PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

SISTEMA DE GESTIÓN CONTABLE PARA EL REGISTRO Y CONTROL DE INGRESOS Y EGRESOS EN LA EMPRESA SARVALTOURS

Proyecto de titulación previo a la obtención del título de: Tecnólogo Superior en Desarrollo de Software

Autor:

Diego Fernando Galindo Sarango

Tutor:

José Luis Galarraga Cañizares

Quito, Ecuador

2026

Dedicatoria

Dedico el presente trabajo de investigación a mi familia, por su apoyo incondicional, comprensión y motivación constante a lo largo de mi formación académica, siendo un pilar fundamental para la culminación de esta etapa.

Asimismo, dedico esta tesis a todas las personas que, con su confianza y respaldo, contribuyeron directa o indirectamente al desarrollo personal y profesional que hizo posible la realización de este proyecto.

Tabla de contenidos

Lista de tablas

Lista de figuras

DECLARACIÓN y AUTORIZACIÓN

Yo, **Diego Fernando Galindo Sarango** con C.I. 1103868962 autor del trabajo de titulación intitulado: **"Sistema de Gestión Contable para el registro y control de Ingresos y Egresos en la empresa Sarvaltours"**, previa a la obtención del título de **Tecnólogo Superior en Desarrollo de Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 6 de febrero de 2026

Diego Fernando Galindo Sarango

C.I. 1103868962

Agradecimientos

Expreso mi más sincero agradecimiento a Dios por brindarme la fortaleza, la constancia y la sabiduría necesarias para culminar esta etapa académica. A la PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR por brindar los recursos académicos y el entorno necesario para el desarrollo de la presente tesis.

De manera especial, agradezco al Ing. José Luis Galarraga Cañizares, por su valiosa orientación, acompañamiento constante y aportes académicos, los cuales fueron fundamentales para la correcta implementación y culminación de este trabajo de desarrollo de software.

A mi familia, por su apoyo incondicional, paciencia y motivación constante a lo largo de todo este proceso. A mis docentes y, en especial, a mi tutor de tesis, por su guía, conocimientos y valiosos aportes que hicieron posible el desarrollo de este trabajo.

Finalmente, agradezco a todas las personas que, de manera directa o indirecta, contribuyeron al desarrollo del presente sistema.

Introducción

La gestión financiera eficiente constituye el pilar fundamental para la sostenibilidad y el crecimiento de cualquier organización, especialmente en sectores dinámicos y competitivos como el turismo. En la actualidad, la globalización y la alta demanda de servicios de viaje exigen a las empresas turísticas, como Sarvaltours, una rigurosidad contable y un control de flujos de efectivo que permitan la toma de decisiones ágil y bien fundamentada.

Sarvaltours, una compañía con trayectoria en el sector de viajes, enfrenta el desafío común de muchas PYMEs: la necesidad de estandarizar y automatizar sus procesos de registro y control de ingresos y egresos. Actualmente, la dependencia de métodos manuales resulta en errores de registro, retrasos en la conciliación bancaria y una visibilidad limitada sobre la rentabilidad real de sus servicios. Situación que impacta directamente en la capacidad de la gerencia para planificar inversiones y optimizar costos. Un problema adicional y crítico es la dificultad para rastrear, calcular y gestionar automáticamente las comisiones de los agentes de venta, un egreso crucial que, al manejarse manualmente, introduce ineficiencias y posibles inconsistencias.

El presente proyecto de tesis aborda esta problemática mediante el diseño e implementación de un Sistema de Gestión Contable especializado. Este sistema buscará centralizar, automatizar y estandarizar el registro de todas las transacciones financieras, desde los ingresos generados por la venta de boletos y paquetes turísticos, hasta los egresos incurridos en la operación, Para lograr esto, la solución tecnológica se desarrollará bajo una arquitectura web moderna, empleando la pila de tecnología **JavaScript** con el entorno de ejecución **Node.js** en el backend para construir una **API REST robusta**, y un framework de **JavaScript** en el frontend para una interfaz de usuario dinámica y eficiente..

El Objetivo General de este trabajo es desarrollar un sistema que no solo sirva como una herramienta de registro, sino también como un instrumento de control proactivo que proporcione reportes financieros detallados a la administración de Sarvaltours. Se espera que, con la implementación de este sistema, Sarvaltours logre una mayor transparencia financiera, una reducción significativa en los tiempos de cierre contable, y mejore su capacidad para tomar decisiones estratégicas basadas en datos fiables. Finalmente, se busca entregar una solución tecnológica que impulse la eficiencia operativa y la solidez financiera de la empresa.

Antecedentes

Sarvaltours es una agencia de viajes con una trayectoria significativa en el mercado ecuatoriano, habiendo sido fundada en 1991. Desde su constitución, la empresa se estableció con el firme objetivo de elevar los estándares de servicio y brindar una mejor experiencia en el sector turístico del Ecuador, consolidando su reputación a lo largo de más de tres décadas. La empresa ha priorizado consistentemente la calidad en la atención al cliente, lo cual ha impulsado su crecimiento y la diversificación de su portafolio de servicios.

Actualmente, el amplio alcance de Sarvaltours se refleja en la compleja matriz de servicios y transacciones que gestiona, incluyendo:

- Transporte de pasajeros.
- Boletos aéreos (Nacionales e Internacionales).
- Paquetes turísticos (Nacionales e Internacionales).
- Asesoría para trámites de visados.
- Reservas de Hotel.
- Cruceros (Servicio directo en las Islas Galápagos).

La longevidad, la variedad de la oferta y los múltiples canales de comercialización de Sarvaltours han generado un elevado volumen de transacciones de ingreso y egreso. Históricamente, la gestión contable se ha desarrollado de forma manual con herramientas no especializadas, lo que crea fricciones al intentar consolidar la información financiera.

Es precisamente la complejidad de sus operaciones y la necesidad de gestionar egresos variables y sensibles, como las comisiones de los agentes de venta lo que justifica la urgencia de modernizar la gestión. Por lo tanto, el presente proyecto se cimienta en la necesidad de optimizar la gestión contable de Sarvaltours para asegurar la precisión en la evaluación de la rentabilidad de cada una de estas líneas de negocio, utilizando su amplia trayectoria como punto de partida para la transformación digital de sus procesos financieros.

Planteamiento del Problema

Sarvaltours, a pesar de su consolidada trayectoria de más de tres décadas en el sector turístico ecuatoriano y su amplia oferta de servicios, enfrenta desafíos críticos derivados de la falta de un sistema de gestión contable especializado e integrado. La naturaleza dinámica del negocio de viajes, caracterizada por un alto volumen y variedad de transacciones (boletos, paquetes, reservas y cruceros), ha generado una sobrecarga en los procesos administrativos y contables actuales.

Actualmente, la gestión financiera se apoya en una combinación de métodos manuales y herramientas genéricas (como hojas de cálculo), lo que a su vez resulta en una gestión de la información dispersa y poco eficiente. Esta metodología presenta cuatro deficiencias operacionales y financieras fundamentales:

1. **Ineficiencia en el Registro y la Conciliación**: El registro de ingresos y egresos es un proceso manual y propenso a errores humanos. Lo cual causa que se retrasen significativamente los tiempos de cierre contable, impidiendo que la gerencia tenga una visión precisa y actualizada del flujo de caja disponible.
2. **Opacidad en la Rentabilidad de los Servicios:** La dispersión de la información financiera impide vincular directamente el ingreso de un servicio específico con sus costos operacionales asociados. Consecuentemente, Sarvaltours carece de métricas precisas para evaluar la rentabilidad individual de cada paquete turístico o línea de negocio.
3. **Restricción en la Toma de Decisiones Estratégicas:** Como resultado de la ineficiencia en los registros y la opacidad de los datos, la gerencia se encuentra limitada en su capacidad para tomar decisiones estratégicas.
4. **Gestión de comisiones a los agentes:** Debido a la ineficiencia en el registro la gestión de comisiones se ve afectada de manera directa lo cual genera demoras y contratiempos al momento de realizar el pago de comisiones lo cual afecta la relación con este canal de ingresos.

Por lo cual, dada la necesidad de estandarizar y automatizar los procesos contables y, específicamente, la gestión de comisiones, se formula la siguiente pregunta de investigación:

> ¿Cómo el diseño e implementación de un Sistema de Gestión Contable, enfocado en el registro y control automatizado de Ingresos y Egresos, puede mejorar la eficiencia, la precisión y la transparencia financiera en la empresa Sarvaltours?

Justificación

El desarrollo de un Sistema de Gestión Contable para el registro y control de Ingresos y Egresos en Sarvaltours se justifica en función de las necesidades operativas y estratégicas de la empresa, así como por su aporte a la metodología de desarrollo de sistemas.

El punto primordial de esta investigación es la necesidad práctica de resolver las ineficiencias operativas detectadas en Sarvaltours. La dependencia actual de procesos manuales genera costos ocultos asociados a la rectificación de errores, retrasos en la generación de informes y el uso ineficiente del tiempo del personal contable.

El sistema propuesto tendrá un impacto directo en las siguientes áreas:

- **Eficiencia en el Proceso Contable:** Automatizar el registro de transacciones, reduciendo drásticamente los tiempos de cierre contable y la necesidad de conciliación manual.
- **Trazabilidad Financiera:** Proveerá una visibilidad inmediata y precisa del flujo de caja, enlazando cada ingreso con sus egresos asociados.
- **Control de Comisiones**: Solucionará el problema crítico del cálculo de comisiones. El módulo automatizado garantizará el cálculo exacto e instantáneo de las comisiones devengadas por los agentes, eliminando errores.

Con el fin de solventar de manera efectiva las necesidades de la empresa Sarvaltours y cumpliendo con el standart académico y metodológico, la presente tesis tiene como objetivo contribuir con:

- **Modelo de Desarrollo Específico:** La investigación implica el diseño de un modelo de arquitectura de software adaptado a las particularidades del sector turístico, que maneja ingresos complejos y egresos variables.
- **Generación de Conocimiento:** Se documentará el proceso de transformación de requisitos de negocio en requerimientos funcionales y no funcionales dentro de un sistema contable. Esto servirá como referencia para futuras investigaciones enfocadas en la digitalización de procesos financieros en PYMEs de servicios en Ecuador.

De igual manera el proyecto ofrece un valor económico claro para Sarvaltours y un impacto social indirecto, debido a:

- **Viabilidad Económica:** Al optimizar la precisión de los egresos y la evaluación de la rentabilidad, la gerencia podrá tomar decisiones estratégicas basadas en datos, lo que conducirá a una mejora en el margen de utilidad de la empresa.
- **Transparencia y Confianza:** Al automatizar el cálculo de comisiones, se fomenta un ambiente de mayor transparencia y confianza con el equipo de ventas.
- **Aplicabilidad Empresarial:** El éxito de esta implementación servirá como un caso de estudio replicable para otras agencias de viajes ecuatorianas que busquen modernizar su infraestructura contable y enfrentar los desafíos de la competencia global.

En resumen, la presente tesis no solo es viable por su enfoque en tecnologías probadas, sino que es fundamental para la sostenibilidad y el crecimiento de Sarvaltours en un mercado cada vez más exigente en materia de control financiero.

Objetivo General

Diseñar e implementar un Sistema de control de Gestión Contable, desarrollado a base de JavaScript y Node.js, para el registro y control automatizado de Ingresos y Egresos con el fin de optimizar la eficiencia y transparencia financiera de la empresa Sarvaltours.

Objetivos Específicos

1. Levantar, analizar y documentar los requerimientos funcionales y no funcionales del sistema, tomando como base los procesos contables actuales de Ingresos y Egresos de Sarvaltours.
2. Diseñar la arquitectura lógica de datos y tecnológica del sistema de gestión contable, especificando la estructura de la API REST a desarrollar con Node.js y la base de datos relacionales PostgreSQL para garantizar la integridad y trazabilidad de los datos financieros.

3. Crear un modelo lógico y físico de la base de datos relacionales para el sistema, asegurando la correcta interconexión entre tablas de ingresos, Egresos.
4. Construir e implementar los módulos de frontend y backend para el registro detallado y clasificación de Ingresas y Egresos.
5. Implementar las funcionalidades de reporteo que permitan la generación de informes financieros claves.
6. Evaluar la funcionalidad y la eficiencia del sistema, mediante pruebas de aceptación de usuario contra los procesos manuales previos de Sarvaltours.

Capítulo I

Levantamiento de Requisitos y Diseño del Sistema

Análisis de requerimientos

El Análisis de Requerimientos es el proceso fundamental en el cual se descubren, documentan y gestionan todas las necesidades que el cliente (Sarvaltours) desea satisfacer a través del sistema a desarrollar. El objetivo de este proceso es garantizar que el producto final sea una solución directa y efectiva a la problemática financiera identificada.

Una definición clave y contemporánea establece la importancia de la fase inicial en la era de los sistemas interconectados:

> "La ingeniería de requisitos es la primera y más crítica actividad en el ciclo de vida del desarrollo de *software*, ya que asegura que los requisitos sean correctos, consistentes y completos antes de que comience el desarrollo. Fallar en esta etapa resulta en altos costos de corrección más adelante en el proyecto." (Sharma & Singh, 2021).

Esta necesidad de precisión es particularmente crítica en el sistema contable de Sarvaltours, donde cada ingreso y egreso debe ser trazable hasta su origen (agente, servicio) y destino (cuenta contable, proveedor) para evitar errores financieros costosos.

Dentro del desarrollo de software, el análisis de requisitos es una parte crucial que, según (Aurum & Wohlin, 2023), se descompone típicamente en cuatro fases interconectadas: Elicitación, Análisis, Especificación y Validación. Cada una de estas fases se basa en el entendimiento de lo que requiere el usuario para solventar un problema o una necesidad y en proponer a través de diferentes herramientas diversas soluciones que ayuden a mitigar dicha necesidad.

Alcance del proyecto

El sistema se enfocará en automatizar el ciclo contable operativo de Sarvaltours, para lo cual se delimitará el alcance del proyecto, tal como se detalla en la Tabla 1.

