UNIVERSIDAD HISPANOAMERICANA

ESCUELA DE INFORMÁTICA

PROYECTO DE TESINA PARA OBTAR POR EL GRADO DE

BACHILLER EN INGENIERIA INFORMATICA

TÍTULO DEL PROYECTO:

Desarrollo de Aplicación web para el Registro de Datos en el Ámbito Empresarial: Herramienta Personalizada con Entrega de Códigos Fuente

Sustentante:

César Enrique Jarquín Méndez

Tutor: Marco Sacasa Soto

Junio, 2024

Tabla de Contenido

Tabla de Ilustraciones

Declaración Jurada

DECLARACIÓN JURADA

Yo César Enrique Jarquín Méndez, mayor de edad, portador de la cedula de identidad número 604140718, egresado de la carrera de ingeniería informática de la Universidad Hispanoamericana, hago constar por medio de este acto y debidamente apercibido y entendido de las penas y consecuencias con las que se castiga en el código penal el delito de perjurio, ante quienes se constituyen en el tribunal examinador de mi trabajo de tesina para optar por el titulo de bachiller en ingeniería informática, juro solemnemente que mi trabajo de investigación titulado: Desarrollo de Aplicación web para el Registro de Datos en el Ámbito Empresarial: Herramienta Personalizada con Entrega de Códigos Fuente, es una obra original que ha respetado todo lo preceptuado por las Leyes Penales, así como la Ley de Derecho de Autor y Derecho Conexos número 6683 del 14 de octubre de 1982 y sus reformas, publicada en la Gaceta número 226 del 25 de noviembre de 1982; incluyendo el numeral 70 de dicha ley que advierte; articulo 70. Es permitido citar a un autor, transcribiendo los pasajes pertinentes siempre que éstos no sean tantos y seguidos, que puedan considerarse como una producción simulada y sustancial, que redunde en perjuicio del autor de la obra original. Asimismo, quedo advertido que la Universidad se reserva el derecho de protocolizar este documento ante Notario Público.

En fe de lo anterior, firmo en la ciudad de Puntarenas a los 9 días del mes de julio del año dos mil veinticuatro.

Firma del Estudiante

Cedula: 6 0414 0718

Carta del Tutor

CARTA DEL TUTOR

Puntarenas, 28 de junio de 2024

Esteban José González Vargas
Sub Director Ingeniería Informática
Universidad Hispanoamericana
Sede Llorente

Estimado

El estudiante **César Enrique Jarquín Méndez**, cédula de identidad número **604140718** me ha presentado, para efectos de revisión y aprobación, el trabajo de investigación denominado **"Desarrollo de Aplicación web para el Registro de Datos en el Ámbito Empresarial: Herramienta Personalizada con Entrega de Códigos Fuente"**, el cual ha elaborado para optar por el grado académico de Bachiller en Ingeniería Informática.

En mi calidad de tutor, he verificado que se han hecho las correcciones indicadas durante el proceso de tutoría y he evaluado los aspectos relativos a la elaboración del problema, objetivos, justificación; antecedentes, marco teórico, marco metodológico, tabulación, análisis de datos; conclusiones y recomendaciones.

De los resultados obtenidos por el postulante, se obtiene la siguiente calificación:

a) Original del tema	10%	10%
b) Cumplimiento de entrega de avances	20%	20%
c) Coherencia entre los objetivos, los instrumentos aplicados y los resultados de la investigación	30%	30%
d) Relevancia de las conclusiones y recomendaciones	20%	20%
e) Calidad, detalle del marco teórico	20%	20%
TOTAL		**100%**

En virtud de la calificación obtenida, se avala el traslado al proceso de lectura.

Atentamente,

MARCO VINICIO SACASA SOTO (FIRMA)

Marco Sacasa Soto

Cédula 602820120

Carta del Lector

CARTA DE LECTOR

San José, 27 de septiembre de 2024.

Universidad Hispanoamericana
Sede Llorente
Carrera Ingeniería Informática

Estimada señora

El estudiante **César Enrique Jarquín Méndez,** cédula de identidad **6-0414-0718,** me ha presentado para efectos de revisión y aprobación, el trabajo de investigación denominado **"Desarrollo de Aplicación web para el Registro de Datos en el Ámbito Empresarial: Herramienta Personalizada con Entrega de Códigos Fuente"** el cual ha elaborado para obtener su grado de **Bachillerato**.

He revisado y he hecho las observaciones relativas al contenido analizado, particularmente lo relativo a la coherencia entre el marco teórico y análisis de datos, la consistencia de los datos recopilados y la coherencia entre éstos y las conclusiones; asimismo, la aplicabilidad y originalidad de las recomendaciones, en términos de aporte de la investigación. He verificado que se han hecho las modificaciones correspondientes a las observaciones indicadas.

Por consiguiente, este trabajo cuenta con mi aval para ser presentado en la defensa pública.

Atte.

ESTEBAN JOSE GONZALEZ VARGAS (FIRMA)

Firmado digitalmente por ESTEBAN JOSE GONZALEZ VARGAS (FIRMA)
Fecha: 2024.09.27 19:15:41 -06'00'

Esteban José González Vargas
1-1251-0724
Código 5563

Carta de Autorización

UNIVERSIDAD HISPANOAMERICANA
CENTRO DE INFORMACION TECNOLOGICO (CENIT)
CARTA DE AUTORIZACIÓN DE LOS AUTORES PARA LA CONSULTA, LA REPRODUCCION PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DE LOS TRABAJOS FINALES DE GRADUACION

Puntarenas, 05 de octubre del 2024

Señores:
Universidad Hispanoamericana
Centro de Información Tecnológico (CENIT)

Estimados Señores:

El suscrito César Enrique Jarquín Méndez con número de identificación 6 0414-0718 autor del trabajo de graduación titulado "Desarrollo de Aplicación web para el Registro de Datos en el Ámbito Empresarial: Herramienta Personalizada con Entrega de Códigos Fuente" presentado y aprobado en el año 2024 como requisito para optar por el título de Bachillerato en ingeniería informática; Si autorizo al Centro de Información Tecnológico (CENIT) para que con fines académicos, muestre a la comunidad universitaria la producción intelectual contenida en este documento.

De conformidad con lo establecido en la Ley sobre Derechos de Autor y Derechos Conexos N° 6683, Asamblea Legislativa de la República de Costa Rica.

Cordialmente,

6 0414 0718

Firma y Documento de Identidad

Dedicatoria

Dedicación a mis familiares por todo el apoyo que he recibido durante este proceso de realización de proyecto de tesina para optar por el grado de bachillerato en ingeniería en sistemas, ya que me hacen recordar todo lo que nos propongamos lo podemos realizar si estamos unidos como familia.

1 Capítulo I Planteamiento del tema

1.1 Antecedentes y justificación

1.1.1 *Marco de referencia empresarial y contextual*

Acqua digital Marketing es una empresa fundada en el año 2023, con el fin de ofrecer servicios de marketing digital bajo una estrategia empresarial centrada en principio morales y buenas conductas y así promover una comunicación afectiva entres sus colaboradores y sus respectivos clientes. La misión de Acqua Digital Marketing es proporcionar servicios de marketing digital innovadoras y adaptándose continuamente a las necesidades del mercado para contribuir al éxito de sus clientes. En términos de su visión es ser reconocidas como líder en el sector, destacando por la calidad y creatividad de sus servicios.

Las oficinas de Acqua Digital Marketing están ubicadas en el campus de Terra mall en tres ríos, San José, dedicada a dar servicios de marketing digital, la empresa se encuentra en sus primeras etapas de adaptación al mercado y búsqueda constante de métodos innovadores para mejorar su imagen y recopilar información valiosa sobre sus clientes y posibles nuevos clientes.

Están ubicados en el mercado de marketing digital y es un sector en constante evolución la empresa se adapta a las tendencias actuales y busca alinearse con las mejores prácticas del sector. Este enfoque permite a Acqua digital marketing justificar su estrategia demostrando que sus acciones son decisiones alineadas con las demandas y oportunidades del mercado en el que opera.

1.1.2 *Justificación*

La idea de implementar una aplicación web para un pequeño gestor de datos en Acqua Digital se puede justificar de varias maneras, lo que evidencia la necesidad de llevar a cabo este proyecto.

En primer lugar, se identifica en la empresa una necesidad de optimizar la gestión de la información interna, abarcando áreas críticas como la administración de datos generales de la empresa, la gestión de activos, el control de planilla, así como pagos y cobros. Al perfeccionar estos procesos, Acqua Digital podrá ofrecer un servicio de mayor calidad a sus clientes, fortaleciendo su posición en el mercado.

Además, Al implementar un gestor de datos, se garantiza una gestión de datos eficiente y una mayor transparencia en las operaciones legales financieras. Este sistema garantiza cumplir con estándares legales y también proteger la información confidencial de la empresa.

Otro beneficio significativo es que mejorara la eficiencia operativa. un gestor de datos es clave para esto, ya que facilita el acceso fácil y rápido a la información y así poder tomar decisiones más acertadas, mejorando la competitividad de la empresa en el mercado de marketing digital.

Por último, El gestor de datos será un carácter estratégico para Acqua Digital. La disponibilidad de la información crítica facilita la toma de decisiones a largo plazo especialmente en lo que respecta el fortalecimiento de relaciones con sus clientes y colaboradores.

En conclusión, El proyecto tendrá un impacto económico importante para la empresa, al mejorar la eficiencia operativa, al manejar la información de planillas, pagos y cobros que se realizan en la empresa se reducirán los errores en las tomas de decisiones contribuyendo en la optimización de los recursos y en un impacto positivo en la rentabilidad de la empresa.

1.2 Definición del problema

1.2.1 *Problemática*

En la era actual, donde los datos juegan un papel crucial en la toma de decisiones, la falta de resguardo de la información clave de la empresa puede llevar a ignorar aspectos críticos relacionados con la planilla, los clientes y la gestión de cobros y pagos.

Sin una solución, corren el riesgo de pasar por alto aspectos importantes en las operaciones del día a día que pueden afectar la eficiencia y puede afectar la capacidad para competir en el entorno empresarial digital.

1.2.2 *Problema general*

¿Como afecta la falta de herramientas integrales de gestión de datos y la ausencia de lineamientos importantes, para la sistematización de actividades en la empresa en el lado operativo y en la toma de decisiones efectivas en áreas claves como en la planilla y clientes?

¿Cómo afecta la falta de retención de datos importantes en la empresa en el lado operativo y en la toma de decisiones efectivas en áreas claves como planilla, clientes y pagos y cobros?

1.2.3 *Problemas específicos*

¿Como afecta la falta de datos en las tomas de decisiones de corto y largo plazo?

¿En que afecta la ausencia de una herramienta manejada por administradores para la interacción con clientes, la retención de los clientes y la identificación de futuros posibles negocios?

¿Cómo influye la falta de una herramienta para la obtención de datos sobre los colaboradores en la identificación de pagos de planilla?

1.3 Objetivo general y objetivos específicos

1.3.1 *Objetivo general*

Implementar un gestor de datos en Acqua Digital dentro de las próximas 20 semanas, con el fin de mejorar la gestión de la planilla, la interacción con los clientes y el seguimiento de cobros y pagos, promoviendo la toma de decisiones informada y fortaleciendo la competitividad de la empresa en el mercado.

1.3.2 *Objetivos específicos*

- Recopilar los requerimientos del sistema a través de reuniones con el representante de la empresa, analizando la información obtenida para garantizar su claridad y pertinencia.
- Diseñar el sistema y estructurar la base de datos, asegurando que se manejen correctamente los requerimientos establecidos
- Desarrollar tanto el BackEnd como el FrontEnd del sistema, integrando lógicamente los componentes y cumpliendo con los requerimientos especificados.
- Realizar pruebas exhaustivas para evaluar el correcto funcionamiento de la aplicación, identificando y corrigiendo posibles errores.
- Elaborar un manual de usuario y preparar los documentos necesarios para la entrega final del proyecto.

1.4 Alcance y limitaciones

1.4.1 *alcances*

- la obtención de requisitos será primero y se realizará una documentación de requisitos funcionales y no funcionales del sistema.

- lo segundo será el diseño del sistema por ende se realizará un desarrollo de diagrama de flujos, clases y el diseño de la base de datos con tablas relaciones y consultas.
- Lo siguiente será el desarrollo del software, se realizará la implementación del Backend y el desarrollo del Frontend con una interfaz de usuario intuitiva y responsiva
- Ejecución de las pruebas necesarios para identificar y corregir problemas en el código y en la interfaz de usuario.
- Se desarrollará un manual de usuario y los preparativos para la entrega final del proyecto.

1.4.2 *Limitaciones*

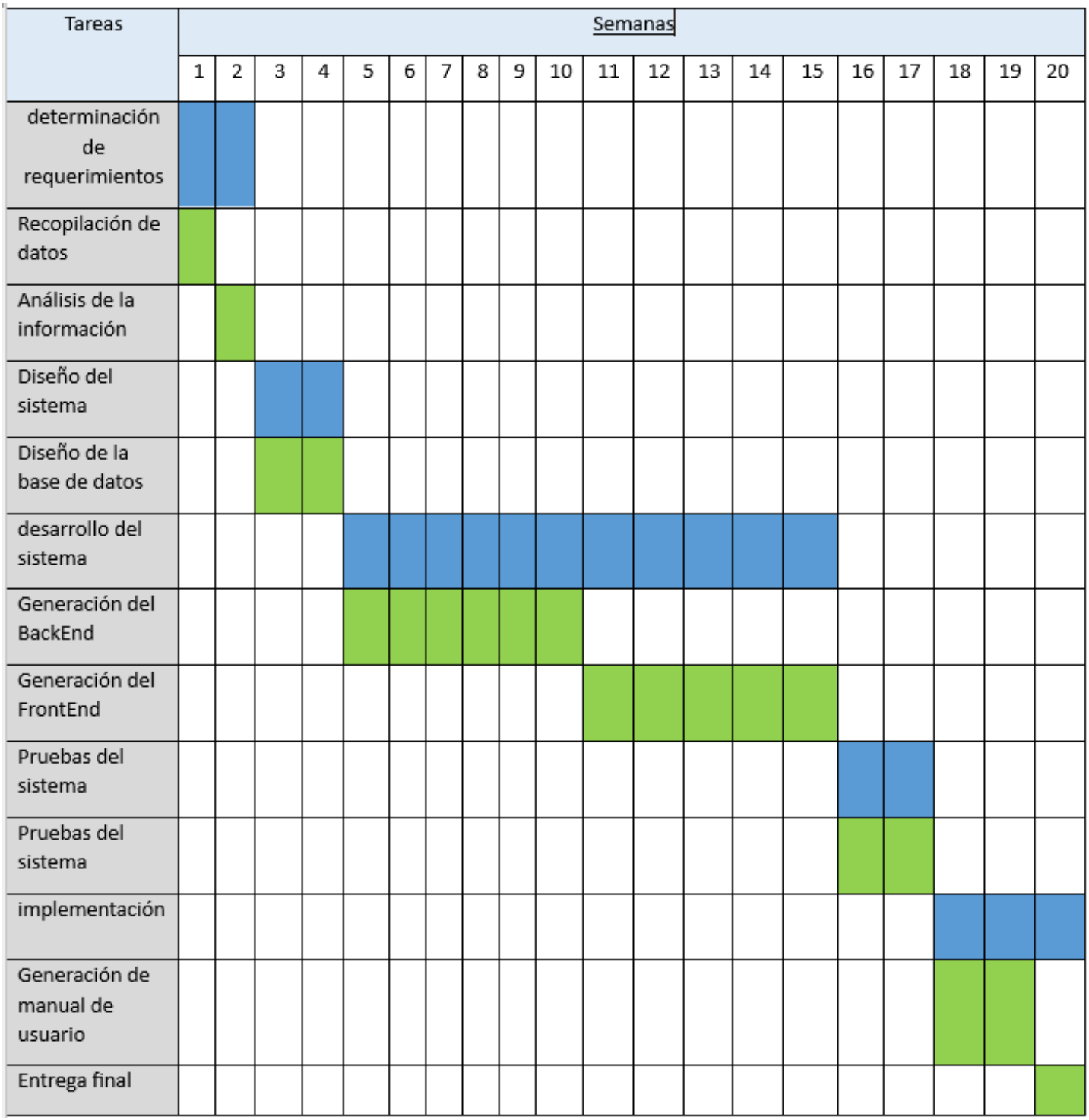
En el desarrollo del proyecto, se identifican algunas limitaciones que están presentes, no obstaculizan de manera significativa su ejecución y finalización.

Una de las limitaciones se relaciona con el proceso de recopilación de requerimientos dado de que las partes involucradas están en provincias diferentes al estar en provincias diferentes, se reconoce que pueden surgir desafíos en la comunicación, en lo cual podría incidir en la comprensión completa y precisa de los requerimientos por parte de todas las partes involucradas.

Adicionalmente, las tecnologías actualmente utilizadas por la empresa representan una limitación en el desarrollo del software. Esto podría implicar restricciones en las funcionalidades del sistema, así como las necesidades de realizar ajustes específicos en función de las herramientas y capacidades técnicas disponibles.

1.5 Cronograma

Se creo el siguiente cronograma para la realización del proyecto mediante tareas y entregables para tener un mejor control del tiempo.



2 Capitulo II Marco Teórico

La necesidad de contar con herramientas que faciliten la recopilación, almacenamiento y gestión de datos empresariales de manera efectiva ha llevado a la creciente demanda de aplicaciones web personalizadas que se adapten a las necesidades específicas de cada empresa. En este contexto, el presente proyecto tiene como objetivo desarrollar una aplicación web diseñada para el registro y gestión de datos en el ámbito empresarial, proporcionando una herramienta personalizada que permita a las empresas optimizar sus procesos internos y mejorar su eficiencia operativa.

El desarrollo de esta aplicación web identificara las necesidades del cliente hasta la entrega de una solución funcional y totalmente escalable. Se explorará el ciclo de vida del desarrollo de software y las mejores prácticas de entrega de proyecto. Se mencionará las tecnologías y herramientas utilizadas en el desarrollo web y las metodologías del desarrollo de software más adecuadas para el proyecto también se abordarán temas como la seguridad, escalabilidad y adaptabilidad en la aplicación.

2.1 Gestión de Datos Empresariales

En la actualidad las aplicaciones web son herramientas bastantes importantes en el ambiente empresarial el cual ayuda en la recolección de datos y el manejo de procesos internos de las empresas y así ayudar a tomar decisiones acertadas basadas en la información.

En la era digital actual, las aplicaciones web han adquirido una importancia significativa en el ámbito empresarial y en la sociedad en general "Según un artículo en el sitio web de Arbentia, desde la automatización de procesos hasta la toma de decisiones, estas aplicaciones tienen un impacto significativo en el éxito comercial actual " (Arbentia, 2023).

En el artículo de Arbentia, “Las aplicaciones empresariales son programas informáticos diseñados con el objetivo de ayudar a las empresas a realizar sus actividades comerciales y administrativas de manera más eficiente" (Arbentia, 2023) se menciona que el desarrollo de aplicaciones web no solo ha revolucionado la forma en que las empresas operan, sino que también ha democratizado el acceso a la tecnología, permitiendo que incluso las empresas más pequeñas aprovechen las herramientas digitales para mejorar su eficiencia y competitividad en el mercado.

En este contexto, el presente proyecto se centra en el desarrollo de una aplicación web diseñada específicamente para el registro y gestión de datos en el ámbito empresarial. La necesidad de una herramienta personalizada que se adapte a las necesidades específicas de cada empresa es evidente, ya que cada organización tiene sus propios procesos y flujos de trabajo que requieren soluciones ágiles y flexibles.

Esta aplicación web no solo facilitará la recopilación y el almacenamiento de datos empresariales, sino que también permitirá su análisis, seguimiento y colaboración en tiempo real entre los usuarios autorizados. Además, la entrega de los códigos fuente garantizará la transparencia y la posibilidad de realizar personalizaciones según las necesidades cambiantes del negocio.

2.2 Aplicaciones Empresariales

Al identificar las necesidades de la empresa se llega a la conclusión de integrar una aplicación web para la gestión de datos, ya que no cuentan con algún software que resguarde los datos para futuras decisiones en la empresa. En la página de Arbentia se menciona que “Las aplicaciones empresariales son programas informáticos diseñados con el objetivo de ayudar a las empresas a realizar sus actividades comerciales y administrativas de manera más eficiente”

(Arbentia, 2023). Por lo tanto, la integración de una aplicación web permitirá mejorar la eficiencia en las operaciones y facilitará la toma de decisiones estratégicas basadas en datos.

En la actualidad, existen varias aplicaciones para la gestión de datos, pero realizar una aplicación personalizada es muy valioso, ya que se diseñará al gusto del cliente y resolverá los requisitos específicos utilizando base de datos relacionales.

Al manejar mucha información de la empresa, lo más adecuado es utilizar una base de datos relacional, ya que se manejarán registros en la aplicación y la información estará relacionada y estructurada de la mejor manera.

2.3 Base de Datos Relacionales

Las bases de datos relacionales han sido durante mucho tiempo la opción preferida para almacenar y gestionar datos en entornos empresariales. Utilizan un modelo de datos tabular, donde la información se organiza en filas y columnas en tablas relacionadas entre sí. Este enfoque permite establecer relaciones complejas entre los datos, garantizando la integridad y consistencia de la información. Según IBM, "Una base de datos relacional organiza los datos en filas y columnas, que en conjunto forman una tabla. Los datos normalmente se estructuran en varias tablas, que se pueden unir a través de una clave principal o una clave externa." (IBM, 2022).

Con este modelado de datos, podemos registrar y consultar los datos de una manera agrupada lo que nos ayuda a realizar consultas más específicas en las aplicaciones. Según la página de IBM" El principal beneficio del enfoque de base de datos relacional es la capacidad de crear información significativa uniendo las tablas. Unir tablas le permite comprender las relaciones entre los datos o cómo se conectan las tablas" (IBM, 2022).

Es importante destacar su relevancia y utilidad en el desarrollo de aplicaciones empresariales. Las bases de datos relacionales, al organizar los datos en tablas conectadas entre sí. Ofrecen un marco estructurado que facilita el manejo eficiente de grandes volúmenes de información. Esta estructura no solo permite almacenar datos de manera organizada, sino que también posibilita realizar consultas complejas y obtener resultados precisos y rápidos.

Finalmente, el uso de bases de datos relacionales en aplicaciones empresariales también mejora la seguridad y el control de acceso. Al estructurar los datos en tablas, es posible implementar controles de acceso granular, asegurando que solo los usuarios autorizados pueden ver o modificar cierta información. Esto no solo permite proteger los datos sensibles de la empresa, sino que también garantiza el cumplimiento de normativas y estándares de seguridad.

2.3.1 *MySQL*

Cuando se implementa una base de datos, es fundamental tener o contar con un gestor de base de datos que permita crear, administrar y manipular información de manera eficiente. En este contexto, MySQL es una opción bastante popular. Según la página de HOSTINGER “es un software o servicio utilizado para crear y administrar bases de datos basadas en un modelo relacional” (Hostinger, 2023). Esto significa que MySQL está diseñado para trabajar con datos estructurados en tablas, donde cada tabla tiene filas y columnas.

MySQL es un sistema de gestión de base de datos relacional, que utiliza el lenguaje SQL para gestionar y manipular datos. MySQL es uno de los sistemas de bases de datos más populares en el mundo. Este software permite a los usuarios crear, modificar y consultar bases de datos de manera eficiente, organizando la información en tablas que pueden relacionarse entre sí.

MySQL ofrece muchas características y funcionalidades que lo hacen ideal para diversas aplicaciones y entornos, es utilizado en una amplia gama de proyectos, desde pequeñas aplicaciones web hasta grandes sistemas empresariales, debido a su escalabilidad, fiabilidad y compatibilidad con diferentes sistemas operativos. Además, MySQL soporta grandes volúmenes de datos, maneja múltiples usuarios simultáneamente y ofrece múltiples herramientas para optimizar el rendimiento y garantizar la seguridad de los datos.

