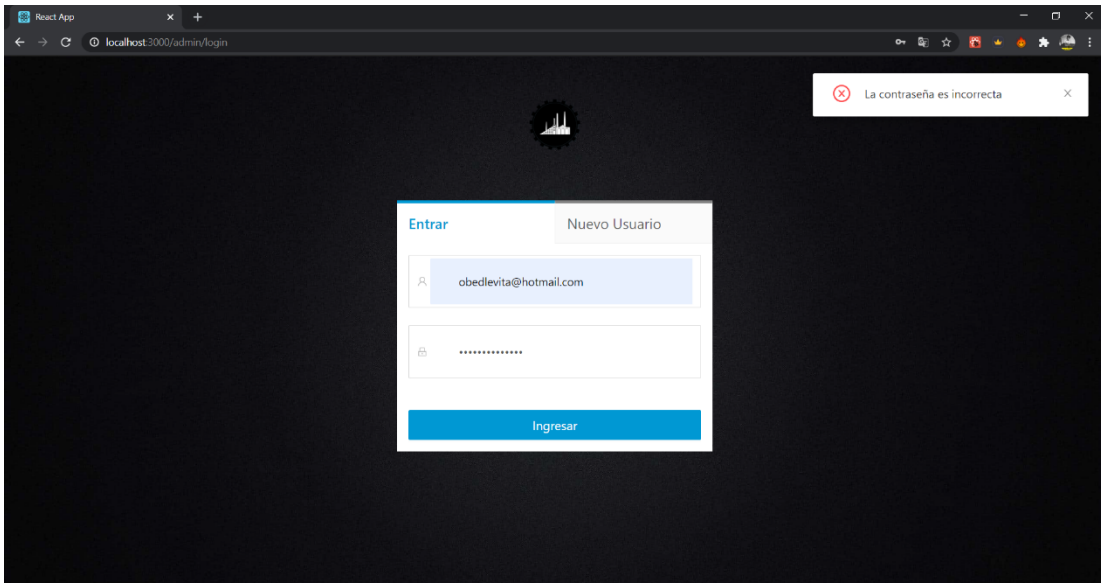


Figura 43 : Login, validando contraseña

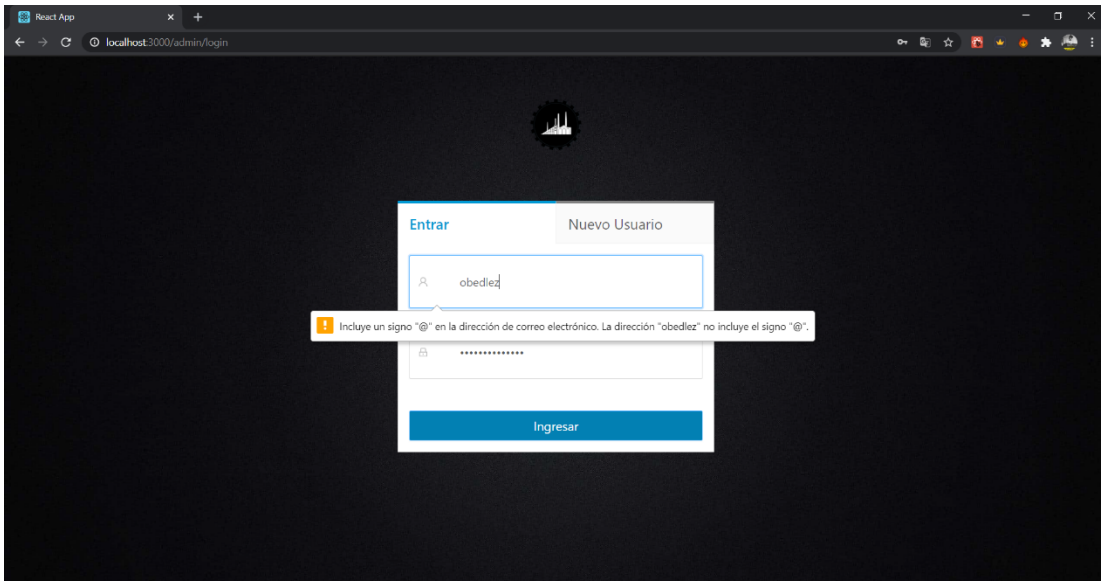


Figura 44 : Login, validando formato de correo

Como se lo menciono con anterioridad la validación del correo se origina en el BackEnd, sin embargo, para dar formato a estas alertas es necesario organizarlas dentro del FronEnd.

Dicha comunicación, presenta la portabilidad de un software, al iniciar sesión, se comunica con la API POST signInUser, el micro servicio devuelve el Token (Figura 39) lo que permite visualizar el menú principal.

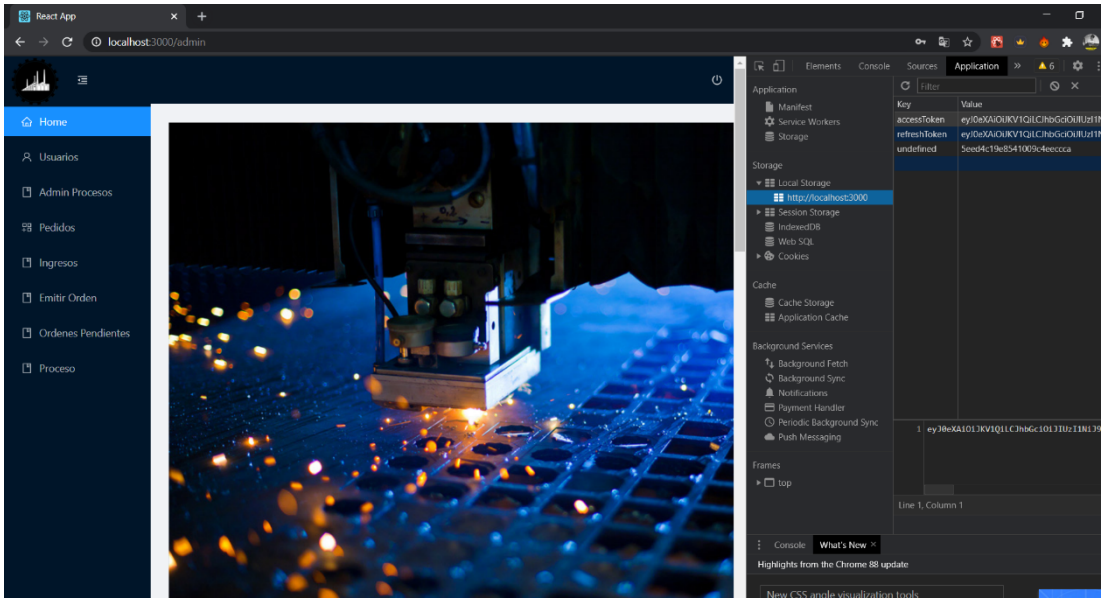


Figura 45 : Menú Principal, creación de Token

Una de las características que definen que una UI es sustentable, es la ergonomía, en otras palabras, que las ventanas a presentar no afecten la parte visual y las funciones sean de fácil acceso. Los modales son parte de esta característica, la Figura 40 refleja un modal de edición de usuario, centralizado y acorde la UI.

