UNIVERSIDAD INTERNACIONAL DE LAS AMÉRICAS

Escuela de Ingeniería Informática

TRABAJO FINAL DE GRADUACIÓN PARA OPTAR POR EL GRADO DE BACHILLERATO EN INGENIERÍA DE SOFTWARE.

Sistema web para la gestión financiero contable para Marsh Asprose

Douglas Justin González Castro

Tutor(a):
Fernando Ríos Vargas

San José

Agosto, 2026

DEDICATORIA

Este documento está dedicado en primer lugar a Dios, por brindarme la sabiduría, la perseverancia y la fortaleza necesarias para culminar satisfactoriamente esta importante etapa académica, a mis padres, quienes han sido un pilar fundamental en mi formación personal, inculcándome valores, disciplina y motivación constante para iniciar y continuar este camino académico, el cual hoy logro culminar gracias a su apoyo incondicional, esfuerzo y confianza en mis capacidades.

Asimismo, a mis docentes, quienes a lo largo de la carrera han contribuido significativamente a mi crecimiento profesional, compartiendo sus conocimientos, experiencia y orientación, elementos que han sido clave para el desarrollo de las competencias necesarias que hicieron posible la realización de este trabajo final de graduación.

AGRADECIMIENTOS

Expreso mi profundo agradecimiento a mis padres, quienes han sido un apoyo fundamental a lo largo de este proceso, brindándome motivación, comprensión y confianza en cada una de las decisiones tomadas para alcanzar este objetivo. Su esfuerzo y acompañamiento han sido clave para lograr este importante paso en mi desarrollo personal y profesional. Asimismo, agradezco a mis docentes y a la directora de carrera, quienes han sido de gran apoyo durante mi proceso de aprendizaje en la universidad, aportando orientación, conocimientos y acompañamiento académico que contribuyeron significativamente a mi formación profesional.

De igual manera, extiendo mi agradecimiento a las personas que colaboraron durante el desarrollo de este proyecto dentro de la compañía, por su disposición, guía y apoyo en la obtención de información relevante para la elaboración del trabajo. En especial, agradezco a don Marlon Zamora, por su colaboración y disposición durante el proceso, así como a mi tutor, don Fernando Ríos, por su orientación, seguimiento y valiosos aportes, los cuales fueron fundamentales para la culminación satisfactoria de este Trabajo Final de Graduación.

CONTENIDO

TABLAS

FIGURAS

ANEXOS

RESUMEN EJECUTIVO

El presente documento tiene como propósito describir de manera estructurada el proceso de investigación y desarrollo de una solución tecnológica orientada a atender una problemática identificada en la compañía Marsh Asprose S.A., ubicada en San José. En el capítulo uno y dos se presentará el contexto general del proyecto, abordando la descripción detallada de la problemática existente, su justificación y la importancia de su resolución dentro del entorno organizacional. Asimismo, se analizará la viabilidad del proyecto desde las perspectivas técnica, operativa y económica, considerando además el marco normativo y las leyes aplicables que deben cumplirse, así como también se contempla la definición de herramientas, arquitectura y definición de los requisitos que tendrá la aplicación.

Posteriormente, el capítulo tres se centrará en el enfoque metodológico que guiará el proceso de investigación, definiendo el tipo de estudio que se realizará, las variables involucradas y los instrumentos que serán utilizados para la recolección de información relevante. Este análisis permitirá obtener datos confiables que sustenten la toma de decisiones en las etapas posteriores del proyecto.

Continuando, el capítulo cuatro estará orientado al desarrollo técnico de la propuesta de solución de software, contemplando las etapas de análisis, diseño, construcción y pruebas del prototipo programado. En esta sección se definirá el modelo funcional del sistema mediante la elaboración de casos de uso y diagramas UML que permitirán representar de forma gráfica la interacción entre los diferentes componentes del sistema y los usuarios. El proceso continuará con la fase diagramación de la base de datos y la codificación del prototipo, en la cual se implementarán las funcionalidades definidas previamente, finalizando con la ejecución de pruebas que permitan validar el correcto funcionamiento del sistema y verificar que cumple con los requerimientos establecidos en el presente documento.

CAPÍTULO I: INTRODUCCIÓN

Descripción del Problema

La compañía corredora de seguros Marsh Asprose es una empresa consolidada en el país desde 1984, Marsh Asprose ofrece servicios de corretaje de seguros y consultoría de riesgos a empresas nacionales e internacionales en todo Costa Rica, con especial foco en programas de Salud y Beneficios para Empleados, en la institución actualmente no se cuenta con una herramienta actualizada que permita el registro, control, seguimiento y monitoreo de procesos importantes de su gestión contable.

Falta de Visibilidad Sobre la Ejecución del Presupuesto Organizacional

Actualmente en la compañía se registran los gastos y compras sin una conexión directa con los límites presupuestarios establecidos y manejado en una hoja de cálculo independiente lo que dificulta conocer sin demora cuanto se ha ejecutado, cuanto está comprometido y cuánto queda disponible en cada partida presupuestaria.

Dificultad con Control de los Registros de Ingresos

Actualmente, la organización enfrenta dificultad en la administración de los ingresos que provienen de procesos de facturación o de las cuentas por cobrar. Esta situación genera dispersión en la información financiera, ya que los ingresos se llevan en sistemas aparte sin una integración adecuada.

Falta de Automatización y Sincronización Entre Información Comercial y Financiera

En este tipo de entornos, la facturación suele operar como un sistema aislado, generando documentos y registros de venta que luego deben ser ingresados manualmente en la contabilidad, lo que incrementa el riesgo de errores de registro, duplicidad de datos, retrasos en los cierres contables y dificultades en la conciliación de ingresos.

Falta de Coherencia en los Registros Contables de las Operaciones Financieras

Los movimientos generados, a cómo se manejan actualmente por la compañía, se registran de forma manual llevando registros en hojas de cálculo, lo cual causa errores de contabilización, duplicidad de asientos y demoras en el cierre contable.

Falta de Trazabilidad Sobre los Gastos Operativos

Al no contar con un sistema que valide automáticamente los comprobantes y consolide la información en tiempo real, se dificulta el seguimiento del presupuesto y la detección de desviaciones financieras, lo que puede afectar la toma de decisiones y el control del flujo de efectivo.

Ineficiencia en la Gestión de las Obligaciones Financieras con Proveedores

En la compañía actualmente no se registran y/o validan las facturas recibidas de proveedores de forma uniforme, lo que genera pérdidas de comprobantes y duplicidad de registros, lo que afecta la relación con los proveedores como la liquidez de la empresa.

Además, la ausencia de un control fuera de una hoja de cálculo complica seguimiento de los compromisos pendientes, la aplicación correcta de retención e impuestos, así como la conciliación de pagos efectuados con los movimientos contables.

Cuentas por Cobrar

Deficiencias en la trazabilidad de siniestros y reclamos, una vez adquirida la póliza, debido a la funcionalidad del sistema actualmente.

Objetivos

Objetivo General

Desarrollar un sistema web para la gestión financiero contable utilizando las herramientas Visual Studio (C#), SQL Server Management Studio (SQL Server) y Visual Studio Code (React.JS)

Objetivos Específicos

- Analizar los requerimientos funcionales y no funcionales del sistema web.
- Construir el diseño de la base de datos y del sistema web mediante la herramienta Workbench
- Programar los diferentes módulos del sistema mediante la herramienta Visual Studio y Visual Studio Code.
- Probar los diferentes módulos de forma independiente y también de forma integral, garantizando los resultados correctos del sistema.

Justificación

Viabilidad Técnica

En el desarrollo del prototipo se cuenta con una viabilidad técnica, ya que, con respecto al hardware la institución ya cuenta con todo equipo necesario para alojar la aplicación, la institución cuenta con seis servidores virtualizados enfocados en funcionar como distintos tipos de ambientes, por ejemplo cuenta con un servidor de desarrollo, un servidor de QA, dos servidores de pre-producción los cuales se encuentran unidos por un balanceador de carga, así como dos servidores productivos los cuales también cuentan con un balanceador de carga, pero, ¿qué es un balanceador de carga?, según F5 (2026):

Un balanceador de carga es una solución que actúa como un proxy de tráfico y distribuye el tráfico de red o aplicación entre puntos finales en varios servidores. Los equilibradores de carga se utilizan para distribuir la capacidad durante las horas pico de tráfico y para aumentar la confiabilidad de las aplicaciones.

Se puede determinar que un balanceador de carga es una herramienta que nos permite administrar el tráfico entrante a un sitio web y distribuirlo entre un nodo a y un nodo b para evitar una sobrecarga en uno o N cantidad de nodos y que el funcionamiento del sitio web sea óptimo y el usuario no experimente un rendimiento bajo del sistema y evitando un bloqueo por exceso de solicitudes. cada uno de estos servidores nodos y balanceadores mencionados anteriormente cuentan con una instancia de Windows server 2022 con una arquitectura de 64 bits, procesador Intel Xeon 2.60 GHz, 16 GB de RAM, un almacenamiento distribuido en tres discos los cuales en total cuentan con un total de 300 GB.

Con respecto a software se estaría trabajando con las licencias que ya poseen como lo son, el gestor de base de datos SQL Server 2019 y el IDE de Visual Studio Profesional 2019 versión 1.16.11.46, esto para el desarrollo del back-end haciendo uso del lenguaje de programación C# en su versión 4.8, utilizando la plantilla de Visual Studio ASP.NET APIs que nos facilita la herramienta, además del uso del IDE Visual Studio Code en su versión 1.107.1 el cual cuenta con una versión de uso comercial gratuita, para el desarrollo de la interfaz se utilizará React JS en su versión 18.2.0, el cual requiere previamente a la creación del proyecto en React JS que en el ambiente de desarrollo se encuentre instalado Node JS en su versión 20.16.0 y NPM en su versión 10.8.0.

Viabilidad Operativa

La compañía Marsh Asprose cuenta con la disposición operativa para hacer uso del prototipo funcional como una herramienta de mejora continua ya que permitirá al área de contaduría identificar oportunidades de automatización, reducción de errores manuales, y mejora en la trazabilidad financiera, la implementación de este software no interfiere con la operatividad actual de la empresa, ya que, poseen una estructura organizacional que permite la adopción de este tipo de herramientas tecnológicas con el fin de optimizar y automatizar labores diarias, en este sentido, el prototipo funcionará como una plataforma de prueba que permitirá recopilar retroalimentación directa de los usuarios, ajustar funcionalidades y asegurar que la solución final se adapte a las necesidades reales de la organización, consolidando así su viabilidad operativa y su potencial impacto positivo en la gestión financiera.

Viabilidad Económica

El prototipo será desarrollado con herramientas las cuales la empresa ya posee, por lo cual se debe contemplar el gasto mensual/anual, además del uso de hardware/software. A pesar de que existe un cobro por la hora de un programador de ¢15.983,96 por día, establecido por el Ministerio de Trabajo, este costo no se tomará en cuenta, ya que al ser un trabajo de graduación no será necesario el pago de estas horas

Tabla 1

Costos de licenciamiento.

Herramienta/Infraestructura	**Precio**
Licencia SQL Server	$ 5434/año
Licencia Visual Studio Professional	$ 1614
Licencia Windows Server Datacenter	$6,771
Mantenimiento en Nube (respaldo)	$660/mes

Fuente: Elaboración propia, 2026

Nota: Estos costos se contemplan a nivel informativo, sin embargo, muchos de estos ya han sido pagados por la compañía con anterioridad.

Se tiene en cuenta que la compañía ya dispone de ventajas significativas en relación con la adquisición y uso de licencias de software y hardware virtualizado, debido a su condición de

empresa consolidada y a la existencia de necesidades previas de herramientas tecnológicas para la gestión del negocio. Esto implica que gran parte de la infraestructura, plataformas y licencias requeridas ya se encuentran implementadas y activas. Asimismo, la experiencia con la administración de estos recursos permite una integración más eficiente de nuevas soluciones disminuyendo los tiempos de implementación y puesta en marcha.

Tabla 2

Costos de desarrollo.

Fase	**Costo por día**	**Cantidad de días**	**Costo Total**
Análisis	¢15.983,96	5	¢79.919,8
Diseño	¢15.983,96	5	¢79.919,8
Desarrollo	¢15.983,96	45	¢719.278,2
Pruebas	¢15.983,96	8	¢127.871,68
Total	¢15.983,96	63	¢1.006.989,48

Fuente: Elaboración propia, 2026

Los costos de desarrollo se calculan, ya que es necesario dar a conocer, sin embargo, estos no se tomarán en cuenta, puesto que al ser un proyecto de graduación se brindará como un servicio gratuito a la institución, por lo que, la compañía no deberá asumir dichos costos.

Viabilidad Legal

El proyecto se respaldará con distintas leyes que aplican actualmente en el país, como lo son las leyes de Delitos Informáticos y Conexos, del Título VII del Código Penal N.º 9048, en la que se asegura que según el artículo 196 bis. De dicha ley, toda la información personal o sensible será guardada de manera segura y no será utilizada para ningún fin diferente al original de la aplicación, garantizando la privacidad de la información y sus colaboradores.

Ley 8148 Adición de los artículos 196 BIS, 217 BIS y 229 BIS al Código Penal, se cumple con los BIS 196 la cual sanciona la violación de comunicaciones electrónicas, o sea, a quien intercepte o acceda indebidamente a correos electrónicos o comunicaciones telemáticas ajenas. Las políticas

de la empresa incluyen altos estándares de seguridad para evitar ataques como hombre en el medio y la empresa cuenta con capacitaciones constantes de ciberseguridad para evitar ataques de ingeniería social, esto también cumple con la BIS 217 que penaliza la manipulación de sistemas de información para obtener beneficios y la BIS 229 que penaliza la destrucción o inhabilitación de datos en sistemas informáticos o redes, dicha ley se podría decir que va de la mano con la ley N° 4573 para reprimir y sancionar los delitos informáticos de la Asamblea Legislativa de la República de Costa Rica del año 2001, la cual fue modificada en 2012 el cual sanciona delitos informáticos como lo son la violación de datos personales, sabotaje informático, acceso ilícito entre otros.

Ley de Derechos de Autor 6683 por parte de la Asamblea Legislativa de la República de Costa Rica del año 1982, la cual protege propiedades intelectuales originales como lo será el software el software que será desarrollado en el presente proyecto, aun en su condición de prototipo funcional, constituye una obra original al ser resultado de un proceso creativo propio que abarca el diseño de la arquitectura del sistema, la lógica de programación, los algoritmos implementados, la estructura de bases de datos y la documentación técnica asociada.

Ley 8968 sobre la protección de la persona frente al tratamiento de sus datos personales, la cual garantiza el derecho de los ciudadanos costarricenses el derecho fundamental a la autodeterminación informativa, asegurando la confidencialidad, integridad y uso legítimo de sus datos personales. Dicha normativa regula de manera estricta los procesos de recolección, almacenamiento, tratamiento, acceso y transferencia de información personal, imponiendo sanciones económicas que pueden superar los trece millones de colones en caso de incumplimiento. Por ende, el prototipo contempla las buenas prácticas de programación y almacenamiento de datos, así como la posibilidad de auditar el tratamiento de datos garantizando que la información personal sea tratada conforme a la legislación vigente.

Proyecciones

Con el desarrollo del prototipo se espera atender y dar respuesta a diversas necesidades que actualmente afectan la eficiencia operativa de la compañía, particularmente en los procesos gestionados por el departamento de contabilidad y finanzas, a través del prototipo se busca establecer una plataforma que facilite la gestión de estos procesos, mejorando la organización, el acceso a la información y la precisión en el manejo de los datos, contribuyendo así a una toma de decisiones más ágil y fundamentada dentro de la organización.

Alcance Funcional

El proyecto estará compuesto de distintos módulos los cuales serán fundamentales para el correcto funcionamiento del total de la aplicación, esto con el fin de brindar una solución a la problemática actual de la institución:

Cuentas por cobrar

Este módulo será responsable de la gestión de los créditos otorgados a los clientes, garantizando un control eficiente sobre los montos, pendientes de cobro, los plazos de pago y el estado de cuenta de cada cuenta. A medida que se realicen los pagos, el sistema aplicará las cancelaciones parciales o totales correspondientes, calculará intereses moratorios en caso de atraso y actualizará el saldo disponible.

Cuentas por pagar

El módulo de cuentas por pagar se encargará de la administración de las funciones financieras que la empresa mantiene con los proveedores, asegurando un control eficiente de los compromisos adquiridos, los plazos de pago y la correcta aplicación contable y tributaria.

Una vez realizados los pagos, el módulo procederá con la conciliación automática, entre las transacciones bancarias, y facturas registradas, empleando un algoritmo de coincidencia por monto, fecha y número de factura, a su vez se calcula las retenciones e impuestos aplicables.

Con esta información cerrada se procede con la finalización en el cierre contable y financiero, el cual clasifica las cuentas según su estado, retención de impuestos, y genera los asientos de gasto.

Gastos

Este módulo se encargará de la gestión de todos los gastos directos e inmediatos del negocio, aquellos que no están asociados a cuentas por pagar ni se realizan a crédito, sino que se ejecutan de forma directa. El módulo validará los comprobantes fiscales correspondientes, calculará los impuestos asociados y generará de forma automática los asientos contables.

Cuentas Contables

Este módulo se encargará de gestionar el catálogo contable, y coordinar la integración automática de todas las transacciones económicas generadas desde los demás módulos. Su función principal es garantizar que cada movimiento se registre correctamente en las cuentas

correspondientes, manteniendo la exactitud de la información financiera, permitiendo definir, estructurar y mantener actualizado el catálogo de cuentas, tipos de cuenta, y reglas de asociación con los distintos procesos del negocio. De manera automatizada, el sistema generará los asientos contables derivados de cada operación, evitando la intervención manual y reduciendo errores en la contabilización. Además, el módulo facilitará la consolidación de información contable, la generación de balances, estados financieros, permitiendo un control estricto de la situación actual de la empresa.

Facturación

Este módulo permitía generar facturas electrónicas asociadas a pólizas, renovaciones, comisiones o servicios adicionales, asegurando el cumplimiento de las normativas fiscales establecidas por el Ministerio de Hacienda (v4.4). Además de automatizar la emisión de recibos de prima, notas de crédito o débito en caso de que aplique. El proceso incluirá la validación de los datos del cliente, la verificación de los montos a facturar y la aplicación de los impuestos correspondientes. Asimismo, permitirá consultar el historial de facturación por cliente o póliza, emitir copias electrónicas, controlar los estados de las facturas (pendientes, pagadas, anuladas)

Gestión presupuestaria

Este módulo facilitará la definición de un presupuesto anual o periódico, dependiendo de la necesidad de cada departamento del negocio, la asignación de partidas por tipo de gasto, o centro de costo y el control de la ejecución presupuestaria mediante la comparación constante entre los montos planificados contra los gastos reales. Este módulo se integrará con el módulo de gastos y cuentas por pagar para garantizar la trazabilidad de cada transacción y su impacto en el presupuesto general.

Ingresos

Este módulo se encargará de la administración de los ingresos de la organización, tomando en cuenta tanto los provenientes de los procesos de facturación como los procesos de ingresos extraordinarios o no contemplados dentro de las cuentas por cobrar.

Además, el módulo deberá contar con un algoritmo que permita la conciliación automática, que facilite la identificación de tendencias, control de flujos de efectivo y análisis.

Mantenimiento

Este módulo será responsable de la gestión de los catálogos de información del sistema, permitiendo el manejo de registros de manera controlada y segura.

A través de este módulo se administrarán catálogos clave relacionados con la ubicación de los activos, los estados operativos, los estados de mantenimiento, los estados de rendimiento y los estados de gestión de garantías, entre otra información, asegurando la consistencia, integridad y trazabilidad de la información.

Consultas

El sistema será capaz de obtener la información ingresada al mismo en los módulos mencionados anteriormente aplicando, sin la necesidad de generar un reporte, esto con el fin de mantener un control día a día de la información en sistema.

Reportes

Este módulo se hará cargo de generar información proporcionada de las diferentes tablas y cálculos necesarios para los reportes, los cuales se generarán en formato XLSX o PDF con distintos parámetros aplicados en forma de filtros los cuales se alimentarán con base a la información ingresada en los módulos anteriormente mencionados.

Seguridad

La principal funcionalidad de este módulo será el de la administración y suministración de los usuarios del sistema, tanto de usuarios, administradores, auditores o usuarios de procesos, los cuales ingresarán la información al sistema.

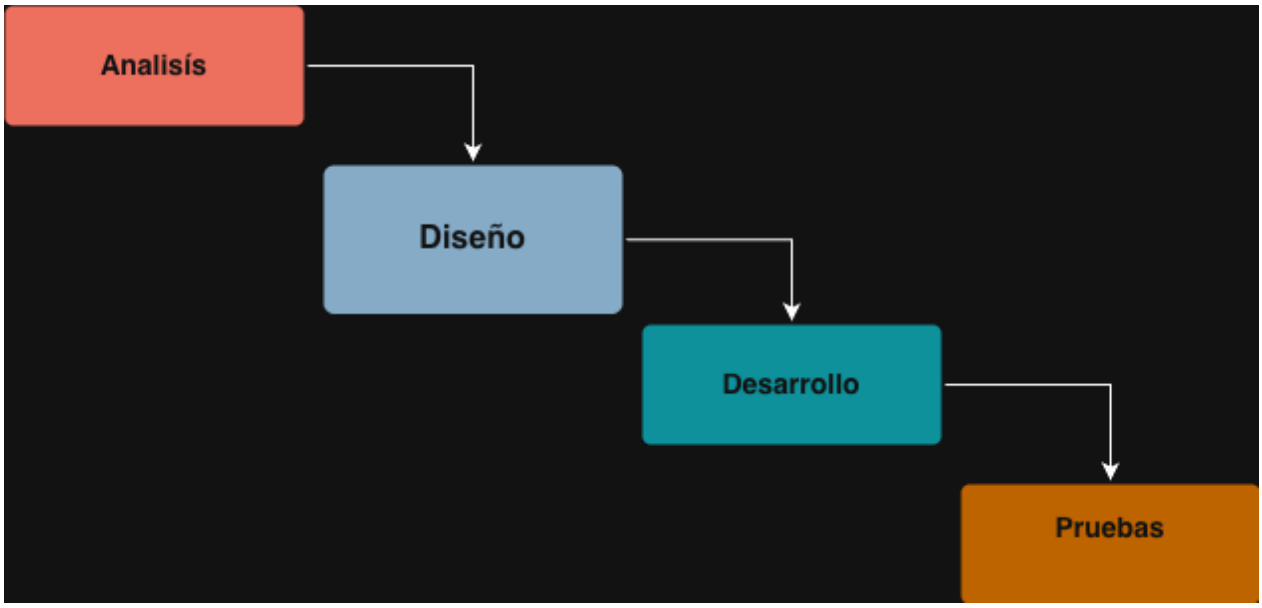
Esto deberá ser permitido no por un ROL de usuario, el cual estará previamente definido, sino por la funcionalidad definida a nivel de negocio para el usuario.

Alcance Metodológico

Para el desarrollo del presente proyecto se optó por la adopción de la metodología en cascada, debido a que la estructura y planificación del trabajo se alinean con un enfoque secuencial y lineal. Esta metodología permite organizar el proceso en cuatro fases claramente delimitadas, donde cada etapa debe completarse y validarse antes de dar inicio a la siguiente.

Ilustración 1

Metodología de desarrollo.



Fuente: Elaboración Propia, 2026.

Además, al emplear la metodología en cascada, se establece desde el inicio una planificación temporal claramente definida para cada fase del proyecto, lo que permite a la compañía conocer con precisión en qué etapa del ciclo de vida se encuentra el desarrollo en todo momento. Esta estructuración secuencial facilita la trazabilidad, ya que cada fase cuenta con entregables específicos, hitos de control y fechas previamente determinadas, lo que posibilita un seguimiento sistemático del avance.

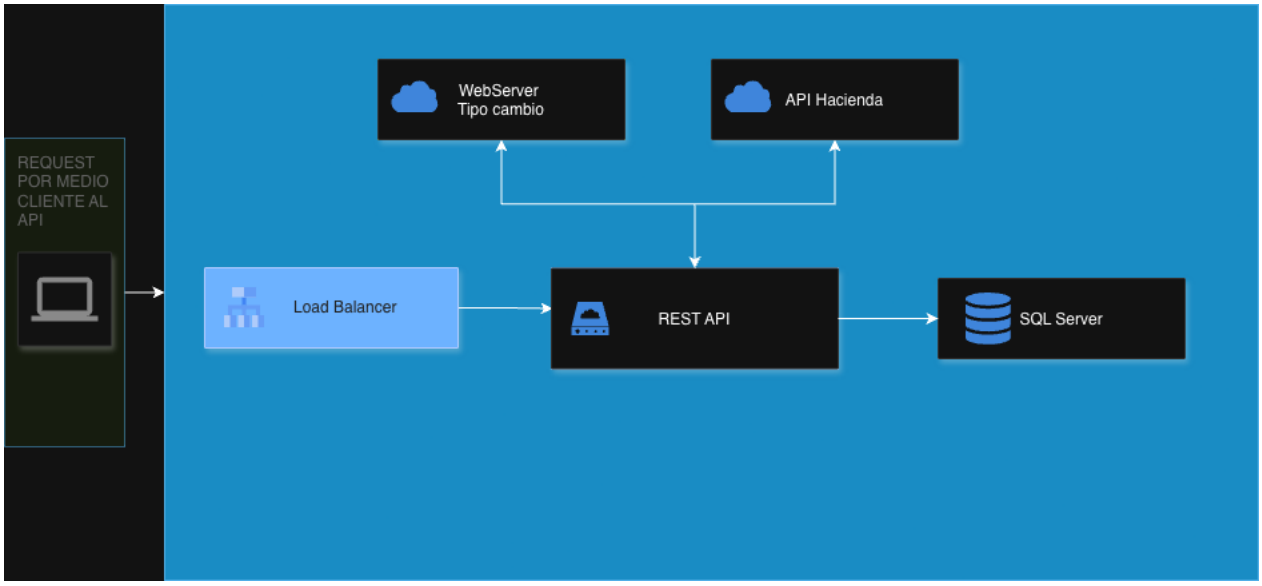
Alcance Tecnológico

El prototipo será desarrollado con un enfoque centrado en la funcionalidad dentro de un entorno web, implementando una arquitectura basada en el estilo API-REST que permita la separación clara entre la capa de presentación y la lógica de negocio. La interfaz de usuario será construida con React JS, lo que posibilita la creación de componentes reutilizables, una gestión eficiente del estado y una experiencia interactiva dinámica para el usuario final. Por su parte, el backend estará conformado por una REST-API desarrollada en C#, encargada de gestionar las reglas de negocio, validaciones, autenticación y exposición de endpoints para el consumo de datos mediante protocolos HTTP. Esta API se conectará a una base de datos SQL Server, donde se

almacenará de manera estructurada la información financiera y contable del sistema, garantizando integridad referencial, consistencia transaccional

Ilustración 2

Arquitectura propuesta.



Fuente: Elaboración propia, 2026.

Dicha aplicación será accesible por cualquier tipo de dispositivo que permita el acceso a la red, por lo cual el diseño deberá poseer la cualidad de ser “Responsive”.

A continuación, se describirá las tecnologías utilizadas para el desarrollo:

React JS.

React es una librería de JavaScript de código abierto desarrollada originalmente por Facebook, diseñada para facilitar la creación de interfaces de usuario modernas, dinámicas y eficientes. React implementa un DOM virtual**,** Astudillo (2021) define el virtual DOM como "una representación del DOM guardada en memoria, que actúa de intermediario entre los estados de la aplicación y los estados del DOM (vistos por el usuario)." (párr.4). Gracias a esta característica, React se ha convertido en una herramienta ampliamente utilizada para el desarrollo de aplicaciones web escalables y con alto rendimiento.

VS Code.

Herramienta de entorno de desarrollo integrado (IDE) utilizada para la creación de aplicaciones web y otros tipos de software. Desarrollado por Microsoft, este editor de código se caracteriza por ser ligero, multiplataforma y contar con una licencia gratuita para uso comercial, permite funcionalidades que facilitan el trabajo de los desarrolladores, como resaltado de sintaxis, autocompletado inteligente, integración con sistemas de control de versiones y una amplia biblioteca de extensiones que permiten adaptar el entorno a diferentes lenguajes y tecnologías de desarrollo.

C#

Microsoft (2026) define C# como "un entorno de desarrollo gratuito, multiplataforma y de código abierto. Los programas de C# se pueden ejecutar en muchos dispositivos diferentes, desde dispositivos de Internet de las cosas (IoT) a la nube y entre todas partes." (párr. 2). Se entiende como un lenguaje de programación orientado a objetos con seguridad de tipos generado por Microsoft, como parte de su plataforma .NET, ampliamente utilizado para la creación de aplicaciones empresariales y servicios web.

Visual Studio

Es un entorno de desarrollo integrado producido por Microsoft, diseñado para escribir, depurar, compilar y publicar aplicaciones para distintas plataformas de .NET, con diferencia de Visual Studio Code, que es un editor de código ligero y altamente extensible orientado a múltiples lenguajes mediante extensiones, Visual Studio funciona como un IDE más robusto y especializado,

especialmente optimizado para el desarrollo de aplicaciones empresariales y de gran escala dentro del ecosistema .NET.

SQL Server

Sistema de gestión de base de datos relacionales creado por Microsoft diseñado para administrar y almacenar información de forma eficiente, basado en esto la principal función que requeriremos de este software s la del motor de base de datos que Microsoft (2025) lo define como "el servicio principal para almacenar, procesar y proteger datos. El motor de base de datos proporciona acceso controlado y procesamiento de transacciones para cumplir los requisitos de las aplicaciones consumidoras de datos más exigentes" (párr.7).

CAPÍTULO II: MARCO REFERENCIAL

Introducción

Dado que el proyecto será desarrollado con una arquitectura la cual posee múltiples herramientas en los cuales interactúan componentes como lo son motores de bases de datos, lenguajes de programación, y entorno de desarrollo integrado (IDE), se desarrollará a continuación una serie de puntos, con el fin de que el lector tenga una interpretación más clara de las tecnologías utilizadas para el desarrollo del prototipo.

El prototipo por desarrollar será una aplicación con una arquitectura tipo API-REST, este tipo de arquitecturas cuentan con una interfaz gráfica, un API-REST a la cual se accederá por medio de protocolo HTTP y un servidor de base de datos la cual se encuentra separada de estas dos últimas aplicaciones.

En conjunto estas aplicaciones será un sistema web, permitiendo una alta disponibilidad a los usuarios que utilicen la aplicación y un acceso centralizado sin necesidad de una instalación individual a cada computador u otro dispositivo, según Indeed (2026) este tipo de aplicaciones trabajan de la siguiente forma:

> Cada aplicación web consta de tres elementos: un servidor web para gestionar las solicitudes de los clientes, un servidor de aplicaciones para ejecutar las tareas solicitadas y una base de datos para almacenar la información. Una aplicación web funciona mediante una combinación de scripts del lado del servidor y del lado del cliente. (p.1)

Esto indica que a pesar de ser una sola aplicación esta estará dividida en varios componentes, los cuales operarán en distintos servidores todo con el fin de dar un servicio, aplicando los parámetros de seguridad para cada una de estas capas.

Herramientas de Desarrollo Front-End

Esta interfaz será desarrollada con el IDE Visual Studio Code, creado por Microsoft en 2015 esta herramienta vino a competir con IDE enfocados al desarrollo de proyectos web como lo es Sublime o Atom, con el potencial de permitir la personalización del mismo y permitir que los usuarios desarrollen plugin para mejorar su experiencia a la hora de programar distintos tipos de proyectos.

Teniendo esto en cuenta se puede comenzar con la creación del proyecto, previamente a iniciar el repositorio en React JS, se debe cumplir con ciertos requisitos, como por ejemplo haber instalado Node JS y NPM en el equipo, estas herramientas serán indispensables para la ejecución e

instalación de librerías, la funcionalidad de Node JS, de acuerdo con la página oficial de Node JS (2026), “Como entorno de ejecución de JavaScript asíncrono basado en eventos, Node.js está diseñado para crear aplicaciones de red escalables.” (parr.1)

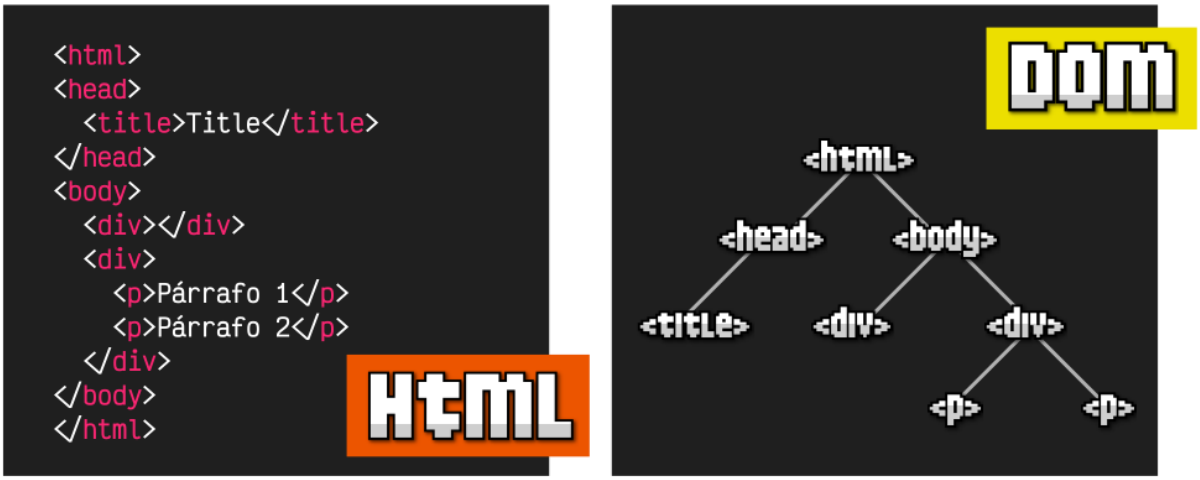
Esto quiere decir que Node JS es una plataforma que nos facilita la ejecución de programas, proporcionando una colección de librerías, para ejecutar código fuera de su contexto original, por ejemplo Node JS nos permite ejecutar código JavaScript en el lado del servidor o crear una especie de servidor virtualizado en el cual se ejecuta el código de la aplicación, al instalar Node JS este nos instala automáticamente administrador de paquetes o como su nombre lo indica Node Package Manager (NPM), el cual consiste en una herramienta por CLI para la instalación de librerías para el proyecto, así como la ejecución y compilación.

Habiendo explicado la aplicación en la cual se trabajará, se continuará con el lenguaje de programación, el lenguaje será JavaScript, utilizando el framework de React JS, desarrollado como un proyecto open-source por Facebook en 2013 con la finalidad de optimizar el manejo de componentes mediante el uso de un DOM (Document Object Model) virtual, en lugar de usar el DOM real del sitio, este se puede definir, según Mozilla (2026) “Representa un documento con un árbol lógico. Cada rama del árbol termina en un nodo y cada nodo contiene objetos. Los métodos DOM permiten el acceso programático al árbol. Con ellos, puede cambiar la estructura, el estilo o el contenido del documento.” (p.1)

Esto quiere decir que el HTML del sitio web se representa como un árbol al cual el API de JavaScript una vez el navegador haya convertido este código HTML en objetos, es capaz de manipularlo, permitiendo agregar, editar y eliminar contenido y estructuras dentro de estos objetos, permitiendo tener páginas web dinámicas e interactivas.

Ilustración 3

Representación del DOM.



Fuente: DOM

Tomando en consideración lo anterior, se puede afirmar que React JS permite la gestión de un DOM Virtual, el cual consiste en una representación del DOM real almacenada en memoria. Esta representación es significativamente más ligera y eficiente que el DOM real del navegador, lo que posibilita realizar comparaciones rápidas entre el estado previo y el estado actualizado de la interfaz.

Ilustración 4

DOM Virtual.

Fuente: DOM Virtual

En este tipo de tecnologías es fundamental que, al ser un tipo de portal web, posean disponibilidad para distintos tipos de dispositivos, por ende se debe de implementar un diseño responsive o adaptativo el cual permita a los usuarios un control mayor de cómo se comporta la visualización de la aplicación en diferentes dispositivos, esto es importante ya que permitirá el acceso a la aplicación en diferentes instancias de la compañía desde Colombia, Panamá y Costa Rica, y en distintos tipos de dispositivos con diferentes resoluciones. Todas estas tecnologías en conjunto con un entorno de desarrollo integrado se complementan entre sí, y por medio del IDE es posible crear aplicaciones de manera integrada y fácil.

Herramientas de Desarrollo Back-End

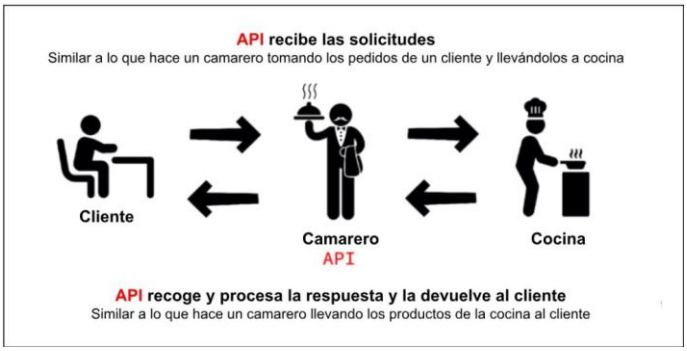
Con relación al desarrollo del back-end es importante mencionar que se compone de dos partes, la primera la cual es el REST-API y la segunda que comprende la base de datos. Primero se debe tener en cuenta que es un API REST, según IBM (2026) un API REST se puede entender como:

> Una API REST es una interfaz de programación de aplicaciones (API) que se ajusta a los principios de diseño del estilo arquitectónico de transferencia de estado representacional (REST), un estilo utilizado para conectar sistemas de hipermedia distribuidos. Las API REST a veces se denominan API RESTful o API web RESTful. (párr.1)

Se puede entender un API como un sistema que es intermediario entre el cliente y la base de datos u otros procesos, un ejemplo pragmático de un API es la del ejemplo del camarero el cual interactúa entre el cliente y la cocina.

Ilustración 5

Ejemplo API.



Fuente: https://appsheet.es/los-webhooks-en-appsheet/

En el API REST será a donde se manejará procesos o tareas del sistema que buscará generar la interacción con la base de datos, como en los módulos de la aplicación, así como procesos más complejos que requieren automatización como la creación de algoritmos para el correcto funcionamiento basado en estándares predispuestos como por ejemplo el módulo de facturación debe generar la factura con un maestro detalle con los cálculos de IVA, excepciones, descuentos, amortizaciones de primas, y la validación del documento previo a su envío al Ministerio de Hacienda basado en el estándar 4.4 el cual a la fecha en que se escribe este documento es el estándar productivo del Ministerio.

Asimismo, se debe considerar cómo funcionan las API que, de acuerdo con IBM (2026):

Las API REST se comunican a través de solicitudes HTTP para realizar funciones de base de datos estándar como crear, leer, actualizar y eliminar registros (también conocido como CRUD) dentro de un recurso.

Por ejemplo, una API REST usaría una solicitud GET para recuperar un registro. Una solicitud POST crea un nuevo registro. Una solicitud PUT actualiza un registro y una solicitud DELETE elimina uno. Todos los métodos HTTP se pueden utilizar en las llamadas a la API. Una API REST bien diseñada es similar a un sitio web que se ejecuta en un navegador web con funcionalidad HTTP integrada. (párr. 10-11)

Para el desarrollo del API REST se hará uso de la herramienta Visual Studio, esta herramienta es un IDE para desarrollo integrado compatible con los lenguajes de programación C#, C++, Visual Basic, sin embargo porque utilizar Visual Studio si ya se está utilizando Visual Studio Code, primero se debe aclarar que estas herramientas no son la misma, y si bien es posible la programación de aplicaciones de C# en Visual Studio Code, se recomienda más hacer uso de Visual Studio ya que este para el lenguaje de programación en cuestión está mejor integrado en esta herramienta ofreciendo un amplio número de plantillas para proyectos avanzados dependiendo de la necesidad del usuario, incluye opciones de depuración avanzada y además de permitir una integridad con el ecosistema .NET con la gestión de paquetes Nuget. Mientras que Visual Studio Code es ideal para proyectos más rápidos con respecto a desarrollo y ligeros en cuanto a la capacidad del programa.

Habiendo explicado la parte del API REST podemos proseguir con la explicación de la parte base de datos, primero definiendo que es una base de datos, según IBM (2025): “Una base de datos

es un repositorio digital para almacenar, gestionar y proteger colecciones organizadas de datos. Los diferentes tipos de bases de datos almacenan datos de diferentes maneras." (párr.1)

Esto quiere decir que una base de datos es una especie de archivero digital donde guardaremos todos los registros de información necesarias para el correcto funcionamiento del prototipo, así como registros de información ya procesada (Facturas, cuentas por cobrar, etc.), el motor de base de datos que se usará para el desarrollo del prototipo será SQL Server 2019, esta es una base de datos creada por Microsoft, la cual se puede definir según Microsoft (2026): "Microsoft SQL Server es un sistema de administración de bases de datos relacionales (RDBMS). Las aplicaciones y las herramientas se conectan a una instancia o base de datos de SQL Server y se comunican mediante Transact-SQL (T-SQL)." (párr.1)

SQL server funciona como un gestor de base de datos relacional, el cual incluye Transact-SQL que es una extensión del lenguaje SQL, lo cual nos permite un lenguaje de programación dentro la base de datos permitiéndonos definir variables, ejecutar comandos de condición, definición de funciones o de procedimientos almacenados, dándonos una funcionalidad más amplia en cuanto al tratamiento de datos, sin necesidad de tener que hacer modificaciones en el código de la aplicación ya que se hacen directamente en la base de datos.

Metodología de Desarrollo

Existen varios tipos de metodologías de desarrollo como metodología espiral, SCRUM, Agile, Cascada, entre otros, para el desarrollo del prototipo se optará por la metodología cascada, la cual se puede definir, según Porras A. (2023) "el cual pretende organizar los proyectos de software por etapas de manera ordenada y secuencial, entre las cuales están por ejemplo, identificar las necesidades del sistema, el análisis, diseño, la codificación, las pruebas y la operación" (p. 17), cascada se define como una metodología que nos permite parametrizar los tiempos de cada fase antes de dar inicio al proyecto, para llevar un cronograma ordenado de cada actividad.

Basado en la ilustración 1 se puede definir que el proyecto poseerá cuatro etapas en modelo cascada, segregándose en:

Análisis

En esta etapa se analizará los requisitos tendrá el prototipo en dos pasos, los cuales el Schneider A (2024) define de la siguiente forma "es cuando los equipos y gerentes estiman los requisitos del proyecto y evalúan la viabilidad de un proyecto o solución propuesta." (párr.8), Una

vez definido los requisitos de entrada y salida del sistema y su interacción con el usuario se proceden a la fase siguiente.

Diseño

Una vez concluida la etapa de análisis, se inicia la fase de diseño, cuyo propósito es definir de manera técnica y estructurada cómo se construirá el prototipo para satisfacer los requisitos previamente establecidos. En esta fase se determina la arquitectura del sistema, los componentes que lo integrarán, la distribución de responsabilidades entre módulos, el diseño de la interfaz de usuario y la experiencia de interacción, así como el comportamiento funcional y no funcional de la aplicación.

Desarrollo

Una vez culminada la fase de diseño, se procede a la etapa de desarrollo, donde las especificaciones técnicas y arquitectónicas previamente definidas se traducen en código fuente ejecutable. Para el desarrollo del prototipo se emplearán los lenguajes de programación C# para la construcción de la lógica de negocio y componentes del lado del servidor, y JavaScript para la implementación de funcionalidades dinámicas en la interfaz de usuario y la interacción en el cliente. Asimismo, se utilizará el sistema de gestión de bases de datos Microsoft SQL Server para la creación, administración y consulta de la estructura de datos, garantizando la persistencia, integridad y consistencia de la información conforme al modelo de datos definido en la fase de diseño.

Pruebas

Se reúne un grupo, casi siempre conformado por personal de aseguramiento de calidad (QA), desarrolladores y personal de negocio que usará la aplicación, con el objetivo de comprobar formalmente que los requisitos definidos durante la fase de análisis se cumplen a cabalidad en el sistema implementado. Durante esta revisión se ejecutan pruebas funcionales y no funcionales, se analizan los resultados obtenidos frente a los criterios de aceptación establecidos y se registran las posibles no conformidades o defectos detectados.

Arquitectura Utilizada

Un aspecto fundamental para definir las bases de un prototipo programado es la definición de la arquitectura en la cual estará desarrollado, una arquitectura de software se puede definir, según Huet (2022) como:

> Un concepto que surge ya en los años 60 y se refiere a una planificación basada en modelos, patrones y abstracciones teóricas, a la hora de realizar una pieza de software de cierta complejidad y como paso previo a cualquier implementación. De esta forma se dispone de una guía teórica detallada que nos permite entender cómo van a encajar cada una de las piezas de nuestro producto o servicio. (párr.2)

Una arquitectura de software determina la estructura del proyecto, estableciendo cual será el funcionamiento de la aplicación, las propiedades que esta tendrá. Para el desarrollo de este prototipo se hará uso de la arquitectura API-REST, la cual se desacopla las aplicaciones entre el cliente y el servidor, la única información que debe poseer el aplicativo (back-end) desde donde se hacen las solicitudes, así como a su vez la aplicación del lado del cliente (front-end) no debe conocer más que la información que obtiene como respuesta de las solicitudes HTTP.

De antemano se mencionó la definición, funcionamiento y propiedades de este tipo de arquitectura, así como se ilustro en Ilustración 2, como se representa a nivel lógico esta arquitectura, por lo cual no se abordará esta información nuevamente. Sin embargo, se hará mención de reglas y buenas prácticas al implementar esta arquitectura, las cuales son fundamentales para cumplir con la calidad de software, como lo son:

Soporte para estructuras de respuestas JSON: al hacer una solicitud a un API no hay respuestas incorrectas a la hora de devolver la información, sin embargo, es importante mantener una homogeneidad con las respuestas, el estándar es que un API al hacer una solicitud nos devuelva una respuesta en JSON (JavaScript Object Notation), como se muestra a continuación.

Ilustración 6

Respuesta JSON.

```
{
  "name" : "Chuck",
  "phone" : {
    "type" : "intl",
    "number" : "+1 734 303 4456"
   },
   "email" : {
     "hide" : "yes"
   }
}
```

Fuente: Ejemplo JSON.

Como se muestra en la ilustración anterior un objeto JSON se estructura bajo el modelo de pares clave–valor, lo que facilita la organización, almacenamiento y acceso eficiente a la información dentro de aplicaciones y sistemas. Cada propiedad se compone de una clave que identifica el dato y un valor que contiene la información asociada, permitiendo una estructura jerárquica clara y de fácil interpretación tanto para humanos como para máquinas.

Autenticación

Es fundamental que un api con estas capacidades posea un tipo de autenticación basado en token ya sea OAuth 2.0 o JWT (JSON Web Token), estos tipos de autenticación si bien pueden ir de la mano es importante conocer las categorías de cada una de ellas, como se muestra a continuación:

Tabla 3

JWT y OAUTH.

Característica	**JWT**	**OAuth**
Lo que es	Un formato de token compacto y seguro para URL para transmitir reclamaciones de forma segura entre partes	Un marco de autorización delegada para otorgar acceso controlado sin compartir credenciales
Propósito principal	Transporte y verificación de reclamaciones (identidad, permisos, etc.)	Otorgar y gestionar de forma segura el acceso a recursos o API
Alcance	Define la estructura y firma del token	Define el proceso para obtener y utilizar tokens
Casos de uso	Autenticación sin estado, acceso a API, tokens de identificación en OpenID Connect	Acceso a API de terceros, inicio de sesión social, control de acceso multiservicio

Característica	JWT	OAuth
Formato del token	Siempre un JWT	Puede ser JWT o token opaco
Gestión estatal	Normalmente apátrida y autónomo	Puede ser sin estado (JWT) o con estado (token opaco)
Revocación	Es más difícil revocarlo antes del vencimiento sin una lista de denegación	Más fácil con la introspección de tokens y la actualización de tokens
Emparejamiento común	Se utiliza a menudo para tokens de identificación y acceso en flujos OAuth/OIDC	A menudo emite tokens en formato JWT

Fuente: JWT vs OAuth

Versionado Semántico

Es una práctica fundamental en el diseño y mantenimiento de APIs y sistemas distribuidos, ya que establece una convención clara para identificar y comunicar la naturaleza de los cambios introducidos en un servicio. La implementación del versionado semántico puede realizarse mediante la URL del end-point, por ejemplo, /api/v1/recursos, o a través de parámetros de solicitud o encabezados HTTP, lo cual permite que múltiples versiones del API coexistan de manera controlada.