Funcionalidad Incluida	**Funcionalidad Fuera de alcance**
Registro de Ingreso y Egresos	Módulo de inventario físico
Gestión de Comisiones	Módulo de nómina.
Reportes contables Básicos	Módulo de CRM
Autenticación y gestión de roles	Integración en tiempo real con plataforma de terceros.
	Soporte para múltiples idiomas.
	Costos de licenciamiento

Tabla 1: Alcance del proyecto.

Requerimientos funcionales

Los requerimientos funcionales constituyen las características y funciones del sistema que permiten a los usuarios completar diferentes tareas para lo cual fue diseñada dicha aplicación. En el ámbito de la ingeniería de *software*:

> "Los requisitos funcionales definen las funciones que un sistema debe ejecutar, constituyendo el núcleo de la aplicación y estableciendo la línea base para el desarrollo de la arquitectura y la codificación." (Adaptado de Sommerville, 2019).

Por ello, dentro del análisis de requerimientos es fundamental identificar cuáles son las funcionalidades que serán el Core de la aplicación, en este caso la gestión contable, para que a partir de ellas se pueda generar la línea base que nos permita iniciar con el prototipado de la solución, definición de entidades y el posterior desarrollo de la aplicación.

La identificación de estos requerimientos funcionales se realiza a través de reuniones específicas con el equipo contable de Sarvaltours, en la cual se definen los comportamientos, funciones y capacidades específicas que el software debe proporcionar para satisfacer las necesidades financieras.

> "El proceso de especificación de requerimientos debe ser un esfuerzo colaborativo y continuo para capturar el comportamiento y las capacidades específicas que el sistema debe ofrecer a sus usuarios para resolver un problema de negocio." (Pressman & Maxim, 2021).

Dentro de este levantamiento se espera definir todos los módulos que requiere tener el sistema, así como cada una de las acciones que realizará el usuario en la aplicación, orientados a la gestión de ingresos y egresos.

El Sistema de Gestión Contable para Sarvaltours tendrá los siguientes requerimientos funcionales, detallados en la Tabla 2:

ID	**Modulo**	**Detalle del requerimiento funcional (RF)**
RF01	Autenticación y Seguridad	El sistema debe gestionar el acceso mediante un proceso de inicio de sesión seguro y controlar las acciones del usuario a través de un esquema de permisos basado en roles (Administrador, Contable, Agente de Ventas).
RF02	Gestión de Ingresos	El sistema debe facilitar el registro detallado de todos los ingresos provenientes de la venta de servicios turísticos, asegurando la vinculación obligatoria de cada transacción con el agente de venta responsable, el cliente y la cuenta contable de destino.
RF03	Gestión de Egresos	El sistema debe permitir el registro de egresos operativos (ej. Pagos a hoteles, aerolíneas, publicidad), proveyendo herramientas para su clasificación por tipo de gasto (categoría

		contable) y su asociación al proveedor correspondiente.
RF04	Cálculo de Comisiones	El sistema debe calcular automáticamente la comisión correspondiente a un agente, en tiempo real, al registrarse y confirmarse un ingreso, aplicando las reglas de negocio y porcentajes predefinidos y vigentes en Sarvaltours.
RF05	Liquidación de Comisiones	El sistema debe ser capaz de generar reportes de liquidación de comisiones por agente para un periodo específico, e incluir la funcionalidad de marcar el egreso por comisión como *pendiente de pago* o *pagado*, asegurando su trazabilidad contable.
RF06	Contabilidad Base	El sistema debe mantener un catálogo o plan de cuentas contables configurable por el administrador, permitiendo la clasificación y asiento automático de todas las transacciones de ingresos y egresos registradas.
RF07	Reportes Financieros	El sistema debe generar, bajo criterios de rango de fechas y filtros definidos, los Reportes Financieros clave: Estado de Resultados (Ingresos vs. Egresos) y Reporte de Flujo de Caja.
RF08	Conciliación Bancaria	El sistema debe permitir el registro y asociación de los movimientos bancarios para facilitar la conciliación de los ingresos por ventas y los egresos por pagos realizados.

Tabla 2: Requisitos Funcionales

Requisitos No Funcionales

Al contrario de los requisitos funcionales, este tipo de requisitos definen la calidad del servicio que se va a brindar a través de la plataforma, así como también la interfaz de usuario que tendrá y el tiempo que tomará cada proceso. Finalmente, también especifican las restricciones sobre cómo deben producirse y entregarse dichos servicios a través del sistema.

> "Los requerimientos no funcionales son las limitaciones y restricciones impuestas a un sistema que definen sus atributos de calidad, tales como rendimiento, usabilidad, seguridad y fiabilidad, y establecen la forma en que el sistema realiza sus funciones." (Basado en Pressman & Maxim, 2021).

Los requisitos no funcionales definen los aspectos que, de cierta forma, son intangibles para el usuario, pero que determinan la calidad del *software*. Entre estos requisitos están la velocidad de la aplicación, la seguridad y la integridad de datos.

> "La calidad de un *software* de gestión crítica depende directamente de la implementación efectiva de los requisitos no funcionales. Estos definen

compromisos que la aplicación adquiere con el usuario en términos de rendimiento bajo carga y la protección de la información sensible." (Altexsoft Editorial Team, 2023).

A continuación, en la tabla 3, se detallan los requisitos no funcionales que tendrá el Sistema de Gestión Contable para Sarvaltours, priorizando la seguridad y el rendimiento del manejo de datos financieros:

ID	Categoría	Requerimiento No Funcional (RNF)
RNF 01	Rendimiento	La respuesta de las operaciones de consulta que involucren consolidación de datos debe responder en un lapso de 5 segundos.
RNF 02	Carga	El sistema debe soportar un mínimo de 30 usuarios concurrentes, sin degradación de rendimiento perceptible.
RNF 03	Seguridad	La aplicación debe usar autenticación a través de JSON Web Tokens, evitando exponer cualquier información sensible a través de las APIs.
RNF 04	Seguridad	Las contraseñas de los usuarios deben almacenarse de forma segura en la base de datos, siendo encriptadas.
RNF 05	Seguridad	Todas las comunicaciones entre el frontend y el backend deben ser cifradas a través de HTTPS/SSL.
RNF 06	Control de Acceso	El sistema debe implementar un control de acceso basado en roles, restringiendo el acceso a módulos críticos solo a usuarios autorizados.
RNF 07	Usabilidad	La interfaz debe ser intuitiva y consistente en todos los módulos de registro.
RNF08	Disponibilidad	El sistema debe tener una disponibilidad operativa anual del 99%, excluyendo los mantenimientos programados previamente notificados.

Tabla 3: Requisitos no funcionales.

Arquitectura Empresarial

La arquitectura empresarial es un framework estructurado que permite relacionar los procesos de negocio de una organización con sus tecnologías de información, garantizando que exista una coherencia estratégica entre todos los niveles y sistemas. En el contexto de la transformación digital de Sarvaltours, la Arquitectura empresarial provee la estructura necesaria para alinear las capacidades operacionales con las soluciones tecnológicas.

"La arquitectura empresarial es una práctica disciplinada que facilita la coordinación entre las funciones de negocio y los sistemas de TI, actuando como un puente que asegura que las inversiones en tecnología estén directamente alineadas para alcanzar los objetivos estratégicos y apoyar la transformación digital de la compañía." (Adaptado de Jevic, 2023).

Debido al alcance delimitado de la tesis, se aplicará una Arquitectura de Segmento, concentrando el esfuerzo de diseño en el área de Gestión Financiera y Comercial de Sarvaltours.

Procesos de Negocio Clave

Los procesos que serán transformados o automatizados por el sistema se centran en el ciclo de vida de la transacción financiera:

- **Proceso de Venta y Registro de Ingreso**: Captura de la venta de servicios turísticos.
- **Proceso de Control de Comisiones:** Cálculo y liquidación de los pagos a agentes de venta.
- **Proceso de Gestión de Egresos:** Registro de pagos a proveedores y gastos operativos.
- **Proceso de Reporteo y Conciliación:** Generación de informes financieros y verificación de movimientos bancarios.

Estructura Organizacional y Actores.

Los principales actores de Sarvaltours que interactuaran con el sistema se especifican en la tabla 4:

Actor	Rol en el Negocio	Impacto del sistema
Agente de venta	Generador de ingresos, responsable de la venta.	Usará el sistema para registrar ventas y verificar sus comisiones devengadas.
Personal contable	Responsable de la clasificación de transacciones y cierres de mes.	Usará el sistema para el registro de egresos, la conciliación y la generación de reportes financieros.
Administrador	Responsable de la toma de decisiones.	Usará el sistema para la generación de reportes estratégicos y la configuración de las reglas de comisión.

Tabla 4: Actores principales

Arquitectura de Aplicación

Describe el diseño lógico del software y cómo el Sistema de Gestión Contable se estructurará internamente para cumplir con los requerimientos funcionales. Se adopta una arquitectura cliente-servidor:

- Capa de Presentación (Frontend): Desarrollada con un framework de JavaScript . Es la interfaz intuitiva con la que interactúan los usuarios, responsable de la usabilidad y la captura de datos.
- Capa de Lógica de Negocio: Desarrollada en Node.js utilizando un framework. Es el motor del sistema, encargado de ejecutar las reglas críticas del negocio, incluyendo la lógica del cálculo automático de comisiones, la validación de datos y la comunicación con la base de datos.
- Capa de Datos: Se accede exclusivamente a través de la API REST del backend para garantizar la seguridad y la integridad, Base de datos.
-

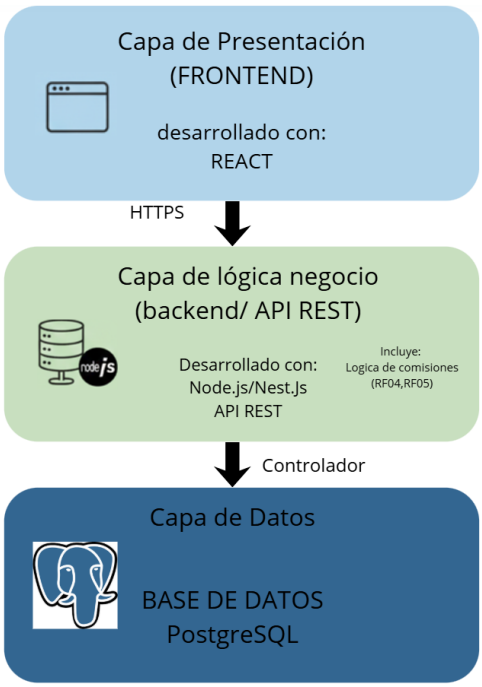


Figura 1. Diagrama Arquitectura de aplicación

Arquitectura de Datos

La Arquitectura de Datos define la estructura y el almacenamiento de la información. Dada la naturaleza transaccional y la necesidad de integridad financiera y trazabilidad, se selecciona una base de datos relacional:

- **Motor de Base de Datos:** PostgreSQL. Se elige por su robustez, integridad transaccional y su manejo avanzado de datos estructurados lo cual la hace ideal para asientos contables, clasificación de ingresos/egresos y el manejo de relaciones clave.
- **Modelado de Datos:** Se implementará un modelo relacional que asegurará la vinculación directa entre:
  - Tabla Ingresos.
  - Tabla ingresos_detalles

- Tabla proveedores.
- Tabla Bitacora
- Tabla Egresos.
- Tabla Usuarios.
- Tabla Clientes

- **Integridad:** Se implementarán restricciones a nivel de base de datos para asegurar que los datos financieros permanezcan consistentes y correctos.

Diagrama Entidad-Relación.

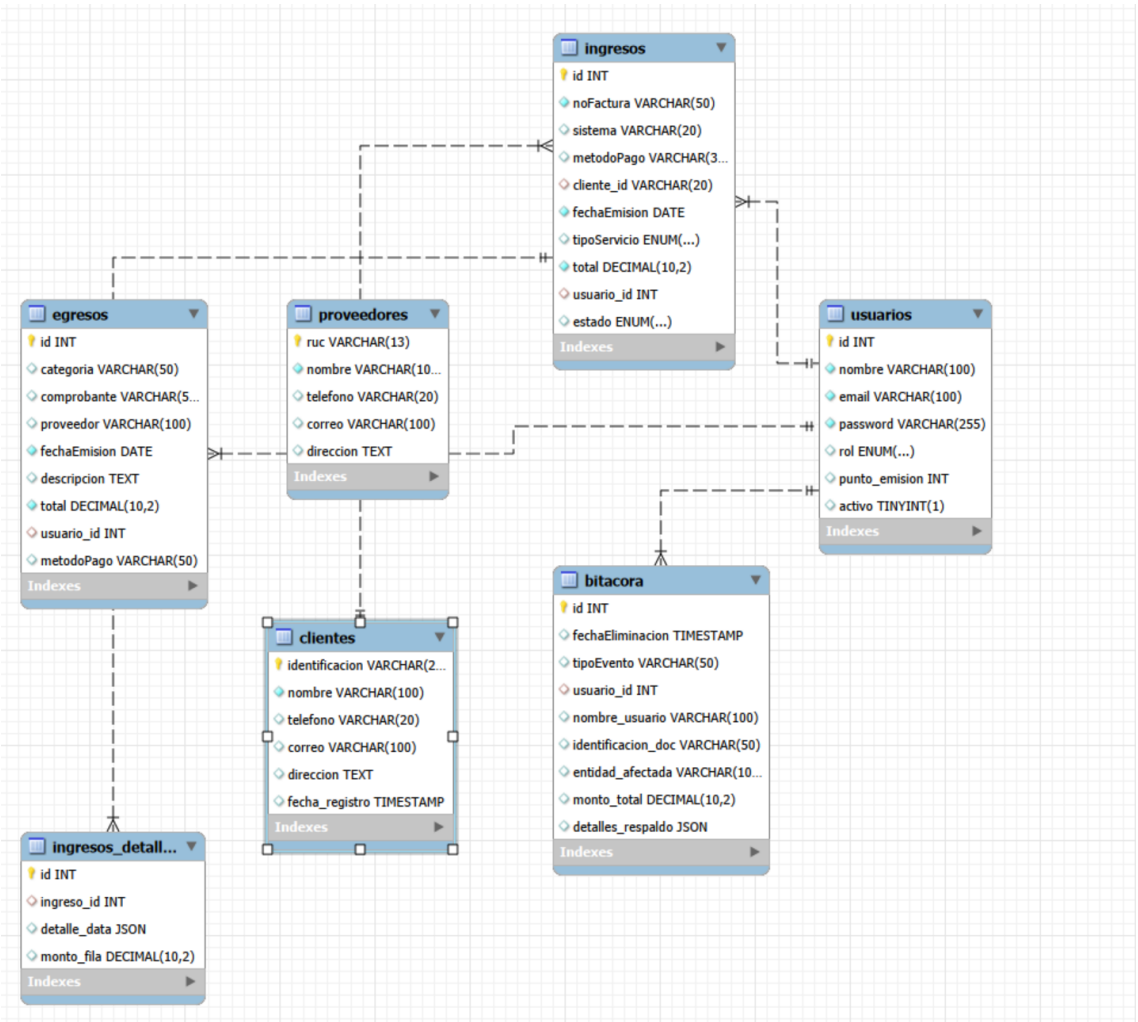


Figura 2. Diagrama Entidad-Relación

Arquitectura Tecnológica

El sistema se desarrollará utilizando JavaScript para la lógica de negocio y presentación. El backend se ejecutará en el entorno Node.js, lo cual permite un manejo eficiente de las operaciones de entrada/salidas asíncronas, haciéndolo ideal para la alta concurrencia de transacciones financieras.