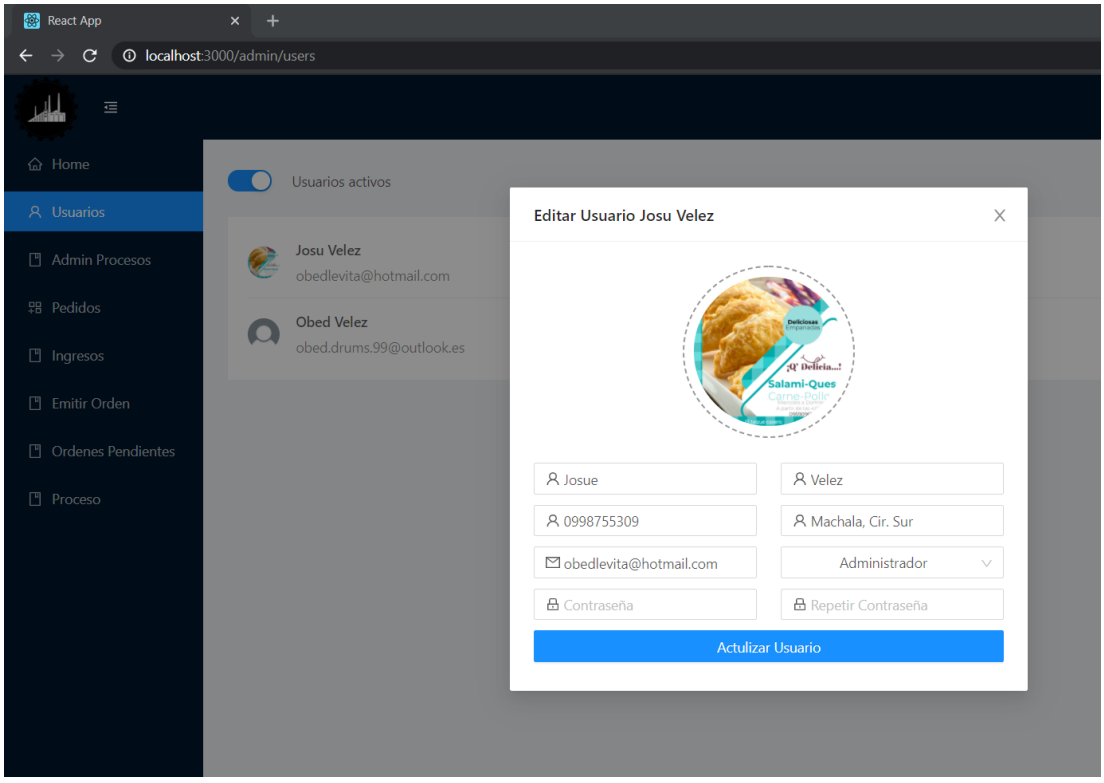


Figura 46 Usuarios, actualización de datos

El control de CRUD dentro de la aplicación esta fundamentado por eventos, dado que, si el resultado es favorable o no, se mostrará una alerta en la parte superior como lo muestra la Figura 41, 42 y 43.