2.4 Metodología de Desarrollo

La metodología es un parte esencial del desarrollo, ya que define cómo se abordará y resolverá el proyecto de manera efectiva. Rootstack nos dice en su página que "Una metodología de desarrollo de software se refiere al proceso o a los procesos que los ingenieros y desarrolladores siguen paso por paso cuando están trabajando en el diseño y creación de un nuevo proyecto tecnológico" (Rootstack, 2022).

Para el desarrollo de la presente aplicación web, se ha seleccionado la metodología incremental como marco de trabajo. Esta metodología caracterizada por su enfoque iterativo permite la construcción del sistema a través de la entrega sucesiva de incrementos funcionales. "La metodología incremental es especialmente útil en proyectos donde los recursos y el tiempo son limitados, y permite al desarrollador trabajar en partes manejables del proyecto, entregando funcionalidad adicional en cada iteración. Este enfoque no solo facilita la adaptación a nuevos requisitos, sino que también asegura una entrega constante de valor y una mejor gestión del riesgo" (Pressman, 2020), con esta metodología, nos aseguramos entregables y avances para que el cliente pueda ver cómo va la aplicación.

2.4.1 *Metodología Incremental*

En el mundo del desarrollo de software, la necesidad de métodos ágiles que permitan a los equipos adaptarse rápidamente a los cambios del entorno y satisfacer las demandas del cliente de manera efectiva ha dado lugar a la utilización de metodologías como la incremental, Como es un método adaptativo podemos ir adaptando la metodología de la forma más conveniente en nuestro proyecto, "Como escribe Schwaber y Sutherland (2020), 'Si algún aspecto de un proceso se desvía fuera de los límites aceptables o si el producto resultante es inaceptable, el proceso que se está aplicando o los materiales que se producen deben ajustarse'(p. n.p.).” Y así sacarle el mayor provecho a esta metodología ágil.

Como señala Sommerville “el desarrollo incremental se basa en la entrega frecuente de pequeñas partes del sistema. En nuestro proyecto, hemos adoptado esta metodología para poder obtener feedback del cliente de manera temprana y realizar ajustes de forma ágil." (Sommerville, 2016, p. 77). Para garantizar que el proyecto se ajuste a las necesidades del cliente, hemos optado por una metodología incremental. Esto implica en dividir el desarrollo en pequeñas etapas, cada una de la cuales culminara con una reunión semanal con el cliente. Con esta colaboración estrecha nos permitirá tener una retroalimentación constante y hacer modificaciones en tiempo real, asegurando que el producto final sea exactamente lo que el cliente quiera.

2.4.1.1 Características

La metodología incremental se distingue por varias características clave que la hacen especialmente acertada para el desarrollo de software, particularmente en entornos donde los requisitos pueden evolucionar o no están completamente

definidos desde el inicio. Como señala Sommerville, "El desarrollo incremental es una estrategia efectiva cuando los requisitos no están claros desde el principio, ya que permite refinar y mejorar el sistema a medida que se avanza en su construcción" (Sommerville, 2022, p. 75). A continuación, se describen las características esenciales de este enfoque:

Incrementos Sucesivos: El desarrollo incremental se basa en la creación de múltiples incrementos, cada uno de los cuales agrega funcionalidad al sistema. En lugar de intentar desarrollar el sistema completo de una vez, el proyecto se divide en partes más manejables. Cada incremento es un conjunto completo de características que proporciona una funcionalidad especifica y operativa. Este enfoque nos permite enfocarnos en desarrollar y perfeccionar secciones de la aplicación de manera más aislada, reduciendo la complejidad y el riesgo de errores. Pressman nos dice que "El enfoque incremental permite la construcción de un sistema de manera modular, donde cada incremento es funcional y se puede evaluar, lo que facilita la identificación y corrección de errores en etapas tempranas" (Pressman, 2020, p. 98).

Entrega Frecuente: Una de las principales fortalezas de la metodología incremental es la entrega frecuente de los desarrollos. Cada incremento, que, aunque no constituya el sistema completo, es la versión funcional que puede ser revisada, aprobada y utilizada por el cliente. Esta característica es especialmente útil para asegurar que el desarrollo se mantenga alineado con las expectativas del cliente, permitiendo ajustes tempranos y minimizando el riesgo de desviarse de los objetivos del proyecto. Así como señala Sommerville, "El desarrollo incremental facilita la entrega temprana y regular de versiones funcionales del software, lo que permite obtener retroalimentación continua y ajustar el desarrollo conforme a las necesidades del cliente" (Sommerville, 2022, p. 82).

Retroalimentación Continua: La entrega de incrementos funcionales facilita una retroalimentación continua por parte del cliente. Cada vez que se completa un incremento, los usuarios pueden evaluar el progreso, proporcionar comentarios y sugerir mejoras. Esta interacción constante es crucial para ajustar el desarrollo del proyecto a las necesidades reales del cliente, permitiendo realizar cambios en el diseño o en las funcionalidades antes de avanzar a las siguientes etapas. La retroalimentación continua no solo mejora la calidad del producto final, sino que también incrementa la satisfacción del cliente al involucrarlo en el proceso de desarrollo. Como menciona Wiegers, "La retroalimentación continua es esencial para ajustar el desarrollo a las expectativas del cliente y mejorar la calidad del software, al permitir que los equipos identifiquen y corrijan problemas en etapas tempranas del ciclo de vida del proyecto" (Wiegers, 2013, p. 104).

Reducción del Riesgo: Al abordar el proyecto en incrementos más pequeños, metodología incremental ayuda a reducir el riesgo. Los problemas se pueden identificar y solucionar en fases tempranas, evitando que se conviertan en obstáculos mayores. Además, la capacidad de realizar ajustes después de cada incremento significa que el proyecto puede adaptarse rápidamente a cambios en los requisitos o a nuevas circunstancias, a lo que es fundamental en entornos dinámicos. Sommerville señala que, "El desarrollo incremental permite realizar ajustes en respuesta a los cambios en los requisitos, reduciendo el riesgo y mejorando la adaptabilidad del proyecto" (Sommerville, 2016, p. 52).

2.5 Determinación de requerimientos

Como punto de partida en un proyecto de desarrollo de software la fase de determinación de requerimientos es muy importante para empezar la construcción de un sistema informático. Esta etapa implica la identificación, análisis y documentación de las necesidades y expectativas de los usuarios finales, así como los objetivos y restricciones del proyecto. La página de Visure dice que "Los requisitos de un proyecto de software son las funciones, características y restricciones que debe cumplir el producto final." (Visure, s.f.).

La determinación de requerimientos es un proceso iterativo y colaborativo que involucra a diferentes personas, incluyendo a clientes, usuarios, desarrolladores y otras personas interesadas. E objetivo principal de esta fase es establecer una comprensión clara y completa de lo que se espera del sistema, tanto en términos de funcionalidad como de calidad.

2.5.1 *Recopilación de datos*

Este es un proceso importante en cualquier investigación o proyecto que requiera información relevante para poder alcanzar un objetivo. Este proceso se debe a la recolección de datos sistemática y organizada de d datos, que pueden ser de naturaleza cualitativa o cuantitativa, dependiendo de nuestros objetivos y metodología del proyecto.

En el caso de este proyecto se tomó la opción de una entrevista por medio de una llamada virtual entre el cliente y el desarrollador ya que estas nos permiten llevar a cabo un proceso de recolección de datos de manera organizada, aprovechando la tecnología para facilitar la comunicación a pesar de estar en diferentes provincias, esta modalidad ofrece una plataforma conveniente y accesible para discutir, aclarar y documentar los requerimientos del proyecto, así como cualquier otro aspecto relevante.

2.6 Diseño

En el ciclo de vida de un proyecto software, encontramos una parte importante que es entender como representar de manera visual y concisa los diferentes aspectos de un sistema de software, empezando con su creación, UML ha sido ampliamente adoptado en la industria de desarrollo de software debido a su flexibilidad, expresividad y capacidad para comunicar de manera efectiva las ideas y conceptos relacionados con el diseño, así como el desarrollo de sistemas complejos.

En la página de UML oficial, nos dicen que "Las aplicaciones empresariales de gran envergadura, aquellas que ejecutan las aplicaciones fundamentales de un negocio y mantienen en funcionamiento una empresa, deben ser más que simplemente un conjunto de módulos de código. Deben estar estructuradas de tal manera que permitan la escalabilidad, la seguridad y la ejecución robusta bajo condiciones estresantes, y su estructura, frecuentemente referida como su

arquitectura, debe estar definida de manera suficientemente clara para que los programadores de mantenimiento puedan (¡rápidamente!) encontrar y corregir un error que aparezca mucho después de que los autores originales hayan pasado a otros proyectos"(UML, s.f.)

A continuación, veremos algunos diagramas importantes dentro del UML

2.6.1 *Diagrama de clases*

Este es uno de los más importantes en el Lenguaje de modelado unificado (UML) ya que se utiliza para representar la estructura estática de un sistema software. Este diagrama muestra las clases del sistema, sus atributos, métodos y las relaciones entre ellas, proporcionando una vista estática y detallada de cómo están organizados y relacionados los componentes del software.

El diagrama de clases es una herramienta muy fuerte para comprender la estructura estática de un sistema software y la relación entre sus componentes. Permite a los desarrolladores y analistas visualiza la arquitectura del sistema, identificar las clases principales y sus interacciones y deseñar la base de datos, en la página de Lucidchart “nos dice que los diagramas de clases son un tipo de diagrama de estructura porque describen lo que debe estar presente en el sistema que se está modelando” (Lucidchart)

Algunos elementos clave que se representan en un diagrama de clase son:

Clases: representan los tipos de objetos en el sistema y encapsulan los datos y comportamientos relacionados, “La figura de clase en sí misma consiste en un rectángulo de tres filas. La fila superior contiene el nombre de la clase” (Lucidchart, s.f.)

Atributos: características de una clase que describen el estado del objeto, “la fila del centro contiene los atributos de la clase” (Lucidchart, s.f.)

Métodos: funciones asociadas a una clase que definen su comportamiento, “la última expresa los métodos o las operaciones que la clase puede utilizar” (Lucidchart, s.f.).

2.6.2 *Diagramas de Casos de Uso*

Los diagramas de casos de uso son una herramienta importante en el análisis y diseño de software, especialmente en la fase de especificación de requisitos. Estos diagramas se utilizan para representar las interacciones entre un sistema y sus actores, mostrando como los usuarios interactúan con el sistema para lograr algunos objetivos.

Actores: en un diagrama de casos de uso, llamamos actores a los que interactúan externamente con el sistema y los diferentes casos de uso que describen las acciones que estos actores pueden realizar el sistema, los casos de uso representan las funciones o servicios que el sistema proporciona a los usuarios para satisfacer sus necesidades o metas.

Los actores pueden ser personas, otros sistemas o cualquier entidad externa que interactúe con el sistema en cuestión. Cada actor se representa como un elemento externo al sistema y se conecta a los casos de uso correspondientes, que describe las acciones que realiza en el sistema, según la página de DiagramasUML” un actor es algo o alguien externo al sistema que interactúa de forma directa con el sistema” (DiagramasUML, s.f.).

Relaciones: las relaciones entre los diferentes casos de uso pueden proporcionar una comprensión más profunda de como interactúan los distintos aspectos del sistema para cumplir con los objetivos del usuario. Estas relaciones capturan las interacciones y dependencias entre los casos de uso, lo que ayuda a definir la secuencia de acciones y eventos, que ocurren dentro del sistema la página de DiagramasUML nos dice que “Las

relaciones conectan los casos de uso con los actores o los casos de uso entre sí" (DiagramasUML, s.f.).

2.6.3 *Diagramas de Secuencia*

Con estos diagramas podemos comprender cual es la secuencia del caso de uso en un tiempo estimado, este tipo de diagrama se centra en mostrar como los distintos elementos del sistema se comunican entre sí, destacando el orden y la secuencia de los mensajes intercambiados durante un proceso específicos, en la página de DiagramasUML nos dicen que "Su objetivo es representar el intercambio de mensajes entre los distintos objetos del sistema para cumplir con una funcionalidad. Define, por tanto, el comportamiento dinámico del sistema de información" (DiagramasUML, s.f.).

Objeto: "Un objeto puede ser una instancia de una clase, un módulo, un grupo de clases, en definitiva, un objeto es un componente software que tiene una funcionalidad específica. Dependerá del nivel de abstracción la representación de cada objeto." (DiagramasUML, s.f.).

Mensaje: "Se utiliza un mensaje en el diagrama de secuencia para representar el paso de un mensaje entre dos objetos o entre un objeto y sí mismo. Se representa utilizando una flecha que incluye el nombre del mensaje y los argumentos que incluye y que va desde el objeto que envía el mensaje hasta el objeto que lo recibe." (DiagramasUML, s.f.).

2.7 BackEnd

En este capítulo no enfocaremos en la parte de codificación del proyecto, el lado del servidor o como lo conocemos "BackEnd" es una de las partes más importantes de una aplicación web, ya que podemos procesar las solicitudes del cliente, realizar operaciones en la base de datos

y devolver una respuesta el lado del usuario. Acá desarrollamos la lógica de la aplicación y aseguramos el correcto funcionamiento de la aplicación, según la página gluo_ “En él se configuran todos los aspectos lógicos de una página web o aplicación; abarca la lógica, el almacenamiento de datos y las funciones de seguridad necesarias para que una aplicación funcione correctamente y sea fiable” (Arizbé Ken, 2023).

En nuestro proyecto de la aplicación web de gestión de datos empresariales, el BackEnd desempeña un papel importante al manejar todas las operaciones relacionadas con el registro, almacenamiento y recuperación de datos de las empresas. Esto incluye el manejo de usuarios, la autenticación, la gestión de la base de datos y la exposición de API para que se pueda interactuar con la aplicación de manera segura y eficiente.

En esta sección del proyecto nos enfocaremos en diseñar y desarrollar el BackEnd utilizando tecnologías como Python, Flask y SQLAlchemy. Con estas herramientas tendremos la flexibilidad y la potencia necesarios para crear una aplicación robusta y que cumpla con los requisitos específicos de la empresa. Además, nos aseguraremos de implementar medidas de seguridad adecuadas para proteger la integridad y confidencialidad de los datos empresariales.

2.7.1 *Lenguaje de programación*

El lenguaje de programación nos ayuda a los desarrolladores a instruir a las computadoras para que realicen tareas específicas. Según Ladrón de Guevara. (s.f.). “Un lenguaje de programación está formado por un conjunto de palabras reservadas, símbolos y reglas sintácticas que definen su estructura el significado de sus elementos y expresiones” (p.1)

Con lo anterior, podemos comprender que el lenguaje de programación es una herramienta fundamental en el desarrollo de software y sistemas informáticos, que nos permite a los programadores traducir sus ideas en código ejecutable.

En nuestro entorno utilizaremos el lenguaje de programación Python que nos da algunas características importantes para desarrollar nuestro proyecto.

2.7.2 *Python*

En los lenguajes de alto nivel nos podemos encontrar a Python, que actualmente es muy utilizado por sus características y adaptabilidad que tiene este lenguaje. Según Marzal, A & Gracia, I (s.f.). "Python presenta una seria de ventajas que lo hacen muy atractivo, tanto para su uso profesional como para el aprendizaje de la programación" introducción a la programación con Python (p.16)

Anteriormente hablamos de adaptabilidad y es que Python tiene una característica de multiplataforma en lo que lo lleva a poder ejecutarse en diferentes sistemas operativos como Windows, MacOS y Linux lo que lo convierte en una opción versátil para algunos desarrolladores.

Python es utilizado en una variedad de campos como el desarrollo web, ciencia de datos e inteligencia artificial por mencionar algunos, y por eso ha ganado mucha popularidad en los últimos años debido a su facilidad de uso, Python es un lenguaje potente y flexible y por eso lo convierte en una opción popular para para principiantes y desarrolladores experimentados.

2.7.3 *Framework*

Según la página de UNIR, “Un framework es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software” (UNIR, 2022)

Podemos entender que el framework nos facilita y agiliza el proceso de desarrollo al ofrecer una seria de funciones, componente y patrones de diseño comunes que pueden ser reutilizados en diferentes proyectos.

Estos frameworks incluyen módulos, clases y otros recursos que permiten a los desarrolladores construir aplicaciones de manera más eficiente al proporcionar soluciones predefinidas para tareas recurrentes, en la página de EBAC vemos que “los frameworks son reutilizables y pueden ser aplicados en múltiples proyectos” (EBAC,2023)

Esta herramienta proporciona una base para el desarrollo de software al ofrecer un conjunto de componentes y patrones de diseño comunes, que permiten a los desarrolladores construir aplicaciones de manera más eficiente y siguiendo las buenas prácticas de programación.

2.7.4 *Flask*

Al utilizar Python em muestro proyecto, uno de los frameworks más populares para este lenguaje es Flask, ya que con su simplicidad y facilidad de uso lo convierte en una excelente opción para desarrolladores principiantes y experimentados, en la página web de EPITECH “Flask es un micro framework escrito en Python y desarrollado para simplificar y hacer más fácil la creación de aplicaciones web bajo el patrón MVC” (EPITECH, 2021)

Con Flask podemos crear aplicaciones web con utilizando el patrón de diseño MVC (Modelo-Vista-Controlador) o cualquier otro patrón de diseño que se prefiera. Flask nos proporciona una amplia opción de extensiones y complementos, que facilitan la integración de características adicionales, como la autenticación de usuarios, la gestión de sesiones y la conexión a base dade datos

A diferencia de otros frameworks más complejos, Flask se centra en proporcionar solo las herramientas necesarias para construir una aplicación web totalmente funcional, lo que permite tener un mayor control sobre el código.

Flask es una buena opción para desarrolladores que buscan un framework ligero, flexible y fácil de usar para crear aplicaciones web y esto lo convierte en una herramienta poderosa para construir todo tipo de aplicaciones web con rapidez y eficiencia.

2.7.5 *ORM*

Por sus siglas en inglés (Object Relation Mapping) Es una técnica de programación que nos permite a los desarrolladores trabajar con base de dato relacionales utilizando objetos de programación en lugar de consultas SQL directas, como se afirma en OpenWebinars" nos permite convertir nuestros datos de estas bases de datos relacionales a objetos de nuestro lenguaje orientado a objetos" (OpenWebinars, 2021)

Al eliminar la necesidad de escribir consultas SQL manualmente y al proporcionar una forma orientada a objetos al interactuar con los datos almacenados en la base de datos. Esto nos permite a los desarrolladores centrarse en la lógica de negocio de la aplicación en lugar de preocuparnos por los detalles de cómo se almacenan y recuperan los datos en la base de datos.

En fin, un ORM es una herramienta que facilita el desarrollo de aplicaciones al proporcionar una capa de comunicación entre el código de la aplicación y la base de datos permitiendo trabajar con los datos de manera más eficiente y productiva.

2.7.6 *SQLAlchemy*

Según la página de J2Logo "SQLAlchemy es una librería para python que facilita el acceso a la base de datos relacional, así como las operaciones a realizar sobre la misma" (J2LOGO, s.f.). con esta librería podemos trabajar con base de datos de manera más intuitiva y eficiente, ya que pueden manipular datos utilizando la sintaxis familiar de Python y aprovechar las capacidades del lenguaje orientado a objetos. Esta herramienta nos ofrece una abstracción sobre las complejidades de las consultas SQL y proporciona una interfaz poderosa y flexible para la gestión de datos en aplicaciones web y otros proyectos de software.

2.7.7 *Base de Datos*

Cuando hablamos de una aplicación web vamos a tener datos que tenemos que manejar y guardarlos. La podemos ver como una estructura de datos que están relacionados entre sí de alguna manera.

Una de las bases de datos más comunes es la base de datos relacional. Estas bases de datos se usan en muchas aplicaciones, como gestores de contenido o gestores de inventarios entre muchas otras aplicaciones que requieren manejar datos de manera estructurada, ya que podemos acceder a una gran cantidad de datos, lo que las hace importantes para muchos sistemas de software y aplicaciones.

La página de IBM nos dice que “Los analistas utilizan consultas SQL para combinar diferentes puntos de datos y resumir el rendimiento empresarial, lo que permite a las organizaciones obtener insights, optimizar los flujos de trabajo e identificar nuevas oportunidades” (IBM, s.f.).

2.8 FrontEnd

En el FrontEnd vamos a encontrar reflejado en la parte del cliente la cara de la aplicación, donde podemos interactuar de forma intuitiva con la aplicación o software, es una parte importante de del desarrollo web, ya que podemos enviar y recibir peticiones al servidor, en la página de ARIMETRICS nos dice que “Un sistema de FrontEnd se utiliza principalmente para enviar preguntas y solicitudes, y recibir datos desde el sistema anfitrión. Sirve o proporciona a los usuarios la capacidad de interactuar y utilizar un sistema de información.” (ARIMETRICS, s.f.).

Para lograr una aplicación o software intuitivo, fácil de manejar, estéticamente agradable y responsiva, podemos encontrar algunas tecnologías para lograr nuestros objetivos en el FrontEnd, en “Un desarrollador front-end es un profesional que tiene a su cargo la tarea de crear la interfaz y la experiencia de usuario (UI/UX) con la que los usuarios interactúan para acceder a la aplicación” (Olawanle, 2022)

Con la unión de las siguientes tecnologías obtendremos un software completo para acceder a los datos del BackEnd e interactuar con la aplicación.

2.8.1 *React*

React es una de las librerías más importantes en estos tiempos, ya que nos permite crear paginas SPA por sus siglas en ingles “Single Page Application” e interfaces de usuario, en la página de Hostinger dice que “React contiene una colección de fragmentos

de código JavaScript reutilizables utilizados para crear interfaces de usuario (UI) llamadas componentes.” (Deyimar, 2023).

Lo que hace tan popular a esta tecnología son sus características y que también va de la mano con otras bibliotecas según HOSTINGER “Una librería de gestión de estados facilita la comunicación y el intercambio de datos entre los componentes. Actualmente existen varias librerías de gestión de estados de terceros, siendo Redux y Recoil las dos más populares.” (Deyimar, 2023).

Al estar basado en JavaScript nos facilita mucho el desarrollo y el aprendizaje en Reactjs. Si un desarrollador ya sabe mucho o poco de JavaScript no tendría problemas en implementar esta tecnología a su aplicación “Los desarrolladores con conocimientos de JavaScript pueden aprender a utilizar React en muy poco tiempo, ya que se basa en JavaScript plano y en un enfoque basado en componentes”

2.8.2 *CSS*

El CSS o cascading style sheets nos ayuda a darle estilos a nuestros componentes de Reactjs, ya que podemos cambiarle la apariencia como los colores tamaños etc, con ellas podemos crear paginas o aplicaciones web más vistosas, Enel blog de HubSpot nos dicen que “Básicamente, es un lenguaje que maneja el diseño y presentación de las páginas web, es decir, cómo lucen cuando un usuario las visita” (HubSpot,2023).

La idea de una página o aplicación web sin CSS no es muy agradable, ya que sería una aplicación muy cuadrada y sin apariencia atractiva para los usuarios finales, al darle los estilos podemos convertir nuestra plataforma en algo muy llamativo para los usuarios y tener una experiencia agradable mientras utilizan la plataforma.

Al utilizar CSS tenemos una gran ventaja, que es que podemos manejar nuestros estilos en una hoja aparte, en una hoja de estilo CSS y así mantener nuestro código de HTML, Reactjs o JavaScript más limpio y así poder darle mantenimiento a la aplicación más fácilmente.

3 Capitulo III, Marco Metodológico

En este marco metodológico se describirá el enfoque y los métodos utilizados para poder llegar a realizar la aplicación web para Acqua Digital. En este punto es muy importante tener definida la metodología y las técnicas que se emplearan para tener una estructura que pueda garantizar la finalización de la aplicación web garantizando la calidad del producto final. Para este proyecto se definió por utilizar la metodología incremental, ya que nos da unas características flexibles y adaptable a los proyectos de tecnologías ya que vemos avances continuos.

Así como nos da a entender Arias (2012 p.16) nos dice que el marco metodológico es el "conjunto de pasos, técnicas y procedimientos que se emplean para formular y resolver problemas". Al tener una estructura para nuestro proyecto podemos identificar a base de hipótesis nuestros problemas y así poder solucionarlos de manera estructurada.