Parámetros de Seguridad

Como en todo proyecto enfocado en el desarrollo web, el proyecto contará con una implementación de HTTPS para cifrar la conexión entre el cliente y el servidor, además de políticas de CORS, el cual se puede definir basado en las palabras de Mozilla (2026) como "un mecanismo basado en cabeceras HTTP que permite a un servidor indicar cualquier dominio, esquema o puerto con un origen distinto del suyo desde el que un navegador debería permitir la carga de recursos.", este mecanismo nos permite definir desde el back-end quien tendrá acceso y quien no, basado en la URL desde donde viene la solicitud.

Toma y Gestión de Requerimientos

Esta fase del proyecto representa el punto de partida en la creación del software, ya que marca la transición desde la etapa conceptual y de levantamiento de requerimientos hacia la materialización técnica de la solución, esta etapa se puede definir, citando a IBM. (2026):

> La gestión de requisitos se puede encontrar en la gestión del ciclo de vida de las aplicaciones (ALM), el desarrollo de software, la ingeniería de software y los métodos de desarrollo ágiles. Mientras que la gestión de proyectos en estos escenarios garantiza que el proyecto se complete de manera eficiente y a tiempo, la gestión de requisitos garantiza que el producto final se construya correctamente. (párr.3)

El fin de la toma y gestión de requisitos es puntualizar el funcionamiento que deberá tener el sistema, y las restricciones y libertades que podrá poseer el sistema basado en las herramientas a usar para su creación, así como la arquitectura, metodología de desarrollo.

Durante esta fase se lleva a cabo la elicitación de requisitos, la cual implica la identificación y recopilación de necesidades mediante técnicas como entrevistas, talleres, análisis documental, observación de procesos existentes y revisión de sistemas legados. Estos requisitos se clasifican típicamente en requisitos funcionales, que describen los servicios, comportamientos y reglas de negocio que el sistema debe implementar, y requisitos no funcionales, que especifican atributos de calidad como rendimiento, seguridad, disponibilidad, escalabilidad, usabilidad y cumplimiento normativo.

Es conveniente tener en cuenta cuando se definen los requisitos que se debe contar con un modelo de madurez de capacidad integrado o por sus siglas en inglés, CMMI (Capability Maturity Model Integration) el cual Microsoft, (2025). "ayuda a evaluar la madurez de los procesos y guía la mejora del proceso para producir resultados más predecibles y productos de mayor calidad. También proporciona un enfoque estructurado para la administración de riesgos y para medir la forma en que una organización administra el riesgo." (párr.7). la integración de CMMI en la definición de requisitos contribuye a que estos sean claros, verificables, priorizados y alineados con los objetivos estratégicos de la organización, fortaleciendo la toma de decisiones y el control del alcance del proyecto. En este sentido, el uso de CMMI no solo actúa como un marco de buenas prácticas, sino también como un instrumento para elevar la madurez organizacional, incrementar la predictibilidad de los resultados y mejorar la calidad del software desarrollado, habiendo abordado las bases de la toma de requisitos, podemos proseguir con los pasos con los cuales se definirá los requisitos

Definición de los Diferentes Tipos de Requisitos

Los tipos de requisitos en un proyecto de desarrollo de software pueden clasificarse en requisitos funcionales, requisitos no funcionales y requisitos de dominio, cada uno de los cuales

cumple un rol específico en la definición integral del sistema y en la reducción de ambigüedades durante su construcción, los requisitos funcionales según IBM (2026), "se refieren a la funcionalidad de un nuevo sistema, producto o aplicación. Estos requisitos son medibles y críticos para cumplir con las expectativas y necesidades de los usuarios finales." (p.1), por ende, describen de manera explícita las funcionalidades, servicios y comportamientos que el sistema debe proporcionar para satisfacer las necesidades del usuario y los objetivos del negocio.

Los requisitos no funcionales definen los atributos de calidad y restricciones bajo los cuales deben operar las funcionalidades del sistema, según Mentores Tech (2025). "especifican cómo debe comportarse el sistema más allá de lo que hace. Están relacionados con atributos de calidad, restricciones técnicas, regulaciones y propiedades generales del sistema." (párr.6). Para culminar los requisitos de dominio están relacionados con el contexto específico del negocio o sector en el cual se implementa el sistema, e incorporan reglas, políticas, normativas y conceptos propios del dominio de aplicación.

Consenso e Identificación de Requisitos

Conforme con IBM (2026):

> los requisitos son cualquier información relacionada con las expectativas de las stakeholder y las necesidades del usuario para un nuevo sistema, producto o aplicación. Estas expectativas y necesidades pueden incluir requisitos textuales, casos de uso, diagramas y descripciones de características. (párr.6)

Basado en el escenario previo se entiende que este punto puede desarrollarse como un proceso estructurado y colaborativo que resulta crítico para el éxito de un proyecto de software. En primer lugar, la correcta identificación de los interlocutores implica reconocer y clasificar a las personas de negocio relevantes (usuarios finales, patrocinadores, responsables de negocio, equipo técnico y otros actores clave), asegurando que cada uno represente una perspectiva válida sobre las necesidades del sistema.

El establecimiento de un consenso es posible mediante dinámicas de validación y negociación de requisitos, donde se priorizan, refinan y aprueban de forma conjunta. Herramientas como prototipos, casos de uso, historias de usuario y modelos visuales facilitan la discusión y alineación de expectativas.

Documentación de Requisitos

Tristancho C. (2026) nos declara que un PRD "forma parte del desarrollo de productos y se utiliza para enumerar las funcionalidades que se incluirán en una versión del producto. Se emplea como un medio de comunicación para los equipos de desarrollo y formación de producto." (párr. 3). En un documento de requisitos del producto (PRD) se establece el propósito, las funciones y la conducta que tendrá el producto final basado en las alineaciones de las partes involucradas. Este documento por establecer deberá contener la información de los tipos de requisitos, además de establecer un orden jerárquico para los requisitos.

Estructura Jerárquica de Requisitos

El documento PRD o por sus siglas en inglés Product Requirements Document, será el documento oficial que describirá de manera clara, concisa y estructurada sobre el comportamiento que tendrá el prototipo, según Bhattacharya (2023):

> Así como las personas se esfuerzan por satisfacer sus necesidades fisiológicas y de seguridad antes de alcanzar aspiraciones más elevadas, los proyectos de software primero deben abordar la infraestructura, la seguridad, la mantenibilidad y la confiabilidad antes de lograr la creatividad y la satisfacción del cliente. (párr.40)

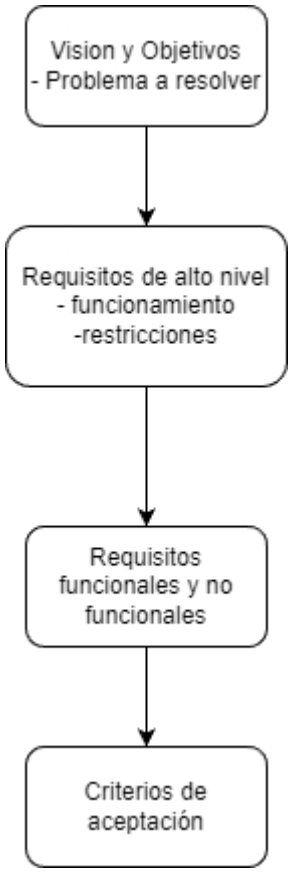
Esta jerarquía facilita la comprensión, gestión y trazabilidad de los requisitos a lo largo del ciclo de vida del prototipo, aun cuando no exista una estructura o nomenclatura estándar para el PRD. Para este proyecto tomaremos una estructura basada en una jerarquía de cuatro niveles, partiendo del nivel más alto de la jerarquía, en donde el PRD concentra la visión y objetivos del producto, definiendo el propósito del prototipo, el problema que se busca resolver y el valor que aporta al negocio. Este nivel está orientado principalmente al negocio estratégicos y responsables de toma de decisiones, y sirve como marco de referencia para todo el proceso de desarrollo, posteriormente, se ubican los requisitos de alto nivel, donde se describen las grandes capacidades o funcionalidades esperadas del sistema, así como las principales restricciones y criterios de éxito. Estos requisitos permiten delimitar el alcance del prototipo y actúan como puente entre la visión del producto y las necesidades concretas de los usuarios.

Se define un nivel intermedio en el cual se toma en cuenta los requisitos funcionales y no funcionales, como lo son los casos de uso, historias de usuario y reglas de negocio. Y para culminar el ultimo nivel el cual compone de los criterios de aceptación y escenarios de validación que nos

permiten de manera subjetiva validar la funcionalidad a cabalidad de estos, en la Ilustración 7 podrá visualizar como se documentará la jerarquía a nivel del documento.

Ilustración 7

Jerarquía de requerimientos.



Fuente: Elaboración propia, 2026

Con base en la imagen anterior, se puede establecer con claridad la estructura jerárquica que seguirá la clasificación de los requerimientos dentro del documento de especificación, permitiendo organizarlos conforme a su nivel de criticidad, dependencia y aporte estratégico, esta jerarquización facilita una priorización fundamentada en el valor de negocio, asegurando que los requerimientos de carácter urgente o de mayor impacto operativo sean atendidos en las primeras fases del desarrollo, además de favorecer la optimización de los recursos disponibles como lo son tiempo, presupuesto y capacidad del equipo, al orientar los esfuerzos hacia los elementos que generan mayor retorno o que resultan indispensables para el cumplimiento de los objetivos del proyecto.

Problemas con la Creación del Documento de Requisitos del Proyecto

Como en todo proceso relacionado con el desarrollo de software puede haber cierta dificultad técnica, organizacional o de comunicación durante el proceso de la recolección de los

requisitos, estos problemas afectan directamente la creación del PRD y por ende la calidad del producto final, a continuación, se mencionará los problemas más comunes los cuales evitaremos en el desarrollo de este proyecto.

Consideración de Requisitos Obvios

Acá se hace referencia a la necesidad de aplicar criterio y juicio profesional al momento de documentar los requisitos dentro de un PRD. El sitio web GeeksforGeeks (2025) nos explica que hay tres tipos de requerimientos, sin embargo enfocados en los requisitos esperados "Estos requisitos son tan obvios que el cliente no necesita indicarlos explícitamente. Ejemplo: protección contra acceso no autorizado" (párr. 8), si bien el objetivo del documento es describir de manera clara el comportamiento esperado del producto, no resulta eficiente ni recomendable detallar exhaustivamente aquellos requisitos que se consideran implícitos, evidentes o universalmente asumidos dentro del contexto del proyecto y del dominio de aplicación.

Identificación Incorrecta de Partes Fundamentales del Negocio

Representa uno de los problemas más críticos durante la definición del documento de requisitos, ya que afecta directamente la alineación del software con los objetivos estratégicos de la organización. Este problema se presenta cuando no se involucra a las partes del negocio adecuados según su rol, conocimiento del dominio y responsabilidad en los procesos del negocio, durante la elicitación y validación de los requisitos. Este escenario suele derivar en conflictos durante las fases posteriores del proyecto, especialmente cuando los usuarios o áreas no representadas identifican que el software no contempla aspectos esenciales para su operación.

Nivel de Detalle que Debe Presentarse

Medhat (2023) nos explica que "La granularidad en la arquitectura de software se refiere al nivel de detalle o abstracción utilizado en el diseño de software, lo que afecta la flexibilidad, mantenibilidad y escalabilidad del sistema" (párr.16), aplica especialmente cuando un mismo requisito puede clasificarse en múltiples categorías Esta ambigüedad genera riesgos de redundancia, inconsistencias semánticas y pérdida de trazabilidad, afectando la claridad del alcance y la correcta interpretación. Si el requisito se replica en varias secciones sin un control estructurado, pueden surgir discrepancias en versiones posteriores, para ello se recomienda el uso de referencias cruzadas a la hora de hacer el documento.

CAPÍTULO III: MARCO METODOLÓGICO

Enfoques de Investigación

Según Sarabia (2025): "La investigación se clasifica, según este criterio, en cualitativa, amparada en el paradigma interpretativo y cuantitativa, amparada en el paradigma positivista. Debe hacerse notar que ambos tipos de investigación son perfectamente válidos para el avance científico "(p.1). Como toda investigación formal, el presente proyecto requiere de la definición explícita de un enfoque metodológico que oriente su desarrollo. Establecer un enfoque claro no solo delimita el alcance y la naturaleza del estudio, sino que también proporciona una estructura lógica para la toma de decisiones, la recolección de información y la validación de resultados.

Enfoque Cuantitativo

El enfoque cuantitativo comprende uno de los paradigmas fundamentales en la investigación, caracterizado por la recolección y el análisis de datos numéricos con el propósito de explicar fenómenos, comprobar hipótesis y responder a una problemática previamente delimitada. Este enfoque se sustenta en la medición objetiva de variables, las cuales deben ser operacionalizadas mediante indicadores observables y cuantificables, permitiendo su análisis estadístico posterior. Podemos ampliar con la siguiente afirmación de Sarabia (2025) "La investigación cuantitativa persigue la demostración de algo por medio de las pruebas de hipótesis al amparo de la estadística y el seguimiento del paradigma positivista." (p. 1).

Enfoque Cualitativo

Para determinar este tipo de enfoque podemos acudir al siguiente enunciado de Valle y Revilla (2023), en el cual nos indica que: ". Se trata, entonces, de comprender la realidad desde la perspectiva de los sujetos; este deseo de comprensión se traduce en los objetivos de la investigación" (p. 11) Dicho esto , se puede entender que un enfoque cualitativo es una forma de investigación orientado a comprender fenómenos desde una perspectiva interpretativa, priorizando el análisis de significados, percepciones, experiencias y contextos específicos, este enfoque emplea técnicas como entrevistas, observación participante y análisis de documentos, buscando identificar patrones, categorías y relaciones conceptuales.

Enfoque Mixto

El enfoque de investigación mixto se fundamenta en la integración sistemática de métodos cualitativos y cuantitativos dentro de un mismo estudio, con el propósito de obtener una

comprensión más amplia, profunda y contextualizada de los objetivos analizados. Este enfoque no se limita a la simple coexistencia de técnicas, sino que articula de manera estratégica la recolección y el análisis de datos numéricos, Padilla-Avalos (2021) nos explica que "dada la naturaleza del problema, se podría concebir un estudio de carácter híbrido. El investigador se podría aproximar al problema, por medio de ambas rutas" (p. 2), lo cual evidencia que la elección de un diseño mixto responde a la complejidad del objeto de estudio y a la necesidad de abordarlo desde múltiples perspectivas analíticas, integrando medición objetiva y comprensión interpretativa en un mismo marco metodológico.

Enfoque de Investigación Seleccionado

Acosta (2023) nos indica que "se puede afirmar que el enfoque cuantitativo es una posición analítica empírica que se caracteriza porque el conocimiento es objetivo y no hay intervención de la realidad, los elementos son considerados como objetos." (p.91), Así que, a lo largo de esta investigación se adopta un enfoque cuantitativo, en virtud de la necesidad de implementar y medir de manera objetiva los procesos que actualmente presentan, deficiencias dentro del negocio. Este enfoque permite traducir los problemas previamente identificados en variables cuantificables, estableciendo indicadores de desempeño (KPIs), asimismo, la aplicación de instrumentos estructurados, como encuestas cerradas, métricas de rendimiento y registros operativos, posibilita generar evidencia empírica verificable, facilitando la comparación entre el estado actual y los resultados esperados tras la implementación de mejoras. Por ende, se selecciona un enfoque cuantitativo por su capacidad objetiva y rigor metodológico en la evaluación de los procesos afectados.

Tipos de Investigación

Citando a Sarabia (2023) que define los tipos como “La clasificación de la investigación responde a diversos criterios que han desencadenado una serie de tratados de clasificación de la investigación lo cual ha permitido tipificarla de distinto modo.” (p.1), la determinación del alcance de la investigación constituye un elemento estratégico dentro del proceso metodológico, ya que actúa como un punto de apoyo entre los hallazgos obtenidos en la revisión.

A continuación, se presenta una breve explicación de los distintos tipos de investigación, analizando las características metodológicas que los distinguen y su pertinencia según la naturaleza del problema planteado. Cada tipo de investigación ofrece ventajas específicas en función del nivel de profundidad, control y alcance que se requiera; sin embargo, también puede presentar limitaciones si no se ajusta adecuadamente a las variables, objetivos y contexto del estudio. En la siguiente ilustración se muestra los tipos de investigación antes de dar una breve descripción de cada una de ellas para dar contexto de estas.

Ilustración 8

Tipos de investigación

Fuente: Imagen generada desde ChatGPT

Basado en la imagen anterior, se procederá a desarrollar una explicación detallada de cada uno de los tipos de investigación presentados, ampliando sus fundamentos conceptuales, características metodológicas y alcances dentro del proceso científico, De esta manera, se busca ofrecer una visión de los conceptos facilitando la identificación según la naturaleza del objetivo de estudio.

Investigación Exploratoria

Una investigación exploratoria se aplica en escenarios donde se desea examinar uno o varios problemas poco conocidos, según Ordoñez (2025): "Busca conocer un fenómeno poco estudiado, con técnicas cuantitativas iniciales como encuestas simples." (p.347), basado en este enunciado se puede decir que una investigación exploratoria se entiende como una estrategia metodológica pertinente cuando el caso de estudio carece de antecedentes empíricos sólidos o presenta un desarrollo teórico incipiente, lo que impide formular hipótesis precisas o establecer variables claramente delimitadas.

Investigación Descriptiva

Este método busca definir las características y comportamientos de un fenómeno o situación sin manipular variables del ambiente, lo que se busca, según, Questa (2022) "el investigador construye una teoría descriptiva que lo guía a lo largo del estudio. Esto supone la identificación de una orientación teórica viable previa a la formulación de las preguntas de investigación." (p. 23)

Se puede inferir que una investigación descriptiva es un enfoque metodológico orientado a caracterizar de manera sistemática un fenómeno determinado, mediante la identificación y especificación de sus propiedades, características y/o perfiles, ya sean estos relativos a personas, procesos u otro tipo de objetos de análisis. Su propósito central no es explicar causalmente los hechos, sino detallar cómo se manifiestan en un contexto específico.

Investigación Correlacional

Tiene como propósito establecer la existencia y el grado de asociación entre dos o más variables de naturaleza cuantitativa, sin que el investigador manipule dichas variables. Este tipo de estudio se centra en analizar cómo cambia una variable en relación con otra, estableciendo la intensidad y dirección de la relación, para ejemplificar, Ordoñez (2025, citando a Hernández et al, 2010) nos menciona que "as investigaciones correlacionales se arraigan a las investigaciones de diseño no experimental y se caracterizan por demostrar relaciones entre 2 o más variables,

evaluando esta relación mediante el empleo de técnicas estadísticas o a través de la asociación de conceptos.” (p. 339).

Investigación Explicativa

Este tipo de investigación van más allá de la definición de conceptos únicamente, Sánchez (2023) citando a Bernal (2016) define este punto como “investigaciones en las que el investigador se plantea como objetivos estudiar el porqué de las cosas, los hechos, los fenómenos o las situaciones.” (p.68), lo que implica un nivel más profundo de análisis orientado a comprender los mecanismos que explican la realidad estudiada y a generar conclusiones con mayor alcance interpretativo.

Investigación Experimental

Basado en la descripción del nombre de este tipo de investigación se puede aludir el enfoque que tendrá, sin embargo, para dar un contexto más amplio, Sarango (2024) lo define como una “Investigación que manipula una o más variables independientes para observar su efecto en una o más variables dependientes, estableciendo relaciones causales.”. Su finalidad radica en comprobar hipótesis mediante la intervención directa del investigador, diferenciándose de los estudios no experimentales al introducir manipulación y control como elementos esenciales para explicar los fenómenos analizados.

Tipo de Investigación Seleccionado

Para esta investigación se opta por un diseño de tipo descriptivo, dado que este permitirá examinar de las distintas variables asociadas a los procesos del negocio que actualmente presentan afectaciones debido a las problemáticas ya identificadas. Mediante la recolección estructurada de datos, provenientes de los futuros usuarios de la aplicación, será posible caracterizar con precisión las condiciones operativas, necesidades funcionales y limitaciones existentes. Este análisis detallado facilitará la identificación de patrones, requerimientos prioritarios y áreas críticas de mejora, proporcionando una base sólida para la toma de decisiones en la etapa de diseño y desarrollo del prototipo, asegurando que la solución propuesta responda de manera objetiva y alineada con el contexto real de uso.

Fuentes de Información

Para el desarrollo de cualquier investigación es necesario recurrir a diversas fuentes de información que respalden teórica y empíricamente el estudio, garantizando la validez, confiabilidad y solidez del análisis realizado. Según Álvarez (2025), “Citar las fuentes muestra a

los lectores cómo el autor ha investigado y llegado a sus conclusiones. Las fuentes proporcionan contexto histórico y permiten a los académicos reconocer el trabajo y las ideas de otros investigadores." (párr.1), dicho esto, La adecuada selección y combinación de estas fuentes permite estructurar un marco teórico consistente que sustente metodológicamente la investigación.

Fuentes Primarias.

Se refiere a los documentos realizados a lo largo de la historia y usados por autor de la investigación como lo pueden ser cartas, guías de entrevista o encuestas, Tigova(2025) nos explica que este tipo de fuentes "ocupa un lugar central, ya que representa la información original, sin alteraciones ni interpretaciones previas, recolectada directamente por el investigador o generada en el momento en que ocurre un fenómeno"(p.1), en consecuencia, las fuentes primarias permiten acceder de manera directa a los datos brutos, garantizando mayor autenticidad, precisión en el análisis, al minimizar la mediación de terceros.

Fuentes Secundarias.

Son aquellos documentos que se generan a partir del procesamiento, análisis y sistematización de la información recolectada mediante fuentes primarias, tales como las respuestas obtenidas en entrevistas, los resultados de encuestas o los registros directos de observación. En este sentido, no presentan datos originales en su estado bruto, sino que los organizan, interpretan y estructuran para facilitar su comprensión y utilización posterior. Como señalan Álvarez C (2023), se trata de "los libros de texto, las obras editadas, los libros y los artículos que interpretan o revisan trabajos de investigación, historias, biografías, críticas e interpretaciones literarias, revisiones de leyes y legislación, análisis políticos y comentarios son ejemplos de fuentes secundarias." (párr. 7).

Fuentes Terciarias.

Vera (2024), nos explica que, "Citan publicaciones previas primarias, secundarias o terciarias para externar una opinión experta acerca de un tema. Son guías físicas o virtuales que contienen información sobre las fuentes secundarias." (p.2). Basado en esta cita, se pueden definir como aquellos recursos que organizan, sistematizan y recopilan información proveniente de fuentes primarias y secundarias, sin aportar datos originales ni análisis propios extensos. Su principal función es la de guiar con el fin de localizar la información relevante sobre un tema en específico, en una investigación académica, las fuentes terciarias suelen utilizarse en las etapas iniciales para

identificar antecedentes, autores relevantes y líneas de investigación relacionadas con el problema de estudio.

Variables

Para el desarrollo de esta investigación, debido a el enfoque cuantitativo seleccionado, es necesario definir cuáles serán las variables para determinar los procesos de recolección de datos, para iniciar debemos de establecer que es una variable, Barillas M (2024) define una variable como "una característica, atributo o propiedad de un objeto de estudio (ya sea una persona, grupo, fenómeno, caso, discurso, etc.) que puede ser observada, descrita, interpretada, medida o procesada para comprender científicamente una porción de la realidad que representa.." (p.53). Estas variables nos ayudarán con la definición, medición y cuantificación de los problemas planteados, estas variables nos darán valor a la investigación al conformarse como una hipótesis, a continuación, se menciona los tipos de variables.

Variables Conceptuales

Psychological Scales (2025), nos aclara que las variables conceptuales son "Una variable conceptual es un elemento teórico esencial que se utiliza para referirse a una idea, cualidad o característica que existe únicamente como un constructo abstracto, en lugar de un valor numérico directo y específico." (párr.1). Hacen referencia a la definición precisa y técnica que se asigna a un término o variable dentro de una investigación, estableciendo con claridad qué se entenderá por dicho concepto en el contexto específico del estudio.

Variables Operacionales

Se pueden definir como el tipo de variable que permite dar claridad para la creación de instrumentos para la recolección de datos, Mamani et al (2024) nos afirma que "Estas variables se caracterizan por ser definidas de manera precisa y transformadas en elementos cuantificables u observables que pueden ser evaluados en un estudio." (p.73). se puede entender que este tipo de variables son requeridas para el desarrollo de una investigación cuantitativa como lo es la presente.

Variables Instrumentales

Estas derivan del tipo de variable anterior, siendo el instrumento o elemento usado para la recolección de datos. Su función principal es servir como una herramienta que permita estimar de manera consistente el efecto causal de una variable independiente sobre una variable dependiente,

para su uso se debe de contar con los tipos de instrumentos de recolección de datos para la investigación como lo son los cuestionarios, escala de actitudes.

Tabla 4

Tabla de variables.

Objetivo específico	**Variable**	**Variable conceptual**	**Variable operacional**	**Variable instrumental**
Analizar los requerimientos funcionales y no funcionales del sistema web.	Análisis de requerimientos	European Knowledge Center for Information Technology. (2023) indica que "Un análisis de requisitos consiste en la recopilación de las necesidades de una empresa para ponerle solución. Para ello, se ha de hacer un estudio interno de la situación actual de la empresa y las problemáticas a las que se enfrenta." (párr.1)	Encuesta Observación	Cuestionario Guía de observación

Objetivo específico	**Variable**	**Variable conceptual**	**Variable operacional**	**Variable instrumental**
Construir el diseño de la base de datos y del sistema web mediante la herramienta Workbench	Diseño de la aplicación	Alvarez O (2022) explican que: "Los patrones de diseño brindan soluciones a problemas que se presentan durante el desarrollo de software, evitan duplicaciones de código y facilitan su reutilización" (p. 2)	Casos de usos. Diagrama entidad relación	MySQL Workbench. Microsoft Word.
Programar los diferentes módulos del sistema mediante la herramienta Visual Studio y Visual Studio Code.	Programar el prototipo	Adobe (2025) expone que "En esta fase, los programadores codifican aplicaciones según los requisitos y especificaciones del proyecto, y también se realizan algunas pruebas e implementación." (párr.15)	Diagrama Entidad-Relación Diagramas UML Documentos de diseño	VS Code, Visual Studio.

Objetivo específico	Variable	Variable conceptual	Variable operacional	Variable instrumental
Probar los diferentes módulos de forma independiente y también de forma integral, garantizando los resultados correctos del sistema.	Pruebas del prototipo	Irrazabal (2022), nos afirma que "tienen como fin observar la ejecución de un sistema para validar si se comporta de la forma prevista o, en caso contrario, detectar sus fallas" (p.11)	Pruebas unitarias	Casos de prueba VS Code

Fuente: Elaboración propia

Población

Julca. J (2024) nos indica que "La población en una investigación se refiere al conjunto total de individuos, objetos o eventos que cumplen con ciertas características específicas y que son el foco de estudio." (párr.2), se puede entender que la población es el conjunto de personas en general con las cuales se espera trabajar, es importante delimitar el tamaño de este grupo con parámetros que nos permita disminuir el tamaño de esta población, la población estará centrada en los empleados activos que pertenecen al departamento de cobros de la empresa Marsh Asprose.

Muestra

Una muestra se puede definir como un grupo selecto de la población o como nos lo aclara Rebollo (2022) "Es una parte o porción extraída de un conjunto por métodos que permiten considerarla como representativa del mismo" (cap.10), la muestra una cantidad de personas a las que se les aplicaran los instrumentos de recolección derivado de la población, es fundamental exponer que la muestra se define basado en la siguiente fórmula:

Ilustración 9

Fórmula de Muestra.

$$n = \frac{N \times Z_a^{\,2} \times p \times q}{d^2 \times (N-1) + Z_a^{\,2} \times p \times q}$$

Fuente: Simeon Pickers, PSYMA

Basado en esta fórmula se puede definir que representa el tamaño de la muestra requerida; **N** corresponde al tamaño total de la población objeto de estudio; $\mathbf{Z_a}$ es el valor crítico de la distribución normal estándar asociado al nivel de confianza seleccionado, **p** es la proporción estimada de ocurrencia del fenómeno o característica en la población; **q** es la proporción complementaria

Ilustración 10

Desarrollo de la formula.

$$n = \frac{30 * 1.96^2 * 0.5 * 0.5}{0.05^2(30-1) + 1.96^2 * 0.5 * 0.5}$$

$$n = \frac{30 * 3.8416 * 0.25}{0.0025 * 29 * 3.8416 * 0.25} = \frac{28.8212}{0.0725 + 0.9604}$$

$$n = \frac{28.812}{1.0329} = 27.9$$

Fuente: Elaboración propia

Para la determinación del tamaño de la muestra se aplicará la fórmula estadística mencionada anteriormente, considerando una población total de 30 personas, un nivel de confianza del 95% y un margen de error del 5%. Bajo estos parámetros, el cálculo arroja como resultado una muestra de 28 sujetos, lo que indica que se deberá recolectar información de 28 participantes mediante la aplicación de encuestas y entrevistas.

Instrumentos de Recolección de Datos

Son los medios técnicos y metodológicos que permiten obtener información relevante para el desarrollo de una investigación, garantizando que los datos recopilados respondan de forma

directa a los objetivos planteados, Vázquez (2025), nos comenta que "Los instrumentos son las herramientas que se utilizan para recolectar datos y así, convertir a un contacto en información útil." (p.17), para esta investigación haremos uso de los instrumentos de recolección de datos que se muestra a continuación

Encuestas

Para esta investigación las encuestas representan el instrumento principal para evaluar de manera objetiva la percepción de las personas de negocio involucradas respecto al funcionamiento que deberá tener la aplicación. Mediante cuestionarios estructurados, se busca medir el nivel de satisfacción general en aspectos como usabilidad, eficiencia y adecuación a los procesos internos, permitiendo obtener indicadores cuantificables que respalden la toma de decisiones. A su vez los resultados de las encuestas facilitarán la identificación de áreas críticas que requieran mayor atención en la fase de análisis y diseño, enfocado al desarrollo del prototipo hacia una solución alineada con las necesidades reales de la organización.

Observación

La observación nos permitirá analizar de manera directa y sistemática la ejecución real de los procesos operativos dentro de la organización, por medio de la observación de las actividades diarias de los funcionarios, se busca identificar inconsistencias, metodologías omitidas e ineficiencias que no han sido reportados formalmente, ya sea porque el personal no los percibe como problemáticos o porque desconoce su impacto en el negocio. En consecuencia, la observación facilitará la detección de requerimientos reales, la validación del comportamiento esperado del sistema y la definición precisa de mejoras que deben incorporarse en el diseño y desarrollo del prototipo.

Proceso para Recolección y Análisis de Datos

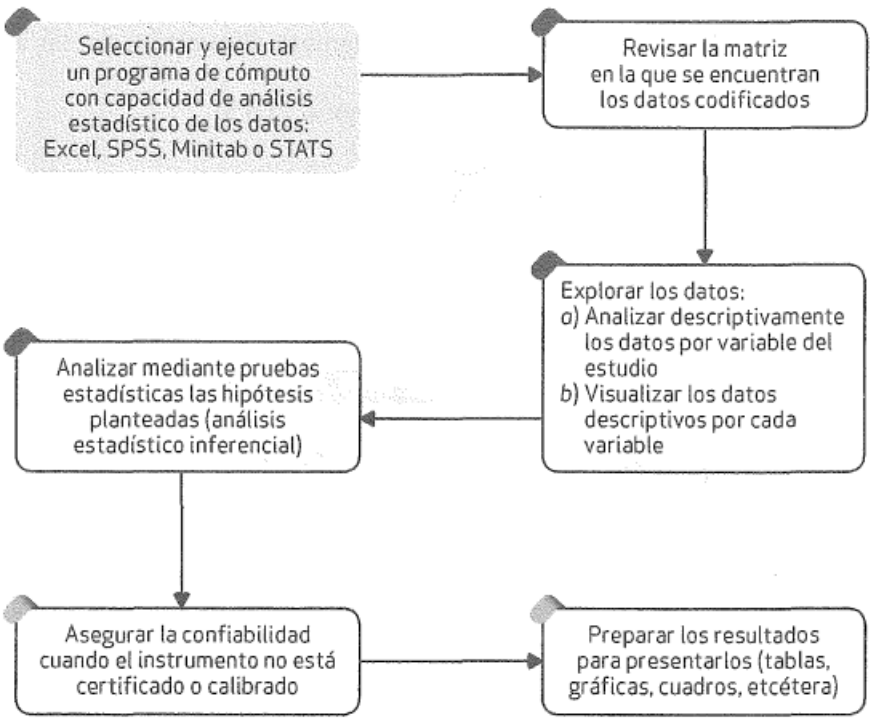
Esta etapa se compone de dos supuestos la cual sería la recolección de datos, la cual Acosta (2023b) nos ilustra en este proceso el cual son "procedimientos y herramientas que permiten la recolección de información, con el objetivo de obtener datos, responder a preguntas de investigación y alcanzar conclusiones." (p.2), se entiende que esta etapa constituye un paso posterior al planteamiento del problema y a la definición de las variables de estudio, y consiste en obtener información empírica de los casos seleccionados, esto implica, medir o capturar la información

pertinente. Para ello se utilizan o desarrollan uno o más instrumentos de recolección de los datos. En este caso se hará uso de las herramientas de encuestas y observación con el fin de obtener información relevante, analizando los procesos actuales del negocio partiendo hacia encuestas relacionadas con los puestos de trabajo y como estas personas desempeñan sus funciones día con día, con las herramientas y datos que utilizan.

Una vez obtenida esta información procederemos con el respectivo análisis de datos, definidas, esto implica una revisión basada en el tipo de enfoque, para esta investigación se hará uso de un enfoque cuantitativo, Vázquez C. (2025) nos explica que "Los métodos cuantitativos se centran en recopilar datos que sean numéricos y medibles; y son ideales cuando quieres obtener información precisa." (párr.14), por ende, se debe hacer un análisis de las variables ya definidas anteriormente, para el análisis se puede guiar con base a la siguiente figura:

Ilustración 11

Diagrama análisis cuantitativo



Fuente: Fundamentos de metodología de la investigación

Este diagrama representa de manera secuencial el procedimiento que se seguirá para el análisis de los datos, iniciando con la verificación y revisión de las variables previamente definidas y codificadas en la matriz de información. A partir de esta validación inicial, se procede a la exploración de los datos mediante técnicas descriptivas que permiten identificar su comportamiento, distribución y posibles inconsistencias. Posteriormente, se aplican las pruebas estadísticas correspondientes para contrastar las hipótesis planteadas, este proceso contribuye a asegurar la confiabilidad y validez de la información obtenida, culminando con la presentación estructurada de los resultados, los cuales servirán de base para la interpretación y toma de decisiones.

CAPÍTULO IV: ANÁLISIS DE RESULTADOS

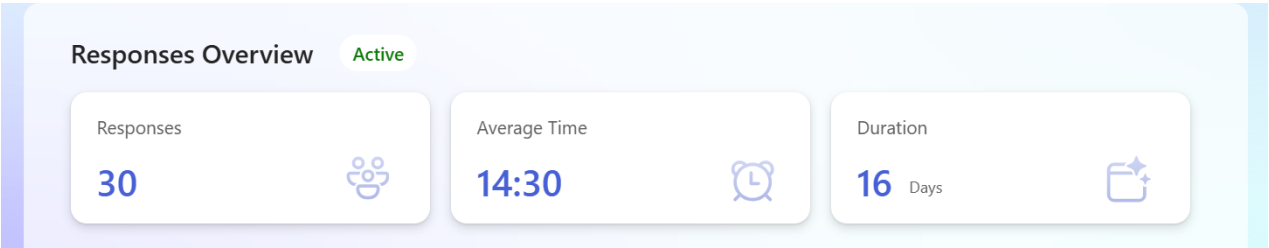
En esta etapa de análisis de resultados es importante recalcar los resultados obtenidos durante la aplicación de los instrumentos, se debe tener en cuenta que, durante la aplicación del instrumento de recolección de datos correspondiente al cuestionario, se obtuvieron resultados específicos enfocados en el área de cobros de la organización, lo cual permitió identificar de manera clara, precisa y objetiva la problemática existente en los procesos asociados a la gestión de cobros. De forma complementaria, se implementó el instrumento de observación directa con el propósito de profundizar en la obtención de información empírica relacionada con las actividades operativas cotidianas del equipo de trabajo, permitiendo analizar el flujo real de los procesos, detectar posibles ineficiencias y validar la consistencia de los datos recopilados mediante el cuestionario. Esta triangulación metodológica contribuye a fortalecer la confiabilidad de los hallazgos y proporciona una base sólida para la posterior formulación de mejoras en los procedimientos evaluados.

Resultados de Cuestionario

A continuación, se presentan los resultados obtenidos por medio del cuestionario incluido en el ANEXO 1. Este instrumento fue diseñado para obtener información relevante para la investigación, y fue respondido por los participantes en un tiempo promedio de 14 minutos, lo cual permitió obtener datos de manera eficiente sin comprometer la calidad de las respuestas. Los resultados reflejan las percepciones y aportes de los encuestados.

Ilustración 12

Total cuestionario

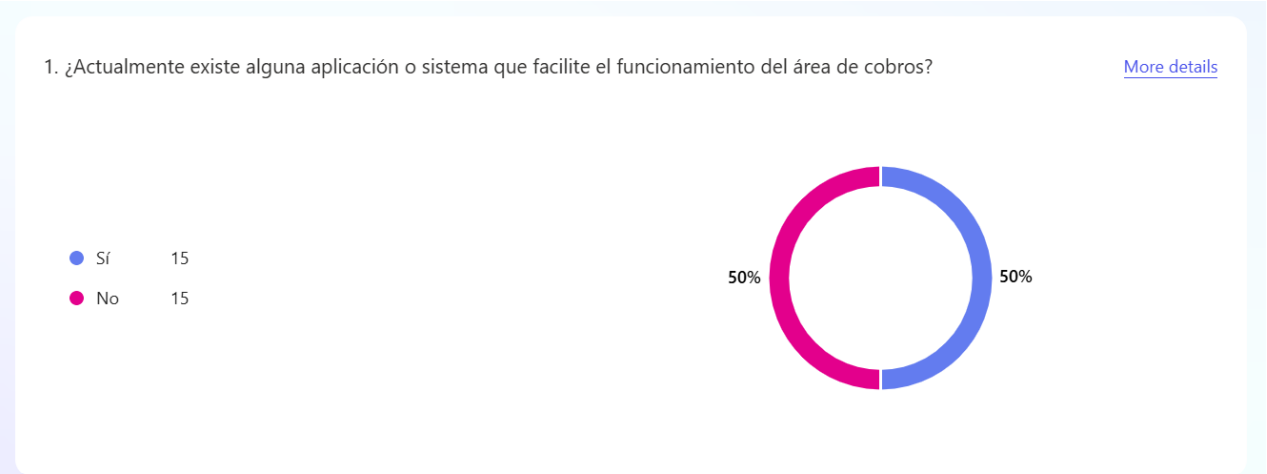


Fuente: Elaboración propia

En la presente imagen se puede observar el total de respuestas obtenidas, evidenciando una muestra compuesta por 30 participantes, conforme a lo establecido previamente en la definición de la muestra. Este número de respuestas permite validar la consistencia de los datos recolectados y garantiza que el análisis se base en la totalidad de los sujetos considerados dentro del alcance del estudio.

Ilustración 13

Primera Pregunta

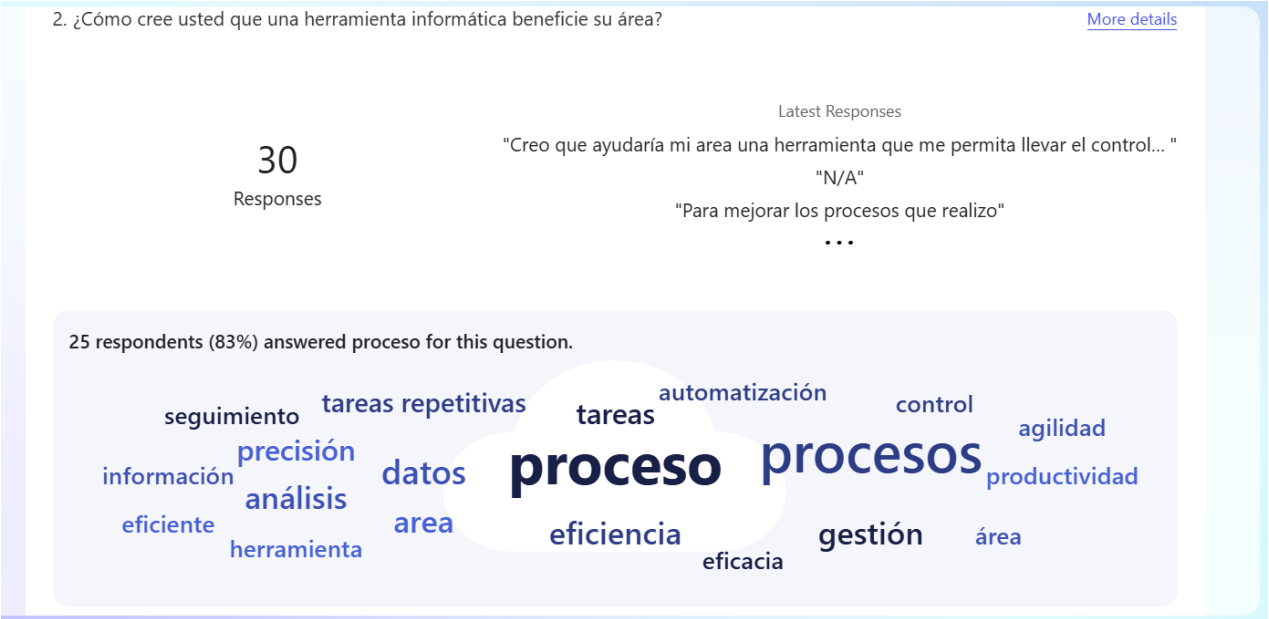


Fuente: Elaboración propia

Como se puede apreciar en la ilustración anterior, existe una diferencia claramente marcada y prácticamente equilibrada entre los encuestados en cuanto a la disponibilidad de herramientas. Esta división evidencia que, a pesar de pertenecer a una misma área, no todos los participantes cuentan con acceso a soluciones de software que faciliten la ejecución de sus funciones. Dicha situación pone de manifiesto una brecha en la adopción tecnológica dentro del equipo, lo cual puede incidir directamente en la eficiencia, estandarización de procesos y productividad general de las actividades desarrolladas.

Ilustración 14

Segunda Pregunta

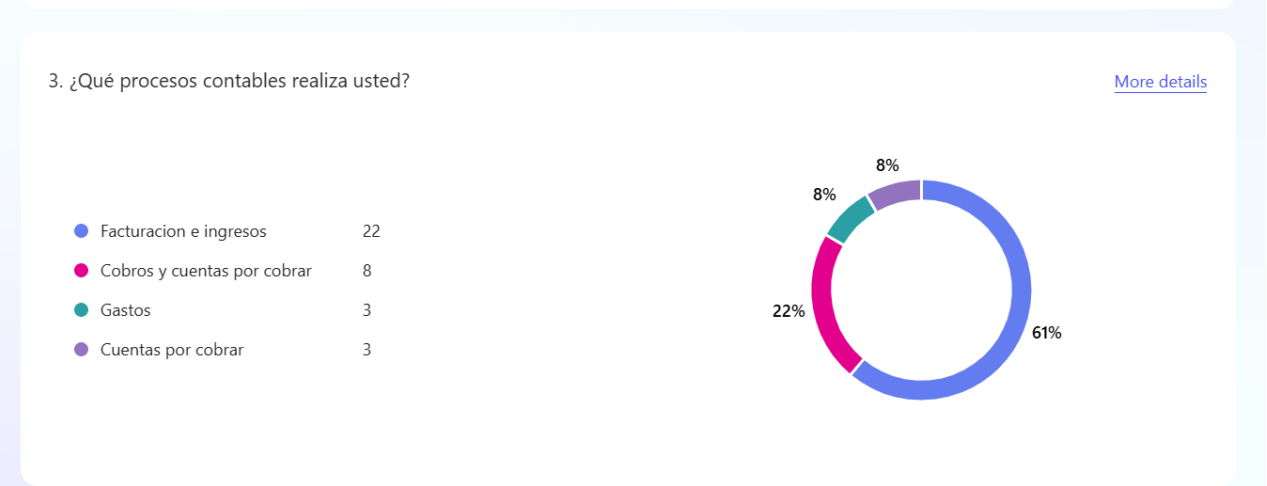


Fuente: Elaboración propia

La ilustración anterior presenta los resultados de una pregunta de carácter abierto, en la cual se identifica un patrón común en las respuestas de los encuestados: la necesidad de optimizar los "procesos". De manera recurrente, los participantes señalan la importancia de contar con una herramienta que les permita mejorar la gestión de estos procesos, destacando especialmente la automatización de tareas repetitivas y la optimización de las actividades actuales. Este hallazgo evidencia una oportunidad clara de implementación tecnológica orientada a incrementar la eficiencia operativa y la calidad en la ejecución de las funciones dentro del área.

Ilustración 15

Tercera Pregunta

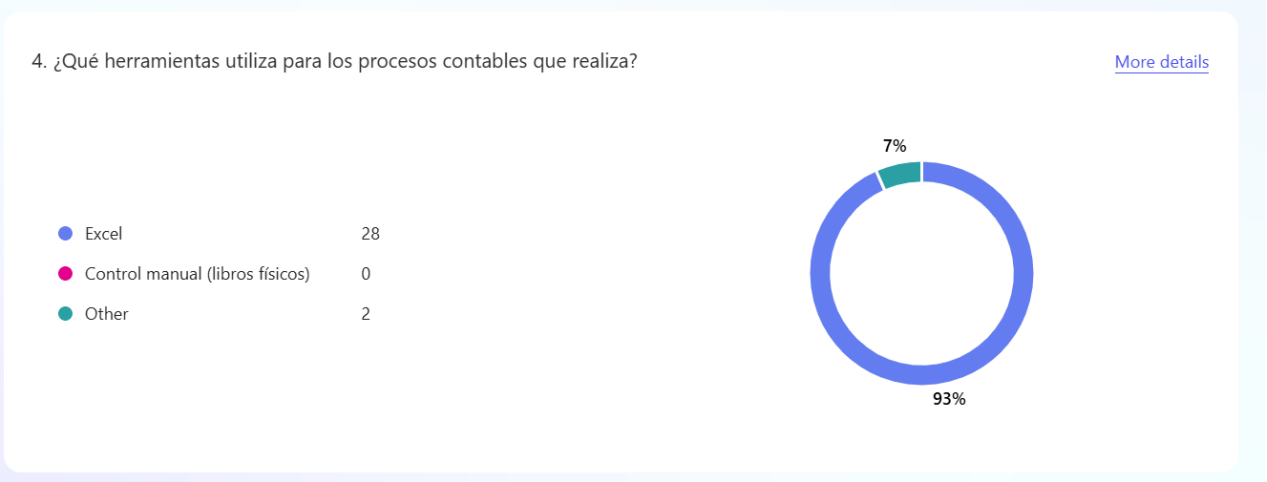


Fuente: Elaboración propia

La ilustración correspondiente a la pregunta tres permite identificar los distintos procesos que ejecutan los colaboradores dentro de la misma área, evidenciando que la mayoría de estos se concentran en actividades relacionadas con la facturación y el registro de ingresos, lo cual los posiciona como un punto de enfoque. En segundo plano, también se destacan los procesos asociados a la gestión de cobros y cuentas por cobrar, reflejando su relevancia dentro del flujo operativo. Este panorama permite delimitar áreas prioritarias para la mejora y optimización, orientando los esfuerzos hacia aquellos procesos con mayor impacto en la gestión financiera.