El despliegue de la API REST de Node.js se realizará en un entorno de Servidor Cloud con sistema operativo Linux. Esta elección garantiza un entorno de producción estable, seguro y de bajo costo operativo.

Se utilizará Docker para empaquetar la aplicación frontend y backend en contenedores portables. Garantizando que el entorno de desarrollo sea idéntico al entorno de producción, facilitando el despliegue y la migración de manera eficxiente.

Como requerimiento crítico, se exigirá el uso del protocolo HTTPS para todas las comunicaciones entre el cliente y el servidor. Esto asegura que los datos financieros críticos viajen de forma cifrada a través de la red.

Casos de Uso

Dentro de la figura 2. Se define el caso de uso que cada rol de usuario debe seguir durante el uso del sistema.

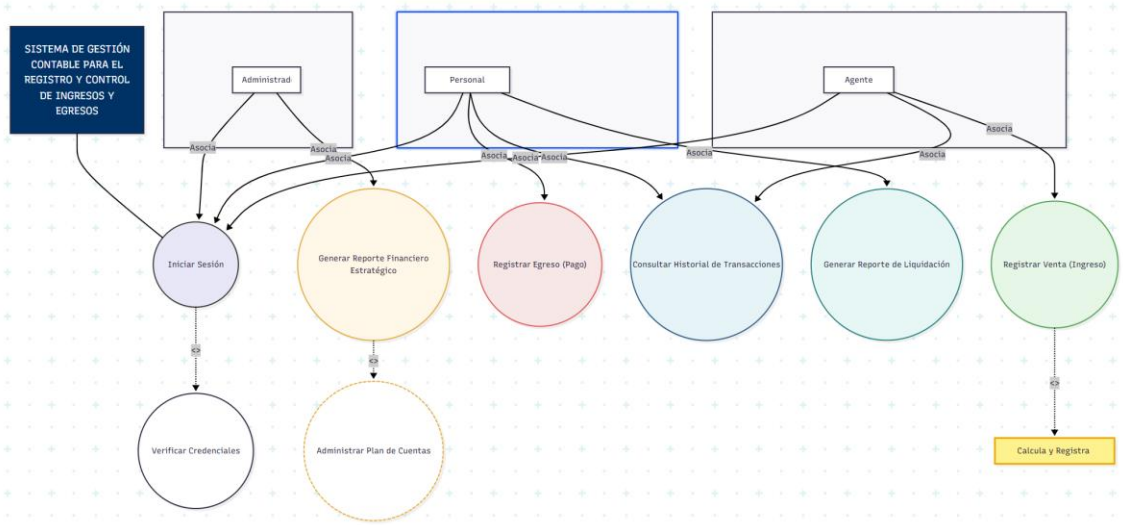


Figura 3UML de uso

Nombre	CU 1.1 Inicio de Sesión
Descripción: Permite a los empleados acceder a las funcionalidades del sistema según su rol.	
Actores:	Agente, Contador, Administrador.
Precondiciones:	El usuario debe estar registrado y en estado "Activo" en la base de datos.
Requisitos no funcionales: - Tiempo de respuesta menor a 3s. - Cifrado de sesión mediante JWT.	
Flujo de Eventos: 1. El usuario ingresa su correo electrónico y contraseña. 2. El sistema valida las credenciales y el estado activo. 3. El sistema genera un token JWT con vigencia de 4 horas. 4. El sistema redirige al usuario al Dashboard.	

Flujo Alterno: 1.1. Si las credenciales son incorrectas, el sistema muestra "Credenciales incorrectas". 1.2. Si la cuenta está inactiva, muestra "Su cuenta se encuentra inactiva".

Nombre	CU 1.2: Restablecimiento de Contraseña
Descripción: Permite a un usuario recuperar el acceso a su cuenta mediante el envío de un enlace seguro a su correo electrónico en caso de olvido de credenciales.	
Actores:	Agente, Contador, Administrador.
Precondiciones:	- El usuario debe poseer una cuenta activa con un correo electrónico institucional registrado en el sistema. - El servidor debe tener acceso a internet para el envío de correos vía SMTP (Nodemailer).
Requisitos no funcionales: - El token de recuperación debe expirar automáticamente en 15 minutos. - No se deben revelar detalles internos si el correo no existe por razones de seguridad.	
Flujo de Eventos: 1. El usuario presiona el enlace "¿Olvidaste tu contraseña?" en la pantalla de inicio de sesión. 2. El sistema despliega un modal solicitando el correo electrónico registrado. 3. El usuario ingresa su correo y presiona "Enviar Enlace". 4. El servidor valida que el correo pertenezca a un usuario activo y genera un token JWT temporal. 5. El sistema envía un correo electrónico al usuario con el enlace dinámico de recuperación. 6. El usuario accede al enlace y es redirigido a la vista de "Nueva Contraseña". 7. El usuario ingresa y confirma su nueva clave de acceso. 8. El sistema valida el token, actualiza la contraseña en la base de datos y redirige al inicio de sesión.	
Flujo Alterno: 1.1. Si el correo no coincide con un usuario activo, el sistema notifica: "El correo no pertenece a un usuario activo". 1.2. Si el usuario accede al enlace después de 15 minutos, el sistema muestra: "Enlace inválido o expirado". 1.3. Si la nueva contraseña y su confirmación son distintas, el sistema detiene el proceso y solicita la corrección.	

Nombre	CU 2.1: Registro de Ingreso (Venta)
Descripción: Captura los datos de una venta de boletos, paquetes o reservas.	
Actores:	Agente, Contador, Administrador.
Precondiciones:	Usuario autenticado; contar con la identificación del cliente.
Requisitos no funcionales: - Integridad transaccional (ACID). - Autocompletado de datos del cliente.	
Flujo de Eventos: 1. El usuario selecciona el tipo de servicio (Boleto/Paquete/Reserva).	

2. El sistema genera el número de factura automático basado en el punto de emisión. 3. El usuario ingresa la identificación del cliente; el sistema busca y carga datos si existen. 4. El usuario ingresa los detalles del servicio y montos. 5. El sistema calcula el total y guarda el registro.
Flujo Alterno: 1.1. Si faltan campos obligatorios, el sistema lanza una alerta y no permite guardar. 1.2. Si el cliente no existe, el usuario completa los datos manualmente para su registro.

Nombre	CU 2.2: Eliminación de Registro de Ingreso
Descripción: Permite la eliminación de un Ingreso, asegurando que toda la información (detalles de boletos, paquetes o reservas) se respalde en la bitácora antes de su borrado físico..	
Actores:	Agente, Contador, Administrador.
Precondiciones:	- El usuario debe estar autenticado con un token JWT. - Los agentes solo pueden disparar este flujo para ingresos de su propia autoría. - El registro de ingreso debe existir en la tabla ingresos.
Requisitos no funcionales: - Si falla el respaldo en bitácora, no se borra el ingreso. - Registro exacto del número de ingreso eliminado.	
Flujo de Eventos: 1. El usuario accede al historial de la vista IngresosView. 2. Presiona el botón de eliminar junto al registro deseado. 3. El sistema solicita confirmación mediante un cuadro de diálogo. 4. Se envía una petición POST al endpoint */api/ingresos/eliminar.* 5. El backend inicia una transacción SQL. 6. Se inserta el evento ELIMINACIÓN_INGRESO en la bitácora con los datos del ingreso en formato JSON. 7. Se ejecuta la instrucción DELETE en la tabla ingresos 8. El servidor confirma la transacción y el frontend refresca los totales globales.	
Flujo Alterno: 1.1. Si el servidor no puede escribir en la bitácora, la transacción se revierte y el registro de ingreso permanece intacto. 1.2. Si el cliente no existe, el usuario completa los datos manualmente para su registro.	

Nombre	CU 3.1: Registro de Egreso (Gasto)
Descripción: Documenta una salida de dinero para pagos a proveedores o gastos administrativos.	
Actores:	Agente, Contador, Administrador.
Precondiciones:	Usuario autenticado
Requisitos no funcionales: - Registro automático de proveedores nuevos.	

Flujo de Eventos: 1. El usuario ingresa el RUC del proveedor. 2. El sistema busca al proveedor; si no existe, solicita datos de contacto 3. El usuario ingresa fecha, descripción del gasto y monto total. 4. El usuario selecciona el método de pago. 5. El sistema guarda el egreso y actualiza el saldo global.
Flujo Alterno: 1.1. Si el monto es menor o igual a cero, el sistema rechaza el registro

Nombre	CU 3.2: Eliminación de Registro de Egreso
Descripción: Proporciona el mecanismo para anular el registro de un gasto o pago a proveedor, salvaguardando los datos originales en el historial de auditoría.	
Actores:	Agente, Contador, Administrador.
Precondiciones:	- El usuario debe estar autenticado con un token JWT. - El egreso debe estar previamente registrado en la tabla egresos.
Requisitos no funcionales: - Si falla el respaldo en bitácora, no se borra el ingreso. - Los agentes solo pueden disparar este flujo para egresos de su propia autoría. - Registro exacto del número de ingreso eliminado.	
Flujo de Eventos: 1. El usuario visualiza la tabla de egresos en EgresosView. 2. Identifica el gasto erróneo y presiona el botón de eliminar. 3. El sistema solicita confirmación mediante un cuadro de diálogo. 4. Se dispara el controlador eliminarEgreso hacia el endpoint */api/egresos/eliminar*. 5. El backend inicia una transacción SQL. 6. Se registra el evento ELIMINACIÓN_EGRESO en la tabla bitacora 7. Se procede a borrar físicamente el registro de la tabla egresos. 8. El servidor confirma la transacción y el frontend refresca los totales globales.	
Flujo Alterno: 1.1. Si un usuario intenta borrar un egreso que no le pertenece (y no posee rol privilegiado), el sistema deniega la acción desde el middleware del backend.	

Nombre	CU 4.1: Eliminación con Auditoría (Bitácora)
Descripción: Permite borrar un registro erróneo dejando un respaldo inmutable para auditoría.	
Actores:	Contador, Administrador.
Precondiciones:	El registro debe existir en la base de datos.
Requisitos no funcionales: - Respaldo obligatorio en formato JSON en la tabla bitacora.	
Flujo de Eventos: 1. El usuario selecciona un registro y presiona "Eliminar". 2. El sistema solicita confirmación de la acción. 3. El sistema copia los datos del registro a la tabla bitacora con el sello del usuario.	

4. El sistema elimina físicamente el registro de la tabla principal. 5. El sistema notifica: "Registro eliminado y auditado".
Flujo Alterno: 1.1. Si ocurre un error en la escritura de la bitácora, la eliminación se cancela (Rollback).

Nombre	CU 5.1: Gestión de Usuarios y Puntos de Emisión
Descripción: Permite crear nuevos perfiles y configurar sus puntos de venta.	
Actores:	Administrador.
Precondiciones:	Rol de Administrador activo.
Requisitos no funcionales: - Unicidad del punto de emisión y correo electrónico. - Seguridad: Solo el rol 'admin' puede ejecutar esta acción. - Tiempo de respuesta: Actualización inmediata en la interfaz de usuario.	
Flujo de Eventos: 1. El Administrador ingresa nombre, email, password y rol del nuevo empleado. 2. Se asigna un número de punto de emisión único (ej. 002). 3. El sistema valida que el email no esté duplicado. 4. El sistema elimina físicamente el registro de la tabla principal. 5. El sistema guarda al usuario en estado activo.	
Flujo Alterno: 1.1. Si el punto de emisión ya existe, el sistema muestra un mensaje de error: "Punto de emisión ya asignado"	

Nombre	CU 5.2: Desactivación / Bloqueo de Usuario
Descripción: Permite al Administrador inhabilitar el acceso de un usuario al sistema sin eliminar su historial de registros contables.	
Actores:	Administrador.
Precondiciones:	Rol de Administrador activo. El administrador debe haber iniciado sesión con éxito. El usuario seleccionado debe existir en la base de datos.
Requisitos no funcionales: - Seguridad: Solo el rol 'admin' puede ejecutar esta acción. - Tiempo de respuesta: Actualización inmediata en la interfaz de usuario.	
Flujo de Eventos: 1. El Administrador accede a la vista de "Configuración" 2. El sistema carga y muestra la lista de todos los usuarios registrados. 3. El Administrador localiza al usuario y presiona el botón de acción "Bloquear Usuario". 4. El frontend envía una petición PUT al servidor con el nuevo estado (activo: false). 5. El servidor valida el token del administrador y actualiza el campo activo en la tabla usuarios. 6. El sistema actualiza el componente visual mostrando el distintivo "Inactivo" en color rojo.	
Flujo Alterno:	

1.1. Si un usuario con rol 'agente' o 'contador' intenta forzar la petición, el servidor responde con un error 403 y el mensaje "Acceso denegado".
1.2. Si el servidor no responde, el sistema muestra una alerta: "Error al actualizar estado".

