UNIVERSIDAD INTERNACIONAL DE LAS AMÉRICAS

FACULTAD DE INGENIERÍA Y ARQUITECTURA

TRABAJO FINAL DE GRADUACIÓN PARA OPTAR POR EL GRADO DE BACHILLER EN INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Sistema web para la gestión financiero-contable para empresa TecniTech Solutions CR, ubicada en San José, Costa Rica

Estudiante

Adrián José Fajardo Pérez

Tutor

Fernando Ríos Vargas

Sede Central

Agosto, 2026

DEDICATORIA

Dedico este trabajo, en primer lugar, a Dios, por brindarme la fortaleza, la sabiduría y la perseverancia necesarias para completar esta etapa tan importante de mi formación profesional.

A mis padres, Verny y Magaly, por su apoyo incondicional, sus valores y el esfuerzo constante que han realizado a lo largo de mi vida para darme las herramientas necesarias para salir adelante. Su guía ha sido fundamental en cada logro alcanzado.

A mi esposa, Keylin, por su comprensión, paciencia y apoyo constante durante todo este proceso. Su acompañamiento ha sido clave para mantener el enfoque y la motivación en los momentos más exigentes.

Este logro también es reflejo del respaldo y la confianza que cada uno de ellos ha depositado en mí.

AGRADECIMIENTOS

Agradezco, en primer lugar, a Dios, por darme la oportunidad, la fortaleza y la disciplina necesarias para completar esta etapa académica.

A mis padres, Verny y Magaly, y a mi esposa, Keylin, por su apoyo constante, comprensión y motivación a lo largo de todo este proceso. Su respaldo ha sido fundamental para alcanzar este logro.

A la Universidad Internacional de las Américas, por brindarme la formación académica y las herramientas necesarias para desarrollarme profesionalmente en el área.

A los docentes que formaron parte de este proceso, por su guía, conocimiento y acompañamiento, los cuales fueron clave para el desarrollo de este trabajo.

De manera especial, agradezco a la directora de carrera, Olda Bustillos, cuya exigencia y altos estándares de calidad han contribuido significativamente a fortalecer mi criterio, tanto a nivel profesional como personal.

Finalmente, agradezco a todas las personas que, de una u otra forma, contribuyeron al cumplimiento de esta meta profesional.

CONTENIDO

FIGURAS

TABLAS

APÉNDICES

RESUMEN EJECUTIVO

El presente proyecto de graduación consiste en el desarrollo de un sistema web, denominado OneCore, para la gestión financiero-contable de TecniTech Solutions CR, una pyme costarricense dedicada al desarrollo de *software* que actualmente opera sin un sistema automatizado e integrado para el control de sus finanzas. La ausencia de esta herramienta ha generado registros manuales dispersos, procesos de facturación desconectados del control de cobros, falta de validaciones automáticas y una capacidad limitada para generar información financiera confiable y oportuna que soporte la toma de decisiones.

La solución propuesta es un sistema web construido bajo una arquitectura API + frontend desacoplado, utilizando .NET Core en el backend, React en el frontend y Azure SQL Database como gestor de base de datos, desplegado sobre infraestructura en la nube mediante Azure App Service y Azure Static Web Apps. El sistema contempla módulos integrados de facturación, cuentas por cobrar, cuentas por pagar, registro de ingresos y gastos, presupuesto, cuentas contables, reportes y control de acceso por roles, asegurando trazabilidad y consistencia en cada transacción registrada.

El desarrollo se ejecuta bajo un modelo incremental complementado con el marco de gestión Scrum, lo que permite construir y validar el sistema por módulos priorizados de forma progresiva. La investigación adopta un enfoque mixto, puesto que combina técnicas cualitativas para el levantamiento de requerimientos con métricas cuantitativas para la validación del prototipo. Dado que el sistema se desarrolla como trabajo final de graduación, no representa una inversión directa para la empresa, lo que refuerza su viabilidad técnica, operativa, económica y legal dentro del contexto normativo costarricense vigente.

CAPÍTULO I: INTRODUCCIÓN

Planteamiento del Problema

TecniTech Solutions CR es una pequeña y mediana empresa (pyme) ubicada en Costa Rica, dedicada al desarrollo de *software* y a la prestación de soluciones informáticas en distintas áreas tecnológicas. La empresa se encuentra en una fase inicial de operación y actualmente carece de un sistema financiero-contable automatizado, por lo que la gestión de sus ingresos, gastos y obligaciones se realiza de forma manual, lo que limita el control y la disponibilidad de información financiera para la toma de decisiones. Al respecto, se detalla una serie de problemáticas detectadas en esta compañía:

- Uso de registros manuales y fuentes de datos no integradas.
- Procesos aislados entre la facturación y el registro de cobros.
- Registro manual y sin validación de facturas de proveedores.
- Ingreso de datos financieros sin validación ni conciliación automática.
- Ausencia de clasificación estandarizada en el registro de gastos.
- Emisión de facturas sin automatización ni control de validaciones.
- Procesos desconectados entre la planificación y la ejecución financiera.

Objetivos

Objetivo General

Desarrollar un sistema web para la gestión financiero contable de la empresa TecniTech Solutions CR, utilizando las herramientas Visual Studio Code y SQL Server.

Objetivos Específicos

1. Analizar los requerimientos funcionales y no funcionales del sistema web.
2. Construir el diseño de la base de datos y del sistema web mediante la herramienta Workbench.

3. Programar los diferentes módulos del sistema web mediante la herramienta Visual Studio Code.
4. Probar los diferentes módulos de forma independiente e integral, garantizando los resultados correctos del sistema.

Justificación

La presente investigación se lleva a cabo con el propósito de diseñar e implementar un sistema web para la gestión financiero-contable de la empresa TecniTech Solutions CR, ante la ausencia de una herramienta automatizada que les permita controlar de forma integrada los ingresos, gastos, cuentas por cobrar, cuentas por pagar y presupuestos.

Actualmente, la gestión financiera de la empresa se realiza mediante registros manuales y herramientas no integradas, lo que incrementa el riesgo de errores, inconsistencias en la información y retrasos en la elaboración de reportes financieros. Esta situación limita la capacidad de la empresa para tomar decisiones oportunas y fundamentadas, especialmente en una etapa inicial de crecimiento donde el control financiero es crítico.

Viabilidad Técnica

El proyecto es técnicamente viable debido a que se basa en el desarrollo de un sistema web con arquitectura API + frontend, utilizando tecnologías modernas, estables y ampliamente adoptadas en entornos empresariales.

El backend del sistema se desarrolla usando .NET Core, el cual construye una API robusta, segura y escalable para la gestión de la lógica de negocio financiero-contable. El frontend se implementa con React, facilitando la construcción de interfaces dinámicas e intuitivas para los usuarios administrativos del sistema. La persistencia de la información se realiza mediante Azure SQL Database, un gestor de base de datos relacional que garantiza integridad, consistencia y confiabilidad en el manejo de datos financieros.

En cuanto a la infraestructura, el sistema se despliega por medio de Azure App Service para el backend y Azure Static Web Apps para el frontend, ambos en planes gratuitos, lo que posibilita la ejecución y validación del prototipo sin incurrir en costos de licenciamiento o infraestructura.

Esta configuración es adecuada para el alcance académico del proyecto y para el contexto operativo de una pyme en etapa inicial.

Desde el punto de vista de *hardware* y *software*, el desarrollo puede efectuarse en equipos de cómputo estándar, empleando herramientas compatibles con las tecnologías seleccionadas. No se requieren equipos especializados ni espacio físico adicional, lo que confirma que el proyecto es ejecutable con los recursos técnicos disponibles y sin limitaciones tecnológicas significativas.

Viabilidad Operativa

El proyecto es operacionalmente factible, ya que el sistema propuesto consiste en una aplicación web, accesible mediante un navegador, lo que facilita su adopción sin requerir instalaciones locales ni configuraciones técnicas complejas por parte de los usuarios.

Los usuarios del sistema son principalmente personal administrativo y responsables de la gestión financiera de la empresa TecniTech Solutions CR, quienes realizan tareas como el registro de ingresos, gastos, facturación, cuentas por cobrar, cuentas por pagar y consulta de reportes. Estas actividades corresponden a procesos que actualmente se ejecutan de forma manual, por lo que el sistema no introduce nuevas funciones, sino que automatiza y ordena las existentes.

El uso del sistema no provoca reducción de personal ni cambios drásticos en la estructura organizacional, sino una mejora en la eficiencia operativa y en la calidad de la información. Si bien es cierto que se requiere una capacitación básica para el uso del sistema, esta es mínima debido a que la interfaz está diseñada de forma intuitiva y alineada con los procesos administrativos comunes. En consecuencia, se espera que el sistema diseñado se utilice de manera efectiva una vez implementado, de modo que alcance los beneficios esperados del proyecto.

Viabilidad Económica

Desde el punto de vista económico, el proyecto es viable, puesto que los costos asociados al desarrollo e implementación del sistema son identificables, controlables y no representan una erogación directa para la empresa, debido a que el sistema es parte de un proyecto académico de graduación. Para efectos de análisis, se detallan los costos estimados de *software*, *hardware* y etapas de desarrollo (incluidos en la Tabla 1); sin embargo, estos no son asumidos por la empresa

TecniTech Solutions CR, dado que el desarrollo es realizado por el estudiante como requisito académico.

Costos de *software*. Para el desarrollo del sistema se utilizan herramientas de programación, *frameworks* y servicios en la nube que cuentan con licencias gratuitas o de uso libre, por lo que no generan costos de adquisición para la empresa. Estos recursos permiten crear, probar y desplegar el sistema dentro del alcance académico del proyecto sin requerir licencias comerciales.

Las tecnologías seleccionadas corresponden a herramientas ampliamente usadas en el desarrollo de aplicaciones web y cuentan con versiones gratuitas orientadas a proyectos de desarrollo y aprendizaje.

Tabla 1

Costos de software.

***Software* / Herramienta**	**Uso dentro del proyecto**	**Costo estimado**
Visual Studio Code	Desarrollo backend y frontend	₡0
.NET Core SDK	Desarrollo del backend	₡0
React	Desarrollo del frontend	₡0
Azure SQL Database	Base de datos	₡0
Azure App Service	Hosting backend	₡0
Azure Static Web Apps	Hosting frontend	₡0

Fuente: Elaboración propia, 2026.

Costos de *hardware*. El proyecto no requiere la adquisición de *hardware* adicional, ya que el desarrollo del sistema se realiza con los equipos de cómputo existentes. Asimismo, la infraestructura necesaria para ejecutar el sistema es proporcionada mediante servicios en la nube de la plataforma Microsoft Azure.

En particular, el backend del sistema es desplegado utilizando Azure App Service en su plan gratuito, mientras que el frontend es publicado en Azure Static Web Apps, también, en su versión gratuita. La base de datos es implementada en Azure SQL Database en un entorno de desarrollo, el cual crea bases de datos con recursos limitados para fines académicos y de prueba.

Estos servicios ofrecen planes gratuitos diseñados para el desarrollo de aplicaciones web, los cuales proporcionan recursos de infraestructura sin necesidad de adquirir servidores físicos.

Debido a que el sistema creado corresponde a un prototipo académico y no a un sistema en producción, los recursos gratis disponibles son suficientes para realizar las pruebas y validaciones del proyecto. Por esta razón, no se incurre en costos de *hardware* asociados al desarrollo del prototipo, como se evidencia, a continuación, en la Tabla 2.

Tabla 2

Costos de hardware.

Hardware	**Uso dentro del proyecto**	**Costo estimado**
Equipo laptop personal	Desarrollo y pruebas del sistema	₡0
Equipo desktop personal	Simulación de base de datos en la nube	₡0
Infraestructura en la nube	Ejecución del prototipo	₡0

Fuente: Elaboración propia, 2026.

Costos por etapas de desarrollo. Las distintas etapas del desarrollo del sistema incluyen actividades de análisis de requerimientos, diseño, programación, pruebas e implementación del prototipo. Estas actividades corresponden al trabajo que normalmente llevaría a cabo un profesional en desarrollo de *software*.

Para estimar el costo teórico del desarrollo del sistema, se tomó como referencia el salario promedio de un desarrollador de *software* según las tablas salariales del Ministerio de Trabajo y Seguridad Social de Costa Rica. A partir de este valor, de acuerdo con la Tabla 3, se realizó una estimación del costo asociado a cada etapa del desarrollo, considerando el tiempo aproximado requerido para completar cada fase del proyecto.

Tabla 3

Costos por etapas de desarrollo.