Ilustración 16

Cuarta Pregunta

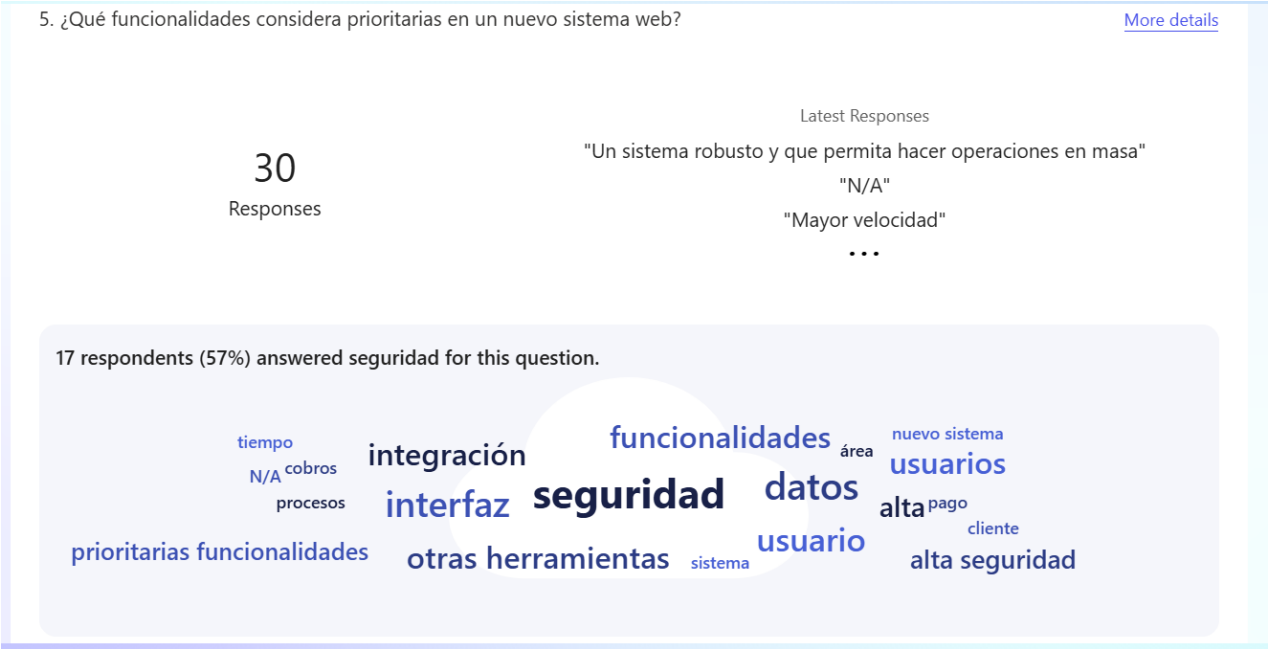


Fuente: Elaboración propia

La cuarta pregunta aborda las herramientas utilizadas por los colaboradores de las distintas áreas, permitiendo identificar una tendencia predominante en el uso de Microsoft Excel como principal apoyo para la ejecución de sus funciones. Este resultado evidencia que la mayoría de los encuestados depende de esta herramienta para la gestión y procesamiento de la información. No obstante, también se observa que un grupo reducido de dos personas utiliza aplicaciones distintas, las cuales, según el análisis de los resultados, corresponden a sistemas o aplicativos internos provistos por la empresa.

Ilustración 17

Quinta Pregunta

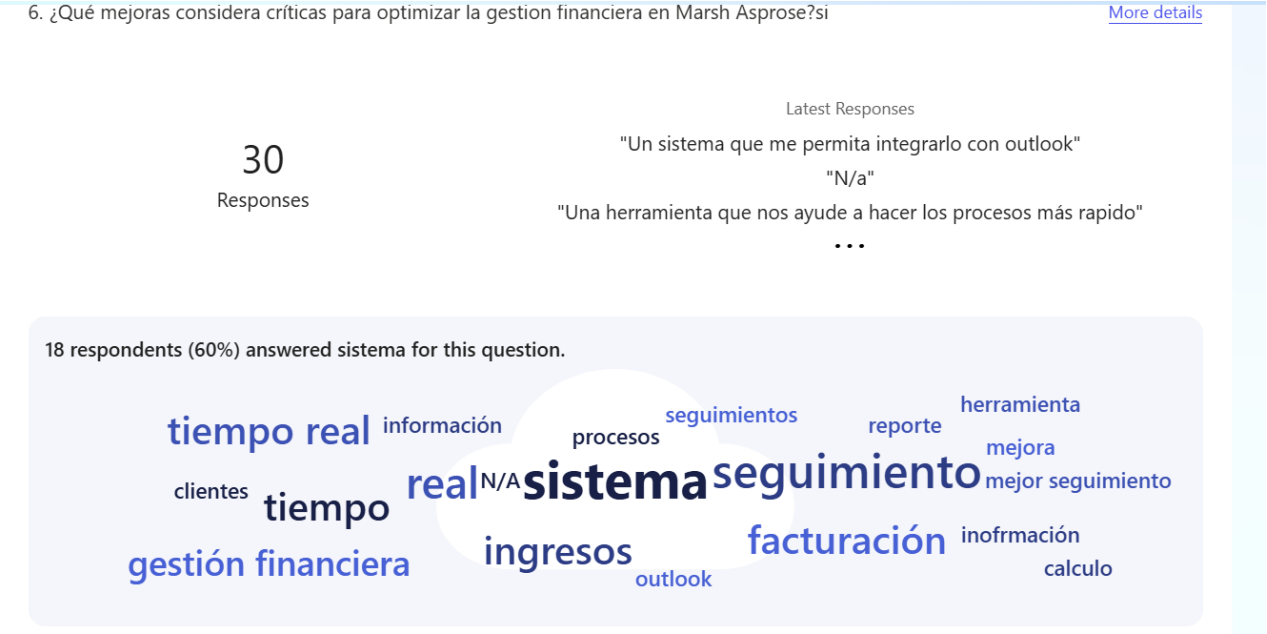


Fuente: Elaboración propia

La quinta pregunta plantea un cuestionamiento de carácter abierto orientado a identificar las funcionalidades que los usuarios consideran esenciales en un nuevo sistema. A partir de las respuestas obtenidas, se puede destacar que la mayoría coincide en señalar la seguridad como un aspecto prioritario, evidenciando la necesidad de proteger la información y garantizar la confiabilidad del sistema. Asimismo, se resalta la importancia de la integración con otras herramientas utilizadas en sus funciones diarias, lo que refleja un interés por contar con soluciones interconectadas que faciliten la continuidad operativa y optimicen los flujos de trabajo dentro de la organización.

Ilustración 18

Sexta Pregunta

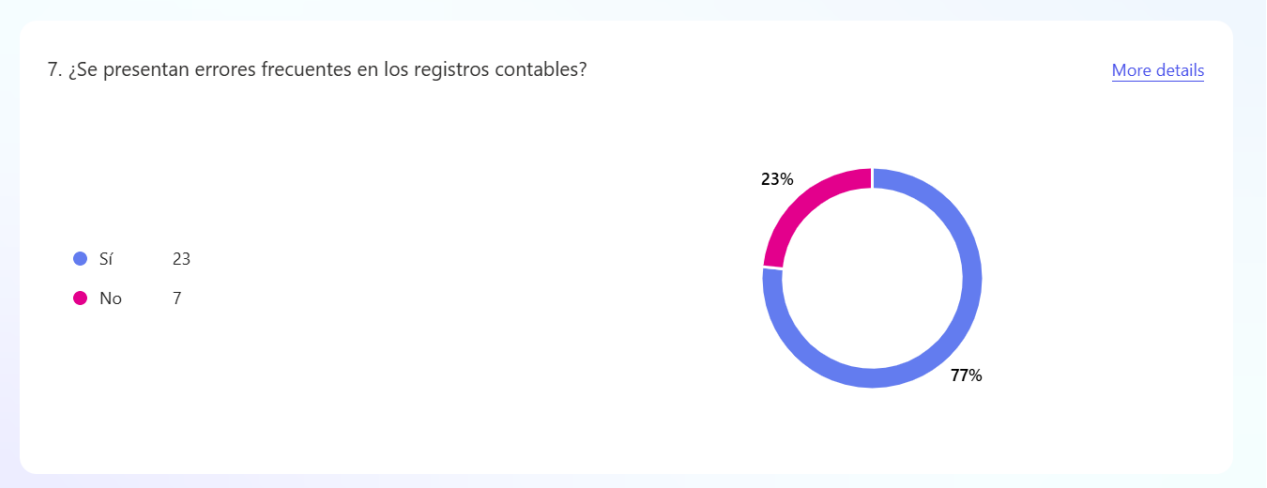


Fuente: Elaboración propia

La sexta pregunta aborta una pregunta abierta relacionada a que creen los usuarios necesarios como algo crítico para optimizar la gestión financiera de la compañía, estas respuestas fueron variadas, como implementación de mejora continua, un sistema integrado que permita eliminar otras aplicaciones innecesarias o complicadas de usar, así como una herramienta que permita manejar de manera exacta la información de clientes, cálculos y reportes.

Ilustración 19

Séptima Pregunta

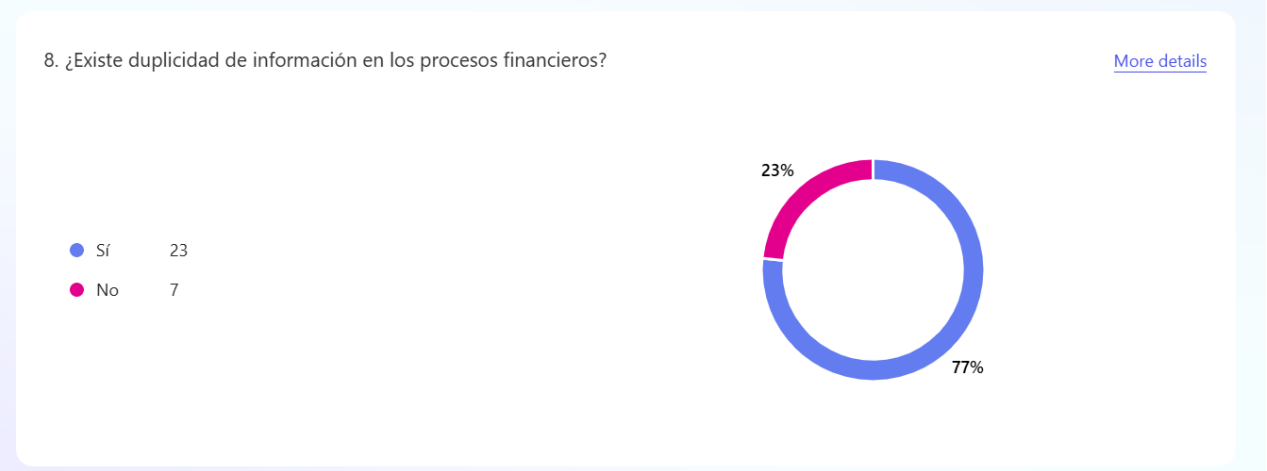


Fuente: Elaboración propia

La séptima pregunta aborda la incidencia de errores en relación con el puesto desempeñado por cada usuario, evidenciando una tendencia significativa dentro de los resultados obtenidos. En este sentido, se observa que 23 de los encuestados han experimentado problemas frecuentes asociados a los registros contables en general, lo cual representa una proporción considerable de la muestra. Este hallazgo pone de manifiesto la existencia de debilidades en los procesos actuales, posiblemente vinculadas a la manipulación manual de la información, la falta de controles automatizados o la ausencia de herramientas especializadas, factores que pueden impactar negativamente en la precisión y confiabilidad de los datos financieros.

Ilustración 20

Octava Pregunta

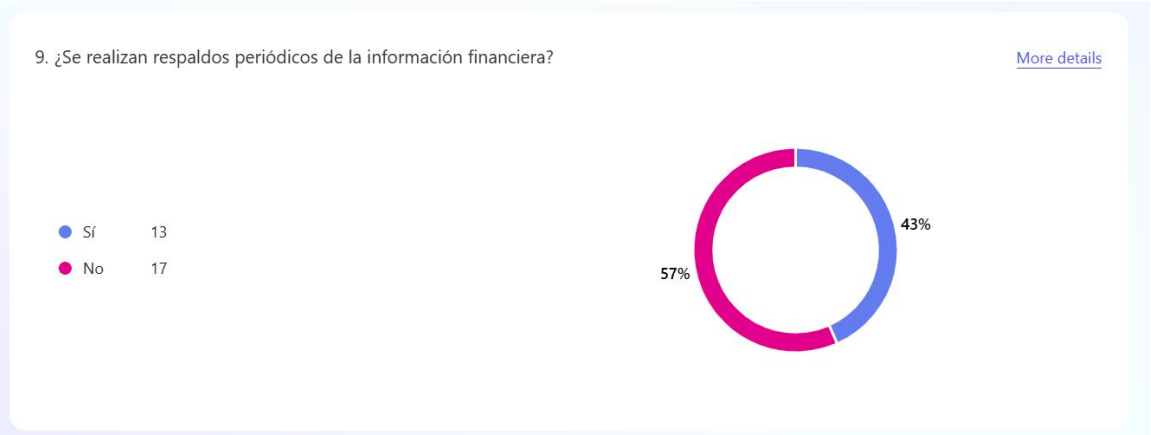


Fuente: Elaboración propia

La pregunta número ocho aborda específicamente la problemática de la duplicidad de registros, permitiendo identificar una relación directa con los resultados obtenidos en la pregunta número siete. A partir de este vínculo, se puede inferir que la duplicidad constituye uno de los errores más frecuentes dentro del sistema, evidenciando deficiencias en los controles de validación y en los mecanismos de gestión de la información. Esta situación refuerza la necesidad de implementar soluciones que permitan prevenir la generación de registros duplicados, garantizando así la integridad, consistencia y confiabilidad de los datos procesados.

Ilustración 21

Novena Pregunta

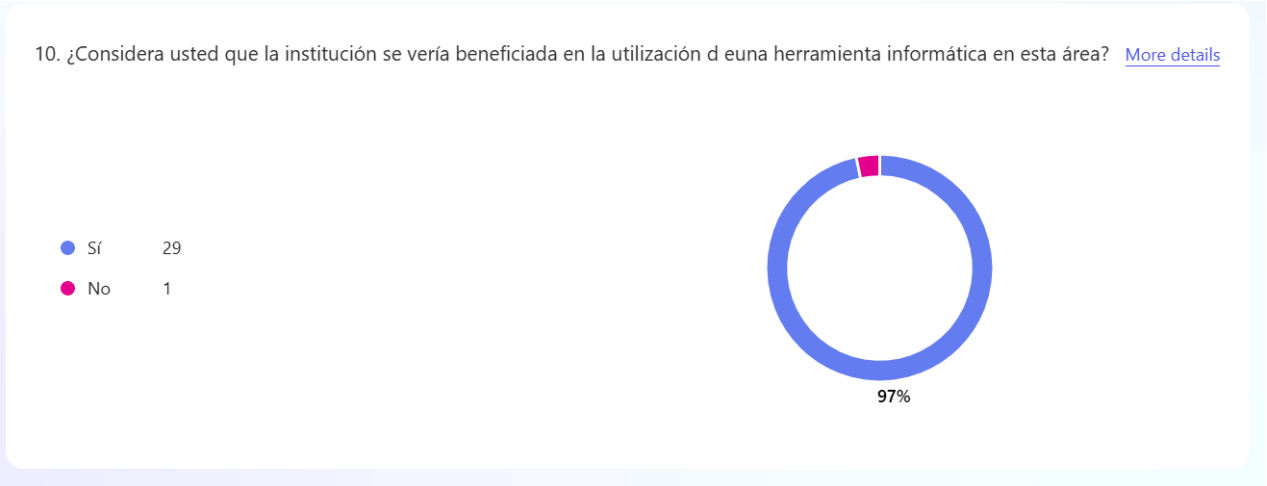


Fuente: Elaboración propia

La novena pregunta analiza el respaldo actual de la información financiera en repositorios alternos, evidenciando una tendencia preocupante en los resultados obtenidos. La mayoría de los encuestados indica que no realiza respaldos periódicos de la información que gestiona, lo cual representa un riesgo significativo para la integridad y disponibilidad de los datos. Esta situación puede deberse tanto a la falta de cultura o conocimiento sobre la importancia de los respaldos, como a limitaciones operativas o tecnológicas que dificultan su ejecución. En consecuencia, se identifica una necesidad clara de implementar políticas y herramientas que faciliten y automaticen los procesos de respaldo.

Ilustración 22

Décima Pregunta



Fuente: Elaboración propia

La décima y última pregunta evidencia un alto nivel de consenso entre los encuestados, donde un 97% manifiesta que la implementación de una herramienta informática orientada al manejo contable resultaría altamente beneficiosa. Según las respuestas, dicha solución permitiría optimizar los procesos no solo del área específica, sino también impactar positivamente en la operación general de la compañía. Este resultado refleja una clara aceptación y necesidad de adopción tecnológica, posicionando la implementación de un sistema como una oportunidad.

Como conclusión de esta cuestionario se puede definir que, actualmente, el equipo enfrenta una brecha operativa marcada por la desigualdad en el acceso a herramientas y una alta dependencia de procesos manuales y de Microsoft Excel como soporte principal, la implementación de un software contable integrado y automatizado no es solo una solicitud de los colaboradores para facilitar su trabajo diario, sino una medida necesaria y respaldada para garantizar la seguridad, confiabilidad y eficiencia financiera de la organización.

Resultados de Observación

A continuación, se presentan los resultados obtenidos a partir de la observación, basada en el ANEXO 2. Dado que se utilizó un único formulario como instrumento de recolección de datos,

se posibilita la exposición íntegra y consolidada de los resultados, facilitando una interpretación clara, coherente y unificada de la información obtenida.

Tabla 5

Resultados de Observación

No	Aspectos por observar	Cumple	No Cumple	Oportunidad de mejora	Detalle de Observación
1	Existencia de un sistema actual para la gestión contable (manual, Excel).			x	Actualmente, si bien existen herramientas, no todas las personas del área pueden tener acceso a ellas
2	Procedimiento actual para el registro de **Cuentas por Cobrar**.	x			La empresa ejecuta los procedimientos de cuentas por cobrar, sin problemas de aplicación
3	Control y seguimiento de morosidad en cuentas por cobrar.			x	Si bien, la empresa posee un control integro, este se lleva de manera sincronizada en un Excel compartido
4	Procedimiento de aprobación y registro de **Cuentas por Pagar**.			x	La empresa, logra completar el procedimiento de cuentas por pagar, siguiendo un proceso manual
5	Control de vencimientos y programación de pagos a proveedores.	x			Este proceso se maneja de manera correcta en la empresa, por medio de una herramienta de software.
6				x	el proceso, se maneja con un proveedor externo el cual da una solución de software que solo

No	Aspectos por observar	Cumple	No Cumple	Oportunidad de mejora	Detalle de Observación
	Flujo actual del proceso de **Facturación**				permite la generación de documentos electrónicos, todo reporte debe ser solicitado a dicho proveedor.
7	Existencia de control presupuestario formal (asignación, ejecución y seguimiento).		x		No se tiene claro la asignación y control presupuestario de la compañía.
8	Procedimiento para el registro de **Ingresos** no relacionados con facturación.			x	Todo ingreso diferenciado de una factura es registrado en un libro de Excel.
9	Procedimiento para el registro y clasificación de **Gastos**.	x			El proceso de registro de gastos es manejado por una parte del equipo con acceso a un sistema de control de gastos.
10	Existencia y actualización del **Catálogo de Cuentas Contables**.			x	La empresa posee un catálogo de cuentas contables, dicho catálogo se encuentra en un archivo compartido.
11	Conciliación bancaria periódica y control de diferencias.		x		La conciliación bancaria no es realizada por el equipo, si no que se delega a un proveedor

No	Aspectos por observar	Cumple	No Cumple	Oportunidad de mejora	Detalle de Observación
12	Generación de reportes financieros (Balance General, Estado de Resultados, Flujo de Caja).			x	Todo reporte financiero es generado mediante tres formas distintas: - Por el proveedor: crea reportes desde la base de datos del software. - Creado a mano: con información base de reportes del proveedor y otros datos recolectados. - Generado desde la herramienta de software: como procesos de cuentas por pagar y gastos.
13	Trazabilidad de transacciones (historial de cambios).			x	Toda la data transaccional se documenta en formato xlsx y se sincroniza automáticamente en un repositorio compartido.
14	Nivel de automatización de procesos financieros.		x		Se identifica una ausencia de flujos de trabajo en los procesos centrales del departamento.
15	Tiempo promedio de registro de una transacción.			x	Se ha identificado una latencia de ejecución de entre 4 y 7 minutos para el completamiento de cada registro.

Fuente: Elaboración propia

Basado en los resultados de la observación se pudo evidenciar que la operación actual de Marsh Asprose es funcional, pero altamente ineficiente e insegura bajo el esquema actual. El equipo invierte un tiempo valioso en labores operativas de bajo impacto (como cruzar datos en Excel), tiempo que puede ser invertido en otras tareas, lo que evidencia la urgencia de integrar soluciones tecnológicas que centralicen, automaticen y democraticen el trabajo del área.

CAPÍTULO V: PROPUESTA

Esta sección nos permitirá una comprensión más amplia de del desarrollo del proyecto partiendo desde las etapas de desarrollo definidas previamente (Análisis, Diseño, Desarrollo y Pruebas), esto con el fin de crear una herramienta final satisfactoria que cumpla con las expectativas y parámetros definidos por las necesidades del negocio.

Análisis

Tal como se indicó en la justificación técnica del presente documento, se prevé la utilización de la infraestructura de hardware actualmente disponible en la organización, optimizando así los recursos tecnológicos existentes y reduciendo la necesidad de inversiones adicionales en etapas iniciales. Asimismo, se aprovechará el licenciamiento de software ya adquirido por la compañía, específicamente herramientas de desarrollo y gestión de bases de datos como Microsoft Visual Studio Professional y Microsoft SQL Server, las cuales permiten soportar de manera adecuada las actividades de construcción, pruebas y validación del prototipo. Durante estas primeras fases del proyecto no se contempla el uso de licencias adicionales relacionadas con servicios de infraestructura externa, tales como contratación de espacio en datacenter o soluciones de respaldo en la nube; no obstante, estos componentes serán evaluados e incorporados en fases posteriores a la etapa de pruebas, una vez finalizado el desarrollo y validadas las funcionalidades principales del sistema.

La solución de software se integrará mediante una arquitectura de tipo RESTful, la cual facilita la interoperabilidad entre los distintos componentes del sistema a través de servicios web desacoplados, permitiendo una comunicación eficiente entre la capa de presentación, la lógica de negocio y la base de datos. Este enfoque arquitectónico favorece la escalabilidad, mantenibilidad y flexibilidad de la solución, tal como se ilustra en la Figura 2.

En lo referente a la capa de telecomunicaciones, se hará uso de una red corporativa de tipo WAN con exposición controlada a internet, necesaria para permitir la integración con servicios externos requeridos por la solución, tales como el consumo del API de la Ministerio de Hacienda de Costa Rica para procesos relacionados con validación o intercambio de información fiscal. El acceso a la aplicación se realizará a través de la red pública, debido a que el sistema estará orientado a un esquema de disponibilidad de cara a internet, permitiendo el acceso simultáneo de usuarios pertenecientes a equipos de trabajo distribuidos geográficamente.

La comunicación se establecerá mediante el puerto 443, el cual se encuentra asociado al protocolo HTTPS, garantizando la transmisión de la información a través de un canal cifrado basado en certificados digitales SSL/TLS, asegurando así la confidencialidad, integridad y autenticidad de los datos intercambiados entre cliente y servidor. Adicionalmente, al utilizar la infraestructura de hardware actualmente disponible en la organización, no se prevé la generación de costos adicionales asociados a servicios de hosting, adquisición de equipamiento especializado ni implementación de protocolos de red adicionales, optimizando de esta manera los recursos tecnológicos existentes durante las fases iniciales del proyecto.

Análisis Detallado del Software a desarrollar

Basado en Turturro (2023) el análisis y diseño de software se puede interpretar como "un enfoque estructurado para desarrollar y mejorar sistemas, que abarca tanto aspectos técnicos como de gestión." (párr.7), se puede entender como el inicio de la fundación de un proyecto de software el cual será utilizado como una herramienta, la cual vendrá a solucionar la problemática por la cual fue concebida en primer lugar. Habiendo establecido la definición podremos describir como serán desarrollados mediante una breve descripción de los módulos a implementar.

Facturación

Este módulo permite la generación de facturas electrónicas asociadas a comisiones o servicios adicionales, asegurando en todo momento el cumplimiento de las normativas fiscales establecidas por el Ministerio de Hacienda, conforme a la versión 4.4 de la facturación electrónica. Adicionalmente, permite la emisión de notas de crédito y débito en los casos en que corresponda. El proceso contempla la validación de los datos del cliente, la verificación de los montos a facturar y la aplicación de los impuestos correspondientes, garantizando la integridad y exactitud de cada documento generado. De igual forma, el módulo habilita la consulta del historial de facturación por cliente, la emisión de copias electrónicas y el control de los distintos estados de las facturas, tales como pendientes, pagadas o anuladas, lo que proporciona una trazabilidad completa sobre el ciclo de vida de cada comprobante.

Gestión presupuestaria

Este módulo se encarga de administrar los presupuestos asignados a las distintas áreas, proyectos y unidades de negocio dentro de la compañía. Entre sus funcionalidades principales se encuentra la definición de presupuestos anuales o periódicos, ajustados a las necesidades específicas

de cada departamento, así como la asignación de partidas según el tipo de gasto o el centro de costo correspondiente. Asimismo, permite el control de la ejecución presupuestaria mediante la comparación continua entre los montos planificados y los gastos efectivamente realizados, lo que posibilita identificar desviaciones de manera oportuna. En este sentido, el módulo incorpora un sistema de alertas por correo electrónico que notifica de forma automática a los responsables cuando la ejecución de un presupuesto se aproxima o supera los límites establecidos, favoreciendo una gestión proactiva y la toma de decisiones a tiempo. Finalmente, para garantizar la trazabilidad de cada transacción y su impacto sobre el presupuesto general, este módulo se integra con los módulos de gastos y cuentas por pagar, consolidando así una visión unificada y coherente de la situación financiera de la compañía.

Ingresos

Este módulo se encarga de la administración de los ingresos de la compañía, contemplando tanto aquellos derivados de los procesos de facturación como los provenientes de ingresos extraordinarios o no previstos dentro de las cuentas por cobrar. De esta forma, centraliza el registro y seguimiento de todas las entradas de capital, independientemente de su origen. Adicionalmente, el módulo incorpora un algoritmo de conciliación automática que coteja los ingresos efectivamente percibidos contra los registros de las cuentas por cobrar, agilizando la detección de discrepancias entre lo facturado y lo recaudado, y reduciendo la carga de las tareas manuales. Esta capacidad facilita, asimismo, la identificación de tendencias, el control del flujo de efectivo y la generación de análisis financieros que respaldan la toma de decisiones estratégicas dentro de la compañía.

Mantenimiento

El módulo de mantenimiento es el encargado de ejecutar las operaciones de creación, edición, actualización y eliminación de registros dentro del sistema. Su función resulta esencial para asegurar que los datos permanezcan vigentes, íntegros y libres de inconsistencias. Mediante este componente se administran los catálogos, las cuentas de usuario, los perfiles y las demás entidades requeridas para la correcta operación de la plataforma. El acceso a sus funcionalidades se encuentra controlado según los perfiles y permisos previamente establecidos. De esta manera, el módulo no solo favorece la estabilidad del sistema, sino que también simplifica su gestión a lo largo del tiempo, al disminuir la necesidad de recurrir a intervenciones técnicas externas.

Consultas

El módulo de consultas permite acceder de forma directa a la información registrada en los módulos previamente descritos, sin que sea necesario generar un reporte formal. Su propósito es ofrecer una vista ágil y en tiempo real de los datos almacenados, facilitando así el seguimiento y control cotidiano de la información contenida en el sistema. De esta manera, los usuarios pueden verificar el estado de los registros de manera inmediata, lo que agiliza la toma de decisiones operativas y la supervisión continua de los procesos.

Reportes

Este módulo se encarga de consolidar la información proveniente de las distintas tablas del sistema, así como de ejecutar los cálculos requeridos para la elaboración de los reportes. Dichos reportes podrán generarse en formato XLSX o PDF, según las necesidades del usuario, y admitirán la aplicación de diversos parámetros en forma de filtros. Estos filtros se alimentan a partir de la información ingresada en los módulos anteriormente mencionados, lo que permite obtener resultados precisos, personalizados y ajustados a los criterios de consulta definidos en cada caso.

Seguridad

La función principal de este módulo es la administración y gestión de los usuarios del sistema, abarcando los distintos roles definidos dentro de la plataforma: usuarios estándar, administradores, auditores y usuarios de procesos, quienes son los responsables de ingresar la información al sistema. A través de este módulo se controlan las credenciales y los privilegios de acceso asociados a cada perfil, garantizando que cada usuario interactúe únicamente con las funcionalidades y los datos que le corresponden según su nivel de autorización.

Análisis Detallado del Hardware

Primero debemos definir lo que implica el hardware, Universae (2025) lo define como: "Cuando hablamos del hardware, hacemos referencia a la parte física y tangible de los dispositivos digitales. Se trata del soporte palpable, que permite que los programas se ejecuten y se visualicen, entre otras muchas funcionalidades." (párr.2).

Para el proceso de desarrollo y pruebas del software se requiere un equipo de hardware lo suficientemente para ejecutar entornos de desarrollo como lo serán el SQL Server, MySQL

Workbench para el diagramado y herramientas de desarrollo como Visual Studio para el back-end, en la siguiente tabla se muestra los requerimientos mínimos de un entorno de desarrollo

Tabla 6

Requerimientos Mínimos Hardware

Componente	Especificaciones	Costo
Laptop	Procesador Intel Core I5 de octava generación, 8 GB de RAM, almacenamiento de 512 GB. Tarjeta gráfica NVIDIA MX230	¢345,000.00
Total		¢345,000.00

Nota: Dicho equipo de desarrollo es propiedad del estudiante por lo cual no representa un gasto durante la elaboración del proyecto.

Una vez finalizado el desarrollo del prototipo, este deberá ser desplegado en un ambiente productivo. Al tratarse de un sistema web orientado a un número reducido de usuarios concurrentes, no se requiere infraestructura de alto rendimiento ni equipos especializados. El sistema fue concebido para operar de forma eficiente con recursos tecnológicos accesibles y de bajo costo, lo que favorece tanto su viabilidad técnica como económica.

En consecuencia, es posible aprovechar la infraestructura con la que actualmente cuenta la empresa, basada en servidores virtualizados. Estos disponen de las siguientes especificaciones: un procesador Intel Xeon a 2.60 GHz, 16 GB de memoria RAM y un esquema de almacenamiento distribuido en tres discos que suman una capacidad total de 300 GB.

Análisis Detallado de los Elementos de Telecomunicaciones

Dado que la aplicación, tanto el front-end como el back-end, se publicará en un servidor localizado en Dallas, se puede establecer como premisa que cualquier colaborador de la compañía que cuente con conexión a internet podrá acceder a ella a través del enlace de acceso correspondiente. A continuación, se presenta una ilustración del flujo de acceso a la aplicación a través de la red.

A nivel de servidor, la aplicación contará con un certificado SSL, de modo que la comunicación entre el cliente y la aplicación, así como la comunicación entre el front-end y el back-end, se realizará mediante una conexión segura a través del puerto 443. Por su parte, la conexión con la base de datos se establecerá mediante el puerto 3433, el cual fue seleccionado por seguridad aplicando Security Thorugh Obscuroty, el cual OKTA (2024) lo define como "La seguridad por ocultación busca mantener un sistema seguro manteniendo en secreto el conocimiento del mismo. Los mecanismos internos y el funcionamiento del sistema se mantienen en secreto, solo para quienes necesitan saberlo." (párr.2). Adicionalmente, a la salida de los equipos se dispone de un intermediario de red para el acceso remoto, correspondiente a la plataforma Zscaler (2026), cuya funcionalidad se describe de la siguiente manera:

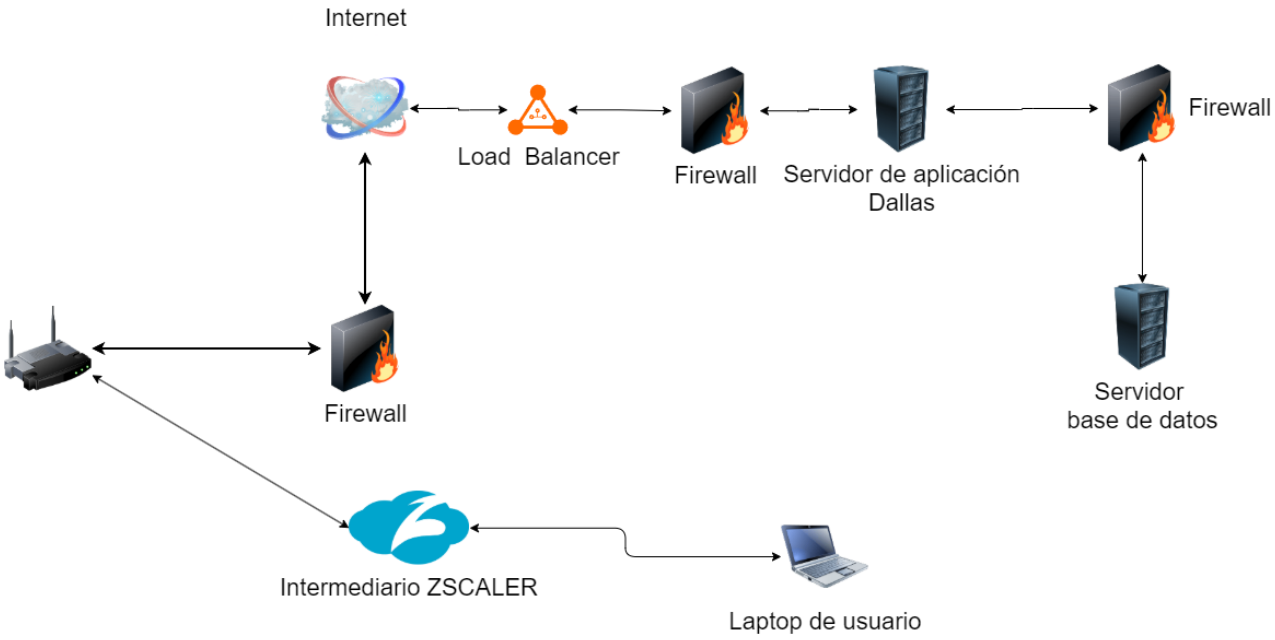
> Proporciona un medio para que se conecten a un centro de datos, red, aplicaciones o recursos en la nube con sus dispositivos remotos a través de conexiones a Internet de redes wifi públicas o domésticas no seguras en lugar de a una red corporativa. (Párr. 1)

Esto permitirá el ingreso a la red corporativa mediante una conexión VPN y, en consecuencia, el acceso a la aplicación en cuestión.

Cabe destacar que toda esta infraestructura, puertos, certificados y aplicaciones intermediarias, ya forma parte de los recursos con los que cuenta la compañía, por lo que su implementación no representa ningún costo adicional.

Ilustración 23

Diagrama Red



Fuente: Elaboración propia

Esta arquitectura garantiza un nivel de seguridad alineado con los estándares vigentes de la compañía. Al integrarse sobre la infraestructura actual, permite una implementación inmediata de la aplicación sin requerir modificaciones estructurales ni ajustes en los puntos de red existentes.

Análisis Detallado del Conocimiento Básico del Talento Humano

El sistema web para la gestión financiero contable para Marsh Asprose priorizará la facilidad de uso para garantizar la autonomía de usuarios sin conocimientos avanzados en sistemas. No obstante, se establece como perfil mínimo del usuario el dominio de capacidades ofimáticas fundamentales, incluyendo el manejo operativo de hardware de entrada y la navegación funcional en entornos web.

Arquitectura detallada de herramientas técnicas

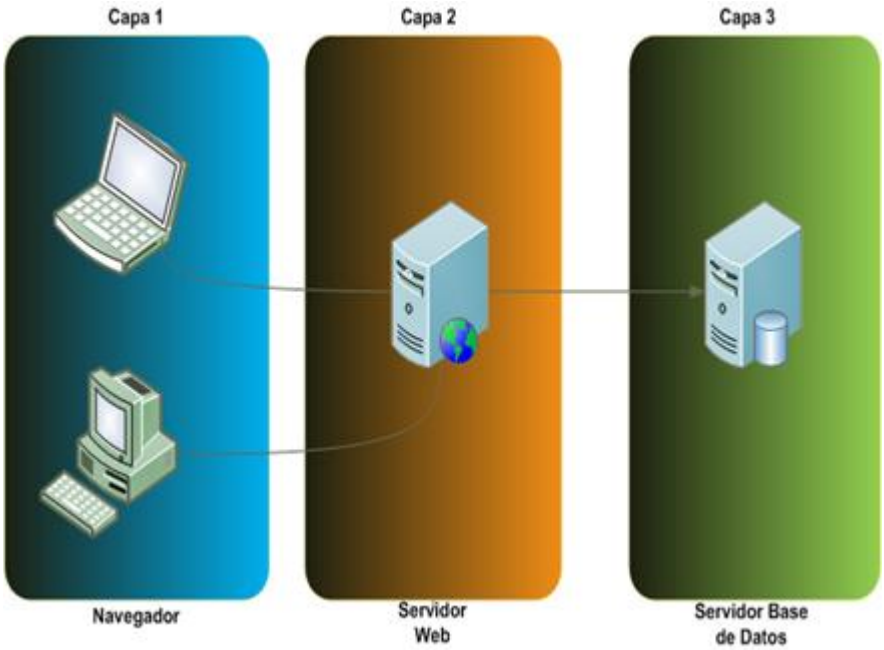
Como se ha mencionado con anterioridad la arquitectura a usar será tipo RESTFul, la cual se puede definir, según Caules (2023) como: “un ESTILO de Arquitectura a la hora de realizar una comunicación entre cliente y servidor” (párr.2). Esto nos permite definir que la arquitectura que se utilizará será un estilo diferente de la de cliente servidor, variando en componentes como el uso de la definición de “Recursos”, pero que es un recurso, Caules (2023) nos explica que: “Un recurso

hace referencia a un concepto importante de nuestro negocio (Facturas, Cursos, Compras etc.). Es lo que habitualmente se denomina un objeto de negocio." (párr. 7).

Esto nos permite organizar de cierta manera los recursos de manera independiente, permitiéndonos una reusabilidad del código haciendo uso del mismo recurso variando la acción HTTPS que vayamos a realizar (GET, POST, PUT, DELETE), a continuación, se muestra un diagrama de la arquitectura a utilizar:

Ilustración 24

Imagen Arquitectura

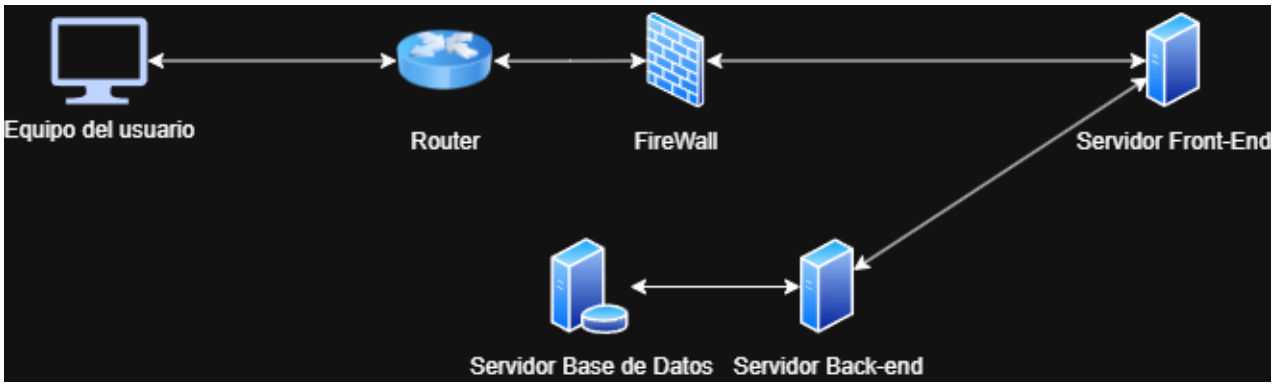


Fuente: https://despliegue.abrilcode.com/lib/exe/fetch.php?media=bloque1:3_capas.png

La presente imagen detalla el flujo de interacción dentro de la plataforma. El proceso inicia en el navegador del usuario, el cual actúa como la interfaz principal para consumir la lógica de negocio. A su vez, el front-end gestiona las peticiones del cliente coordinando la comunicación con el back-end, donde reside la capa de servicios. Finalmente, el back-end actúa como intermediario para realizar consultas y persistir información en el servidor de base de datos, completando así el ciclo de transferencia de datos y asegurando la integridad de la operación de extremo a extremo. Sobre la base de lo expuesto, podemos continuar con la definición de la arquitectura propia del sistema a desarrollar, el cual se muestra en la siguiente figura:

Ilustración 25

Arquitectura a implementar



Fuente: Elaboración propia

Habiendo definido la arquitectura se debe recapitular los costos de licenciamiento y uso de algunas plataformas de desarrollo, esto se hace referencia en la Tabla 1 Costes de licenciamiento, y a continuación se desarrollará las herramientas a utilizar para el desarrollo explayando el motor de base de datos, IDE´s y tecnologías a usar para el desarrollo.

Motor de base de datos

Para la gestión de la base de datos se empleará el motor SQL Server. En el ambiente de desarrollo se utilizará la versión 15.0.2170, mientras que para el ambiente productivo se contará con la versión 16.0.4252. Si bien existe una diferencia de versión entre ambos entornos, las características utilizadas por la aplicación son compatibles entre ambas, por lo que no se anticipan inconvenientes de portabilidad. No obstante, se recomienda alinear ambas versiones en el mediano plazo para garantizar la paridad entre los entornos de desarrollo y producción, reduciendo así el riesgo de comportamientos divergentes.

Como herramienta de administración y consulta se utilizará SQL Server Management Studio (SSMS) en su versión 20.6, la cual se empleará de manera transversal tanto en el entorno de desarrollo como en el productivo.

IDE´s

Para el desarrollo de la solución tecnológica se utilizarán dos entornos de desarrollo integrados (IDE) principales, seleccionados según las necesidades específicas de cada componente de la aplicación. Para la construcción del REST API en C#, se hará uso de Visual Studio Professional, debido a que proporciona un ecosistema robusto para el desarrollo, integración nativa con proyectos .NET, herramientas avanzadas de depuración y generación de plantillas preconfiguradas que agilizan el ciclo de desarrollo y garantizan mejores prácticas de arquitectura y mantenibilidad del código.

Por otra parte, para el desarrollo de la capa Front-End se utilizará Visual Studio Code, herramienta ligera y altamente extensible que ofrece compatibilidad eficiente con tecnologías modernas basadas en JavaScript y React. Su soporte avanzado para JSX, integración con extensiones especializadas, control de versiones mediante Git nos permite un desarrollo dinámico y de fácil mantenimiento, esta combinación de herramientas permite establecer un entorno de trabajo eficiente, modular y alineado con estándares modernos de desarrollo de software.

Tecnologías

Como se mencionó anteriormente el proyecto constara de dos partes los cuales son el front-end y back-end, para el front-end se hará uso de React en su versión 19.2.5, así como una serie de dependencias fundamentales para el proyecto como se muestra en la siguiente tabla:

Tabla 7

Dependencias Front-end

Dependencia	Versión	Función
Axios	1.15.2	Conexión al back-end por medio de HTTP.
Datatables.net-react	1.0.2	Diseño de tablas dinámicas con funciones predeterminadas.
I18next	26.0.10	Soporte de múltiples idiomas.

Dependencia	Versión	Función
React-jwt	2.1.1	Soporte para la manipulación y validación de JWT token.
React-router-dom	7.14.2	Enrutado del sitio para URL's del sistema.

Fuente: Elaboración propia

La tabla anterior representa algunas de las dependencias esenciales que conforman la capa de presentación del proyecto, es decir, el front-end de la aplicación. Una vez definidos estos componentes y establecida su función dentro del prototipo, podemos avanzar hacia la especificación de las dependencias correspondientes a la capa de back-end, las cuales se detallan a continuación:

Tabla 8

Dependencias Back-End

Dependencia	Versión
.Net Framework	4.7.2
Newtonsoft.Json	13.04
Microsoft.Owin	4.2.3
IdentityModel.Token.Jwt	8.18.0
Microsoft.AspNet.Cors	5.3.0

Fuente: Elaboración propia

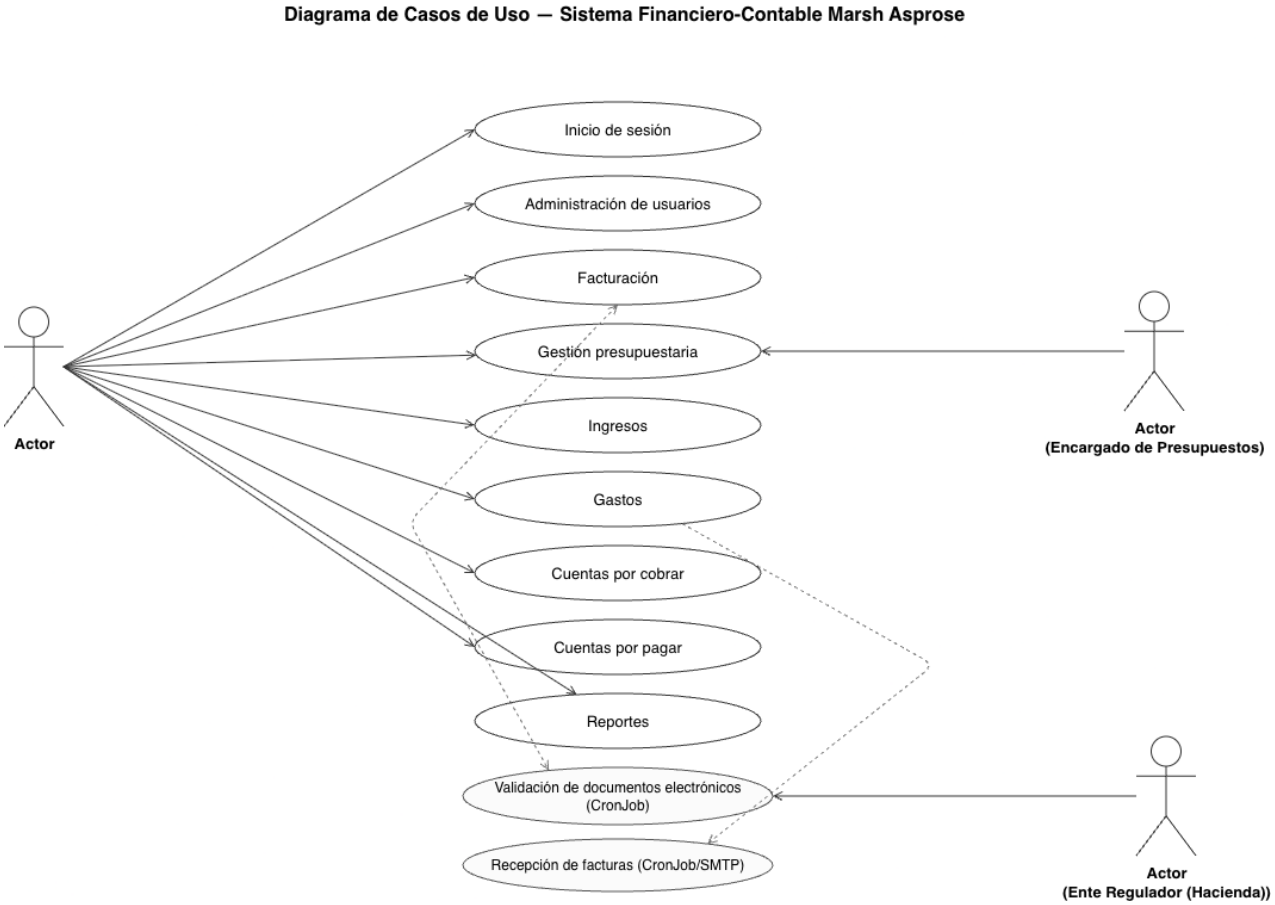
Estas dependencias constituyen la base sobre la cual se sustenta la ejecución del proyecto en C#, ya que proporcionan los componentes y configuraciones esenciales para el correcto funcionamiento de la aplicación. Entre sus responsabilidades principales se encuentra la gestión de JSON Web Tokens (JWT), mecanismo que habilita un esquema de autenticación y autorización basado en tokens, garantizando que únicamente los usuarios con credenciales válidas puedan acceder a los recursos protegidos del sistema. Asimismo, incorporan la configuración de CORS (Cross-Origin Resource Sharing), la cual define las políticas de acceso entre dominios y permite que el front-end consuma de forma segura y controlada los servicios expuestos por la API.