3.1 Tipo de Investigación

En este proyecto se decidió utilizar el tipo de investigación aplicada, ya que su enfoque en la solución de problemas prácticos se alinea perfectamente con el desarrollo de una aplicación web. Al igual que la metodología de desarrollo incremental, la investigación aplicada busca obtener resultados tangibles de manera rápida y eficiente. Este tipo de investigación implica diseño, implementación y evaluación de soluciones tecnológicas que abordan necesidades específicas de los involucrados.

Nuestro objetivo es crear una aplicación web que es crear un producto totalmente nuevo y funcional que va a cumplir con los requisitos del usuario. Este tipo de investigación se adapta muy bien ya que tendremos resultados inmediatos. Según Hernández, Fernández y baptista (2014), la investigación aplicada "busca el conocimiento que tenga aplicación práctica en la solución de problemas específicos" (p. 12).

3.2 Enfoque de investigación

Se adoptará un enfoque de investigación cualitativo en este proyecto. Ya que este enfoque se centra en cumplir las expectativas y necesidades del cliente con métodos para la obtención de datos.

3.3 Justificación del enfoque cualitativo

3.3.1 Recopilación de requerimientos

Se utilizarán entrevistas semiestructuradas con el líder de Acqua Digital para entender sus necesidades y expectativas. Estas entrevistas se llevarán a cabo de manera virtual con plataformas de videoconferencias. Con el enfoque cualitativo podemos obtener un contexto más detallado de los requerimientos funcionales y no funcionales de la aplicación.

El enfoque cualitativo para este proyecto es bastante adecuando ya que nos permite obtener una información bastante detallada e importante para el desarrollo de una aplicación web y cumplir con los requisitos específicos del cliente y se ajuste a sus necesidades y expectativas

para asegurarnos que la aplicación cumpla con todos los requerimientos y expectativas del cliente, se utilizaran algunas técnicas de recopilación de requerimientos. Aunque las reuniones serán virtuales, estas técnicas permitirán obtener una comprensión clara y detallada de los requerimientos.

3.4 Diseño de la investigación

En esta parte desarrollaremos el cómo se llevará a cabo la investigación, incluyendo los métodos y técnicas que se utilizaran para obtener y analizar los datos. A continuación, se mostrará la estructura del diseño de la investigación.

3.4.1 *Técnica de Recopilación de Requerimientos*

3.4.1.1 Entrevistas con el cliente

se realizarán entrevistas mediante plataformas de videoconferencia, permitiéndonos hacer preguntas abiertas facilitándonos información detallada y permitiéndonos profundizar en los temas específicos necesarios. Según Somerville (2016), “Las entrevistas formales o informales con participantes del sistema son una parte de la mayoría de los procesos de ingeniería de requerimientos. En estas entrevistas, el equipo de ingeniería de requerimientos formula preguntas a los participantes sobre el sistema que actualmente usan y el sistema que se va a desarrollar.” (p. 104). Se llevarán a cabo al inicio del proyecto y de manera regular durante el desarrollo de la aplicación para asegurar una comunicación constante en el caso de que los requerimientos vayan evolucionando conforme a las necesidades del cliente.

Durante la entrevista regulares también se mantendrán revisiones para revisar y actualizar los requerimientos y así mantener las expectativas y objetivos del cliente.

3.4.2 *Validación de Requerimientos*

Después de obtener los requerimientos validarlos es un proceso importante para asegurar que los requerimientos sean precisos, y relevantes a lo que quiere el cliente, sin la validación de los requerimientos no se puede seguir con el desarrollo de la aplicación.

3.4.2.1 Revisión inicial de requerimientos

Al obtener los requerimientos se le dan forma en un documento donde se diferencian entre requerimientos funcionales y no funcionales y después dándole un código a cada uno de los requerimientos para identificarlos de manera más sencilla, después de revisan los requerimientos para ver la coherencia con las expectativas y necesidades del cliente

3.4.2.2 Validación con el cliente

Mediante una reunión se leerán y entregarán los requerimientos al cliente para que proporcione una retroalimentación sobre los requerimientos y se tomaran notas detalladas de los ajustes o aclaración necesario

3.4.3 Proceso de diseño y modelado

Para mantener una integración coherente entre la apariencia visual y la estructura de datos el proceso de diseño y modelado, se incluirán el diseño de la interfaz de usuario y el diseño y modelado de la base de datos basándonos en los requerimientos del cliente Acqua Digital.

- **Interfaz de usuario**

Se desarrollará la interfaz de usuario siguiendo algunos requerimientos funcionales y no funcionales, basándonos en la usabilidad, accesibilidad y experiencia del usuario.

- **Base de datos**

Se definirán las entidades, atributos y relaciones de la base de datos a través de un modelo entidad relación. Incluirá la identificación de las tablas, llaves primarias y foráneas y su respectiva relación y así creando un diagrama de entidad relación de la base de datos.

Para apoyar se realizarán algunos diagramas para poder detallar los requerimientos y facilitar el diseño de la aplicación con casos de uso, diagramas de casos de uso y diagramas de secuencia.

3.5 Desarrollo de software

Al tener el diseño de la aplicación, damos por iniciada la parte donde se desarrolla la aplicación según los requerimientos, diseño y modelado todo esto con validación de Acqua Digital, como se acordó la entrega de código de BackEnd y FrontEnd, se maneja en conjunto con la metodología incremental adaptado a una sola persona ya que esta metodología está pensada en un trabajo grupal, para generar el código deseado.

Este código se generará utilizando las siguientes tecnologías:

3.5.1 *Lenguajes de programación*

- **Python:** este será el lenguaje principal de la aplicación, para desarrollar el API o servidor deseado, es simple y tiene facilidad de uso, obteniendo un desarrollo rápido y eficiente.

- **JavaScript:** este lenguaje se utilizará para el desarrollo de la interfaz de usuario, dando interactividad y dinamismo al software.

3.5.2 Frameworks y librerías

- **Flask:** este es un framework de Python que se utilizará para el desarrollo del API o del servidor, ya que es un framework simple y flexible nos permite desarrollar la aplicación web de manera rápida y estructurada.
- **SQLAquemy:** con este ORM (Object-Relational Mapping) para Python facilitara la manipulación de la base de datos relacionales mediante la utilización de objetos de Python.
- **React:** Esta biblioteca para JavaScript, nos ayudará a generar la interfaz de usuario, su enfoque en reutilización de componentes nos permitirá avanzar de manera rápida.
- **CSS:** se utilizará para el diseño y estilo de la aplicación asegurando una experiencia de usuario agradable para el usuario final.

Al utilizar estas tecnologías desarrollaremos esta aplicación web y al final tendremos una aplicación robusta y agradable visualmente para el usuario final.

3.6 Pruebas y evaluación

Se realizan unas series de pruebas para asegurar que la aplicación web cumple con los requisitos específicos y que funciona correctamente. Las pruebas funcionales verifican que cada función de la aplicación opere basándose con las expectativas. Según Beizer (1990), "las pruebas funcionales son cruciales para validar que el software

cumple con los requisitos especificados y funciona según lo esperado, asegurando que cada característica opere correctamente en condiciones normales de uso" (p. 123).

3.6.1 *Pruebas funcionales*

Se pondrá a prueba la aplicación y Se desarrollará un documento con el resultado de las pruebas funcionales con la descripción de cada requerimiento, se anotará si la funcionalidad es correcta o no y se anota el detalle del fallo de la prueba.

3.6.1.1 Criterios de evaluación del software

- **Funcionalidad**

Criterio: La aplicación debe cumplir con todos los requerimientos funcionales especificados. Cada función probada debe de funcionar correctamente.

Evaluación: Se verifica que todas las características especificadas en los requerimientos funcionen según los esperado durante las pruebas funcionales.

- **Usabilidad**

Criterio: La interfaz de usuario debe ser intuitiva y fácil de usar para el usuario ya estos deben de utilizar la aplicación sin ningún problema.

Evaluación: Se evalúa la facilidad de uso de la interfaz mediante pruebas funcionales, asegurando que los usuarios pueden completar tareas comunes de manera eficiente.

- **Rendimiento**

Criterio: La aplicación debe de tener un rendimiento rápido con forme a las interacciones del usuario y manejar correctamente la información del BackEnd.

Evaluación: Se evalúa el tiempo de respuesta y eficiencia de las operaciones durante las pruebas y así asegurar un rendimiento adecuando para la aplicación.

- **Seguridad**

Criterio: La aplicación debe de proteger los datos del usuario y manejar de manera segura las operaciones sensibles.

Evaluación: Se verifica que los datos sensibles estén protegidos y las rutas estén protegidas así asegurando que ningún usuario pueda ingresar a ningún modulo al que no tenga permisos.

3.6.2 *Documentación de la aplicación*

Se desarrollará un manual de usuario con como parte de la implementación, ya que así podremos dar al cliente instrucciones necesarias para utilizar la aplicación de una manera efectiva. Según Pressman (2014), "el desarrollo de un manual de usuario claro y completo es fundamental para garantizar que los usuarios puedan utilizar la aplicación de

manera efectiva y aprovechar al máximo sus funcionalidades" (p. 345). A continuación, puntos importantes del manual de usuario:

Creación de contenido

Se redactan las secciones del manual de usuario, incluyendo:

Introducción y descripción de la aplicación.

Instrucciones paso a paso para realizar las tareas y funcionalidades.

Recomendaciones para aprovechar al máximo la aplicación.

Diseño y formato

Se diseña el manual de usuario con formato claro y fácil de entender utilizando lenguaje sencillo y directo, así como utilización de numeración en los pasos.

Se incluyen capturas de pantalla de la aplicación, así como el uso de flechas y círculos para ilustrar los procesos y funcionalidades de la aplicación.

4 Capitulo IV Recolección de Requerimientos

4.1 Técnica de recolección de requerimientos

Para la obtención de los requerimientos de la aplicación web se mantuvieron reuniones virtuales para conversar con el cliente y que el expresara cuales eran sus necesidades y expectativas del proyecto de la aplicación del gestor de datos para Acqua Digital. Las reuniones virtuales fueron la principal técnica para recolectar los requerimientos. Como somos de provincias diferentes esto nos facilitó la comunicación con el cliente. Estas reuniones se programaron de manera regular y estructuradas de la siguiente manera:

4.1.1 *Planificación de reuniones*

Frecuencia: las reuniones se planificaron semanalmente para tener una comunicación constante con el cliente y así poder evacuar dudas y clarificar los requerimientos.

Duración: las reuniones fueron aproximadamente de una hora, tiempo suficiente para discutir con el cliente los requerimientos y expectativas de la aplicación web.

Formato: Estas reuniones se llevaron a cabo de manera virtual, permitiendo a las dos partes involucradas la disponibilidad requerida para las reuniones virtuales. Todo esto utilizando plataformas como Zoom y Google meet

4.1.2 *Desarrollo de las reuniones*

Inicio: Las reuniones dieron inicio revisando los objetivos de la sesión y un pequeño resumen de los puntos tratados en la reunión anterior.

Recopilación de información: Se discutieron los requerimientos específicos del proyecto, incluyendo las funcionalidades deseadas, prioridades y si se presenta algún cambio necesario.

Documentación: Se tomaron registros de cada requerimiento en un documento para asegurarnos que quedaran registrados correctamente. Además, se compartía la pantalla para discutir los requerimientos y actualizar la documentación en tiempo real.

Cierre: Al final de cada reunión, se hacía un resumen de los puntos clave discutidos.

4.1.3 *Validación de requerimientos*

Confirmación y Validación: al tener todos los requerimientos se hizo una última sesión con el cliente para la confirmación de los requerimientos para asegurar las funcionalidades de la aplicación.

Especificaciones técnicas: realización de documentos técnicos que describen los requisitos técnicos y las limitaciones del proyecto.

4.2 análisis de Requerimientos

A continuación, la tabla de los requerimientos obtenidos de las reuniones con el cliente:

		Aplicacion web de gestor de datos para Acqua Digital			
		requerimientos Funcionales			
	ID	Modulos	caracteristicas	Prioridad	Estado
	RF001	registro de empleados	Nombre, apellido, cédula, email, fecha de ingreso, puesto, vacaciones, horario y salario.	ALTA	COMPLETADO
	RF002	registro de pago de planilla	Nombre del empleado, fecha de pago, salario base, horas trabajadas, deducciones y total bruto.	ALTA	COMPLETADO
	RF003	registro de activos	Nombre del item y modelo.	ALTA	COMPLETADO
	RF004	registro de pagos	Pagos realizados a entidades de servicios (entidad, número de identificación de servicio, fecha de corte y monto). Pagos por compra de software o hardware (nombre, proveedor, fecha de corte y monto).	ALTA	COMPLETADO
	RF005	registro de ingresos	Ingresos de la empresa por servicios de marketing (cliente, fecha de pago y monto).	ALTA	COMPLETADO
					COMPLETADO
		requerimientos de seguridad y acceso	caracteristicas		
	RF006	autenticacion de usuarios	Implementar un sistema de login para acceder a la plataforma.	ALTA	COMPLETADO
	RF007	permisos de usuarios	• Los Super Admins tendrán acceso completo a todas las partes de la aplicación. • Los Admins tendrán acceso limitado según su rol (podrán realizar registros de pagos e ingresos, pero no tendrán acceso a la parte de planilla ni a al modulo de reportes	ALTA	COMPLETADO
		requerimientos de administracion de usuarios	caracteristicas		COMPLETADO
	RF008	administracion de usuarios	Super Admins podrán agregar, eliminar y editar usuarios que registran los pagos y cobros y de la aplicación.	ALTA	COMPLETADO
		requerimientos de reportes	caracteristicas		
	RF009	reportes	se creara un modulo de reportes (planilla, pagos de servicios y clientes e ingresos) incluirá una sección de reportes	ALTA	COMPLETADO
		requerimientos No Funcionales			
		BackEnd	caracteristicas		
	RNF001	Base de Datos	• Utilizar MySQL para gestionar la base de datos.	ALTA	COMPLETADO
	RNF002	API	Desarrollar una API utilizando Python, Flask y SQLAlchemy para manejar todas las operaciones de la aplicación, incluyendo la autenticación de usuarios y el acceso a los datos.	ALTA	COMPLETADO
		requerimientos de interfaz de usuario	caracteristicas		
	RNF003	Pagina SPA Responsiva	Crear una página de aplicación de una sola página totalmente responsiva para dispositivos más pequeños.	ALTA	COMPLETADO
	RNF004	estilo	Utilizar React junto con CSS y MUI para la parte de estilos.	ALTA	COMPLETADO
	RNF005	Diseño	Seguir la línea de colores del logo de la empresa para mantener la coherencia visual en toda la aplicación.	ALTA	COMPLETADO

Ilustración 1 Tabla de Requerimientos.

4.2.1 *Módulos*

Se llega a la conclusión de que la aplicación web deberá tener los siguientes módulos para poder cumplir con los requerimientos y expectativas del cliente, estos son los siguientes:

Módulo de autenticación: los usuarios deberán ser autenticados para poder utilizar la aplicación

Módulo de planilla: este módulo nos permite hacer un CRUD de los empleados y también registrar los pagos a un empleado en específico.

Módulo de pagos de servicio: con este módulo cumplimos con el registro de todos los pagos a servicios que se hacen mes a mes.

Clientes: Este módulo nos permite hacer un CRUD a los clientes. Nos permite llevar el registro de los clientes de la empresa.

Activos: este módulo nos permite llevar el registro de los activos de la empresa y también el registro de si algún empleado tiene asignado uno de estos activos de la empresa.

Ingreso: Con este módulo se lleva el registro de los ingresos de la empresa y de cual cliente generó el ingreso.

Compras: en este módulo llevamos el registro de todas las compras de software y hardware realizadas por la empresa.

Reportes: con este módulo llegamos a los reportes de la empresa, con información de la información requerida por la empresa.

4.2.2 *Usuarios*

Analizando los requerimientos se identifica que solo habrá 2 tipos de usuarios con los que se puede autenticar el usuario, que son los siguientes.

Superadmin: este usuario es el rol más importante o el que tiene un mayor permiso dándole acceso total a toda la aplicación de gestión de datos de Acqua Digital.

Admin: Este rol tiene menos permisos, limitando el acceso a ciertos módulos a los cuales no tendrá visibilidad ya que estarán bloqueados.

4.2.3 *Seguridad*

Se llego a una conclusión con el cliente sobre la seguridad de la aplicación, teniendo en cuenta las siguientes características de seguridad:

Autenticación: la aplicación debe de tener un sistema de autenticación con los datos proporcionados por los administradores.

Restricción de módulos: Dependiendo de su autenticación el usuario tendrá un rol guardado en una cookie para restringir ciertos módulos a los que el usuario podrá ingresar.

Ruta protegida: EL perfil de administrador deberá ser una ruta protegida la cual solo por autenticación podrá tener acceso y al contrario se redirigirá al módulo de autenticación.

5 Capitulo V: Diseño de la Aplicación

5.1 Diseño de la interfaz de usuario (UI)

Diseñar la interfaz de usuario es muy importante y también hay que tener en cuenta parte de las expectativas del cliente como los requerimientos, esto nos lleva a tomar decisiones sobre cómo vamos a desarrollar la interfaz. Se seguirán los estándares en las aplicaciones para mantener una aplicación intuitiva de fácil manejo para los usuarios finales.

5.1.1 *Principios de diseño*

En el mundo de las aplicaciones web hay estándares para mantener las aplicaciones intuitivas y de fácil manejo, esto con el fin de que el usuario ya se sienta familiarizado con la aplicación sin antes haberla utilizado. Para esto se deben de seguir algunos principios básicos de diseño de interfaces como:

Simplicidad: mantener una interfaz simple ayuda a que el usuario se mantenga tranquilo y pueda comprender y utilizar la aplicación sin ningún problema. Mantener una interfaz limpia y centrada en las funciones importantes, proporciona que los usuarios se centren en los objetivos de las tareas.

Consistencia: mantener concordancia con los colores de los botones y los mensajes hacen que el usuario no cometa errores, por ejemplo: los botones de aceptación por estándar son verdes y los de cancelar son rojos. Se mantienen ese tipo de estándares para que el usuario tenga la intuición de lo que puede pasar con solo ver los colores de los botones

Legibilidad: Asegurar que el texto sea legible es muy importante para que le sea fácil el uso de la aplicación al usuario, tomando en cuenta el tamaño de la fuente en los textos y el contraste del color de fondo para facilitar la lectura de los textos.

5.1.2 *Proceso de diseño iterativo*

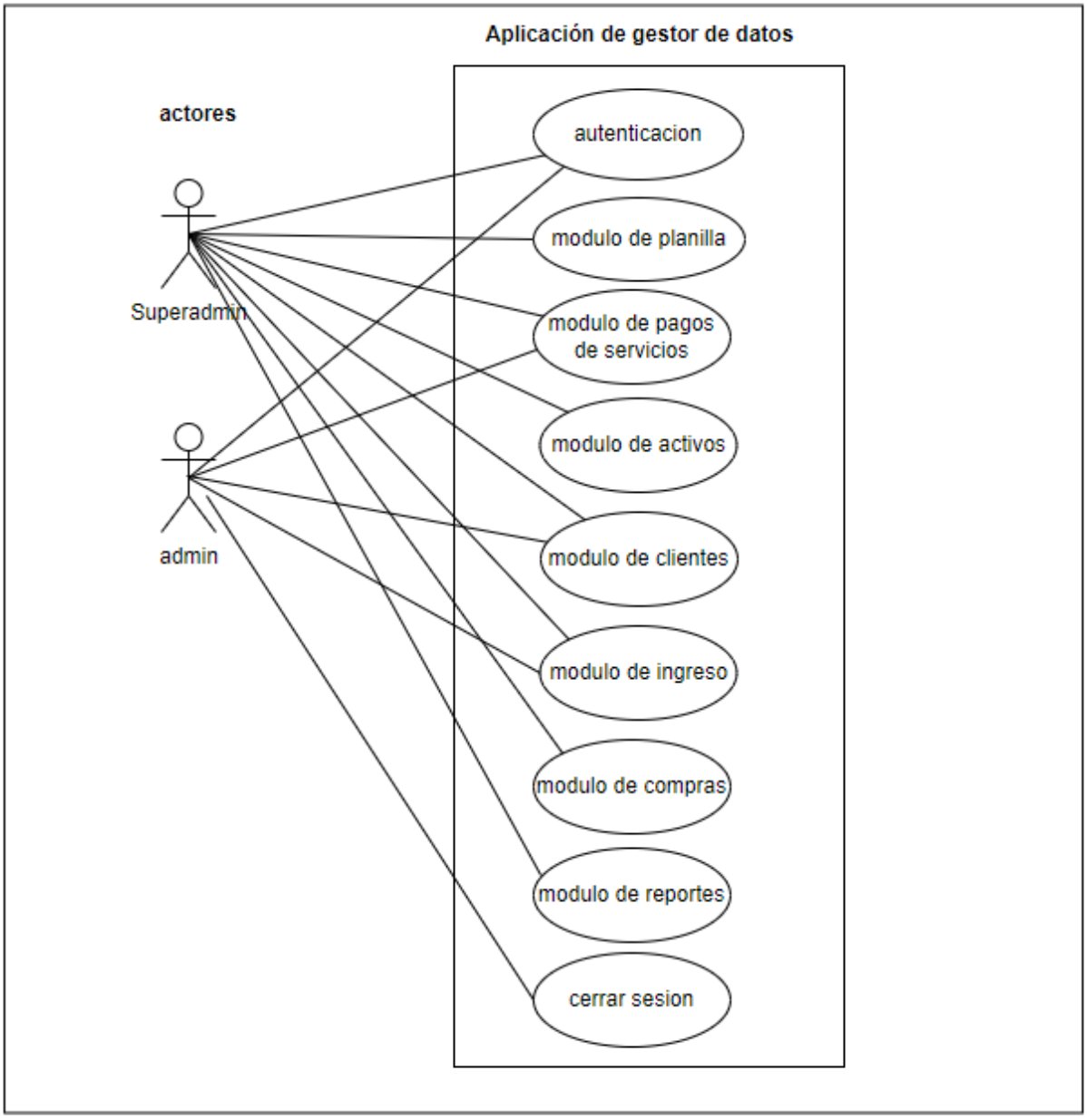
No se utilizaron wireframes y mockups antes del desarrollo, el diseño de la interfaz se desarrolló de manera iterativa, así ajustándose continuamente en función del feedback y las pruebas:

Desarrollo Incremental: se siguió la metodología de desarrollo incremental, implementado y probando funcionalidades de la interfaz de usuario en fases sucesivas. Este enfoque nos permitió construir la aplicación de manera gradual, entregando versiones funcionales cada vez mas completas y permitiendo realizar ajustes y mejoras basados en la retroalimentación del cliente en cada etapa.

Feedback Continuo: al mantener reuniones constantes con el cliente se pudo refinar la interfaz de usuario con las sugerencias que tenía en cada una de las reuniones, asegurando que la interfaz cumpliera con las expectativas y requisitos funcionales.

5.1.3 *Casos de Uso*

Para la realización de la interfaz de usuario también se utilizó los casos de uso para saber cómo interactúan los usuarios con la aplicación y también para saber que funcionalidades debe tener la aplicación.



Fuente: creación propia

Ilustración 2 diagrama de casos de uso

5.1.3.1 Caso de uso de autenticación

en este caso de uso desarrolla la interacción del usuario con la aplicación a la hora de autenticarse en la plataforma.

Caso de uso	CU-01
nombre	autenticación
descripción	El actor se autentica con sus respectivas credenciales
Actores	Superadministrador – admin
Secuencia normal	
Actor	software
1-Usuario ingresa a la plataforma	
	2-Plataforma solicita las credenciales al usuario
3-Usuario ingresa las credenciales	
	4-Se autentica el usuario en la base de datos
	5-Se redirige al perfil registrado
Flujo alternativo	
4- si las credenciales no se encuentran en la base de datos, la plataforma negara el acceso al usuario	
Caso de Uso relacionados	Sin relaciones
precondición	Sin precondición
postcondición	Otorgara el acceso a la aplicación

Fuente: Creación propia

Ilustración 3. caso de uso: autenticación

5.1.3.2 Caso de uso de módulo de planilla

En este caso tendremos la interacción entre el usuario y el ingreso y el CRUD de planilla, viendo como solo el superadministrador tiene acceso a este módulo.