Etapa	**Uso dentro del proyecto**	**Costo estimado**
Análisis de requerimientos	Levantamiento y documentación de requerimientos	₡225,000
Diseño del sistema	Diseño de base de datos y arquitectura	₡168,750
Desarrollo	Programación de módulos del sistema	₡450,000
Pruebas	Pruebas unitarias e integrales	₡168,750

Implementación del prototipo	Despliegue en entorno de prueba	₡112,500

Fuente: Elaboración propia, 2026.

A pesar de que el proyecto tiene un costo teórico asociado al tiempo y esfuerzo de desarrollo, estimado según el salario promedio de un desarrollador de *software*, la empresa TecniTech Solutions CR no incurre en estos gastos. Esto se debe a que el sistema es ejecutado como parte del trabajo final de graduación del estudiante, por lo que los precios antes desglosados forman parte del proceso académico y no representan una inversión económica para la organización. En consecuencia, el sistema representa una oportunidad de mejora operativa para la empresa sin generar costos directos de desarrollo, lo que refuerza la viabilidad económica del proyecto.

Viabilidad Legal

El proyecto es legalmente viable, ya que su desarrollo y ejecución se ajustan a la normativa vigente en Costa Rica relacionada con el uso de tecnologías de información, la protección de datos, los derechos de autor y la prevención de delitos informáticos. El sistema propuesto es de uso interno para la empresa TecniTech Solutions CR y no contempla actividades que contravengan las disposiciones legales aplicables. Por lo anterior, seguidamente, se describen las principales leyes aplicables al proyecto y la forma en que este cumple con cada una de ellas.

Cabe aclarar que, tratándose de normativa legal vigente, las referencias a leyes y reglamentos citadas en este documento no se sujetan al criterio de vigencia editorial de cinco años aplicado a la literatura académica y técnica. Su fecha de referencia corresponde a la de su promulgación oficial, y su validez se mantiene mientras no sean derogadas o reformadas por la Asamblea Legislativa o el Poder Ejecutivo, según corresponda.

Ley N.ª 8148 – Adición de los artículos 196 BIS, 217 BIS y 229 BIS al Código Penal. Esta ley incorpora al Código Penal costarricense disposiciones sobre delitos informáticos, tales como el acceso indebido a sistemas informáticos, la alteración de información y el uso no autorizado de datos. Su objetivo es sancionar conductas que atenten contra la confidencialidad, integridad y disponibilidad de la información almacenada en sistemas computacionales.

El sistema web propuesto no incumple esta ley, ya que se diseña con mecanismos básicos de autenticación y control de acceso, permitiendo únicamente el ingreso de usuarios autorizados. Además, el sistema no busca acceder, modificar ni interceptar información de terceros, sino gestionar datos financieros propios de la empresa dentro de un entorno controlado.

Ley N.ª 4573 – Ley para reprimir y sancionar los delitos informáticos (2001). Esta ley establece sanciones para actos ilícitos vinculados con el uso indebido de sistemas informáticos, tales como la manipulación de datos, el fraude electrónico y el daño a sistemas de información. Esta normativa busca proteger la seguridad y el uso legítimo de las tecnologías de información.

El proyecto cumple con esta ley, en vista de que el sistema se utiliza exclusivamente para fines administrativos y contables legítimos, sin realizar actividades fraudulentas ni afectar sistemas ajenos. Asimismo, el desarrollo de este proyecto se orienta a fortalecer el control de la información y no a vulnerar la seguridad informática de la empresa TecniTech Solutions CR.

Ley N.ª 6683 – Ley de Derechos de Autor y Derechos Conexos. La Ley de Derechos de Autor protege las obras intelectuales y regula el uso legal de *software*, código fuente y material digital. Esta normativa prohíbe la reproducción, modificación o distribución no autorizada de obras protegidas por derechos de autor.

El proyecto no infringe esta ley, puesto que en el desarrollo del sistema se emplean tecnologías, frameworks y herramientas de *software* libre o con licencias gratuitas, respetando las condiciones de uso establecidas por cada una. De igual modo, el código fuente del sistema es de elaboración propia y se entrega a la empresa TecniTech Solutions CR, quien funge como propietaria del producto.

Ley N.ª 8968 – Ley de Protección de la Persona frente al Tratamiento de sus Datos Personales. En Costa Rica, la Ley N.º 8968 regula el tratamiento de datos personales con el fin de garantizar la privacidad y el derecho a la autodeterminación informativa de las personas. Establece principios como el consentimiento, la confidencialidad y el uso adecuado de la información personal.

El sistema propuesto acata esta ley porque los datos que se gestionan corresponden a información financiera y administrativa de la empresa y sus clientes, utilizados únicamente para

fines internos y legítimos. El acceso a la información está restringido a usuarios autorizados y no se contempla la divulgación ni comercialización de datos personales; por consiguiente, garantiza la confidencialidad y el uso responsable de la información.

Con base en el análisis realizado, se concluye que el proyecto no presenta impedimentos legales para su desarrollo, pues cumple con las disposiciones establecidas en la legislación costarricense vigente. El sistema se desarrolla respetando los principios de legalidad, confidencialidad y uso adecuado de la información, lo que confirma la viabilidad legal de la investigación.

Proyecciones

Este proyecto de investigación tiene como propósito dotar a la empresa TecniTech Solutions CR de un sistema web para la gestión financiero-contable, llamado OneCore, que permita automatizar y centralizar los principales procesos económicos de la organización. Con el producto final, se espera contar con una herramienta funcional que facilite el registro, control y consulta de la información financiera de forma ordenada, confiable y oportuna.

A nivel organizacional, el sistema mejora el control interno, reduce errores asociados a los registros manuales y dispondrá de información financiera estructurada para apoyar la toma de decisiones. Asimismo, el prototipo sirve como base tecnológica para futuras mejoras o ampliaciones del sistema, conforme la empresa incremente su volumen de operaciones y complejidad administrativa.

Alcance Funcional

El sistema web cuenta con capacidades orientadas a la gestión integral de la información financiero-contable de la empresa. Entre sus principales funcionalidades se incluyen las siguientes:

Cuentas Contables. Permite registrar, modificar y controlar el plan de cuentas de la empresa. Ejecuta procesos automáticos de contabilización y conciliación, generando asientos contables al momento de registrar ingresos o egresos. Valida la correspondencia entre cuentas activas, pasivas y de resultado, garantizando la integridad de los estados financieros.

Cuentas por Cobrar. Procesa las facturas emitidas y los pagos realizados por los clientes, actualizando automáticamente los saldos pendientes y calculando intereses por morosidad. Aplica abonos parciales o totales, genera alertas de vencimiento y realiza conciliaciones automáticas entre facturas, pagos y movimientos bancarios.

Cuentas por Pagar. Registra y controla las facturas de proveedores, órdenes de compra y compromisos financieros. Calcula automáticamente los montos pendientes por pagar, aplica retenciones e impuestos y actualiza el estado de cuenta del proveedor. Ejecuta validaciones para evitar pagos duplicados y genera reportes de vencimientos próximos.

Ingresos. Registra los ingresos provenientes de los distintos servicios de la empresa, aplicando validaciones con las facturas correspondientes. Ejecuta cálculos automáticos de impuestos y asientos contables, generando movimientos directos en el módulo de cuentas contables. Identifica las fuentes de ingreso y analiza su comportamiento por periodo.

Gastos. Contabiliza el registro y control de los gastos operativos de la empresa, clasificándolos por tipo y centro de costo. Ejecuta validaciones de presupuesto disponible, calcula el impacto de cada gasto en el flujo financiero y actualiza los registros contables asociados. Incluye alertas cuando se superan los límites definidos para cada categoría.

Facturación. Automatiza la emisión de facturas electrónicas, el cálculo de impuestos y la validación de datos del cliente. Genera consecutivos automáticos, aplica descuentos y registra las transacciones en los módulos de cuentas por cobrar e ingresos. Facilita la trazabilidad completa de la factura desde su emisión hasta su cobro.

Presupuesto. Define, controla y compara los presupuestos anuales o por proyecto. Realiza cálculos automáticos de ejecución y genera desviaciones porcentuales entre lo planificado y lo ejecutado. Se integra con los módulos de gastos e ingresos para actualizar en tiempo real el estado financiero de la empresa.

Mantenimientos. Realiza operaciones de creación, modificación, actualización y eliminación de los datos maestros del sistema, tales como clientes, proveedores, cuentas contables y parámetros generales.

Consultas. Visualiza la información almacenada en el sistema mediante consultas estructuradas, permitiendo a los usuarios acceder a datos financieros de forma rápida y ordenada.

Reportes. Genera reportes financieros básicos a partir de la información registrada en el sistema, los cuales se visualizan en pantalla o se exportan según los requerimientos del usuario.

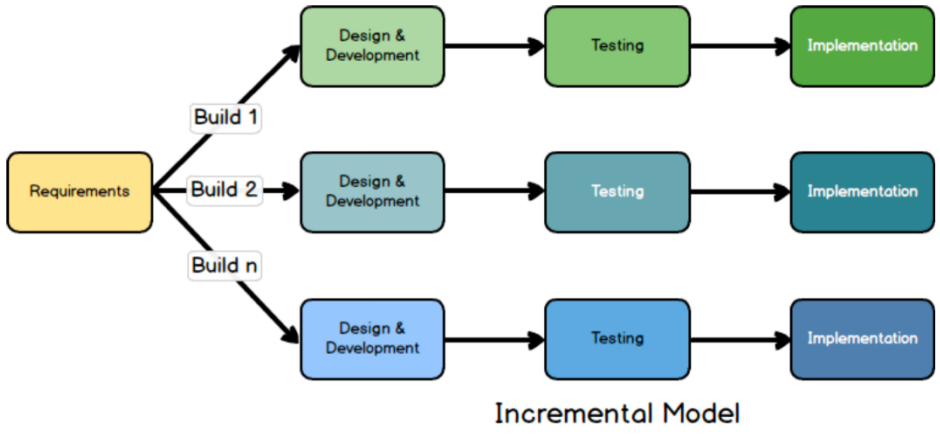
Seguridad. Gestiona la autenticación de usuarios y la definición de perfiles de acceso, garantizando que cada usuario solo pueda acceder a las funcionalidades autorizadas de acuerdo con su rol dentro de la organización.

El sistema no contempla integraciones con *software* externos ni funcionalidades avanzadas fuera del alcance académico del proyecto. El desarrollo se enfoca en asegurar que los módulos definidos funcionen de manera integrada y consistente, cumpliendo con los objetivos planteados para el prototipo.

Alcance Metodológico

Para la ejecución del sistema se emplea un método de desarrollo incremental, el cual construye el *software* de forma progresiva mediante la incorporación gradual de funcionalidades. Este enfoque resulta adecuado para el proyecto, ya que el sistema puede dividirse en módulos independientes que se desarrollan y validan por etapas.

Figura 1
Método de desarrollo incremental.



Fuente: Elaboración propia, 2026.

La Figura 1 muestra las fases de análisis de requerimientos, diseño, implementación, pruebas, integración, despliegue y mantenimiento, las cuales se aplican de manera repetitiva a cada incremento del sistema. En lugar de ejecutar estas fases una sola vez para todo el proyecto, los distintos módulos del sistema financiero-contable recorren este conjunto de etapas de modo independiente, generando incrementos funcionales que pueden ser evaluados antes de continuar con el siguiente.

En el contexto del proyecto, cada incremento corresponde al desarrollo de uno o más módulos del sistema, permitiendo validar su funcionamiento, corregir errores y asegurar que cumpla los requerimientos definidos. Este procedimiento facilita el control del avance, reduce riesgos y posibilita la obtención de un prototipo funcional de forma ordenada y progresiva.

El uso del método incremental garantiza que el sistema sea desarrollado de manera estructurada y controlada, asegurando la calidad de cada módulo antes de su integración final. Asimismo, este enfoque es coherente con el alcance académico del proyecto y con la necesidad de contar con un prototipo probado y documentado al finalizar el proceso de investigación.

Alcance Tecnológico

El proyecto corresponde al desarrollo de un sistema web, implementado bajo una arquitectura API + frontend. Para ello, se utilizan las siguientes tecnologías:

- **Backend:** API en Microsoft .NET Core.
- **Frontend:** interfaz de usuario con React.
- **Base de datos:** Azure SQL Database para almacenar la información financiera.
- **Infraestructura:** Azure App Service para el backend y Azure Static Web Apps para el frontend.

El sistema no se ejecuta en servidores locales, sino que es desplegado en la nube, lo que favorece su acceso desde cualquier navegador web autorizado y facilita su mantenimiento y escalabilidad dentro del alcance del proyecto.