CASOS DE USO

De acuerdo con Wrike Team (2025), los casos de uso se pueden definir como “un concepto utilizado en el desarrollo de software, el diseño de productos y otros ámbitos para describir la utilidad de un sistema para realizar tareas concretas o conseguir objetivos específicos” (párr. 1). Esto nos indica que un caso de uso será una herramienta la cual nos permitirá describir las funciones del sistema, incluyendo un actor o usuario involucrado en como este se desempeñará dentro del sistema, a continuación, se detallan los casos de uso para el sistema web de gestión financiera de Marsh Asprose:

Ilustración 26

Diagrama de Casos de Uso



Fuente: Elaboración propia

Con base en la figura anterior, a continuación se documentan en detalle los casos de uso del sistema, describiendo su funcionalidad operativa y justificando cada conexión entre actores y acciones. Para cada caso de uso se especifican las precondiciones, el flujo básico, los subflujos asociados a las operaciones disponibles y los flujos alternos que atienden validaciones fallidas o escenarios excepcionales.

Tabla 9

Caso de uso 1 - Inicio de sesión

Sistema web para la gestión financiero contable para Marsh Asprose.	
Número de Caso de Uso: 1	**Nombre de Caso de Uso: Inicio de Sesión**
Fecha de Elaboración:	10/4/2026
Descripción del Caso de Uso:	Flujo y comportamiento sobre un intento de inicio de sesión
Autor Caso de Uso:	Douglas Gonzalez Castro
Actores relacionados:	Usuario
Precondiciones:	Las personas deben de contar con credenciales de acceso preestablecidos por el Departamento de TI
Flujo Básico del caso de uso	
1. El Usuario ingresa a la URL del sistema 2. El Usuario digita las credenciales (correo y contraseña) 3. El Usuario da clic al botón de acceder/ingresar 4. En caso de que la autenticación sea exitosa el sistema generará un token JWT e iniciara una sesión de una hora para ese usuario 5. En caso de que la autenticación no sea exitosa se continuara en Sub-Flujo 1 6. En caso de que el usuario no exista se continuara en Sub-flujo 2 7. El Sistema muestra la ventana de alerta o de inicio de sesión exitoso 8. A dos minutos de que expire la sesión de una hora se mostrará una alerta al usuario indicando que su sesión está pronta a expirar, si el usuario cancela la sesión se cerrará sin embargo si desea extenderla se seguirá en FA-4	
Sub Flujos	
Sub-Flujo 1: máximo de intentos de inicio de sesión	Si el usuario o contraseña son incorrectos el sistema devolverá un mensaje que alguno de estos campos es inválidos e iniciará el contador de intentos fallidos, si se repite este proceso un total de tres veces se continuará en Flujo – alterno 1
Sub-Flujo 2: Usuario inexistente	Si el usuario ingresa correo y contraseña y este usuario no existe en la base de datos se mostrará un mensaje de que el usuario no ha sido creado y debe solicitar un acceso como lo dicta la precondición.
Flujos Alternos	
Flujo Alternativo 1: Usuario bloqueado	En caso de que el usuario falle su autenticación en tres ocasiones diferentes el sistema bloqueara el acceso en un intervalo de tiempo de diez minutos y mostrara un mensaje de que es inaccesible durante ese periodo de tiempo.
Flujo Alternativo 2: Usuario inexistente	El usuario recibe una alerta que indica que las credenciales son erróneas, por favor inténtelo de nuevo o comuníquese con el departamento de TI
Flujo Alternativo 3: Usuario refresca sesión	A dos minutos de que expire la sesión de una hora se mostrará una alerta al usuario indicando que su sesión esta pronta a expirar, si

	desea salir del sistema o extender su sesión, esto generará un nuevo token
Flujo Alternativo 4: Extender sesión	Si el usuario extiende su sesión, el sistema generará un nuevo token de sesión con una duración de una hora.
Requerimientos especiales	
Acceso a internet	
Contar con credenciales del sistema	
Post-Condiciones	
Usuario ingresa a una sesión activa	

Fuente: Elaboración propia

Tabla 10

Caso de uso 2 Administración de usuarios

Sistema web para la gestión financiero contable para Marsh Asprose.	
Número de Caso de Uso: 2	**Nombre de Caso de Uso: Administración de usuarios**
Fecha de Elaboración:	10/4/2026
Descripción del Caso de Uso:	Comportamiento de la administración de usuarios internos del sistema
Autor Caso de Uso:	Douglas Gonzalez Castro
Actores relacionados:	Usuarios del sistema
Precondiciones:	1. El Usuario ingresa al sistema 2. El Usuario inicia sesión 3. El usuario debe tener permisos para administrar usuarios
Flujo Básico del caso de uso	
1. El usuario selecciona la opción de Usuarios 2. El sistema muestra la lista de los usuarios con las opciones de consultar, modifica, ingresar. 3. El usuario selecciona las opciones: - Consultar (SF-01) - Ingresar (SF-02) - Modificar (SF-03)	
Sub-Flujos	
SF-01 Consultar	1. El sistema carga un formulario donde se mostrarán los datos del usuario. 2. El sistema habilita un campo de texto para el ingreso de la identificación del usuario a consultar. Junto con los botones Consultar, Borrar y Cancelar. 3. El usuario ingresa los datos del usuario en el campo de texto habilitado.

	4. En caso de que la información no sea válida se mostrara una alerta de que la información de usuario no es valida 5. Si la información es válida se buscara la información del usuario en la base de datos y se habilitará estas tres opciones. - Consultar. (FA-01) - Borrar. (FA-04) - Cancelar. (FA-03)
SF-02 Ingresar	1. El sistema muestra un formulario para el ingreso de los datos. Junto con los botones Confirmar, Borrar y Cancelar. 2. El usuario ingresa los datos en los campos de texto. 3. El sistema valida que la información del correo e ID no exista actualmente para otro usuario, en caso de existir el sistema muestra una alerta de que ese usuario ya existe en la base de datos. 4. El usuario selecciona una de las opciones. - Confirmar (FA-05) - Borrar (FA-04) - Cancelar (FA-03)
SF-03 Modificar	1. El sistema muestra un formulario donde se cargarán los datos a modificar. 2. El sistema habilita un campo de texto para el ingreso de la identificación del empleado a modificar. Junto con el botón Consultar. 3. El usuario ingresa los datos en el campo de texto habilitado. 4. El usuario selecciona el botón Consultar. (FA-02) (FA-01): 5. El sistema carga el formulario con los datos obtenidos. 6. El sistema habilita los botones Confirmar, Borrar y Cancelar. 7. El usuario modifica los datos del usuario que el sistema permite. 8. El usuario selecciona una de las opciones. - Confirmar (FA-05) - Borrar (FA-04) - Cancelar (FA-03)
Flujos Alternos	
FA-01: Validar usuario	1. El sistema valida que los datos ingresados sean correctos, comparándolos con los empleados existentes de la base de datos. 2. El sistema retorna los datos encontrados o el mensaje de error correspondiente.
FA-02: Validar campos	1. El sistema valida que todos los campos de texto obligatorios se encuentren con los datos correspondientes.
FA-03: Cancelar	El sistema cancela la operación actual.
FA-04: Limpiar datos	El sistema limpia todos los campos de texto.
FA-05: Confirmar	1. El sistema valida los tipos de datos. 2. Basado en el rol seleccionado se crean un perfil de asignación de permisos para los distintas opciones dentro del sistema.

	3. El sistema retorna la respuesta correspondiente al ingreso o actualización de los datos.
Requerimientos especiales	
Una vez creado el usuario la información de este es almacenada en un log, esto para llevar un control de auditoria **El sistema notifica por medio de correo al usuario que haya sido ingresado y en caso de que se haya modificado la contraseña del usuario, así como cualquier cambio de ROL**	
Post-Condiciones	

Fuente: Elaboración propia

Tabla 11

Caso de Uso 3 Facturación

Sistema web para la gestión financiero contable para Marsh Asprose.	
Número de Caso de Uso: 3	**Nombre de Caso de Uso: Facturación**
Fecha de Elaboración:	10/4/2026
Descripción del Caso de Uso:	Facturación
Autor Caso de Uso:	Douglas Gonzalez Castro
Actores relacionados:	Usuarios encargados de facturación
Precondiciones:	El usuario debe haber iniciado sesión en el sistema El usuario debe tener permisos para el módulo de facturación El usuario ingresa al módulo de facturación.
Flujo Básico del caso de uso	
1. El sistema muestra las siguientes opciones: Crear, Consultar, Descartar y Agregar 2. El usuario selecciona una de las opciones: - Crear (SF-01) - Consultar (SF-02) - Descartar (SF-03) - Agregar (SF-04) - Aceptación de Facturas (SF-05)	
Sub-Flujos	
SF-01 Crear	1. El sistema muestra un formulario para el ingreso de los datos. Junto con los botones Confirmar, Limpiar y Cancelar. 2. El usuario ingresa los datos en los campos de texto. 3. El sistema valida la información ingresada 4. El usuario selecciona una de las siguientes opciones. - Facturar. (FA-04) - Limpiar. (FA-03) - Cancelar. (FA-02)

SF-02 Consultar	1. El sistema muestra un formulario donde se cargarán los datos. 2. El sistema habilita un campo de texto para el ingreso de la identificación del cliente facturado a consultar. Junto con el botón Ver. 3. El usuario ingresa los datos en el campo de texto habilitado. 4. El usuario selecciona el botón Consultar. (FA-01): 5. El sistema carga el formulario con los datos obtenidos.
SF-03 Descartar	1. El sistema muestra un formulario donde se cargarán los datos de la factura a descartar 2. El usuario selecciona la factura a descartar 3. El usuario selecciona el botón Descartar y Confirma la acción 4. El usuario selecciona una de las opciones - Descartar (FA-06) - Cancelar (FA-02)
SF-04 Agregar	1. El sistema muestra un formulario donde se cargarán los datos de la factura a la cual se desee agregar información 2. El usuario ingresa la identificación de la factura a agregar 3. El usuario selecciona el botón de Agregar y confirma la acción 4. El sistema muestra un formulario para el ingreso de los datos. Junto con los botones Agregar información, Borrar y Cancelar. 5. El usuario ingresa los datos en los campos de texto. 6. El usuario selecciona una de las siguientes opciones. - Agregar Información. (FA-07) - Limpiar. (FA-03) - Cancelar. (FA-04)
Flujos Alternos	
FA-01: Consultar	1. El sistema valida que los datos ingresados sean correctos, comparándolos con los empleados existentes de la base de datos. 2. El sistema retorna los datos encontrados o el mensaje correspondiente.
FA-02: Limpiar	El sistema limpia todos los campos de texto.
FA-03: Cancelar	El sistema cancela la operación actual.
FA-04: Factura (detalle)	1. El sistema genera un documento electrónico (XML) con base a la información de la factura y detalle de la factura 2. El sistema valida el documento electrónico ante el ente regulatorio 3. En caso de no ser posible validar el documento electrónico el sistema almacenará el documento con el estado

	"pendiente de validar" para intentar él envió nuevamente mediante un CronJob (FA-07) 4. Si el sistema recibe una respuesta el documento electrónico queda con el estado "pendiente confirmación" para que un CronJob valide con el ente regulador la respuesta del mismo (FA-07). 5. Una vez confirmada la respuesta del ente regulador se asignará al documento electrónico la respuesta dada por el ente regulador la cual seria "aprobado" o "rechazado"
FA-05: Descartar (Nota Crédito)	1. El sistema obtiene el documento electrónico de referencia y valida que esta se encuentre en estado "aprobado por el ente regulador" de lo contrario mostrará una alerta indicando que ese documento no puede ser descartado. 2. El sistema genera un documento electrónico (XML) de descarte (Nota de crédito) 3. El sistema valida el documento ante el ente regulador 4. En caso de no poder validar en el momento el documento se almacenará con un estado "pendiente de validar" para intentar él envió nuevamente mediante un CronJob (FA-07) 5. Una vez confirmada la respuesta del ente regulador se asignará al documento electrónico la respuesta dada por el ente regulador la cual seria "aprobado" o "rechazado"
FA-06: Agregar Información (Nota Débito)	1. El sistema obtiene el documento electrónico de referencia (Factura electrónica). 2. El sistema genera un documento electrónico (XML) con base a la información suministrada (Nota de débito). 3. El sistema valida el documento ante el ente regulador. 4. En caso de no poder validar en el momento el documento se almacenará con un estado "pendiente de validar" para intentar él envió nuevamente mediante un CronJob (FA-07) 5. Una vez confirmada la respuesta del ente regulador se asignará al documento electrónico la respuesta dada por el ente regulador la cual sería "aprobado" o "rechazado".
FA-07: Validación de documentos electrónicos (CronJob)	1. El sistema cada hora validará los documentos electrónicos que estén en estado "pendiente de validar" o "pendiente confirmación", tomando la información de la factura, generando un nuevo token ante el API del ente regulador y validando la información 2. En caso de no obtener información o un error en cualquier consulta se guardará un log de auditoría sobre la tarea, fecha y documentos los cuales no se pudieron validar además de la razón del error y código del error.

	3. Si se obtiene respuesta se guardará en base de datos el XML de respuesta del ente regulador y el estado para que no sea tomado en cuenta en otra corrida del job.
Requerimientos especiales	
Para la estructura de Factura Electrónica, Nota de crédito y nota de débito se seguirá el patrón de XML para la versión 4.4 definido por Hacienda https://www.hacienda.go.cr/docs/Anexosyestructuras.pdf	
Post-Condiciones	

Fuente: Elaboración propia

Tabla 12

Caso de Uso 4 Gestión Presupuestaria

<table>
<tr><td colspan="2">Sistema web para la gestión financiero contable para Marsh Asprose.</td></tr>
<tr><td>Número de Caso de Uso: 4</td><td>Nombre de Caso de Uso: Gestión Presupuestario</td></tr>
<tr><td>Fecha de Elaboración:</td><td>23/4/2026</td></tr>
<tr><td>Descripción del Caso de Uso:</td><td>Administración del ciclo de vida de la gestión presupuestaria</td></tr>
<tr><td>Autor Caso de Uso:</td><td>Douglas Gonzalez Castro</td></tr>
<tr><td>Actores relacionados:</td><td>Usuarios del sistema</td></tr>
<tr><td>Precondiciones:</td><td>El usuario debe haber iniciado sesión en el sistema
El usuario debe tener permisos para gestión presupuestaria
Se debió haber definido los centros de costos</td></tr>
<tr><td colspan="2">Flujo Básico del caso de uso</td></tr>
<tr><td colspan="2">1. El usuario selecciona la opción de “Gestión Presupuestaria”
2. El sistema muestra la lista de opciones para la gestión presupuestaria
3. El usuario selecciona las opciones:
- Consultar (SF-01)
- Ingresar (SF-02)
- Modificar (SF-03)
- Eliminar (SF-04)</td></tr>
<tr><td colspan="2">Sub-Flujos</td></tr>
<tr><td>SF-01 Consultar</td><td>1. El sistema muestra una lista de todas las gestiones presupuestarias
2. El sistema habilita un campo de texto para el ingreso de la identificación de la gestión a consultar. Junto con los botones Consultar, Borrar y Cancelar.
3. El usuario ingresa los datos en el campo de texto habilitado y da clic en el botón Consultar.
4. El sistema valida la información ingresada, en caso de que la información sea incorrecta se mostrará un mensaje de advertencia indicando que la información ingresada no es válida. (FA-01).
5. El sistema muestra la información consultada, en caso de no encontrar un registro mostrará una alerta (FA-02).
6. El usuario puede seleccionar alguno de las siguientes opciones:
- Limpiar. (FA-04)
- Cancelar. (FA-03)</td></tr>
<tr><td>SF-02 Ingresar</td><td>1. El sistema muestra un formulario para el ingreso de los datos. Junto con los botones Guardar, Borrar y Cancelar.</td></tr>
</table>

	2. El usuario ingresa los datos en los campos de texto y da clic en el botón Guardar. 3. El sistema valida los tipos de datos ingresados y si la información no es válida muestra una alerta (FA-01). 4. Si la información es correcta se registra la gestión presupuestaria para el periodo de tiempo indicado, siendo este afectado a posterior por ingresos o gastos. 5. Una vez guardado el presupuesto inicial el sistema mostrará una ventana en donde se permitirá digitar las estimaciones de gastos e ingresos relacionados a los centros de costos. 6. Cualquier transacción para este registro será notificada (FA-07)
SF-03 Modificar	1. El sistema muestra un formulario donde se cargarán los datos de la gestión. 2. El usuario modifica los datos y da clic en el botón Actualizar (FA-06). 3. En caso de que haya algún error con los datos ingresados se mostrará una alerta indicando cual es campo que debe ser corregido 4. El sistema permitirá modificar el mismo presupuesto un máximo de cinco veces. 5. Si el proceso es exitoso el sistema deberá validar si el presupuesto cuenta con una o más transacciones aplicadas (gastos, ingresos) para hacer un recalculo, en caso de haberse modificado montos. 6. Independiente de si el presupuesto inicial del sistema es modificado o no el sistema mostrará una ventana en donde se permitirá ajustar las estimaciones de gastos e ingresos relacionados a los centros de costos. 7. Una vez hecho algún ajuste sobre cualquier presupuesto de centro de costo que tenga un monto mayor a cinco millones se notificará al encargado de presupuestos para que de su aprobación o rechazo del movimiento.
SF-03 Eliminar	1. El sistema solicita confirmar la acción 2. El sistema elimina la gestión presupuestaria. 3. Si hubiese un error con el proceso de eliminación se mostrará una alerta bloqueante indicando que la acción no puede ser realizada en ese momento. 4. En caso de que la gestión esté relacionada en proceso, y tenga relacionado una o más transacciones como ingresos o cobros esta se desactivará
Flujos Alternos	
FA-01: Información invalida	1. El sistema muestra un mensaje de alerta de advertencia indicando que el tipo de dato ingresado no es permitido 2. El sistema limpia los campos de texto
FA-02: Información no encontrada	1. El sistema muestra una alerta informativa indicando que para los filtros aplicados no se encontraron registro

FA-03: Cancelar	1. El sistema cancela la operación actual.
FA-04: Limpiar	1. El sistema limpia todos los campos de texto.
FA-05: Guardar	1. El sistema valida los tipos de datos ingresados 2. El sistema almacena los datos en la base de datos. 3. El sistema retorna la respuesta correspondiente al ingreso de los datos.
FA-06: Actualizar	1. El sistema valida los tipos de datos ingresados, en caso de haber un error con algún tipo de dato se devolverá una alerta acerca de que campo debe ser corregido. 2. El sistema evalúa si el presupuesto posee transacciones en caso de verifica la información de los ingresos y gastos aplicados con el fin de que el registro quede actualizado con la información nueva más la información de amortizaciones ya aplicadas. 3. El sistema almacena los datos en la base de datos y al ser un proceso fundamental almacena información de la acción en un log para auditoria 4. El sistema retorna la respuesta correspondiente a la actualización de los datos.
FA-07: Alerta de transacción	1. Cualquier transacción creada en un presupuesto grabado en el sistema será notificada al grupo de personas encargadas de presupuestos para su aprobación o descarte.
Requerimientos especiales	
Post-Condiciones	

Fuente: Elaboración propia

Tabla 13

Caso de Uso 5 Ingresos

Sistema web para la gestión financiero contable para Marsh Asprose.	
Número de Caso de Uso: 5	**Nombre de Caso de Uso: Ingresos**
Fecha de Elaboración:	23/4/2026
Descripción del Caso de Uso:	Administración del ciclo de vida de los Ingresos
Autor Caso de Uso:	Douglas Gonzalez Castro
Actores relacionados:	Usuarios del sistema
Precondiciones:	El usuario debe haber iniciado sesión en el sistema El usuario debe tener permisos para administrar ingresos
Flujo Básico del caso de uso	
1. El usuario selecciona la opción de "Ingresos" 2. El sistema muestra la lista de opciones para los ingresos 3. El usuario selecciona de las siguientes opciones: - Consultar (SF-01) - Registrar Ingreso (SF-02) - Modificar (SF-03) - Eliminar (SF-04)	
Sub-Flujos	
SF-01 Consultar	1. El sistema habilita un campo de texto para escribir la referencia del ingreso a consultar. Junto con los botones Consultar, Borrar y Cancelar. 2. El usuario ingresa los datos de los filtros en el campo de texto habilitado. 3. El sistema valida la información ingresada, en caso de no ser válida muestra un mensaje de advertencia y limpia los campos de texto (FA-04). 4. El usuario da clic en el botón consultar y esta muestra el contenido del detalle del ingreso. 5. El usuario selecciona una de las dos opciones. - Limpiar. (FA-03) - Cancelar. (FA-02)
SF-02 Registrar Ingreso.	1. El sistema muestra un formulario para la digitación de los datos. Junto con los botones Guardar, Borrar y Cancelar. 2. El usuario ingresa los datos en los campos de texto y da clic en el botón Guardar. 3. El sistema valida la información ingresada, en caso de encontrar algún error con los tipos de datos el sistema muestra una alerta de advertencia indicando que uno o más campos tienen información incorrecta.

	4. Si la validación es correcta el sistema procesa el ingreso (FA-05) 5. El sistema redirecciona al usuario a la consulta del ingreso recién creado.
SF-03 Modificar	1. El sistema muestra un formulario donde se cargarán los datos del registro de ingreso. 2. El usuario modifica los datos de los campos que el sistema le habilita. 3. El sistema valida los tipos de datos ingresados por el usuario. 4. El usuario da clic en el botón Actualizar iniciando el proceso de actualización (FA-06) 5. El sistema redirecciona al usuario a la consulta del ingreso actualizado.
SF-03 Eliminar	1. El sistema solicita confirmar la acción. 2. Si el ingreso no está relacionado a una factura, el sistema desactiva el ingreso y reversa los cambios que este hizo en la tabla de Presupuestos 3. En caso de que el ingreso posea una factura y documento electrónico se generara un descarte (nota de crédito) para este ingreso y ajustara los cambios aplicados a la tabla de presupuesto. 4. En caso de haber un error en alguno de estos pasos el sistema mostrará al usuario una alerta indicando en el paso que sucedió y los pasos a seguir para corregirlo.
Flujos Alternos	
FA-01: Consultar	1. El sistema valida la información del ingreso seleccionado 2. El sistema retorna los datos encontrados o el mensaje de error correspondiente.
FA-02: Cancelar	1. El sistema cancela la operación actual.
FA-04: Limpiar	1. El sistema limpia todos los campos de texto.
FA-05: Validar creación de ingreso	1. El sistema valida la información de los datos ingresados. 2. El sistema valida que el ingreso haya sido asignado a un presupuesto (Saldo Inicial) activo y valido, de lo contrario mostrará un mensaje de error indicando que no se puede crear el ingreso ya que no está asignado a un presupuesto valido 3. En caso de que el presupuesto sea válido el sistema ajusta el presupuesto en base al monto del ingreso registrado, para controlar ganancias y costos de utilidad. 4. El sistema notifica al grupo de personas encargadas del presupuesto. 5. El sistema almacena la información del ingreso 6. En caso de fallar en algún proceso se mostrará un error indicando en que paso fallo y que proceso debe hacer para corregirlo. 7. El sistema consulta si desea de ese ingreso generar una factura electrónica de compra o un recibo de dinero.

FA-06: Actualizar	1. El sistema valida los tipos de datos ingresados 2. El sistema verifica las diferencias entre la información actualizada contra la información ya guardada para hacer ajustes en la tabla de presupuestos basados en utilidades y ganancias. 3. En caso de existir una factura el sistema generará un ajuste a esta tabla de facturas. 4. En caso de que se exista una factura electrónica de compra el sistema creará un ajuste por disminución o aumento (nota crédito o débito) 5. El sistema almacena los datos en la base de datos. 6. El sistema retorna la respuesta correspondiente a la actualización de los datos. 7. En caso de haber un error en alguno de estos pasos el sistema mostrará una alerta indicándole al usuario en que paso fallo y que debe hacer para solucionarlo.
Requerimientos especiales	
Post-Condiciones	

Fuente: Elaboración propia

Tabla 14

Caso de Uso 6 Gastos

Sistema web para la gestión financiero contable para Marsh Asprose.	
Número de Caso de Uso: 6	**Nombre de Caso de Uso: Gastos**
Fecha de Elaboración:	23/4/2026
Descripción del Caso de Uso:	Administración del ciclo de vida de los gastos
Autor Caso de Uso:	Douglas Gonzalez Castro
Actores relacionados:	Usuarios del sistema
Precondiciones:	El usuario debe haber iniciado sesión en el sistema El usuario debe tener permisos para administrar gastos Tener configurado un presupuesto Tener configurado una cuenta SMTP para recibir correos.
Flujo Básico del caso de uso	
1. El usuario selecciona la opción de “Gastos” 2. El sistema muestra la lista de opciones para los gastos 3. El usuario selecciona las opciones: - Consultar (SF-01) - Ingresar (SF-02) - Modificar (SF-03)	

- Eliminar (SF-04)	
Sub-Flujos	
SF-01 Consultar	1. El sistema muestra una lista de todos los gastos 2. El sistema habilita un campo de texto para el ingreso de la identificación del gasto a consultar. 3. El usuario ingresa los datos en el campo de texto habilitado. 4. El sistema valida los tipos de datos ingresados al sistema, en caso de no ser información valida mostrará un mensaje de alerta. 5. En caso de no encontrar registros el sistema mostrará un mensaje informativo de que no encontró el registro del gasto consultado. 6. El usuario selecciona una de las tres opciones. - Limpiar. (FA-03) - Cancelar. (FA-02)
SF-02 Ingresar	1. El sistema muestra un formulario para el ingreso de los datos. 2. El usuario ingresa los datos en los campos de texto y da clic en el botón Guardar. 3. El sistema valida los tipos de datos ingresados en el formulario, en caso de un error muestra un mensaje de advertencia indicando cuales campos deben ser corregidos. 4. En caso de ser satisfactorio el sistema procesa la información (FA-05) 5. El usuario selecciona una de las tres opciones. - Limpiar. (FA-04) - Cancelar. (FA-02)
SF-03 Modificar	1. El sistema muestra un formulario donde se cargarán los datos del gasto. 2. El usuario ingresa los datos en los campos de texto habilitados. 3. El sistema habilita los botones Actualizar, Borrar y Cancelar. 4. El usuario edita o ingresa nuevos datos. 5. El usuario selecciona una de las opciones. - Actualizar (FA-06) - Limpiar (FA-03) - Cancelar (FA-02)
SF-03 Eliminar	1. El sistema muestra un formulario para el ingreso de los datos de identificación del ingreso. 2. El sistema solicita confirmar la acción. 3. El sistema elimina el registro del gasto. 4. En caso de que esté gasto se encuentre relacionada a otro proceso esta se desactiva.

SF-04: Aceptación de Facturas	1. El usuario ingresa a la opción de Aceptación de facturas 2. El sistema muestra las facturas pendientes a aceptación obtenidas por el CronJob de Aceptación (FA-08) 3. El usuario selecciona las facturas que desea aceptar y da clic en el botón Aceptar (FA-07) 4. En caso de que el usuario desee rechazar la factura deberá seleccionar las facturas recibidas y dar clic en el botón Rechazar (FA-07).
Flujos Alternos	
FA-01: Consultar	1. El sistema valida la información del ingreso seleccionado. 2. El sistema retorna los datos encontrados o el mensaje de error correspondiente.
FA-02: Cancelar	1. El sistema cancela la operación actual.
FA-04: Limpiar	1. El sistema limpia todos los campos de texto.
FA-05: Guardar	1. El sistema valida que los tipos de datos sean válidos, en caso de no serlo el sistema mostrará una alerta de advertencia indicando los valores que deben ser corregidos. 2. En caso de que los datos sean correctos el sistema almacena el registro del gasto y su detalle, y aplica el gasto a la tabla de presupuesto designado para el periodo hábil. 3. El sistema notifica a las personas encargadas del presupuesto. 4. El sistema consulta si desea crear un documento electrónico (Factura electrónica de compra), esto sucede si el gasto registrado no posee una referencia a un documento electrónico. 5. En caso de que el usuario solicite crear una Factura electrónica de compra, el sistema generara un documento el cual validará con el ente regulador, este documento será validado en caso de no obtener una respuesta, por el CronJob de validación de facturas hasta obtener una respuesta. 6. El sistema almacena los datos en la base de datos. 7. El sistema retorna la respuesta correspondiente al ingreso de los datos.
FA-06: Actualizar	1. El sistema valida los tipos de datos ingresados 2. En caso de que los datos sean incorrectos el sistema mostrará una alerta indicando cuales datos deben ser corregidos. 3. En caso de que la información sea válida el sistema guardara la información en la base de datos. 4. El sistema modifica la información del presupuesto al cual este asignado para ajustar los montos del presupuesto ya sea de aumento o disminución.

	5. En caso de que el ingreso tenga una factura de compra el sistema generara una nota de crédito o débito para dicho gasto y factura de compra para equiparar la información. 6. En caso de que haya un error en alguno de estos procesos el sistema muestra un mensaje de error indicando en que paso sucedió e indicando los pasos a seguir para corregirlo. 7. El sistema retorna la respuesta correspondiente a la actualización de los datos.
FA-07: Aceptación/Rechazo de Factura	1. El sistema toma las claves numéricas y documento electrónico de referencia de la factura recibida. 2. El sistema genera un token en el API del ente regulador. 3. El sistema envía al ente regulador el mensaje de que acepta la factura emitida por el proveedor.
FA-08: CronJob obtener Facturas	1. El sistema consigue la configuración de acceso a una cuenta SMTP donde se reciben los correos de facturas. 2. El sistema obtiene los correos con adjuntos de facturas emitidos hacia la empresa. 3. El sistema almacena los documentos electrónicos (PDF, XML, XML Respuesta). 4. En caso de un error con la conexión el sistema notifica por medio de correo al equipo de administradores/TI para la corrección de los datos de la cuenta SMTP.
Requerimientos especiales	
Para la estructura de Factura Electrónica de compra se seguirá el patrón de XML para la versión 4.4 definido por Hacienda https://www.hacienda.go.cr/docs/Anexosyestructuras.pdf	
Post-Condiciones	

Fuente: Elaboración propia

Tabla 15

Caso de Uso 7 Cuentas por Cobrar

Sistema web para la gestión financiero contable para Marsh Asprose,	
Número de Caso de Uso: 7	**Nombre de Caso de Uso: Cuentas por cobrar**
Fecha de Elaboración:	23/4/2026
Descripción del Caso de Uso:	Administración del ciclo de vida de las cuentas por cobrar
Autor Caso de Uso:	Douglas Gonzalez Castro
Actores relacionados:	Usuarios del sistema
Precondiciones:	El usuario debe haber iniciado sesión en el sistema El usuario debe tener permisos para administrar cuentas por cobrar
Flujo Básico del caso de uso	

1. El usuario selecciona la opción de “Cuentas por Cobrar” 2. El usuario selecciona las opciones: - Consultar CXC(SF-01)	
Sub-Flujos	
SF-01 Consultar CXC	1. El sistema muestra una lista de todas las cuentas por cobrar 2. El sistema habilita un campo de texto para el ingreso de la identificación del cliente a consultar. 3. El usuario ingresa los datos en el campo de texto habilitado y da clic en el botón Consultar. 4. En caso de que la información ingresada sea incorrecta el sistema muestra un mensaje de alerta indicando que la información no es válida y limpiando los campos de texto (FA-03). 5. Si la información es válida el sistema mostrará un resumen de la cuenta por cobrar indicando (FA-01).
Flujos Alternos	
FA-01: Consultar	1. El sistema valida toma la información de las facturas que apliquen para los filtros aplicados. 2. El sistema crea un balance de las facturas ingresadas para los clientes que cumplen los criterios del filtro. 3. El sistema crea una tabla auxiliar para ponderar el crédito en un lapso de 0 a 120 días separado por un intervalo de 30 días dependiendo del crédito de la factura/cliente. 4. El sistema retorna los datos encontrados o el mensaje de error correspondiente.
FA-02: Cancelar	1. El sistema cancela la operación actual.
FA-04: Limpiar	1. El sistema limpia todos los campos de texto.
Requerimientos especiales	
Post-Condiciones	

Fuente: Elaboración propia

Ilustración 27

Casos de Uso 8 Cuentas por Pagar

Sistema web para la gestión financiero contable para Marsh Asprose,	
Número de Caso de Uso: 8	**Nombre de Caso de Uso: Cuentas por cobrar**
Fecha de Elaboración:	23/4/2026
Descripción del Caso de Uso:	Administración del ciclo de vida de las cuentas por pagar
Autor Caso de Uso:	Douglas Gonzalez Castro

Actores relacionados:	Usuarios del sistema
Precondiciones:	El usuario debe haber iniciado sesión en el sistema El usuario debe tener permisos para administrar cuentas por pagar
Flujo Básico del caso de uso	
3. El usuario selecciona la opción de "Cuentas por Cobrar" 4. El usuario selecciona las opciones: - Consultar CXP(SF-01)	
Sub-Flujos	
SF-01 Consultar CXP	6. El sistema muestra una lista de todas las cuentas por pagar 7. El sistema habilita un campo de texto para el ingreso de la identificación del proveedor u otra información a consultar. 8. El usuario ingresa los datos en el campo de texto habilitado y da clic en el botón "Consultar". 9. En caso de que la información ingresada sea incorrecta el sistema muestra un mensaje de alerta indicando que la información no es válida y limpiando los campos de texto (FA-03). 10. Si la información es válida el sistema mostrará un resumen de la cuenta por pagar indicando (FA-01).
Flujos Alternos	
FA-01: Consultar	5. El sistema valida toma la información de los gastos, facturas de compra y facturas aceptadas que apliquen para los filtros aplicados. 6. El sistema crea un balance de los gastos por proveedor ingresadas para los clientes que cumplen los criterios del filtro. 7. El sistema crea una tabla auxiliar para ponderar el crédito dado por los proveedores para la empresa en un lapso de 0 a 120 días separado por un intervalo de 30 días dependiendo del crédito de la factura de compra o aceptada/proveedor. 8. El sistema retorna los datos encontrados o el mensaje de error correspondiente.
FA-02: Cancelar	2. El sistema cancela la operación actual.
FA-04: Limpiar	2. El sistema limpia todos los campos de texto.
Requerimientos especiales	
Post-Condiciones	

Fuente: Elaboración propia

Tabla 16

Caso de Uso 8 Reportes

Sistema web para la gestión financiero contable para Marsh Asprose,	
Número de Caso de Uso: 8	**Nombre de Caso de Uso: Cuentas por pagar y cuentas por cobrar**
Fecha de Elaboración:	27/4/2026
Descripción del Caso de Uso:	Administración de la generación de reportes
Autor Caso de Uso:	Douglas Gonzalez Castro
Actores relacionados:	Usuarios del sistema
Precondiciones:	El usuario debe haber iniciado sesión en el sistema
Flujo Básico del caso de uso	
1. El usuario inicia sesión. 2. Accede al módulo de reportes. 3. Selecciona el tipo de reporte. 4. Aplica filtros (colaborador, fecha, tipo, estado). 5. El sistema genera una vista previa del reporte. 6. El usuario selecciona si desea exportarlo (PDF o Excel) o imprimirlo. 7. El sistema genera el documento solicitado	
Sub-Flujos	
Flujos Alternos	
Requerimientos especiales	
Post-Condiciones	

Fuente: Elaboración propia

Tabla 17

Caso de Uso 9 Recuperar Contraseña

Sistema web para la gestión financiero contable para Marsh Asprose,	
Número de Caso de Uso: 9	**Nombre de Caso de Uso: Recuperar contraseña**
Fecha de Elaboración:	27/4/2026
Descripción del Caso de Uso:	Recuperación de contraseña
Autor Caso de Uso:	Douglas Gonzalez Castro
Actores relacionados:	Usuarios del sistema
Precondiciones:	El usuario debe poseer un usuario previamente creado en el sistema.
Flujo Básico del caso de uso	
1. El usuario da clic en la opción recuperar contraseña. 2. El usuario ingresa su correo. 3. Si el usuario posee un correo existente en base de datos el sistema envía un correo con un código de verificación, en caso contrario el sistema informará que el cliente no puede ser validado. 4. El usuario ingresa el código enviado al correo. 5. Si el código es válido se redirecciona la pantalla a una pantalla de restablecimiento de contraseña. 6. El usuario ingresa su nueva contraseña. 7. El sistema redirecciona al usuario al inicio de sesión.	
Sub-Flujos	
Flujos Alternos	
Requerimientos especiales	
Post-Condiciones	

Fuente: Elaboración propia

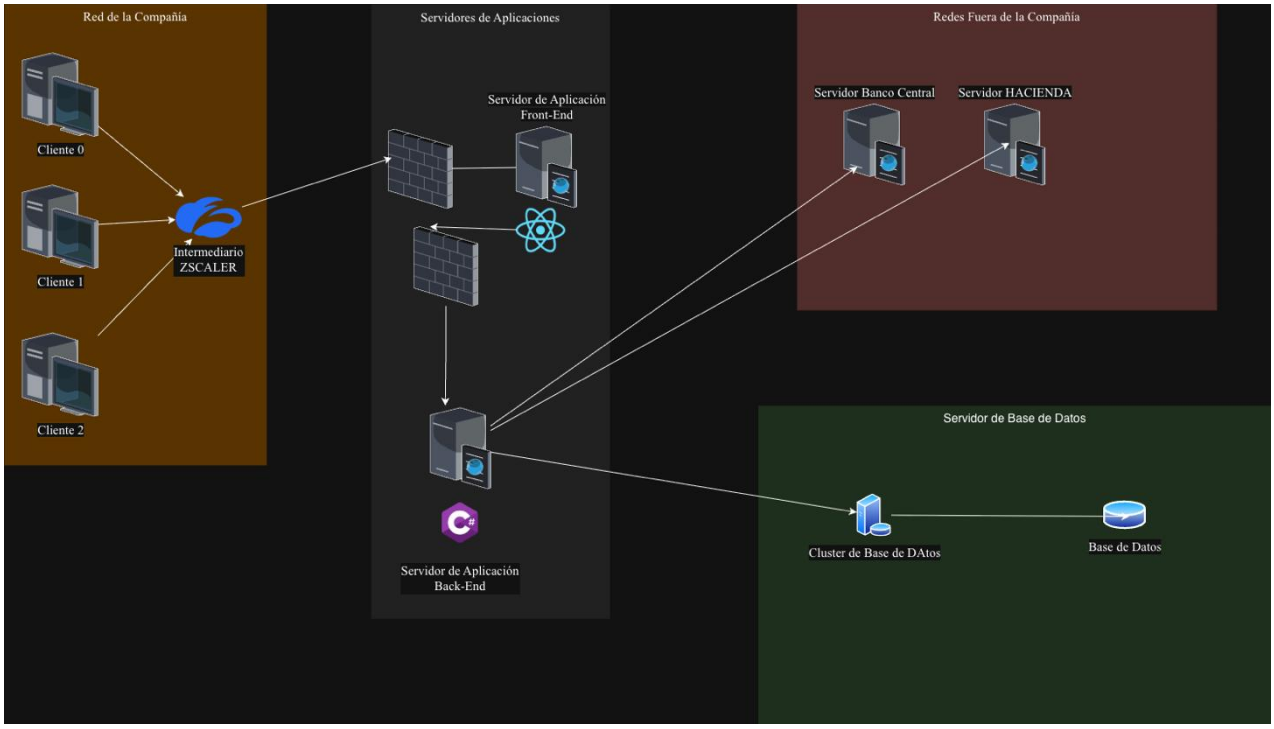
Diseño

Arquitectura del sistema

Como se ha señalado en apartados anteriores del presente documento, la arquitectura que se implementará corresponde al modelo cliente-servidor, la cual se desplegará sobre la infraestructura tecnológica existente de Marsh Asprose. Si bien dicha arquitectura ha sido descrita y representada previamente, a continuación, se presenta una representación simplificada que permite visualizar de manera más clara la disposición general de los componentes que conforman dicha infraestructura.

Ilustración 28

Diagrama Arquitectura del Sistema Diseño



Fuente: Elaboración propia.

Como se aprecia en la figura anterior, el acceso a la aplicación se realizará a través de la red interna corporativa de la organización. Dicha red contará con un intermediario de red (Zscaler) el cual se encargará de integrar cada maquina del usuario en una VPN de la empresa, lo cual nos permite movernos al cortafuegos (firewall) encargado de validar y controlar el tráfico hacia el servidor que aloja la capa de presentación (front-end). Esta capa, a su vez, establece la comunicación

con la capa de lógica de negocio (back-end) mediante solicitudes HTTP (HTTP Requests), la cual gestiona e intermedia el acceso a la base de datos donde reside la información del sistema.

Arquitectura del Software

Este proyecto implementa el uso de la arquitectura de tipo REST, el cual, se puede definir, según Gowda (2024):

> REST (transferencia de estado representacional) es la arquitectura utilizada para diseñar servicios consumidos en diferentes plataformas y entornos para admitir la interoperabilidad y la WWW (World Wide Web). La ausencia de estado y la preparación para el consumo multiplataforma son dos atributos principales de esta arquitectura. Se ha convertido en una forma estandarizada ampliamente utilizada para publicar servicios en Internet. Las interfaces de programación de aplicaciones (API) REST forman parte integral del diseño de microservicios. Se han realizado esfuerzos de investigación para extender la arquitectura REST para admitir sistemas distribuidos. (párr.2)

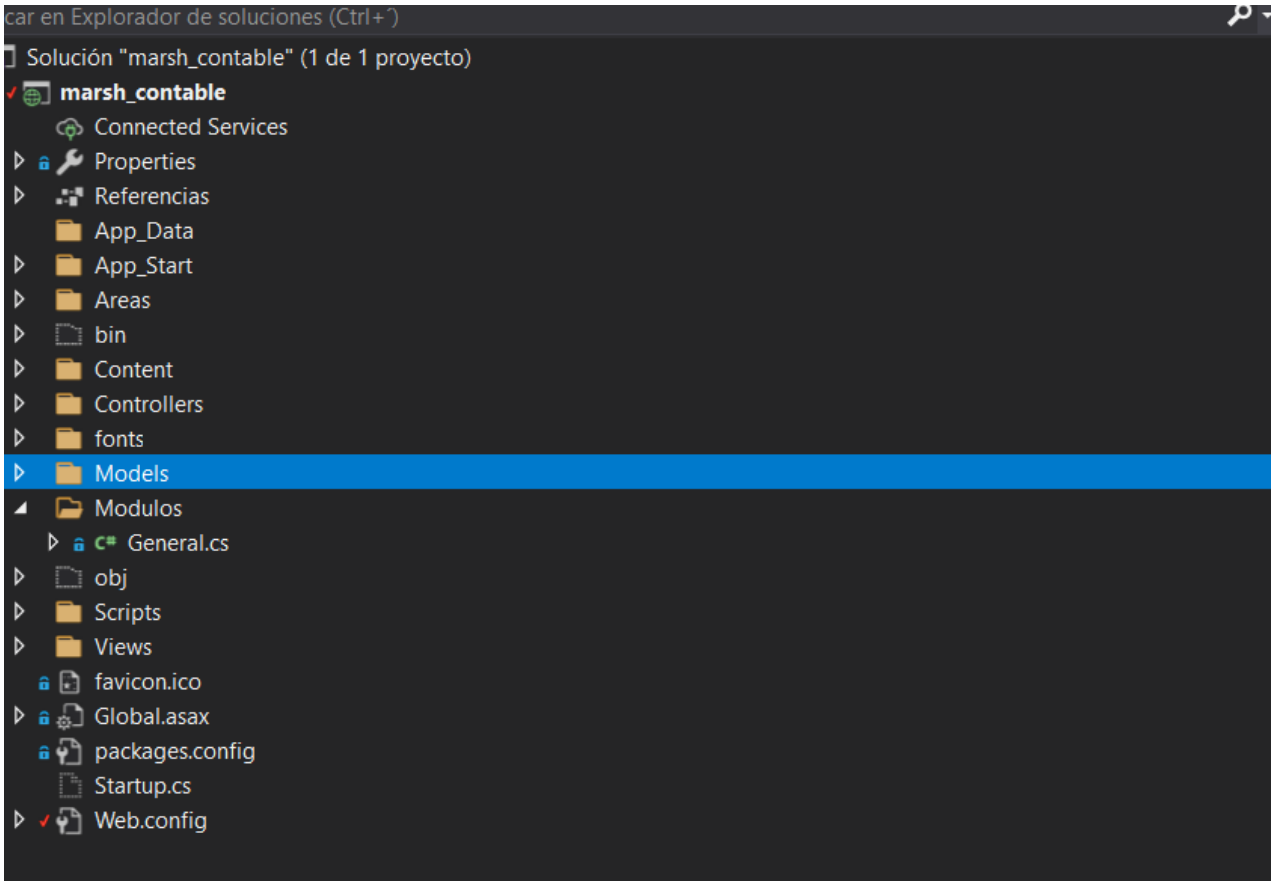
Para la implementación de la arquitectura, se empleará el entorno de desarrollo Visual Studio Professional, haciendo uso de la plantilla ASP.NET Core Web API. Esta plantilla proporciona una base estructural que favorece la construcción de proyectos escalables, permitiendo tanto el desarrollo de una interfaz de programación de aplicaciones como la integración de un sitio web dentro de un mismo proyecto. No obstante, se optó por externalizar la capa de interfaz gráfica de usuario a una solución independiente, aspecto que será abordado en detalle en secciones posteriores del presente documento.

El desarrollo del proyecto se iniciará con la definición de un archivo de modelado de base de datos con extensión .edmx, cuya función principal es establecer una correspondencia sincronizada entre la estructura relacional de la base de datos y las clases del proyecto, facilitando así una integración más coherente y mantenible entre ambas capas. Paralelamente, se definirán los ViewModels, estructuras de datos que permiten organizar y adaptar la información a los requerimientos específicos de cada operación del sistema. Estos resultan particularmente valiosos en los procesos de generación de reportes, dado que su naturaleza flexible permite consolidar datos provenientes de múltiples tablas mediante operaciones de combinación (JOIN) en el lenguaje de

SQL. Una vez establecidas esto, se procede al desarrollo del núcleo funcional del sistema, los controladores, estos componentes son los responsables de recibir las solicitudes entrantes mediante el protocolo HTTP, a través de los métodos GET, POST, PUT y DELETE, procesar la lógica de negocio asociada y gestionar la persistencia de los datos en la base de datos. Es en esta capa donde se concentran y articulan los principales algoritmos que gobiernan el comportamiento de la aplicación. A continuación, se presenta una representación gráfica de la estructura de carpetas que organizará el proyecto de C# correspondiente al REST API:

Ilustración 29

Estructura Carpetas Back-End



Fuente: Elaboración propia

Como se aprecia en la figura anterior, el proyecto se organiza en dos carpetas principales que estructuran la arquitectura interna de la aplicación. En la carpeta Models se alojará el archivo de modelado de base de datos con extensión .edmx, responsable de mantener la correspondencia entre el esquema relacional y las clases del proyecto, así como los archivos ViewModel, cuya

función es definir y delimitar con precisión los campos de datos involucrados en cada operación de consulta, edición o creación dentro del sistema.