Caso de uso	CU-02
nombre	Módulo de planilla
descripción	El actor ingresa al módulo requerido
Actores	Superadministrador
Secuencia normal	
Actor	software
1-Usuario ingresa a módulo de planilla.	
2- selecciona la opción entre registrar empleado, editar y eliminar.	
	3-la aplicación registra, edita o elimina el empleado.
Flujo alternativo	
3- si falta algún dato en algún formulario no se realizará el registro o el editar.	
Caso de Uso relacionados	Sin relaciones
precondición	El usuario debe estar debidamente autenticado
postcondición	Dependiendo de cuál sea la opción que ejecute, se actualizara la tabla en la base de datos.

Fuente: creación propia

Ilustración 4. caso de uso de módulo de planilla

5.1.3.3 Caso de uso de módulo de pago planilla

Podemos ver la interacción del superadministrador a la hora de registrar un pago que se le hace a un empleado ya registrado.

Caso de uso	CU-03
nombre	Módulo de pago de planilla
descripción	El actor ingresa al módulo para registrar un pago
Actores	Superadministrador
Secuencia normal	
Actor	software
	1-la plataforma despliega la lista de los empleados registrados.
2- usuario selecciona al empleado en la tabla.	
	3- la aplicación despliega un formulario.
4-el usuario llena el formulario y acepta.	
	5- la aplicación verifica los datos y registra el pago al cliente seleccionado por el usuario.
Flujo alternativo	
4- si falta algún dato en el formulario no se realizará el registro y enviará un mensaje de error	
Caso de Uso relacionados	Sin relaciones
precondición	El usuario debe estar debidamente autenticado
postcondición	Se registrará el pago al usuario.

Fuente: creación propia

Ilustración 5caso de uso de pago de planilla

5.1.3.4 Caso de uso de módulo de pagos de servicios.

En este caso tendremos la interacción entre el usuario y el ingreso y el CRUD de pagos de servicios, viendo como superadministrador y admin tienen acceso a este módulo.

Caso de uso	CU-04
nombre	Módulo de pagos de servicio
descripción	El actor ingresa al módulo de pagos de servicios
Actores	Superadministrador - admin
Secuencia normal	
Actor	software
1-Usuario ingresa a módulo de pagos de servicios.	
2- selecciona la opción entre registrar empleado, editar y eliminar.	
3-el usuario acepta la acción	
	4-la aplicación registra, edita o elimina el pago de servicio.
Flujo alternativo	
3- si falta algún dato en algún formulario no se realizará el registro o el editar.	
Caso de Uso relacionados	Sin relaciones
precondición	El usuario debe estar debidamente autenticado
postcondición	Dependiendo de cuál sea la opción que ejecute, se actualizara la tabla en la base de datos.

Fuente: creación propia

Ilustración 6. caso de uso de pagos de servicios

5.1.3.5 Caso de uso de módulo de clientes

En este caso tendremos la interacción entre el usuario y el ingreso y el CRUD de clientes, viendo como superadministrador y admin tienen acceso a este módulo.

Caso de uso	CU-05
nombre	Módulo de clientes
descripción	El actor ingresa al módulo de clientes
Actores	Superadministrador – admin
Secuencia normal	
Actor	Software
1-Usuario ingresa a módulo de pagos de servicios.	
2- selecciona la opción entre registrar empleado, editar y eliminar.	
3-el usuario acepta la acción	
	4-la aplicación registra, edita o elimina el cliente.
Flujo alternativo	
3- si falta algún dato en algún formulario no se realizará el registro o el editar.	
Caso de Uso relacionados	Sin relaciones
precondición	El usuario debe estar debidamente autenticado
postcondición	Dependiendo de cuál sea la opción que ejecute, se actualizara la tabla en la base de datos.

Fuente: creación propia

Ilustración 7. caso de uso de clientes

5.1.3.6 Caso de uso de módulo de activos

En este caso tendremos la interacción entre el usuario y el ingreso y el CRUD de pagos de servicios, viendo como solo el superadministrador tiene acceso a este módulo.

Caso de uso	CU-06
nombre	Módulo de activos
descripción	El actor ingresa al módulo de activos
Actores	Superadministrador
Secuencia normal	
Actor	Software
1-Usuario ingresa a módulo de pagos de servicios.	
2- selecciona la opción entre registrar empleado, editar y eliminar.	
3-el usuario acepta la acción	
	4-la aplicación registra, edita o elimina el cliente.
Flujo alternativo	
3- si falta algún dato en algún formulario no se realizará el registro o el editar.	
Caso de Uso relacionados	Sin relaciones
precondición	El usuario debe estar debidamente autenticado
postcondición	Dependiendo de cuál sea la opción que ejecute, se actualizara la tabla en la base de datos.

Fuente: creación propia

Ilustración 8.caso de uso de modulo de activos

5.1.3.7 Caso de uso de asignar activo

En este caso de uso podemos ver cuál es la interacción del superadministrador con la plataforma al asignarle un empleado al activo.

Caso de uso	CU-07
nombre	Asignar activo
descripción	El actor ingresa al módulo de activos y asigna un activo a un empleado previamente registrado
Actores	Superadministrador
Secuencia normal	
Actor	Software
1-ingresa al módulo de activos.	2- se despliega la tabla con todos los activos registrados
3-el usuario selecciona asignar	4-Se despliega una tabla con todos los empleados registrados.
6-El usuario selecciona al empleado y selecciona asignar	5-Se muestra el mensaje de asignación completada
Flujo alternativo	
Caso de Uso relacionados	Caso de uso #6
precondición	El usuario debe estar debidamente autenticado
postcondición	Se actualizará el activo, dándole la asignación a un empleado

Fuente: creación propia

Ilustracion 9.caso de uso de asignar activo

5.1.3.8 Caso de uso de desasignar activo

En este caso vemos la interacción del superadministrador a la hora de desasignar un empleado al activo que ya tenía un empleado asignado

Caso de uso	CU-08
nombre	Desasignar activo
descripción	El actor ingresa al módulo de activos y elimina la asignación que tiene un activo a un empleado.
Actores	Superadministrador
Secuencia normal	
Actor	Software
1-ingresa al módulo de activos.	2- se despliega la tabla con todos los activos registrados y nombres del empleado que tiene asignado ese activo.
3-el usuario selecciona Desasignar	4-se cancela la asignación que tenia ese activo con el empleado
Flujo alternativo	
Caso de Uso relacionados	Caso de uso #6
precondición	El usuario debe estar debidamente autenticado
postcondición	Se actualizará el activo, quitándole la asignación que tenía con el empleado

Fuente: creación propia

Ilustración 10.caso de uso de asignar activo

5.1.3.9 Caso de uso de módulo de ingresos

En este caso tendremos la interacción entre el usuario y el registro de un ingreso, en el cual ya sea superadministrador o admin pueden ingresar a este módulo.

Caso de uso	CU-09
nombre	Módulo de ingresos
descripción	El actor ingresa al módulo de registro de ingresos
Actores	Superadministrador - admin
Secuencia normal	
Actor	Software
1-Usuario ingresa a módulo de ingresos.	2 se despliega la lista de todos los clientes registrados
3- selecciona la cliente que está pagando	4-se despliega un formulario con los detalles del ingreso
5-el usuario llena el formulario	6-Se muestra un mensaje de éxito
Flujo alternativo	
5- si falta algún dato en algún formulario no se realizará el registro o el editar.	
Caso de Uso relacionados	Sin relaciones
precondición	El usuario debe estar debidamente autenticado
postcondición	Se actualiza la tabla de ingresos.

Fuente: creación propia

Ilustración 11.caso de uso de registro ingresos

5.1.3.10Caso de uso de módulo de compras

En este caso tendremos la interacción entre el usuario y el ingreso y el CRUD de pagos de servicios, viendo como solo el superadministrador tiene acceso a este módulo.

Caso de uso	CU-010
nombre	Módulo de compras
descripción	El actor ingresa al módulo de compras
Actores	Superadministrador
Secuencia normal	
Actor	Software
1-ingresa al módulo de compras.	
2- selecciona la opción entre registrar empleado, editar y eliminar.	
3-el usuario acepta la acción	
	4-la aplicación registra, edita o elimina el cliente.
Flujo alternativo	
Caso de Uso relacionados	Sin relaciones
precondición	El usuario debe estar debidamente autenticado
postcondición	Se actualizará la compra en la base de datos.

Fuente: creación propia

Ilustración 12.caso de uso de compras

5.1.3.11Caso de uso de módulo de reportes

En este caso de uso podemos ver la interacción del usuario superadministrador con el módulo de reportes, desde abriendo el submenú de reportes y eligiendo la opción deseada para desplegar el reporte seleccionado.

Caso de uso	CU-011
nombre	Módulo de reportes
descripción	El actor ingresa al módulo de reportes
Actores	Superadministrador
Secuencia normal	
Actor	Software
1-ingresa al módulo de reportes.	2-Se despliega un submenú de reportes
3 selecciona la opción deseada	4- se despliega el reporte deseado
Flujo alternativo	
Caso de Uso relacionados	Sin relaciones
precondición	El usuario debe estar debidamente autenticado
postcondición	Se despliega el reporte requerido por el usuario.

Fuente: creación propia

Ilustración 13.caso de uso de reportes

5.1.3.12caso de uso de cerrar sesión

en este caso vemos como los usuarios pueden cerrar la sesión que tenían abierta y volver a la página de autenticación.

Caso de uso	CU-012
nombre	Cerrar sesión
descripción	El actor cierra la sesión
Actores	Superadministrador – admin
Secuencia normal	
Actor	Software
1-se selecciona la opción de cerrar sesión	2-Se despliega un mensaje de confirmación
	3-se redirige a la vista de autenticación
Flujo alternativo	
Caso de Uso relacionados	Sin relaciones
precondición	El usuario debe estar debidamente autenticado
postcondición	Se despliega el reporte requerido por el usuario.

Fuente: creación propia

Ilustración 14.caso de uso de cerrar sesión

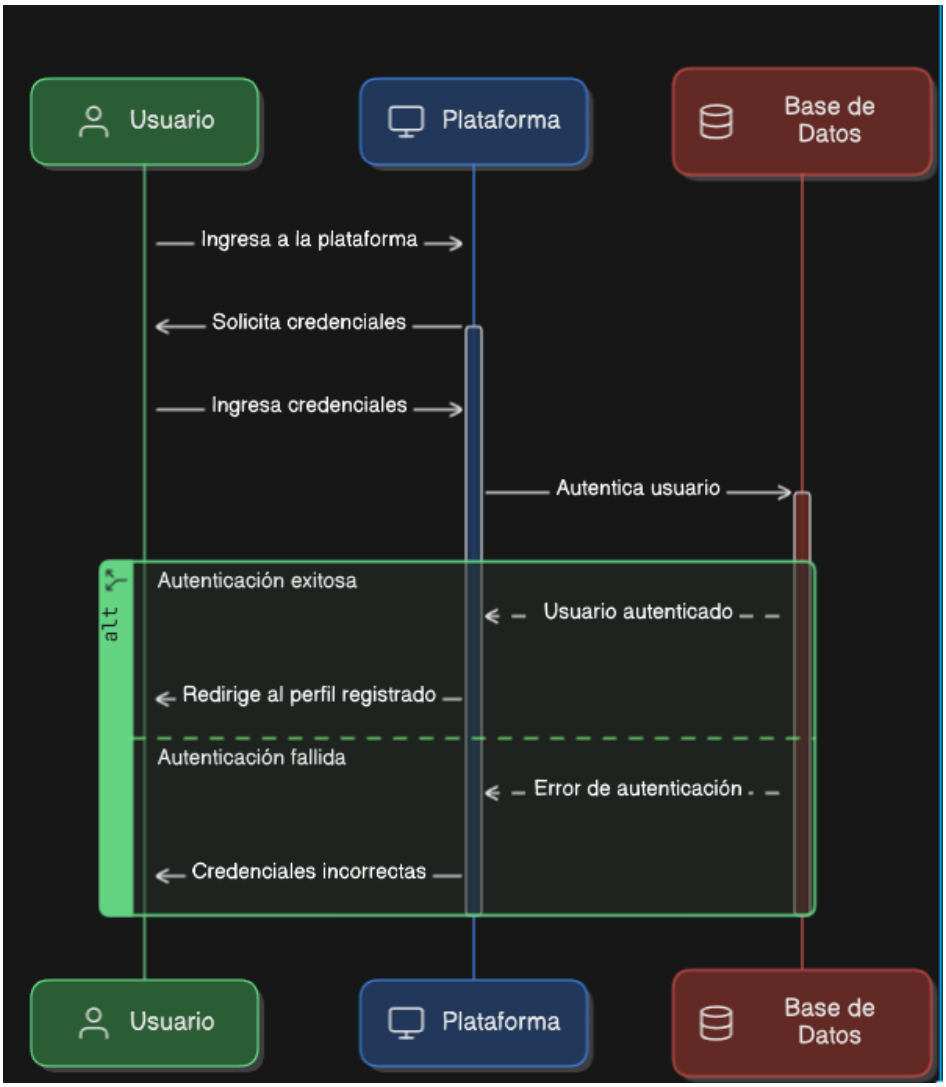
5.1.4 *diagramas de secuencia*

los diagramas de flujo nos ayudan a entender cuál es la secuencia de los procesos de la plataforma, mediante los mensajes entre los involucrados en el proceso que se está ejecutando, también nos da una mejor vista de los casos de uso ya que son un complemento de ellos.

5.1.4.1 Diagrama de secuencia del proceso de autenticación

Detalles del proceso

- ID del caso de uso: CU-01
- Nombre: Autenticación
- Detalle: El actor se autentica con sus respectivas credenciales



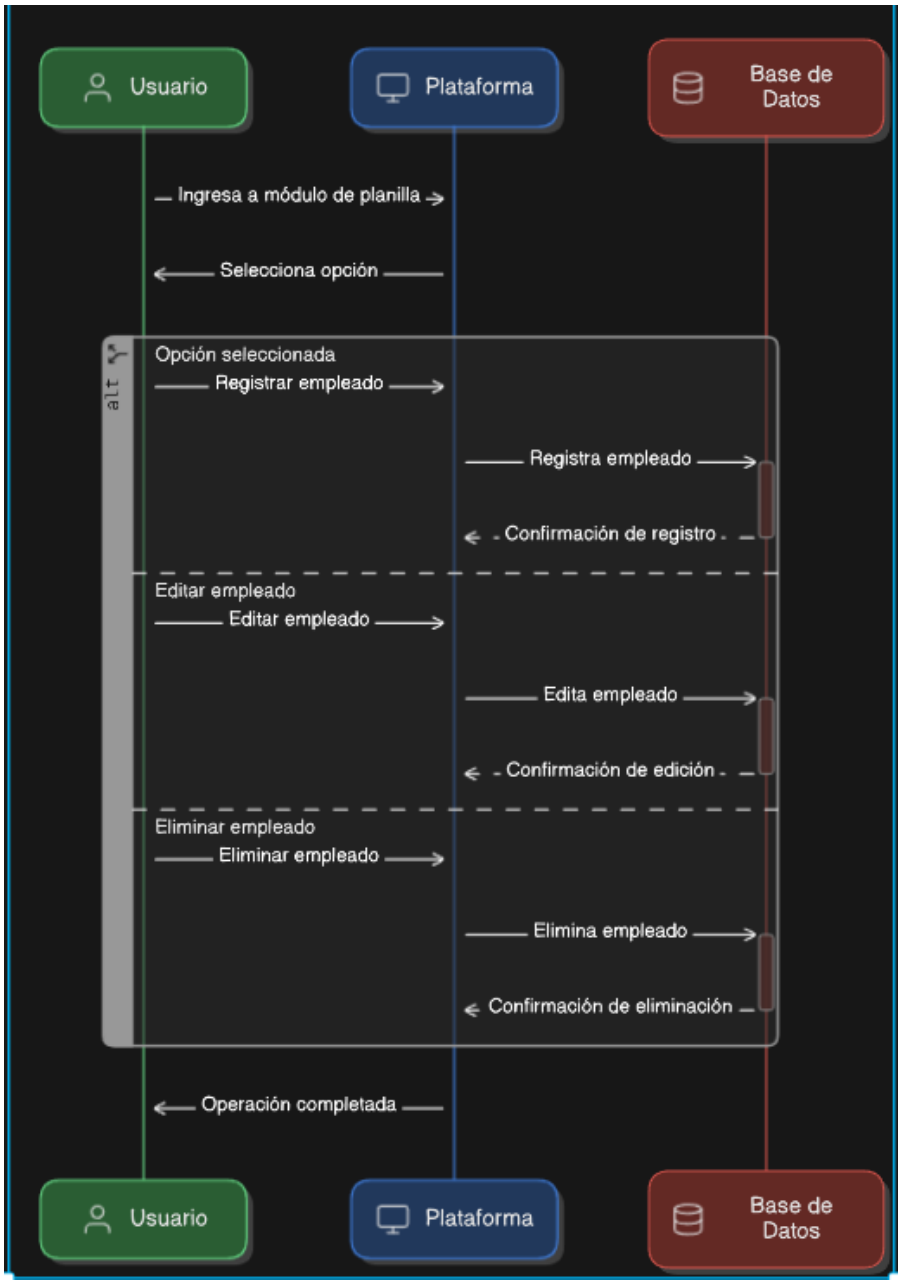
Fuente: Creación propia

Ilustración 15.Diagrama de flujo autenticación

5.1.4.2 Diagrama de secuencia del módulo de planilla

Detalles del proceso

- ID del caso de uso: CU-02
- Nombre: Modulo de planilla
- Detalle: El actor ingresa al módulo de planilla



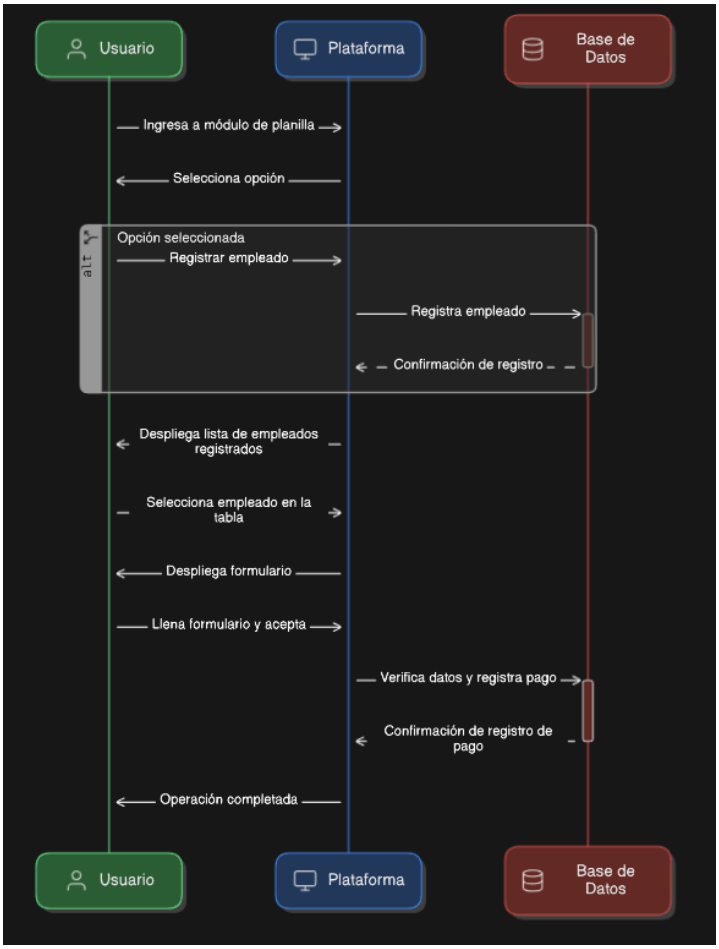
Fuente: Creación propia

Ilustración 16.Diagrama de secuencia de proceso de módulo de planilla

5.1.4.3 Diagrama de secuencia del proceso de pago de planilla

Detalles del proceso

- ID del caso de uso: CU-03
- Nombre: Módulo de pago de planilla
- Detalle: El actor ingresa al módulo de planilla y registra un pago a un cliente



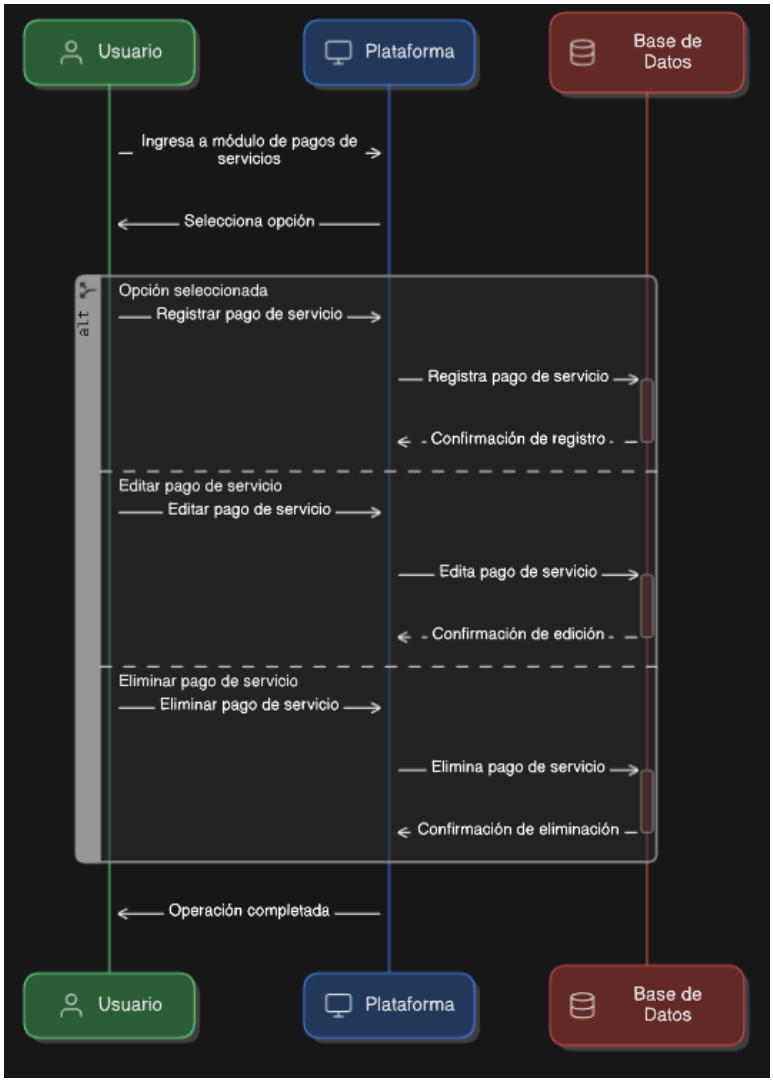
Fuente: Creación propia

Ilustración 17.Dagrama de secuencia de pago de planilla

5.1.4.4 Diagrama de secuencia del proceso de pagos de servicios

Detalles del proceso

- ID del caso de uso: CU-04
- Nombre: Módulo de pagos de servicio
- Detalle: El actor ingresa al módulo de pagos de servicios