Capítulo II

Construcción del Sistema

Descripción de Módulos del sistema.

El presente SISTEMA DE GESTIÓN CONTABLE PARA EL REGISTRO Y CONTROL DE INGRESOS Y EGRESOS EN LA EMPRESA SARVALTOURS se fundamenta en una arquitectura modular que garantiza la integridad de los datos contables, la escalabilidad del software y la segregación de funciones.

Cada uno de los módulos ha sido desarrollado para responder a las necesidades específicas del proceso de gestión de ingresos y egresos de la agencia de viajes, asegurando un entorno seguro y eficiente.

Módulo de autenticación y gestión de Identidad.

El módulo de autenticación constituye la primera capa de seguridad del sistema. Su principal función es validar la identidad de los usuarios y gestionar la persistencia de la sesión mediante un standard de seguridad *"Stateless"*.

Funcionalidades clave:

- Control de acceso: Valida las credenciales (email y password) contra la base de datos mediante peticiones asincrónicas.
- Seguridad JWT: Emisión de un JSON WEB TOKEN con vigencia de 4 horas, el cual es requerido en cada petición para interactuar con la API.
- Recuperación de Credenciales: Integración con el servicio Nodemailer para enviar enlaces de restablecimiento temporales al email del usuario. (con validez de 15 minutos).
- Estado de Cuenta: Verificación inmediata del estado del usuario siendo ACTIVO o INACTIVO, restringiendo el acceso si la cuenta es inhabilitada por el administrador.

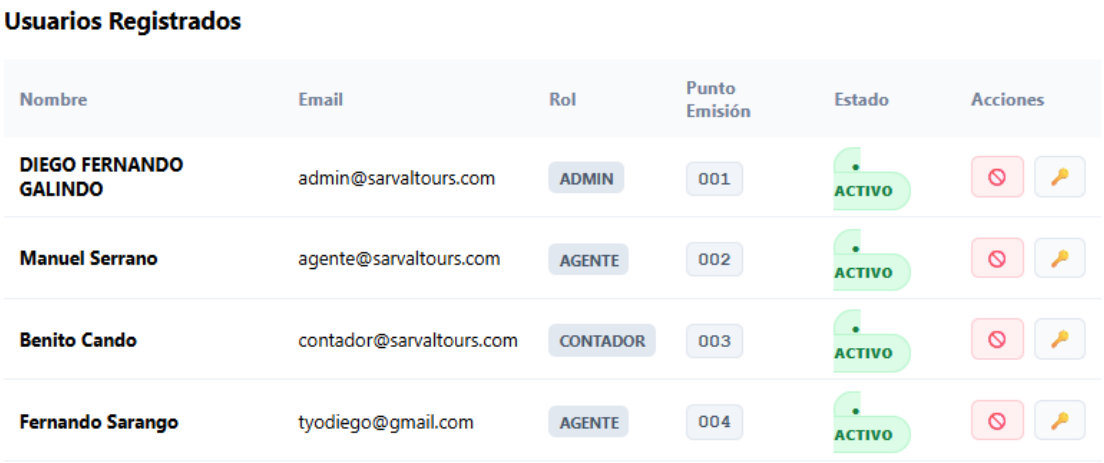


Figura 4. Módulo de autenticación y gestión de identidad

Implementación técnica:

Hace uso del controlador *useAuth,js* en el frontend y el archivo *auth.js* ubicado en el backend, gestionando la limpieza del *localStorage* para cierre de sesión seguro.

Módulo de Gestión de Ingresos (Ventas).

El módulo de gestión de ingresos es el núcleo operativo del sistema, se especializa en capturar la entrada de capital derivada de la actividad comercial, diferenciando la naturaleza técnica de cada servicio turístico prestado.

Con el fin de poder diferenciar la entrada de capital se hizo uso de sub- módulos, que nos permiten llevar un registro diferencial de a qué servicio corresponde cada ingreso siendo estos sub-módulos:

- Emisión de Boletos: Formulario especializado para transporte aéreo que registra aerolíneas, ruta, pasajero y el desglose de valores (tarifa base, Tax y fee de emisión).
- Paquetes: Registra los servicios complejos que incluyen destino, hotel, duración en días del servicio y servicios adicionales.
- Reservas: Comparte campos similares a Paquetes, es necesario la distinción debido a la logia de negocio que maneja la agencia dentro de los servicios que presta.

Lógica y Automatización:

Con la finalidad de llevar un control ordenado del número de ingresos de cada agente utilizamos un cálculo dinámico del próximo número de factura basado en el Punto de emisión asignado a cada usuario y el ultimo secuencial en la base de datos.

De igual manera el módulo utiliza una lógica de búsqueda inteligente de clientes, lo cual nos permite el consumo del **endpoint** */clientes/:id* para el auto llenado de datos fiscales al ingresar la identificación del cliente.

Implementación Técnica.

El módulo de ingresos utiliza una estructura transaccional que permite al guardar un ingreso por medio de *POST /api/ingresos*, el sistema inserte los datos en la tabla *ingresos* y, al mismo tiempo, recorre un arreglo de detalles para probar la tabla *ingresos_detalles*, esto con la finalidad de garantizar la consistencia de la información.

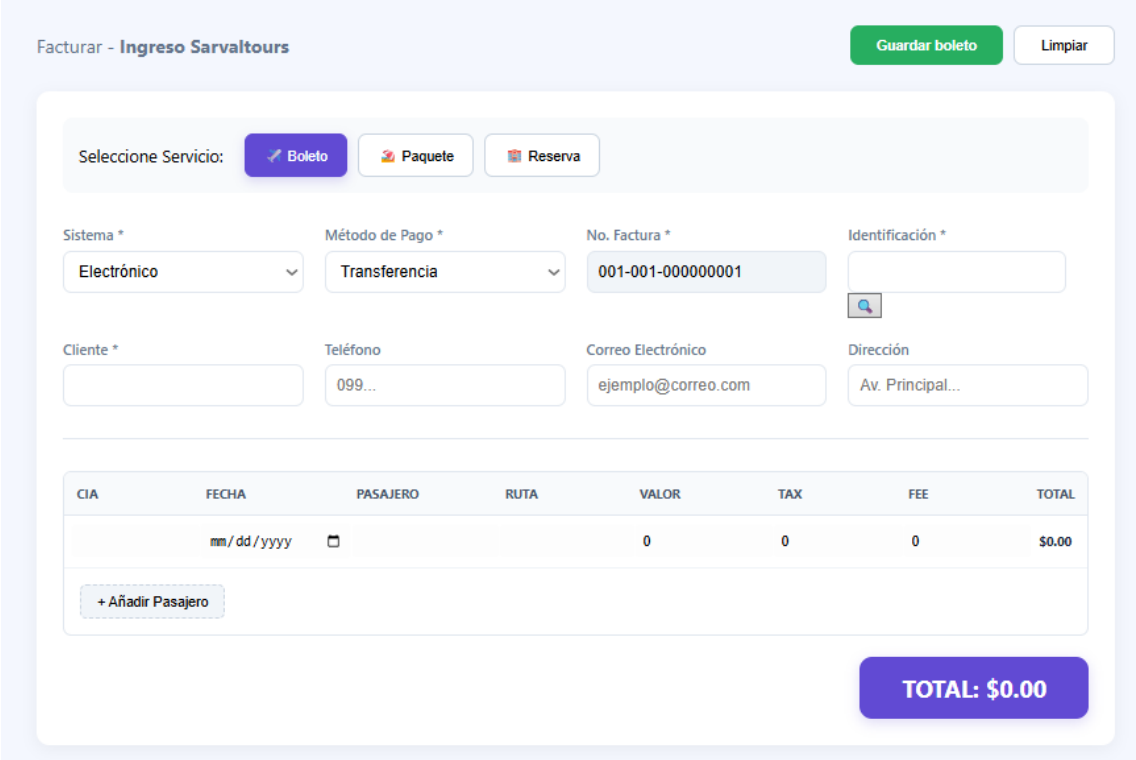


Figura 5. Módulo de Ingresos

Módulo de gestión de Egresos y Gastos Administrativos

Este módulo es el encargado de registrar todas las salidas de capital, permitiendo un control estricto sobre los costos operativos y los pagos realizados a proveedores externos.

Funcionalidades clave:

- Registro y Categorización: Captura los datos fiscales (RUC) y clasifica la naturaleza del gasto (mayorista, servicios básicos, nomina, etc).
- Asociación de Pagos: Registra el método utilizado siendo posible en: Transferencia, Efectivo, Tarjeta de crédito, Cheques).

Implementación Técnica:

El módulo de gestión de Egresos y gastos administrativos se encuentra gestionado por el controlador egresos.js, el mismo que incluye una lógica de validación que permite detectar si un proveedor es nuevo o ya existente, en caso de detectar un nuevo proveedor, el sistema lo registra de forma automática en la base de datos antes de procesar el egreso, optimizando la integridad referencial.

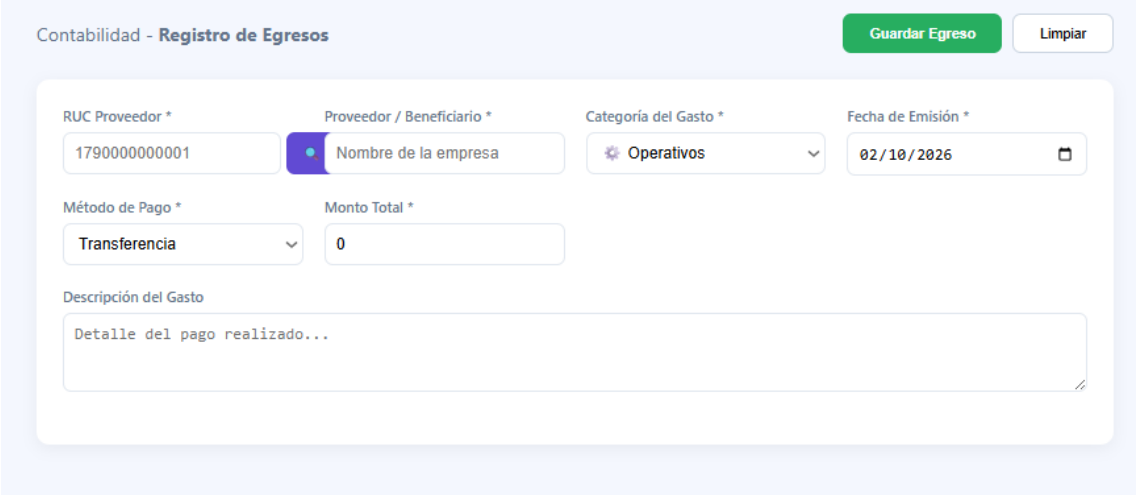


Figura 6. Modulo de Egresos.

Módulo de Auditoria e Inmutabilidad (Bitácora)

Este módulo actúa como un componente de control interno crítico, asegurando que ninguna transacción sea eliminada sin dejar rastro, lo cual provee al sistema de una capa extra de seguridad.

Mecanismos de Control:

- Respaldo de datos: Al ejecutar una eliminación, sea esta de un ingreso o egreso, el sistema extrae el objeto completo en formato JSON y lo almacena en la tabla *bitácora* antes de su destrucción física en la tabla principal.
- Trazabilidad: Almacena los datos del usuario responsable de la eliminación, al igual que la fecha exacta, el monto afectado y los detalles de respaldo.

Implementación técnica:

El módulo utiliza transacciones SQL automáticas que garantizan que el registro en la bitácora y la eliminación del dato original ocurran siempre en un solo paso, impidiendo de esta manera registros “huérfanos”.

Figura 7. Modulo de Auditoria.

Módulo de Dashboard e Inteligencia de negocio.

El módulo de Dashboard proporciona una interfaz ejecutiva para la toma de decisiones basada en la transformación de registros crudos en información visual estratégica.

Capacidades Analíticas:

- Métricas en Tiempo Real: Visualización comparativa de totales mensuales y anuales de flujo efectivo.
- Análisis Dinámico: Gráficos de barras interactivas, que muestran la relación entre ventas y gastos mediante filtros de fechas personalizados.

Implementación Técnica:

Mediante el archivo *Dashboard.js* se realizan las consultas mediante el uso de *UNION ALL* y *DATE_FORMAT* en SQL para unificar movimientos de distintas tablas, haciendo entrega al *frontend* de datos listos para generar la renderización de las gráficas.

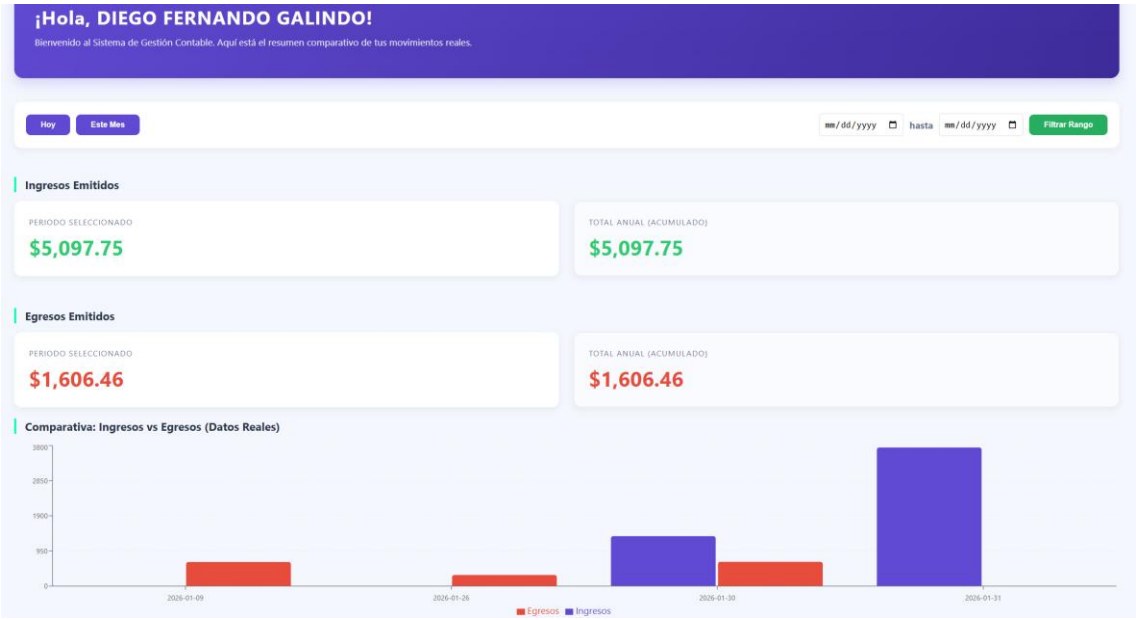


Figura 8. Módulo de Dashboard.