Por su parte, la carpeta Controllers concentrará el conjunto de controladores de la aplicación, los cuales constituyen el punto de entrada de las solicitudes HTTP provenientes de la capa del front-end hacia el REST API. De este modo, cada controlador actúa como intermediario entre la lógica de negocio del sistema y las peticiones iniciadas desde la interfaz de usuario, garantizando una separación clara de responsabilidades entre las distintas capas de la arquitectura.

Diseño de entradas

En el presente apartado se exponen las principales interfaces de entrada del sistema mediante representaciones visuales preliminares. Las figuras que se presentan a continuación tienen carácter de wireframes, es decir, prototipos esquemáticos de baja o media fidelidad, por lo que el resultado final de la implementación podrá diferir, en mayor o menor medida, de las representaciones aquí incluidas. El propósito de estas ilustraciones es ofrecer una aproximación visual al aspecto y comportamiento esperado del sistema en su estado finalizado, con el objetivo de establecer y delimitar tanto la funcionalidad prevista como los criterios de diseño de la interfaz de usuario del front-end.

Ilustración 30

Entrada 1 Inicio de Sesión

Fuente: Elaboración propia.

La presente imagen muestra la pantalla de inicio de sesión en la cual el usuario ingresará las credenciales previamente definidas por el departamento de TI, con un correo válido y una contraseña la cual debe poseer caracteres alfanuméricos y un carácter especial. Además de poder acceder a la opción de recuperar contraseña en caso de que el usuario extraviara sus credenciales.

Ilustración 31

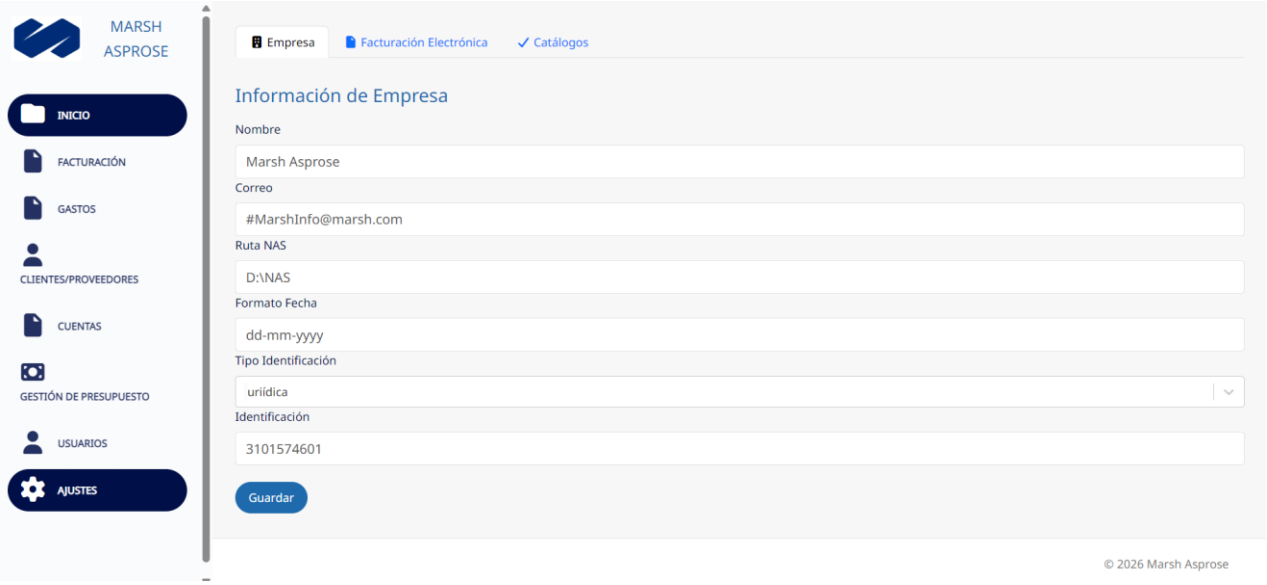
Entrada Recuperar Contraseña

Fuente: Elaboración propia

En la presente pantalla el usuario inicia un proceso de tres pasos para la recuperación de su contraseña, en donde ingresa su correo electrónico, en donde se notificará un código para continuar con el cambio de contraseña en el sistema.

Ilustración 32

Entrada Ajustes

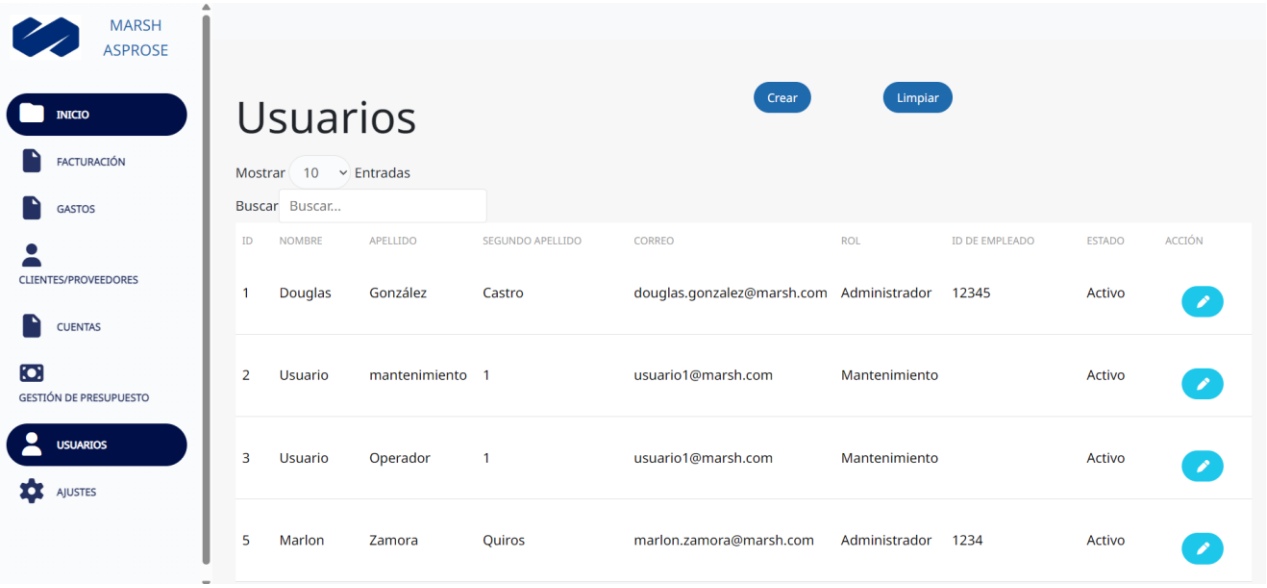


Fuente: Elaboración propia.

En la ilustración 32 se muestra el módulo de mantenimiento en donde se ingresa la información de la empresa, así como información importante para la funcionalidad como lo es el correo, nombre de fantasía, formatos de fecha e información de facturación electrónica.

Ilustración 33

Entrada Usuarios



Fuente: Elaboración propia.

Con respecto a la entrada de Usuarios se muestra como se verá la información en el sistema de los usuarios, así como su creación y edición, además de una lista de búsqueda para filtrar los resultados.

Ilustración 34

Entrada Gestión Presupuestaria

Fuente: Elaboración propia.

En la sección de entrada de gestión del presupuesto el usuario podrá administrar la creación, edición y asignación de presupuestos organizacionales, donde podrá incluir la información de los centros de costos, categorías presupuestarias y la definición del presupuesto mes a mes.

Ilustración 35

Entrada Formulario Creación

Fuente: Elaboración propia.

En la presente pantalla de creación de la gestión de presupuesto se muestra lo que es la categoría presupuestaria así como los respectivos centros de costos, esto con el fin de que el usuario pueda definir los totales de monto presupuestado por departamento y a su vez una vez hecho esto pueda asignar el presupuesto a lo largo del año.

Diseño de base de datos

Para este proceso se procederá con el diseño y estructuración de la base de datos, utilizando la herramienta MySQL Workbench como entorno de modelado, lo que permitirá definir de manera clara las entidades, sus atributos y las relaciones existentes entre ellas. De forma paralela, se elaborará un diccionario de datos que documentará detalladamente cada tabla, especificando la funcionalidad de sus campos, los tipos de datos correspondientes, así como su nivel de importancia

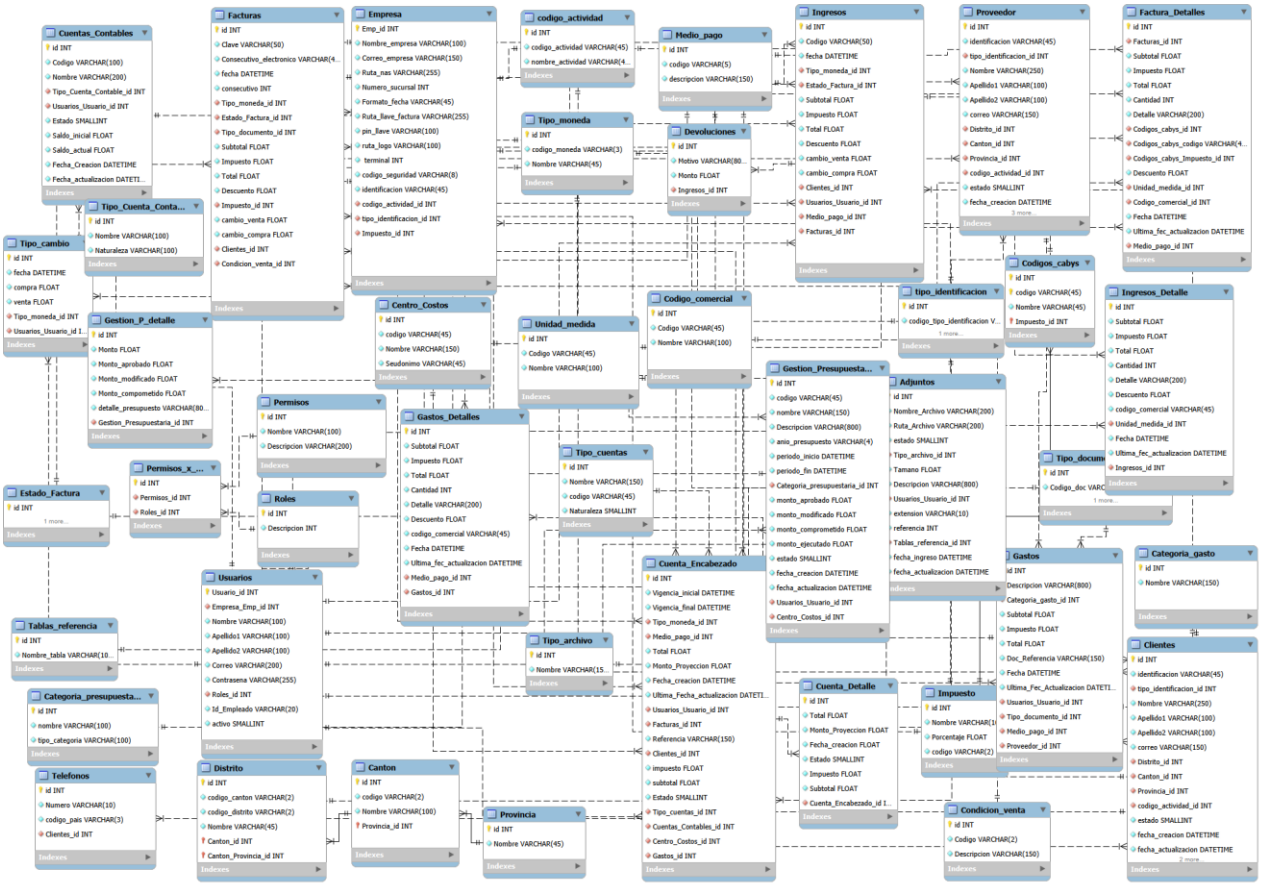
dentro del modelo, incluyendo la identificación de llaves primarias y foráneas. Este enfoque garantiza una organización coherente, integridad referencial y una adecuada comprensión del sistema de información. A continuación, se presenta la estructura propuesta de la base de datos:

Diagrama Entidad Relación

La universidad europea define un diagrama entidad relación como "una herramienta para el modelado de datos que ayuda a representar entidades en una base de datos de forma visual e intuitiva" (párr.1), se entiende que el diagrama entidad–relación de la base de datos constituye la base estructural del sistema, ya que define de manera clara y organizada cómo se representarán las entidades, sus atributos y las relaciones entre ellas. Este modelo no solo permite establecer la lógica de almacenamiento de la información, sino que también asegura la integridad, consistencia y coherencia de los datos dentro del sistema. A través del diagrama entidad relación se facilita la comprensión del flujo de información a nivel conceptual, sirviendo como guía para el diseño físico de la base de datos y permitiendo optimizar consultas, evitar redundancias y mantener un orden adecuado en la gestión de los datos dentro del programa.

Ilustración 36

Diagrama entidad relación



Fuente: Elaboración propia

Una vez definida la estructura de la base de datos mediante el modelo entidad–relación, se procede a la elaboración del diccionario de datos correspondiente, el cual permitirá documentar de forma detallada cada una de las tablas diseñadas. Este diccionario especificará los atributos que componen cada tabla, incluyendo el nombre de los campos, su tipo de dato, de esta manera, se establece una base clara y estandarizada para la implementación y mantenimiento de la base de datos, facilitando la comprensión de su estructura y asegurando la consistencia en el manejo de la información.

Diccionario de Datos

Conecta Software define un diccionario de datos como: "una colección centralizada de información sobre los datos, como su significado, relaciones, origen, uso y formato." (párr. 1), esto nos da una idea de que un diccionario de datos es un recurso que nos permite estandarizar la definición de los datos, asegurar su correcta interpretación y facilitar la comunicación entre desarrolladores, analistas y usuarios.

Tabla 18

Tabla Adjuntos

Nombre	Tipo de Dato	Default	Comment
id	INT		ID del adjunto.
Nombre_Archivo	VARCHAR(200)		Nombre del archivo.
Ruta_Archivo	VARCHAR(200)		Ruta del archivo (cifrado).
estado	SMALLINT		Activo o Inactivo.
Tipo_archivo_id	INT		El tipo de archivo.
Tamano	FLOAT		Tamaño en KB del archivo.
Descripcion	VARCHAR(800)		Descripción del archivo.
Usuarios_Usuario_id	INT		Usuario que registra el archivo.
extension	VARCHAR(10)		Extensión del archivo PDF, MSG, DOCX, etc.
referencia	INT		Número de referencia.

Nombre	Tipo de Dato	Default	Comment
Tablas_referencia_id	INT		Tabla a la cual pertenece, por ejemplo, Facturas, Gastos, Ingresos, etc.
fecha_ingreso	DATETIME		Fecha de creación.
fecha_actualizacion	DATETIME		Última fecha de actualización del documento.

Fuente: Desarrollo propio

Tabla 19

Cuenta_Encabezado

Nombre de la columna	Tipo de dato	Primary Key	Default	Comentario
id	INT	✓		ID de la cuenta por pagar.
Vigencia_inicial	DATETIME			Vigencia de inicio de la cuenta.
Vigencia_final	DATETIME			Vigencia final de la cuenta.
Tipo_moneda_id	INT			Moneda de la cuenta.
Medio_pago_id	INT			Medio de pago de la cuenta.
Total	FLOAT			Total de la cuenta.
Monto_Proyeccion	FLOAT			Monto de proyección de la cuenta.
fecha_Creacion	DATETIME			Fecha de registro en sistema.

Nombre de la columna	Tipo de dato	Primary Key	Default	Comentario
Ultima_Fecha_Actualizacion	DATETIME			Última fecha de actualización.
Usuarios_Usuario_id	INT			Usuario que registra la cuenta.
Facturas_id	INT			Id de referencia de la factura.
Estado_cuentas_id	INT			Referencia al estado de cuenta.
Referencia	VARCHAR(150)			Documento de referencia de la cuenta.
Clientes_id	INT			Id del cliente al cual se adjudica la cuenta.
impuesto	FLOAT			Impuesto de la cuenta.
subtotal	FLOAT			Subtotal de la cuenta.
Estado	SMALLINT			Activo o inactivo.
Tipo_Cuentas_id	Int			Tipo de cuenta CXC o CXP.
Cuentas_Contables_id	Int			Referencia al tipo de cuenta contable.
Centro_costos_id	Int			Referencia al centro de costos asignado.
Gastos_id	Int			Gastos en caso de que sea obligatorio.

Fuente: Desarrollo propio

Tabla 20

Cuenta_Detalle

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID de la cuenta detalle.
Total	Float			Total de la cuenta detalle.

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
Monto_Proyeccion	FLOAT			Monto proyección del detalle.
impuesto	FLOAT			Impuesto del detalle cuenta.
Subtotal	FLOAT			Subtotal del detalle.
Estado	FLOAT			Estado de la cuenta detalle
Cuenta_Encabezado_id	INT			ID del maestro cuenta.

Fuente: Desarrollo propio

Tabla 21

Canton

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del cantón.
Codigo	VARCHAR(2)			Código del cantón.
Nombre	VARCHAR(100)			Nombre del cantón.
Provincia_id	INT	✓		ID de la provincia.

Fuente: Desarrollo propio

Tabla 22

Categoria_gasto

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		id de la categoría gasto.
Nombre	VARCHAR(150)			Nombre de la categoría gasto.

Fuente: Desarrollo propio

Tabla 23

Categoria_presupuestaria

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID categoría de gestione presupuestaria.
nombre	VARCHAR(100)			Nombre de categoría de gestión.
tipo_categoria	VARCHAR(100)			Tipo de categoría código.

Fuente: Desarrollo propio

Tabla 24

Centro_Costos

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		Id interno del centro de costos.
Código	VARCHAR(45)			Código del centro de costos.
Nombre	VARCHAR(150)			Nombre del centro de costos.
Seudonimo	VARCHAR(45)			Código de referencia del centro de costos.

Fuente: Desarrollo propio

Tabla 25

Clientes

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del cliente.
Identificacion	VARCHAR(45)			Identificación del cliente.

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
tipo_Identificacion_id	INT			Tipo de identificación ej.: Físico, Jurídico, Dimex, NITE.
Nombre	VARCHAR(250)			Nombre del cliente.
Apellido1	VARCHAR(100)			Primer apellido del cliente.
Apellido2	VARCHAR(100)			Segundo apellido del cliente.
correo	VARCHAR(150)			Correo del cliente.
Distrito_id	INT			Distrito al que pertenece.
Canton_id	INT			Cantón al que pertenece.
Provincia_id	INT			Provincia a la que pertenece.
Codigo_actividad_id	INT			Código de actividad del cliente.
estado	SMALLINT			Activo o inactivo.
fecha_Creacion	DATETIME			Fecha en que fue ingresado al sistema.
fecha_Actualizacion	DATETIME			Fecha de última actualización del cliente.
exonerado	SMALLINT			Si el cliente es exonerado de impuestos.
OtrasSenas	VARCHAR(200)			Dirección exacta.

Fuente: Desarrollo propio

Tabla 26

Codigo_comercial

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del código comercial.
Codigo	VARCHAR(45)			Código comercial para documentos electrónicos.
Nombre	VARCHAR(100)			Nombre del código comercial.

Fuente: Desarrollo propio

Tabla 27

Codigos_cabys

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID interno código CABYS.
Codigo	VARCHAR(45)	✓		ID Hacienda código CABYS.
Nombre	VARCHAR(45)			Nombre del código.
Impuesto_id	INT	✓		Impuesto que aplica para este código CABYS.

Fuente: Desarrollo propio

Tabla 28

Condición_venta

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		Id interno de la condición de venta.
Codigo	VARCHAR(2)			Código de la condición de venta para documentos electrónicos.
Descripcion	VARCHAR(150)			Descripción de la condición de venta.

Fuente: Desarrollo propio

Tabla 29

Cuentas_Contables

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID interno cuentas contables.
Codigo	VARCHAR(100)			Código de la cuenta contable.
Nombre	VARCHAR(200)			Nombre de la cuenta contable.
Tipo_Cuenta_Contable_id	INT			Tipo de cuenta contable referencia.

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
Usuarios_Usuario_id	INT			ID del usuario que registro.
Estado	SMALLINT			Activo o inactivo.
Saldo_inicial	FLOAT			Saldo iniciar de la cuenta.
Saldo_actual	FLOAT			Saldo actual de la cuenta.
fecha_Creacion	DATETIME			Fecha de creación.
Fecha_Actualizacion	DATETIME			Última fecha de actualización.

Fuente: Desarrollo propio

Tabla 30

Distrito

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del distrito.
Codigo_canton	VARCHAR(2)			Código del cantón.
Codigo_distrito	VARCHAR(2)			Código del distrito.
Nombre	VARCHAR(45)			Nombre del distrito.
Canton_id	INT	✓		ID del cantón al que pertenece.
Canton_Provincia_id	INT	✓		ID de la provincia a la que pertenece.

*Fuente:*Desarrollo propio

Tabla 31

Empresa

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
Emp_id	INT	✓		ID de la empresa, registro único.
Nombre_empresa	VARCHAR(100)			Nombre por mostrar de la empresa.
Correo_empresa	VARCHAR(150)			Correo de la compañía.

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
Ruta_nas	VARCHAR(255)			Configuración ruta de la NAS donde estarán los archivos.
Numero_sucursal	INT			Número de sucursal, para facturación.
Formato_fecha	VARCHAR(45)			Formato de fecha a mostrar en el sistema.
Ruta_llave_factura	VARCHAR(255)			Ruta donde estará la llave p12 para firmar documentos electrónicos.
pin_llave	VARCHAR(100)			PIN de la llave firma documentos electrónicos (cifrado).
ruta_logo	VARCHAR(100)			URL del logo a mostrar.
terminal	INT			ID terminal para facturación.
Codigo_seguridad	VARCHAR(8)			Código de seguridad para documentos electrónicos, en caso de no tener uno definido el sistema generara uno dinámicamente.
Identificacion	VARCHAR(45)			Cedula jurídica de la empresa.
Codigo_actividad_id	INT			Código de actividad económica.
tipo_Identificacion_id	INT			Tipo de identificación (Jurídica).
Impuesto_id	INT			Impuesto por defecto usado por el sistema.

Fuente: Desarrollo propio

Tabla 32

Estado_Factura

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del estado de la factura.
Nombre	VARCHAR(100)			Nombre de estado de la factura ej. Pagado, Pendiente, etc.

Fuente: Desarrollo propio

Tabla 33

Tipo_cuentas

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del estado de cuenta.
Nombre	VARCHAR(150)			Nombre del estado cuenta.
Código	VARCHAR(45)			Código de la cuenta por tipo.
Naturaleza	Smallint			Naturaleza del tipo de cuenta.

Fuente: Desarrollo propio

Tabla 34

Factura_Detalles

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID interno detalle factura.
Facturas_id	INT			ID de la factura.
Subtotal	FLOAT			Subtotal del detalle.
Impuesto	FLOAT			Impuesto del detalle.
Total	FLOAT			Total, del detalle.
Cantidad	INT			Cantidad de ítems.
Detalle	VARCHAR(200)			Descripción de la línea.
Codigos_cabys_id	INT			ID del código CABYS para

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
				documentos electrónicos.
Codigos_cabys_Código	VARCHAR(45)			Código CABYS.
Codigos_cabys_Impuesto_id	INT			Impuesto del Código CABYS.
Descuento	FLOAT			Descuento de la línea.
Unidad_medida_id	INT			Unidad de medida para documentos electrónicos.
Codigo_comercial_id	INT			Código comercial para documentos electrónicos.
Fecha	DATETIME			Fecha registro.
ultima_Fec_Actualizacion	DATETIME			Última fecha actualización.
Medio_pago_id	INT			ID del medio de pago para documentos electrónicos.

Fuente: Desarrollo propio

Tabla 35

Facturas

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID interno de la factura.
Clave	VARCHAR(50)			Clave de 50 dígitos para documentos electrónicos.
Consecutivo_electronico	VARCHAR(45)			Consecutivo para documentos electrónicos.
fecha	DATETIME			Fecha de creación del documento.
consecutivo	INT			Consecutivo interno del documento.

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
Tipo_moneda_id	INT			Tipo de moneda de la factura.
Estado_Factura_id	INT			Estado en que se encuentra la factura.
Tipo_documento_id	INT			Tipo de documento nota crédito, factura, nota débito.
Subtotal	FLOAT			Subtotal de factura.
Impuesto	FLOAT			Impuesto de la factura.
Total	FLOAT			Total de la factura.
Descuento	FLOAT			Descuento aplicado a la factura.
Impuesto_id	INT			Referencia al impuesto aplicado.
cambio_venta	FLOAT			Tipo de cambio venta en el que fue creada la factura.
cambio_compra	FLOAT			Tipo de cambio compra en que fue creado la factura.
Clientes_id	INT			Cliente al cual se le facturó.
Condicion_venta_id	INT			Condición de venta para documentos electrónicos.

Fuente: Desarrollo propio

Tabla 36

Gastos

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del gasto.
Descripcion	VARCHAR(800)			Descripción del gasto.
Categoria_gasto_id	INT			Categoría del gasto.
Subtotal	FLOAT			Subtotal del gasto.
Impuesto	FLOAT			Total, de impuesto del gasto.
Total	FLOAT			Total del gasto.

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
Doc_Referencia	VARCHAR(150)			Documento de referencia del gasto.
Fecha	DATETIME			Fecha de creación del gasto.
ultima_Fec_Actualizacion	DATETIME			Última fecha de actualización del gasto.
Usuarios_Usuario_id	INT			Usuario que registra el gasto.
Tipo_documento_id	INT			Referencia tipo de documento.
Medio_pago_id	INT			Referencia medio de pago.
Proveedor_id	INT			Referencia del proveedor gasto.

Fuente: Desarrollo propio

Tabla 37

Gastos_Detalles

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		Id interno del detalle gasto .
Subtotal	FLOAT			Subtotal del detalle de gasto.
Impuesto	FLOAT			Impuesto del detalle de gasto.
Total	FLOAT			Total del detalle de gasto.
Cantidad	INT			Cantidad del detalle .
Detalle	VARCHAR(200)			Descripción del detalle.
Descuento	FLOAT			Descuento del detalle de gasto.
Codigo_comercial	VARCHAR(45)			Código comercial.
Fecha	DATETIME			Fecha de registro.
ultima_Fec_Actualizacion	DATETIME			Última fecha de actualización.
Medio_pago_id	INT			Medio de pago del detalle de gasto.
Gastos_id	INT			ID del maestro gasto.

Fuente: Desarrollo propio

Tabla 38

Gestion_P_detalle

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		Id interno del detalle de gestión presupuestaria.
Monto	FLOAT			Monto del detalle de la gestión.

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
Monto_aprobado	FLOAT			Monto aprobado del detalle de la gestión.
Monto_modificado	FLOAT			Monto modificado del detalle de la gestión detalle.
Monto_compometido	FLOAT			Monto comprometido del detalle.
detalle_presupuesto	VARCHAR(800)			Descripción del detalle de la gestión por línea.
Gestion_Presupuestaria_id	INT			Referencia al maestro de gestión presupuestaria.

Fuente: Desarrollo propio

Tabla 39

Gestion_P_detalle

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		Id interno del detalle de gestión presupuestaria.
Monto	FLOAT			Monto del detalle de la gestión.
Monto_aprobado	FLOAT			Monto aprobado del detalle de la gestión.
Monto_modificado	FLOAT			Monto modificado del detalle de la gestión detalle.
Monto_compometido	FLOAT			Monto comprometido del detalle.
detalle_presupuesto	VARCHAR(800)			Descripción del detalle de la gestión por línea.
Gestion_Presupuestaria_id	INT			Referencia al maestro de gestión presupuestaria.

Fuente: Desarrollo propio

Tabla 40

Gestion_Presupuestaria

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID de gestión presupuestaria.
Código	VARCHAR(45)			Código referencia de gestión.
nombre	VARCHAR(150)			Nombre de la gestión.
Descripción	VARCHAR(800)			Descripción sobre que es la gestión.
anio_presupuesto	VARCHAR(4)			Año del presupuesto.
periodo_inicio	DATETIME			Período en que inicia la gestión.
periodo_fin	DATETIME			Período en que finaliza la gestión.
Categoria_presupuestaria_id	INT			Categoría de la gestión.
monto_aprobado	FLOAT			Monto aprobado de la gestión.
monto_modificado	FLOAT			Monto modificado en caso de que haya sido actualizado.
monto_comprometido	FLOAT			Monto comprometido de la gestión.
monto_ejecutado	FLOAT			Monto que se ejecutó en esta gestión.
estado	SMALLINT			Estado activo o inactivo.
fecha_Creacion	DATETIME			Fecha en que se creó la gestión.
fecha_Actualizacion	DATETIME			Última fecha de actualización de la gestión.
Usuarios_Usuario_id	INT			Usuario que registra la gestión.

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
Centro_Costos_id	INT			ID del centro de costos asignado.

Fuente: Desarrollo propio

Tabla 41

Impuesto

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID tipo de impuesto.
Nombre	VARCHAR(100)			Nombre del impuesto.
Porcentaje	FLOAT			Porcentaje del impuesto.
Codigo	VARCHAR(2)			Código del impuesto para documentos electrónicos.

Fuente: Desarrollo propio

Tabla 42

Ingresos

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del ingreso.
Codigo	VARCHAR(50)			Código de referencia del ingreso.
fecha	DATETIME			Fecha en que se registró el ingreso.
Tipo_moneda_id	INT			Tipo de moneda del ingreso.
Estado_Factura_id	INT			Estado del ingreso.
Subtotal	FLOAT			Subtotal sumado de los detalles del ingreso.
Impuesto	FLOAT			Total, de impuestos del ingreso.
Total	FLOAT			Total, sumado de los detalles del ingreso.
Descuento	FLOAT			Total, de descuento del ingreso.

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
cambio_venta	FLOAT			Tipo de cambio de venta en el momento del registro.
cambio_compra	FLOAT			Tipo cambio compra en el momento del registro.
Clientes_id	INT			Id del cliente al cual se le atribuye el ingreso.
Usuarios_Usuario_id	INT			Usuario que registra el ingreso.
Medio_pago_id	INT			ID del medio de pago ej. Efectivo, Tarjeta.
Facturas_id	Int			Referencia de factura a la que aplica.

Fuente: Desarrollo propio

Tabla 43

Ingresos_Detalle

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del detalle del ingreso.
Subtotal	FLOAT			Subtotal del detalle.
Impuesto	FLOAT			Impuesto del detalle.
Total	FLOAT			Total sumado del detalle.
Cantidad	INT			Cantidad del detalle.
Detalle	VARCHAR(200)			Descripción del detalle.
Descuento	FLOAT			Descuento del detalle.
Codigo_comercial	VARCHAR(45)			Código comercial del detalle.
Unidad_medida_id	INT			Unidad de medida del detalle ingreso.
Fecha	DATETIME			Fecha de creación.
ultima_Fec_Actualizacion	DATETIME			Última fecha de actualización.
Ingresos_id	INT			Referencia al maestro ingresos.

Fuente: Desarrollo propio

Tabla 44

Medio_pago

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID medio pago.
Codigo	VARCHAR(5)			Código del medio de pago para documentos electrónicos.
Descripción	VARCHAR(150)			Nombre del medio de pago.

Fuente: Desarrollo propio

Tabla 45

Permisos

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID de los permisos.
Nombre	VARCHAR(100)			Nombre del permiso.
Descripcion	VARCHAR(200)			Descripción de que se hará con ese permiso.

Fuente: Desarrollo propio

Tabla 46

Permisos_x_rol

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID auto incremental de la tabla de permisos_x_rol.
Permisos_id	INT			ID del permiso nos permitirá definir los permisos específicos por rol.
Roles_id	INT			Rol id necesario para hacer la relación entre el rol con los permisos que este tendrá.

Fuente: Desarrollo propio

Tabla 47

Proveedor

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del proveedor.
Identificacion	VARCHAR(45)			Cedula del proveedor.
tipo_Identificacion_id	INT			Tipo de identificación del proveedor.
Nombre	VARCHAR(250)			Nombre del proveedor.
Apellido1	VARCHAR(100)			Primer apellido del proveedor.
Apellido2	VARCHAR(100)			Segundo apellido del proveedor.
correo	VARCHAR(150)			Correo del proveedor.
Distrito_id	INT			Distrito del proveedor.
Canton_id	INT			Canton del proveedor.
Provincia_id	INT			Provincia del proveedor.
Codigo_actividad_id	INT			Código de actividad del proveedor.
estado	SMALLINT			Activo o inactivo.
fecha_Creacion	DATETIME			Fecha en que se creó el proveedor.
fecha_Actualizacion	DATETIME			Última fecha de actualización.
exonerado	SMALLINT			Si cliente es exonerado.
OtrasSenas	VARCHAR(200)			Dirección exacta del proveedor.

Fuente: Desarrollo propio

Tabla 48

Provincia

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID de provincia.
Nombre	VARCHAR(45)			Nombre de la provincia.

Fuente: Desarrollo propio

Tabla 49

Roles

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del ROL.
Descripcion	INT			Breve descripción del rol.

Fuente: Desarrollo propio

Tabla 50

Tablas_referencia

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID tablas de referencia.
Nombre_tabla	VARCHAR(100)			Nombre descriptivo de las tablas de para la relación con su respectivo archivo.

Fuente: Desarrollo propio

Tabla 51

Telefonos

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del teléfono.
Numero	VARCHAR(10)			Número de teléfono.
Codigo_pais	VARCHAR(3)			Código de país del teléfono.
Clientes_id	INT			Cliente al cual pertenece el teléfono.

Fuente: Desarrollo propio

Tabla 52

Tipo_Cuenta_Contable

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		Id interno del tipo de cuenta contable.
Nombre	VARCHAR(100)			Nombre del tipo de cuenta.

Naturaleza	VARCHAR(100)			Naturaleza del tipo de cuenta ej. Deudora, Acreedora.

Fuente: Desarrollo propio

Tabla 53

Tipo_archivo

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID de tipo de archivo.
Nombre	VARCHAR(150)			Nombre descriptivo del tipo de archivo.

Fuente: Desarrollo propio

Tabla 54

Tipo_cambio

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID tabla tipo de cambio.
fecha	DATETIME			Fecha en que se registra.
compra	FLOAT			Monto compra.
venta	FLOAT			Monto venta.
Tipo_moneda_id	INT			Tipo de moneda (colones, dólares, euros).
Usuarios_Usuario_id	INT			Usuario que registra el tipo de cambio.

Fuente: Desarrollo propio

Tabla 55

Tipo_documento

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		Id interno del tipo documento.
Codigo_doc	VARCHAR(45)			Código del tipo de documento.
Nombre	VARCHAR(100)			Nombre del tipo de documento.

Fuente: Desarrollo propio

Tabla 56

Tipo_moneda

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del tipo moneda.
Codigo_moneda	VARCHAR(3)			Código ISO de la moneda ej. CRC, USD.
Nombre	VARCHAR(45)			Nombre del tipo de moneda.

Fuente: Desarrollo propio

Tabla 57

Unidad_medida

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID interno de unidad de medida.
Codigo	VARCHAR(45)			Código para unidad de medida para documentos electrónicos.
Nombre	VARCHAR(100)			Nombre de la unidad de medida.

Fuente: Desarrollo propio

Tabla 58

Usuarios

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
Usuario_id	INT	✓		ID del usuario.
Empresa_Emp_id	INT			ID de la empresa para relacionarlo al sistema.
Nombre	VARCHAR(100)			Nombre del usuario.
Apellido1	VARCHAR(100)			Primer apellido del usuario.
Apellido2	VARCHAR(100)			Segundo apellido del usuario.
Correo	VARCHAR(200)			Correo del usuario.
Contrasena	VARCHAR(255)			Contraseña cifrada del usuario.
Roles_id	INT			ID del Rol que tendrá el usuario.
Id_Empleado	VARCHAR(20)			Campo para uso interno de id de empleado de la compañía.
activo	SMALLINT			Estado.

Fuente: Desarrollo propio

Tabla 59

Codigo_actividad

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID del código de actividad.
Codigo_actividad	VARCHAR(45)			Código de actividad definido por hacienda para documentos electrónicos.
nombre_actividad	VARCHAR(45)			Nombre del tipo de actividad.

Fuente: Desarrollo propio

Tabla 60

tipo_Identificación

Nombre columna	Tipo de Dato	Primary Key	Default	Comentario
id	INT	✓		ID tipo identificación.
Codigo_tipo_Identificación	VARCHAR(3)			Código del tipo identificación para documentos electrónicos.
Nombre	VARCHAR(45)			Nombre tipo identificación.

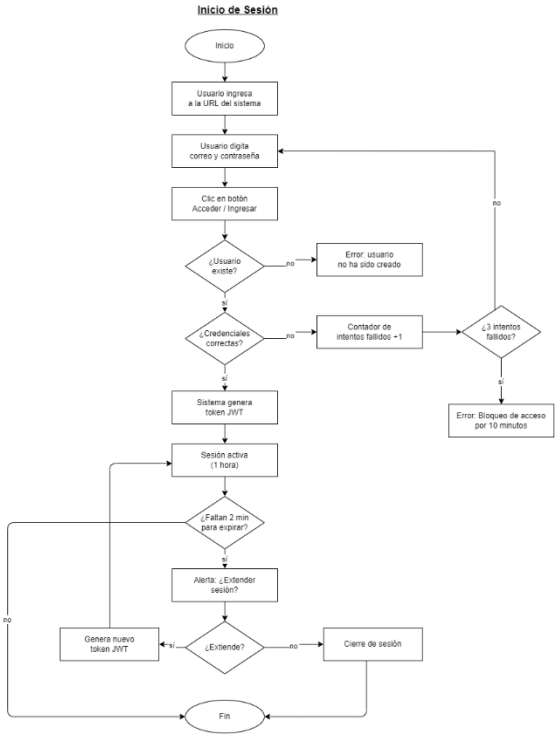
Fuente: Desarrollo propio

Diseño de Procesos

En el presente apartado se abordan, mediante representaciones gráficas, los procesos principales que ejecuta la aplicación, con el propósito de facilitar la identificación y comprensión de los flujos de trabajo que la componen, así como las interacciones que se producen entre sus distintos componentes y actores. La representación visual de estos procesos constituye un instrumento fundamental para el análisis funcional del sistema, en tanto permite verificar que cada flujo responde de manera coherente a los requerimientos establecidos en etapas previas del proyecto. Asimismo, el diseño de los procesos se ha concebido atendiendo a criterios de eficiencia operativa, de modo que cada flujo de trabajo minimice la redundancia de pasos, optimice el uso de los recursos del sistema y garantice una experiencia de usuario fluida y predecible.

Ilustración 37

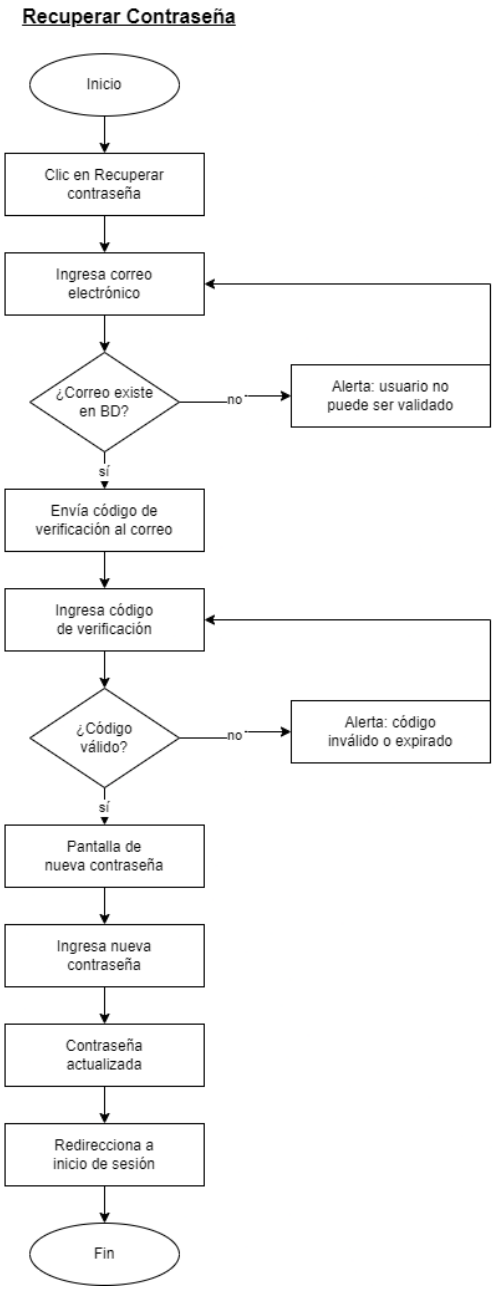
Diagrama Proceso Inicio Sesión



Fuente: Elaboración Propia

Ilustración 38

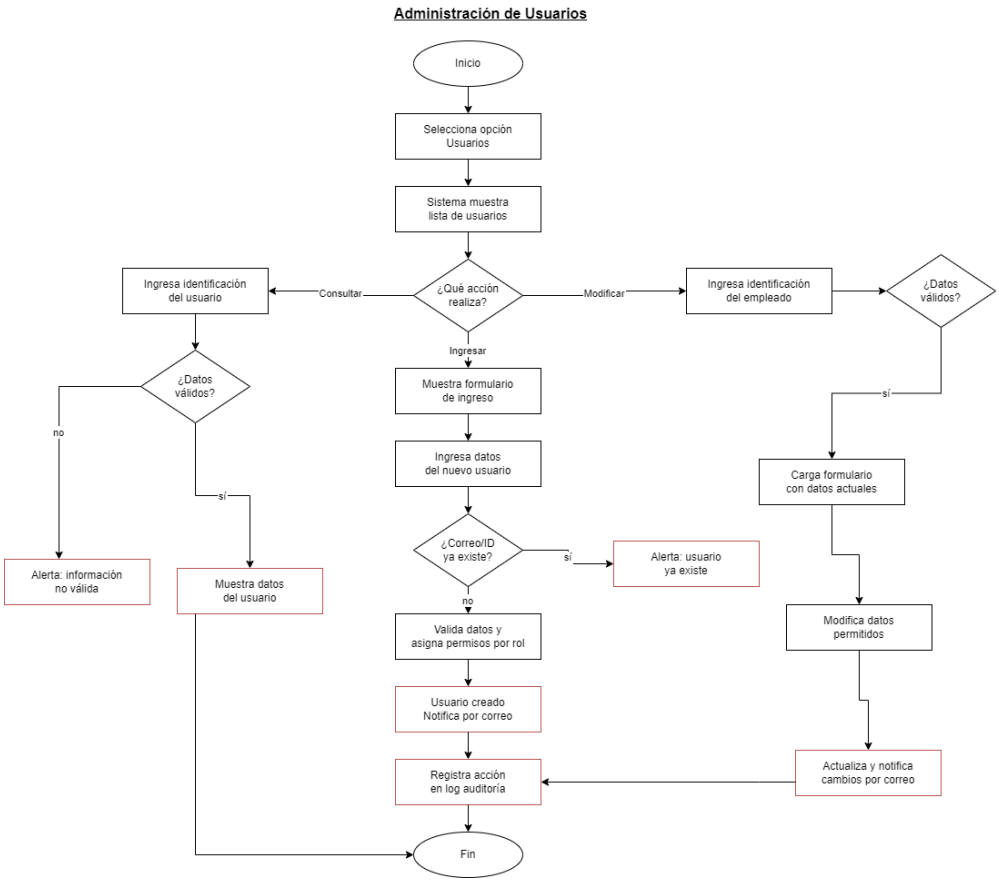
Diagrama Proceso Recuperar Contraseña



Fuente: Elaboración propia.

Ilustración 39

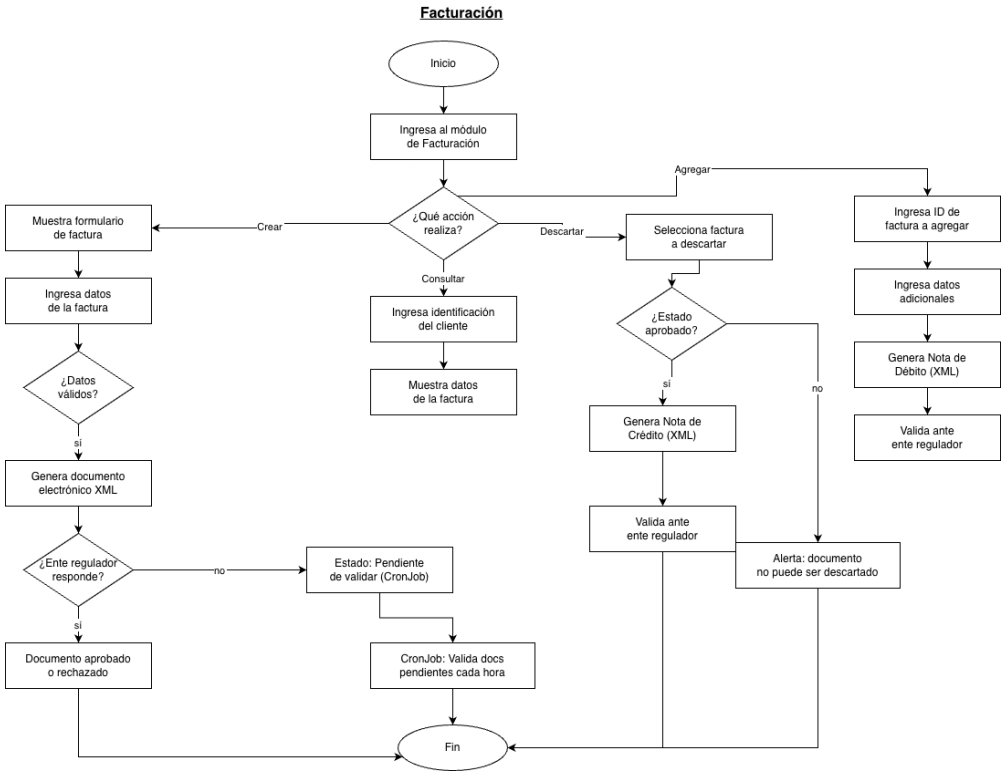
Diseño Proceso Administración Usuarios



Fuente: Elaboración propia.