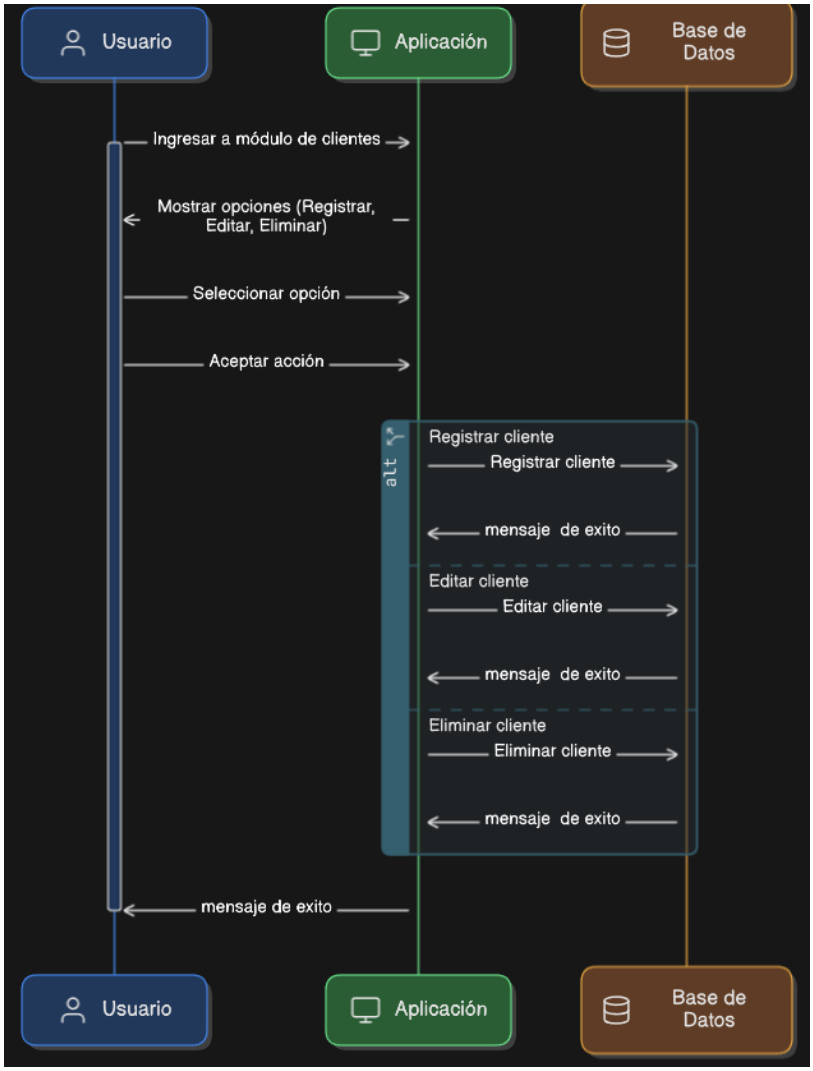
Fuente: Creación propia

Ilustración 18Diagrama de secuencia del módulo de pagos de servicio

5.1.4.5 Diagrama de secuencia del módulo de clientes

Detalles del proceso

- ID del caso de uso: CU-05
- Nombre: Módulo de Clientes
- Detalle: El actor ingresa al módulo de clientes



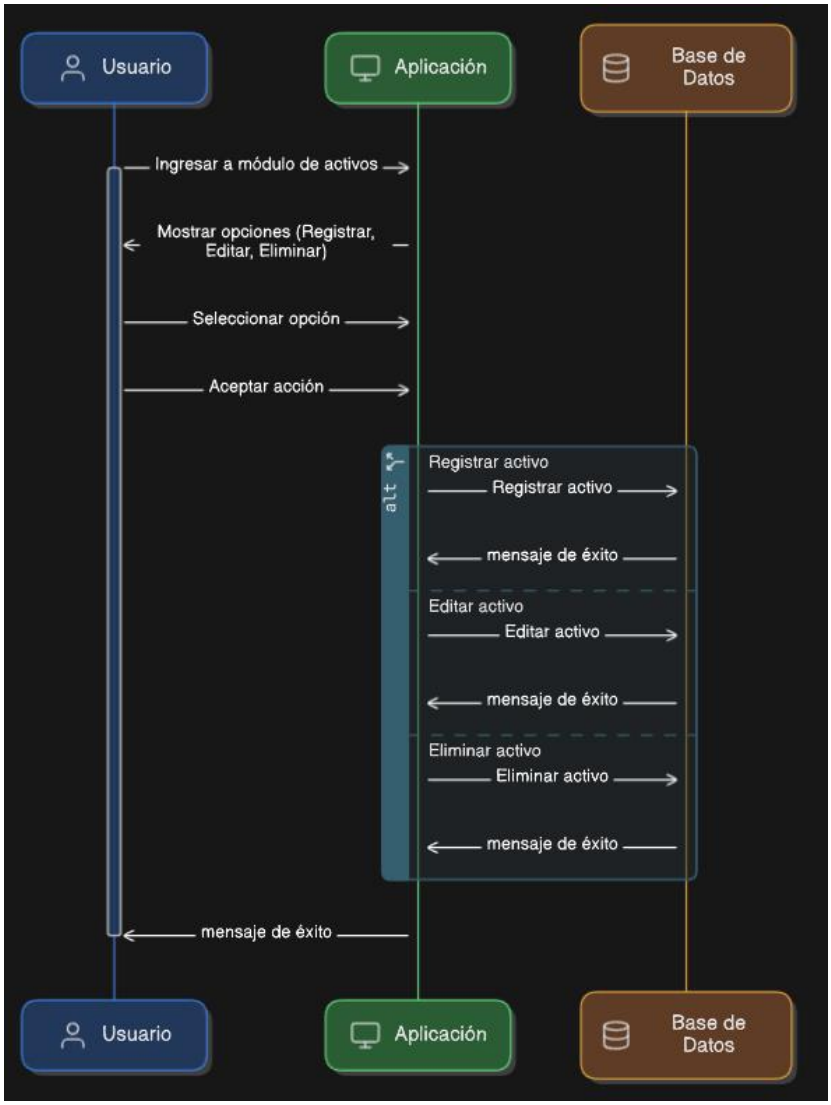
Fuente: Creación Propia

Ilustración 19: Diagrama de secuencia del módulo de clientes

5.1.4.6 Diagrama de secuencia del módulo de activos

Detalles del proceso

- ID del caso de uso: CU-06
- Nombre: Módulo de activos
- Detalle: El actor ingresa al módulo de activos



Fuente: Creación Propia

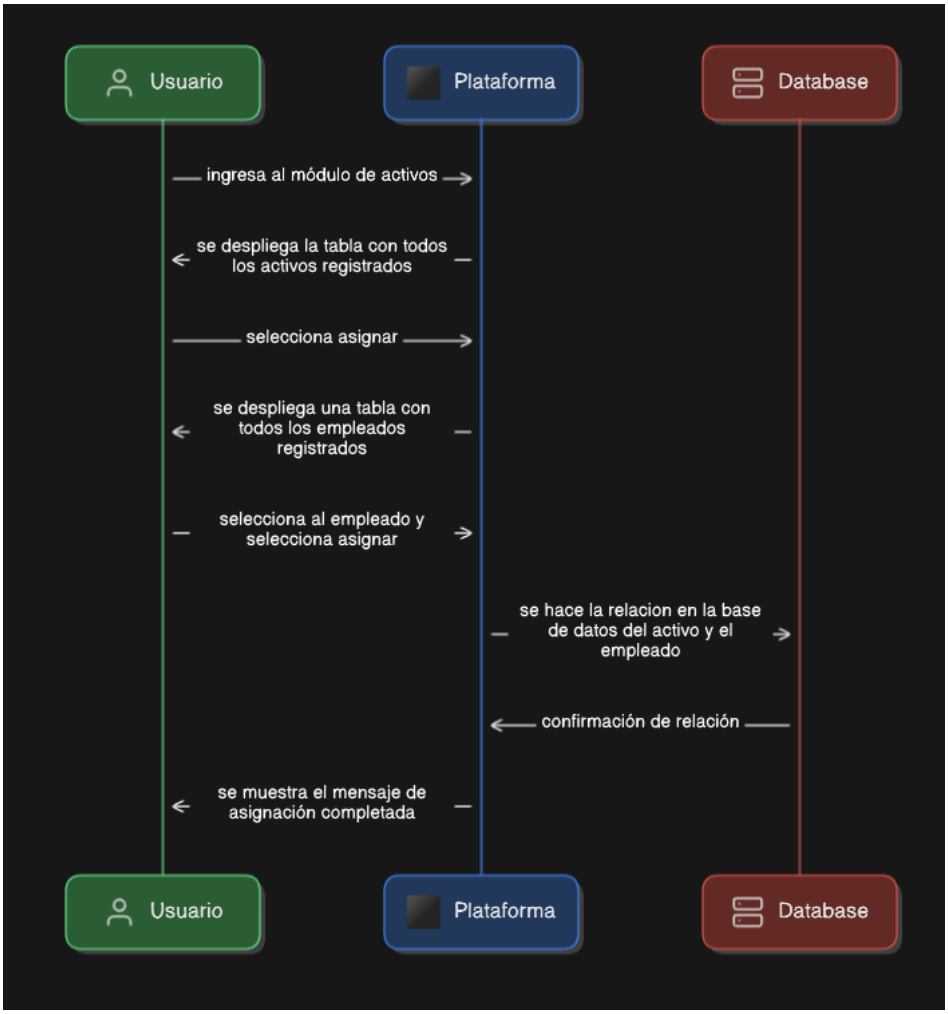
Ilustración 20: Diagrama de secuencia del módulo de activos

5.1.4.7 Diagrama de secuencia del asignar activos

Detalles del proceso

- ID del caso de uso: CU-07
- Nombre: Asignar activo
- Detalle: El actor ingresa al módulo de activos y asigna un activo

a un empleado previamente registrado



Fuente: Creación Propia

Ilustración 21: Diagrama de secuencia de asiganar un activo

5.1.4.8 Diagrama de secuencia de desasignar activos

Detalles del proceso

- ID del caso de uso: CU-08
- Nombre: Desasignar activo
- Detalle: El actor ingresa al módulo de activos y desasigna un activo a un empleado previamente registrado

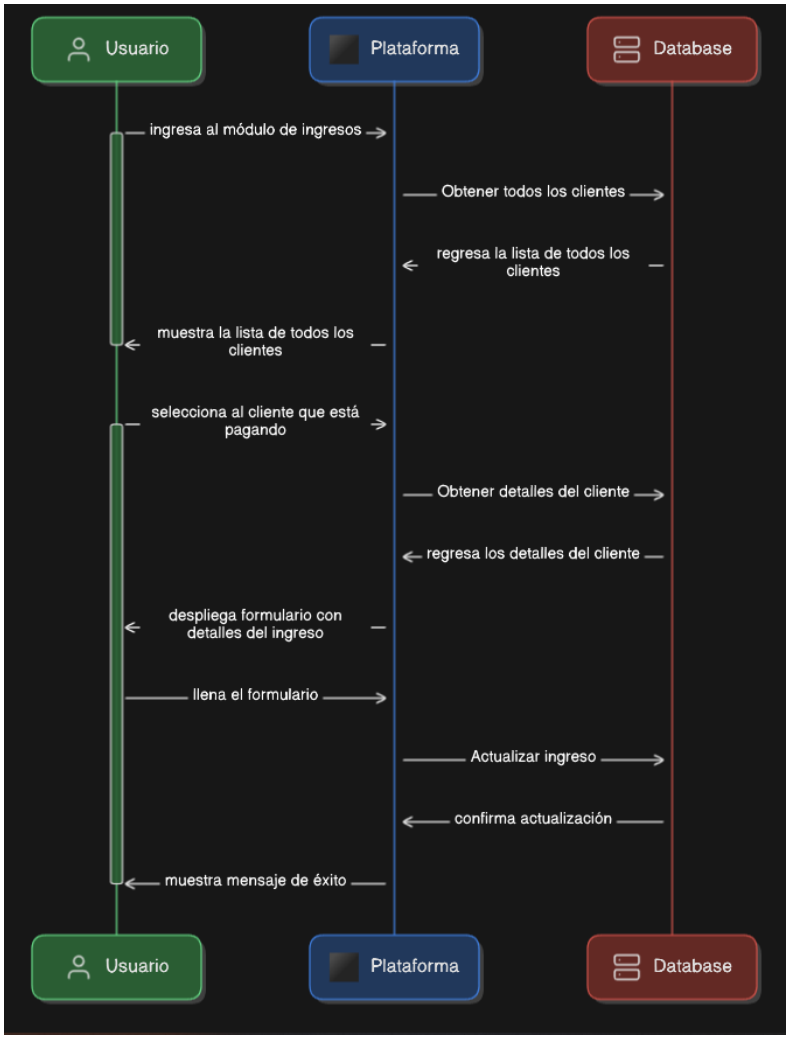
Fuente: Creación Propia

Ilustración 22: Diagrama de secuencia de desasignar un activo

5.1.4.9 Diagrama de secuencia de Ingresos

Detalles del proceso

- ID del caso de uso: CU-09
- Nombre: el actor ingresa al módulo de ingresos
- Detalle: El actor ingresa al módulo de registro de ingresos



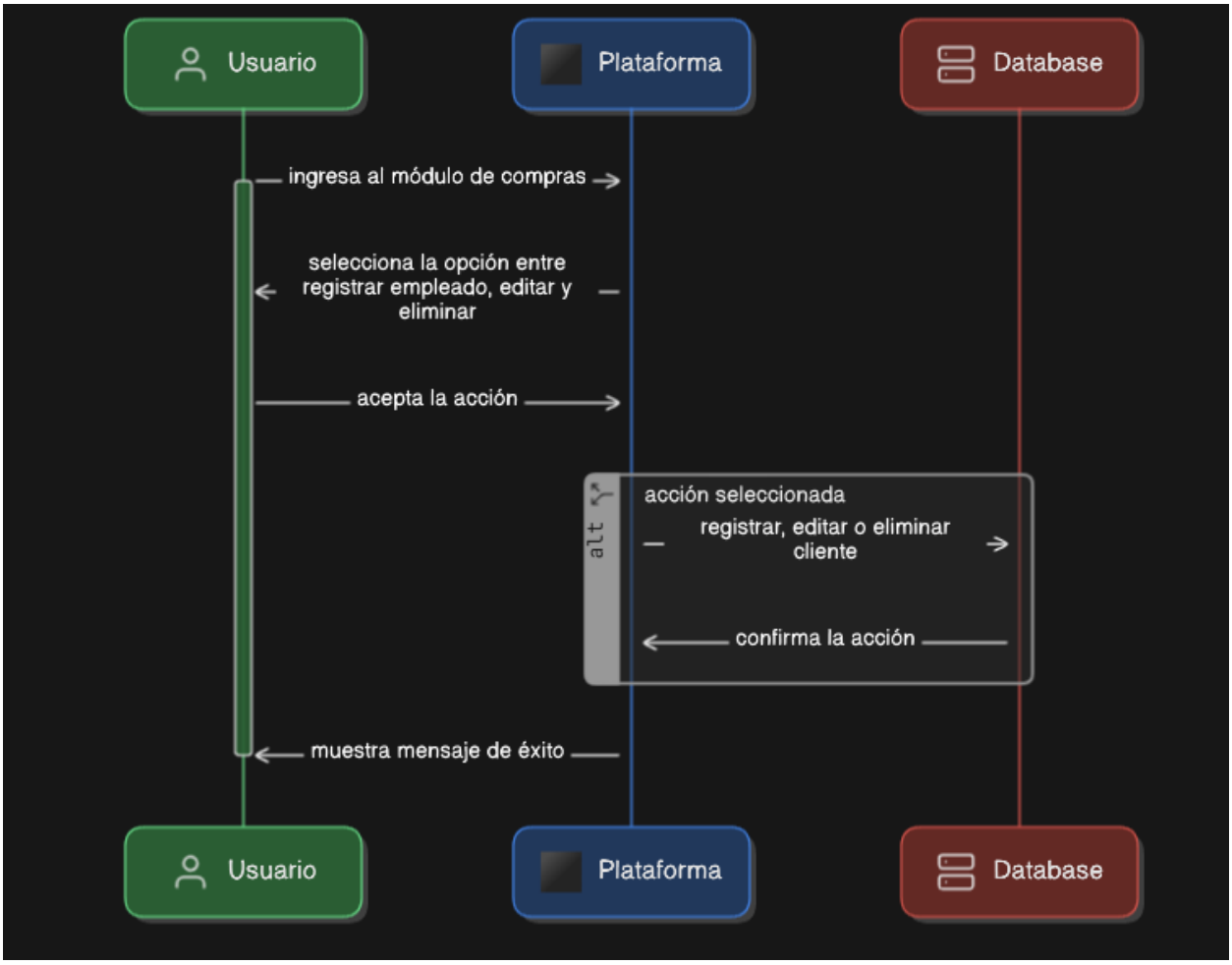
Fuente: Creación Propia

Ilustración 23: Diagrama de secuencia de registro de ingreso

5.1.4.10Diagrama de secuencia de módulo de compras

Detalles del proceso

- ID del caso de uso: CU-10
- Nombre: el actor ingresa al módulo de compras
- Detalle: El actor ingresa al módulo de compras



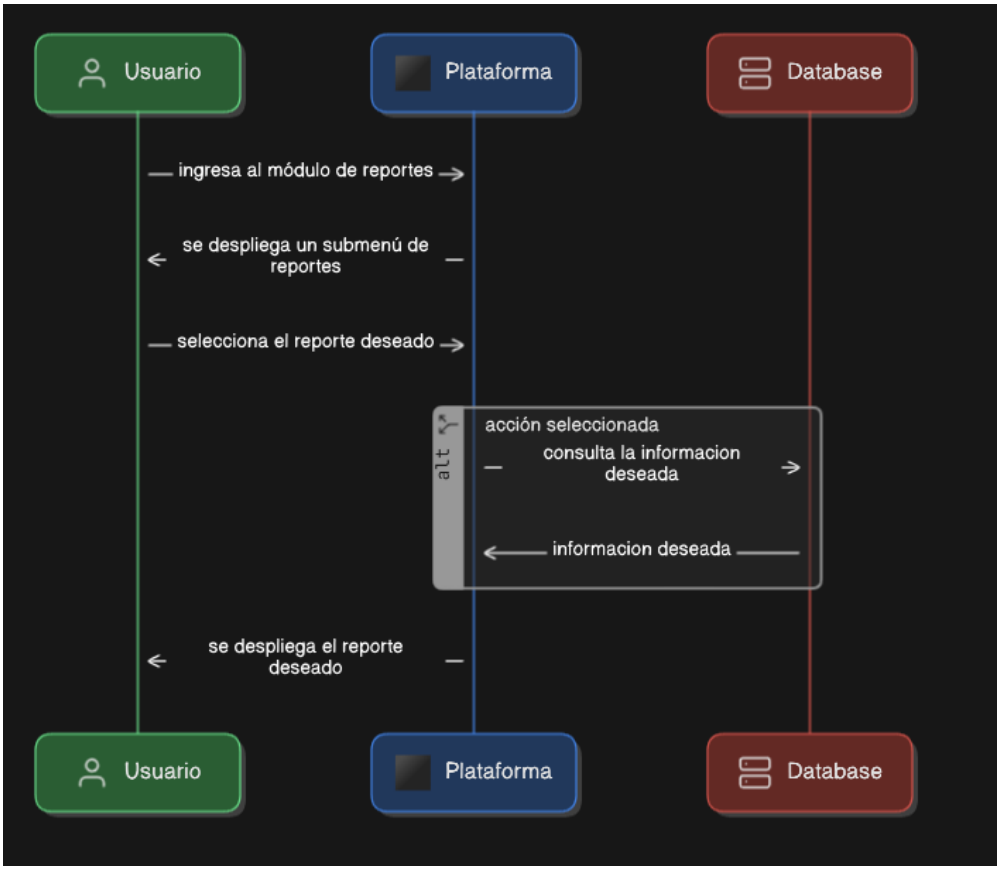
Fuente: Creación Propia

Ilustración 24: Diagrama de secuencia del módulo de compras

5.1.4.11Diagrama de secuencia de módulo de reportes

Detalles del proceso

- ID del caso de uso: CU-11
- Nombre: el actor ingresa al módulo de reportes
- Detalle: El actor ingresa al módulo de reportes y elige una opción



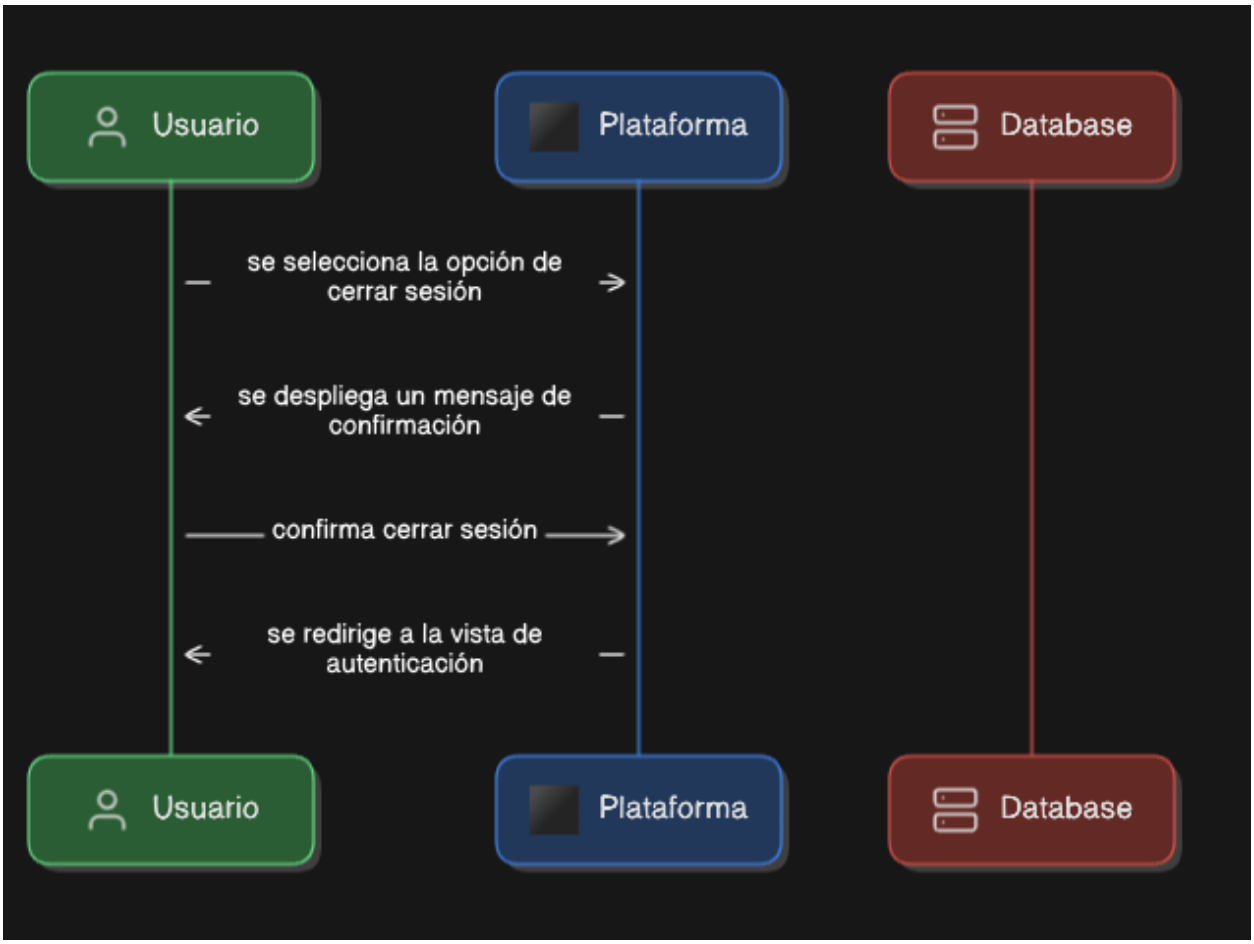
Fuente: Creación Propia

Ilustración 25:Diagrama de secuencia del módulo de reportes

5.1.4.12 Diagrama de secuencia de cierre de sesión

Detalles del proceso

- ID del caso de uso: CU-12
- Nombre: Cerrar sesión
- Detalle: el actor cierra la sesión

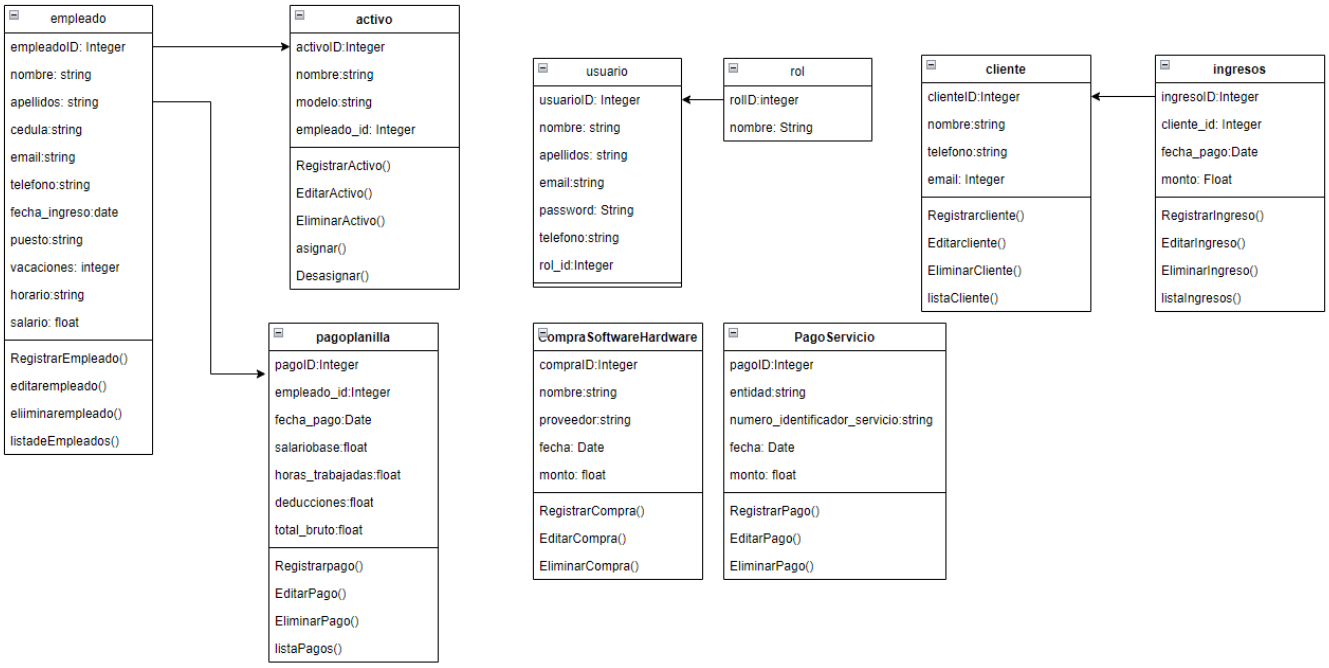


Fuente: Creación Propia

Ilustración 26: Diagrama de secuencia de cerrar sesión

5.1.5 *Diagrama de clases*

Se crea el diagrama de clases para obtener un mejor entendimiento de la aplicación mediante la creación de las clases con sus débitos atributos y sus métodos importantes, para reconocer cual es la interacción que tiene con las otras clases.