CAPÍTULO II: MARCO REFERENCIAL

El desarrollo de un sistema financiero-contable para una pequeña empresa no puede abordarse únicamente desde la perspectiva técnica del *software*. De hecho, requiere comprender la naturaleza de la información contable, su función dentro de la organización y el impacto que tiene su gestión en la toma de decisiones, el control interno y la sostenibilidad empresarial.

En el caso de TecniTech Solutions CR, la ausencia de un sistema integrado ha ocasionado procesos manuales, registros dispersos y falta de validaciones automáticas, lo que compromete la calidad y confiabilidad de la información financiera. Bajo ese contexto, el presente marco referencial tiene como propósito fundamentar teóricamente el proyecto desde tres dimensiones principales: contable, tecnológica y normativa.

Primero, se analiza el concepto de sistema de información contable y su relevancia en las pymes; segundo, se abordan aspectos estructurales relacionados con arquitectura de *software* y modelos de desarrollo; finalmente, se estudian los elementos de seguridad y protección de datos que deben considerarse en un sistema financiero web en Costa Rica.

Sistemas de Información Contable y Control Financiero

La contabilidad, en su concepción actual, es más que la simple acumulación de registros históricos. Constituye un sistema estructurado que organiza, clasifica y transforma datos económicos en información útil para la gestión empresarial. Este enfoque sistémico es especialmente relevante cuando se analiza la necesidad de automatización en organizaciones pequeñas que aún operan con registros manuales o herramientas no integradas.

Bolaños et al. (2021), en su propuesta de implementación de un sistema contable para una asociación costarricense, señalan lo siguiente:

> la contabilidad es un sistema de información integrado a la empresa, que permite identificar, clasificar, registrar, resumir, interpretar, analizar y evaluar, en términos monetarios, las operaciones y transacciones económicas de una empresa con miras a ofrecer información útil, veraz, oportuna y eficiente. (p. 65).

Esta definición es particularmente relevante porque introduce tres atributos que determinan la calidad del sistema: utilidad, veracidad y oportunidad. No basta con registrar datos; la información debe ser confiable y estar disponible en el momento en que se requiere para la toma de decisiones. Cuando una empresa opera mediante hojas de cálculo aisladas o registros manuales, estos atributos se ven comprometidos. La veracidad depende del criterio individual de quien registra la información, la oportunidad se limita a la capacidad de consolidar datos manualmente, mientras que la utilidad se reduce si no existen mecanismos de análisis automatizado.

En la misma investigación, Bolaños et al. (2021) describen cómo la ausencia de estructura operativa puede generar "inconsistencias en la estructura operativa, lo cual induce a problemas de distribución de excedentes, falta de transparencia y control de datos" (p. 15). Aunque su estudio se desarrolló en una asociación solidarista, el fenómeno es comparable al contexto de una pyme tecnológica que carece de integración contable formal.

Este punto conecta directamente con la realidad de TecniTech Solutions CR, visto que, al no contar con un sistema financiero integrado, surgen diversos inconvenientes:

- Los procesos de facturación no están conectados con el registro de cobros.
- No existe conciliación automática.
- No hay validaciones estructuradas en el ingreso de datos.
- No se garantiza trazabilidad histórica.

Desde una perspectiva técnica, un sistema de información contable debe cumplir al menos con las siguientes funciones estructurales:

1. Registro sistemático de transacciones.
2. Clasificación bajo un catálogo de cuentas estandarizado.
3. Generación automática de reportes financieros.
4. Validación de integridad en el ingreso de datos.
5. Mecanismos de control interno.

Chinchilla (2020), al analizar la implementación de un sistema contable en una empresa costarricense, destaca la importancia del análisis previo de requerimientos y la adecuación del sistema a la realidad organizacional. Asimismo, señala que la implementación de un sistema

contable no debe entenderse como la simple adquisición de *software*, sino como un proceso estructurado que incluye planificación, análisis de requerimientos, diseño, desarrollo, prueba, integración y operación (pp. 27–29). Con base en lo anterior, este enfoque es concordante con el proyecto actual, ya que no se trata únicamente de programar una aplicación web, sino de diseñar una solución alineada con los procesos reales de la empresa.

Otro elemento clave es la calidad de la información contable. La Ley N.º 8968, aunque está orientada a la protección de datos personales, introduce principios aplicables a cualquier sistema que gestione información. En su artículo 6, establece que los datos deben ser "actuales, veraces, exactos y adecuados al fin para el que fueron recolectados" (Asamblea Legislativa de la República de Costa Rica, 2011, art. 6). Ahora bien, a pesar de que el artículo se refiere a datos personales, el principio de calidad es igualmente aplicable a la información financiera: si los datos no son exactos ni adecuados, el sistema pierde su razón de ser.

En consecuencia, el desarrollo del sistema financiero para TecniTech Solutions CR debe garantizar los siguientes aspectos:

- Integridad referencial en base de datos.
- Validaciones en el backend.
- Control de acceso basado en roles.
- Registro de auditoría (log de cambios).
- Reportes consistentes y automatizados.

Desde el punto de vista estratégico, un sistema contable automatizado permite transformar la información en conocimiento. No solo muestra cuánto se facturó, sino qué clientes generan mayor rentabilidad, cuál es la estructura real de costos y cómo evolucionan los flujos de caja. En una pyme, donde el dueño cumple simultáneamente funciones administrativas, técnicas y contables, la automatización no representa un lujo tecnológico, sino una necesidad operativa que contribuye a reducir errores humanos, optimizar tiempos y mejorar la capacidad de análisis.

Por tanto, el sistema de información contable no debe entenderse como un módulo aislado, sino como el núcleo estructural del proyecto. A partir de él se articulan los demás componentes tecnológicos, metodológicos y normativos que se desarrollan en los siguientes apartados del marco referencial.

Contexto de las Pymes Costarricenses y Transformación Digital

El análisis del contexto nacional es indispensable para justificar la pertinencia del proyecto. La realidad operativa de una pyme costarricense no se desliga del entorno económico en el que se desarrolla. En Costa Rica, las micro, pequeñas y medianas empresas constituyen un componente estructural del tejido productivo, tanto en generación de empleo como en dinamización económica.

El Ministerio de Economía, Industria y Comercio (MEIC), a través de la Dirección General de Apoyo a la Pequeña y Mediana Empresa, documenta el crecimiento sostenido del registro pyme durante la última década. En el informe *Estado de Situación de la PYME en Costa Rica 2021*, se indica que el registro activo ha mostrado una tendencia creciente desde 2011 hasta 2021 (MEIC, 2021, p. 70). Este aumento evidencia dinamismo empresarial; sin embargo, también implica mayores exigencias en materia de formalización y gestión administrativa.

El crecimiento cuantitativo no garantiza madurez estructural. Muchas pymes presentan debilidades en sus procesos financieros internos, particularmente en áreas como control de gastos, conciliación de ingresos y clasificación contable. Bolaños et al. (2021), al analizar la estructura operativa de una asociación costarricense, advierten que la falta de organización contable puede generar “problemas de distribución de excedentes, falta de transparencia y control de datos” (p. 15). Aunque su estudio se desarrolló en un contexto distinto, el problema subyacente es comparable: cuando no existe un sistema estructurado, la información financiera pierde trazabilidad y consistencia.

A nivel regional, la discusión sobre digitalización productiva también evidencia la importancia de incorporar la tecnología dentro de los procesos empresariales. La Comisión Económica para América Latina y el Caribe (CEPAL) señala que “la integración de tecnologías digitales permite optimizar procesos, reducir costos, diversificar mercados y desarrollar nuevos modelos de negocio” (Patiño et al., 2025, p. 6).

Este planteamiento permite entender que la digitalización empresarial no se limita a incorporar herramientas tecnológicas, sino que busca generar mejoras concretas en la forma en que las organizaciones realizan sus actividades. En el caso de TecniTech Solutions CR, esta situación resulta especialmente relevante, ya que se trata de una empresa dedicada al desarrollo de soluciones informáticas para terceros, mientras que algunos de sus procesos internos de gestión financiera aún presentan oportunidades de automatización e integración.

La transformación digital, por tanto, debe analizarse en dos niveles:

- **Externo:** desarrollo de soluciones tecnológicas para clientes.
- **Interno:** digitalización de procesos administrativos y financieros propios.

El segundo nivel es el que fundamenta este proyecto. La automatización contable permite integrar facturación, registro de ingresos, control de gastos y generación de reportes financieros en un único sistema estructurado. Esta integración reduce errores humanos, mejora la disponibilidad de información y fortalece el control interno.

Además, el cumplimiento normativo exige formalidad en la gestión financiera. Bolaños et al. (2021) recomiendan explícitamente la inscripción ante el Ministerio de Hacienda y la implementación de sistemas alineados con normativa contable vigente como condición para garantizar información fidedigna (p. 103). Esto refuerza la idea de que la estructuración contable no es opcional, sino un requisito para la sostenibilidad empresarial.

Desde una perspectiva estratégica, la automatización financiera no solo mejora eficiencia operativa, sino que también impacta directamente la capacidad de planificación. Una empresa que visualiza en tiempo real su flujo de caja, sus cuentas por cobrar y su estructura de costos tiene mejores herramientas para tomar decisiones de inversión y crecimiento.

En una pyme en la que el propietario cumple simultáneamente funciones técnicas, administrativas y contables, la ausencia de automatización incrementa el riesgo operativo. No necesariamente por desconocimiento, sino por sobrecarga funcional y falta de control sistemático.

Por ello, el desarrollo de un sistema financiero-contable web no representa únicamente una mejora tecnológica; más bien, constituye un paso hacia la madurez organizacional, alineado con las tendencias de digitalización productiva regional y con las exigencias estructurales del entorno empresarial costarricense.

Integración de Procesos Financieros y Calidad de la Información

Uno de los problemas más comunes en las pequeñas empresas no es la falta de datos, sino la fragmentación de estos. Por ejemplo, se facturan servicios en un archivo, se registran pagos en otro, los gastos se anotan en una hoja distinta y los reportes se construyen manualmente cuando se

requieren. Esta separación genera lo que en términos de sistemas se conoce como “islas de información”.

Un sistema financiero integrado elimina esa segmentación al centralizar los procesos bajo una misma estructura de datos y reglas de negocio. La integración no es simplemente un tema tecnológico, se trata de un principio de control interno.

Chinchilla (2020) analiza la implementación de un sistema contable en una empresa costarricense y describe que el proceso debe incluir etapas de análisis de requerimientos, diseño, desarrollo, prueba, integración y operación (pp. 27–29). Esta secuencia demuestra que la integración no ocurre de forma espontánea; al contrario, debe planificarse y diseñarse estructuralmente. Con respecto a lo anterior, al carecer de esa integración, se afecta la calidad de la información contable por las siguientes razones:

- La facturación no impacta automáticamente las cuentas por cobrar.
- Los cobros no actualizan estados financieros en tiempo real.
- Los gastos no se clasifican bajo un catálogo estandarizado.
- La conciliación bancaria se realiza manualmente.
- Los reportes financieros dependen de cálculos externos.

La Ley N.º 8968, pese a estar orientada a la protección de datos personales, establece un principio aplicable a cualquier sistema de información. En su artículo 6 indica que los datos deben ser “actuales, veraces, exactos y adecuados al fin para el que fueron recolectados” (Asamblea Legislativa de la República de Costa Rica, 2011, art. 6). Aunque el artículo se refiere a datos personales, el principio de calidad es igualmente relevante para la información financiera. Si los registros contables no son exactos o están desactualizados, el sistema pierde confiabilidad.

Desde la perspectiva contable, la calidad de la información se sostiene en cuatro dimensiones fundamentales:

- **Integridad:** los datos deben estar completos y sin alteraciones no autorizadas.
- **Exactitud:** los valores registrados deben corresponder fielmente a la transacción real.
- **Oportunidad:** la información debe estar disponible en el momento en que se requiere.
- **Consistencia:** los criterios de clasificación deben mantenerse uniformes en el tiempo.

Bolaños et al. (2021) destacan que la ausencia de controles internos adecuados puede afectar la transparencia y el control de datos (p. 15). En su estudio, identifican que las inconsistencias operativas impactan directamente la capacidad de análisis financiero. Esto confirma que la calidad de la información no depende únicamente del conocimiento contable, sino del diseño estructural del sistema.

En un sistema web desarrollado bajo arquitectura API + frontend desacoplado, la integración se logra mediante reglas de negocio centralizadas en el backend. Esto permite que:

- Cada factura genere automáticamente un registro contable.
- Cada cobro actualice el estado de la cuenta correspondiente.
- Cada gasto se clasifique bajo un catálogo predefinido.
- Los reportes financieros se construyan a partir de consultas estructuradas y no cálculos manuales.