Ilustración 40

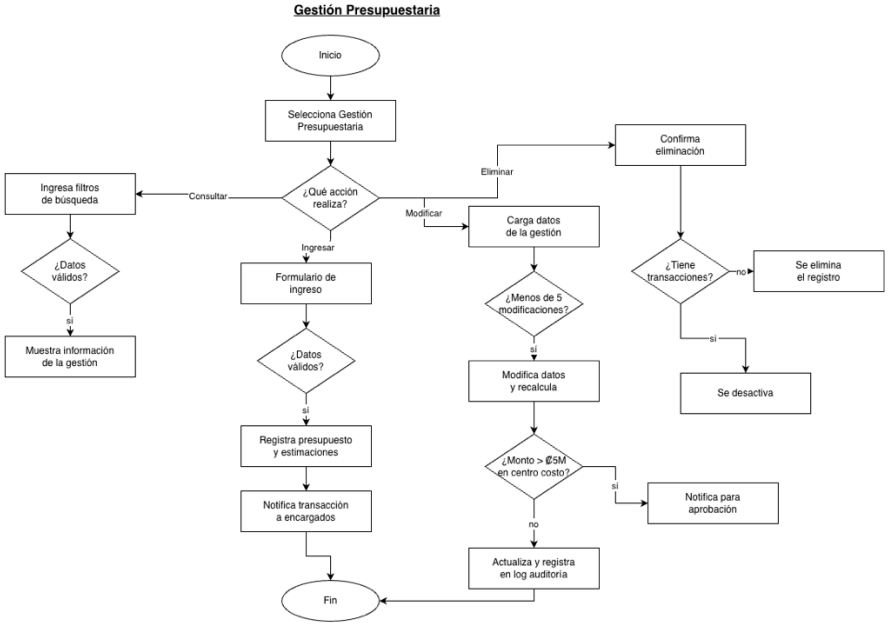
Diseño de Procesos Facturación



Fuente: Elaboración propia

Ilustración 41

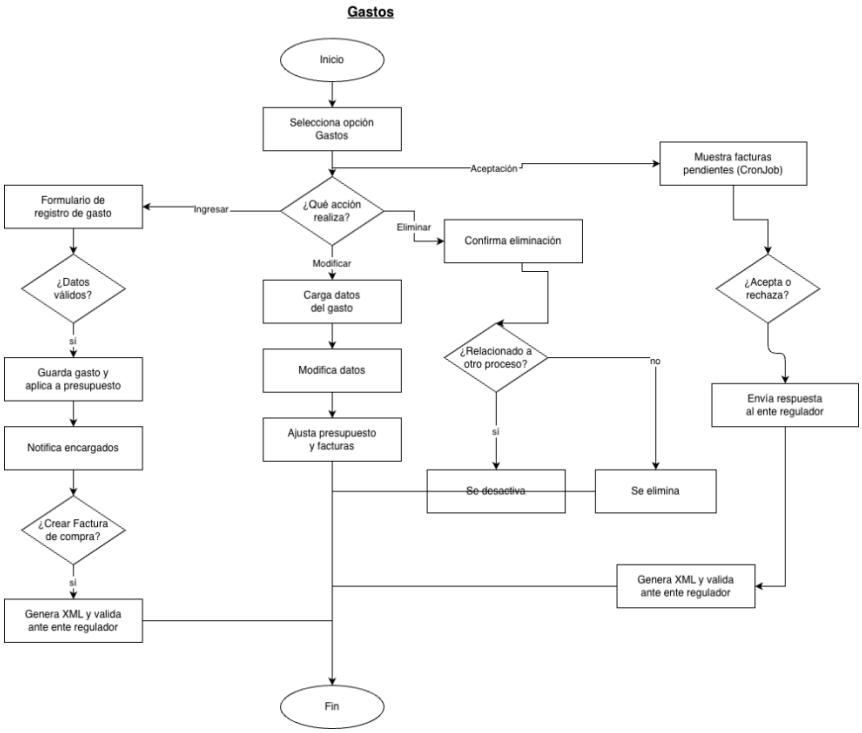
Diseño de Procesos Gestión Presupuestaria.



Fuente: Elaboración propia

Ilustración 42

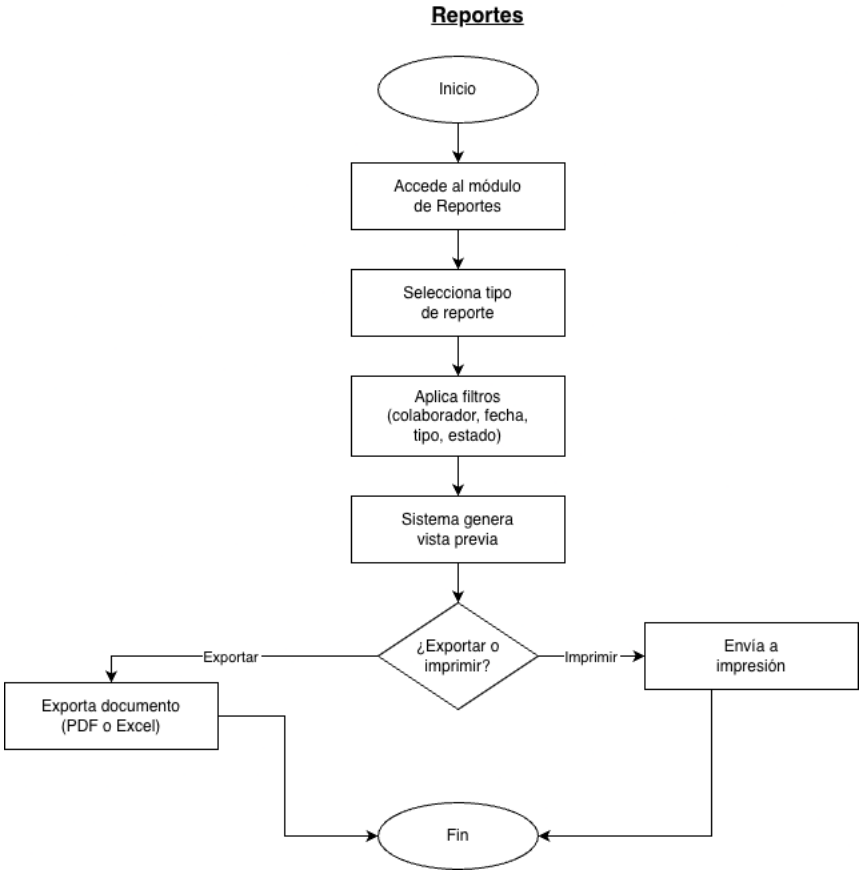
Diseño de Procesos Gastos



Fuente: Elaboración propia

Ilustración 43

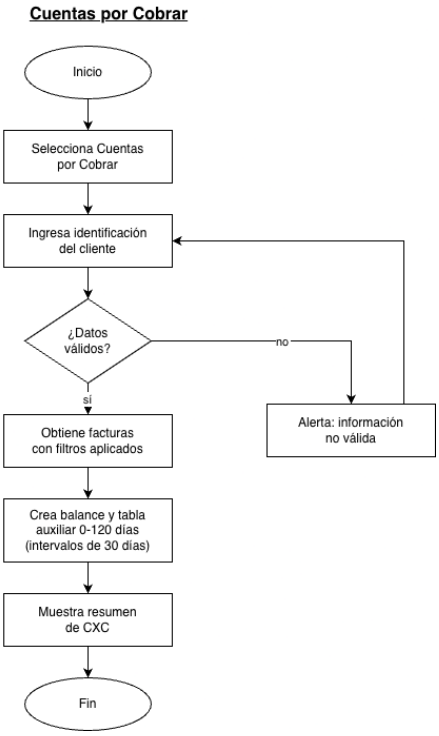
Diseño de Procesos Reportes



Fuente: Elaboración propia.

Ilustración 44

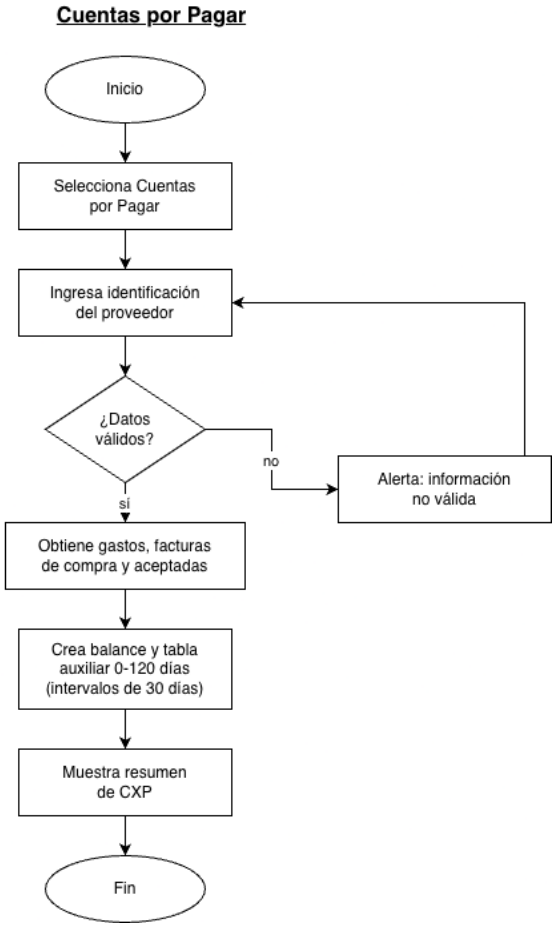
Diseño de Procesos Cuentas por Cobrar



Fuente: Elaboración propia.

Ilustración 45

Diseño de Procesos Cuentas por Pagar



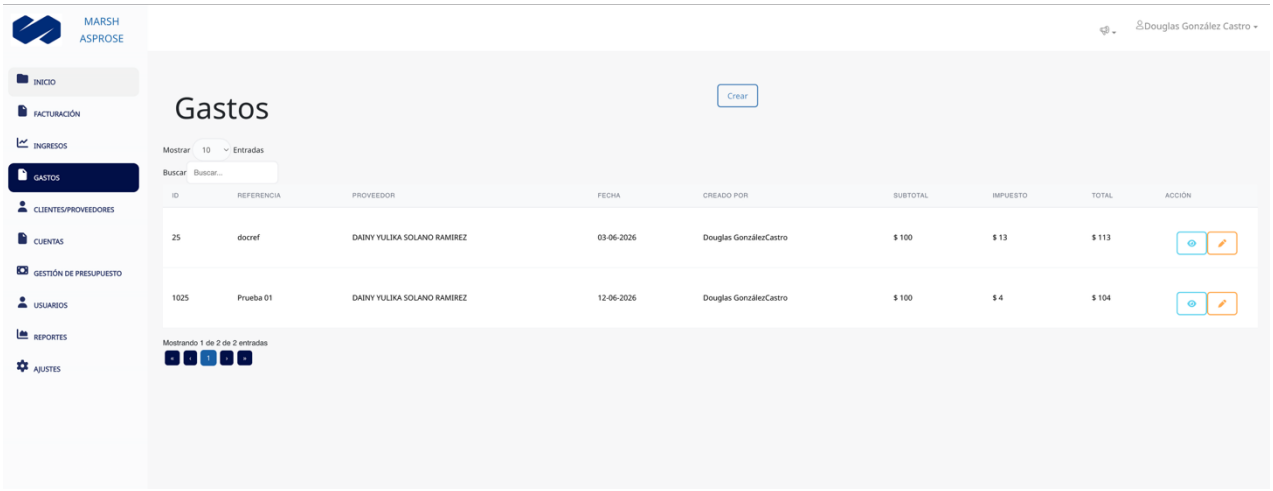
Fuente: Elaboración propia.

Diseño de Salidas

El diseño de interfaces del sistema web para la gestión financiero contable de Marsh Asprose, busca que la información almacenada por cada proceso sea entendible por el usuario final de una forma clara y concisa, mostrando información mediante tablas dinámicas, vistas resumidas para su análisis correspondiente. A continuación, se muestran algunas de las distintas salidas que tendrá el sistema.

Ilustración 46

Diseño de Salidas Gastos

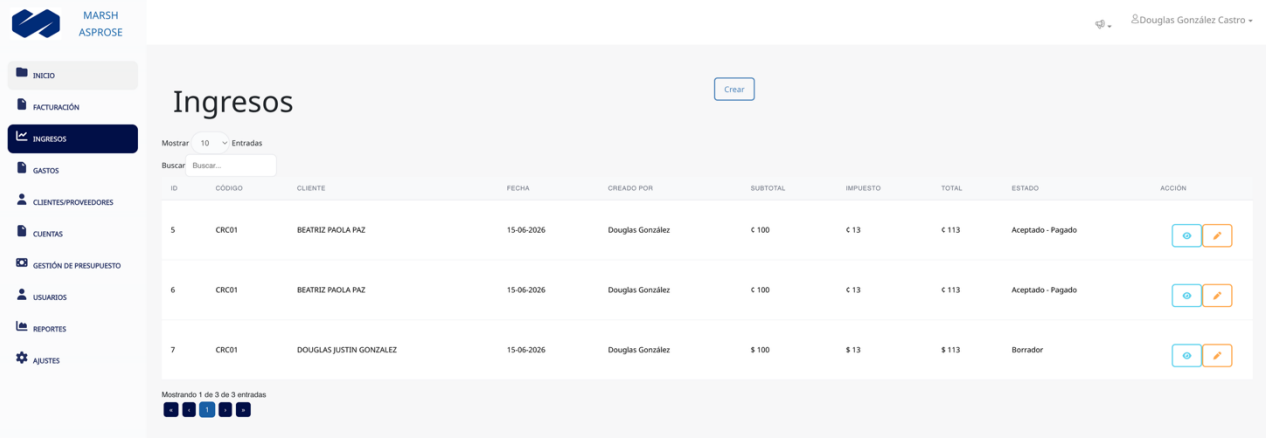


Fuente: Elaboración propia.

La figura anterior corresponde a la gestión de gastos en el sistema, se muestra la información relevante de cada gasto: el proveedor, los códigos asociados y el usuario que creó el registro, así como los valores de subtotal, impuesto y total, con el fin de agilizar el filtro de los registros.

Ilustración 47

Diseño de Salidas Ingresos

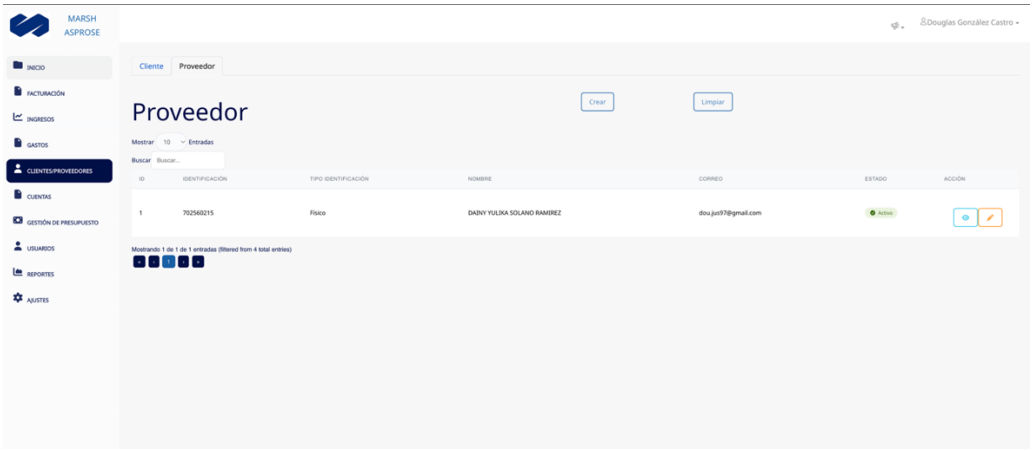


Fuente: Elaboración propia.

En la figura anterior se muestra el inicio del módulo de ingresos, y como este muestra la información principal de los gastos registrados en el sistema, por medio de este módulo los usuarios con los permisos correspondientes pueden administrar ingresos los cuales se verán reflejados en la contabilidad de los presupuestos a los cuales se asignen.

Ilustración 48

Diseño de Salidas Proveedor/Cliente



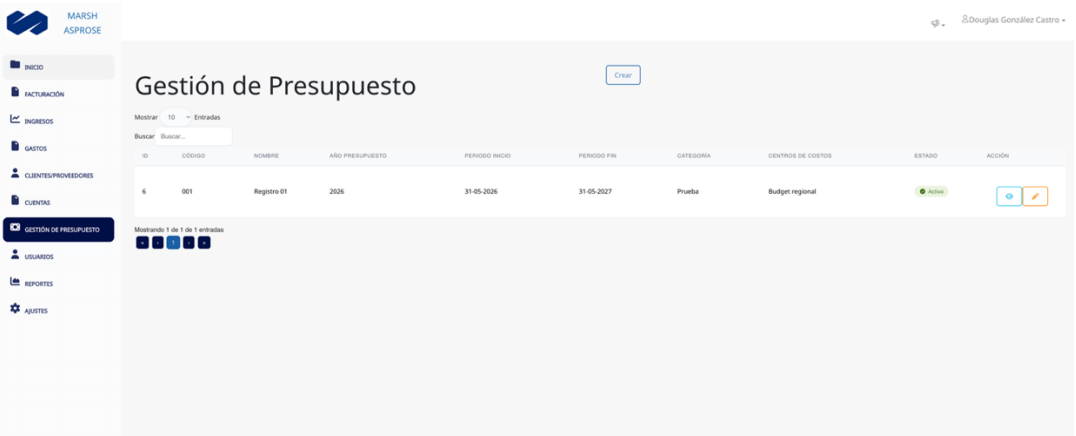
Fuente: Elaboración propia.

Este módulo se encarga de la administración y gestión de los proveedores y clientes registrados en el sistema, presentando en formato de tabla la información esencial de cada uno, que

incluye el identificador, el número de identificación, el tipo de identificación, el nombre, el correo electrónico y el estado del registro. Los proveedores y clientes registrados en este módulo resultan indispensables para su asignación en otros procesos del sistema, tales como la gestión de gastos, ingresos y facturación, garantizando así la integridad de la información a lo largo de los distintos flujos operativos.

Ilustración 49

Diseño de Salidas Gestión Presupuesto

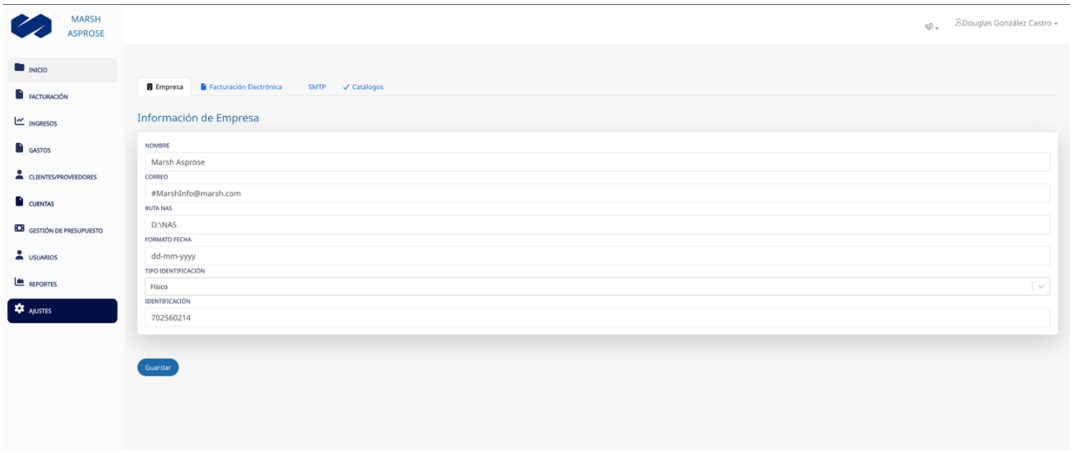


Fuente: Elaboración propia.

La figura anterior abarca el módulo de gestión de presupuestos. En esta pantalla se presentan los presupuestos previamente registrados en el sistema, permitiendo al usuario visualizar la información consolidada de cada uno para llevar a cabo su respectiva administración, seguimiento y control.

Ilustración 50

Diseño de Salidas Ajustes

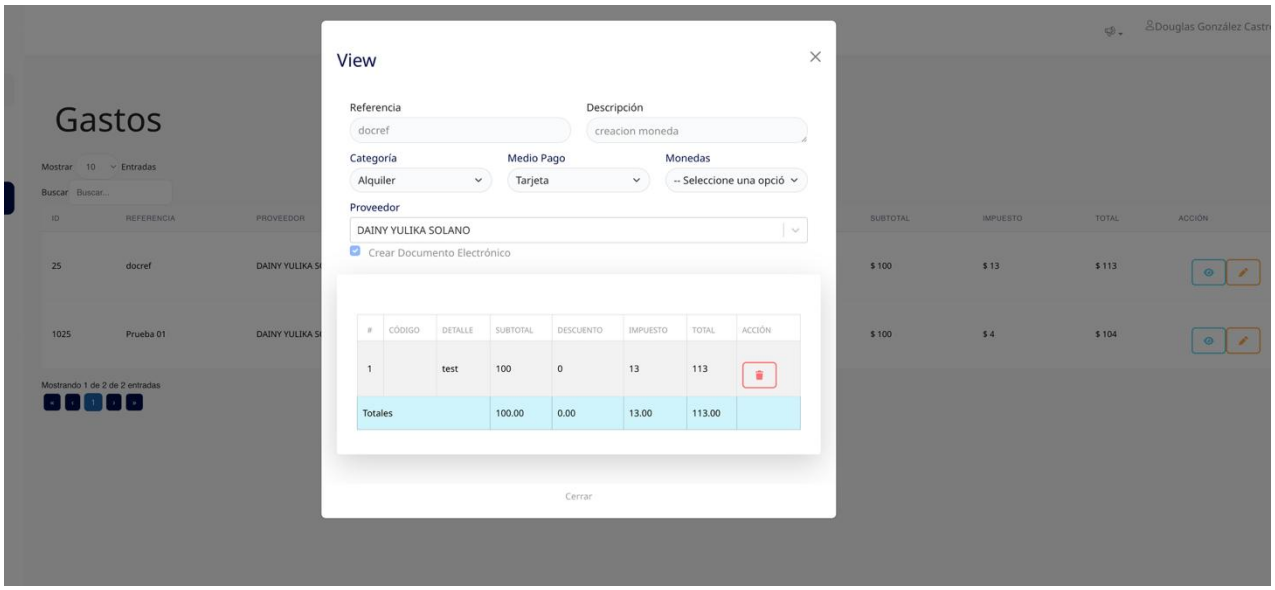


Fuente: Elaboración propia.

La figura anterior corresponde al módulo de Ajustes del sistema (Mantenimiento). En esta pantalla el usuario administrador puede registrar y actualizar los datos generales de la organización, entre los que se encuentran el nombre de la empresa, el correo electrónico, la ruta NAS, el formato de fecha, el tipo de identificación y el número de identificación. Adicionalmente, la pantalla cuenta con las pestañas de Facturación Electrónica, SMTP y Catálogos, las cuales permiten ampliar la configuración del sistema según las necesidades del usuario.

Ilustración 51

Diseño de Salidas Detalle Gastos



Fuente: Elaboración propia.

La ilustración 51 presenta la pantalla de detalle de un registro de gasto dentro del sistema. En ella se despliega la información completa asociada al gasto seleccionado, permitiendo al usuario revisar los datos ingresados durante su creación. Una vez que el gasto ha sido registrado y almacenado, el sistema lo procesa automáticamente, generando un impacto directo sobre los presupuestos asignados al período vigente, lo que garantiza la trazabilidad y consistencia de la información financiera.

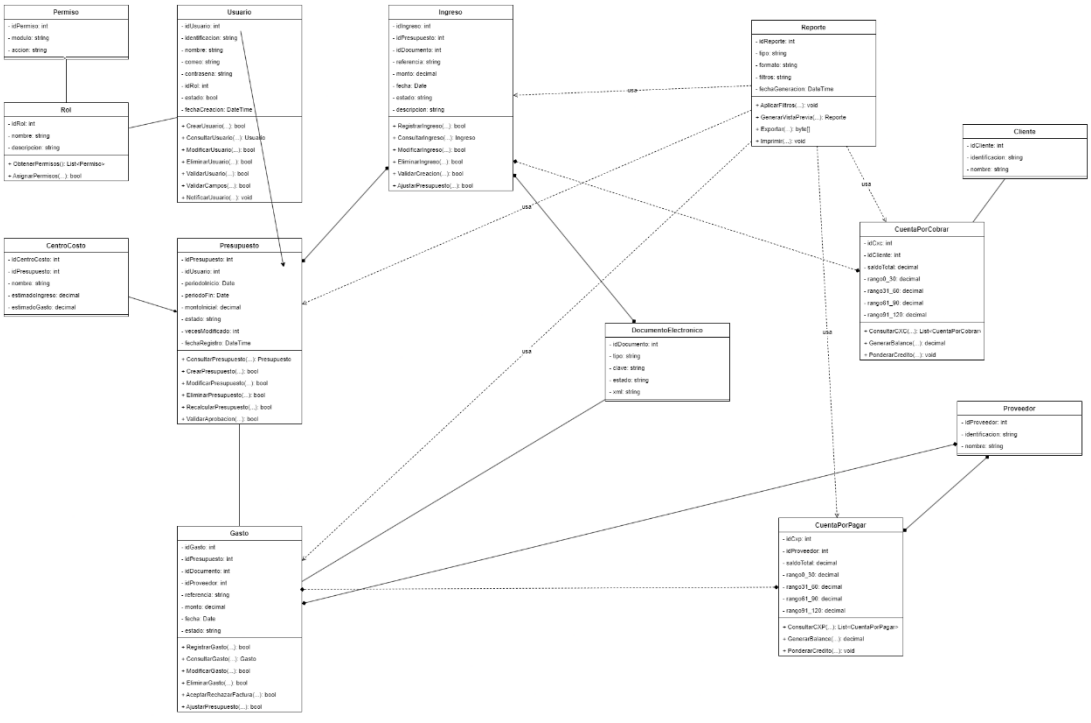
Diagramas UML

Para elaborar este punto se debe definir primero que entendemos por diagrama UML, VanZandt (2022), lo define como “una plantilla dinámica que facilita la visualización de procesos y secuencias. Esta representación visual capta meticulosamente los elementos esenciales de un sistema, incluidos los actores, las funciones, las acciones y los artefactos.” (párr.2).

A partir de lo expuesto anteriormente, los diagramas UML constituyen una herramienta fundamental para la representación gráfica y estructurada de la información técnica de un sistema. Su aplicación facilita tanto la comprensión de la lógica de programación como la validación de los requerimientos funcionales y no funcionales definidos durante la etapa de análisis. Los diagramas de secuencia que se presentan a continuación permiten comprender de manera detallada el flujo de interacciones entre los distintos componentes y actores que intervienen en la aplicación, describiendo paso a paso la manera en que el sistema responde ante las acciones del usuario.

Ilustración 52

Diagrama de Clase



Fuente: Elaboración propia.

El diagrama de clases representa la arquitectura del sistema de gestión financiero-contable Marsh Asprose, organizado en torno a trece entidades principales interconectadas, el núcleo de seguridad está compuesto por las clases Permiso, Rol y Usuario. Un Rol agrupa múltiples permisos, y cada Usuario está vinculado a un Rol específico que determina las funcionalidades a las que tiene acceso dentro del sistema.

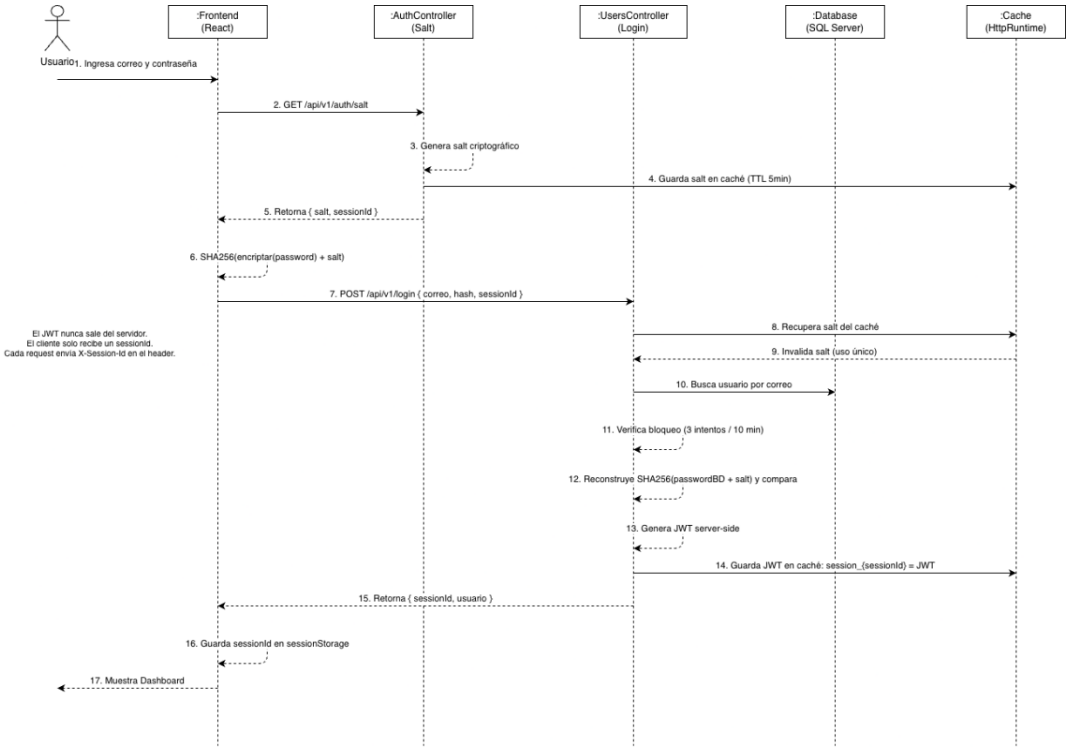
La gestión financiera se articula en tres flujos: los Ingresos, asociados a un Cliente y a un Presupuesto, que pueden generar documentos electrónicos y ajustar automáticamente los montos presupuestarios; los Gastos, vinculados a un Proveedor, que también impactan el presupuesto y permiten la aceptación de facturas electrónicas de compra mediante AceptarElectronicaFactura(); y el Presupuesto, que centraliza el control de montos aprobados, ejecutados y modificados por Centro de Costo, incluyendo validaciones de aprobación.

El DocumentoElectronico funciona como un intermediario que conecta tanto Ingresos como Gastos con el proceso de facturación electrónica de Hacienda, almacenando el tipo de documento, la clave numérica, el estado de validación y el XML del comprobante.

Las clases CuentaPorCobrar y CuentaPorPagar proporcionan el control de cartera mediante tablas auxiliares de antigüedad de saldos segmentadas en rangos de 0 a 120 días, con métodos para generar balances y ponderar los créditos vigentes. CuentaPorCobrar se alimenta de los Clientes y sus documentos, mientras que CuentaPorPagar se vincula a los Proveedores y sus gastos asociados.

Ilustración 53

Diagrama Secuencia Login

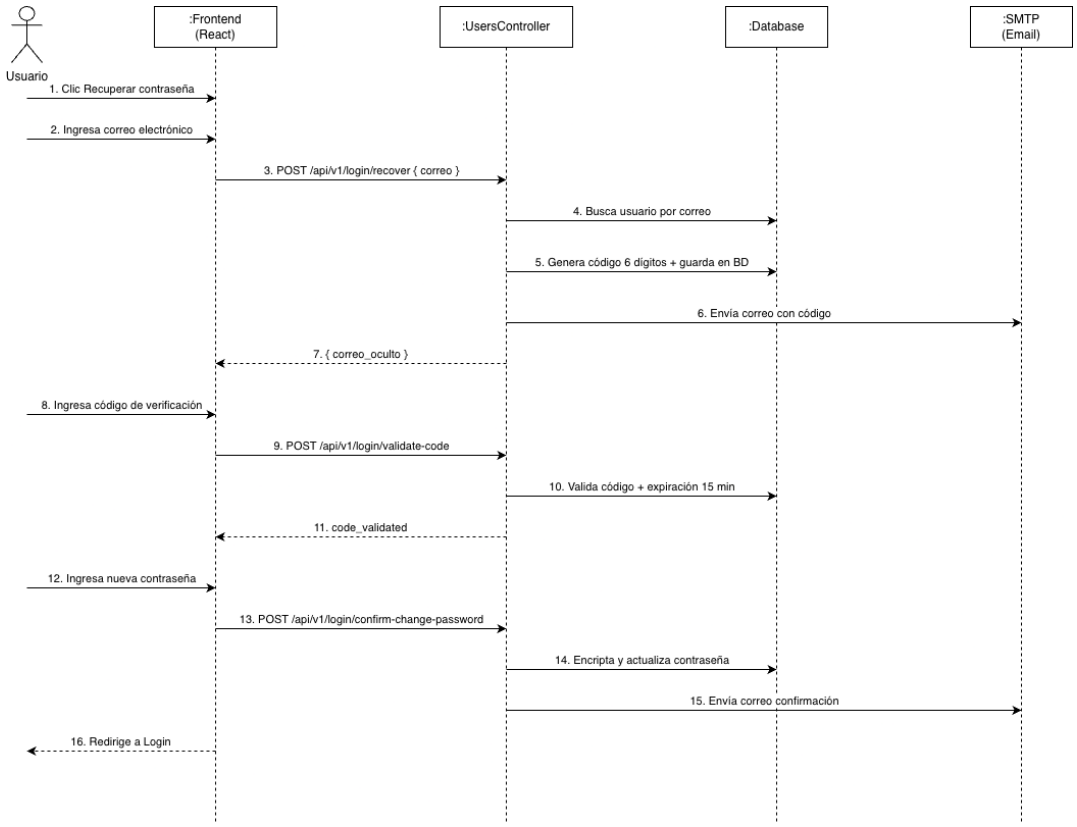


Fuente: Elaboración propia.

El diagrama de secuencia presentado anteriormente muestra el proceso lógico correspondiente al proceso de inicio de sesión en la aplicación. El flujo se inicia con la autenticación del usuario frente a la base de datos, donde se verifican las credenciales ingresadas. Una vez validada la identidad, el sistema procede a la creación de una sesión de tipo SALT, la cual consiste en la generación de un token gestionado mediante una cookie del lado del servidor. Paralelamente, se retorna al cliente un identificador de sesión, denominado *session ID*, el cual será el responsable de gestionar la autenticación y la validación de permisos en las solicitudes subsiguientes, garantizando así la seguridad e integridad de cada interacción entre el cliente y el servidor.

Ilustración 54

Diagrama Secuencia Recuperar Contraseña

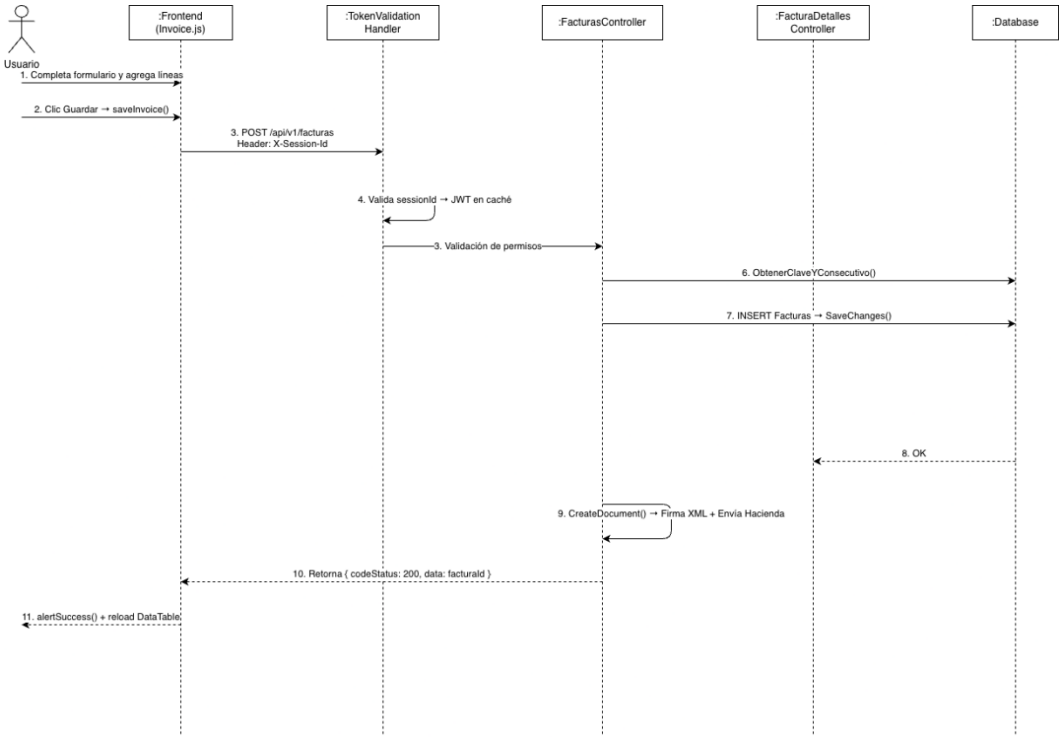


Fuente: Elaboración propia.

La figura anterior ilustra el diagrama de secuencia correspondiente al proceso de recuperación de contraseña. En primer lugar, el sistema valida la existencia del correo electrónico ingresado por el usuario en la base de datos; si el registro es encontrado, se genera y envía automáticamente un mensaje al correo asociado, el cual contiene un código de verificación. Posteriormente, el usuario deberá ingresar dicho código en el sistema para confirmar su identidad y continuar con el proceso de cambio de contraseña. Una vez que el proceso se completa de forma exitosa, el sistema redirige al usuario a la pantalla de inicio de sesión, desde donde podrá acceder a la aplicación con sus nuevas credenciales.

Ilustración 55

Diagrama Secuencia Facturas

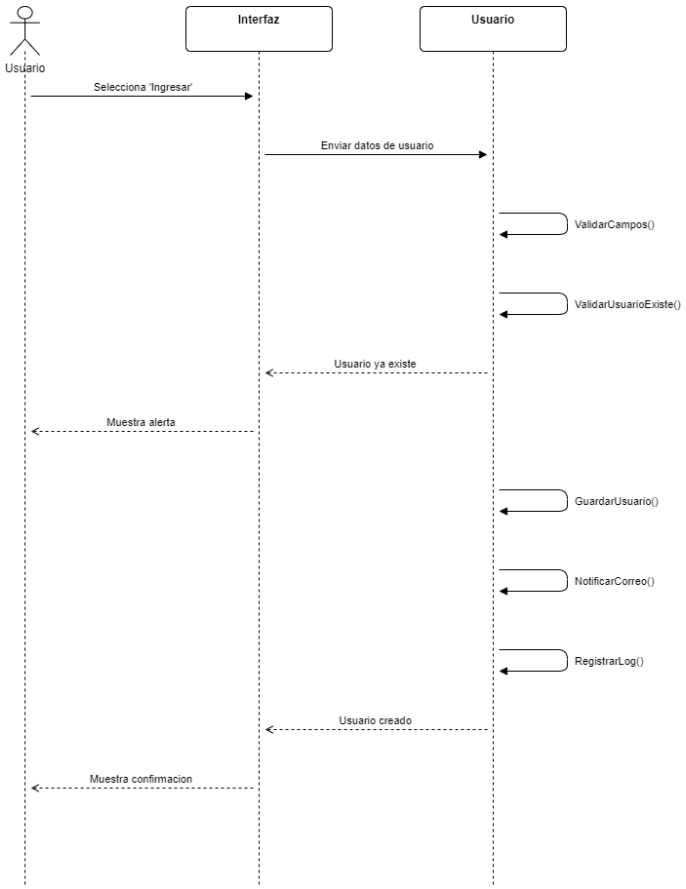


Fuente: Elaboración propia.

La ilustración 54 nos ilustra como será el proceso de creación de una factura, partiendo desde la interfaz en que se llena el formulario, pasando por los parámetros de autenticación del API donde se valida el SALT Session, hasta guardar el registro y crear los archivos XML correspondientes para su validación con el ente regulador.

Ilustración 56

Diagrama Secuencia Usuarios

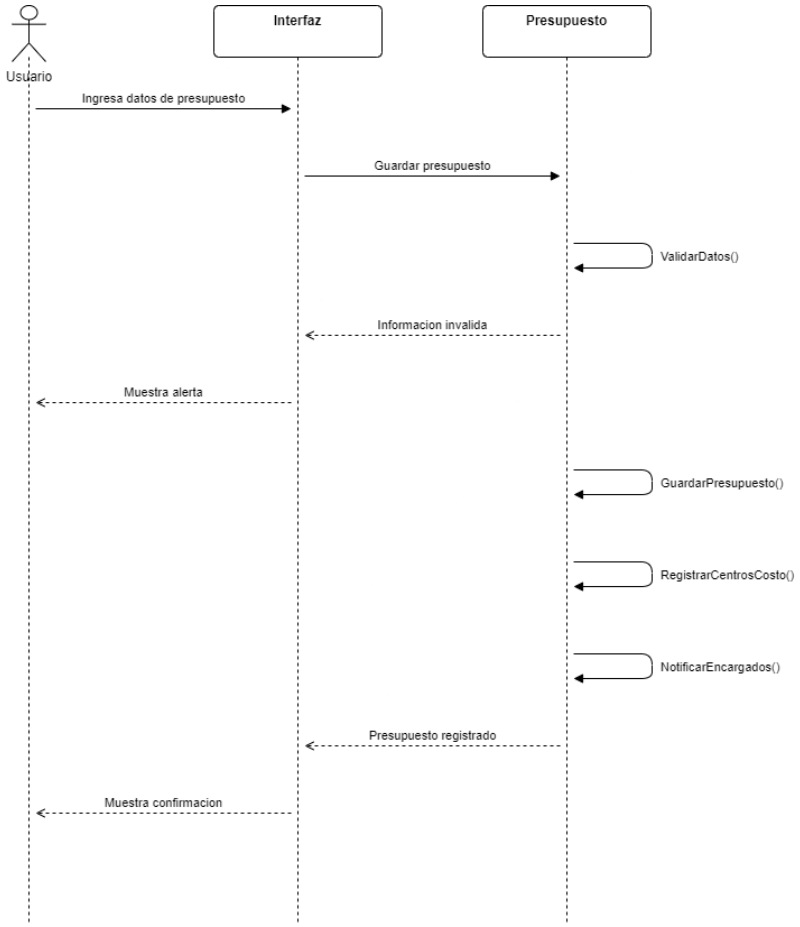


Fuente: Elaboración propia.

El diagrama de la figura anterior muestra cómo se define la secuencia del ciclo de vida de un usuario, el cual como es ingresado por un usuario ya existente previamente en el sistema validando los datos digitados con la función ValidaCampos() y validaUsuarioExiste(), en caso de ya existir se notifica que no puede ser creado, en caso de no ser así, se guarda el usuario y se notifica al usuario por medio de la función NotificarCorreo() y se muestra una alerta en el sistema indicando que el usuario fue creado satisfactoriamente.

Ilustración 57

Diagrama Secuencia Presupuesto

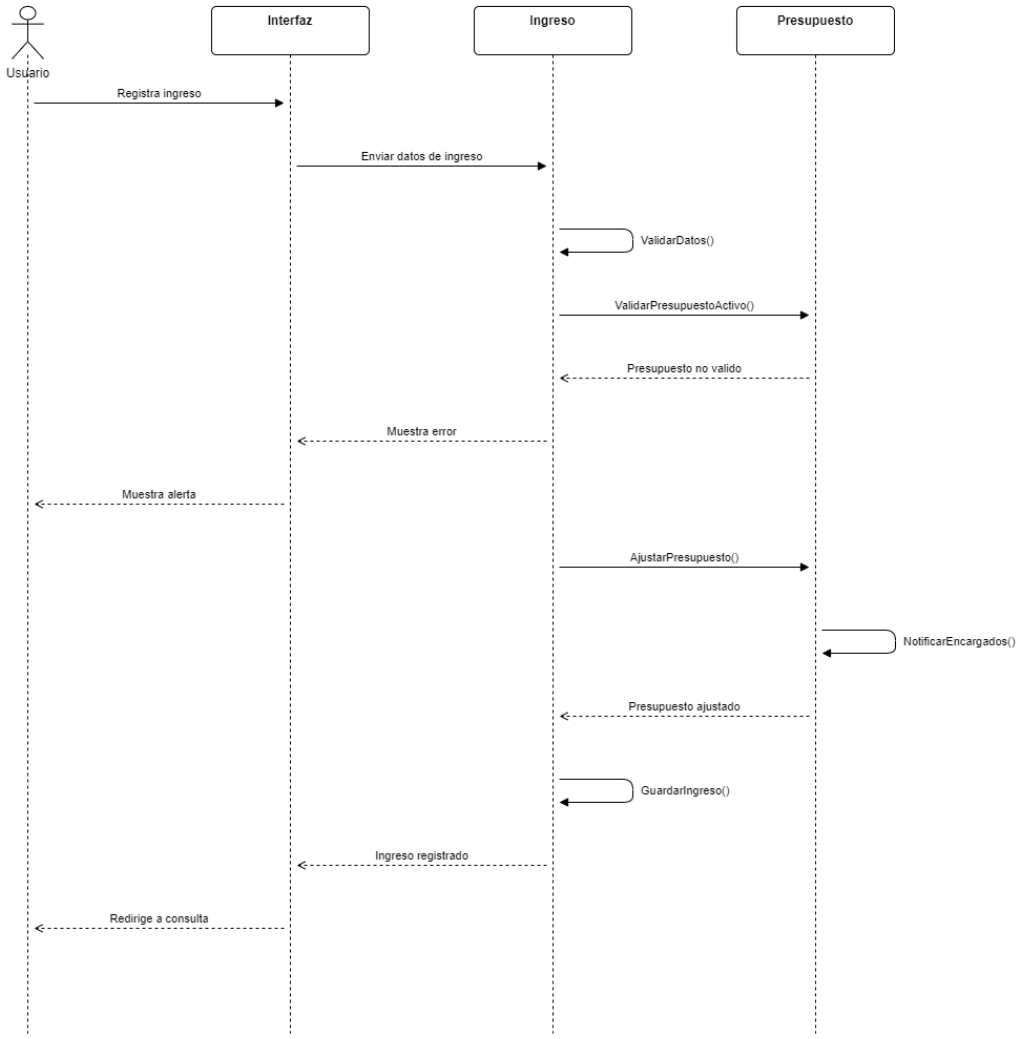


Fuente: Elaboración propia.

La figura anterior muestra como está definido el flujo de la creación de un presupuesto dentro del sistema, el proceso inicia cuando el usuario completa el formulario con los datos presupuestarios desde la interfaz y solicita su almacenamiento. La interfaz transmite la petición al componente Presupuesto, el cual ejecuta internamente el método ValidarDatos() para verificar la integridad y consistencia de la información recibida.

Ilustración 58

Diagrama Secuencia Ingresos.



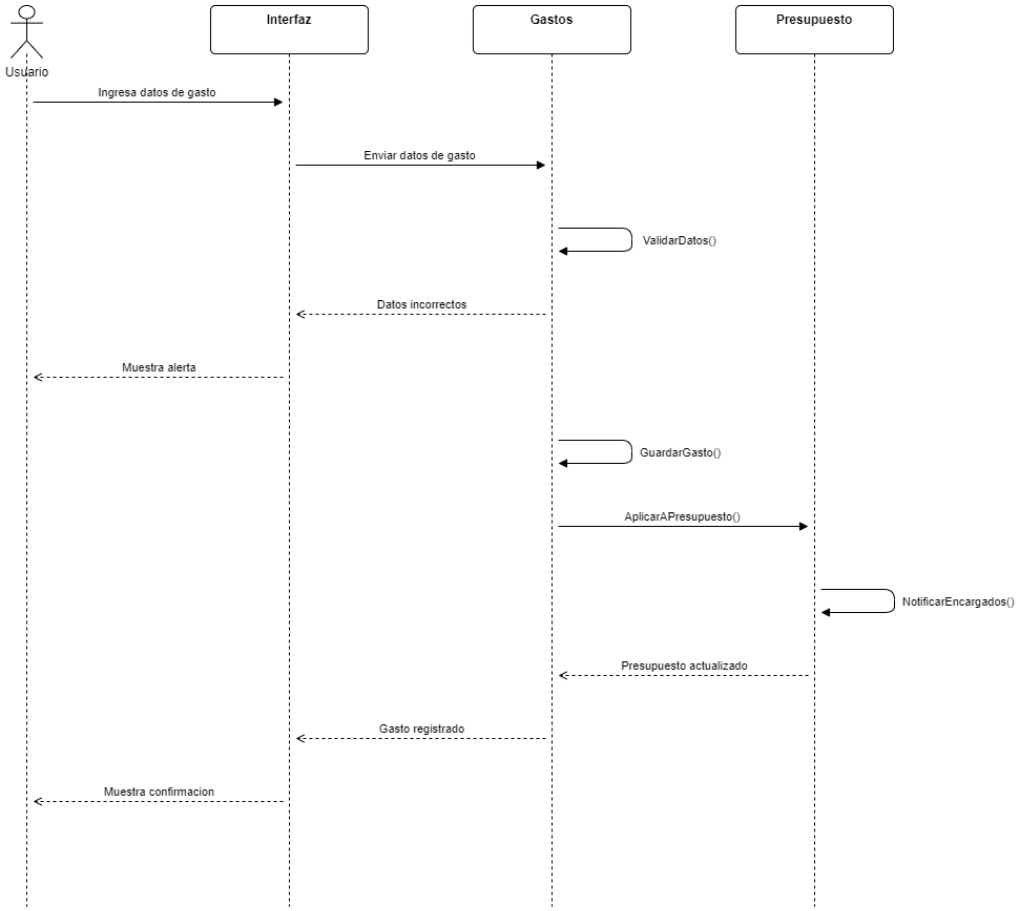
Fuente: Elaboración propia.

El diagrama de secuencia describe el proceso de registro de un ingreso y su interacción con el módulo de gestión presupuestaria. El flujo comienza cuando el usuario completa los datos del ingreso desde la interfaz, la cual envía la información al componente Ingreso para su procesamiento. Este componente ejecuta primeramente el método ValidarDatos() para verificar que los campos obligatorios y los formatos sean correctos, y a continuación invoca ValidarPresupuestoActivo() en el componente Presupuesto para confirmar que existe un presupuesto vigente al cual asociar el ingreso.