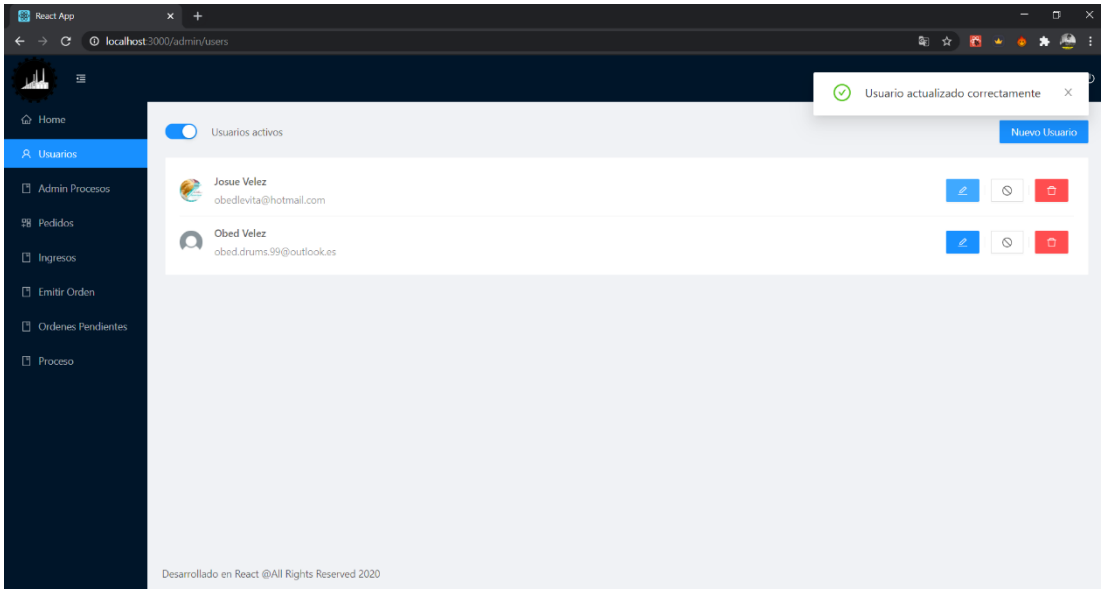


Figura 47 : Usuarios, actualización correcta

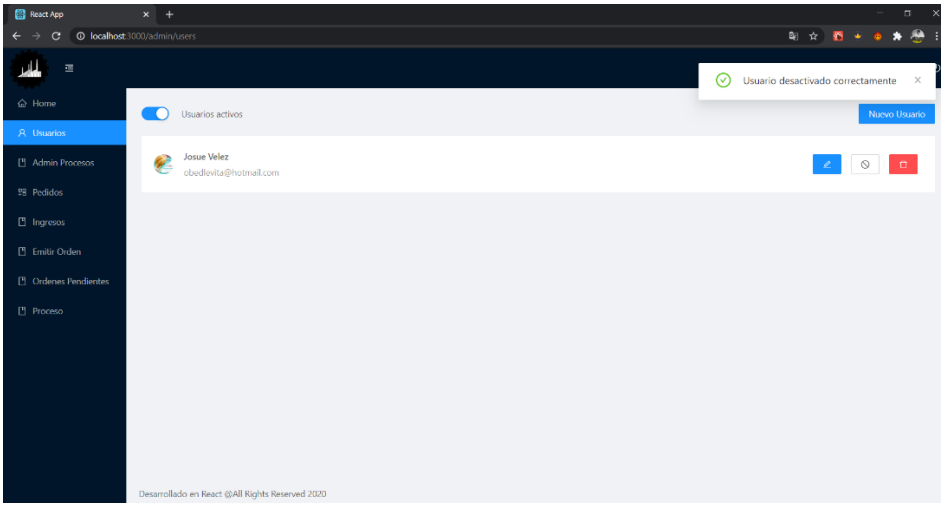


Figura 48 : Usuarios, desactivación de usuario

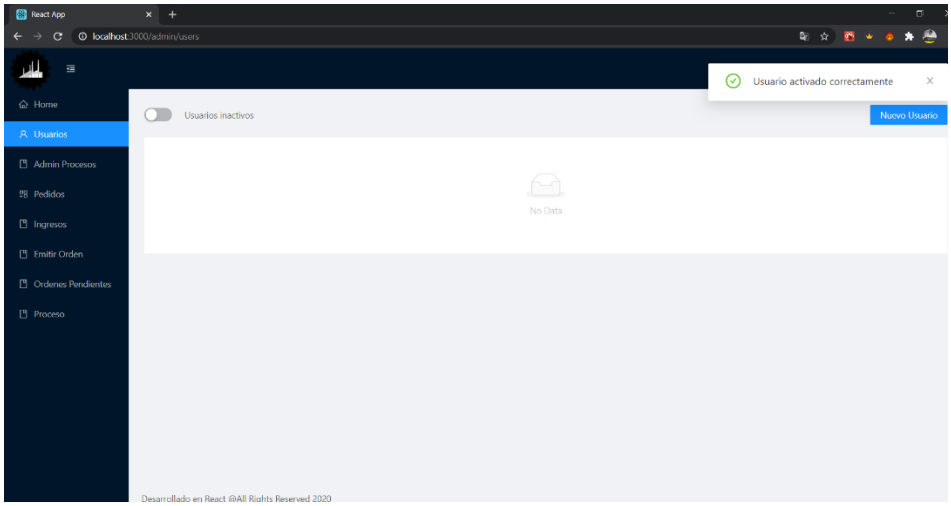


Figura 49 : Usuarios, activación de usuario

Al analizar con anterioridad que los procesos en cada micro empresa van a ser distintos dependiendo de la necesidad que se presente, no se puede globalizar el control de los mismos. Para ello se presenta un administrador de procesos (Figura 44), un CRUD para que el usuario pueda gestionar los procedimientos conforme a sus requerimientos.

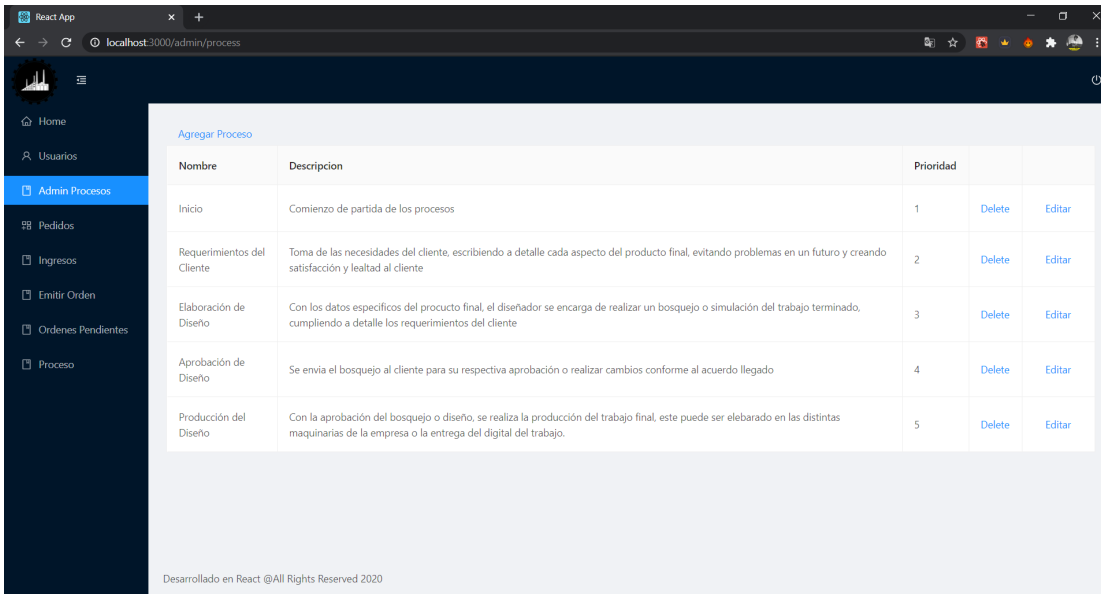


Figura 50 : Administración de Procesos

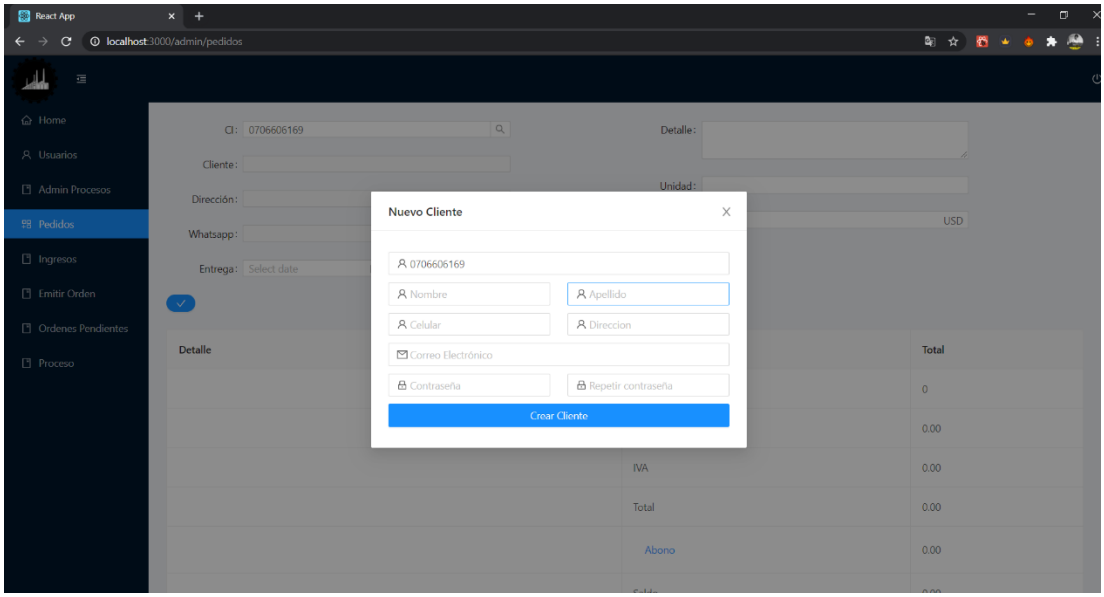


Figura 51: Pedido, creación de un nuevo usuario

En la Figura 45 se aprecia un modal que se ha activado por medio de un useState, a causa de que la cédula de identidad de ciudadanía ingresada no se ha encontrado en la base de datos, permitiendo al usuario crear un nuevo cliente sin interrumpir su proceso de ingreso de orden.

Estos procedimientos crean satisfacción al consumidor por cumplir estándares de calidad ergonómicos y de funcionamiento, como se muestra en la Figura 46, evitando que el usuario interactúe demasiado con la interfaz gráfica. Mientras menos acciones realiza el usuario, menos errores puede presentar el sistema.