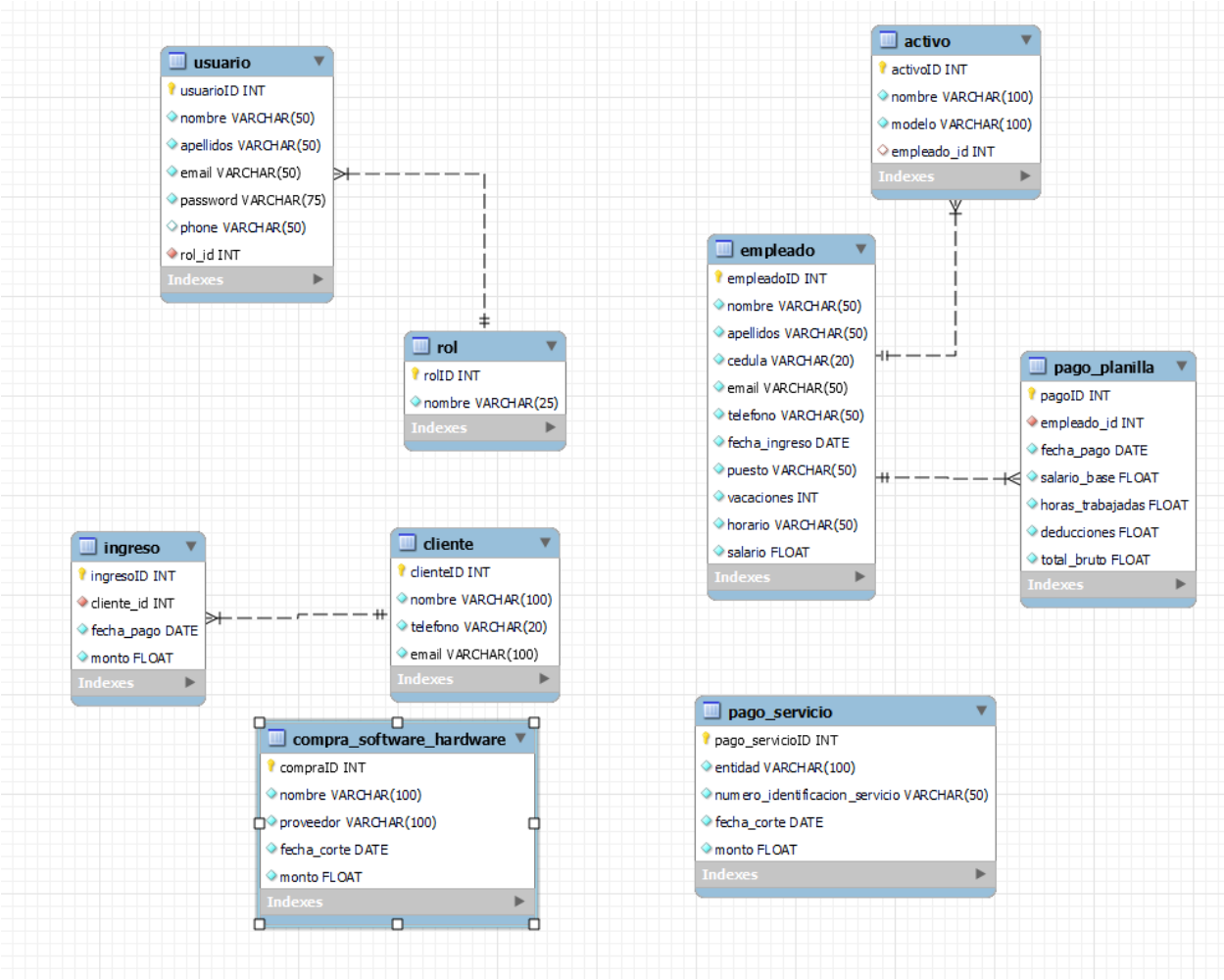
Fuente: Creación Propia

Ilustración 27:Diagrama de clases

5.1.6 *Diseño de la base de datos*

para el diseño de la base datos se diseñó el diagrama de entidad relación para saber cómo se almacenarán los datos de la aplicación mediante tablas y atributos y sus relaciones con otras tablas. Esto es muy importante porque podemos ver la estructura de las tablas y poder diseñar la base de datos de una manera más estructurada y entendible.

5.1.6.1 Diagrama entidad relación



Fuente: creación Propia

Ilustración 28: Diagrama de entidad relación

5.2 Desarrollo

En esta etapa de desarrollo, nos enfocamos en la implementación del proyecto basándonos en los diagramas previamente elaborados, enfocándonos en la creación del código del BackEnd y FrontEnd para visualizar nuestra aplicación, ya que uno de los requerimientos es crear una aplicación web de gestor de datos con la finalidad de cumplir y concluir esta etapa.

Como nos enfocamos en la metodología de desarrollo de incremental, completamente adaptable para mi persona, durante esta fase, de incrementos en los cuales se trabajó para finalizar el proyecto llevando una estructura clara de trabajo, para así obtener información sobre los avances en la aplicación.

5.2.1 Incremento 1: Creación del API

En este incremento nos enfocamos en la creación del API para la aplicación, la cual se desarrolló en Python con la utilización de tecnologías como Flask y del ORM SQLAlchemy.

5.2.1.1 Creación de la base de datos y endpoints

Para la creación de la base de datos, se utiliza el ORM en un .py para la generación de la base de datos en el sistema de gestión de base de datos de MYSQL.

```
1  from flask_sqlalchemy import SQLAlchemy
2  from flask_marshmallow import Marshmallow
3  from flask_jwt_extended import JWTManager
4  from flask_cors import CORS
5
6  crs = CORS()
7  jwt = JWTManager()
8  db = SQLAlchemy()
9  ma = Marshmallow()
```

Fuente: Creación Propia

Ilustración 29:Importacion de SQLAlchemy

models > empleado.py > Empleado

```python
from utils.db import db, ma


class Empleado(db.Model):
    empleadoID = db.Column(db.Integer, primary_key=True, autoincrement=True)
    nombre = db.Column(db.String(50), nullable=False)
    apellidos = db.Column(db.String(50), nullable=False)
    cedula = db.Column(db.String(20), nullable=False, unique=True)
    email = db.Column(db.String(50), nullable=False, unique=True)
    telefono = db.Column(db.String(50), nullable=False)
    fecha_ingreso = db.Column(db.Date, nullable=False)
    puesto = db.Column(db.String(50), nullable=False)
    vacaciones = db.Column(db.Integer, nullable=False)
    horario = db.Column(db.String(50), nullable=False)
    salario = db.Column(db.Float, nullable=False)
    pagos_planilla = db.relationship('PagoPlanilla', backref='Empleado', lazy=True)

    def __init__(self, nombre, apellidos, cedula, email, telefono, fecha_ingreso, puesto, vacaciones, horario, salario):
        self.nombre = nombre
        self.apellidos = apellidos
        self.cedula = cedula
        self.email = email
        self.telefono = telefono
        self.fecha_ingreso = fecha_ingreso
        self.puesto = puesto
        self.vacaciones = vacaciones
        self.horario = horario
        self.salario = salario
```

Fuente: Creación Propia

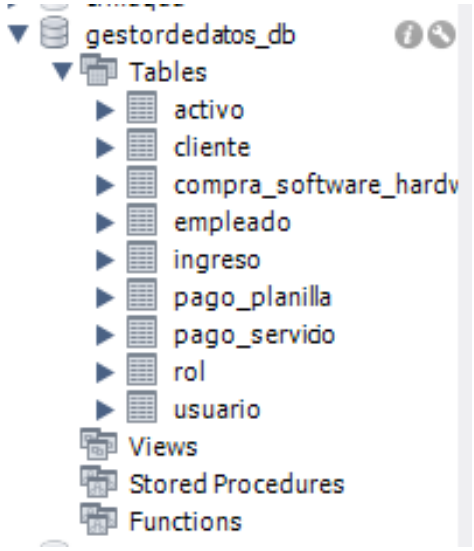
Ilustración 30: Clase de empleado

app.py

```python
from flask import Flask
from routes.usuarios import usuariosbp
from routes.roles import rolsbp
from routes.empleados import empleadosbp
from routes.activos import activosbp
from routes.pagosplanillas import pagos_planillasbp
from routes.clientes import clientesbp
from routes.ingresos import ingresosbp
from routes.comprassoftwarehardware import comprasSHbp
from routes.pagosServicios import pagos_serviciosbp


app = Flask(__name__)
#creacion de la key para el token
app.config["JWT_SECRET_KEY"] = "secret key" #camabiar el secret key
#conexion a la base de datos
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:123456@localhost/gestordedatos_db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False


app.register_blueprint(usuariosbp)
app.register_blueprint(rolsbp)
app.register_blueprint(empleadosbp)
app.register_blueprint(activosbp)
app.register_blueprint(pagos_planillasbp)
app.register_blueprint(clientesbp)
app.register_blueprint(ingresosbp)
app.register_blueprint(pagos_serviciosbp)
app.register_blueprint(comprasSHbp)
```

Fuente: Creación Propia

Ilustración 31conexión de base de datos



Fuente: Creación Propia

Ilustración 32: Base de datos

El ORM mediante la conexión de la base de datos y la creación de clases, al iniciar la aplicación, crea la base de datos en MYSQL.

empleados.py

routes > empleados.py > registro_empleado

```python
from flask import Blueprint, request, jsonify, send_from_directory
from utils.db import db
from models.empleado import Empleado
from flask_jwt_extended import create_access_token, jwt_manager
import os

empleadosbp = Blueprint('Empleados',__name__)

#enpoint para registrar un nuevo empleado en la base de datos
@empleadosbp.route('/registro_empleado', methods=['POST'])
def registro_empleado():
    try:
        #se obtiene la informacion del nuevo empleado
        data = request.get_json()
        #se verifica que los campos no lleguen vacios
        campos_requeridos = ['nombre','apellidos','cedula',
                             'email','telefono','fecha_ingreso','puesto',
                             'vacaciones','horario','salario']
        for campo in campos_requeridos:
            if campo not in data:
                return jsonify({'message': f'Falta el campo requerido: {campo}'}), 400
        #se crea una nueva instancia del nuevo empleado con sus datos
        nuevo_empleado = Empleado(
            nombre = data['nombre'],
            apellidos = data ['apellidos'],
            cedula = data ['cedula'],
            email = data ['email'],
            telefono= data ['telefono'],
            fecha_ingreso = data ['fecha_ingreso'],
            puesto = data['puesto'],
            vacaciones = data ['vacaciones'],
            horario = data['horario'],
            salario = data ['salario'],
        )

        #se agrega el nuevo empleado a la db
        db.session.add(nuevo_empleado)
        db.session.commit()
        return jsonify({'message': 'Empleado registrado exitosamente'}), 200
    except Exception as e:
        return jsonify({'message': f'Error durante el registro: {str(e)}'}), 500
```

Fuente: Creación Propia

Ilustración 33: Endpoint Empleados

Para la finalización, se crean los endpoints con sus respetivos métodos POST, GET, PUT, DELETE

5.2.2 Incremento 2: Autenticación

En este incremento, nos enfocamos en la autenticación de la aplicación para proporcionar la seguridad de que solo los usuarios que están registrados pueden acceder e interactuar con la plataforma. Además, implementamos el uso de tokens para manejar los usuarios, proporcionando así el nivel de seguridad requerido según los requerimientos.

usuarios.py

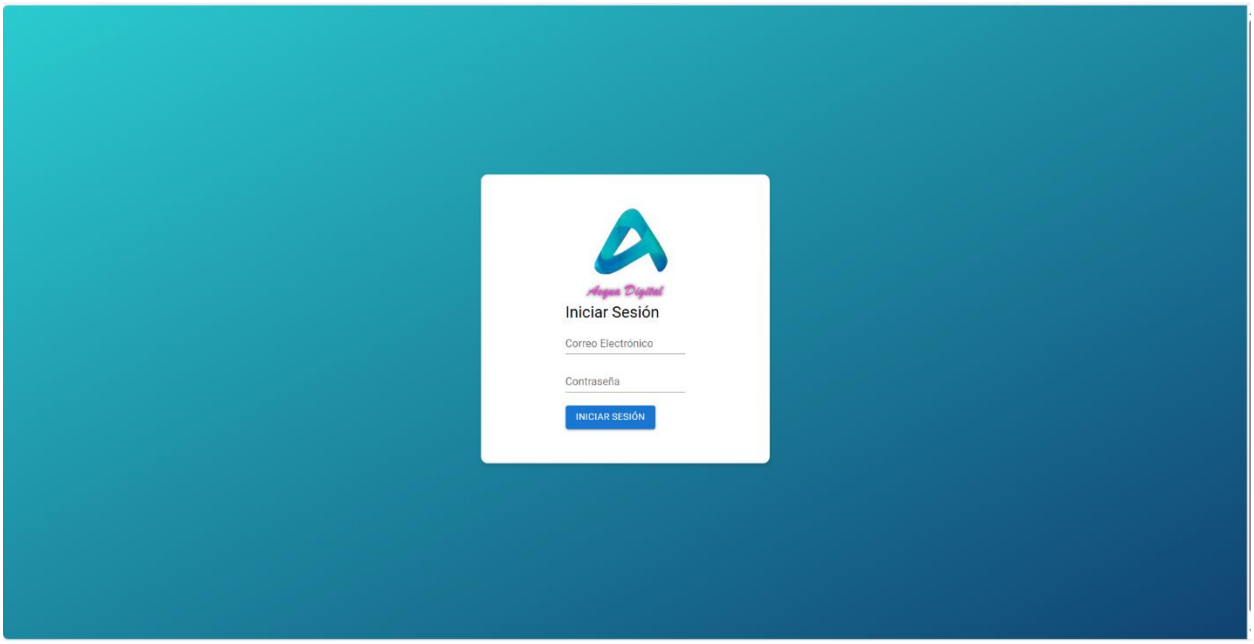
routes > usuarios.py > login

```python
from flask import Blueprint, request, jsonify, send_from_directory
from utils.db import db
from models.usuario import Usuario
from models.role import Rol
from datetime import datetime,timedelta
from sqlalchemy import text
from flask_jwt_extended import create_access_token, jwt_manager
import os

usuariosbp = Blueprint('Usuarios',__name__)

@usuariosbp.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    email = data['email']
    password = data['password']
    # Buscar al usuario en la base de datos por email
    user = Usuario.query.filter_by(email=email).first()
    if user and user.check_password(password):
        rolid = user.rol_id
        rol_nombre = Rol.query.filter_by(rolID=rolid).value(text('nombre'))
        # Obtener la fecha y hora actual
        current_time = datetime.utcnow()
        # Definir la duración de expiración
        expiration_duration = timedelta(minutes=120)

        # Calcular la fecha de expiración sumando la duración a la fecha y hora actual
        expiration = current_time + expiration_duration

        # Contraseña válida, el usuario se autentica exitosamente
        access_token = create_access_token(identity=user.usuarioID, expires_delta=timedelta(minutes=120), additional_claims={'role': rol_nombre, 'name':user.nombre}) #
        print(rol_nombre)
        response = jsonify({'message': 'Inicio de sesión de usuario exitoso', 'rol':  rol_nombre, 'token':access_token})
        response.headers['Authorization'] = f'Bearer {access_token}'  # Agregar el token al encabezado
        return response
    else:
        return jsonify({'message': 'Credenciales incorrectas'}), 401
```

Fuente: Creación Propia

Ilustración 34:Código Login

Se genera una página para el control del acceso a la plataforma, utilizando MUI y CSS con JavaScript para manejar los datos del lado del cliente y enviarlos al lado del servidor.



Fuente: Creación Propia

Ilustración 35:Vista Login

Con los datos correctos, se ingresa a la plataforma autenticando que rol se tiene, se muestra la vista dependiendo del rol.

Fuente: Creación Propia

Ilustración 36:Vista Bienvenida

5.2.3 Incremento *3: Creación del Módulo de Planilla*

Se crea el módulo de planilla con sus respectivos POST, PUT, DELETE para completar el CRUD necesario para cumplir con los requerimientos en la parte del BackEnd.

En la parte del FrontEnd se crea el módulo de planilla donde están las opciones para registrar editar y eliminar el empleado y también para registrar un pago al empleado en específico.

Para ver parte del código del endpoint que pertenece al módulo de planilla ver ilustración 3: Endpoint Empleados.

Fuente: Creación propia

Ilustración 37:Modulo Planilla

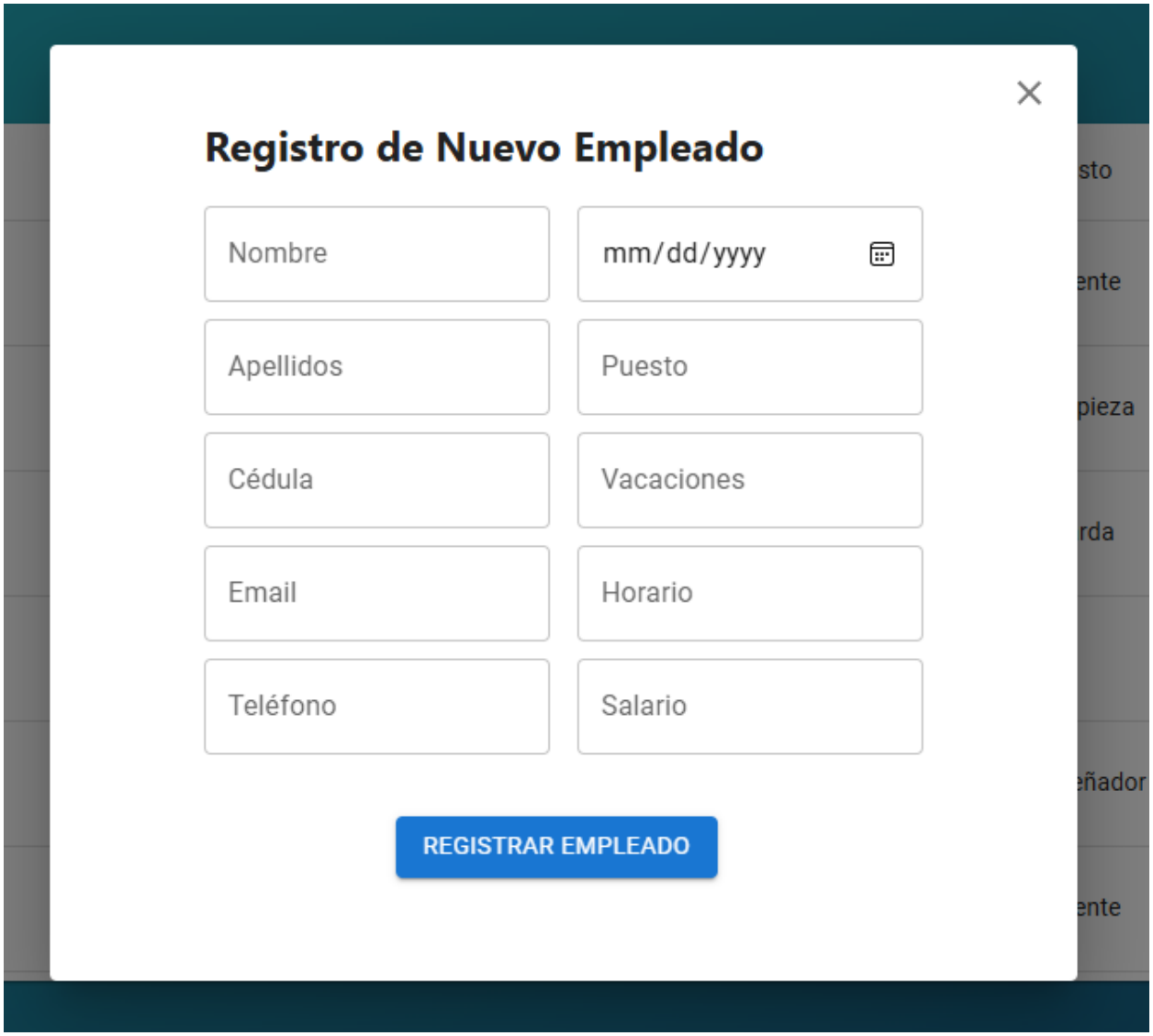


Ilustración 38:Registrar empleado

Fuente: Creación Propia

5.2.4 Incremento *4: creación del módulo de pagos de servicios*

Se crean los endpoint que corresponden a este módulo, así como su vista al lado del cliente para poder tener la interacción entre el usuario y la aplicación, utilizando los

métodos GET, POST, PUT, DELETE. Se carga la lista de todos los pagos de servicios realizados desde el lado del servidor para mostrarlos en una tabla del lado del cliente para que el usuario interactúe con ella, esta tabla tiene las opciones para registrar, editar y eliminar un pago de servicio.