Cuando ambas validaciones resultan satisfactorias, el componente Ingreso procede a invocar el método AjustarPresupuesto() en el componente Presupuesto, el cual actualiza los montos ejecutados con el valor del nuevo ingreso y ejecuta internamente NotificarEncargados() para informar a los responsables presupuestarios sobre la modificación realizada. Una vez confirmado el ajuste presupuestario, el componente Ingreso ejecuta GuardarIngreso() para persistir el registro en la base de datos.

Ilustración 59

Diagrama de Secuencia Gastos



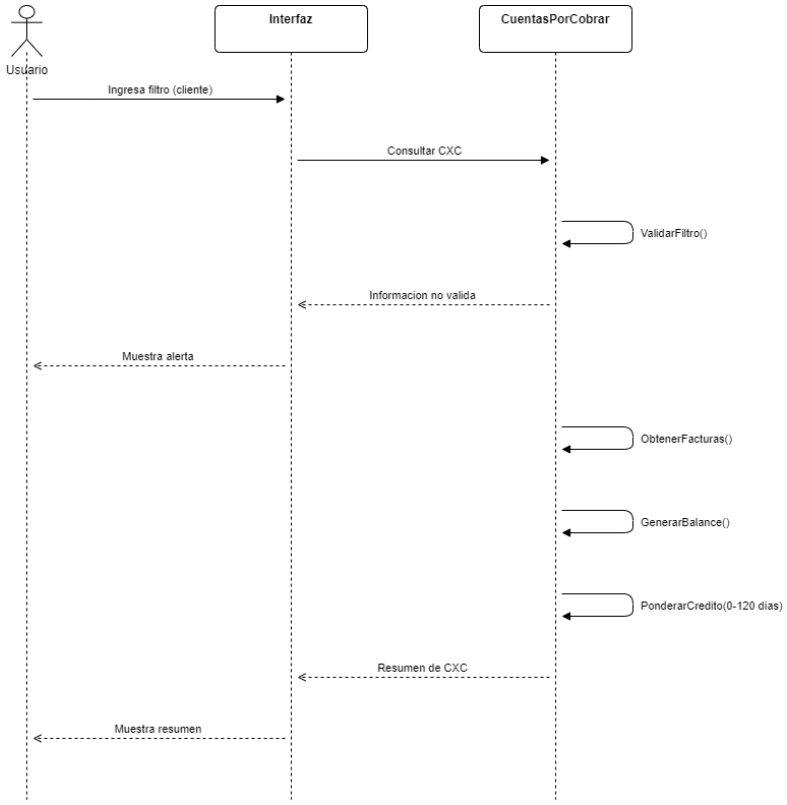
Fuente: Elaboración propia.

Basado en la ilustración anterior, el proceso da comienzo cuando se ingresa los datos del gasto desde la interfaz, la cual transmite la información al componente Gastos para su

procesamiento. Este componente ejecuta internamente el método ValidarDatos() para verificar que los campos requeridos cumplan con las reglas de formato y consistencia establecidas. Una vez superada la validación, el componente Gastos ejecuta GuardarGasto() para persistir el registro en la base de datos y, de forma inmediata, invoca el método AplicarAPresupuesto( ) en el componente Presupuesto para actualizar los montos ejecutados restando el valor del gasto registrado. El componente Presupuesto, a su vez, ejecuta internamente NotificarEncargados() para informar a los responsables presupuestarios sobre la variación realizada, y retorna la confirmación de presupuesto actualizado al componente Gastos.

Ilustración 60

Diagrama de Secuencia CXC



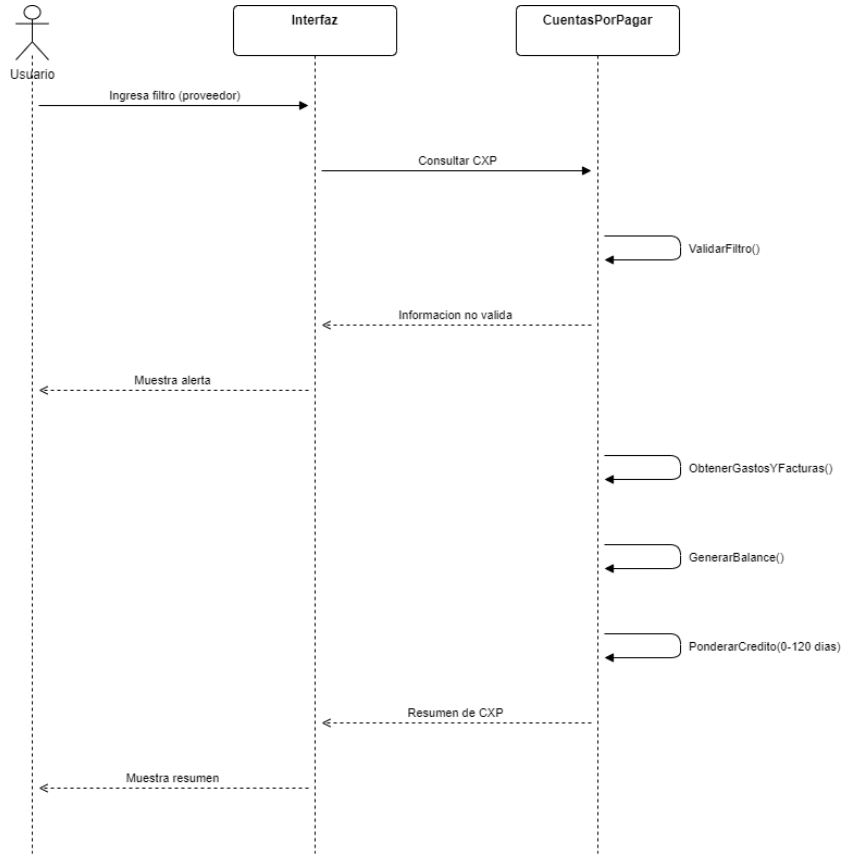
Fuente: Elaboración propia.

La imagen anterior muestra la secuencia de los informas de CXC, este componente ejecuta primeramente el método ValidarFiltro() para verificar que el criterio de búsqueda proporcionado sea válido y corresponda a un cliente existente en el sistema, cuando el componente es valido se ejecuta secuencialmente tres operaciones internas: ObtenerFacturas(), que recupera todas las

facturas pendientes de cobro asociadas al cliente consultado; GenerarBalance(), que calcula los saldos totales a partir de las facturas obtenidas diferenciando los montos vigentes de los vencidos; y PonderarCredito(0-120 días), que clasifica los saldos en una tabla auxiliar de antigüedad segmentada en intervalos de treinta días, abarcando desde los créditos corrientes hasta aquellos con más de ciento veinte días de vencimiento.

Ilustración 61

Diagrama de Secuencia CXP

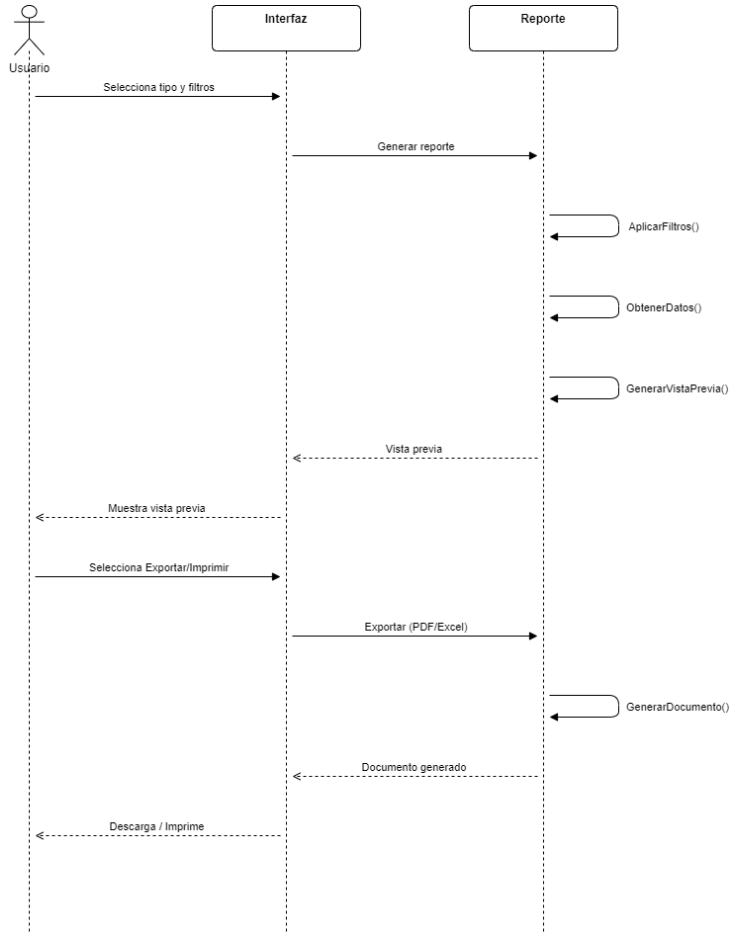


Fuente: Elaboración propia.

Con una funcionalidad similar al diagrama mostrado anteriormente de CXC, cuentas por pagar valida los filtros aplicados por el usuario, y obtiene la información de los gastos por medio de la función ObtenerGastosYFacturas(), generando a su vez el balance y ponderando los totales de cero a ciento veinte días.

Ilustración 62

Diagrama Secuencia Reportes



Fuente: Elaboración propia.

En la figura anterior se muestra la funcionalidad del proceso de reporte, dando inicio cuando el usuario selecciona el tipo de reporte deseado y configura los filtros de búsqueda desde la interfaz, tales como el rango de fechas, el estado de los documentos o el colaborador asociado. La interfaz transmite la solicitud al componente Reporte, el cual ejecuta secuencialmente tres operaciones internas: AplicarFiltros(), que establece los criterios de segmentación sobre el conjunto de datos disponible; ObtenerDatos(), que consulta la base de datos recuperando únicamente los registros que satisfacen los filtros aplicados, una vez obtenidos los datos se llama a GenerarVistaPrevia(), que construye una representación visual preliminar del reporte con los datos obtenidos.

Una vez que el usuario examina la información y decide materializar el reporte, selecciona la opción de exportar o imprimir desde la interfaz. Esta envía la solicitud de exportación al

componente Reporte especificando el formato de salida deseado, ya sea PDF o Excel. El componente ejecuta el método GenerarDocumento(), que transforma los datos del reporte en el formato seleccionado, y retorna el documento generado a la interfaz.

Programación

En este apartado se abordara el proceso de programación del sistema, tomando en cuenta todas las funciones a nivel de código usadas en las pantallas de entradas y salidas, partiendo del enunciado de que es la programación vs la codificación, según Feder (2026): "la programación informática se abre en una ventana nueva, implica juntar estas diferentes funciones (codificación) para crear un programa. Los programadores utilizan procesos específicos para compilar código, probarlo y producir un programa función" (párr.6). habiendo mencionado esto se abordará el proceso de programación del sistema en donde damos sentido a los enunciados anteriores como diagramas de flujo, UML, diagramas de base de datos y casos de uso, para concebir una aplicación funcional.

Entradas

En este apartado comprende el ingreso de información por parte del usuario, para este apartado comprende dos escenarios el front-end y el back-end, cuyos procesos se validan tanto en el front-end como en el back-end para asegurar la integridad de los datos que se vayan a guardar en la base de datos.

A nivel del front-end se maneja de forma similar, toda variable de input se almacena en el state de React, como se muestra en la siguiente imagen:

Ilustración 63

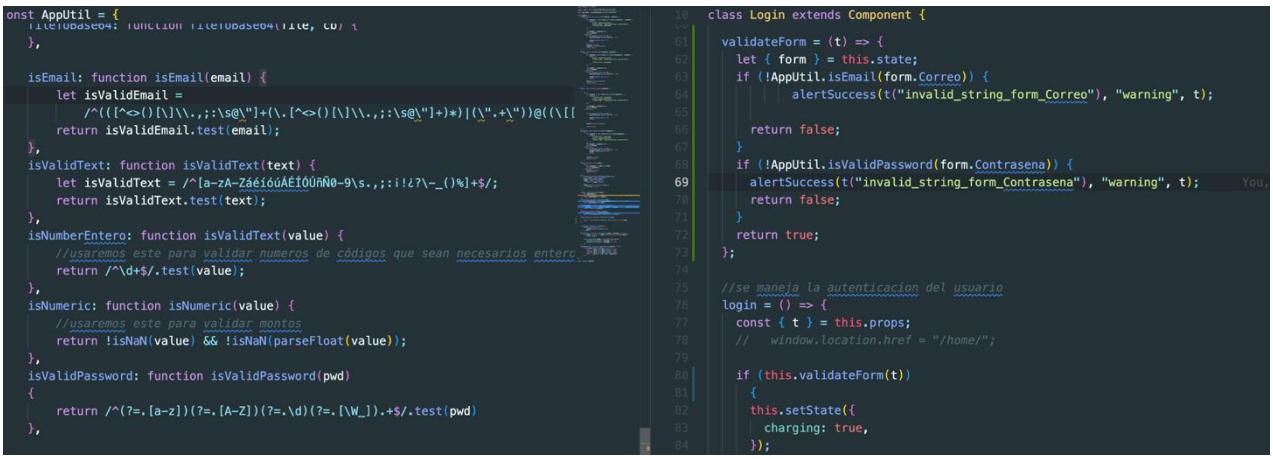
Variables Front-end

Fuente: Elaboración propia.

Una vez el usuario haya terminado de digitar la información y de clic en el botón que ejecutará la acción se llamará una función para la validación de los campos, por ejemplo la siguiente imagen muestra como se hace la validación de esta información, por medio de regex:

Ilustración 64

Validación campos



```
const AppUtil = {
  },

  isEmail: function isEmail(email) {
      let isValidEmail =
          /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[
      return isValidEmail.test(email);
  },
  isValidText: function isValidText(text) {
      let isValidText = /^[a-zA-ZáéíóúÁÉÍÓÚñÑ0-9\s.,;:¡!¿?\-_()%]+$/;
      return isValidText.test(text);
  },
  isNumberEntero: function isValidText(value) {
      //usaremos este para validar numeros de códigos que sean necesarios entero
      return /^\d+$/.test(value);
  },
  isNumeric: function isNumeric(value) {
      //usaremos este para validar montos
      return !isNaN(value) && !isNaN(parseFloat(value));
  },
  isValidPassword: function isValidPassword(pwd)
  {
      return /^(?=.[a-z])(?=.[A-Z])(?=.\d)(?=.[\W_]).+$/.test(pwd)
  },
```

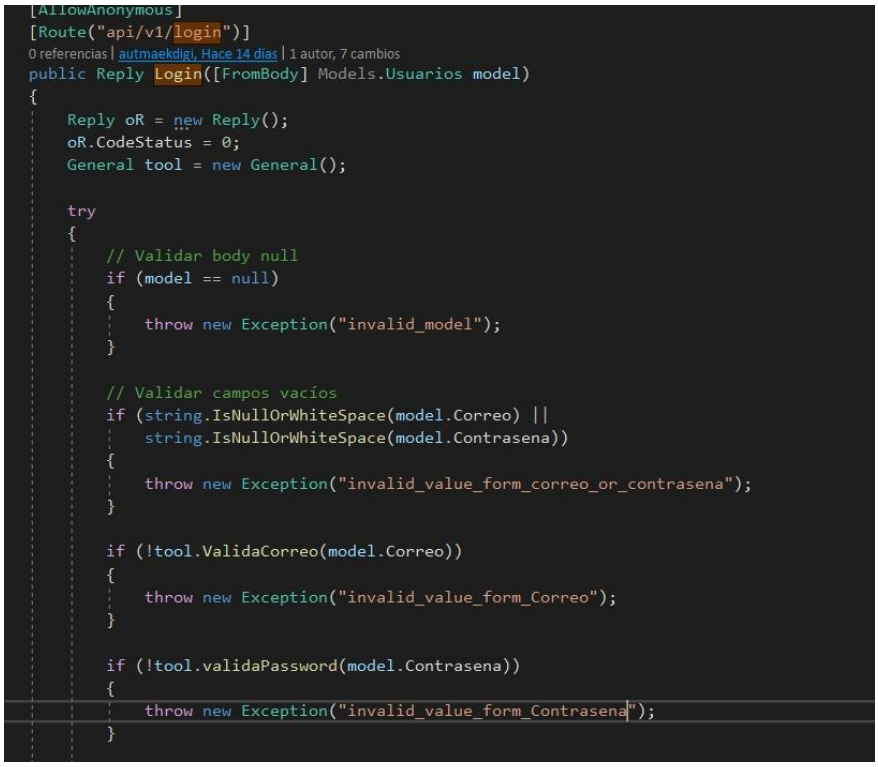
```
class Login extends Component {

  validateForm = (t) => {
    let { form } = this.state;
    if (!AppUtil.isEmail(form.Correo)) {
            alertSuccess(t("invalid_string_form_Correo"), "warning", t);

      return false;
    }
    if (!AppUtil.isValidPassword(form.Contrasena)) {
      alertSuccess(t("invalid_string_form_Contrasena"), "warning", t);
      return false;
    }
    return true;
  };

  //se maneja la autenticacion del usuario
  login = () => {
    const { t } = this.props;
    //   window.location.href = "/home/";

    if (this.validateForm(t))
      {
      this.setState({
        charging: true,
      });
```

Fuente: Elaboración propia.

Una vez validada la información se envía por medio de una solicitud de HTTPS al servidor back-end, en donde se captura, y se válida para confirmar la veracidad de la información antes de hacer algún tipo de proceso con la información enviada.

Ilustración 65

Validación Back-End



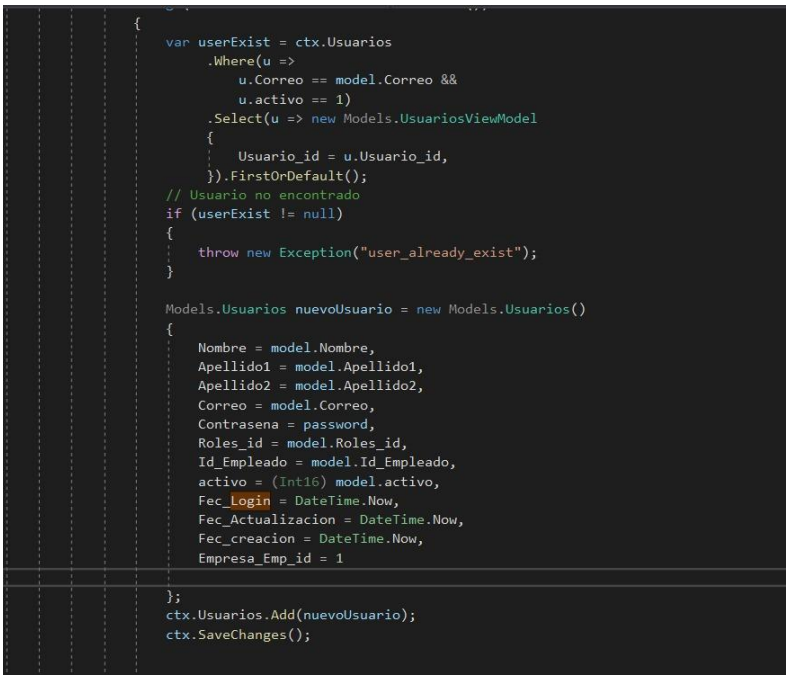
```
[AllowAnonymous]
[Route("api/v1/login")]
0 referencias | autmaekdigi, Hace 14 días | 1 autor, 7 cambios
public Reply Login([FromBody] Models.Usuarios model)
{
    Reply oR = new Reply();
    oR.CodeStatus = 0;
    General tool = new General();

    try
    {
        // Validar body null
        if (model == null)
        {
            throw new Exception("invalid_model");
        }

        // Validar campos vacíos
        if (string.IsNullOrWhiteSpace(model.Correo) ||
            string.IsNullOrWhiteSpace(model.Contrasena))
        {
            throw new Exception("invalid_value_form_correo_or_contrasena");
        }

        if (!tool.ValidaCorreo(model.Correo))
        {
            throw new Exception("invalid_value_form_Correo");
        }

        if (!tool.validaPassword(model.Contrasena))
        {
            throw new Exception("invalid_value_form_Contrasena");
        }
```

Fuente: Elaboración propia.

En esta imagen podemos aprecia la validación de los campos enviados desde el front-end del lado del servidor, donde por medio de un regex se valida que el correo tenga un formato valido así como la contraseña posea un formato alfanumérico y contenga símbolos especiales, habiendo sanitizado la información se procede a almacenar en la base de datos, hacer consultas o recurrir a procesos más complejos.

Ilustración 66

Creación de Usuario.



```
{
    var userExist = ctx.Usuarios
         .Where(u =>
             u.Correo == model.Correo &&
             u.activo == 1)
         .Select(u => new Models.UsuariosViewModel
         {
             Usuario_id = u.Usuario_id,
         }).FirstOrDefault();
    // Usuario no encontrado
    if (userExist != null)
    {
        throw new Exception("user_already_exist");
    }

    Models.Usuarios nuevoUsuario = new Models.Usuarios()
    {
        Nombre = model.Nombre,
        Apellido1 = model.Apellido1,
        Apellido2 = model.Apellido2,
        Correo = model.Correo,
        Contrasena = password,
        Roles_id = model.Roles_id,
        Id_Empleado = model.Id_Empleado,
        activo = (Int16) model.activo,
        Fec_Login = DateTime.Now,
        Fec_Actualizacion = DateTime.Now,
        Fec_creacion = DateTime.Now,
        Empresa_Emp_id = 1

    };
    ctx.Usuarios.Add(nuevoUsuario);
    ctx.SaveChanges();
```

Fuente: Elaboración propia.

Como se puede observar en la imagen una vez validada la información esta es almacenada en la base de datos o usada en procesos que requieran mayor complejidad, como lo es por ejemplo el proceso de inicio de sesión la cual toma el correo, valida los permisos del usuario y genera un token en caché para mantener viva su sesión.

Salidas

En este apartado se abordara como se realiza el proceso de obtención de información partiendo desde el request al front-end hacia el servidor, como el servidor responde y como el front-end recibe y procesa esta información, a nivel del front-end el evento de obtención de información puede ser por medio de un evento que llama al back-end o al cargar la página.

Ilustración 67

Obtener Información

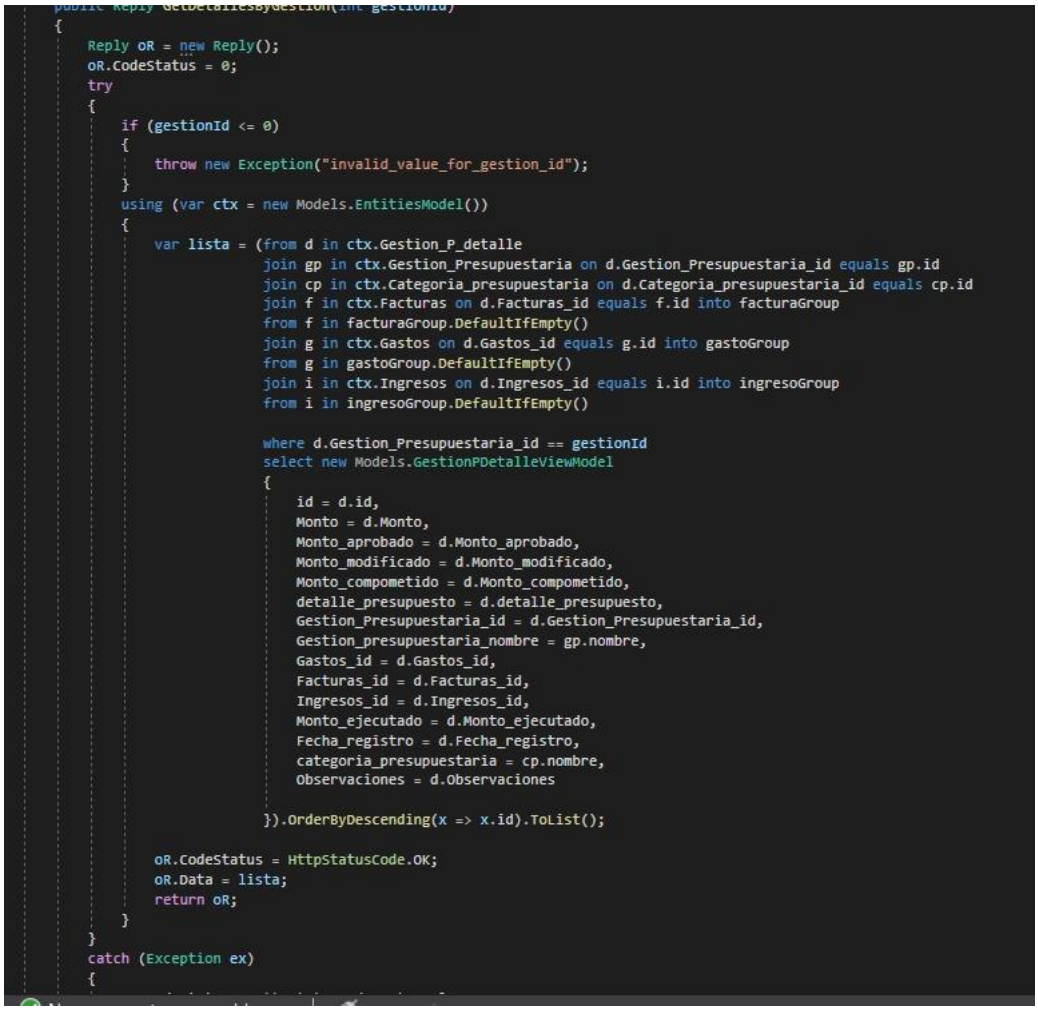
```
componentDidMount() {
        this.getUserInfo();
    const { gestionID } = this.props;
    if (gestionID) {
        this.getMovements(gestionID);
    }
}

getMovements = (gestionID) => {
    AppUtil.getAPI(`gestion_p_detalle/gestion/${gestionID}`).then((response) => {
        let movements = response ? response.data : [];


        this.setState({ movements });
    });
};
```

Fuente: Elaboración propia.

En este apartado se ve como se invoca a la función para obtener los movimientos al endpoint de gestión_p_detalle, para los endpoints es necesario un permiso para ejecutar las acciones, el cual depende del rol del usuario, en caso de no poseerlo no podrá ejecutar un llamado al endpoint, a continuación se muestra un llamado a una función de obtención de información:

Ilustración 68

Obtener Detalle Gestión Presupuestaria.



```
public Reply GetDetallesbyGestion(int gestionId)
{
    Reply oR = new Reply();
    oR.CodeStatus = 0;
    try
    {
        if (gestionId <= 0)
        {
            throw new Exception("invalid_value_for_gestion_id");
        }
        using (var ctx = new Models.EntitiesModel())
        {
            var lista = (from d in ctx.Gestion_P_detalle
                         join gp in ctx.Gestion_Presupuestaria on d.Gestion_Presupuestaria_id equals gp.id
                         join cp in ctx.Categoria_presupuestaria on d.Categoria_presupuestaria_id equals cp.id
                         join f in ctx.Facturas on d.Facturas_id equals f.id into facturaGroup
                         from f in facturaGroup.DefaultIfEmpty()
                         join g in ctx.Gastos on d.Gastos_id equals g.id into gastoGroup
                         from g in gastoGroup.DefaultIfEmpty()
                         join i in ctx.Ingresos on d.Ingresos_id equals i.id into ingresoGroup
                         from i in ingresoGroup.DefaultIfEmpty()

                         where d.Gestion_Presupuestaria_id == gestionId
                         select new Models.GestionPDetalleViewModel
                         {
                             id = d.id,
                             Monto = d.Monto,
                             Monto_aprobado = d.Monto_aprobado,
                             Monto_modificado = d.Monto_modificado,
                             Monto_compometido = d.Monto_compometido,
                             detalle_presupuesto = d.detalle_presupuesto,
                             Gestion_Presupuestaria_id = d.Gestion_Presupuestaria_id,
                             Gestion_presupuestaria_nombre = gp.nombre,
                             Gastos_id = d.Gastos_id,
                             Facturas_id = d.Facturas_id,
                             Ingresos_id = d.Ingresos_id,
                             Monto_ejecutado = d.Monto_ejecutado,
                             Fecha_registro = d.Fecha_registro,
                             categoria_presupuestaria = cp.nombre,
                             Observaciones = d.Observaciones

                         }).OrderByDescending(x => x.id).ToList();

            oR.CodeStatus = HttpStatusCode.OK;
            oR.Data = lista;
            return oR;
        }
    }
    catch (Exception ex)
    {
```

Fuente: Elaboración propia.

En esta función se aprecia que en base al id se obtiene la información para ser mostrada al usuario, en las cabeceras del endpoint se validan los permisos y que el usuario posea una sesión activa, la información es devuelta al front-end, en donde es capturada para ser mostrada en una tabla o un detalle de los datos, para ser vista o editada, a continuación se muestra cómo se procesa los datos en una tabla:

Ilustración 69

Renderización de Datos

```
<Row>
    <DataTable
        ref={this.datatableRef}
        data={movements}
        columns={[
            { data: "id",                    title: t("id") },
            { data: "monto",                 title: t("amount") },
            { data: "detalle_presupuesto",   title: t("detail") },
            { data: "categoria_presupuestaria",title: t("category") },
            { data: "gastos_id",             title: t("spent") },
            { data: "facturas_id",           title: t("invoice") },
            { data: "ingresos_id",           title: t("income") },
            { data: "monto_ejecutado",       title: t("amount_executed") },
            {
                data: "fecha_registro",
                title: t("date"),
                render: (data) =>
                    data ? moment(data).format(this.user.formatoFecha.toUpperCase()) : "",
            },
            { data: "observaciones",         title: t("detail") },
        ]}
        className="display table cell-border compact stripe"
        options={{
            language: {
                zeroRecords:       t("zeroRecords"),
                emptyTable:        t("emptyTable"),
                search:            t("search"),
                paginate:          t("paginate"),
                searchPlaceholder: t("searchPlaceholder"),
                info:              t("info"),
                lengthMenu:        t("lengthMenu"),
            },
            layout: {
                topStart:    "pageLength",
                topEnd:      "search",
                bottomStart: "info",
                bottomEnd:   "paging",
            },
        }}
    />
</Row>
```

Fuente: Elaboración propia.

Como se puede apreciar en la figura anterior, una vez obtenida los datos solicitados se hace uso de la librería Datatable para renderizar la información, indicando los nombres a usar, el titulo que tendrá y en caso de ser necesario un renderizado adicional para ciertas columnas, por ejemplo en la figura anterior se renderiza la fecha en base a los ajustes de formato de fecha definidos para la empresa, esto se obtiene al hacer inicio de sesión en la aplicación.

Procesos

La aplicación posee un conjunto de procesos para los cálculos para los totales de ingresos, egresos, y facturas, las cuales a su vez comparados con los presupuestos vigentes a la fecha de creación de cada movimiento, en primera instancia se tiene por ejemplo los cálculos de detalles, los cuales se toman en cuanto el usuario ingresa la información, por ejemplo en la siguiente función se puede observar como el cambio de información altera los totales de un auxiliar antes de ser ingresados como detalle de una factura:

Ilustración 70

Calculo Previo a Agregar Línea.

```
_calculaInput = (e, isSelect = false) => {
    let { name, value, selectedIndex } = e.target;

    if (isSelect) {
        const optionElement = e.target.options[selectedIndex];
        const taxOption = optionElement.getAttribute("attr");
        name = "porcentaje";
        value = taxOption;

    }

    this.setState({ AuxLine: { ...this.state.AuxLine, [name]: value } }, () => {
        let { AuxLine } = this.state;


        let subtotal  = isNaN(AuxLine.subtotal) || AuxLine.subtotal ===""  ? 0 : parseFloat(AuxLine.subtotal);
        let tax       = isNaN(AuxLine.porcentaje) || AuxLine.porcentaje ===""  ? 0 : parseFloat(AuxLine.porcentaje);
        let cantidad  = isNaN(AuxLine.cantidad) || AuxLine.cantidad ===""  ? 1 : parseFloat(AuxLine.cantidad);
        let descuento = isNaN(AuxLine.descuento) || AuxLine.descuento === ""  ? 0 : parseFloat(AuxLine.descuento);
 autmaekdigi, 2 weeks ago • inicio facturas
        let impuesto = ((subtotal * cantidad) - descuento) * (tax / 100);
        let total    = ((subtotal * cantidad) - descuento) + impuesto;


        AuxLine.impuesto = impuesto;
        AuxLine.total    = total;
        this.setState({ AuxLine });
    });
};
```

Fuente: Elaboración propia.

Habiendo definido los datos por agregar a la línea se invoca la función recalcularTotales() para agregar información de la línea y sumar los montos ingresados por el usuario y calcular los totales de la factura.

Ilustración 71

Recalcular Totales

```
_recalcularTotales = () => {
    const { lines } = this.state;
    const subtotal  = lines.reduce((acc, l) => acc + l.subtotal,  0);      autmaekdigi,
    const impuesto  = lines.reduce((acc, l) => acc + l.impuesto,  0);
    const total     = lines.reduce((acc, l) => acc + l.total,     0);
    const descuento = lines.reduce((acc, l) => acc + l.descuento, 0);

    this.setState((prevState) => ({
        invoice: { ...prevState.invoice, subtotal, impuesto, total, descuento },
    }));
};
```

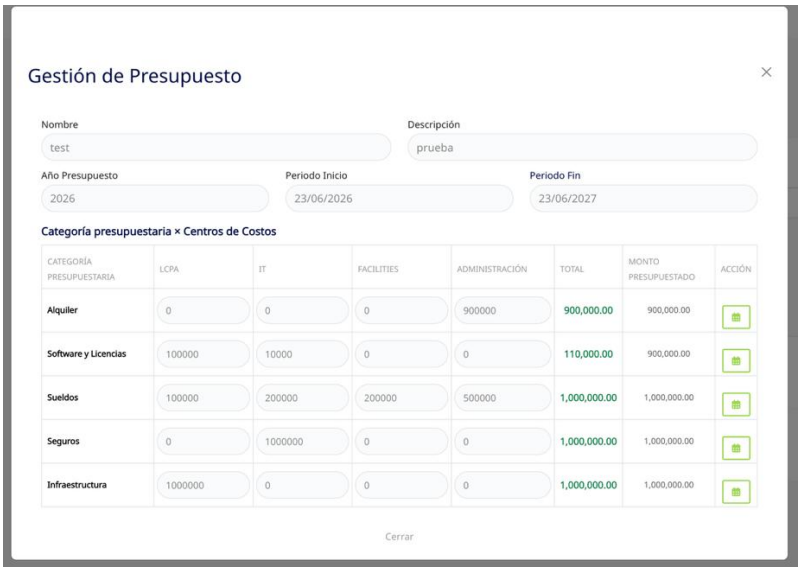
Fuente: Elaboración propia.

Todos los cálculos asociados a los detalles de movimientos financieros, tales como facturas, gastos e ingresos, son procesados directamente en el front-end. Esta decisión de diseño garantiza la consistencia entre los valores visualizados por el usuario y los datos que finalmente se persisten en la base de datos, eliminando discrepancias derivadas de recálculos en capas intermedias.

Otro proceso relevante dentro del sistema es la carga de la matriz de presupuestos. Dicho proceso se ejecuta de forma dinámica, construyendo la matriz en función de los centros de costos registrados y las categorías de presupuesto configuradas, tal como se ilustra a continuación:

Ilustración 72

Gestión de Presupuesto.



Fuente: Elaboración propia.

Ilustración 73

Gestión de Presupuesto Calculo de Límites.

```
{presupuestary_category?.map(cat => {
  const rowTotal = (cost_center || []).reduce((sum, cc) => (
    sum + (parseFloat(this.state.budgetMatrix[`${cat.id}_${cc.id}`]) || 0)
  ), 0);
  const limit = parseFloat(cat.monto_presupuestado || 0);
  const exceeded = limit > 0 && rowTotal > limit;
  return (
    <tr key={cat.id}>
      <td className="fw-semibold small">{cat.nombre}</td>
      {cost_center?.map(cc => (
        <td key={cc.id} className="p-1">
          <Form.Control
            type="number"
            size="sm"
            min={0}
            step="0.01"
            disabled={isView}
            required
            value={this.state.budgetMatrix[`${cat.id}_${cc.id}`] ?? ""}
            onChange={(e) => this._saveMatrixCell(cat.id, cc.id, e.target.value)}
          />
        </td>
      ))}
      <td classNam (property) formatNumber: (number: any, isInteger?: boolean) => string 8754' }}>
        { AppUtil.formatNumber(rowTotal.toFixed(2))}      You, yesterday • presupiuesto por mes …
      </td>
      <td className="text-center text-muted small">
        {AppUtil.formatNumber(limit.toFixed(2))}
      </td>
```

Fuente: Elaboración propia.

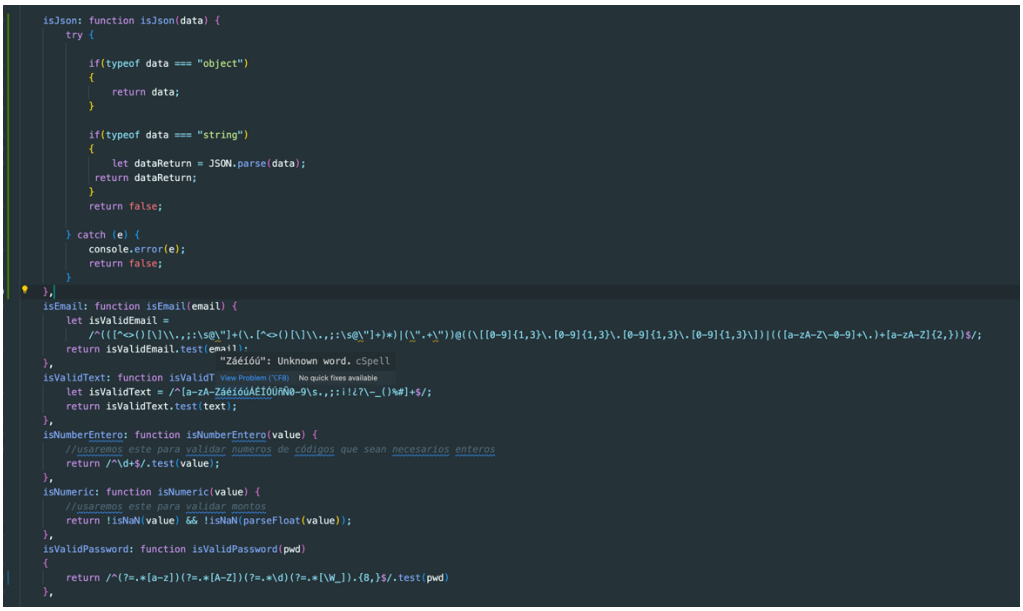
Las figuras anteriores demuestran el comportamiento dinámico del sistema al momento de gestionar el ingreso de presupuestos. El sistema genera automáticamente un formulario adaptado a las categorías disponibles, y aplica validaciones en tiempo real sobre los montos ingresados, garantizando que ningún valor exceda el límite máximo establecido para cada categoría.

Validaciones

Como se mencionó anteriormente la validación de información se hace tanto en el back-end como en el front-end, las validaciones hechas en ambas capas de software son hechas por medio de regex, validando textos válidos, que no posean caracteres que puedan corromper la estructura de base de datos, así como números para información numérica, correos, y contraseñas.

Ilustración 74

Validación de Información.



```
isJson: function isJson(data) {
    try {

        if(typeof data === "object")
        {
            return data;
        }

        if(typeof data === "string")
        {
            let dataReturn = JSON.parse(data);
         return dataReturn;
        }
        return false;

    } catch (e) {
        console.error(e);
        return false;
    }
},
isEmail: function isEmail(email) {
    let isValidEmail =
        /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
    return isValidEmail.test(email);
},
isValidText: function isValidT View Problem (⌥F8)  No quick fixes available
    let isValidText = /^[a-zA-ZáéíóúÁÉÍÓÚñÑ0-9\s.,;:¡!¿?\-_()%#]+$/;
    return isValidText.test(text);
},
isNumberEntero: function isNumberEntero(value) {
    //usaremos este para validar numeros de códigos que sean necesarios enteros
    return /^\d+$/.test(value);
},
isNumeric: function isNumeric(value) {
    //usaremos este para validar montos
    return !isNaN(value) && !isNaN(parseFloat(value));
},
isValidPassword: function isValidPassword(pwd)
{
    return /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[\W_]).{8,}$/.test(pwd)
},
```

"ZáéíóÚ": Unknown word. cSpell

Fuente: Elaboración propia.

Módulos Señalados en el Alcance.

Cuentas Contables (Cuentas por Cobrar y Cuentas por Pagar)

Con el fin llevar una eficiencia grafica de y de funcionalidad se manejara estos módulos en una sola opción de menú, localizando todo en la opción de menú llamada Cuentas, el cual nos permite el seguimiento de las cuentas por cobrar y pagar. Debido a la magnitud del proyecto y acciones que ejecuta una sola clase se mostrará un vistazo a las secciones y como estas manejan el módulo.

Ilustración 75

Módulo Cuentas por Cobrar y Pagar

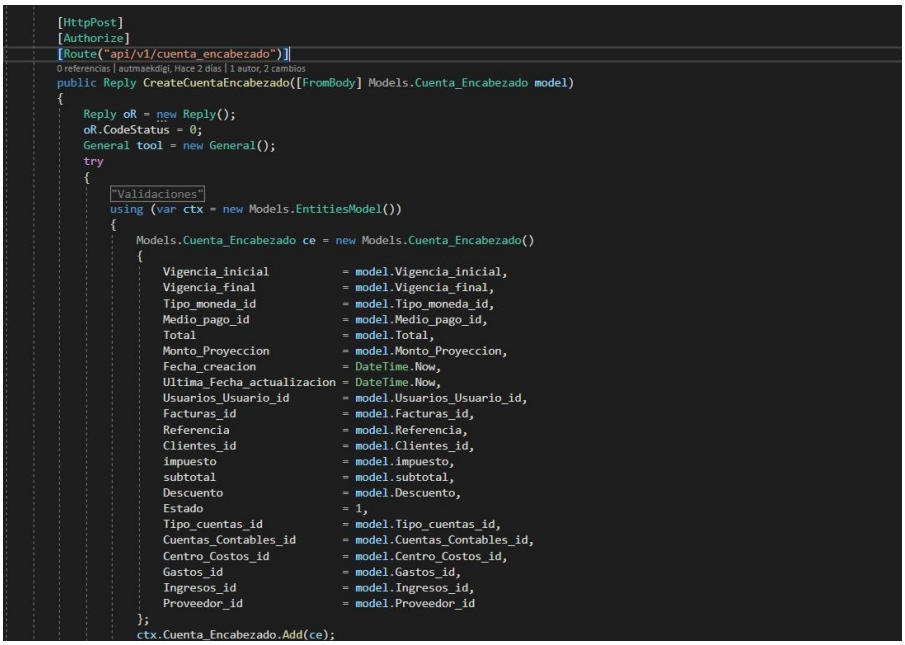
```
You, 12 minutes ago | 2 authors (autmaekdigi and one other)
class Accounts extends Component {
  constructor(props) {
    super(props);
    this.user = null;
    this.datatableRef = createRef();

    this.state = {…
    };
  }
// funciones de limpieza de campos
  _emptyEncabezado() {…
  }
  _emptyPago() {…
  }
  //  arranque
  componentDidMount() {
    this.getUserInfo();
    this.getCuentas();
  }      autmaekdigi, 2 days ago • cuentas contables
  getUserInfo = () => {…
  };

  // catálogos
  _loadCatalogs = () => {…
  };

  _loadPaymentCatalogs = () => {…
  };
  // API — Cuenta_Encabezado
  getCuentas = (tipoCuentaId = "") => {…
  };
  getCuentaById = (id, isView = false) =>…
    });
  openDetalle = (id) =>…
    });
  saveCuentaEncabezado = (e) => {…
  };
  deleteCuentaEncabezado = async (id) => {…
  };
  //  API — Cuenta_Detalle (pagos)
  openPagoModal = (encabezadoId) => {…
```

Fuente: Elaboración propia.

Para el manejo de toda la información se hace uso de dos controller llamados cuentas encabezado y cuentas detalle el cual maneja por medio de peticiones HTTP la acción a ejecutar con la información enviada, por ejemplo en la siguiente imagen se puede apreciar el proceso posterior a la ejecución de guardado del encabezado de cuentas:

Ilustración 76

Módulos Cuenta (Encabezado)



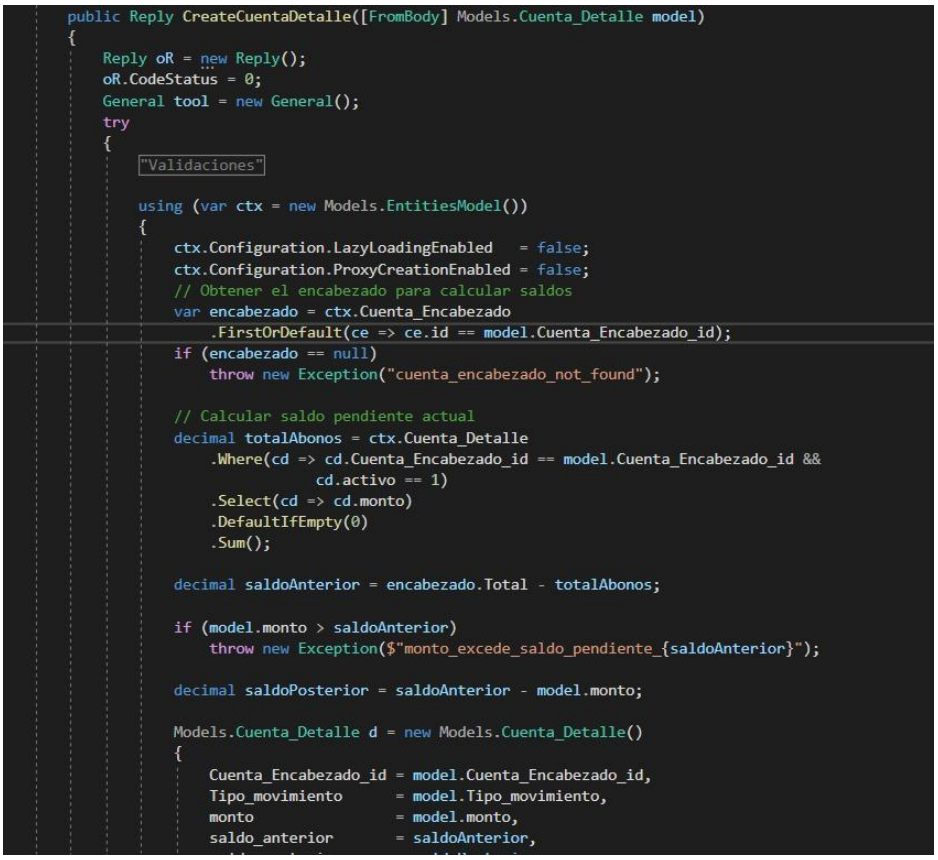
```
[HttpPost]
[Authorize]
[Route("api/v1/cuenta_encabezado")]
0 referencias | autmaekdigi, Hace 2 días | 1 autor, 2 cambios
public Reply CreateCuentaEncabezado([FromBody] Models.Cuenta_Encabezado model)
{
    Reply oR = new Reply();
    oR.CodeStatus = 0;
    General tool = new General();
    try
    {
        "Validaciones"
        using (var ctx = new Models.EntitiesModel())
        {
            Models.Cuenta_Encabezado ce = new Models.Cuenta_Encabezado()
            {
                Vigencia_inicial           = model.Vigencia_inicial,
                Vigencia_final             = model.Vigencia_final,
                Tipo_moneda_id             = model.Tipo_moneda_id,
                Medio_pago_id              = model.Medio_pago_id,
                Total                      = model.Total,
                Monto_Proyeccion           = model.Monto_Proyeccion,
                Fecha_creacion             = DateTime.Now,
                Ultima_Fecha_actualizacion = DateTime.Now,
                Usuarios_Usuario_id        = model.Usuarios_Usuario_id,
                Facturas_id                = model.Facturas_id,
                Referencia                 = model.Referencia,
                Clientes_id                = model.Clientes_id,
                impuesto                   = model.impuesto,
                subtotal                   = model.subtotal,
                Descuento                  = model.Descuento,
                Estado                     = 1,
                Tipo_cuentas_id            = model.Tipo_cuentas_id,
                Cuentas_Contables_id       = model.Cuentas_Contables_id,
                Centro_Costos_id           = model.Centro_Costos_id,
                Gastos_id                  = model.Gastos_id,
                Ingresos_id                = model.Ingresos_id,
                Proveedor_id               = model.Proveedor_id
            };
            ctx.Cuenta_Encabezado.Add(ce);
```

Fuente: Elaboración propia.