La integración también reduce el riesgo de error humano, puesto que los procesos que dependen de un ingreso manual repetitivo presentan mayor probabilidad de inconsistencias. En cambio, cuando el sistema valida datos en el momento de su registro, el margen de error disminuye significativamente.

Además, la trazabilidad se convierte en un elemento clave. Un sistema integrado debe permitir identificar:

- Quién registró una transacción.
- Cuando se registró.
- Qué modificaciones se realizaron.
- Bajo qué criterio contable se clasificó.

Desde la perspectiva tecnológica, esto conlleva la implementación de:

- Integridad referencial en base de datos (claves foráneas).
- Validaciones en backend.
- Registro de auditoría (logs).
- Control de acceso basado en roles.

La integración de procesos financieros no solo mejora la operación diaria, también fortalece la capacidad de análisis estratégico. Por otro lado, permite generar estados financieros en tiempo real, evaluar márgenes de rentabilidad y proyectar flujos de caja con mayor precisión.

En el caso de TecniTech Solutions CR, la implementación de un sistema integrado conecta facturación, cobros y gastos bajo una misma estructura. Esto elimina la dependencia de registros manuales dispersos y reduce la necesidad de conciliaciones externas.

En síntesis, la integración de procesos financieros es el puente entre la teoría contable y la ingeniería de *software* aplicada. No se trata únicamente de desarrollar módulos funcionales, sino de diseñar un sistema coherente donde cada transacción impacte automáticamente la estructura financiera global de la empresa.

Arquitectura de *Software* en Sistemas Empresariales

Cuando se desarrolla un sistema financiero-contable para una organización, la arquitectura de *software* no es una decisión secundaria. En este sentido, define la estructura fundamental del sistema, la forma en que sus componentes interactúan y la capacidad que tendrá para evolucionar en el tiempo. En un entorno empresarial, una arquitectura mal diseñada generaría dependencias rígidas, dificultades de mantenimiento y vulnerabilidades de seguridad.

En términos generales, la arquitectura de *software* establece la organización estructural del sistema, sus módulos, sus interfaces y las reglas que gobiernan su interacción. En el caso de sistemas empresariales webs modernos, el enfoque desacoplado entre frontend y backend se ha convertido en una práctica ampliamente adoptada, debido a su capacidad para mejorar escalabilidad y mantenibilidad.

Chinchilla (2020), al describir el proceso de implementación de un sistema contable, plantea que el desarrollo debe contemplar fases de diseño, desarrollo, prueba e integración (pp. 27–29). Aunque su estudio no profundiza específicamente en arquitectura web desacoplada, sí enfatiza la importancia de estructurar el sistema de forma planificada y alineada con los requerimientos organizacionales. Esto implica que la arquitectura debe responder a las necesidades reales del negocio, no únicamente a criterios tecnológicos.

En el presente proyecto, la arquitectura propuesta se basa en:

- Backend desarrollado en .NET Core.

- API como capa de servicios.
- Frontend desarrollado en React.
- Base de datos en Azure SQL Database.
- Infraestructura desplegada en Azure App Service y Azure Static Web Apps.

Este modelo corresponde a una arquitectura desacoplada, donde el frontend consume servicios expuestos por una API. La separación de responsabilidades permite que cada componente evolucione de forma independiente. El backend concentra las reglas de negocio, validaciones financieras y acceso a datos, mientras que el frontend gestiona la interacción con el usuario.

Desde la perspectiva de sistemas empresariales, esta separación genera ventajas estructurales:

- **Escalabilidad:** el backend puede escalar de forma independiente si aumenta la carga transaccional.
- **Mantenibilidad:** cambios en la interfaz no afectan la lógica financiera.
- **Seguridad:** las reglas de validación se centralizan en el servidor.
- **Reutilización:** el API podría consumirse en el futuro por otros clientes (por ejemplo, una aplicación móvil).

La arquitectura también debe considerar requisitos no funcionales, especialmente en un sistema financiero. Entre ellos:

- Seguridad.
- Disponibilidad.
- Integridad de datos.
- Rendimiento.
- Trazabilidad.

La Ley N.º 8968 establece en su artículo 10 que el responsable de una base de datos “debe adoptar las medidas de índole técnica y de organización necesarias para garantizar la seguridad de los datos” (Asamblea Legislativa de la República de Costa Rica, 2011, art. 10). Esta disposición no solo aplica a grandes corporaciones, cualquier sistema que almacene datos personales o financieros debe cumplir con este principio.

En una arquitectura desacoplada, la seguridad se refuerza al:

- Limitar acceso directo a la base de datos.
- Exponer únicamente *endpoints* controlados.
- Implementar autenticación y autorización en el backend.
- Utilizar HTTPS para cifrado en tránsito.

Además, la utilización de Azure como infraestructura introduce estándares de disponibilidad y resiliencia propios de servicios en la nube. Si bien es cierto que en este proyecto se usan planes gratuitos, la estructura está diseñada para escalar en el futuro sin rediseñar el sistema desde cero.

Otro elemento relevante es la coherencia entre arquitectura y modelo de desarrollo. Un diseño modular facilita la implementación incremental. Cada módulo (facturación, gastos, reportes) puede desarrollarse como componente independiente dentro de la misma arquitectura, manteniendo cohesión interna y bajo acoplamiento.

Desde una perspectiva técnica, el backend en .NET Core permite estructurar la solución bajo principios como separación de capas (presentación, aplicación, dominio e infraestructura), lo que mejora claridad y mantenibilidad del código. Aunque el proyecto no necesariamente implementa un patrón de arquitectura empresarial complejo (como microservicios), sí adopta principios de diseño que reducen dependencias innecesarias.

En síntesis, la arquitectura propuesta no es arbitraria, pues responde a buenas prácticas actuales de desarrollo web, necesidades de integración financiera, requisitos de seguridad normativa y posibilidad de crecimiento futuro. Por consiguiente, para una pyme como TecniTech Solutions CR, esta arquitectura representa un equilibrio entre simplicidad estructural y robustez técnica. No se trata de sobreingeniería, sino de diseñar una base sólida que soporte la operación financiera presente y la expansión futura.

Modelo Incremental Aplicado al Desarrollo del Sistema

El desarrollo de *software* en entornos empresariales pequeños requiere equilibrio entre planificación estructurada y capacidad de adaptación. En el caso de TecniTech Solutions CR, donde el equipo operativo es reducido y el propietario asume múltiples funciones, un modelo rígido

y completamente secuencial generaría retrasos innecesarios o sobrecarga técnica. Bajo ese contexto, el modelo incremental se presenta como una alternativa adecuada.

Lo anterior se debe a que el modelo incremental consiste en dividir el sistema en partes funcionales que se desarrollan y entregan de manera progresiva. Cada incremento incorpora nuevas funcionalidades, manteniendo la coherencia con la arquitectura general del sistema. A diferencia de un enfoque completamente lineal, el modelo incremental valida resultados parciales antes de avanzar a etapas posteriores.

Chinchilla (2020) detalla el proceso de implementación de un sistema contable y expone que este debe contemplar planificación, análisis de requerimientos, diseño, desarrollo, prueba, integración y operación (pp. 27–29). A pesar de que su planteamiento no utiliza explícitamente el término “modelo incremental”, la secuencia estructurada que describe evidencia la necesidad de organizar el desarrollo por fases claramente definidas.

En el presente proyecto, el modelo incremental se adapta a la naturaleza modular del sistema financiero-contable, dado que el sistema puede dividirse en incrementos funcionales tales como:

- Gestión de usuarios y roles.
- Módulo de facturación.
- Registro de gastos.
- Control de cuentas por cobrar.
- Reportes financieros.
- Conciliación básica.

Es posible que cada módulo se desarrolle, pruebe y valide de forma independiente antes de integrarse al sistema completo. Esta estrategia reduce riesgos técnicos y detecta errores en etapas tempranas. En adición a lo anterior, una ventaja clave del modelo incremental es la reducción del riesgo operativo. En sistemas financieros, cualquier error puede afectar cálculos contables, saldos acumulados o reportes tributarios. Al implementar incrementos controlados, se limita el impacto potencial de defectos, ya que cada módulo se valida antes de pasar al siguiente.

Si se analiza desde una perspectiva técnica, el modelo incremental es coherente con la arquitectura desacoplada definida anteriormente. Al mantener separación entre frontend, API y base de datos, cada incremento puede enfocarse en:

- Definir entidades en base de datos.
- Crear *endpoints* en el API.
- Implementar validaciones de negocio.
- Desarrollar interfaz correspondiente.
- Ejecutar pruebas unitarias e integrales.

Este enfoque favorece la trazabilidad del desarrollo y facilita la identificación de responsabilidades dentro del sistema. Además, el modelo incremental prioriza funcionalidades críticas. En el caso de TecniTech Solutions CR, resulta estratégico iniciar con módulos que impacten directamente el control financiero, como facturación y registro de gastos, antes de incorporar funcionalidades más analíticas o complementarias.

Es decir, visto desde el control interno, esta priorización permite que la empresa comience a obtener beneficios operativos desde etapas tempranas del proyecto, sin necesidad de esperar a la finalización total del sistema.

Otro elemento relevante es la alineación con los recursos disponibles. En una pyme con estructura reducida, el tiempo destinado al desarrollo debe administrarse cuidadosamente. El modelo incremental posibilita la organización del trabajo en bloques manejables y evita la acumulación de tareas sin resultados visibles.

En términos de calidad, cada incremento debe cumplir con criterios definidos previamente. Esto implica que antes de avanzar al siguiente módulo, se verifiquen aspectos como:

- Funcionamiento correcto de reglas de negocio.
- Validación adecuada de datos.
- Integridad referencial en base de datos.
- Coherencia entre interfaz y backend.
- Cumplimiento de requisitos definidos.

Ahora bien, el modelo incremental no elimina la necesidad de planificación, sino que la organiza en ciclos progresivos. Esto resulta particularmente adecuado cuando el proyecto se desarrolla como trabajo de graduación, donde se requiere demostrar metodología, control y validación técnica.

En síntesis, el modelo incremental no solo es una elección metodológica, sino una decisión estratégica alineada con múltiples rasgos, entre ellos:

- Modularidad del sistema.
- Arquitectura desacoplada.
- Disponibilidad de recursos.
- Necesidad de validación progresiva.
- Reducción de riesgo en sistemas financieros.

A partir de este modelo, es posible integrar prácticas ágiles que fortalezcan la gestión del proyecto, lo cual conduce naturalmente al análisis del marco Scrum como complemento metodológico.

Scrum Como Marco de Gestión en el Desarrollo del Sistema

El modelo incremental define cómo se estructura técnicamente el desarrollo del sistema; sin embargo, la gestión del proyecto requiere un marco que organice el trabajo, priorice funcionalidades y garantice el seguimiento continuo. En este contexto, Scrum se incorpora como un marco de gestión ágil que complementa el enfoque incremental adoptado.

Sobre esto, Scrum no es una metodología tradicional secuencial, sino un marco ligero que organiza el trabajo en ciclos cortos denominados *sprints*. Su propósito no es definir detalladamente cada fase técnica, sino proporcionar una estructura que facilite la adaptación y la entrega continua de valor.

Schwaber y Sutherland (2020) definen Scrum como “un marco dentro del cual las personas pueden abordar problemas adaptativos complejos, mientras entregan productos del más alto valor posible de manera productiva y creativa” (p. 3). Esta acepción resulta pertinente en el contexto del proyecto, ya que el desarrollo de un sistema financiero-contable en una pyme implica enfrentarse a requerimientos que pueden ajustarse conforme avanza la implementación. La comprensión de los procesos internos evoluciona a medida que se digitalizan, lo que exige flexibilidad en la gestión.

Desde una perspectiva de gestión, Scrum introduce tres elementos fundamentales:

- Transparencia.
- Inspección.

- Adaptación.

Estos principios permiten que el avance del proyecto sea visible, evaluado periódicamente y ajustado cuando sea necesario. En un entorno pequeño como TecniTech Solutions CR, donde el equipo está compuesto esencialmente por el propietario y una persona administrativa, la comunicación directa facilita la aplicación de este marco.

Resulta viable estructurar la gestión bajo Scrum en este proyecto de la siguiente manera:

- Cada módulo funcional (por ejemplo, facturación o registro de gastos) se planifica como un *sprint*.
- Se definen objetivos claros al inicio del ciclo.
- Se desarrolla el incremento funcional.
- Se revisa el resultado con el usuario.
- Se ajustan requerimientos para el siguiente ciclo.