Módulo de Administración y Configuración de Personal.

El módulo de administración y configuración de personal es un módulo exclusivo para el usuario con rol Administrador, tiene como fin la gestión del capital humano y los parámetros técnicos del sistema.

Gestión de perfiles:

- Creación y edición de usuarios bajo roles de Agente, Contador o Administrador.
- Asignación de Puntos de emisión únicos para el control de la numeración legal de ingresos.
- Control de seguridad: El módulo cuenta con la capacidad de bloquear o desbloquear usuarios y poder gestionar las contraseñas.

Implementación Técnica:

El módulo de Administración y configuración de Personal se gestiona mediante el uso de *ConfiguracionView.jsx,* el cual se encarga de validar en tiempo real que no existan correos electrónicos o puntos de emisión duplicados antes de persistir la información en el servidor.

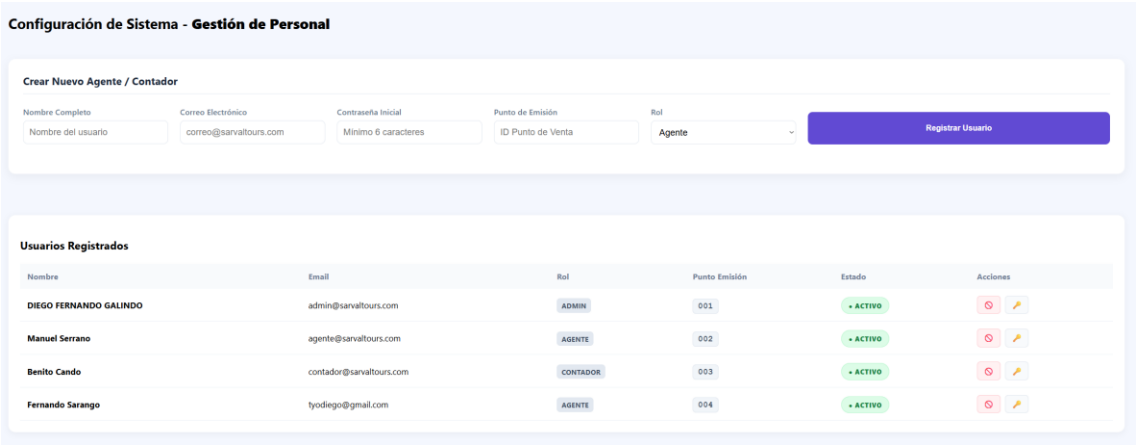


Figura 9. Módulo de Administración y Configuración

Estructura a nivel de código y patrones de diseño.

El SISTEMA DE GESTIÓN CONTABLE PARA EL REGISTRO Y CONTROL DE INGRESOS Y EGRESOS EN LA EMPRESA SARVALTOURS implementa una arquitectura desacoplada basada en el paradigma Cliente-Servidor. Esta elección técnica fue utilizada en el desarrollo debido a la posibilidad de escalabilidad futura del; sistema, al igual que facilitar el mantenimiento del sistema a futuro.

Frontend: Implementación del Patron MVC en React.

Aunque React es una librería orientada a componentes y no establece de forma obligatoria la implementación del patrón Modelo-Vista-Controlador (MVC), en el presente proyecto se adoptó una arquitectura basada en dicho patrón con el propósito de estructurar de manera eficiente la lógica del frontend y garantizar una adecuada separación de responsabilidades.

En esta implementación:

- Los **Modelos**, ubicados en el directorio src/Models/, representan la estructura y consistencia de los datos del sistema, mediante la definición de clases y constructores que encapsulan la información y las validaciones necesarias, como se evidencia en el manejo de sesiones de usuario.
- Las **Vistas**, localizadas en src/Views/, corresponden a componentes React encargados exclusivamente de la presentación visual y el diseño de la interfaz de usuario, utilizando JSX y hojas de estilo, sin incorporar lógica de negocio ni gestión directa del estado.
- Los **Controladores**, implementados a través de *Custom Hooks* en el directorio src/Controllers/, actúan como intermediarios entre los Modelos y las Vistas, gestionando el estado de la aplicación, las transformaciones de datos y las interacciones con servicios externos, tales como las llamadas a la API.

Esta arquitectura contribuye a mejorar la mantenibilidad, escalabilidad y reutilización del código, facilita la realización de pruebas unitarias y reduce el acoplamiento entre la

lógica de negocio y la capa de presentación, los cuales son aspectos fundamentales en el desarrollo de aplicaciones web de mediana y gran complejidad.

Backend: Modelo de Rutas y API RESTful.

La infraestructura del servidor de SARVALTOURS se ha implementado utilizando Node.js y el framework Express, bajo un enfoque de API RESTful (Representational State Transfer). A diferencia de los modelos monolíticos tradicionales, donde el servidor asume la carga de renderizado de vistas, este sistema delega la lógica de presentación al cliente y se especializa exclusivamente en el procesamiento de recursos y la persistencia de datos.

Arquitectura Stateless y Comunicación asincrónica.

El backend opera bajo una filosofía de arquitectura sin estado (stateless). Esto significa que el servidor no almacena información de sesión en memoria, lo que permite una mayor escalabilidad horizontal.

- Protocolo de Intercambio: La comunicación se realiza estrictamente mediante el formato JSON, asegurando la interoperabilidad entre el frontend y el backend.
- Gestión de Sesiones: La autenticación se delega a JSON Web Tokens. Cada petición HTTP enviada por el cliente es autodescriptiva y contiene las credenciales necesarias en la cabecera *Authorization*, permitiendo al servidor validar la identidad del usuario en tiempo real mediante algoritmos de firma criptográfica.

Entidades y encapsulamiento de rutas.

Se ha aplicado el principio de *Separación de Incumbencias (Separation of Concerns)* mediante el uso de *express.Router()*. La lógica de la aplicación no reside en un único archivo, sino que se segmenta en controladores de rutas específicos para cada entidad del dominio:

- Rutas de Autenticación (auth.js): Gestiona los procesos de *handshake* inicial, generación de tokens y lógica de recuperación de cuenta.
- Rutas Financieras (ingresos.js, egresos.js): Encapsulan los métodos CRUD (Create, Read, Update, Delete) para los movimientos contables.
- Rutas de Auditoría (bitacora.js): Endpoint especializado para la recuperación de logs de seguridad.

Esta estructura permite que el archivo principal index.js actúe únicamente como el punto de entrada, donde se registran los *middlewares* globales y se definen los prefijos de las rutas *(ej. /api/auth, /api/ingresos)*, facilitando el mantenimiento y la implementación de pruebas unitarias.

Capa de Middleware y Pipeline de peticiones.

El sistema implementa una arquitectura basada en middlewares, que interceptan el ciclo de solicitud-respuesta antes de que la petición alcance la lógica de negocio.

- Control de Acceso: El middleware *verificarToken* se encarga de validar la integridad del token JWT y extrae los privilegios del usuario, permitiendo o denegando el acceso a rutas sensibles según el modelo de Control de Acceso Basado en Roles.
- Validación de Datos: Antes de interactuar con la base de datos, los middlewares validan que el cuerpo de la petición (*req.body*) cumpla con los esquemas requeridos, evitando inyecciones de datos malformados.

Gestión de persistencia y transaccionalidad.

La conexión con MySQL se gestiona a través de un Pool de Conexiones optimizado, que permite la reutilización de hilos y mejora el rendimiento bajo concurrencia.

- Integridad de Datos: En procesos críticos como el registro de ingresos, en el que se afecta la tabla maestra y la de detalles, el backend implementa Transacciones ACID *(connection.beginTransaction())*. Esto asegura que el registro sea atómico: o todas las operaciones se confirman exitosamente, o se realiza un rollback completo ante cualquier fallo, garantizando que la contabilidad de la empresa nunca presente inconsistencias o registros huérfanos.

Tabla comparativa de arquitecturas (FRONTEND Y BACKEND).

CARACTERISTICAS	**Frontend (React MVC)**	**Backend (express Routes)**
Responsabilidad	Gestión de estados y experiencia de usuario.	Persistencia de datos y lógica de negocio.
Comunicación	Peticiones HTTP (fetch)	Respuestas JSON y códigos de estado HTTP.
Seguridad	Manejo de Tokens en *localStorage.*	Validacion de JWT y transacciones SQL.
Logica	Centrada en Hooks y Context.	Centrada en Rutas y Controladores de BD.

Tabla 5: comparación de arquitecturas.

Seguridades, Permisos y Roles.

El sistema garantiza la integridad de la información mediante capas de seguridad lógica:

- Autenticación JWT**:** Se utiliza el estándar *JSON Web Token*. El servidor firma un token con un secreto JWT_SECRET, que el cliente debe enviar en cada cabecera de petición (*Authorization: Bearer ...*).
- Roles de Usuario:
    1. **Administrador:** Acceso total, gestión de usuarios y visualización de bitácora completa.
    2. **Contador:** Acceso a registros financieros, generación de informes y estadísticas de todos los agentes.
    3. **Agente:** Solo puede registrar y visualizar sus propios movimientos y ventas.
- Seguridad a nivel de Base de Datos: Se utilizan Pools de conexiones y Transacciones SQL *(connection.beginTransaction())* en procesos críticos como el registro de ingresos. Esto asegura que, si algo falla durante el guardado de los detalles, no se cree un registro huérfano, manteniendo la consistencia de los datos.

Interacción entre Módulos y Flujo de Datos.

La interacción entre los módulos del SISTEMA DE GESTIÓN CONTABLE PARA EL REGISTRO Y CONTROL DE INGRESOS Y EGRESOS EN LA EMPRESA SARVALTOURS se rige por un modelo de comunicación bidireccional entre la capa de presentación y la capa de servicios. Lo que garantiza que cualquier cambio en el estado financiero de la empresa se refleje de manera inmediata y consistente en todos los componentes del software.

Comunicación Asíncrona vía API RESTful

La interacción primaria se produce a través de peticiones HTTP gestionadas de forma asíncrona.

- Punto de Origen: El usuario interactúa con una Vista (ej. Registro de Ingresos).
- Mediador: El Controlador captura los datos y dispara una petición asíncrona hacia el Backend.
- Procesamiento: El servidor recibe la petición, la valida mediante el Middleware de Seguridad y ejecuta la lógica de negocio.
- Sincronización: Una vez que la base de datos confirma la operación, el servidor devuelve un código de estado (ej. 201 Created) y el objeto procesado, permitiendo que el Frontend actualice su interfaz sin necesidad de refrescar la página.

Gestión de Estado Global (React Context API)

Uno de los puntos de interacción más críticos ocurre dentro del Frontend mediante el *ContableContext*. Este componente se encarga de eliminar el acoplamiento excesivo entre vistas.

- Sincronización Automática: Cuando el módulo de *Egresos* registra un nuevo gasto, invoca la función *cargarDatos()* del contexto. Esta acción actualiza el estado global de la aplicación.
- Efectos Colaterales: Debido a que el módulo de *Dashboard* y el de *Informes* están suscritos a este mismo contexto, reaccionan automáticamente al cambio, recalculando los totales y actualizando los gráficos en tiempo real sin intervención adicional del usuario.

Flujo Transaccional: Ingresos y Gestión de Clientes

La interacción entre el Módulo de Ingresos y el Módulo de Clientes es de naturaleza consultiva y transaccional:

1. Al ingresar una identificación en el formulario de ventas, el sistema dispara una consulta al endpoint */api/usuarios/clientes/:id.*
2. Si el módulo de usuarios devuelve un registro positivo, los campos de nombre, teléfono y dirección se autocompletan.
3. En caso de que la respuesta sea negativa, el módulo de ingresos permite la creación del cliente en la tabla *caliente*, interactuando con la base de datos antes de finalizar la transacción de venta.

Integración de Auditoría: Interacción entre Ingresos, Egresos y Bitácora

La interacción entre los módulos financieros y el de Bitácora se basa en un flujo de intercepción. No es posible eliminar un registro de manera aislada; el sistema impone una relación de dependencia obligatoria:

- Intercepción: La petición de eliminación es capturada por el controlador de *egresos/ingresos* en el backend.
- Persistencia Doble: Antes de ejecutar la instrucción DELETE en SQL, el sistema invoca una función interna que serializa el registro original y lo inserta en la tabla de bitácora.
- Finalización: Solo si la inserción en auditoría es exitosa, se procede con la eliminación del registro contable, garantizando que no exista pérdida de información sin su respectivo respaldo.

Enlace entre roles y módulos.

El Token de Acceso JWT actúa como puente que permite la interacción legítima entre módulos. Sin este token, los módulos se encuentran aislados por una capa de Middleware de Seguridad.

- Cada interacción que requiera datos sensibles debe pasar por un proceso de validación de firma y expiración.
- El token transporta el Rol del Usuario, lo que determina dinámicamente qué módulos pueden "hablar" entre sí. Por ejemplo:
  El módulo de Configuración no puede interactuar con un usuario con un rol de "Agente", cerrando el flujo de datos a nivel de servidor.

Tabla de Independencias técnicas:

Módulo de Origen	Módulo de Destino	Tipo de interacción	Canal Utilizado
Ingresos	Dashboard	Actualización de Estado	ContableContext
Egresos	Bitácora	Registro de Auditoria	Transacción SQL
Autenticación	Todos los Módulos	Autorización de Acceso	JWT / HEADERS
Dashboard	Backend	Consulta Analítica	API REST (GET)
Configuración	Usuarios	Gestión de Identidades	API REST (PUT/POST)

Tabla 6: Independencias Técnicas.