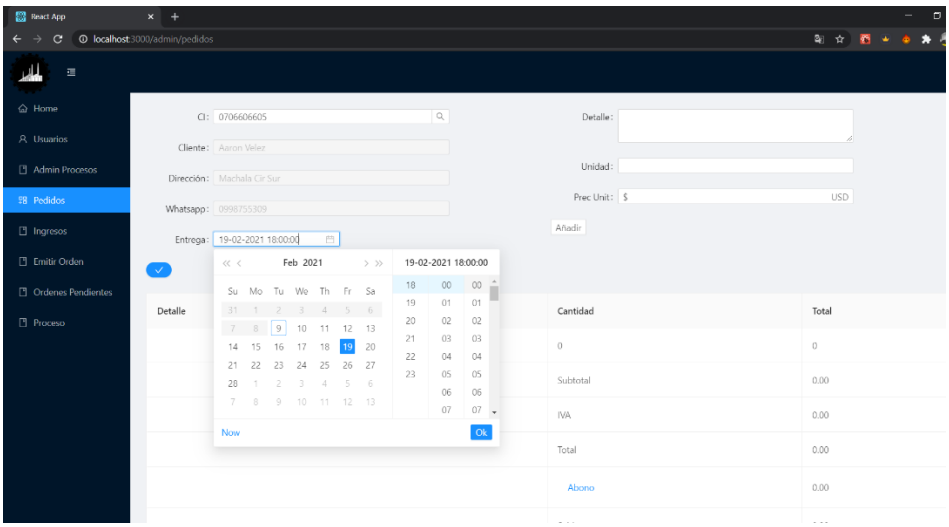


Figura 52 :Pedidos, ingreso de la orden y control de fecha hora

Al generar la orden (Figura 47), se presenta un modelo de compras, desglosando un subtotal, Iva, Total y descuento de ser el caso, el usuario procede a ingresar el abono o pago total de la orden, presentando un modal con la forma de pago (Figura 48), para su posterior clasificación.

El uso de modales gestiona una mejor forma de vista y evita el exceso de componentes, enfatizando la reutilización y refactorización de código.

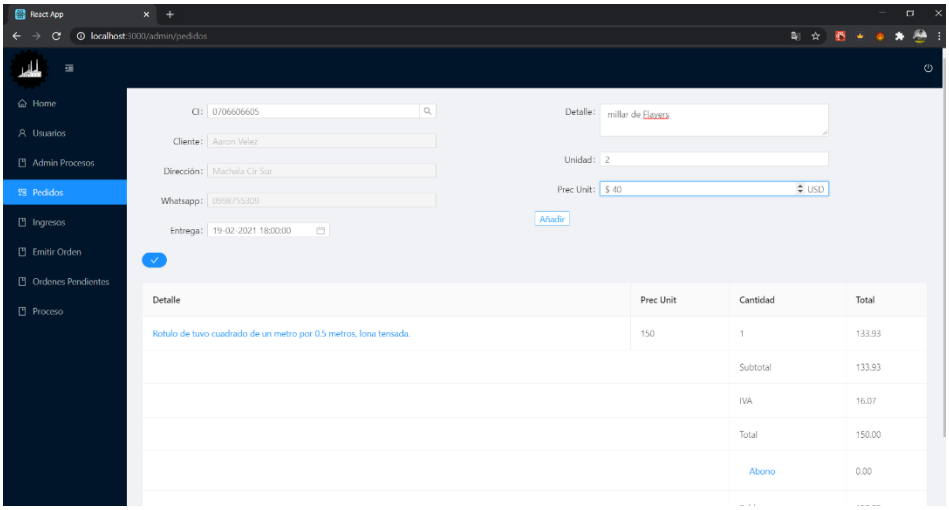


Figura 53 : Pedido, ingreso detalle

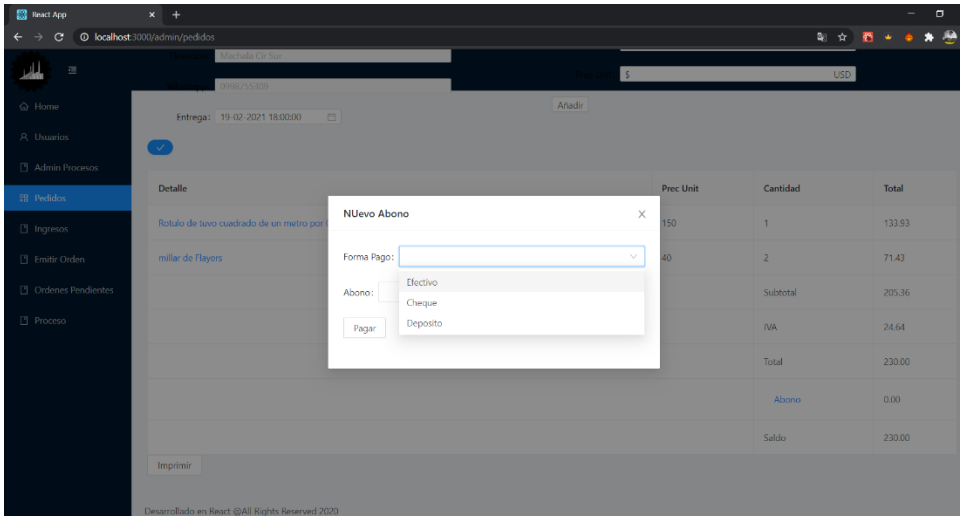


Figura 54 : Pedido, ingreso abono

- **Informes**

Los informes son importantes dentro del control de procesos, sobre todo verificar los ingresos diarios o por una fecha en específico (Figura 49). Sin embargo, las interacciones resaltan la calidad del documento, la Figura 50, se visualiza los ingresos parametrizados por efectivo, de igual manera se da la opción de filtro para los otros tipos de pago.