Este enfoque favorece la retroalimentación temprana y evita el desarrollo prolongado sin validación intermedia. En sistemas financieros, en los que la precisión es crítica, la revisión continua reduce el riesgo de errores acumulativos. Además, Scrum promueve la priorización basada en valor. En el contexto del proyecto, esto implica que las funcionalidades que impactan directamente el control financiero deben desarrollarse primero, garantizando beneficios operativos desde etapas tempranas.

Otro aspecto relevante es la responsabilidad compartida. Aunque el proyecto es desarrollado como trabajo de graduación, la participación del propietario en la validación de incrementos asegura alineación con la realidad organizacional. Esto evita la desconexión entre diseño técnico y operación diaria.

En cuanto al punto de vista académico, la incorporación de Scrum también evidencia que el proyecto no solo contempla desarrollo técnico, sino gestión estructurada. La planificación iterativa, la definición de entregables parciales y la revisión continua fortalecen la trazabilidad del proceso. Es importante aclarar que Scrum no reemplaza el modelo incremental, lo complementa. Mientras el modelo incremental ordena el sistema por módulos funcionales, Scrum organiza la gestión del trabajo en ciclos controlados. Ambos enfoques convergen en la entrega progresiva y

validación constante. De esta manera, el proyecto no solo se fundamenta en teoría contable y arquitectura técnica, sino también en principios modernos de gestión de proyectos de *software*.

Sistemas web empresariales

Los sistemas web empresariales han evolucionado: pasaron de ser herramientas de consulta a convertirse en plataformas centrales para la gestión organizacional. En el contexto actual, donde la operación empresarial depende cada vez más de entornos digitales, el uso de aplicaciones web permite integrar procesos, centralizar información y facilitar acceso remoto seguro.

Un sistema web empresarial puede definirse como una aplicación basada en arquitectura cliente-servidor que opera a través de protocolos web, lo cual posibilita la interacción entre usuarios y bases de datos mediante interfaces accesibles desde distintos dispositivos. A diferencia de aplicaciones locales tradicionales, los sistemas web reducen la dependencia de instalaciones físicas y facilitan mantenimiento centralizado.

En el caso del presente proyecto, el sistema financiero-contable se desarrolla bajo una arquitectura web que permite:

- Acceso remoto controlado.
- Actualizaciones centralizadas.
- Integración con servicios en la nube.
- Escalabilidad progresiva.

Chinchilla (2020) resalta que la implementación de un sistema contable implica no solo el desarrollo de funcionalidades, sino la integración del sistema dentro del entorno operativo real de la organización (pp. 27–29). Esta integración es particularmente eficiente cuando el sistema se diseña como aplicación web, ya que elimina barreras de instalación y facilita el acceso controlado desde cualquier ubicación.

Desde una perspectiva empresarial, algunas ventajas estructurales que presentan los sistemas web relevantes para una pyme incluyen:

- Reducción de costos de infraestructura local.
- Centralización de información.
- Respaldo automático en servidores remotos.

- Mayor disponibilidad operativa.
- Escalabilidad sin rediseño completo.

La adopción de servicios en la nube, como Azure App Service y Azure SQL Database, permite alojar la aplicación sin necesidad de servidores físicos propios. Esto resulta coherente con la realidad de TecniTech Solutions CR, pues carece de infraestructura dedicada para la gestión financiera interna. Además, el uso de arquitectura API + frontend desacoplado fortalece la interoperabilidad. Un sistema web estructurado mediante servicios admite futuras integraciones, por ejemplo:

- Exportación de datos a sistemas tributarios.
- Integración con herramientas de análisis.
- Desarrollo posterior de aplicaciones móviles.

A nivel operativo, la centralización que ofrece un sistema web reduce la dispersión de archivos locales y evita versiones inconsistentes de documentos financieros. Cuando los datos residen en una única base estructurada, se garantiza su uniformidad y coherencia.

Otro elemento relevante es la disponibilidad. Un sistema web correctamente desplegado en la nube ofrece acceso continuo, sujeto a políticas de seguridad y autenticación. Esto favorece que el propietario de la empresa consulte reportes financieros o registre información desde distintas ubicaciones sin comprometer la integridad del sistema.

En términos de control interno, la implementación web facilita el uso de:

- Autenticación mediante credenciales.
- Control de roles y permisos.
- Registro de actividad de usuarios.
- Restricción de acceso según perfil.

La estructura web también favorece la aplicación de estándares de seguridad en tránsito mediante el uso de HTTPS, protegiendo la información financiera contra interceptaciones externas. Desde una perspectiva estratégica, el uso de un sistema web empresarial en una pyme tecnológica como TecniTech Solutions CR fortalece la coherencia entre su actividad comercial y su estructura

interna. Una empresa que desarrolla soluciones digitales para terceros debe mantener consistencia digital en su propia gestión operativa.

Seguridad de la Información y Protección de Datos

El desarrollo de un sistema financiero-contable implica la gestión de información sensible, tanto de carácter económico como personal. En consecuencia, la seguridad de la información no puede considerarse un complemento opcional del diseño técnico, sino un requisito estructural del sistema.

Desde una perspectiva conceptual, la seguridad de la información se fundamenta en tres principios básicos: confidencialidad, integridad y disponibilidad. Estos garantizan que los datos sean accesibles únicamente por usuarios autorizados, que no sean alterados de forma indebida y que se encuentren disponibles cuando se requieran.

En el contexto costarricense, la protección de datos personales está regulada por la Ley N.º 8968, Ley de Protección de la Persona Frente al Tratamiento de sus Datos Personales. En su artículo 1, establece que su finalidad es "garantizar a cualquier persona, independientemente de su nacionalidad, residencia o domicilio, el respeto a sus derechos fundamentales, concretamente, su derecho a la autodeterminación informativa" (Asamblea Legislativa de la República de Costa Rica, 2011, art. 1).

Este principio de autodeterminación informativa implica que cualquier sistema que almacene datos personales debe garantizar transparencia en su tratamiento y protección adecuada contra accesos no autorizados.

Además, el artículo 6 de la misma ley establece que los datos deben ser "actuales, veraces, exactos y adecuados al fin para el que fueron recolectados" (Asamblea Legislativa de la República de Costa Rica, 2011, art. 6). Aunque esta disposición se orienta a datos personales, el principio de calidad es plenamente aplicable a la información financiera gestionada por el sistema propuesto.

El componente más directamente relacionado con el diseño técnico se encuentra en el artículo 10, en el que se indica que el responsable de una base de datos "debe adoptar las medidas de índole técnica y de organización necesarias para garantizar la seguridad de los datos y evitar su alteración, pérdida, tratamiento o acceso no autorizado" (Asamblea Legislativa de la República de Costa Rica, 2011, art. 10). Este artículo tiene implicaciones directas sobre la arquitectura del

sistema. No basta con almacenar datos, se deben implementar mecanismos técnicos que prevengan riesgos.

El Reglamento a la Ley N.º 8968 (Decreto Ejecutivo N.º 37554-JP) refuerza este principio al desarrollar el concepto de autodeterminación informativa y establecer obligaciones específicas para el tratamiento de datos. En su artículo 12 define este derecho como:

> el derecho fundamental de toda persona física, a conocer lo que conste sobre ella en cualquier base de datos o registro público o privado, el fin para el que se utiliza dicha información, así como exigir que sea rectificada, actualizada, complementada o suprimida. (Procuraduría General de la República, 2012, art. 12).

En términos prácticos, esto implica que el sistema debe permitir:

- Identificación clara de los datos almacenados.
- Corrección o actualización de información cuando sea necesario.
- Protección contra alteraciones indebidas.
- Registro de accesos o modificaciones.

Desde el punto de vista técnico, el cumplimiento de estos principios en el sistema propuesto se traduce en:

- Autenticación de usuarios mediante credenciales seguras.
- Autorización basada en roles (por ejemplo, administrador y usuario administrativo).
- Uso de protocolo HTTPS para cifrado en tránsito.
- Restricción de acceso directo a la base de datos.
- Implementación de validaciones en backend.
- Registro de actividad para trazabilidad.

Además, el uso de infraestructura en la nube (Azure App Service y Azure SQL Database) introduce capas adicionales de seguridad administradas por el proveedor, tales como cifrado de datos en reposo y mecanismos de respaldo automático. Cabe señalar que, a pesar de que TecniTech Solutions CR es una pyme y el sistema se desarrolla como proyecto académico, la responsabilidad

legal no se reduce por el tamaño de la empresa. La normativa aplica a cualquier responsable de base de datos que trate información personal.

En cuanto a la perspectiva de gestión, la seguridad debe incorporarse desde el diseño (*security by design*), no como una etapa posterior. Esto significa que las decisiones arquitectónicas, las validaciones de datos y la estructura de autenticación deben contemplarse desde las primeras fases del desarrollo incremental.

En suma, la seguridad de la información en el sistema propuesto no se limita a proteger datos financieros, sino que garantiza cumplimiento normativo, protección de derechos fundamentales y reducción de riesgos operativos. La integración de principios legales y técnicos fortalece la solidez del proyecto y evidencia que el desarrollo no se limita a la programación de funcionalidades, sino que incorpora responsabilidad estructural en el tratamiento de la información.

CAPÍTULO III: MARCO METODOLÓGICO

Enfoques de Investigación

En este capítulo se define cómo se ejecuta el proyecto. Primero, se describen los enfoques cuantitativo, cualitativo y mixto. Después, se justifica el enfoque seleccionado para el desarrollo de un sistema web de gestión financiero-contable para TecniTech Solutions CR. La idea es simple: levantar requerimientos con la empresa, construir el prototipo y demostrar su funcionamiento con evidencia verificable.

Enfoque cuantitativo

El enfoque cuantitativo trabaja con datos numéricos y mide resultados de forma objetiva. Se utiliza cuando se requiere registrar valores, comparar indicadores o evaluar el desempeño de un producto mediante métricas. En proyectos de *software*, este enfoque es útil para documentar resultados de pruebas, cantidad de incidencias por módulo, porcentajes de casos aprobados y tiempos de respuesta en operaciones críticas (Hernández-Sampieri y Mendoza, 2023).

Enfoque cualitativo

El enfoque cualitativo busca comprender el problema desde el contexto real de la organización. Se apoya en técnicas como entrevistas, observación y revisión documental para identificar necesidades, reglas de negocio y puntos de mejora. En el desarrollo de sistemas, este enfoque es clave para que el producto responda a cómo trabaja la empresa, no a suposiciones (Hernández-Sampieri y Mendoza, 2023).

Enfoque mixto

El enfoque mixto combina técnicas cualitativas y cuantitativas para lograr una visión completa del problema y de la solución. Permite interpretar el contexto (por ejemplo, procesos actuales y requerimientos) y, al mismo tiempo, medir resultados de validación del prototipo (por

ejemplo, pruebas y métricas de calidad). Esta combinación implica un proceso sistemático de recolección, análisis e integración conjunta de datos de ambos tipos, lo que fortalece la consistencia de las conclusiones y la evidencia del proyecto (Hernández-Sampieri y Mendoza, 2023).

Enfoque de Investigación Seleccionado

Para este proyecto se selecciona un enfoque mixto, en vista de que se requiere comprender y documentar el proceso financiero-contable actual de TecniTech Solutions CR (parte cualitativa) y, posteriormente, verificar el comportamiento del sistema con resultados medibles (parte cuantitativa). Con esto, el diseño se basa en necesidades reales y la validación del prototipo se sustenta en evidencia.

En la práctica, el componente cualitativo se usa para levantar requerimientos, definir reglas de negocio y validar con usuarios clave. Por otro lado, el enfoque cuantitativo se emplea para registrar y consolidar resultados de pruebas, por ejemplo: porcentaje de casos aprobados, incidencias por módulo y cumplimiento de criterios de aceptación.

Tipos de Investigación

En este apartado, se presentan tipos de investigación aplicables a proyectos tecnológicos orientados a la solución de un problema real. Se incluyen tres tipos: investigación aplicada, estudio de caso e investigación de desarrollo tecnológico (prototipo). Posteriormente, se define el tipo seleccionado y se justifica su relación con el objetivo del proyecto (Hernández-Sampieri y Mendoza, 2023).

Investigación Aplicada

La investigación aplicada se orienta a resolver un problema concreto en un contexto real. Su valor está en producir un resultado utilizable (no solo conocimiento teórico) y en demostrar que la solución propuesta funciona para la organización. En este proyecto, la aplicación es directa: construir un sistema web que mejore el control y la integración de la gestión financiero-contable.