Arquitectura de Datos y Modelo Relacional

Debido a que MySQL utiliza por defecto el motor de almacenamiento InnoDB, el cual proporciona soporte avanzado para transacciones ACID, se seleccionó este sistema gestor de bases de datos relacional para el desarrollo del Sistema de Gestión Contable para el Registro y Control de Ingresos y Egresos en la empresa SARVALTOURS. Esta elección garantiza la integridad, consistencia y confiabilidad de la información almacenada, aspectos fundamentales para el presente desarrollo.

Descripción de las Entidades y Diccionario de Datos

El presente modelo utilizado se divide en tres categorías funcionales: Maestros, Transaccionales y de Control.

Entidades Maestras:

- Usuarios: Centraliza el acceso al sistema. Incluye el campo rol para el control de acceso y el punto_emision (UNIQUE), que vincula al usuario con un local físico de emisión de ingresos.

- Clientes: Utiliza la identificación como Primary Key (PK), facilitando la búsqueda rápida por cédula o RUC sin necesidad de IDs autoincrementales adicionales.
- Proveedores: Almacena la información fiscal de las entidades a las que la agencia realiza pagos, utilizando el RUC como identificador primario.

Entidades Transaccionales:

- Ingresos: Registra las ventas. La restricción de unicidad en noFactura garantiza que no existan documentos duplicados ante la autoridad tributaria.
- Ingresos_detalles: Implementa una relación de Maestro-Detalle con la tabla Ingresos. Destaca el uso del tipo de dato JSON para el campo *detalle_data*, permitiendo almacenar estructuras flexibles, en este caso: vuelos, hoteles o tours sin alterar el esquema de la base de datos.
- Egresos: Captura las salidas de dinero, vinculando cada gasto a un usuario responsable para fines de auditoría financiera.

Entidades de Control:

- Bitácora: Se encarga de la persistencia de registros eliminados. Almacena la imagen del objeto borrado en formato JSON, *detalles_respaldo*, asegurando que la información original nunca se pierda totalmente.

Restricciones de Integridad y Lógica de Negocio

Las restricciones actúan como reglas de negocio inquebrantables a nivel de motor de base de datos:

- Foreign Keys: Todas las transacciones están ancladas a un *usuario_id* y a un *cliente_id*. Esto impide la existencia de facturas "huérfanas" y permite generar reportes de rendimiento por empleado con precisión.
- Integridad de Dominio: El uso de campos ENUM para *rol* y *tipoServicio* limita los valores posibles a los permitidos por la lógica del negocio, evitando errores de entrada de datos a nivel de servidor.
- Eliminación en Cascada: La tabla *ingresos_detalles* utiliza la instrucción ON DELETE CASCADE. Esto garantiza que, si un ingreso es eliminado, luego de ser respaldado en la *bitácora*, todos sus detalles asociados desaparezcan automáticamente, manteniendo la limpieza del almacenamiento.
- Unique Constraints:
  El campo *email* en Usuarios: Evita duplicidad de cuentas.
  punto_emision en Usuarios: Asegura que dos agentes no emitan facturas bajo el mismo código de terminal.
  noFactura en Ingresos: Garantiza la correlatividad legal de los documentos.

Tabla de tecnologías utilizadas:

Componentes	Tecnología	Justificación Técnica
Entorno de ejecución	Node.js	Asincronía y alta concurrencia para peticiones API.
Base de Datos	MySQL	Consistencia relacional y manejo de transacciones.
Seguridad	JWT y Nodemailer	Autenticación descentralizada y comunicación segura.
Interfaz de Usuario	React	Modularidad mediante componentes y estado global eficiente.
Visualización	Recharts	Representación de datos complejos mediante SVG dinámicos.

Tabla 7: tecnologías utilizadas y justificación técnica.

Metodología de Desarrollo: Scrum.

Con la finalidad de organizar el desarrollo del SISTEMA DE GESTIÓN CONTABLE PARA EL REGISTRO Y CONTROL DE INGRESOS Y EGRESOS EN LA EMPRESA SARVALTOURS se utilizará la metodología ágil SCRUM, la cual ofrece una ventaja competitiva vital para este desarrollo, la capacidad de adaptar el software de manera iterativa a las necesidades financieras reales de la agencia.

Scrum nos permite separar la complejidad del control de ingresos y egresos en ciclos cortos de desarrollo, Sprints, lo que facilita una visibilidad constante del progreso, así como la detección temprana de errores o cambios en los requerimientos.

Al involucrar de manera activa al Product Owner y al Scrum Master en el proceso de revisión, se ha podido asegurar que el producto final sea técnicamente robusto y cumpla a la perfección los objetivos del negocio y los estándares de calidad exigidos. Al respecto, Schwaber y Sutherland (2020) definen este marco como una solución diseñada para generar valor de manera adaptable ante problemas complejos. Bajo esta premisa, la implementación de Scrum en el sistema de SarvalTours no solo organiza el trabajo técnico, sino que asegura que cada iteración del software entregue una herramienta funcional y alineada a la realidad financiera de la empresa, permitiendo ajustes dinámicos que un modelo rígido no podría soportar.

Roles del Equipo SCRUM.

Con la finalidad de cumplir con los requerimientos de la agencia de viajes Sarvaltours, los siguientes roles han sido definidos:

- Product Owner: Se encuentra representado por el administrador responsable de la agencia Sarvaltours. Su función principal es la definición de las prioridades del Product Backlog, se asegura de que el sistema responda a las necesidades de registro de ingresos y egresos de la empresa.
- Scrum Master: Función desempeñada por el Ing. Jose Luis Galarraga, quien como director del proyecto de tesis, orienta y acompaña el desarrollo del trabajo,

garantizando la correcta aplicación metodológica, la solución de dificultades técnicas y el cumplimiento de los criterios académicos establecidos.

- Equipo de desarrollo: Integrado por Diego Fernando Galindo, quien es responsable de: la arquitectura del sistema, el desarrollo de módulos solicitados, implementación de la base de datos MySQL y ejecución de pruebas de calidad necesarias para el presente sistema.

Planificación de Sprint.

La ejecución del proyecto se encuentra estructurada en ciclos de trabajo denominados SPRINTS, los cuales cuentan con una duración promedio de dos semanas cada uno. Este enfoque permite la transformación del cronograma de actividades en entregables funcionales y medibles.

- **Fase de Definición conformada por el Sprint 1 y Sprint 2:**
  En esta etapa se efectuó el levantamiento y análisis de requisitos funcionales y no funcionales, así como el diseño de la arquitectura del sistema, definiendo los componentes, capas y tecnologías involucradas. Como resultado, se estableció una estructura técnica consensuada que garantiza la correcta implementación del módulo de control contable.

- **Fase de Prototipado y Desarrollo Principal conformada por el Sprint 3 y el Sprint 4:**
  En esta etapa se realizó la configuración del entorno de desarrollo, incluyendo herramientas, frameworks y repositorios, y se llevó a cabo la implementación de los módulos principales del sistema. La retroalimentación continúa proporcionada por el Product Owner facilitó la validación oportuna del sistema.

- **Fase de implementación y Seguridad conformada por el sprint 5 y el Sprint 6:**
  Esta estuvo orientada al desarrollo y refinamiento de la interfaz de usuario, así como a la implementación de mecanismos de recuperación de credenciales y validación de datos. Adicionalmente, se estableció un ciclo sistemático de detección y corrección de errores, con el objetivo de garantizar la integridad, consistencia y confiabilidad de la información financiera procesada por el sistema.

- **Fase de Estabilización conformada por el sprint 7:**
  Esta fase estuvo orientada a la realización de ajustes finales sobre los componentes del sistema y a la ejecución de pruebas unitarias, con el propósito de validar el correcto funcionamiento de cada módulo. Este incremento final permitió fortalecer la estabilidad y robustez del software, garantizando su adecuado desempeño previo a la entrega y presentación del proyecto.

Eventos y Artefactos.

Con la finalidad de supervisar y gestionar los avances del proyecto, se emplearon los siguientes artefactos y eventos de control dentro del marco de la metodología ágil SCRUM:

- Sprint Planning: Reuniones de carácter quincenal destinadas a la planificación y definición del proyecto, en las cuales se seleccionaba y desglosaba el conjunto de tareas priorizadas del cronograma general, estableciendo objetivos, estimaciones de esfuerzo y asignación de actividades.
- Product Backlog: Artefacto principal de SCRUM que consolidó el listado completo de requerimientos funcionales y no funcionales solicitados por SarvalTours, incluyendo procesos como el registro de transacciones contables, gestión de información financiera y generación automatizada de reportes.
- Sprint Review: Reuniones formales de evaluación de los incrementos desarrollados, en las cuales se verificó, junto con al Scrum Master y el Product Owner, que los entregables alcanzaran los criterios de aceptación establecidos.

Capítulo III

Pruebas y Estabilización

Dentro de este capítulo se detallan las acciones técnicas y metodológicas implementadas con el propósito de asegurar la calidad, coherencia, protección y funcionamiento adecuado del SISTEMA DE GESTIÓN CONTABLE PARA EL REGISTRO Y CONTROL DE INGRESOS Y EGRESOS EN LA EMPRESA SARVALTOUR.

Las acciones anterior mente mencionadas fueron desarrolladas bajo la metodología SCRUM y se concentraron principalmente en las fases finales de los Sprints, cuyo propósito fundamental fue obtener un producto solido y confiable. Listo para su implementación en un entorno productivo real.

El proceso de pruebas y estabilización permitió comprobar que el sistema satisface tanto los requisitos funcionales como los no funcionales, reduciendo posibles riesgos operativos y garantizando una experiencia de uso eficiente y adecuada para el personal administrativo de la agencia.

Pruebas unitarias

Con el propósito principal de validar el funcionamiento correcto de cada componente lógico del sistema, de manera aislada, garantizando que la lógica de negocio implementada respondiera fielmente a los requerimientos definidos durante la fase de análisis, se implementaron pruebas unitarias, lo que permitió identificar errores en etapas tempranas del desarrollo, reduciendo la posibilidad de fallos en fases posteriores de desarrollo, lo cual permitió fortalecer la estabilidad del sistema.

Realizamos la evaluación de funciones y módulos principales sin depender de otros componentes del sistema, asegurando así que cada unidad cumpliera con su responsabilidad de forma predecible.

Implementación de las Pruebas

Las pruebas unitarias fueron realizadas e implementadas utilizando el framework JEST, debido a su eficiencia y facilidad de integración en Node.js. Estas pruebas se ejecutaron de forma directa en controladores, middleware y módulos de lógica del sistema, lo que permitió validar la lógica interna, así como la interacción con otros componentes.

Alcance de las Pruebas unitarias

El enfoque central de las pruebas unitarias fue revisar que no exista cualquier inconsistencia a nivel de los componentes principales como los componentes contables, ya que, en caso de existir inconsistencias, se vería afectada directamente la integridad de la información financiera.

En particular se evaluaron los siguientes aspectos:

- **Funciones de calculo de ingresos, egresos y saldos netos:** Se verifico que las operaciones matemáticas de ejecutaran correctamente bajo distintos escenarios y que los resultados reflejaran valores precisos.
- **Validación de reglas contables básicas:** Se verifico la imposibilidad de registrar egresos o ingresos con montos cero o negativos, así como el control de accesos a información financiera según el rol del usuario.
- **Manejo adecuado de valores decimales y redondeos financieros:** Estas pruebas son fundamentales para mantener la precisión necesaria para este tipo de sistemas.
- **Lógica de generación de ingresos:** Se comprobó que el sistema siguiera la lógica estipulada para la generación de ingresos, respetando la numeración por número de agente y serialización.

Durante las pruebas se verifico que:

- Los datos recibidos desde el frontend cumplieran con los tipos de formatos y restricciones definidos, evitando errores por entradas invalidas
- Las operaciones de inserción, consulta y actualización en la base de datos MySQL se realizan correctamente lo cual garantiza la persistencia de la información.
- Los mecanismos de seguridad como la autenticación y autorización mediante json web tokens, funciona correctamente, bloqueando accesos no autorizados y protegiendo rutas sensibles del sistema.
- Las reglas de control de usuarios, como la imposibilidad de que un administrador se desactive a si mismo se aplican de manera estricta.

Pruebas unitarias realizadas:

Con el fin de comprobar el correcto funcionamiento de la lógica de negocio del sistema, se diseñaron y ejecutaron pruebas unitarias puntuales enfocadas en validar los procesos de generación de Ingresos, la administración de secuenciales y el tratamiento de fechas, todo ello de forma independiente a servicios externos y sin interacción directa con la base de datos.

De manera específica, dichas pruebas permitieron evaluar los siguientes aspectos:

- La correcta incorporación de ceros a la izquierda en los valores secuenciales de las facturas, garantizando una longitud constante de nueve caracteres.
- La construcción adecuada del identificador de factura bajo el esquema XXX-XXX-XXXXXXXXX, en cumplimiento con los lineamientos definidos.
- La correcta segmentación y análisis de los componentes de una factura previamente generada, así como la simulación del aumento progresivo del número secuencial.
- La extracción precisa y validación del formato de fecha a partir de cadenas en formato ISO, asegurando el cumplimiento del patrón YYYY-MM-DD.

Resultados obtenidos de las pruebas unitarias.

Las pruebas se llevaron a cabo de manera exitosa entre el 19 y el 25 de enero, completándose la ejecución de cinco suites de prueba correspondientes a los módulos de autenticación, ingresos, egresos, gestión de usuarios y lógica de negocio. En conjunto, se realizaron 15 pruebas, todas con resultados favorables, sin presentarse errores ni fallas críticas, lo que demostró una cobertura adecuada y un alto grado de confiabilidad del sistema.

Durante el proceso se identificaron pequeñas inconsistencias vinculadas al manejo de valores decimales, las cuales fueron solucionadas de forma inmediata mediante la optimización de la lógica de cálculo y los criterios de redondeo. Estas mejoras fortalecieron la exactitud de los resultados financieros y elevaron la confiabilidad del sistema contable.