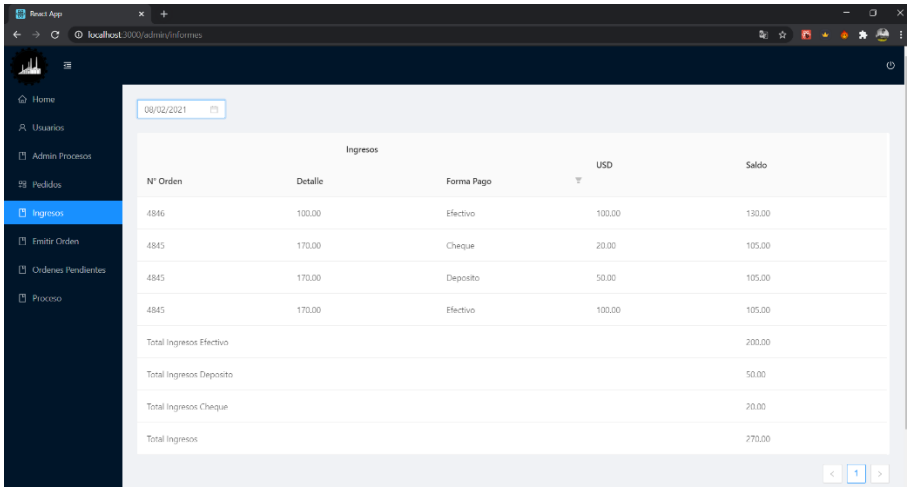


Figura 55 : Ingresos, informe de ingresos por fecha

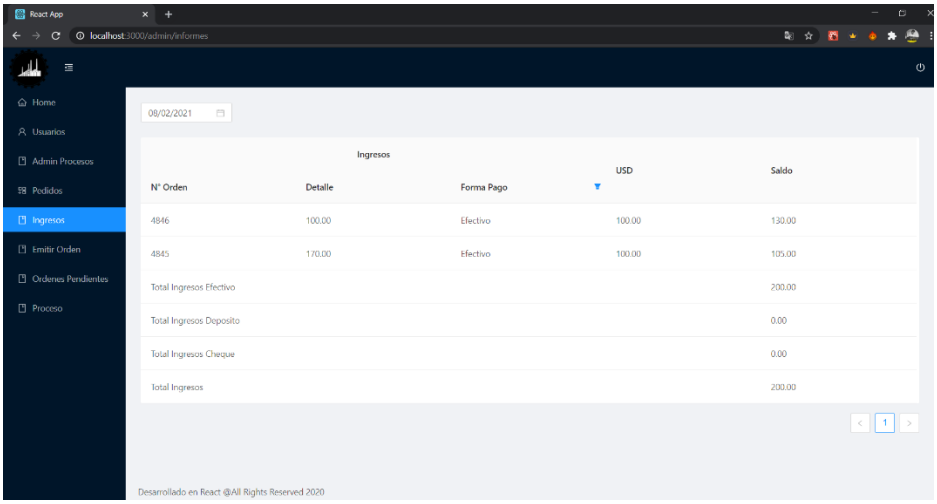


Figura 56 : Ingresos, informe parametrizado en Efectivo

En la observación directa, se analizó la necesidad de verificar los pedidos a entregar en la semana, la Figura 52 muestra la vista de ordenes diarias a primer loading y un campo parametrizable por fecha. Al igual que emitir la orden para impresión es fundamental dentro de los procesos (Figura 53).

Todos estos informes son básicos dentro de una micro empresa de servicios de diseño y corte láser. Teniendo un control de ingresos brutos, ordenes pendientes e impresión de la misma.

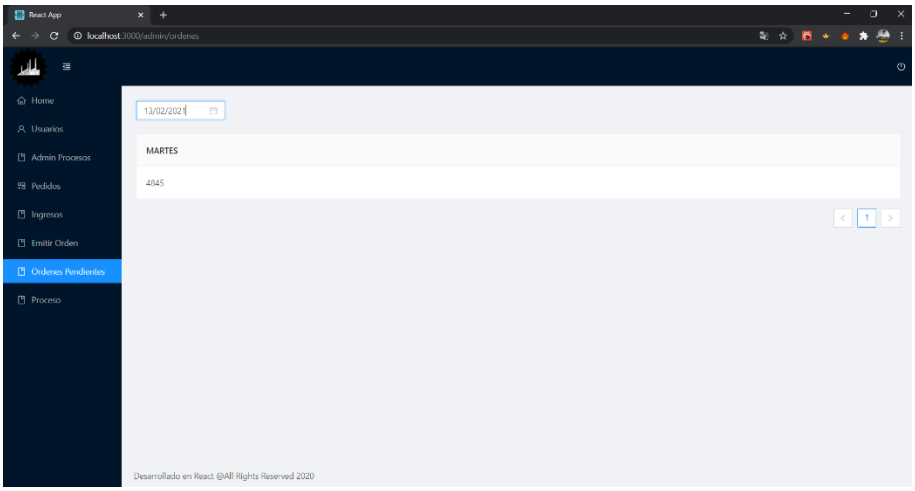


Figura 57 : Informe, Ordenes Pendientes

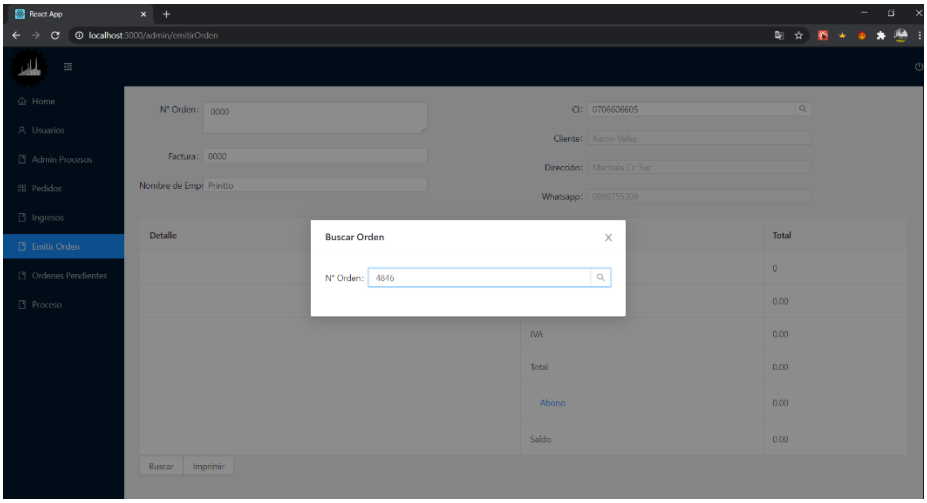


Figura 58 : Informe, Emitir Orden

El conocer una orden pendiente solo es el primer paso, para la gestión de procesos de elaboración del producto, la ventana Procesos (Figura 54), donde refleja las ordenes pendientes y el proceso actual en el que se encuentra.

El usuario puede tener control total del producto, en que punto de la producción se encuentra, analizar el tráfico de elaboración y en base a eso tomar mejores decisiones para la micro empresa.

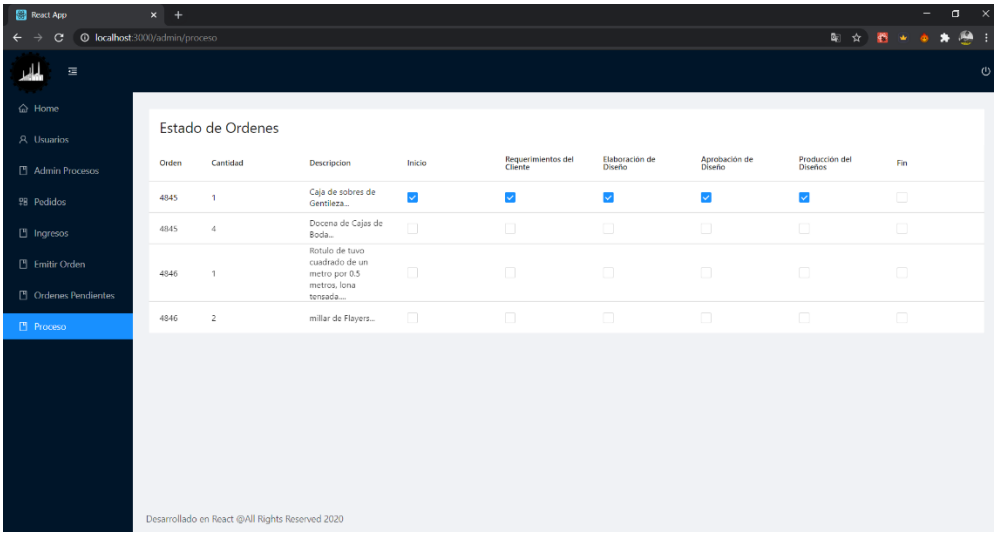


Figura 59 : Manejo de Producción

- **Móvil**

El diseño móvil es similar al administrador web, dando facilidad al usuario tener acceso desde su celular al sistema de control de procesos. (Figura 55), gestionando de igual manera todas las funciones como el ingreso de procesos (Figura 56)

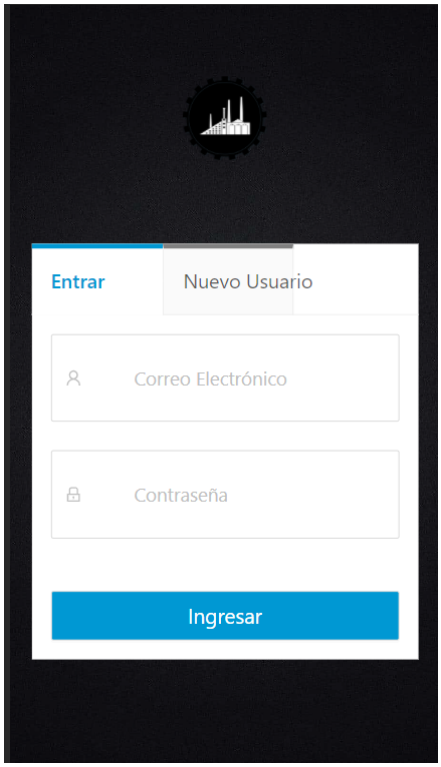


Figura 60 : Login en aplicativo móvil.