Estudio de Caso

El estudio de caso analiza una situación específica en una organización particular, considerando su contexto, actores y forma de operar. Este tipo de investigación permite documentar el proceso actual, identificar necesidades y establecer criterios de aceptación basados en la realidad de la empresa. En este proyecto, el caso corresponde a TecniTech Solutions CR y a su necesidad de pasar de registros manuales a una gestión integrada y trazable.

Investigación de Desarrollo Tecnológico (prototipo)

La investigación de desarrollo tecnológico se centra en diseñar y construir un artefacto (prototipo) como respuesta al problema identificado. El resultado esperado es un producto funcional que sea probado y validado. En el caso específico de este proyecto, el prototipo es el sistema web (API + frontend) con base de datos, desarrollado por incrementos y validado mediante pruebas y retroalimentación de usuarios.

Tipo de Investigación Seleccionado

Para este proyecto se selecciona principalmente un tipo de investigación aplicada. Se justifica porque el objetivo es resolver una necesidad real de TecniTech Solutions CR mediante un producto utilizable: un sistema web para la gestión financiero-contable. Operativamente, la investigación se ejecuta como un estudio de caso (la empresa y su proceso actual) y culmina con el desarrollo tecnológico de un prototipo funcional, el cual es probado y validado antes del cierre del proyecto.

Fuentes de Información

En toda investigación, es necesario identificar las fuentes de información que permitieron obtener los datos necesarios para comprender la situación actual del objeto de estudio. Las fuentes de información corresponden a los medios a través de los cuales se obtienen datos relevantes para

el desarrollo del proyecto. En el presente trabajo, se recurrió, principalmente, a fuentes de información primarias, ya que los datos se obtendrán directamente de las personas involucradas en los procesos administrativos y financieros de la empresa TecniTech Solutions CR. Estas fuentes propician que se conozca de forma directa cómo se realizan en la actualidad los procesos financiero-contables y las necesidades que el sistema propuesto busca atender.

Fuentes de Información Primarias

Se utilizaron fuentes primarias que, en este caso, corresponden a las personas que actualmente ejecutan y controlan la operación financiero-contable de la empresa TecniTech Solutions CR. En específico, se usaron entrevistas semiestructuradas para recopilar información de los siguientes funcionarios de la compañía:

- Dueño de la empresa (rol mixto: dueño, desarrollador y responsable de la gestión contable).
- Administrativa (apoyo en registros, seguimiento operativo y documentación de transacciones).

Fuentes de Información Secundarias

Aunque la empresa carece de documentación formal estandarizada, sí existe evidencia operativa útil para comprender el proceso real y definir campos, validaciones y reportes requeridos. Se consideraron como fuentes secundarias los registros disponibles, tales como:

- Facturas emitidas (PDF, correo o archivo digital).
- Comprobantes de pago y transferencias (SINPE / bancos).
- Estados de cuenta bancarios.
- Registros informales existentes (hojas de cálculo, notas, archivos de texto).
- Conversaciones o confirmaciones operativas en correo o mensajería (cuando aporten evidencia del flujo de cobro/pago).

Fuentes de Información Terciarias

Las fuentes terciarias se utilizan como herramientas de consulta para localizar y organizar información relevante sobre el tema de investigación. Estas fuentes incluyen repositorios académicos, entre ellos los universitarios; bases de datos digitales, como las de organismos internacionales y portales institucionales que facilitan el acceso a documentos científicos, investigaciones previas y normativa relacionada con el proyecto, dado que proporcionan información sobre el contexto empresaria y tecnológico.

Variables de Investigación

Las variables de investigación se derivan directamente de los objetivos específicos del proyecto. Cada objetivo permite identificar un elemento clave que debe ser analizado, diseñado, implementado o evaluado durante el desarrollo del sistema. Para cada variable se establece su definición conceptual y operacional, así como el instrumento mediante el cual se recolectó la información necesaria para el desarrollo del proyecto.

Variable Conceptual

La variable conceptual corresponde a la definición teórica del elemento que se desea analizar dentro de la investigación. Esta definición favorece la comprensión del significado de la variable dentro del contexto del proyecto y establece el marco conceptual que orienta su estudio. En este caso, las variables conceptuales están relacionadas con los componentes fundamentales del desarrollo del sistema, como los requerimientos del sistema, el diseño del modelo de datos, la implementación de los módulos y la calidad del prototipo desarrollado.

Variable Operacional

Con respecto a la variable operacional, esta describe la forma en que la variable conceptual se manifiesta o se mide dentro del proceso de investigación. En otras palabras, indica cómo se

observó o evaluó la variable en la práctica. En el presente proyecto, las variables operacionales se reflejan en productos concretos del desarrollo del sistema, tales como la lista de requerimientos identificados, el modelo entidad-relación del sistema, los módulos implementados y los resultados durante las pruebas del prototipo.

Variable Instrumental

La variable instrumental hace referencia a los instrumentos o técnicas utilizadas para recolectar la información necesaria para analizar cada variable. Estos instrumentos posibilitan la obtención de datos relevantes para el desarrollo del sistema y valida los resultados durante el proceso de investigación. Para este proyecto, se usaron principalmente entrevistas semiestructuradas, revisión documental, listas de verificación de desarrollo y matrices de casos de prueba.

A continuación, se presenta la tabla de variables de investigación, en la cual se relacionan los objetivos específicos del proyecto con las variables identificadas, así como sus definiciones conceptuales, operacionales y los instrumentos utilizados para su análisis.

Tabla **4**
Variables de la investigación.

Objetivo específico	Variable	Variable conceptual	Variable instrumental	Variable operacional
Analizar los requerimientos funcionales y no funcionales del sistema web.	Requerimientos del sistema	Conjunto de necesidades funcionales (qué debe hacer) y no funcionales (cómo debe comportarse) del sistema financiero-contable.	Entrevista semiestructurada; revisión documental.	Requerimientos identificados

Construir el diseño de la base de datos y del sistema web.	Diseño del sistema y modelo de datos	Estructura lógica del sistema (componentes) y del almacenamiento (entidades, relaciones y restricciones) para soportar la gestión financiera-contable.	Revisión documental; casos de uso; Documentos de análisis	MySQL Workbench
Programar los diferentes módulos del sistema.	Implementación de módulos	Construcción de funcionalidades del sistema conforme a los módulos definidos para la gestión financiero-contable.	Diagrama entidad-relación Diagramas UML	Visual Studio Code
Probar los diferentes módulos de forma independiente e integral.	Calidad del prototipo	Grado en que el prototipo cumple requerimientos, funciona correctamente por módulo y mantiene consistencia al integrarse.	Casos de prueba	Guía de casos de prueba

Fuente: Elaboración propia, 2026.

Población

La población corresponde a las personas que participan directamente en la ejecución y control del proceso financiero-contable en TecniTech Solutions CR. Por la naturaleza de la empresa, (pyme en etapa inicial), la población está conformada por dos personas:

N = 2 (dueño y administrativa).

Muestra

Se trabajó con un muestreo censal, es decir, se incluyó a la totalidad de la población. En consecuencia, no se aplicó fórmula de cálculo de tamaño de muestra.

n = 2 (censo del 100% de la población).

Instrumentos de Recolección de Datos

Para obtener la información necesaria para el desarrollo del proyecto, se definieron instrumentos de recolección de datos que permiten identificar la situación actual de la empresa y los requerimientos del sistema propuesto. Los instrumentos para recopilar datos corresponden a las herramientas utilizadas para obtener información de las fuentes primarias de la investigación. En este estudio, se emplearon principalmente entrevistas semiestructuradas y revisión documental, ya que ayudan a comprender cómo se realizan actualmente los procesos financieros de la empresa y qué necesidades debe atender el sistema web que se desarrolló.

A continuación, se describen los instrumentos empleados en el proceso de recolección de información.

Entrevista Semiestructurada

La entrevista semiestructurada fue el instrumento principal para levantar requerimientos, reglas de negocio, validaciones, prioridades y necesidades de información relacionadas con la gestión financiera de la empresa TecniTech Solutions CR. Este tipo de entrevista permite obtener información directamente de las personas involucradas en los procesos administrativos y financieros, facilitando la identificación de los procedimientos actuales y las necesidades que debió cubrir el sistema propuesto.

La entrevista se aplicó al propietario de la empresa y al personal administrativo, quienes actualmente participan en la gestión de los registros financieros y operativos. Para su aplicación, se diseñó una guía de preguntas que orienta la conversación y favorece la obtención de datos relevantes sobre los procesos de facturación, registro de gastos, control de cuentas por cobrar y cuentas por pagar, así como los requerimientos de reportes y control financiero.

Revisión Documental

La revisión documental se usó como técnica complementaria para analizar la evidencia operativa disponible en la empresa. Esta técnica propicia la identificación de las estructuras reales de los datos utilizados en los procesos financieros, tales como campos de información, formatos de registro y reglas de negocio utilizadas en la operación diaria.

A través de esta revisión, se analizaron documentos generados durante la gestión administrativa de la empresa, los cuales permiten comprender cómo se registran en la actualidad las operaciones financieras y qué información fue necesaria para el diseño del modelo de datos del sistema propuesto. La revisión documental no depende de documentación formal, sino de los registros que la empresa genera como parte de su operación cotidiana.

Observación Guiada

De forma complementaria, se realizó una observación guiada breve del proceso actual para confirmar pasos reales y validaciones manuales. La observación se enfocó en ejecutar y documentar un ejemplo completo de:

1. Emisión de factura.
2. Registro de un pago o gasto, incluyendo los puntos donde hoy se realizan controles manuales.

Con base en la estructura de la empresa (N=2), la estrategia metodológica se enfoca en obtener información directa y verificable mediante entrevistas semiestructuradas y revisión de evidencia operativa disponible. Esto permite levantar requerimientos reales, diseñar el modelo de datos con campos correctos y validar el prototipo con pruebas y aceptación del usuario, con el fin de asegurar coherencia entre lo investigado, lo construido y lo demostrado.

Proceso de Recolección y Análisis de Datos

La recolección de datos se realiza combinando técnicas cualitativas y cuantitativas, alineadas al enfoque mixto seleccionado. El objetivo es obtener información suficiente para diseñar el sistema y, luego, validar el prototipo mediante evidencia verificable.

Técnicas Cualitativas (levantamiento y validación de requerimientos):

- Entrevistas semiestructuradas con personal clave (administración/contabilidad): identificación de necesidades, reglas y problemas actuales.
- Revisión documental de registros actuales (plantillas, facturas, comprobantes, reportes manuales): comprensión del flujo real y datos requeridos.
- Observación del proceso actual (si aplica): confirmación de pasos y validaciones que hoy se realizan de forma manual.

Técnicas Cuantitativas (validación del prototipo):

- Ejecución de casos de prueba por módulo y registro de resultados (aprobado/rechazado, incidencias, severidad).
- Consolidación de métricas básicas de calidad (porcentaje de casos aprobados, número de errores por módulo, tiempo promedio de respuesta en operaciones críticas).
- Validación de reportes mediante comparación de resultados esperados vs. obtenidos con datos de prueba.

La información recolectada se documentó en artefactos de análisis (lista de requerimientos, reglas de negocio, modelos de datos y casos de prueba), los cuales sirvieron como insumo directo para el diseño y la construcción del sistema.

Plan de Actividades por Realizar

El plan de actividades se organiza en etapas coherentes con el desarrollo de un sistema web por incrementos. Cada etapa produce entregables verificables que permiten controlar el avance y asegurar calidad antes de integrar módulos.

Tabla 5

Plan de actividades.

Etapa	Actividades principales	Técnicas / herramientas	Entregables	Evidencia / validación
Levantamiento	Entrevistas, revisión documental, levantamiento de requerimientos funcionales y no funcionales.	Entrevistas; revisión de documentos; *checklist* de requerimientos.	Lista de requerimientos; reglas de negocio; alcance detallado.	Aprobación de requerimientos por usuario clave.
Diseño	Diseño de arquitectura (API + frontend), modelo de datos y prototipos de pantalla.	Diagramas; modelado BD (Workbench); diseño UI.	Modelo ER; diagramas; prototipos; diccionario de datos.	Revisión de diseño con criterios de consistencia y trazabilidad.
Desarrollo incremental	Implementación por módulos priorizados (ej.: catálogos + facturación + CxC; luego gastos + CxP; luego reportes/presupuesto).	.NET Core; React; Azure SQL; control de versiones.	Módulos implementados por incremento; *endpoints* y UI.	Revisión por Sprint / incremento; *checklist* de criterios de aceptación.
Pruebas	Pruebas unitarias y pruebas integrales por flujo (facturación-cobro, proveedor-pago, gasto-presupuesto).	Casos de prueba; datos de prueba; bitácora de incidencias.	Matriz de pruebas; reporte de incidencias; correcciones.	Porcentaje de casos aprobados; trazabilidad de correcciones.