En conclusión, las pruebas unitarias permitieron comprobar de manera eficaz la estabilidad, exactitud y seguridad del sistema, consolidándose como un elemento clave para asegurar un producto sólido, confiable y listo para su posterior implementación en un entorno de producción real.

Pruebas de integración y de sistema:

Una vez realizadas las pruebas correspondientes al correcto funcionamiento de los módulos de forma individual se realizó la evaluación de la interacción entre los distintos componentes del sistema. Para lo cual durante esta etapa se tuvo como objetivo la verificación de la operabilidad del sistema, de forma integrada y consistente, dentro de un entorno lo mas cercano posible a condiciones reales de uso.

La intervención del Scrum Master fue vital durante esta fase debido a que permitió validar el flujo adecuado de información entre el frontend y el backend, así como la correcta ejecución de los procesos completos que forman parte de la lógica del negocio.

Pruebas de Interfaz de usuario:

Durante las pruebas de interfaz de usuario se verifico que:

- Los formularios de registro y de gestión contable del sistema Sarvaltours presenten una interfaz intuitiva y de fácil comprensión para el usuario.
- Los mensajes de validación, error y confirmación fueran claros, precisos y oportunos.
- La navegación entre las diferentes pantallas se ejecutará de forma continua, sin interrupciones ni comportamientos inesperados.

Verificación de Seguridad en la Capa de Autenticación

Para llevar a cabo las pruebas de integración, se estableció un entorno de desarrollo local apoyado en el entorno de ejecución Node.js. La interacción entre la API y la herramienta

de pruebas Postman se efectuó mediante el protocolo HTTP a través del puerto local localhost:3000, lo que permitió verificar de manera directa las respuestas emitidas por el servidor, así como la correcta persistencia de los datos en el sistema de gestión de base de datos MySQL.

Postman fue utilizada para ejecutar pruebas funcionales sobre los servicios REST proporcionados por la API. En este proceso, se analizaron los mecanismos de seguridad mediante el envío de encabezados de autorización que incluían tokens JWT, corroborando que la lógica de negocio implementada y las restricciones definidas en la base de datos MySQL mantuvieran la consistencia de los saldos ante la concurrencia de múltiples solicitudes.

Durante la ejecución de las pruebas de integración con Postman, se verificó que el middleware de seguridad denegara adecuadamente las solicitudes que carecían de credenciales válidas o que intentaban acceder a los recursos del sistema sin cumplir con la estructura de datos establecida. Tal como se presenta en el Anexo 5, el sistema devolvió el código de estado 401 Unauthorized, garantizando que únicamente el personal autorizado de la agencia SarvalTours pueda registrar y administrar los movimientos financieros del sistema.

Pruebas de base de Datos.

En estas pruebas se valido la correcta persistencia e integridad de la información almacenada, asegurando que:

- Cada registro de ingresos o egresos se almacenará correctamente en la base de datos.
- Los reportes contables se generarán de manera consistente, evitando perdida de datos e inconsistencias.
- Se mantiene la integridad referencial entre las tablas relacionadas.

Evaluación de Rendimiento y Carga del Sistema

Con el propósito de validar la solidez y el correcto desempeño del Sistema de Gestión Contable SarvalTours frente a escenarios de alta concurrencia, se incorporó la herramienta k6 dentro del proceso de pruebas. Esta incorporación respondió a la necesidad de garantizar que el sistema sea capaz de procesar transacciones financieras críticas de forma simultánea, sin afectar la integridad de la base de datos MySQL ni la disponibilidad de los servicios expuestos por la API.

Justificación y Rol de la Herramienta k6

La herramienta k6 fue seleccionada por tratarse de un framework de código abierto orientado específicamente a la evaluación de rendimiento. En el contexto del presente proyecto, su función principal consistió en la automatización de pruebas de carga mediante la simulación de tráfico concurrente a través de scripts desarrollados en

JavaScript. Esto permitió analizar el tiempo de respuesta del servidor Node.js y la estabilidad de las conexiones con la base de datos bajo condiciones de estrés. Adicionalmente, k6 facilitó su integración con la arquitectura del sistema al soportar mecanismos modernos de autenticación, como el uso de tokens JWT empleados para la validación de solicitudes.

Tipo de Prueba: Pruebas de Carga o Load Testing

La evaluación efectuada correspondió a una Prueba de Carga, cuyo objetivo fue medir el comportamiento del sistema bajo condiciones de carga normal y de demanda máxima esperada. El diseño de la prueba se definió a partir de los siguientes parámetros técnicos:

- Usuarios Concurrentes: Se simuló la interacción simultánea de 200 usuarios virtuales con los endpoints de la API.
- Modelo de Incremento de Carga: Se aplicó un aumento progresivo del número de usuarios durante el primer minuto con el fin de observar la capacidad de adaptación de los recursos del entorno de ejecución local, seguido de un periodo de carga constante de tres minutos para evaluar la estabilidad del sistema.
- Transacciones Evaluadas: La prueba se enfocó en el endpoint de registro de ingresos *POST /api/ingresos*, debido a que representa una de las operaciones con mayor consumo de recursos, al involucrar validaciones de datos, inserciones en múltiples tablas y la actualización de la bitácora del sistema.

Resultados de la prueba de rendimiento.

Una vez obtenido el resultado de la prueba, anexo 8 Resultado de Load Testing, se procedió con el análisis respectivo con el fin de evaluar el comportamiento de Sistema bajo condiciones de alta concurrencia.

Los indicadores registraron evidencia de un desempeño estable y altamente eficiente.

Resultados de Validación y Tasa de Éxito

Los resultados alcanzados evidencian una ejecución completamente satisfactoria de las pruebas realizadas. El indicador *checks_failed* registró un porcentaje de 0,00 %, lo que demuestra que las 143.178 validaciones establecidas fueron cumplidas de manera correcta. Estas validaciones contemplaron la comprobación del código de respuesta HTTP 200, la correcta persistencia de los registros de ingresos en la base de datos, así como el cumplimiento de un tiempo de respuesta menor a 800 ms por operación.

Asimismo, el indicador *http_req_failed* presentó un valor de 0,00 %, lo que indica que las 47.726 solicitudes HTTP enviadas al sistema fueron atendidas exitosamente, sin presentarse errores de conexión, interrupciones del servicio ni fallos a nivel de la API. Este comportamiento confirma la solidez y estabilidad del servidor Node.js durante la ejecución sostenida de procesos críticos.

Conclusiones

La ejecución del proyecto permitió alcanzar de forma efectiva los objetivos técnicos definidos en la fase inicial, a través del diseño e implementación de una arquitectura de software modular y desacoplada, apoyada en tecnologías actuales como Node.js para la capa de servidor y React para la capa de presentación. Esta decisión arquitectónica posibilitó una diferenciación clara entre los componentes de negocio y la interfaz de usuario, lo que favorece la mantenibilidad del sistema, su escalabilidad y su capacidad de adaptación ante futuras ampliaciones, sin afectar su estabilidad operativa.

De igual manera, la automatización de los procesos financieros generó un impacto positivo en la gestión interna de la agencia de viajes. La sustitución de procedimientos manuales basados en hojas de cálculo por un sistema digital permitió optimizar el control de ingresos y egresos. Funcionalidades como la localización eficiente de clientes y el llenado automático de información contribuyeron a disminuir los tiempos de registro de transacciones y a reducir la incidencia de errores en la asignación de valores financieros, incluyendo tarifas, impuestos y comisiones.

En relación con la protección y consistencia de la información, el sistema integra mecanismos de autenticación y control de acceso mediante JSON Web Tokens (JWT), así como el uso de transacciones ACID en la base de datos MySQL, garantizando la confidencialidad, integridad y confiabilidad de los datos financieros. Adicionalmente, la incorporación de un módulo de auditoría permite conservar un historial inalterable de las operaciones realizadas, impidiendo la eliminación definitiva de registros y asegurando la trazabilidad de la información conforme a principios de control interno.

Por último, las pruebas de rendimiento ejecutadas mediante la herramienta k6 demostraron un comportamiento estable del sistema ante escenarios de alta concurrencia. Los resultados obtenidos, con tiempos de respuesta inferiores a 800 milisegundos bajo una simulación de 200 usuarios simultáneos, confirman la solidez y capacidad de escalamiento de la aplicación, validando su implementación en el entorno operativo real de la agencia Sarvaltours, incluso en temporadas de alta demanda turística.

Recomendaciones

Como línea de trabajo futura, se propone la integración del sistema con los servicios web del Servicio de Rentas Internas (SRI), con el propósito de automatizar la generación y autorización de comprobantes electrónicos en el momento del registro de los ingresos. Esta integración permitiría optimizar el cumplimiento de las disposiciones tributarias vigentes y disminuir la carga operativa asociada a los procesos de facturación manual.

Asimismo, se recomienda el desarrollo de un módulo de nómina que amplíe las funcionalidades actuales relacionadas con egresos y comisiones. La implementación de este componente facilitaría la automatización del pago de salarios, beneficios legales y obligaciones laborales, consolidando la gestión del recurso humano dentro de una única plataforma y aumentando la eficiencia administrativa.

Desde una perspectiva de infraestructura, resulta conveniente evaluar la migración de la base de datos hacia servicios de nube gestionados, tales como AWS RDS o Google Cloud SQL. Esta transición permitiría contar con respaldos automáticos, mayor disponibilidad del servicio y tolerancia a fallos, fortaleciendo la continuidad operativa y la resiliencia del sistema.

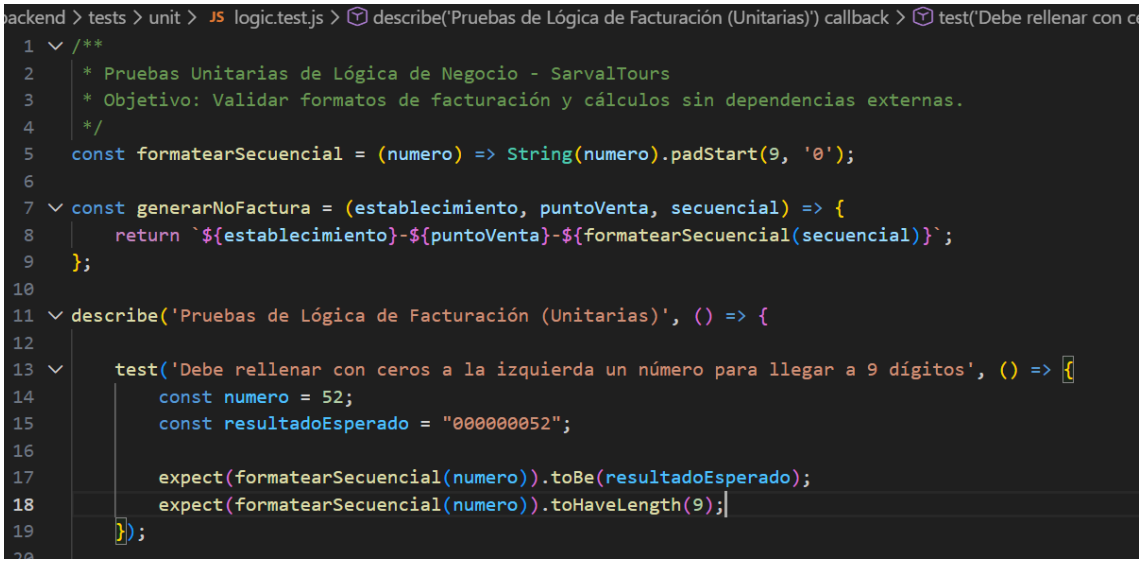
De igual forma, se considera indispensable establecer un plan de capacitación continúa dirigido al personal operativo y contable de la agencia. Dicho plan debe enfocarse en el uso correcto del sistema y en la adecuada clasificación de los egresos, con el objetivo de preservar la exactitud y confiabilidad de los reportes financieros generados por el dashboard.

Finalmente, para garantizar la calidad del software en futuras actualizaciones, se aconseja la adopción de prácticas de Integración Continua y Despliegue Continuo (CI/CD) mediante el uso de herramientas como GitHub Actions o GitLab CI/CD. La aplicación de estas prácticas permitirá automatizar pruebas, reducir errores en producción y asegurar una evolución controlada del sistema.