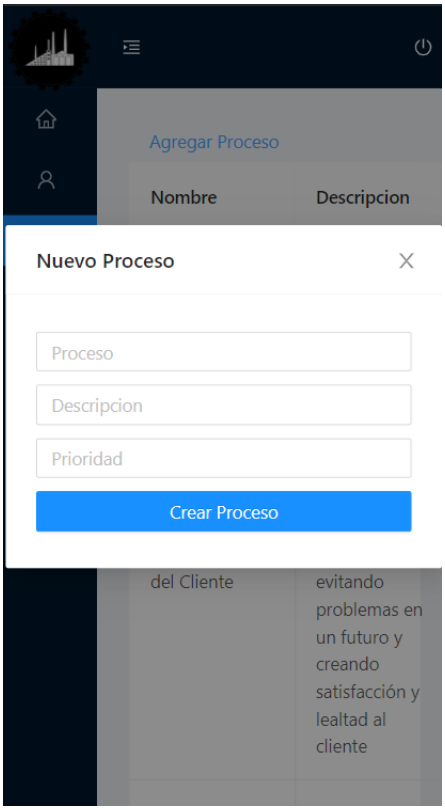


Figura 61: Administración de Procesos en aplicativo móvil.

- **Retroalimentación**

Para mayor conocimiento de la arquitectura a nivel de programación y visualizaciones revisar los Anexos.

CAPÍTULO IV:

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- El desarrollar un sistema generalizando procesos o limitando el control de operaciones, no es funcional para una microempresa que ofrece servicios de publicidad y corte láser, es lo que impulso desplegar el control CRUD de los procesos para la elaboración del servicio o producto que dicha organización pueda tener, personalizando la gestión de los mismos. La gestión de este modulo puede llegar a ser atractiva en el mercado de software, otorgándole una característica significativa ante la toma de una decisión.
- MERN es una de las arquitecturas Full Stack innovadoras, la comunicación que existe entre la base de datos, con los servicios y el cliente se optimizan a motivo de hacer interconexiones en frio y caliente, en otras palabras, se crea una puerta general (Comunicación en Frio) y dentro de ella existen varios compartimentos (Comunicación en Caliente) que, dependiendo de la ruta, se acceden a los micro servicios. El consumo de los servicios es una de las características del código limpio, pero no es la única, la creación de estos debe estar organizado y fragmentado, llegando a un punto de poder reutilizar código. El trabajo investigativo, cuenta con un esquema de código, creando distintas capas, separando funciones demasiado largas, combinando funciones callback con funciones de flecha o respuestas then, evitando el callbak hell. En el Back se creo 5 niveles de comunicación, un archivo de variables globales y encriptadas, un archivo principal de configuración y ejecución el cual hace un llamado a todos los elementos de la carpeta rutas, cada uno de estos elementos de comunican con los middlewares y controladores, terminando en la capa de modelos o clases de nuestra base de datos.
- La reutilización de código es uno de los fuertes de este trabajo, generalizando rutas, servicios y funciones, sin embargo, esta característica se presenta de gran manera en el cliente, dado que al utilizar el framework React y su función para móvil, React Native, el compartimiento de funciones entre dichas herramientas

es muy frecuente, especialmente en las funciones de llamado a los servicios REST.

- Con todos los beneficios que presenta tal arquitectura, no se puede descartar una de las más importantes, la seguridad de los datos. El aumento de delincuencia cibernética es indiscutible, con gran frecuencia se analizan casos de robo de datos, o la manipulación de los mismos, esto se debe al desconocimiento de los usuarios al manejar una herramienta y las vulnerabilidades que puede tener un software. Un sistema se puede considerar óptimamente funcional cuando el usuario posee un porcentaje mínimo de interactividad con el programa, lo que se ve resuelto con disparadores y utilizar las herramientas adecuadas, sin embargo, la seguridad debe estar implícito en el desarrollo, siendo el punto de partida para la generación de tokens para acceder a los servicios y la encriptación de datos delicados, como lo son las contraseñas, a su vez, al realizar el manejo de una base de datos No SQL, evitamos ataques de inject SQL, y al necesitar autorización y autenticación para acceder a las rutas que contienen datos vulnerables o funciones capaces de dañar el sistema, los ciberataques se ven frenados.

4.1. Recomendaciones

- Al tener un sistema escalable, el software puede crecer, por lo que se recomendaría que conforme los proceso crezcan, se analicen nuevos módulos a implementar. Adicional se realice un estudio de la gestión de procesos para una microempresa de servicios de publicidad y corte láser, para así poder generalizar los procesos y sobretodo, tener una base firme para el crecimiento organizacional y poder ver resultados más rápidos, logrando una gestión limpia y con un gran flujo.

- Las tecnologías se van innovando y fortaleciendo conforme a las nuevas necesidades que se presentan, la arquitectura usada en este trabajo puede mejorar usando micro servicios especiales o específicos, en otras palabras, el uso de contenedores. Estos permiten separar cada funcionalidad de la arquitectura y optimizar el uso de recursos, en este caso se recomendaría, usar un contenedor para alojar la base de datos, otro para los servicios y otro para

el cliente, cada docker o kubernet puede ser configurado conforme a las necesidades de cada función, donde los recursos se concentran en un solo objetivo o proceso.

- La refactorización y reutilización comúnmente van de la mano, y a pesar de que el trabajo investigativo cuenta con ambas, recomendaría un análisis de código por un programador externo, de allí el termino programar en pares, donde mientras el uno codifica el otro analiza o testea la codificación, puntualizando los temas a mejorar.

- La seguridad es uno de los temas que se mantendrá en vigencia, cada vez que se presenta la solución ante una vulnerabilidad, de un momento al otro se refleja otra. Mongo es una base de datos que no ha mostrado muchas vulnerabilidades, dado que al igual que los sistemas operativos de código abiertos y libres, no son muy comunes. Esta premisa no impide que sean atacados, por lo que se recomendaría una asesoría al usuario sobre el manejo de la plataforma y concentrarse en todas las vulnerabilidades posibles, controlando el ingreso de datos, la salida de los mismos, el llamado a los servicios, las comunicaciones entre capas, entre muchas mas.

BIBLIOGRAFIA

[1] D. Vohra, «With Docker, CoreOS Linux, and Other Platforms,» de *Kubernetes Management Design Patterns*, White Rock, British Columbia, 2017, p. 399.