Despliegue y cierre	Despliegue en Azure (App Service + Static Web Apps) y validación final con usuarios.	Azure App Service; Azure SWA; Azure SQL.	Prototipo desplegado; manual breve de usuario; reporte final.	Acta de validación / retroalimentación; evidencias de ejecución.

Fuente: Elaboración propia, 2026.

Este plan permite que el proyecto se ejecute con control y evidencia: primero, se entiende el problema (levantamiento); luego, se define la solución (diseño); después, se construye por incrementos (desarrollo incremental) y se valida con pruebas y retroalimentación. Por lo anterior, se asegura coherencia entre lo investigado, lo construido y lo demostrado.

CAPÍTULO IV: ANÁLISIS DE RESULTADOS

Presentación de los Resultados

En esta sección, se exponen de manera sistemática los hallazgos tras la aplicación de los instrumentos de recolección de datos en la empresa TecniTech Solutions CR. Los datos se presentan mediante cuadros que muestran la brecha entre la situación actual y los requerimientos del sistema propuesto.

La entrevista semiestructurada se aplicó a dos actores clave dentro de la operación de TecniTech Solutions CR: el Administrador de la empresa, encargado de la gestión general del negocio y con conocimiento directo de los procesos de facturación y control financiero, y la jefa Administrativa, responsable del registro diario de gastos, pagos a proveedores y control de cuentas por cobrar y por pagar.

El objetivo de la entrevista fue identificar cómo se ejecutaban los procesos financieros y contables de la empresa, así como determinar las limitaciones operativas y los requerimientos funcionales que debía cubrir el sistema propuesto. Específicamente, se buscó levantar información sobre los procesos de facturación, registro de gastos, control de cuentas por cobrar y por pagar, manejo del tipo de cambio, control de acceso a la información y trazabilidad de los movimientos financieros.

Como resultado de la entrevista, se identificó que:

- Los registros financieros se llevaban mediante archivos de Excel independientes y sin integración entre sí, lo que generaba duplicidad de información y riesgo de pérdida de datos.
- Los asientos contables se registraban sin validación automática de partida doble, lo que provocaba descuadres frecuentes al conciliar la información.
- El acceso a los archivos financieros era abierto para el personal que los utilizaba, sin un control de acceso basado en roles.
- No existía un registro de auditoría que permitiera rastrear los cambios realizados sobre la información financiera.
- La facturación se emitía de forma manual, sin registro ante la Administración Tributaria, lo que exponía a la empresa a riesgo fiscal.

Estos hallazgos se sintetizan en la Tabla 6, la cual muestra la brecha entre la situación actual y los requerimientos del sistema propuesto.

Tabla 6
Resumen de hallazgos.

No.	Aspecto de Gestión Observado	Estado Actual	Impacto en la Operación	Requisitos del sistema
1	Centralización de la Información	No cumple. Datos en archivos Excel locales.	Duplicidad y riesgo de pérdida de datos.	Base de datos centralizada
2	Validación de Partida Doble	Manual. No existe restricción de sistema.	Descuadres financieros frecuentes.	Validación automática
3	Control de Acceso y Roles	Inexistente. Acceso abierto a archivos.	Vulnerabilidad de datos sensibles.	Control de acceso basado en roles (RBAC)
4	Trazabilidad de Movimientos	No existe log de auditoría.	Imposibilidad de rastrear cambios.	Registro de auditoría
5	Integración con Hacienda	Inexistente. Uso de facturación manual sin registro ante la Administración Tributaria.	Alto riesgo fiscal y operativo.	Facturación electrónica integrada

Fuente: Elaboración propia, 2026.

Análisis de los Resultados

El análisis de los resultados realiza un cruce entre los hallazgos empíricos, los fundamentos teóricos de la ingeniería de *software* y el criterio técnico del investigador.

1. **Cruce con la Teoría:** De acuerdo con las mejores prácticas de Clean Architecture, la lógica de negocio debe centralizarse para garantizar consistencia transaccional, evitando validaciones dispersas en múltiples capas del sistema.
2. **Seguridad y Auditoría:** La ausencia de control detectada se contrapone a los estándares de seguridad de la información. Esto se justifica implementando RBAC bajo el principio de mínimo privilegio, restringiendo el acceso únicamente a las operaciones necesarias según el rol.
3. **Automatización:** La entrevista reveló que el tipo de cambio se registra de forma manual, lo cual impacta directamente la confiabilidad de los estados financieros y la toma de decisiones.
4. **Integración de Procesos:** Los hallazgos evidencian que los procesos financieros se encuentran aislados, lo que genera inconsistencias en la información. Desde la perspectiva de sistemas de información, la integración de módulos propicia que cada transacción impacte automáticamente la estructura financiera, reduciendo errores y mejorando la trazabilidad.
5. **Calidad de la Información:** La ausencia de validaciones y controles afecta directamente la calidad de los datos financieros. Según los principios de integridad, exactitud y consistencia, el sistema propuesto debe garantizar que la información registrada sea confiable y utilizable para la toma de decisiones.

Diagnóstico de la Situación Actual

El diagnóstico integral de TecniTech Solutions CR refleja un escenario de vulnerabilidad operativa y técnica. La empresa carece de una fuente única y centralizada de información, lo que se traduce en:

- **Inconsistencia de Datos:** Al no existir integridad referencial (como la que provee Azure SQL), los nombres de clientes o proveedores varían entre archivos, lo que dificulta la búsqueda de información histórica.
- **Riesgo de Incumplimiento Normativo:** La empresa no realiza actualmente facturación electrónica ni reporta sus comprobantes a la Administración Tributaria, operando únicamente con facturación manual no registrada. Esta situación expone a la entidad a posibles sanciones, inconsistencias fiscales y falta de respaldo legal en sus transacciones. Lo anterior evidencia la necesidad de implementar un sistema que automatice la generación, validación y envío de comprobantes electrónicos, así como la gestión de su aceptación o rechazo por parte de la Administración Tributaria.
- **Dependencia Humana:** Los procesos dependen del conocimiento de personas específicas y no de un sistema institucionalizado, lo que genera cuellos de botella operativos y riesgo de interrupción en la continuidad del proceso financiero.

Este escenario no solo limita la eficiencia operativa, sino que compromete la confiabilidad de la información financiera utilizada para la toma de decisiones estratégicas.

Conclusiones del Diagnóstico

Como resultado del diagnóstico, se establecen las siguientes conclusiones que fundamentan la necesidad del desarrollo del sistema:

1. **Indispensabilidad de la Nube:** Se recomienda el uso de infraestructura en la nube como base del sistema, para permitir acceso remoto seguro, alta disponibilidad y reducción de dependencia de infraestructura local.
2. **Priorización del Módulo Contable:** La corrección del problema de la partida doble es la prioridad. El sistema debe impedir el guardado de asientos que no estén balanceados a nivel de base de datos (chk_debit_credit), garantizando integridad financiera desde la capa de datos y no únicamente a nivel de aplicación.
3. **Viabilidad Técnica:** Se confirma que la empresa cuenta con los requisitos de conectividad necesarios para implementar una arquitectura desacoplada (React + .NET

Core 8), lo cual moderniza la gestión sin requerir inversiones excesivas en *hardware*, alineándose con un enfoque de optimización de costos operativos para una pyme.

4. **Fundamento de la Propuesta:** El diagnóstico justifica plenamente el paso a la fase de construcción, pues demuestra que el sistema propuesto no es solo una mejora de *software*, sino una herramienta de control interno indispensable para la continuidad del negocio.
5. **Necesidad de Respaldo y Recuperación:** Se identifica la ausencia de mecanismos de respaldo de la información, por lo que el sistema debe incorporar estrategias de *backup* automático y recuperación ante fallos, garantizando la disponibilidad y continuidad del servicio.

Los resultados y el diagnóstico realizado establecen las bases para la propuesta del sistema presentada en el capítulo siguiente, donde se definen la arquitectura, el modelo de datos y los componentes necesarios para resolver las problemáticas identificadas.

CAPITULO V: PROPUESTA

Análisis

En este apartado se presenta el análisis técnico y funcional que sustenta la propuesta de solución para TecniTech Solutions CR. El análisis abarca los componentes de software y hardware requeridos, la arquitectura de telecomunicaciones sobre la cual opera el sistema, las herramientas empleadas durante el desarrollo, el conocimiento técnico y funcional necesario por parte de los usuarios, así como el diseño general de la base de datos. En conjunto, estos elementos permiten determinar la viabilidad técnica del sistema propuesto y establecer las bases necesarias para su posterior diseño e implementación.

Software del Sistema

El sistema web para la gestión financiero-contable de TecniTech Solutions CR se desarrolla con el objetivo de centralizar los procesos financieros y administrativos de la empresa dentro de una sola plataforma. La solución busca reducir la dependencia de registros manuales, minimizar inconsistencias en la información y mejorar el control sobre las operaciones contables realizadas diariamente.

La integración entre módulos permite que los movimientos registrados dentro del sistema generen actualizaciones automáticas en los distintos procesos financieros, manteniendo coherencia entre facturación, ingresos, gastos, cuentas pendientes y registros contables. De este modo, la plataforma facilita el acceso a información confiable y reduce el tiempo requerido para tareas operativas y de conciliación.

Módulo de Seguridad. El acceso al sistema esta controlado mediante autenticación de usuarios y asignación de permisos según el perfil correspondiente. Este módulo tendrá la responsabilidad de proteger la información financiera almacenada y restringir el acceso a funciones específicas según el rol de cada usuario dentro de la organización. La implementación de perfiles y permisos controla operaciones sensibles como modificaciones de registros, acceso a reportes

financieros o administración de configuraciones generales. Esto mantendrá un entorno más seguro y ordenado dentro de la plataforma.

Módulo de Facturación. La emisión de facturas se gestiona de forma automatizada, lo que posibilita registrar la información del cliente, aplicar impuestos y generar consecutivos de manera inmediata. El sistema reduce errores relacionados con cálculos manuales y mantendrá un control más preciso sobre las transacciones realizadas. Cada factura generada se vincula con los procesos de ingresos y cuentas por cobrar, así mantiene trazabilidad completa desde la creación del documento hasta la cancelación de la obligación pendiente. Por tanto, se facilita el seguimiento financiero y mejora el control administrativo de la empresa.

Módulo de Cuentas por Cobrar. El control de saldos pendientes de clientes se realiza desde el módulo de cuentas por cobrar. Aquí se registran pagos parciales o totales, fechas de vencimiento y estados de cada factura asociados a un cliente específico. El sistema actualiza de modo automático los montos restantes conforme se registren pagos, con esto reduce inconsistencias entre movimientos financieros y documentos emitidos. También, se identifican obligaciones vencidas y genera alertas relacionadas con morosidad o atrasos en cobros.

Módulo de Cuentas por Pagar. Las obligaciones financieras adquiridas con proveedores se administran desde este módulo, permitiendo registrar facturas, controlar vencimientos y mantener seguimiento sobre pagos pendientes. La plataforma ejecuta validaciones para evitar registros duplicados y mantendrá un mejor control sobre compromisos financieros de la empresa. A su vez, los movimientos generados impactan automáticamente los registros contables relacionados con cada operación realizada.

Módulo de Ingresos. Los ingresos registrados dentro del sistema están vinculados con las operaciones financieras ejecutadas por la empresa, evitando diferencias entre facturación, movimientos económicos y registros contables. El módulo valida automáticamente la información ingresada y genera los movimientos contables correspondientes sin necesidad de procesos manuales adicionales. Esto favorece una mayor precisión en los datos financieros y facilita el análisis de ingresos según periodos específicos.

Módulo de Gastos. El registro de gastos clasifica los egresos de la empresa según categorías y centros de costo previamente definidos. Esto agiliza el análisis financiero e identifica cómo se distribuyen los recursos dentro de la organización. También, se incorporan controles relacionados con límites presupuestarios y validaciones sobre disponibilidad financiera, ayudando a detectar desviaciones antes de que afecten significativamente el flujo económico de la empresa.

Módulo de Presupuesto. La planificación financiera se administra mediante el módulo de presupuesto, donde se definen montos proyectados y se comparan con los valores reales registrados en el sistema. La integración con ingresos y gastos posibilita la visualización de variaciones presupuestarias en tiempo real, facilitando la detección de diferencias entre lo planificado y lo ejecutado. Esto aporta mayor control sobre el desempeño financiero de la organización.