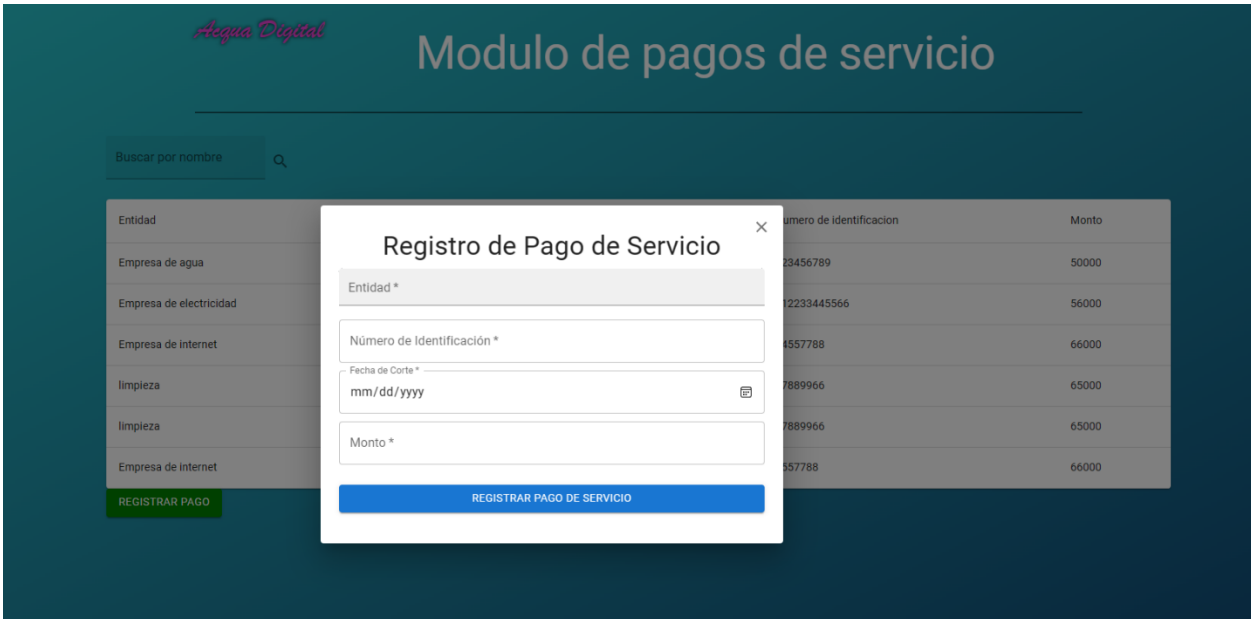
Fuente: Creación Propia

Ilustración 39: Módulo de pagos de servicios

```python
@pagos_serviciosbp.route('/Lista_pagos_servicios', methods=['GET'])
def Lista_pagos_servicios():
    try:
        # Lógica para obtener la lista los pagos de servicios
        lista = PagoServicio.query.all()

        if not lista:
            return jsonify({'mensaje': 'No hay pagos de servicios registrados en la base de datos'}), 200

        # Convierte cada pago a un formato serializable y se crea la lista
        lista_serializables = [
            {   'pago_servicioID': pagoservicio.pago_servicioID,
                'entidad': pagoservicio.entidad,
                'numero_identificacion_servicio': pagoservicio.numero_identificacion_servicio,
                'fecha_corte': pagoservicio.fecha_corte,
                'monto': pagoservicio.monto
            }
            for pagoservicio in lista
        ]
        #se devuelve  la lista en formato json
        return jsonify(lista_serializables), 200

    except Exception as e:
        return jsonify({'error': f'Error al obtener la lista de pagos de servicios: {str(e)}'}), 500
```

Fuente: Creación Propia

Ilustración 40:código de lista de pagos de servicios



Fuente: Creación Propia

Ilustración 41: Registro de pago de servicio

5.2.5 Incremento *5: Modulo de Clientes*

Se crean los endpoint que corresponden a este módulo, así como su vista en el lado del cliente para poder tener la interacción entre el usuario y la aplicación, utilizando los métodos GET, POST, PUT, DELETE. Se carga la lista de todos los clientes registrados desde el lado del servidor para mostrarlos en una tabla del lado del cliente para que el usuario interactúe con ella, esta tabla tiene las opciones para registrar, editar y eliminar un cliente.

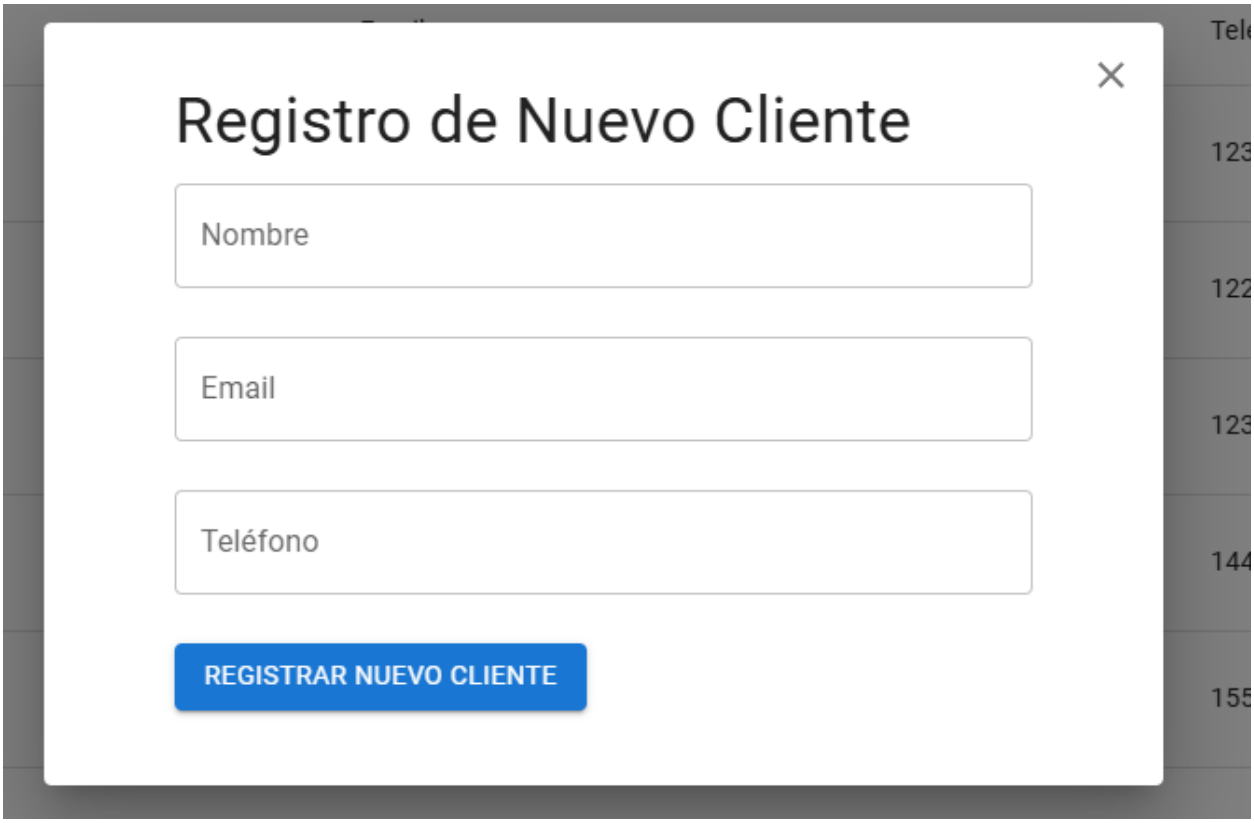
```python
#endpoint para obtener la lista completa de los clientes
@clientesbp.route('/Lista_Clientes', methods=['GET'])
def Lista_Clientes():
    try:
        # se obtienen todos los clientes de la base de datos
        clientes = Cliente.query.all()

        if not clientes:
            return jsonify({'mensaje': 'No hay clientes registrados en la base de datos'}), 200

        # Convierte cada cliente a un formato serializable y se crea la lista
        lista_serializada = [
            {
                'clienteID': cliente.clienteID,
                'nombre': cliente.nombre,
                'telefono': cliente.telefono,
                'email': cliente.email
            }
            for cliente in clientes
        ]
        #se devuelve la lista mediante un json
        return jsonify(lista_serializada), 200

    except Exception as e:
        return jsonify({'error': f'Error al obtener la lista de clientes: {str(e)}'}), 500
```

Fuente: Creación Propia

Ilustración 42: Endpoint Lista de Clientes

Fuente: Creación Propia

Ilustración 43: Vista de Modulo de Clientes



Fuente: Creación Propia

Ilustración 44: Vista de formulario de Registro de Cliente

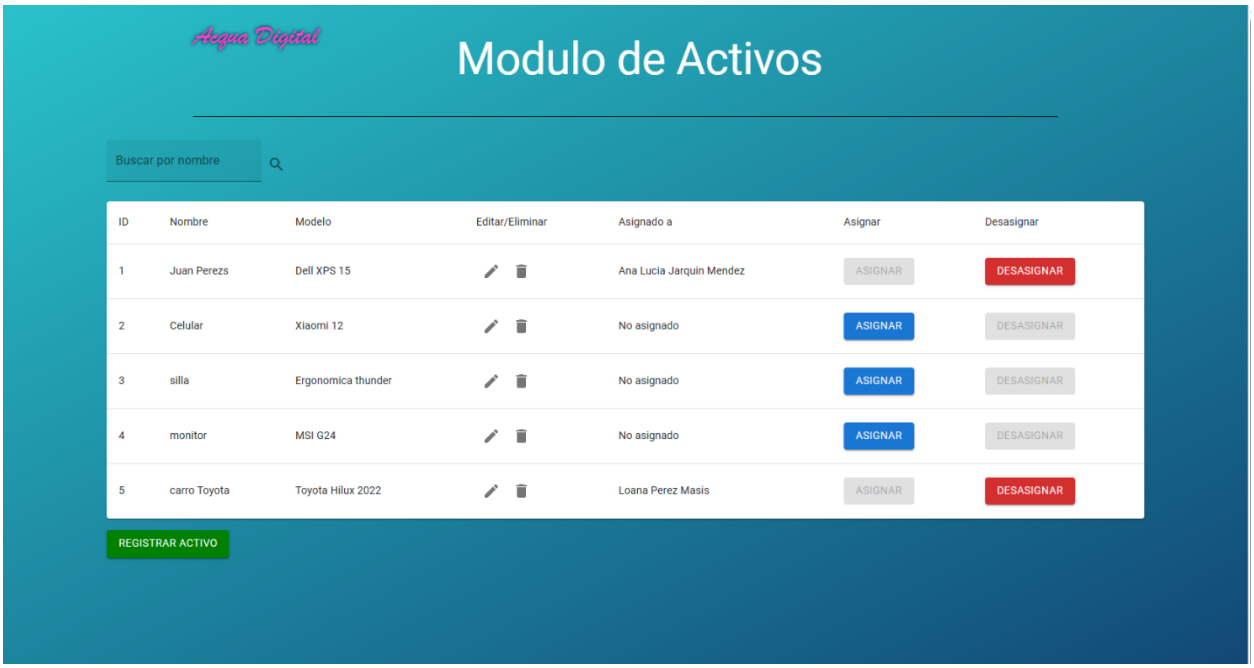
5.2.6 Incremento *6: Creación del Módulo de Activos*

Se crean los endpoint que corresponden a este módulo, así como su vista en el lado del cliente, para facilitar la interacción entre el usuario y la aplicación, utilizando los métodos GET, POST, PUT, DELETE. Se carga la lista de todos los activos registrados desde el lado del servidor para mostrarlos en una tabla del lado del cliente, para que el usuario interactúe con ella, esta tabla tiene las opciones para registrar, editar y eliminar un activo, así como también asignar un activo a un empleado o desasignar un activo a un empleado.

```python
#enpoint obtener la lista completa de los activos
@activosbp.route('/lista_activos', methods=['GET'])
def lista_activos():
    try:
        # Se realiza una unión entre la tabla de activos y la tabla de empleados
        activos_con_empleados = db.session.query(
            Activo.activoID,
            Activo.nombre,
            Activo.modelo,
            Empleado.nombre.label('nombre_empleado'),
            Empleado.apellidos.label('apellidos_empleado')
        ).join(Empleado, Activo.empleado_id == Empleado.empleadoID, isouter=True).all()

        #se verifica si no se encontraron activos
        if not activos_con_empleados:
            return jsonify({'mensaje': 'No hay activos en la base de datos'}), 200

        # se convierte cada activo a un formato serializable y crea la lista
        activos_serializables = [
            {
                'activoID': activo.activoID,
                'nombre': activo.nombre,
                'modelo': activo.modelo,
                'nombre_empleado': activo.nombre_empleado + ' ' + activo.apellidos_empleado if activo.nombre_empleado else None
            }
            for activo in activos_con_empleados
        ]

        return jsonify(activos_serializables), 200

    except Exception as e:
        return jsonify({'error': f'Error al obtener la lista de activos: {str(e)}'}), 500
```

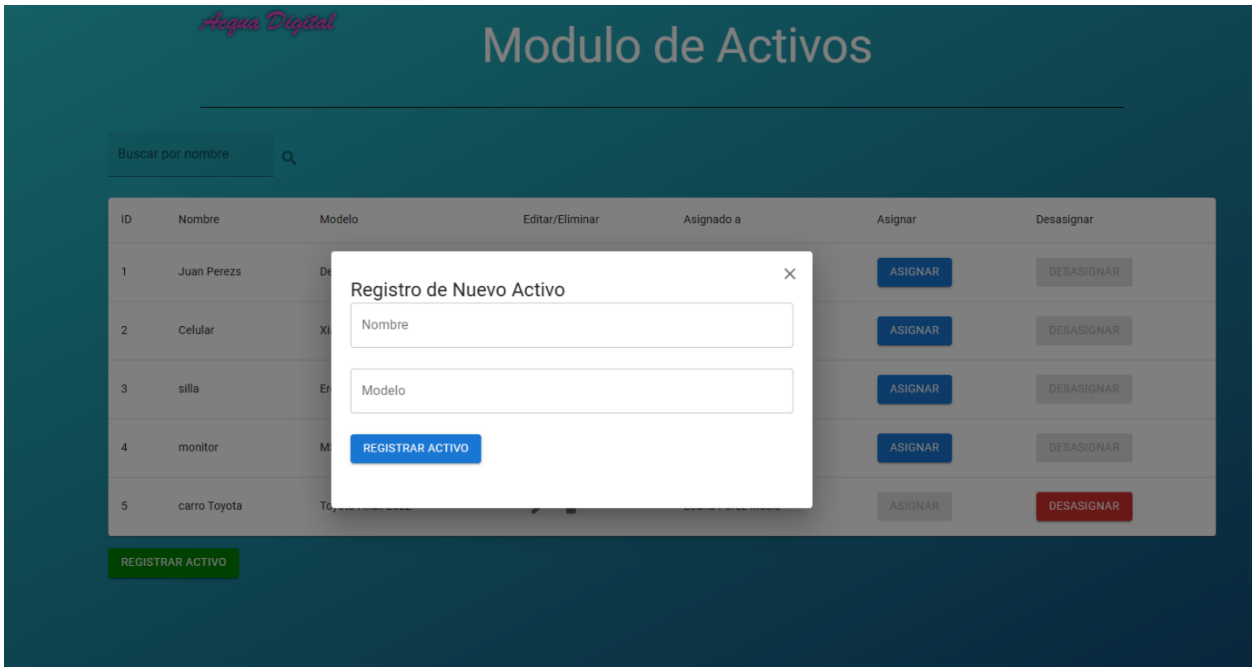
Fuente: Creación Propia

Ilustración 45: Endpoint de lista de activos



Fuente: Creación Propia

Ilustración 46: Vista del Módulo de activos



Fuente: Creación Propia

Ilustración 47: Formulario de Registro

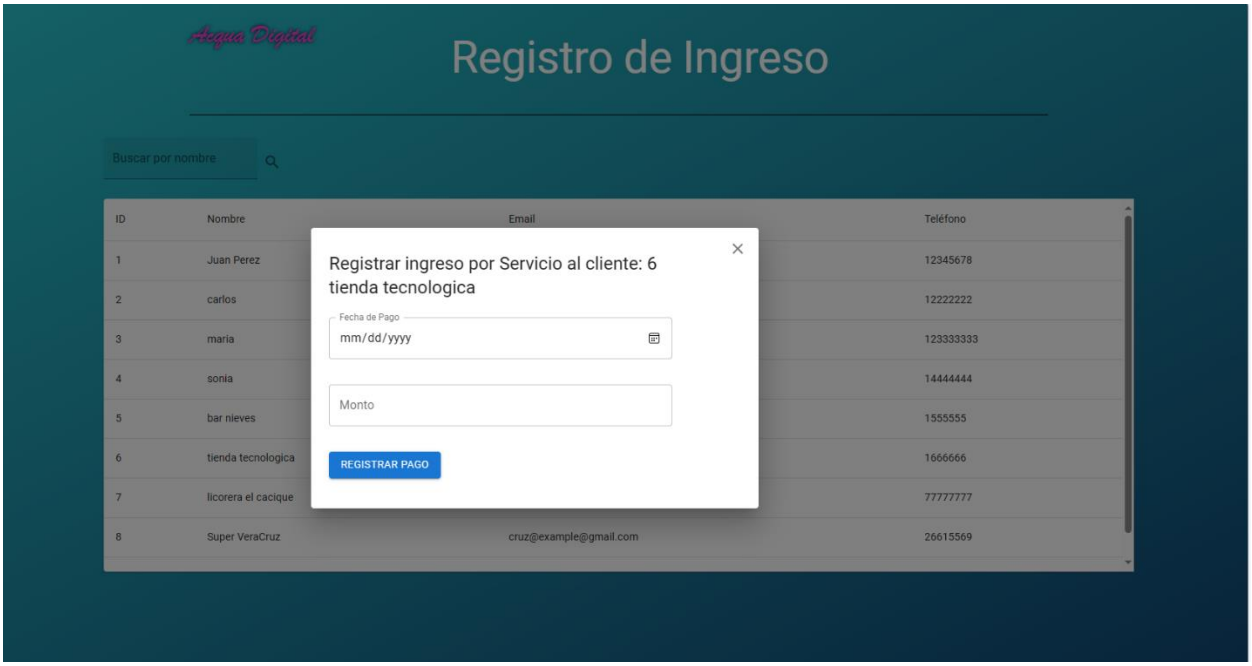
5.2.7 Incremento *7: Creación del Módulo de Ingresos*

Se crean los endpoint que corresponden a este módulo, así como su vista en el lado del cliente para poder tener la interacción entre el usuario y la aplicación. utilizando los métodos GET, POST, PUT, DELETE. Se carga la lista de todos los clientes registrados desde el lado del servidor para mostrarlos en una tabla del lado del cliente para que el usuario interactúe con ella, esta tabla tiene la particularidad de que el usuario debe seleccionar al cliente para registrar un ingreso que está generando ese cliente.

Para ver el ejemplo del código de lista de clientes, ver la ilustración 41.

Fuente: Creación Propia

Ilustración 48: Vista de Ingresos



Fuente: Creación Propia

Ilustración 49: Formulario de registro de ingreso

5.2.8 Incremento *8: creación del módulo de compras*

Se crean los endpoint que corresponden a este módulo, así como su vista en el lado del cliente para poder tener la interacción entre el usuario y la aplicación. utilizando los métodos GET, POST, PUT, DELETE. Se carga la lista de todas las compras registradas desde el lado del servidor para mostrarlos en una tabla del lado del cliente para que el usuario interactúe con ella, esta tabla tiene las opciones para registrar, editar y eliminar una compra.

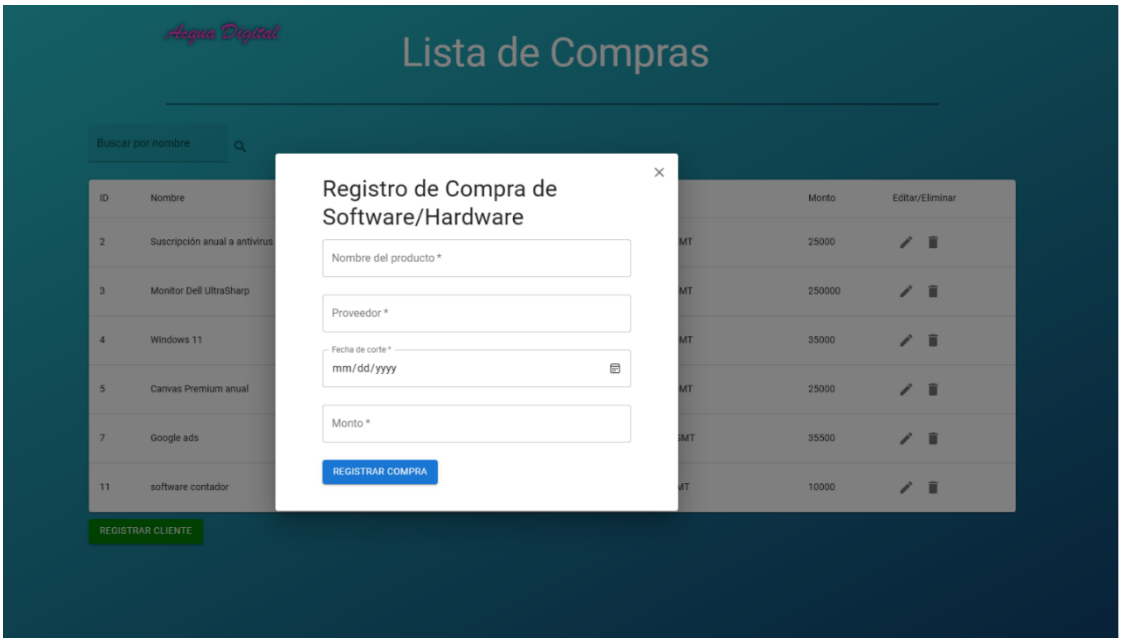
```python
#enpoint para obtener la lista de todas las compras realizadas
@comprasSHbp.route('/Lista_compras', methods=['GET'])
def Lista_pagos_servicios():
    try:
        # Lógica para obtener la lista de las compras realizadas
        lista = CompraSoftwareHardware.query.all()

        if not lista:
            return jsonify({'mensaje': 'No hay compras registrados en la base de datos'}), 200

        # Convierte cada compra a un formato serializable y se crea la lista
        lista_serializables = [
            {
                'compraID':compra.compraID,
                'nombre': compra.nombre,
                'proveedor': compra.proveedor,
                'fecha_corte': compra.fecha_corte,
                'monto': compra.monto
            }
            for compra in lista
        ]
        #se devuelve la lista de las compras en un json
        return jsonify(lista_serializables), 200
    except Exception as e:
        return jsonify({'error': f'Error al obtener la lista de pagos de servicios: {str(e)}'}), 500
```

Fuente: Creación Propia

Ilustración 50: Código Lista de Compras

Fuente: Creación Propia

Ilustración 51: Vista de Modulo de Compras



Fuente: creación Propia

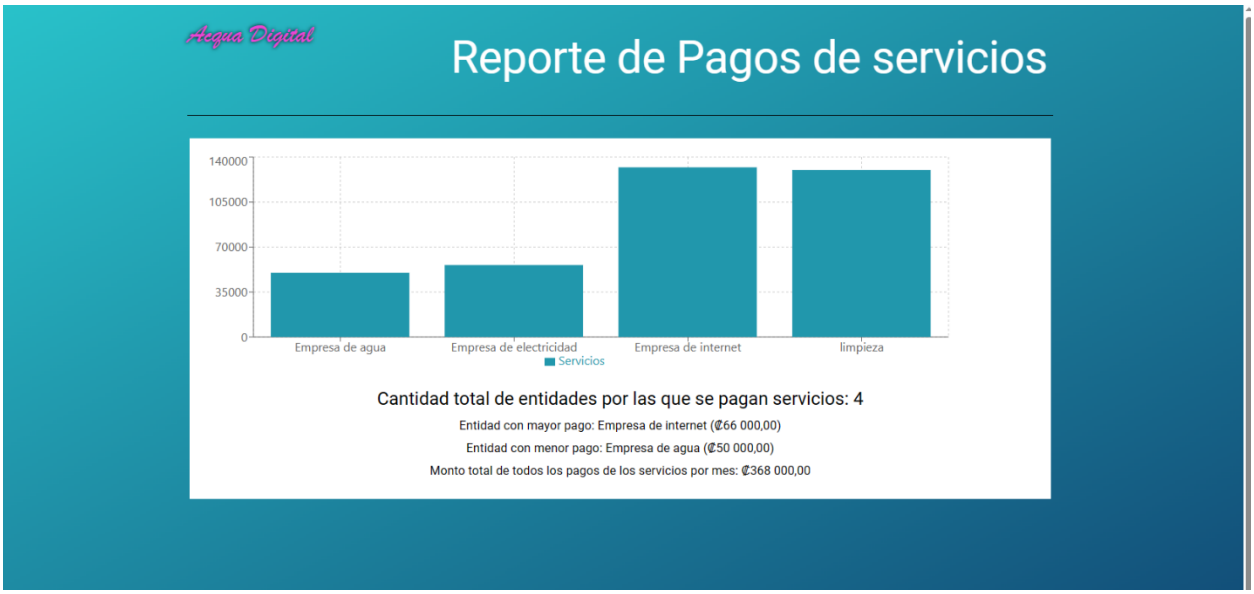
Ilustración 52: Formulario de Registro de Compra

5.2.9 Incremento *9: creación del Módulo de Reportes*

Se crea el módulo de reportes con sus opciones para visualizar informes basados en la información seleccionada por el usuario, esto permite tomar decisiones fundamentadas en datos, asegurando que decisiones sean más acertadas.