[2] ticPortal, «IaaS (Infraestructura como Servicio),» 12 06 2018. [En línea]. Available: https://www.ticportal.es/glosario-tic/iaas-infraestructura-servicio. [Último acceso: 14 11 2019].

[3] M. Attaran, «Cloud Computing Technology: Leveraging the Power of the Cloud Computing Technology: Leveraging the Power of the Internet to Improve Business Performance Internet,» *Journal of International Technology and Information Management,* vol. 26, nº 6, p. 27, 2017.

[4] Enrique Castro-Leon, Robert Harmon, «Understanding the Service,» de *Cloud as a Service*, Portland, Oregon, USA, 2017, p. 315.

[5] S. V. y. G. Sachin, Basic NoSQL injection analysis and detection on MongoDB, Palwal: IEEE, 2018.

[6] H. Brito, Santos, Gomes y Bernardino, Análise de aprendizagem de frameworks móvies JavaScript, Coimbra: Iberian Conference on Information Systems and Technologies, 2019.

[7] Kewal y Harsh, Analysis of Cross-Platform Mobile App Development Tools, Mumbai: IEEE, 2019.

[8] Jitender y Varsha, Security analysis of unstructured dara in NoSQL MongoDB Database, Jaypee: IEEE, 2017.

[9] Brito, Santos, Bernardino y Gomes, JavaScript em aplicações móveis: React Native vs Ionic vs NativeScript vs desenvolvimiento nativo, Corimbra: 14th Iberian Conference on Information Systems and Technologies (CISTI), 2019.

[10] Brito, Santos, Bernardino y Gomes, Desenvolvimento móvel em Swift, Java e React Native: i,a avaliação experimental em áudioguias, Coimbra: 14th Iberian Conference on Information Systems and Technologies (CISTI), 2019.

[11] L. D, Testing Spatial Data Deliverance in SQL and NoSQL Database using NodeJS Fullstack Web App, Yogyakarta: IEEE, 2018.

[12] T. P, Web Service Based Food Additive Inventory Management with Forecasting System, Bankok: IEEE, 2018.

[13] W. S y C. Z. M, Docker-based Web Server Instructional System, Beijing: IEEE, 2019.

[14] M. P, «Revista Líderes,» [En línea]. Available: https://www.revistalideres.ec/ideres/mundo-utiliza-apps.html. [Último acceso: 2020 Mayo 1].

[15] CEUPE, «Ceupe,» [En línea]. Available: https://www.ceupe.com/blog/el-mundo-de-lasaplicaciones.html.. [Último acceso: 2020 Mayo 1].

[16] C. P, «perecondom,» [En línea]. Available: http://www.perecondom.com/2017/08/28/la-curva-s-una-tecnologia/. [Último acceso: 2020 Mayo 1].

[17] C. D, Refactorización de código y consideraciones sobre la complejidad ciclomática, Serie Documentos de Trabajo, 2016.

[18] JESUITES EDUCACIO, «X Xtended Studies,» Julio 16 2018. [En línea]. Available: https://fp.uoc.fje.edu/blog/que-es-un-full-stack-developer/. [Último acceso: 2020 Mayo 5].

[19] MongoDb, «MongoDB,» [En línea]. Available: https://www.mongodb.com/es/what-is-mongodb. [Último acceso: 2020 Mayo 5].

[20] Express, «Express Js,» [En línea]. Available: https://expressjs.com/es/. [Último acceso: 2020 Mayo 5].

[21] Node, «Node,» [En línea]. Available: https://nodejs.dev/. [Último acceso: 2020 Mayo 5].

[22] React, «React Js,» [En línea]. Available: https://es.reactjs.org/docs/getting-started.html. [Último acceso: 2020 Mayo 5].

[23] V. Subramanian, Pro MERN Stack, Apress, 2017.

[24] M. Á y R. L. Valenciano, «Lenguajes de programación COBOL y PL/I, historia y características principales,» [En línea]. Available: http://www.di-mare.com/adolfo/cursos/2007-2/pp-cobol-pl1.pdf. [Último acceso: 2020 Junio 25].

[25] Tech-Blog, «GB Advisor,» 28 Febrero 2019. [En línea]. Available: https://www.gb-advisors.com/es/softwarepara-medianas-empresas/. [Último acceso: 5 Agosto 2020].

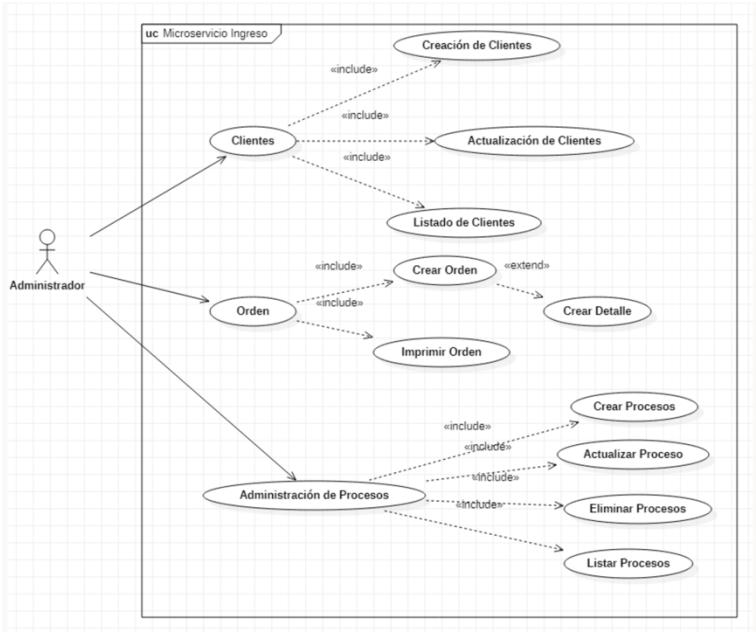
[26] M. C, Desarrollo de aplicaciones web, Catalunya: UOC, 2004.

[27] I. Y. A, P. M, Ruiz y I. J. Z. A, «Condicionantes económicos de la adopción de una innovación por parte del consumidor: análisis de la compra de servicios online,» INNOVAR Revista de CIencias Administrativas y Sociales, 2010, pp. 173-186.

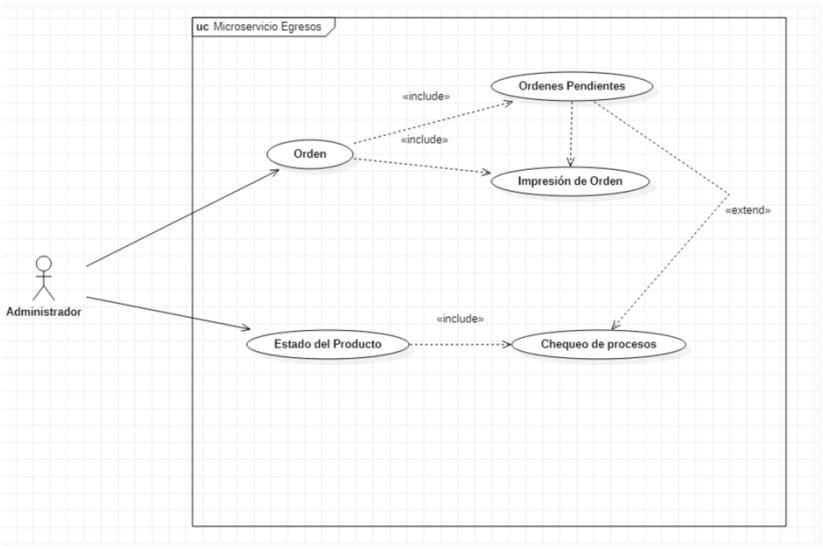
ANEXOS

Diagrama de Casos de Uso.