Referencias bibliográficas

1. **Aurum, A. & Wohlin, C.** (2023). *Engineering and Managing Software Requirements*. Springer.
2. **Pressman, R. S., & Maxim, B. A.** (2021). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.
3. **Sharma, A., & Singh, P.** (2021). Requirements Engineering: Practices and Challenges. *Journal of Software Engineering and Applications*, 14(1), 1-15.
4. **Sommerville, I.** (2019). *Software Engineering* (10th ed.). Pearson Education.
5. **Jevic, K.** (2023). Enterprise Architecture for Digital Transformation: A Comprehensive Guide. *Journal of Business Strategy and Digital Transformation*, 1(2), 50-65.
6. **Altexsoft Editorial Team.** (2023). *What Are Non-Functional Requirements? Types, Examples, and Best Practices*.
7. **Bellavista, D.** (2024). *Artículo sobre Requisitos No Funcionales y Calidad de Servicio.*
8. **Schwaber, K., & Sutherland, J.** (2020). *La Guía de Scrum: Las Reglas del Juego*. Scrum.org. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf

Anexos



```
backend > tests > unit > JS logic.test.js > describe('Pruebas de Lógica de Facturación (Unitarias)') callback > test('Debe rellenar con c
/**
 * Pruebas Unitarias de Lógica de Negocio - SarvalTours
 * Objetivo: Validar formatos de facturación y cálculos sin dependencias externas.
 */
const formatearSecuencial = (numero) => String(numero).padStart(9, '0');

const generarNoFactura = (establecimiento, puntoVenta, secuencial) => {
    return `${establecimiento}-${puntoVenta}-${formatearSecuencial(secuencial)}`;
};

describe('Pruebas de Lógica de Facturación (Unitarias)', () => {

    test('Debe rellenar con ceros a la izquierda un número para llegar a 9 dígitos', () => {
        const numero = 52;
        const resultadoEsperado = "000000052";

        expect(formatearSecuencial(numero)).toBe(resultadoEsperado);
        expect(formatearSecuencial(numero)).toHaveLength(9);
    });
```

anexo 1 logic.test.js

```
    test('Debe generar el formato completo de factura 001-001-XXXXXXXXX', () => {
        const est = "001";
        const pto = "005";
        const sec = 125;
        const esperado = "001-005-000000125";

        const resultado = generarNoFactura(est, pto, sec);
        expect(resultado).toBe(esperado);
    });
```

anexo 2logic.test.js 2

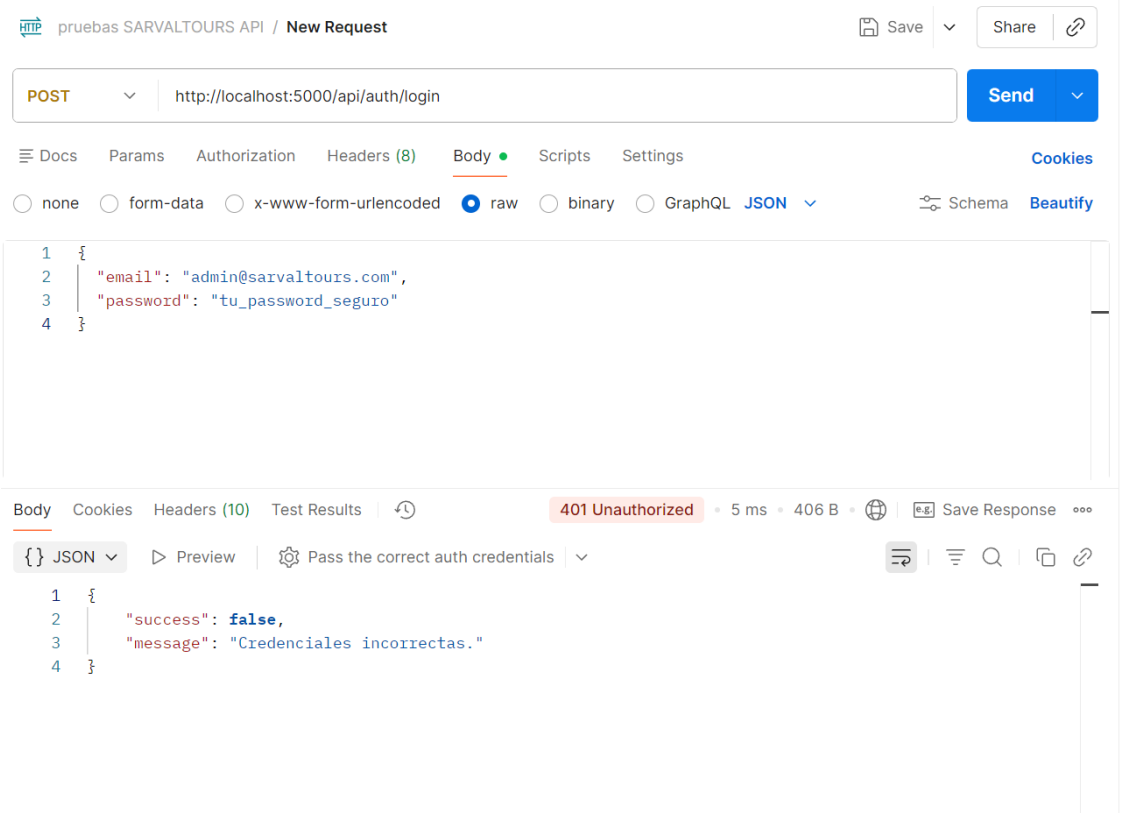
```
    test('Debe separar correctamente las partes de un ingreso existente', () => {
        const facturaExistente = "001-002-000000010";
        const partes = facturaExistente.split('-');
        expect(partes).toHaveLength(3);
        expect(partes[0]).toBe("001");
        expect(partes[2]).toBe("000000010");
        // Simulación de incremento para el siguiente ingreso
        const proximo = parseInt(partes[2]) + 1;
        expect(proximo).toBe(11);
    });
});
```

anexo 3logic.text.js 3

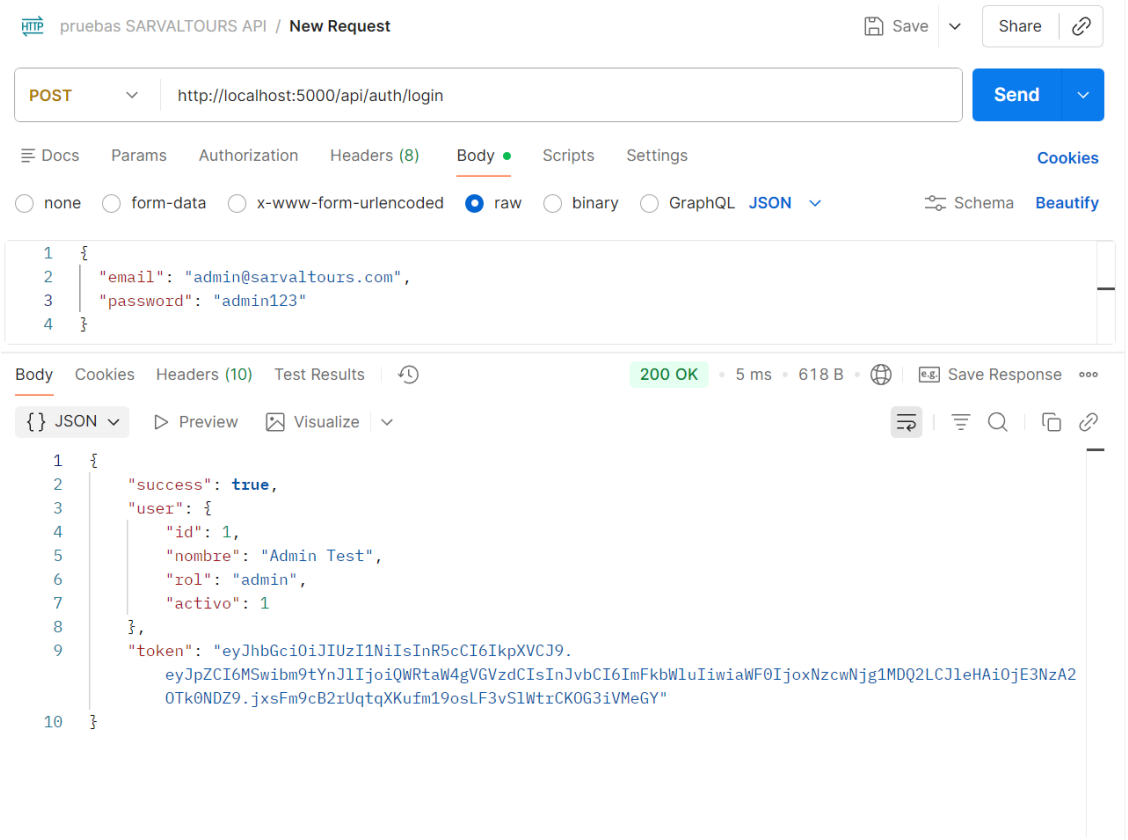
```
describe('Pruebas de Lógica de Fechas', () => {
    test('Debe extraer solo la fecha (YYYY-MM-DD) de un objeto ISO String', () => {
        const fechaISO = "2026-02-09T22:30:00.000Z";
        const soloFecha = fechaISO.split('T')[0];

        expect(soloFecha).toBe("2026-02-09");
        expect(soloFecha).toMatch(/^\d{4}-\d{2}-\d{2}$/);
    });
});
```

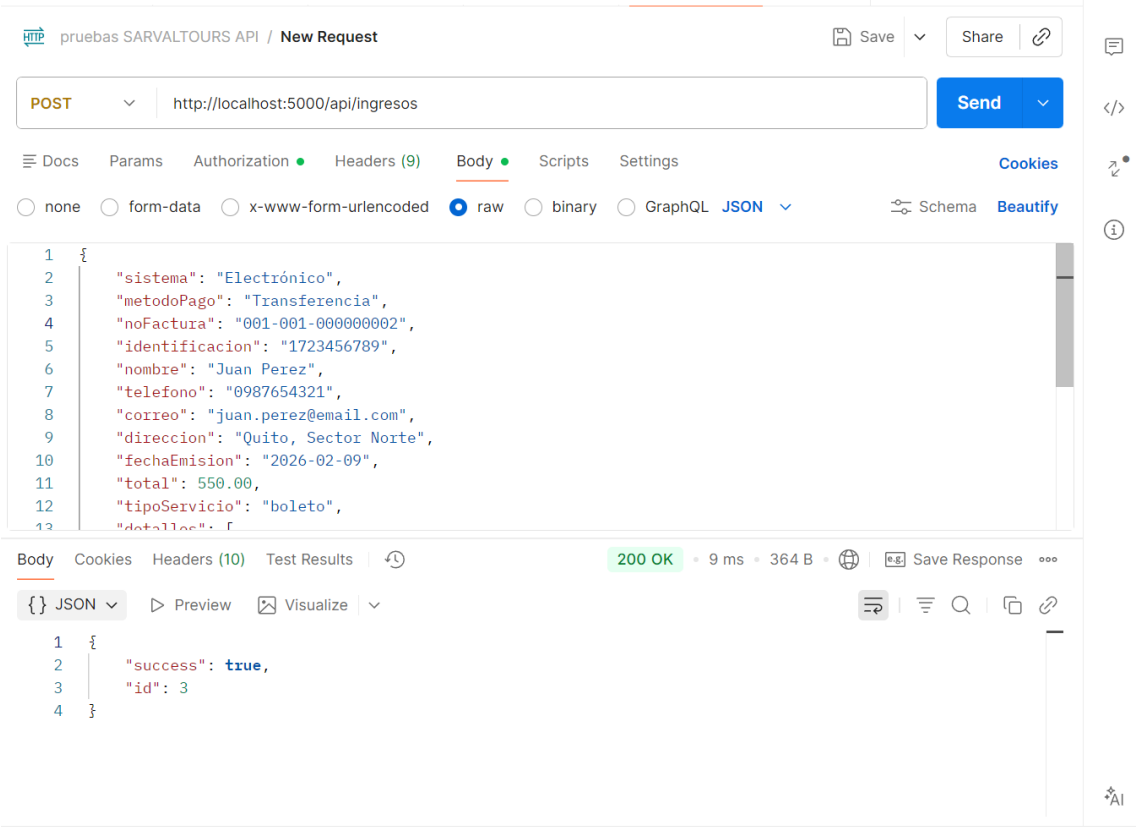
anexo 4logic.test.js 4



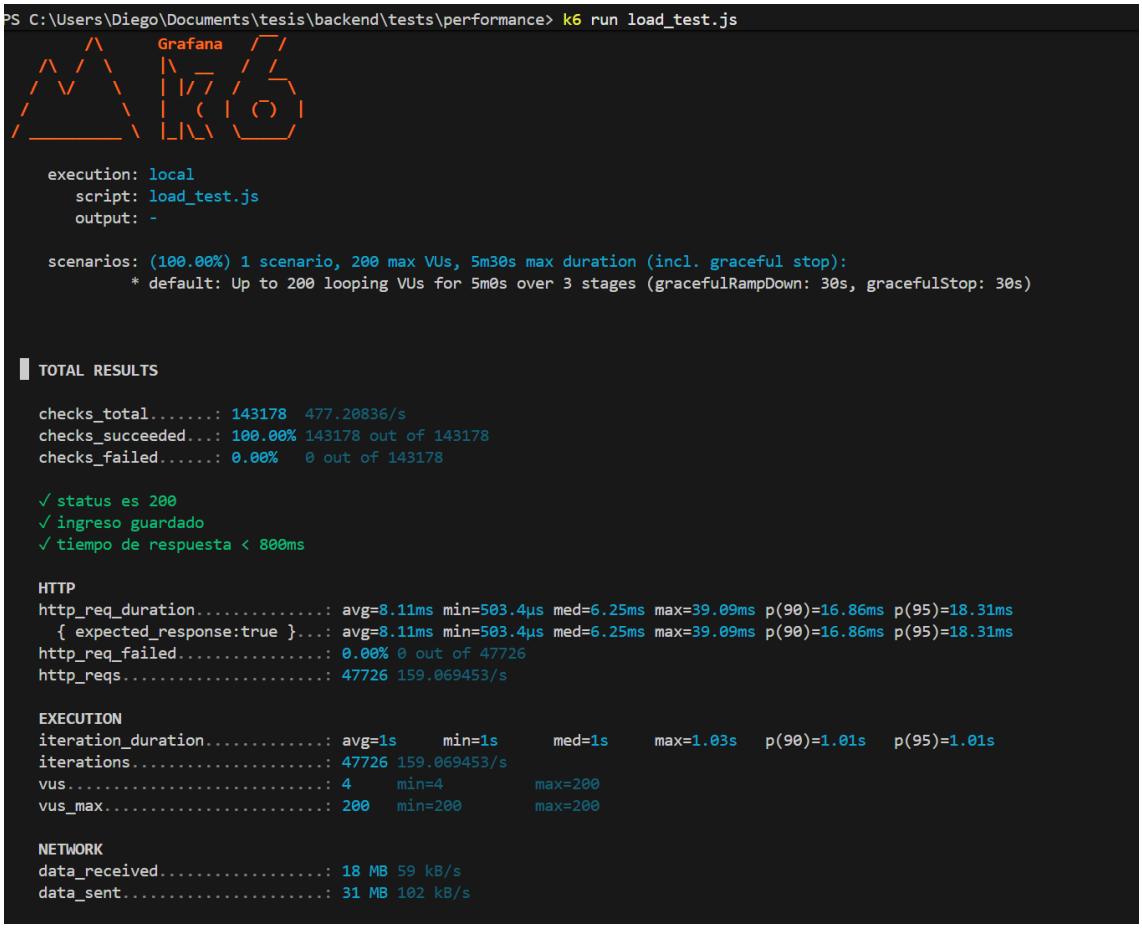
anexo 5Pruebas de integración con POSTMAN



anexo 6Prueba de integración Postman 2



anexo 7Prueba POST en ingresos usando JWT



anexo 8 Resultado load testing