Fuente: Creación Propia

Ilustración 53:Reporte de Clientes e Ingresos



Fuente: Creación Propia

Ilustración 54:Reporte de Pagos de Servicios

Fuente: Creación Propia

Ilustración 55: Reporte de Planilla

5.3 Pruebas funcionales

Las pruebas es una parte importante del desarrollo de la aplicación, ya que nos hace ver los errores de la aplicación y depurarlos antes de terminal la implementación del software.

Se hacen pruebas funcionales para obtener información de si los requerimientos funcionales se están cumpliendo de manera exitosa. Esto nos confirma que se cumplen los requerimientos del sistema cumpliendo con las exigencias del cliente.

5.3.1 *Registro de Empleados (RF001):*

5.3.1.1 Prueba de ingreso de un nuevo empleado

Resultado**:** al ingresar todos los datos de manera correcta y completa en el formulario de registro. Se envió el formulario, el sistema proceso la información correctamente y el empleado fue registrado exitosamente en la base de datos. Se verifico que todos los campos estuvieran presentes y que los datos ingresados cumplieran con los criterios de validación establecidos. Se confirmo la creación exitosa del empleado mediante la visualización de un mensaje de éxito.

5.3.1.2 Prueba de modificación de los datos de un empleado existente:

Resultado: al mostrar el formulario de editar empleado al llenar los campos a editar, se envía el formulario al servidor que verifica que los campos estén llenos y aplica los cambios correspondientes y se guardan correctamente en la base de datos, si falta algún dato o no encuentra al empleado finaliza con error.

5.3.1.3 Prueba de eliminación de un empleado del sistema

Resultado: Al dar click al botón para eliminar el empleado se muestra un mensaje de confirmación al aceptar se envía la petición al servidor para consultar

si el empleado existe y elimina al empleado de la base de datos, arrojando un mensaje de que el empleado se eliminó correctamente siendo una prueba exitosa.

5.3.1.4 Prueba de validación de datos

Resultado: La prueba fue exitosa ya que el sistema valida que cada campo obligatorio este completado si no está, envía mensajes de error al igual que cuando se ingresan datos incorrectos

Al evaluar los resultados de las pruebas anteriores, podemos decir que el requerimiento RF001 relacionado con el registro de empleados ha sido completado con éxito. Las pruebas de ingreso, modificación y eliminación de empleados demostraron que el sistema funciona correctamente al registrar eliminar información de empleados, cumpliendo así con los criterios establecidos en los requerimientos funcionales.

5.3.2 *Registro de Pago de Planilla (RF002):*

5.3.2.1 Prueba de registro de un nuevo pago de planilla para un empleado especifico:

Resultado: se completaron todos los campos requeridos para registrar un nuevo pago para un empleado especifico de la planilla, incluyendo los campos requeridos. Al enviar el formulario, el sistema proceso la información correctamente y se registró el pago de la planilla en la base de datos. Se verifico que todos los datos ingresados fueran precisos y se reflejaran correctamente en la base de datos y se confirmó que el pago se realizó correctamente al empleado especifico.

5.3.2.2 Prueba de modificación de los detalles de un pago realizado ya existente:

Resultado: se accedió al formulario de edición de un pago existente y se realizaron las modificaciones en los detalles del pago. Al enviar los cambios, el sistema proceso la solicitud correctamente y actualizo los detalles del pago de planilla en la base de datos. Se verifico que las modificaciones realizadas se reflejaran correctamente en la interfaz de usuario y que los datos actualizados fueran precisos y consistentes con los cambios realizados.

5.3.2.3 Prueba de eliminación de un registro de pago del sistema:

Resultado: se seleccionó el registro de pago de planilla a eliminar y se confirmó la acción. El sistema proceso la solicitud de eliminación correctamente y elimino el registro seleccionado de manera exitosa de la base de datos. Se verifico que el registro eliminado ya no estuviera presente en la interfaz de usuario y que los datos asociados al mismo fueran eliminados de manera adecuada de la base datos.

Con estos resultados se confirma que el requerimiento R002, relacionado con el registro de pago de planilla, ha sido completado con éxito, demostrando el correcto funcionamiento de las funcionalidades asociadas con dicho registro.

5.3.3 *Registro de activos RF003:*

5.3.3.1 Prueba de ingreso de un nuevo activo con nombre y modelo

Resultado: se completaron los campos requeridos para registrar un nuevo activo, incluyendo nombre y modelo. Tras enviar el formulario, el sistema proceso la información correctamente y se registró el activo en la base de datos. Se

verifico que todos los datos ingresados fueran precisos y se reflejaran correctamente en la interfaz de usuario. Se confirmo que el activo se agregó correctamente al sistema con el nombre y modelo específicos.

5.3.3.2 Prueba de modificación de los detalles de un activo existente:

Resultado**:** se accedió al formulario de un activo existente y se realizaron las modificaciones en sus detalles. Al enviar los cambios, el sistema proceso la solicitud correctamente y actualizo los detalles del activo en la base de datos. Se verifico que las modificaciones realizadas se reflejaran en la interfaz de usuario y que los datos actualizados fueran precisos con los cambios realizados. Se confirmo que el activo fue modificado exitosamente con los nuevos detalles.

5.3.3.3 Prueba de eliminación de un activo del sistema:

Resultado: se seleccionó el activo a eliminar y se confirmó la acción. El sistema proceso la solicitud de eliminación correctamente y elimino el activo seleccionado de manera exitosa de la base de datos. Se verifico que el activo eliminado ya no estuviera presente en la interfaz de usuario ni en la base de datos y que los datos asociados al mismo fueran eliminados de manera adecuada de la base de datos se confirmó que el activo fue eliminado exitosamente del sistema.

5.3.3.4 Prueba de asignación y desasignacion de los activos a un empleado en específico:

Resultado: al seleccionar el activo que se quiere asignar a un empleado se abre una lista con la lista de los empleados ya registrados y al seleccionar el empleado al que se quiere asignar el sistema envía la solicitud y asigna el activo al empleado seleccionado, se verifico que el activo tiene un empleado asignado y

al darle al botón de desasignar el sistema envía la solicitud para desasignar el empleado del activo correctamente.

Con estos resultados exitosos, se confirma que el requerimiento RF003, relacionado con el registro de activos, ha sido completado con éxito, demostrando el correcto funcionamiento de las funcionalidades asociadas con dicho registro.

5.3.4 *Registro de Pagos (RF004)*

5.3.4.1 Prueba de registro de un nuevo pago a una entidad de servicios:

Resultado: Se completaron todos los campos requeridos para registrar un nuevo pago a una entidad de servicios, al enviar el formulario, el sistema proceso la información correctamente y se registró el pago en la base de datos, también se verifico que todos los datos ingresados fueran precisos y se reflejaran correctamente en la interfaz de usuario. Se confirmo que el pago a la entidad de servicios se registró exitosamente con los detalles especificados.

5.3.4.2 Prueba de registro de un nuevo pago por compra de software o hardware:

Resultado**:** Se completaron todos los campos requeridos para registrar un nuevo pago por compra de software o hardware, tras enviar el formulario, el sistema proceso la información correctamente y se registró el pago en la base de datos. Se verifico que todos los datos ingresados fueran precisos y se reflejaran correctamente en la interfaz del usuario. Se confirmo que el pago por compra de software o hardware se registró exitosamente con los detalles especificados.

5.3.4.3 Prueba de modificación de los detalles de un pago existente:

Resultado**:** Se accedió al formulario de edición de un pago existente y se realizaron modificaciones en sus detalles. Al enviar los cambios, el sistema proceso la solicitud correctamente y actualizo los detalles del pago en la base de datos. Se verifico que las modificaciones realizadas se reflejaran correctamente en la interfaz de usuario y que los datos actualizados fueran precisos y consistentes con los cambios realizados. Se confirmo que el pago fue modificado exitosamente con los nuevos detalles especificados.

5.3.4.4 Prueba de eliminación de un registro de pago del sistema:

Resultado**:** Se seleccionó el registro de pago a eliminar y se confirmó la acción. El sistema proceso la solicitud de eliminación correctamente y elimino el registro seleccionado de manera exitosa de la base de datos. Se verifico que el pago eliminado ya no estuviera presente en la interfaz de usuario que los datos asociados al mismo fueran eliminados de manera adecuada de la base de datos. Se confirmo que el registro de pago fue eliminado exitosamente del sistema.

Con estos resultados exitosos, se confirma que el requerimiento RF004, relacionado con el registro de pagos, ha sido completado con éxito, demostrando el correcto funcionamiento de las funcionalidades asociadas con dicho registro.

5.3.5 *Registro de ingresos (RF005)*

5.3.5.1 Prueba de registro de un nuevo ingreso por servicios

Resultado**:** Se completaron todos los campos requeridos para registrar un nuevo ingreso por servicios de marketing digital, al enviar el formulario, el sistema proceso la información correctamente y se registró el ingreso en la base

de datos. Se verifico que todos los datos ingresados fueran precisos y se reflejaran correctamente en la interfaz de usuario. Se confirmo que el ingreso por servicios de marketing se registró exitosamente con los detalles especificados.

5.3.5.2 Prueba de modificación de los detalles de un ingreso existente

Resultado**:** Se accedió al formulario de edición de un ingreso existente y se realizaron modificación en sus detalles. tras enviar los cambios, el sistema proceso la solicitud correctamente y actualizo los detalles del ingreso en la base de datos. Se verifico que las modificaciones realizadas se reflejaran correctamente en la interfaz de usuario y que los datos actualizados fueran precisos con los cambios realizados. Se confirmo que el ingreso fue modificado exitosamente con los nuevos detalles específicos.

5.3.5.3 Prueba de eliminación de un ingreso del sistema:

Resultado**:** se seleccionó el ingreso a eliminar y se confirmó la acción. El sistema proceso la solicitud de eliminación correctamente y elimino el ingreso seleccionado de manera exitosa de la base de datos. Se verifico que el ingreso eliminado ya no estuviera presente n la interfaz de usuario y que los datos asociados al mismo fueran eliminados de manera adecuada de la base de datos. Se confirmo que el ingreso fue eliminado exitosamente del sistema

5.3.5.4 Prueba de búsqueda y recuperación de ingresos por cliente

Resultado**:** se realizó una búsqueda de ingresos por cliente y el sistema recupero los resultados correspondientes de manera precisa y eficiente. Se verifico que los ingresos recuperados coincidieran con los criterios de búsqueda especifico y que se mostraran correctamente en la interfaz de usuario. Se

confirmo que la funcionalidad de búsqueda y recuperación de ingresos funciona correctamente y proporciona resultados precisos según los criterios de búsqueda específicos.

Con los resultados arrojados por las pruebas, se confirma que el requerimiento RF005, relacionado con el registro de ingresos, ha sido completado con éxito, demostrando el correcto funcionamiento de las funcionalidades asociadas con dicho registro.

5.3.6 *Pruebas de autenticación de usuarios (RF006)*

5.3.6.1 Prueba de presentación de usuario

Resultado**:** Al acceder a la plataforma, se presenta un formulario de inicio de sesión donde los usuarios pueden ingresar a sus credenciales para autenticarse en el sistema. El formulario es claro y accesible, permitiendo a los usuarios ingresar sus datos de manera intuitiva.

5.3.6.2 Prueba de inicio de sesión

Resultado: los usuarios pueden iniciar sesión correctamente utilizando sus credenciales validas, incluyendo su correo electrónico y contraseña. al ingresar la información requerida y presionar el botón de inicio de sesión, el sistema verifica las credenciales proporcionadas y autentica el usuario en la plataforma.

5.3.6.3 Pruebas de acceso denegado a funciones protegidas sin iniciar sesión:

Resultado**:** Se verifica que los usuarios que no han iniciado sesión no pueden acceder a las funciones protegidas de la plataforma. Al intentar acceder a áreas restringidas sin haberse autenticado previamente, el sistema redirige al

usuario al formulario de inicio de sesión, asegurando que solo los usuarios autenticados pueden acceder a estas funciones.

5.3.6.4 Prueba de manejo adecuado de errores de inicio de sesión:

Resultado: el sistema maneja correctamente los errores de inicio de sesión, como credenciales invalidas, se verifica que, al ingresar credenciales incorrectas, el sistema presente un mensaje de error claro indicando que las credenciales son invalidas.

Con estos resultados exitosos, se confirma que la funcionalidad de autenticación de usuarios cumple con los criterios establecidos y garantiza un acceso seguro y adecuado a la plataforma.

5.3.7 *Pruebas de permisos de usuarios RF007*

5.3.7.1 Pruebas de acceso completo para super Admins:

Resultado**:** después de iniciar sesión con credenciales de super admins, se verifico que el usuario tuviera acceso completo a todas las partes de la aplicación. Se comprobó que los super admins pudieran acceder y realizar acciones en todas las funciones y modulo del sistema, incluyendo registros de pagos e ingresos, planilla y módulo de reportes. No se encontraron restricciones de acceso para los super admins una vez que habían iniciado en la plataforma.

5.3.7.2 Pruebas de acceso limitado para admins

Resultado**:** tras iniciar sesión con credenciales de admin, se verifico que el usuario tuviera acceso limitado según su rol. Se comprobó que los admins pudieran acceder únicamente a las funciones permitidas por su rol, como registros

de pagos e ingresos, sin acceso a la parte de planilla ni al módulo de reportes. No se observó que los admins pudieran realizar acciones para las cuales no tenían permisos, como acceder a funciones restringidas o realizar cambios en áreas prohibidas.

Con estos resultados exitosos, se concluye que el sistema cumple con el requerimiento RF007 y que los permisos de usuarios se implementan correctamente, garantizando un acceso adecuado y seguro a las diferentes partes de la aplicación según los roles asignados.

5.3.8 *Pruebas de reportes RF009*

5.3.8.1 Pruebas de disponibilidad del módulo de reportes:

Resultado**:** Tras revisar la interfaz de usuario de super Admin, se confirmó que el módulo de reportes esta visible y accesible. Se encontró un botón que abre las opciones del módulo de reportes y se accedió correctamente desde el área designada de la plataforma.

Prueba de generación de reportes de planilla:

Resultado: después de seleccionar la opción para generar un reporte de planilla, el sistema presento un reporte detallado con la información solicitada. El reporte incluida de manera precisa los datos relacionados con los pagos de los empleados, así como también el total de empleados, el empleado que mayor gana y el que menos gana y el monto total de la planilla.

5.3.8.2 Prueba de generación de reporte de pagos de servicios:

Resultado**:** después de seleccionar la opción para generar un reporte de pagos de servicios, el sistema presento un reporte completo y detallado. El reporte incluyo información precisa sobre los pagos realizados a diferentes entidades de servicios, mostrando los detalles como entidad y monto, el total de entidades a las que se le paga y ala que más y menos se les pagan y el total de todos los pagos de los servicios por mes.

5.3.8.3 Prueba de generación de reportes de clientes e ingresos:

Resultado: tras solicitar la generación de un reporte que combine los datos de clientes e ingresos, el sistema presento un reporte completo que proporciono una visión global de la situación que generan los clientes. El reporte incluye información detallada sobre los clientes e ingresos generados por los servicios de marketing digital, permitiendo un análisis de la información. La presentación fue clara y precisa, lo que facilita la toma de decisiones por parte de los usuarios.

5.3.8.4 Pruebas de funcionalidades de análisis de los reportes:

Resultado**:** tras revisar la generación de los reportes se generan gráficos que facilitan la lectura e interpretación de la información.

6 Capítulo VI Implementación

Con la implementación terminamos el proyecto e implementamos la fase de entrega al cliente. Al ser una entrega de código, se entregará al cliente el código fuente del API y del FrontEnd el cual ellos revisan antes de implementarlos en sus respectivos servidores.

Fuente: Creación Propia

Ilustración 56: Portada de manual de usuario

6.1 Manual de Usuario

el manual de usuario es un documento importante para el cliente y la aplicación web, ya que nos informa cómo funciona la aplicación web. El cliente podrá consultar alguna duda que tenga en el manual que se le estará entregando al cliente. El cliente puede utilizar el documento del manual de usuario para futuras capacitaciones sobre la aplicación web, facilitando el fácil adaptamiento a la plataforma, consta de:

Portada: el manual de usuario consta de una portada para identificar el documento.

Índice: cuenta con un índice para el fácil movimiento en el manual.

Introducción: cuenta con una pequeña introducción del manual.

Cuenta con toda la información de los módulos respectivos de la aplicación, así como su paso a paso de cómo funcionan los módulos y sus respectivos procesos, con imágenes incluidas para facilitar el proceso de entendimiento sobre la aplicación web.

7 Capitulo VII: Conclusiones y Recomendaciones

A continuación, se presentan las conclusiones del proyecto de desarrollo de aplicación web para Acqua Digital, evaluando los objetivos planteados al inicio del proyecto y el aprendizaje a lo largo del desarrollo de la aplicación.

7.1 Conclusiones

Objetivo general: Implementar en la empresa Acqua Digital Marketing un gestor de datos que contribuya al mejoramiento de la toma de decisiones informada, optimizando las áreas de gestión de planilla, interacción con clientes, seguimiento de cobros y pagos, durante el cuarto trimestre del 2024.

- Se creo el gestor de datos para la recolección de información sobre la empresa y clientes, facilitando la toma de decisiones mediante la información recolectada por la aplicación y sus administradores.
- La información recogida por la aplicación promueve la toma de decisiones informada al dar reportes importantes sobre la empresa, evitando decisiones apresuradas que podrían poner en riesgo la competitividad ganada en el mercado de marketing digital.

Objetivo específico: recopilar los requerimientos del sistema atreves de reuniones con el representante de la empresa, analizando la información obtenida para garantizar su claridad y pertenencia.

- Se realizaron reuniones con el representante de la empresa Acqua Digital para obtener de los requerimientos del sistema y verificar que se cumplieran sus expectativas sobre la aplicación.

Objetivo específico: diseñar el sistema y estructurar la base de datos, asegurando que se manejen correctamente los requerimientos establecidos.

- Mediante el análisis de los requerimientos se realizó el diseño del sistema y la estructura de la base de datos, cumpliendo con los requerimientos del cliente.

Objetivo específico: desarrollar tanto el BackEnd y el FrontEnd del sistema, integrando lógicamente los componentes y cumpliendo con los requerimientos especificados.

- Se desarrollo un sistema integral entre el BackEnd y FrontEnd de manera lógica y fluida. El análisis de requerimientos permitió construir un sistema coherente y totalmente funcional, donde ambas partes actúan de manera natural.

Objetivo específico: realizar pruebas necesarias para evaluar el correcto funcionamiento de la aplicación identificando y corrigiendo posibles errores.

- La realización de pruebas funcionales garantizo que la aplicación funcionara de manera correcta en las situaciones previstas. La identificación y corrección temprana de errores mejoro la calidad del sistema, generando una interfaz de usuario y experiencia de usuario correcta.

Objetivo específico: elaborar un manual de usuario y preparar los documentos para la entrega final de proyecto.

- Se desarrollo un manual de usuario completo, asegurando una adaptación al sistema por parte de los usuarios el entendimiento rápido de la plataforma, entendiendo el completo funcionamiento de la plataforma.

Los objetivos fueron alcanzados con éxito. Cumpliendo las necesidades y expectativas del cliente, la empresa cuenta con una herramienta bastante solida la para la toma de decisiones informadas apoyando su crecimiento en el mercado del marketing digital.

7.2 Recomendaciones

A continuación, se presentan algunas recomendaciones para el uso de la plataforma y futuras implementaciones sobre ella:

- A la empresa se recomienda hacer una capacitación de introducción a la plataforma para facilitar el uso y el entendimiento de ella. A demás proveer una copia del manual de usuario para que los usuarios la tengan a mano y poder consultarla en el momento que deseen.
- Se recomienda la utilización de feedback con los usuarios para poder optimizar la plataforma mediante la información que pueda generar los usuarios sobre la aplicación web.
- Registrar datos sobre las decisiones tomadas, esto para poder identificar áreas de mejoras en la aplicación web y poder optimizarla en un futuro.
- Mantenerse actualizado con las medidas de seguridad para mantener sus datos a salvo y evitar la pérdida de información importante.
- La aplicación es flexible, se puede adaptar mediante la empresa vaya creciendo, la aplicación podrá crecer y ser una aplicación más robusta.

Al seguir estas recomendaciones la empresa podrá sacarle el máximo provecho a la aplicación web y mantener una toma de decisiones informada manteniendo la competitividad en el mercado a largo plazo. También, facilitara la evolución de la plataforma a medida que las necesidades de la empresa crezcan.

8 Bibliografía

Somerville, I. (2016). Ingenieria de software (10th ed.). Pearson.

Beizer, B. (1990). Software Testing Techniques (2nd ed.). Van Nostrand Reinhold.

Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education.

Marzal, A., Gracia, I., & García, P. (2014). Introducción a la programación con Python 3.

Martínez Ladrón de Guevara, J. (2020). Fundamentos de programación en Java.

Wiegers, K. E., & Beatty, J. (2013). Software Requirements (3rd ed.). Microsoft Press.

Schwaber, K., & Sutherland, J. (2020). La guía definitiva de Scrum: Las reglas del juego.

Hostinger. (s.f.). What is MySQL? Recuperado el June 3, 2024, de https://www.hostinger.com/tutorials/what-is-mysql

Rootstack. (s.f.). ¿Qué metodología es mejor para el desarrollo web? Recuperado el 3 de junio de 2024, de https://rootstack.com/es/blog/que-metodologia-es-mejor-para-el-desarrollo-web

IBM. (s.f.). Bases de datos relacionales. Recuperado el 3 de junio de 2024, de https://www.ibm.com/mx-es/topics/relational-databases

Visure Solutions. (s.f.). Definición de requisitos. Recuperado el 3 de junio de 2024, de https://visuresolutions.com/es/blog/definicion-de-requisitos/

Epitech. (s.f.). Flask: Guía completa para Python. Recuperado de https://www.epitech-it.es/flask-python/

HubSpot. (s.f.). Qué es CSS. HubSpot Blog. Recuperado de https://blog.hubspot.es/website/que-es-css

Deloitte. (s.f.). ¿Qué es ORM? Recuperado de https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html

OpenWebinars. (s.f.). ¿Qué es un ORM? Recuperado de https://openwebinars.net/blog/que-es-un-orm/

J2logo. (s.f.). SQLAlchemy: Tutorial de Python SQLAlchemy, guía de inicio. Recuperado de https://j2logo.com/python/sqlalchemy-tutorial-de-python-sqlalchemy-guia-de-inicio/

Lucidchart. (s.f.). Tutorial de diagrama de clases UML. Recuperado de https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml

Diagramas UML. (s.f.). Diagramas de secuencia UML. Recuperado de https://diagramasuml.com/secuencia/

Trigas, M. (2012). Diseño e implementación de un sistema de bases de datos para la gestión de un colegio. Universitat Oberta de Catalunya. Recuperado de https://openaccess.uoc.edu/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf

California State University, Long Beach. (s.f.). Entity Relationship Modeling Examples. Recuperado de https://web.csulb.edu/colleges/coe/cecs/dbdesign/dbdesign.php?page=models.html

UNIR. (s.f.). Ingeniería y tecnología: Framework. Recuperado de https://unirfp.unir.net/revista/ingenieria-y-tecnologia/framework/

EBAC. (s.f.). Frameworks. Recuperado de https://ebac.mx/blog/frameworks

Arimetrics. (s.f.). Glosario Digital: Frontend. Recuperado de https://www.arimetrics.com/glosario-digital/frontend

Olawanle, J. (s.f.). Artículos de Joel Olawanle. Recuperado de https://www.freecodecamp.org/news/author/joel-olawanle/

Hostinger. (s.f.). ¿Qué es React? Recuperado de https://www.hostinger.mx/tutoriales/que-es-react