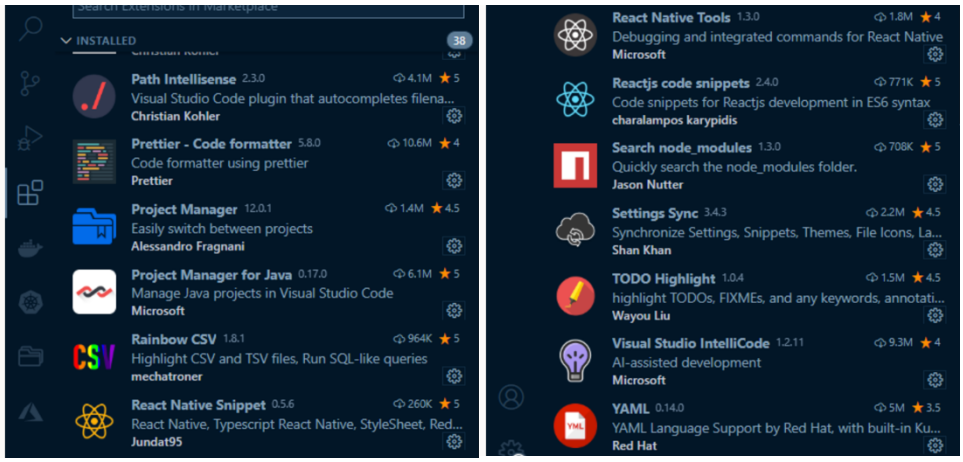
Interacción del administrador de micro servicios en ingresos de clientes, ordenes y procesos.



Interacción del administrador con la salida de datos.

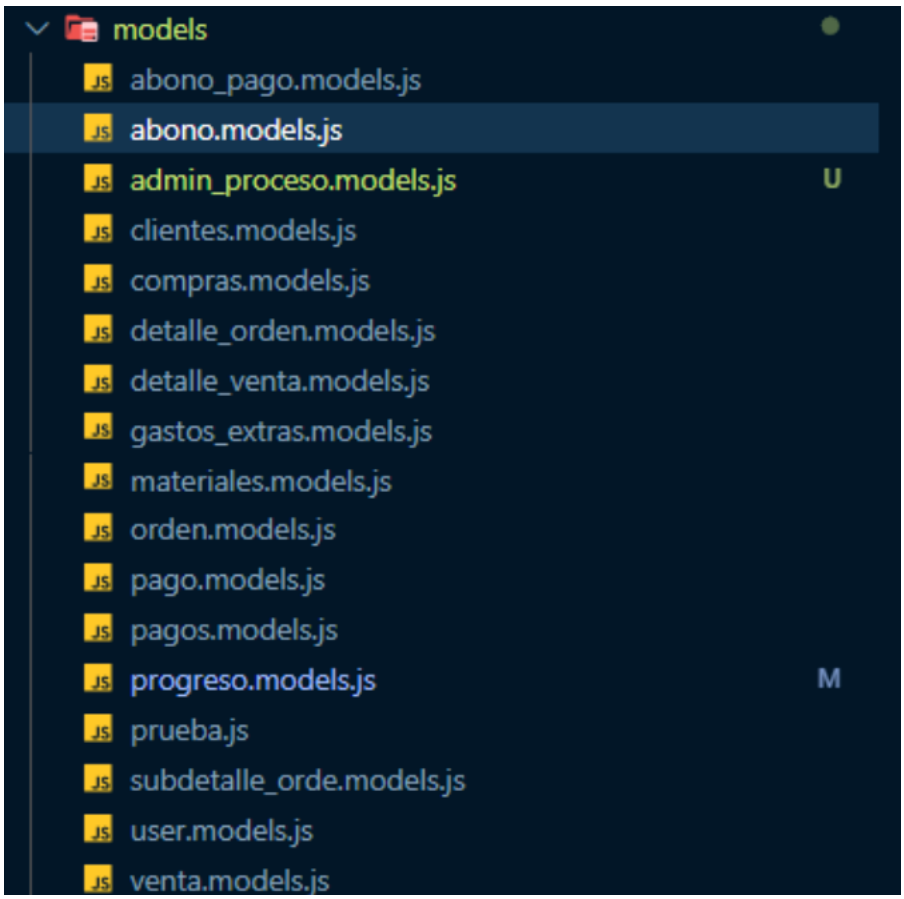


Librerías necesarias en visual studio code.

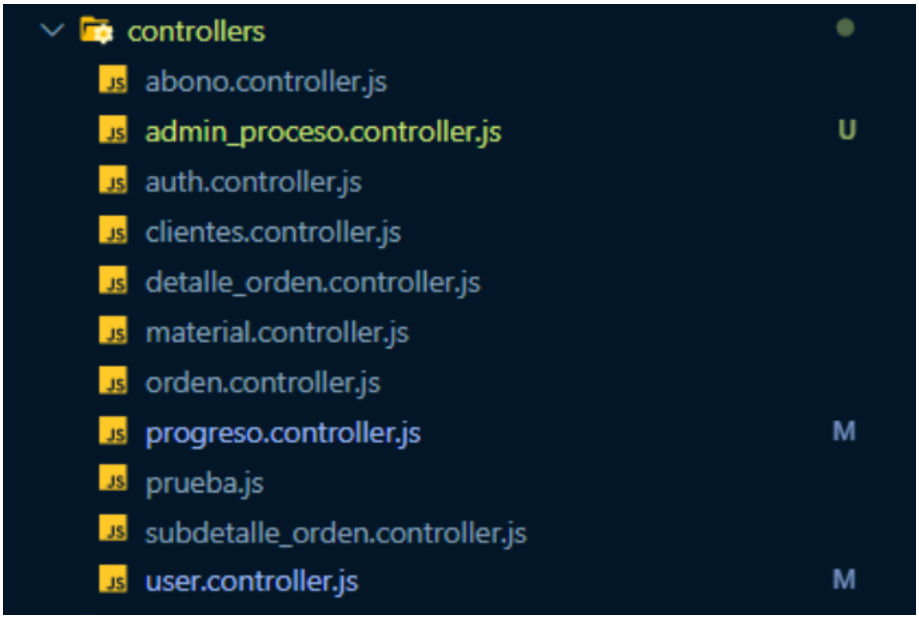


Estructura de carpetas:

Capa de nivel 0, donde se modela los documentos de la base de datos, creando los esquemas, reglas, validaciones, auto numéricos, entre otros.



La capa 1 cuenta con los controladores y los middlewares, realizando las funciones, promesas y call backs, gestionando el CRUD de la data.



Las rutas se encuentran en la capa 2, el módulo express permite crear los lineamientos de comunicación en frío y caliente, el uso de middlewares para la autenticación, autorización y la gestión de imágenes en el servidor.