Una vez se tiene un encabezado definido se puede proceder con la creación de detalles, o pagos a la cuenta por cobrar o pagar, a continuación, se muestra el código de creación de estos registros.

Ilustración 77

Módulo Cuentas (Detalles)



```
public Reply CreateCuentaDetalle([FromBody] Models.Cuenta_Detalle model)
{
    Reply oR = new Reply();
    oR.CodeStatus = 0;
    General tool = new General();
    try
    {
        "Validaciones"

        using (var ctx = new Models.EntitiesModel())
        {
            ctx.Configuration.LazyLoadingEnabled   = false;
            ctx.Configuration.ProxyCreationEnabled = false;
            // Obtener el encabezado para calcular saldos
            var encabezado = ctx.Cuenta_Encabezado
                .FirstOrDefault(ce => ce.id == model.Cuenta_Encabezado_id);
            if (encabezado == null)
                throw new Exception("cuenta_encabezado_not_found");

            // Calcular saldo pendiente actual
            decimal totalAbonos = ctx.Cuenta_Detalle
                .Where(cd => cd.Cuenta_Encabezado_id == model.Cuenta_Encabezado_id &&
                             cd.activo == 1)
                .Select(cd => cd.monto)
                .DefaultIfEmpty(0)
                .Sum();

            decimal saldoAnterior = encabezado.Total - totalAbonos;

            if (model.monto > saldoAnterior)
                throw new Exception($"monto_excede_saldo_pendiente_{saldoAnterior}");

            decimal saldoPosterior = saldoAnterior - model.monto;

            Models.Cuenta_Detalle d = new Models.Cuenta_Detalle()
            {
                Cuenta_Encabezado_id = model.Cuenta_Encabezado_id,
                Tipo_movimiento      = model.Tipo_movimiento,
                monto                = model.monto,
                saldo_anterior       = saldoAnterior,
```

Fuente: Elaboración propia.

Gastos

El módulo de gastos comprende un estructura a nivel de front-end muy similar a la de otros módulos, en la cual se realiza los cálculos de los detalles y se envía todo para ser almacenado y procesado en el back-end. Para este proceso hacemos uso de una función la cual se encarga de tomar la información conforme el usuario va cambiando campos de texto para la realización de los cálculos.

Ilustración 78

Módulo Gastos Cálculo

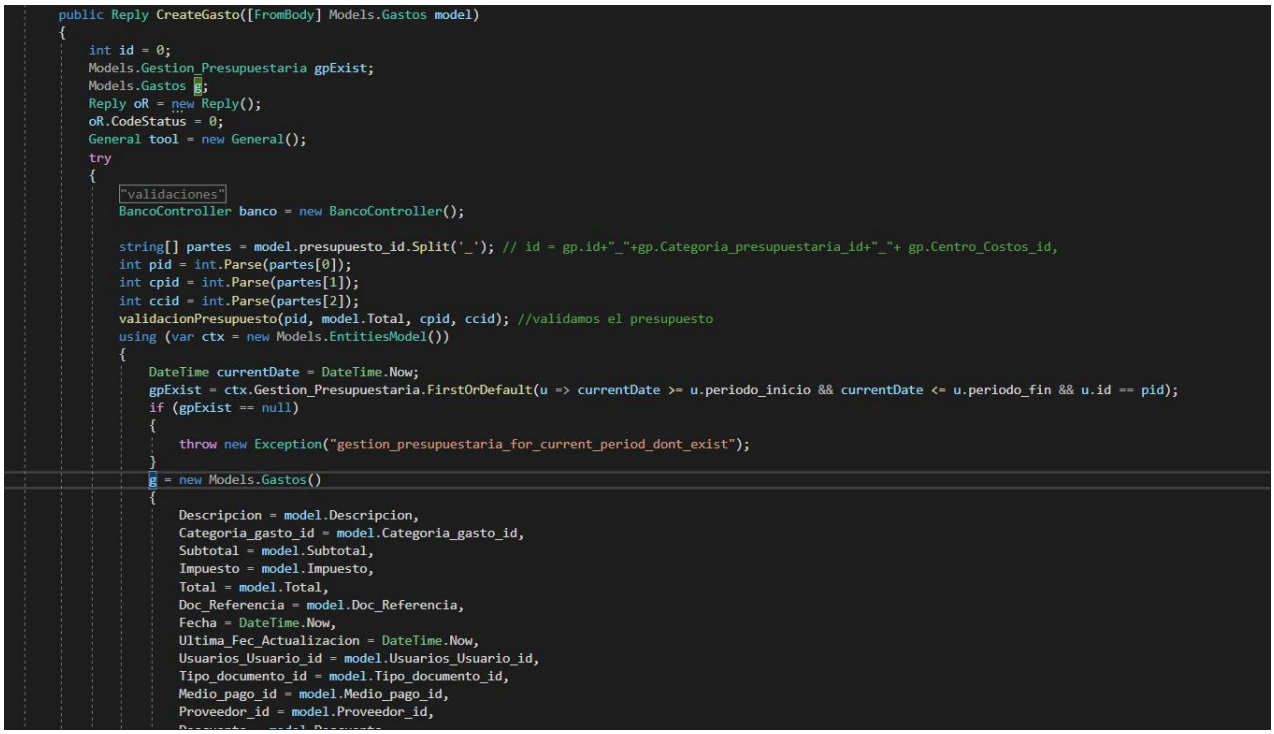
```
_recalcularTotales = () => {
  const subtotal = lines.reduce((acc, l) => acc + l.subtotal, 0);
  const impuesto = lines.reduce((acc, l) => acc + l.impuesto, 0);
  const total = lines.reduce((acc, l) => acc + l.total, 0);
  const descuento = lines.reduce((acc, l) => acc + l.descuento, 0);
  this.setState((prevState) => ({
    spent: {
      ...prevState.spent,
      subtotal,
      impuesto,
      total,
      descuento
    },
  }));
};
_calculaInput = (e, isSelect= false) =>{
 let {name, type, value, selectedIndex} = e.target;
  if(isSelect)
  {
    const index = selectedIndex;
    const optionElement = e.target.options[index];
    const taxOption = optionElement.getAttribute('attr');
    name = 'porcentaje'
    value = taxOption
    }

  this.setState({ AuxLine: { ...this.state.AuxLine, [name]: value }}, () =>{

  let {AuxLine} = this.state;
  let subtotal = isNaN(AuxLine.subtotal) ? 0 :AuxLine.subtotal,
   tax = isNaN(AuxLine.porcentaje) ? 0 : AuxLine.porcentaje ,
   impuesto =0,
   total = 0,
   cantidad = isNaN(AuxLine.cantidad) ? 1 : AuxLine.cantidad ,
   descuento = isNaN(AuxLine.descuento)? 0 :AuxLine.descuento ;
   impuesto = ((subtotal*cantidad)-descuento) * (tax/100)
   total = ((subtotal*cantidad)-descuento) + impuesto;
  AuxLine.total = total;
  AuxLine.impuesto = impuesto
  this.setState({AuxLine})
   });
```

Fuente: Elaboración propia.

Una vez se tiene la información desde el front-end, es enviada para ser procesada por el back-end, como se puede apreciar en la siguiente sección de código se procede a validar el presupuesto enviado desde el front-end, esto para descubrir que la operación puede ser procesada por los fondos del presupuesto y el banco.

Ilustración 79

Módulo Gastos Creación



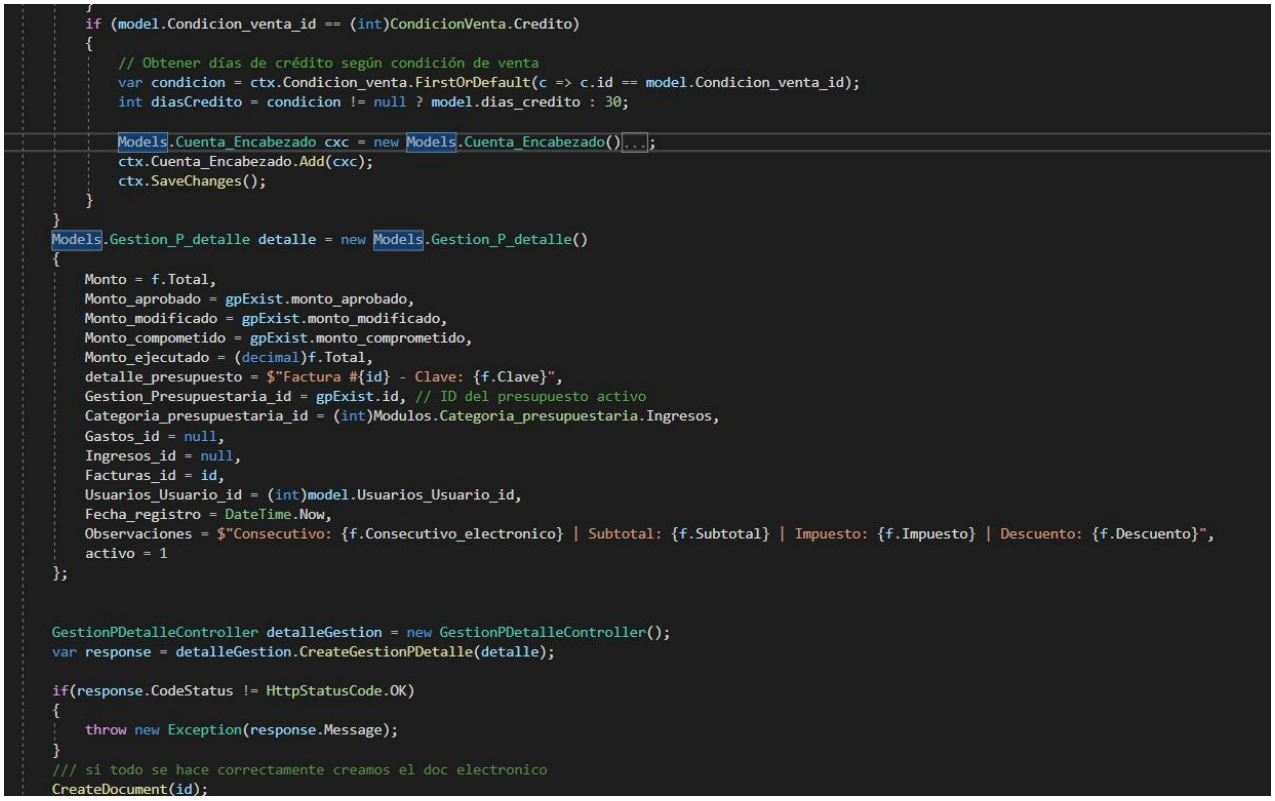
```
public Reply CreateGasto([FromBody] Models.Gastos model)
{
    int id = 0;
    Models.Gestion_Presupuestaria gpExist;
    Models.Gastos g;
    Reply oR = new Reply();
    oR.CodeStatus = 0;
    General tool = new General();
    try
    {
        "validaciones"
        BancoController banco = new BancoController();

        string[] partes = model.presupuesto_id.Split('_'); // id = gp.id+"_"+gp.Categoria_presupuestaria_id+"_"+ gp.Centro_Costos_id,
        int pid = int.Parse(partes[0]);
        int cpid = int.Parse(partes[1]);
        int ccid = int.Parse(partes[2]);
        validacionPresupuesto(pid, model.Total, cpid, ccid); //validamos el presupuesto
        using (var ctx = new Models.EntitiesModel())
        {
            DateTime currentDate = DateTime.Now;
            gpExist = ctx.Gestion_Presupuestaria.FirstOrDefault(u => currentDate >= u.periodo_inicio && currentDate <= u.periodo_fin && u.id == pid);
            if (gpExist == null)
            {
                throw new Exception("gestion_presupuestaria_for_current_period_dont_exist");
            }
            g = new Models.Gastos()
            {
                Descripcion = model.Descripcion,
                Categoria_gasto_id = model.Categoria_gasto_id,
                Subtotal = model.Subtotal,
                Impuesto = model.Impuesto,
                Total = model.Total,
                Doc_Referencia = model.Doc_Referencia,
                Fecha = DateTime.Now,
                Ultima_Fec_Actualizacion = DateTime.Now,
                Usuarios_Usuario_id = model.Usuarios_Usuario_id,
                Tipo_documento_id = model.Tipo_documento_id,
                Medio_pago_id = model.Medio_pago_id,
                Proveedor_id = model.Proveedor_id,
```

Fuente: Elaboración propia.

Facturación

Ilustración 80

Módulo Facturación



```
        if (model.Condicion_venta_id == (int)CondicionVenta.Credito)
        {
            // Obtener días de crédito según condición de venta
            var condicion = ctx.Condicion_venta.FirstOrDefault(c => c.id == model.Condicion_venta_id);
            int diasCredito = condicion != null ? model.dias_credito : 30;

            Models.Cuenta_Encabezado cxc = new Models.Cuenta_Encabezado()...;
            ctx.Cuenta_Encabezado.Add(cxc);
            ctx.SaveChanges();
        }
    }
    Models.Gestion_P_detalle detalle = new Models.Gestion_P_detalle()
    {
        Monto = f.Total,
        Monto_aprobado = gpExist.monto_aprobado,
        Monto_modificado = gpExist.monto_modificado,
        Monto_compometido = gpExist.monto_comprometido,
        Monto_ejecutado = (decimal)f.Total,
        detalle_presupuesto = $"Factura #{id} - Clave: {f.Clave}",
        Gestion_Presupuestaria_id = gpExist.id, // ID del presupuesto activo
        Categoria_presupuestaria_id = (int)Modulos.Categoria_presupuestaria.Ingresos,
        Gastos_id = null,
        Ingresos_id = null,
        Facturas_id = id,
        Usuarios_Usuario_id = (int)model.Usuarios_Usuario_id,
        Fecha_registro = DateTime.Now,
        Observaciones = $"Consecutivo: {f.Consecutivo_electronico} | Subtotal: {f.Subtotal} | Impuesto: {f.Impuesto} | Descuento: {f.Descuento}",
        activo = 1
    };


    GestionPDetalleController detalleGestion = new GestionPDetalleController();
    var response = detalleGestion.CreateGestionPDetalle(detalle);

    if(response.CodeStatus != HttpStatusCode.OK)
    {
        throw new Exception(response.Message);
    }
    /// si todo se hace correctamente creamos el doc electronico
    CreateDocument(id);
```

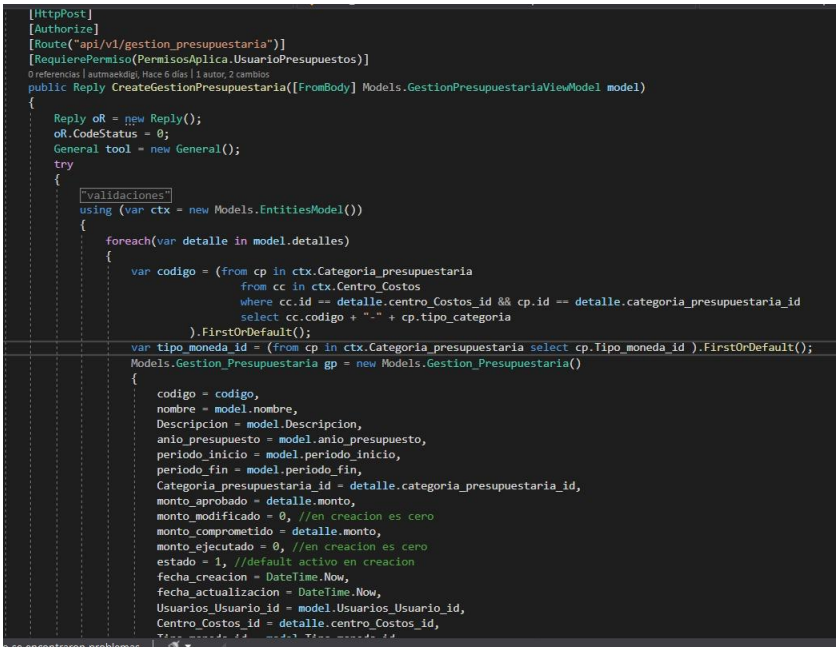
Fuente: Elaboración propia.

El módulo de Facturación permite generar el comprobante de venta correspondiente a cada transacción realizada para los clientes. Cuando la venta se realiza bajo la modalidad de crédito, el sistema calcula automáticamente el plazo otorgado y genera un registro dentro de las Cuentas por Cobrar, de manera que se pueda dar seguimiento al monto pendiente de pago y a su fecha de vencimiento. Al mismo tiempo, cada factura queda vinculada a la gestión presupuestaria vigente, actualizando los montos comprometidos y ejecutados dentro del presupuesto correspondiente. De esta forma, la facturación no solo respalda la venta ante el cliente, sino que también alimenta de manera automática el control financiero y presupuestario de la organización, manteniendo la consistencia entre lo vendido, lo cobrado y lo presupuestado.

Gestión Presupuestaria

Ilustración 81

Módulo Gestión Presupuestaria



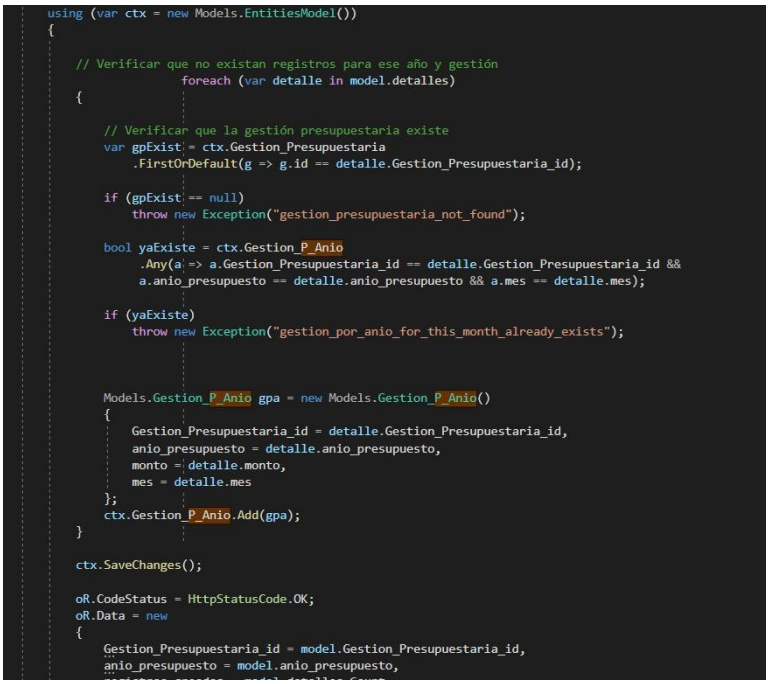
```
[HttpPost]
[Authorize]
[Route("api/v1/gestion_presupuestaria")]
[RequierePermiso(PermisosAplica.UsuarioPresupuestos)]
0 referencias | autmaekdigi, Hace 6 días | 1 autor, 2 cambios
public Reply CreateGestionPresupuestaria([FromBody] Models.GestionPresupuestariaViewModel model)
{
    Reply oR = new Reply();
    oR.CodeStatus = 0;
    General tool = new General();
    try
    {
        "validaciones"
        using (var ctx = new Models.EntitiesModel())
        {
            foreach(var detalle in model.detalles)
            {
                var codigo = (from cp in ctx.Categoria_presupuestaria
                              from cc in ctx.Centro_Costos
                              where cc.id == detalle.centro_Costos_id && cp.id == detalle.categoria_presupuestaria_id
                              select cc.codigo + "-" + cp.tipo_categoria
                        ).FirstOrDefault();
                var tipo_moneda_id = (from cp in ctx.Categoria_presupuestaria select cp.Tipo_moneda_id ).FirstOrDefault();
                Models.Gestion_Presupuestaria gp = new Models.Gestion_Presupuestaria()
                {
                    codigo = codigo,
                    nombre = model.nombre,
                    Descripcion = model.Descripcion,
                    anio_presupuesto = model.anio_presupuesto,
                    periodo_inicio = model.periodo_inicio,
                    periodo_fin = model.periodo_fin,
                    Categoria_presupuestaria_id = detalle.categoria_presupuestaria_id,
                    monto_aprobado = detalle.monto,
                    monto_modificado = 0, //en creacion es cero
                    monto_comprometido = detalle.monto,
                    monto_ejecutado = 0, //en creacion es cero
                    estado = 1, //default activo en creacion
                    fecha_creacion = DateTime.Now,
                    fecha_actualizacion = DateTime.Now,
                    Usuarios_Usuario_id = model.Usuarios_Usuario_id,
                    Centro_Costos_id = detalle.centro_Costos_id,
```

Fuente: Elaboración propia.

El módulo de Gestión Presupuestaria constituye la base sobre la cual se administra el uso de los recursos económicos de la institución durante un período determinado. Permite crear los presupuestos asignados a cada centro de costos y categoría, definiendo el monto aprobado inicial, el período de vigencia y la moneda en la que se maneja cada partida. A partir de esta asignación, el sistema da seguimiento constante a cuánto se ha comprometido, cuánto se ha modificado y cuánto se ha ejecutado de cada presupuesto.

Ilustración 82

Gestión Presupuestaria por Año



```
using (var ctx = new Models.EntitiesModel())
{

    // Verificar que no existan registros para ese año y gestión
                foreach (var detalle in model.detalles)
    {

        // Verificar que la gestión presupuestaria existe
        var gpExist = ctx.Gestion_Presupuestaria
            .FirstOrDefault(g => g.id == detalle.Gestion_Presupuestaria_id);

        if (gpExist == null)
            throw new Exception("gestion_presupuestaria_not_found");

        bool yaExiste = ctx.Gestion_P_Anio
             .Any(a => a.Gestion_Presupuestaria_id == detalle.Gestion_Presupuestaria_id &&
             a.anio_presupuesto == detalle.anio_presupuesto && a.mes == detalle.mes);

        if (yaExiste)
            throw new Exception("gestion_por_anio_for_this_month_already_exists");



        Models.Gestion_P_Anio gpa = new Models.Gestion_P_Anio()
        {
            Gestion_Presupuestaria_id = detalle.Gestion_Presupuestaria_id,
            anio_presupuesto = detalle.anio_presupuesto,
            monto = detalle.monto,
            mes = detalle.mes
        };
        ctx.Gestion_P_Anio.Add(gpa);
    }

    ctx.SaveChanges();

    oR.CodeStatus = HttpStatusCode.OK;
    oR.Data = new
    {
        Gestion_Presupuestaria_id = model.Gestion_Presupuestaria_id,
        anio_presupuesto = model.anio_presupuesto,
```

Fuente: Elaboración propia.

Este código es un complemento a la gestión presupuestaria, permite distribuir el presupuesto aprobado en periodos mensuales dentro de un mismo año. Antes de registrar una nueva asignación, el sistema verifica que la gestión presupuestaria a la que pertenece exista y que no se haya definido previamente un monto para ese mes y año, evitando así duplicidad de información. Esta distribución mensual facilita una planificación más detallada del presupuesto, permitiendo comparar lo proyectado contra lo realmente ejecutado mes a mes y contar con mayor control sobre el comportamiento financiero de la institución a lo largo del año.

Ingresos

Ilustración 83

Módulo Ingresos

```
gpExist = ctx.Gestion_Presupuestaria.FirstOrDefault(u => currentDate >= u.periodo_inicio && currentDate <= u.periodo_fin);

if (gpExist == null)
{
    throw new Exception("gestion_presupuestaria_for_current_period_dont_exist");
}
 i = new Models.Ingresos()
{
    Codigo = model.Codigo,
    fecha = DateTime.Now,
    Tipo_moneda_id = model.Tipo_moneda_id,
    Estado_Factura_id = model.Estado_Factura_id,
    Subtotal = model.Subtotal,
    Impuesto = model.Impuesto,
    Total = model.Total,
    Descuento = model.Descuento,
    cambio_venta = model.cambio_venta,
    cambio_compra = model.cambio_compra,
    Clientes_id = model.Clientes_id,
    Usuarios_Usuario_id = model.Usuarios_Usuario_id,
    Medio_pago_id = model.Medio_pago_id,
    Condicion_venta_id = model.Condicion_venta_id
    //Facturas_id = null//model.Facturas_id
};
ctx.Ingresos.Add(i);
ctx.SaveChanges();

// Guardar detalles usando el mismo contexto
IngresosDetalleController ingresosDetalle = new IngresosDetalleController();
foreach (var detalle in model.Ingresos_Detalle)
{
    id = i.id;
    detalle.Ingresos_id = i.id;
    var result = ingresosDetalle.CreateIngresoDetalle(detalle, ctx);
    if (result.CodeStatus != HttpStatusCode.OK)
    {
        throw new Exception(result.Message);
    }
}
```

Fuente: Elaboración propia.

Se puede definir que el módulo de Ingresos registra el dinero que efectivamente recibe la institución, ya sea producto de las ventas realizadas o de otras fuentes de entrada de fondos. Antes de registrar un ingreso, el sistema confirma que exista una gestión presupuestaria activa para la fecha en que se realiza la operación, asegurando que todo ingreso quede correctamente asociado al período correspondiente. Cada registro incluye información como el monto, el tipo de moneda, el medio de pago utilizado, el cliente relacionado y la condición de la venta, y puede desglosarse en distintas líneas de detalle. De esta forma, el módulo funciona como contraparte de las Cuentas por Cobrar generadas en Facturación, reflejando el momento en que dichas cuentas pendientes se hacen efectivas y se convierten en recursos disponibles para la institución.

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

El presente capítulo expone las conclusiones derivadas del desarrollo del sistema web para la gestión financiero contable de Marsh Asprose S.A., contrastando los resultados obtenidos con los objetivos planteados desde el inicio de la investigación y con una problemática identificada en la organización. Posteriormente, se presentan las recomendaciones orientadas a fortalecer la solución construida y a asegurar su adecuada puesta en producción y su continuidad en el tiempo.

Conclusiones

1. **Objetivo general alcanzado.**

Se cumplió el objetivo general del proyecto al desarrollar un sistema web para el apoyo de la gestión financiera contable de Marsh Asprose, utilizando las herramientas Visual Studio con C#, SQL Server como motor de base de datos y React JS sobre Visual Studio Code. El prototipo centraliza en una sola plataforma procesos que anteriormente se ejecutaban de forma dispersa y manual en hojas de cálculo, ofreciendo una base tecnológica sólida para la operación contable de la compañía. Y su crecimiento a futuro.

2. **Análisis de requerimientos.**

   El análisis de requerimientos, sustentado en la aplicación de un cuestionario a treinta colaboradores y en la observación directa de los procesos, permitió identificar con claridad las necesidades funcionales y no funcionales del sistema. Los hallazgos confirmaron una dependencia de Microsoft Excel, la ausencia de respaldos automáticos, la presencia de registros duplicados y errores contables, así como tiempos de registro de entre cuatro y siete minutos por transacción. El consenso del 97 por ciento de los encuestados respecto a la utilidad de una herramienta informática validó la pertinencia de la solución y orientó la priorización de la integración entre módulos, la seguridad y la automatización.

3. **Diseño de la base de datos y del sistema.**

Se construyó el diseño de la base de datos y del sistema mediante el modelado entidad-relación, el diccionario de datos y los diagramas de casos de uso y UML correspondientes. La adopción de una arquitectura organizada por capas y la definición de las entradas, procesos y salidas permitieron establecer la interconexión entre los módulos, condición indispensable para eliminar la doble digitación de la información y garantizar la coherencia de los registros financieros.

4. **Programación de los módulos.**

   Se programaron los módulos definidos en el alcance: cuentas por cobrar y por pagar, gastos, facturación electrónica, gestión presupuestaria, ingresos, reportes y seguridad. El módulo de facturación cumple con la normativa del Ministerio de Hacienda en su versión 4.4, incorporando la firma electrónica de los comprobantes y su transmisión al ente recaudador. La generación automática de asientos contables, la validación del presupuesto disponible al momento de registrar un gasto y la vinculación automática entre la facturación, las cuentas por cobrar y la gestión presupuestaria demuestran que el sistema mantiene la consistencia entre lo vendido, lo cobrado y lo presupuestado sin intervención manual.

5. **Pruebas del sistema.**

   Se probaron los módulos de forma independiente e integral, incorporando validaciones tanto en la capa de presentación como en la capa de servicios mediante expresiones regulares, además de validaciones de negocio como la comprobación de fondos presupuestarios y bancarios antes de procesar una operación. Estas pruebas verificaron el correcto funcionamiento del sistema y su apego a los requerimientos establecidos, aunque se reconoce que la validación se realizó de manera manual.

6. **Resolución de la problemática.**

En conjunto, el sistema atiende de forma directa la problemática planteada: brinda visibilidad sobre la ejecución presupuestaria, ordena el control de los ingresos, sincroniza la información comercial y financiera, reduce la incoherencia y la duplicidad en los registros contables, mejora la trazabilidad de los gastos operativos y ordena la gestión de las obligaciones con proveedores.

De esta manera, el prototipo transforma una operación funcional pero ineficiente e insegura en un flujo automatizado, centralizado y trazable.

Recomendaciones

1. **Fortalecer el manejo de credenciales.**

   Sustituir el envío de contraseñas en texto plano por correo electrónico durante la creación o restablecimiento de usuarios. Se recomienda implementar un flujo de recuperación basado en enlaces o tokens temporales de un solo uso y almacenar las contraseñas aplicando funciones de hash con sal. Este ajuste es prioritario antes de la puesta en producción, dado que el manejo de credenciales en texto plano representa un riesgo de seguridad relevante para información financiera sensible.

2. **Validar la compilación en el ambiente de entrega.**

   Verificar la compilación y la construcción de la solución en el ambiente de entrega antes de cada despliegue. Se sugiere apoyarse en los servidores de QA y preproducción ya disponibles en la infraestructura de la compañía para confirmar que el código compila y se ejecuta correctamente en un entorno equivalente al productivo, evitando así incidencias detectadas únicamente en fases tardías.

3. **Parametrizar convenciones internas de datos.**

   Revisar la convención mediante la cual las facturas de compra se asocian a un cliente fijo con identificador uno. Se recomienda parametrizar o documentar explícitamente este comportamiento para prevenir inconsistencias en los datos y facilitar el mantenimiento futuro del módulo de cuentas por pagar.

4. **Asegurar la integración con Hacienda.**

   Confirmar el uso exclusivo del endpoint de producción del Ministerio de Hacienda para la recepción de comprobantes electrónicos, considerando la migración a TRIBU-CR. Adicionalmente, se aconseja reforzar el manejo de los estados de respuesta del ente recaudador, incorporando reintentos automáticos y una cola de

procesamiento para los comprobantes que queden en estado de pendiente de confirmación.

5. **Establecer respaldos automatizados.**

   Implementar una política de respaldos automáticos y periódicos de la base de datos, acompañada de pruebas de restauración. Los resultados de la investigación evidenciaron la ausencia de respaldos como un riesgo latente, por lo que la automatización de esta tarea resulta indispensable para garantizar la disponibilidad e integridad de la información contable.

6. **Automatizar las pruebas.**

   Incorporar de forma progresiva pruebas unitarias y automatizadas, así como un mecanismo de integración continua. Dado que las validaciones se realizaron manualmente, contar con pruebas automatizadas ofrecería mayor confiabilidad, en especial sobre la generación automática de asientos contables y sobre las validaciones presupuestarias, reduciendo el riesgo de regresiones ante futuros cambios.

7. **Consolidar los módulos y reportes pendientes.**

   Completar los componentes contemplados en el alcance que consolidan la independencia respecto al proveedor externo: el algoritmo de conciliación bancaria y de cuentas por pagar, los módulos de mantenimiento y consultas, y la generación de reportes financieros como el Balance General, el Estado de Resultados y el Flujo de Caja. Con ello, la organización podría dejar de depender de terceros para la obtención de sus reportes.

8. **Habilitar la trazabilidad y auditoría de cambios.**

   Incorporar un registro de auditoría que documente el historial de cambios sobre las transacciones. Esta funcionalidad fortalecería la trazabilidad identificada como oportunidad de mejora durante la observación y respaldaría eventuales procesos de control interno o fiscalización.

9. **Prever la evolución tecnológica del sistema.**

   Evaluar, en fases posteriores, la evolución tecnológica de la solución hacia versiones más recientes del entorno de ejecución y del mapeo objeto-relacional, migrando gradualmente desde el modelo basado en Entity Framework con EDMX hacia enfoques más actuales. Esta modernización favorecería la mantenibilidad, el soporte a largo plazo y la escalabilidad del sistema, aprovechando además la infraestructura con balanceo de carga ya disponible en la compañía.

10. **Acompañar la adopción por parte de los usuarios.**

    Definir un plan de capacitación y acompañamiento para el personal del área contable, junto con la documentación del esquema de permisos por funcionalidad. Esto permitiría cerrar la brecha de acceso a herramientas evidenciada en la investigación y asegurar la adopción efectiva del sistema en la operación diaria.

Ejecución de Recomendaciones.

En relación con el apartado anterior, se determina que, si bien la aplicación requiere un conjunto de mejoras ya identificadas, estas no deben implementarse de forma aislada, sino a partir de un análisis técnico y funcional más profundo.

Dicho análisis consistirá en la revisión conjunta de cada recomendación, la estimación de las modificaciones necesarias en el código y en la base de datos, y la priorización de los cambios según su impacto sobre la seguridad, la trazabilidad y la continuidad operativa del sistema. Esta labor estará a cargo del departamento de Tecnologías de la Información, en coordinación con el departamento de Cobros.

Al primero le corresponderá el diseño e implementación de las modificaciones sobre la aplicación y la base de datos, mientras que el segundo aportará el conocimiento del proceso contable y de cobros, de modo que las mejoras respondan a las necesidades reales de la operación y no únicamente a criterios técnicos.

El proceso deberá ejecutarse en un intervalo no mayor a un año, plazo que permite abordar de manera escalonada las mejoras prioritarias, como el fortalecimiento del manejo de credenciales, la validación de la compilación y el establecimiento de respaldos automatizados, antes de la puesta en producción, y reservar las de menor urgencia, como la evolución tecnológica de la solución, para fases posteriores. Cada cambio deberá tramitarse mediante una solicitud formal en la que se detallen

sus beneficios, su impacto y los riesgos asociados, con el fin de que el negocio apruebe tanto el desarrollo como la conformación de un equipo específicamente destinado a la validación y prueba de dichas modificaciones.

La ejecución de estas recomendaciones se justifica en la necesidad de que la empresa Marsh Asprose disponga de una solución plenamente funcional, capaz de otorgar trazabilidad a cada acción registrada en el sistema y de garantizar la disponibilidad e integridad de la información contable a lo largo del tiempo. De este modo, la organización aprovecha al máximo el ciclo de vida del software, reduce su dependencia de proveedores externos y asegura la continuidad operativa dentro de un entorno productivo.

REFERENCIAS

Adobe For Business (2025, 23 mayo). Metodología Cascada. https://business.adobe.com/blog/basics/waterfall

Acosta, S. F. F. (2023a). Criterios para la selección de técnicas e instrumentos de recolección de datos en las investigaciones mixtas. Revista Honoris Causa, 15(2), 62-83.

Acosta, S. F. F. (2023b). Los enfoques de investigación en las Ciencias Sociales. Revista Latinoamericana Ogmios, 3(8), 82-95.

Álvarez, O. D. G., Larrea, N. P. L., & Valencia, M. V. R. (2022). Análisis comparativo de Patrones de Diseño de Software. Polo del Conocimiento: Revista científico-profesional, 7(7), 2146-2165.

Álvarez, C. (2023, 4 abril). ¿Cuál es la diferencia entre una fuente primaria y secundaria? WGU. https://www.wgu.edu/blog/what-difference-between-primary-secondary-source2304.html

Astudillo, F J. (2021). Qué es el virtual DOM en React. https://styde.net/que-es-el-virtual-dom-en-react/

Barillas, M. R. M. (2024). Midiendo la realidad: El papel de las variables en la investigación científica. Revista Docencia Universitaria, 5(2), 51-68.

Bhattacharya, A. (2023, 21 julio). Maslow's hierarchy in software. Philosophy Of Software Engineering. https://avishekbhattacharya.wordpress.com/2023/07/21/maslows-hierarchy-in-software/

Caules, C. Á. (2023, 7 marzo). ¿Qué es REST ? Arquitectura Java. https://www.arquitecturajava.com/que-es-rest/

Conecta Software. (2024, 12 febrero). Diccionario de datos - Conecta Magazine. Conecta Magazine. https://www.conectasoftware.com/magazine/glosario/diccionario-de-datos/

European Knowledge Center for Information Technology. (2023, 28 noviembre). Análisis de requisitos de software: ¿Cómo saber qué se necesita y a qué darle prioridad? TIC Portal. Recuperado en 8 de abril de 2026 de https://www.ticportal.es/glosario-tic/analisis-requisitos-software

F5. (2026, 02 de febrero). ¿Qué es un balanceador de carga? https://www.f5.com/glossary/load-balancer

Feder, M. (2026, 8 mayo). The Benefits of Learning to Code | University of Phoenix s. University Of Phoenix. https://www.phoenix.edu/articles/it/the-benefits-of-learning-to-code.html

GeeksforGeeks. (2025, septiembre 27). Requirements elicitation Software engineering. https://www.geeksforgeeks.org/software-engineering/software-engineering-requirements-elicitation/

Gowda, P., & Gowda, A. N. (2024). Best practices in REST API design for enhanced scalability and security. Journal of Artificial Intelligence, Machine Learning and Data Science, 2(1), 827-830.

Huet, P. (2022, 24 agosto). Arquitectura de software: Qué es y qué tipos existen. OpenWebinars.net. https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/

IBM. (2026, 02 de febrero). ¿Qué es la API REST? https://www.ibm.com/es-es/think/topics/rest-apis

IBM. (2026, 02 de febrero). ¿Qué es una base de datos? https://www.ibm.com/mx-es/think/topics/database

IBM. (2026, 08 de febrero). ¿Qué es la gestión de requisitos? https://www.ibm.com/mx-es/think/topics/database

Indeed. (2026, 01 de febrero). What Is a Web Application? How a Web Application Works, Benefits and Examples. https://www.indeed.com/career-advice/career-development/what-is-web-application

Irrazábal, E. A., & Mascheroni, M. A. (2022). Fundamentos de las pruebas continuas de software.

Julca, J. E. (2024, 5 septiembre). ¿Cuál es la diferencia entre población, muestra y unidad de análisis? - Escuela de Investigación. Escuela de Investigación. https://escueladeinvestigacion.com/2024/09/05/cual-es-la-diferencia-entre-poblacion-muestra-y-unidad-de-analisis/

Mamani, J. H., Gamarra, J. E. M., & Lucana, E. C (2024). Guia practica para investigación cuantitativa. Científica digital

Medhat, N. (2023, 19 agosto). Granularidad en la arquitectura de software. Medium. https://nadermedhatthoughts.medium.com/granularity-in-software-architecture-bec7c432d6d3

Microsoft. (2026, 23 de febrero). Paseo por el lenguaje C# https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/overview

Microsoft. (2026, 02 de febrero). ¿Qué es SQL Server? https://learn.microsoft.com/es-es/sql/sql-server/what-is-sql-server?view=sql-server-ver17

Microsoft. (2025, 17 de diciembre). Contexto de la integración del modelo de madurez y de capacidad (CMMI) https://learn.microsoft.com/es-es/azure/devops/boards/work-items/guidance/cmmi/guidaace-background-to-cmmi?view=azure-devops

Mozilla. (2026, 01 de febrero). Modelo de objetos de documento (DOM). https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model

Node JS. (2026, 02 de febrero). About Node JS https://nodejs.org/en/about

Okta. (2025, 3 junio). Security Through Obscurity (STO): History, criticism & Risks. Okta. https://www.okta.com/identity-101/security-through-obscurity/

Ordoñez-Pacheco, Á. F. (2025). Metodología de la Investigación Metodología académica con aplicación a las investigaciones sociales: enfoques, tipos, métodos y diseños. Sociedad & Tecnología, 8(2), 335-357.

Padilla-Avalos, C. A., & Marroquín-Soto, C. (2021). Enfoques de investigación en odontología: cuantitativa, cualitativa y mixta. Revista estomatológica herediana, 31(4), 338-340.

Porras, A. A. (2023). Metodologías ágiles para el desarrollo de software. Universidad Distrital Francisco José de Caldas.

Psychological Scales. (2025, 3 de diciembre). How to Define and Use Conceptual Variables in Your Research? https://scales.arabpsychology.com/stats/what-is-a-conceptual-variable/

Questa-Torterolo, M., Cabrera Borges, C., & Tejera Techera, A. (2022). ¿ Qué es la investigación?. Competencias y herramientas de investigación aplicada con foco en la gestión educativa.

Rebollo, P. A., & Ábalos, E. M. (2022). Metodología de la investigación/recopilación. Editorial Autores de Argentina.

Sánchez, A. D. C. G. (2023). Metodología y estadística en la investigación científica. PUERTO MADERO EDITORIAL eBooks

Sarabia, C. (2025, septiembre). Enfoques, tipos y niveles de la investigación. Researchgate. https://www.researchgate.net/publication/395942767_Enfoques_tipos_y_niveles_de_la_investigacion

Sarango, A. H., Pallmay, E. R. C., Sarzosa, J. P. R., & Pozo, J. E. C. (2024). Tipos y clasificación de las investigaciones. Latam: revista latinoamericana de Ciencias Sociales y Humanidades, 5(2), 39.

Schneider, A. (2024, 26 noviembre). A complete guide to the Waterfall methodology. monday.com Blog. https://monday.com/blog/project-management/waterfall-methodology/

Mentores Tech. (2025, 11 mayo). Requerimientos Funcionales y No Funcionales: qué son, en qué se diferencian y por qué son clave en el diseño de software.. https://www.mentorestech.com/resource-blog-content/requerimientos-funcionales-y-no-funcionales-que-son-en-que-se-diferencian-y-por-que-son-clave-en-el-diseno-de-software

Tigova, J. (2025). Fuente Primaria de Datos: Pilar fundamental en la investigación científica. Archivos de medicina, 21(2), 7.

Tristancho C. (2026). Plantilla de documento de requisitos de producto. Project Manager. https://www.projectmanager.com/es/plantilla-de-documento-de-requisitos-de-producto

Turturro, N. (2023, 14 noviembre). *Análisis y diseño de sistemas: Explorando los Sistemas Modernos*. DOOR3. https://www.door3.com/es/blog/system-analysis-and-design

Universae. (2026, 23 marzo). Hardware y Software: Qué son, Diferencias y Ejemplos (2026). UNIVERSAE. https://universae.com/blog/hardware-software/

Universidad Europea (2025, 13 junio). Diagrama Entidad Relación: tipos, características y ventajas. Universidad Europea. https://universidadeuropea.com/blog/modelo-entidad-relacion/

Valenciano, J. A. A. (2022). Las variables como elemento sustancial en el método científico. Revista Educación, 46(1), 1-10

Valle, A., Manrique, L., & Revilla, D. (2022). La investigación descriptiva con enfoque cualitativo en educación

VanZandt, P. (2022, 14 marzo). ¿Qué es un diagrama UML? Definición, casos de uso y cómo hacer. IdeaScale. https://ideascale.com/es/blogs/uml-diagram-definition/

Vázquez, C. (2025, 11 diciembre). Recolección de Datos: Estrategias, Métodos e Instrumentos. Clientify. https://clientify.com/blog/marketing/recoleccion-de-datos-metodos-tecnicas-e-instrumentos

Vera Carrasco, O. (2024). Tipos de fuentes bibliográficas para ser utilizadas en una revista médica. Revista Médica La Paz, 30(2), 7-8.

Wrike Team. (2025, 14 marzo). ¿Qué es un caso de uso? Blog Wrike. https://www.wrike.com/es/blog/que-es-un-caso-de-uso/

ANEXOS

Anexo 1

Cuestionario

CUESTIONARIO

En el marco de una investigación sobre sistema web de gestión financiero contable en Marsh Asprose le invitamos a completar este cuestionario. Su participación es de gran importancia para comprender cómo el tema en estudio influye en la actividad de la organización.

Este cuestionario es confidencial. Sus respuestas solo se utilizarán con fines académicos y no serán compartidas con otras personas. Completar el cuestionario tomará aproximadamente 10 minutos.

1. ¿Actualmente existe alguna aplicación o sistema que facilite el funcionamiento del área de cobros?
   a) SI
   b) NO
2. ¿Cómo cree usted que una herramienta informática beneficie su área? _____
   _____
   _____
3. ¿Qué procesos contables realiza usted?
   a) Facturación e ingresos
   b) Cobros y cuentas por cobrar
   c) Gastos
   d) Cuentas por cobrar
4. ¿Qué herramientas utiliza para los diferentes procesos contables que realiza?
   a) Excel
   b) Control manual (libros físicos)
   c) Herramientas de software, especifiqué: _____
   d) Otra herramienta, especifiqué: _____

5. ¿Qué funcionalidades considera prioritarias en un nuevo sistema web?

   _____

   _____

   _____

6. ¿Qué mejoras considera críticas para optimizar la gestión financiera en Marsh Asprose?

   _____

   _____

   _____

7. ¿Se presentan errores frecuentes en los registros contables?
   a) SI
   b) NO
8. ¿Existe duplicidad de información en los procesos financieros?
   a) SI
   b) NO
9. ¿Se realizan respaldos periódicos de la información financiera?
   a) SI
   b) NO
10. ¿Considera usted que la institución se vería beneficiada en la utilización de una herramienta informática en esta área?
    a) SI
    b) NO

Fuente: Desarrollo propio

Anexo 2

Observación

GUÍA DE OBSERVACIÓN

Entidad:	**Marsh Asprose**
Dirección física de la entidad:	**San José, Zapote Barrio Cordoba**
Fecha de la actividad de observación:	**Dd/mm/yyyy**
Nombre del estudiante:	**Douglas González Castro**

Tabla de control de aspectos observados:

No	Aspectos por observar	Cumple	No Cumple	Oportunidad de mejora	Detalle de Observación
1	Existencia de un sistema actual para la gestión contable (manual, Excel).				

No	Aspectos por observar	Cumple	No Cumple	Oportunidad de mejora	Detalle de Observación
2	Procedimiento actual para el registro de **Cuentas por Cobrar**.				
3	Control y seguimiento de morosidad en cuentas por cobrar.				
4	Procedimiento de aprobación y registro de **Cuentas por Pagar**.				

No	Aspectos por observar	Cumple	No Cumple	Oportunidad de mejora	Detalle de Observación
5	Control de vencimientos y programación de pagos a proveedores.				
6	Flujo actual del proceso de **Facturación**				
7	Existencia de control presupuestario formal (asignación, ejecución y seguimiento).				

No	Aspectos por observar	Cumple	No Cumple	Oportunidad de mejora	Detalle de Observación
8	Procedimiento para el registro de **Ingresos** no relacionados con facturación.				
9	Procedimiento para el registro y clasificación de **Gastos**.				
10	Existencia y actualización del **Catálogo de Cuentas Contables**.				

No	Aspectos por observar	Cumple	No Cumple	Oportunidad de mejora	Detalle de Observación
11	Conciliación bancaria periódica y control de diferencias.				
12	Generación de reportes financieros (Balance General, Estado de Resultados, Flujo de Caja).				
13	Trazabilidad de transacciones (historial de cambios).				

No	Aspectos por observar	Cumple	No Cumple	Oportunidad de mejora	Detalle de Observación
14	Nivel de automatización de procesos financieros.				
15	Tiempo promedio de registro de una transacción.				

Fuente: Desarrollo propio