Módulo de Cuentas Contables. El módulo de cuentas contables tendrá la responsabilidad de administrar el catálogo de cuentas empleado dentro del sistema y mantener la estructura financiera de la empresa organizada de forma adecuada. Los movimientos generados desde otros módulos producen automáticamente los asientos contables correspondientes, por lo que se mantendrá consistencia entre operaciones financieras y registros contables. Gracias a ello, mejora la confiabilidad de la información utilizada para reportes y estados financieros.

Módulo de Mantenimientos. La administración de información general y de las configuraciones del sistema se realiza desde el módulo de mantenimientos. Este apartado registra, modifica o actualiza datos usados por distintos procesos de la plataforma. Su función principal es mantener organizada la información base necesaria para el funcionamiento del sistema, facilitando la administración y actualización de parámetros operativos.

Módulo de Consultas. La consulta de información financiera y administrativa se lleva a cabo mediante filtros y búsquedas sobre los registros almacenados en la base de datos. Este módulo facilita la localización rápida de información histórica y movimientos específicos dentro de la plataforma. La disponibilidad de consultas centralizadas reduce tiempos de búsqueda y mejora el acceso a información utilizada frecuentemente por los usuarios.

Módulo de Reportes. La generación de reportes presenta información financiera de forma estructurada para apoyar procesos de análisis y toma de decisiones. Los usuarios obtendrán reportes relacionados con ingresos, gastos, cuentas pendientes y movimientos contables registrados dentro del sistema. La información se ve directamente en pantalla o podrá imprimirse, así se simplifica la revisión de resultados financieros y el seguimiento de indicadores relevantes para la empresa.

Hardware Requerido

Para la implementación del sistema web de gestión financiero-contable, se cuenta con un equipo con capacidad suficiente para ejecutar simultáneamente herramientas de desarrollo, servicios de base de datos, pruebas locales y plataformas de despliegue en la nube. El entorno de trabajo está conformado por una computadora portátil MacBook Air con procesador Apple M1, el cual proporciona el rendimiento necesario para trabajar con tecnologías modernas de desarrollo web y soluciones de virtualización ligera.

Durante el proceso de desarrollo se utilizan entornos de programación, herramientas de control de versiones, motores de base de datos y navegadores para pruebas funcionales del sistema. Debido a que gran parte de la infraestructura se despliega en servicios cloud de Microsoft Azure, no es necesario contar con servidores físicos locales para la ejecución del sistema.

***Hardware* Utilizado para el Desarrollo.** Para el desarrollo del sistema se utiliza un equipo de cómputo con características adecuadas para la ejecución de herramientas de programación, bases de datos y pruebas de *software*. La Tabla 7 presenta las especificaciones técnicas del equipo empleado durante las actividades de análisis, diseño, implementación y validación del prototipo.

Tabla 7
Especificaciones del equipo de desarrollo.

Componente	Especificación
Equipo	MacBook Air
Procesador	Apple M1
Memoria RAM	8 GB

Almacenamiento	SSD
Sistema operativo	macOS

Fuente: Elaboración propia, 2026.

Las características del equipo permiten ejecutar de manera estable herramientas como Visual Studio Code, SQL Server mediante contenedores, navegadores web para pruebas y servicios de desarrollo relacionados con .NET y React. El almacenamiento SSD contribuye a mejorar los tiempos de compilación, carga de proyectos y ejecución de procesos locales.

***Hardware* Requerido para la Implementación.** La solución propuesta funciona bajo una arquitectura web cliente-servidor, por lo que los usuarios únicamente necesitan un equipo con acceso a internet y un navegador actualizado para utilizar el sistema.

En cuanto a la infraestructura en la nube, el sistema OneCore se despliega sobre los siguientes servicios de Microsoft Azure, distribuidos en dos regiones según el recurso:

Tabla 8.
Especificaciones del servicio en la nube (Microsoft Azure)

Componente	Servicio Azure	Especificación	Región
API backend	App Service	Plan F1 (Free tier)	Central US
Frontend	Static Web Apps	Plan gratuito	Central US
Gateway de API	API Management	Capa Consumption	-
Base de datos	Azure SQL Database	General Purpose Serverless, Gen5, 2 vCores (oferta gratuita)	West US 2

Fuente: Elaboración propia, 2026.

Es importante señalar que los niveles de servicio utilizados (Free tier y Serverless) corresponden a la etapa de desarrollo y validación del prototipo, ya que ofrecen el rendimiento necesario para las pruebas funcionales del sistema sin incurrir en costos de infraestructura durante el proyecto. Para un entorno de producción con uso continuo por parte de TecniTech Solutions CR, se recomienda migrar únicamente la base de datos a un nivel de servicio de pago (por ejemplo, Azure SQL Database Standard), dado que el nivel Serverless gratuito puede introducir latencia adicional al reactivarse tras períodos de inactividad, algo crítico para un sistema que centraliza la operación financiera diaria de la empresa. El App Service en capa Free puede mantenerse mientras

el volumen de usuarios concurrentes sea bajo, ya que no afecta la integridad de los datos, solo el tiempo de respuesta inicial.

Debido a que la aplicación es desplegada sobre infraestructura cloud de Microsoft Azure, el procesamiento principal, el almacenamiento de datos y la ejecución de servicios se realiza de forma remota. Esto reduce considerablemente los requerimientos de hardware para los usuarios finales y facilita el acceso desde diferentes ubicaciones.

Tabla 9
Requerimientos mínimos para uso del sistema.

Componente	Especificación
Procesador	Dual Core
Memoria RAM	4 GB
Almacenamiento	20 GB disponibles
Navegador	Google Chrome, Microsoft Edge o Mozilla Firefox
Conectividad	Acceso estable a internet

Fuente: Elaboración propia, 2026.

El sistema es viable desde computadoras portátiles o de escritorio sin necesidad de instalar *software* adicional, ya que el acceso se hace directamente mediante navegador web. Esto facilita la implementación dentro de la empresa y reduce costos asociados a instalación y mantenimiento de infraestructura local. Adicionalmente, el uso de servicios cloud permite mejorar la disponibilidad de la plataforma, realizar respaldos automáticos y mantener una mayor estabilidad operativa frente a soluciones tradicionales ejecutadas únicamente en equipos locales.

Telecomunicaciones

El sistema financiero-contable se implementa bajo una arquitectura web cliente-servidor, lo que posibilita a los usuarios acceder a la aplicación desde cualquier ubicación por medio de un navegador web y una conexión a internet. Esta arquitectura centraliza la información financiera de la empresa y distribuye las responsabilidades entre los diferentes componentes del sistema.

La solución está conformada por una aplicación web desarrollada en React, un gateway configurado en Azure API Management, un API desarrollado en ASP.NET Core y una base de datos Azure SQL Database. Cada uno de estos componentes cumple una función específica, lo que facilita el mantenimiento del sistema y favorece que futuras mejoras se lleven a cabo sin afectar el funcionamiento del resto de la solución.

La interfaz de usuario se publica a través de Azure Static Web Apps y todas las solicitudes generadas por los usuarios pasan primero por Azure API Management. Este servicio actúa como punto de entrada único hacia la aplicación, encargándose de dirigir las solicitudes al API y de aplicar las políticas de seguridad y control de acceso definidas para el sistema.

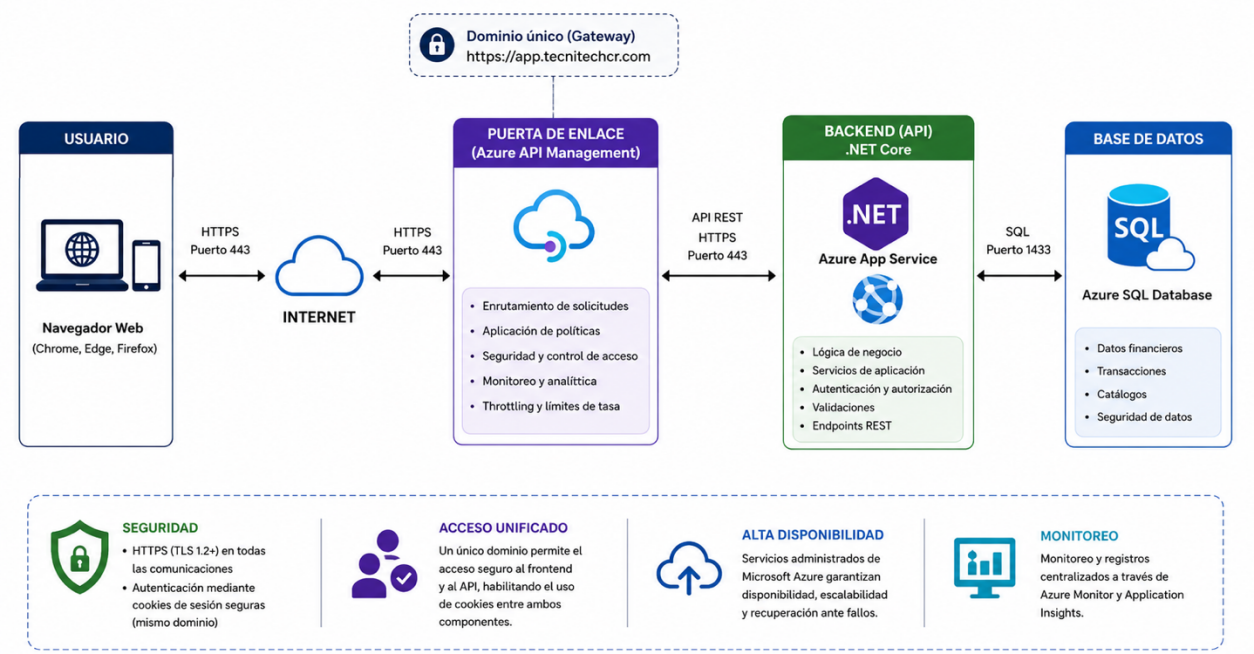
El procesamiento de la información se efectua en un API desarrollado en ASP.NET Core y situado en Azure App Service. En esta capa se concentra la lógica de negocio, las validaciones, la autenticación de usuarios y el acceso a la base de datos. La comunicación entre los componentes se lleva a cabo mediante servicios REST utilizando el formato JSON sobre el protocolo HTTPS.

Como parte de la estrategia de seguridad, la autenticación se implementa con tokens JSON Web Token (JWT) almacenados en cookies HttpOnly, las cuales son administradas bajo un dominio unificado a través de Azure API Management. De esta manera, el navegador envia automáticamente la información de sesión en cada solicitud, sin exponer el token al código JavaScript de la aplicación, de modo que se reducen los riesgos asociados a ataques como el robo de credenciales por medio de *scripts* maliciosos.

Toda la infraestructura se despliega sobre Microsoft Azure. El frontend se hospeda en Azure Static Web Apps, el gateway es administrado por Azure API Management, el API se ejecuta en Azure App Service y la información se almacena en Azure SQL Database. Esta configuración aprovecha servicios administrados que simplifican las tareas de despliegue, mantenimiento y monitoreo, además de ofrecer alta disponibilidad para la aplicación.

En conjunto, esta arquitectura proporciona una solución desacoplada, segura y escalable, preparada para soportar los procesos financieros, contables y administrativos de TecniTech Solutions CR, así como el crecimiento futuro del sistema.

Figura 2.
Arquitectura de telecomunicaciones propuesta.



Fuente: Elaboración propia, 2026.

La Figura 2 muestra la arquitectura de telecomunicaciones propuesta para el sistema. La aplicación cliente se desarrolla utilizando React y se despliega mediante Azure Static Web Apps. Todas las solicitudes del cliente hacia el backend se dirigen a través de Azure API Management, configurado como puerta de enlace bajo un dominio único que unifica el acceso al frontend y al API, lo cual permite el uso de cookies de sesión seguras entre ambos componentes sin depender de configuraciones de origen cruzado (cross-domain). Desde el gateway, las solicitudes se comunican en servicios REST con el API desarrollado en .NET Core, alojado en Azure App Service, el cual gestiona el acceso a la base de datos Azure SQL Database.

Herramientas de Desarrollo

Para el desarrollo de la solución propuesta se emplean diferentes herramientas tecnológicas que permiten implementar, desplegar y mantener el sistema financiero-contable de TecniTech Solutions CR de forma eficiente. Estas herramientas abarcan el desarrollo de *software*, la persistencia de datos, la infraestructura de despliegue y los mecanismos de seguridad requeridos para la operación del sistema.

Persistencia de Datos. La base de datos del sistema se administra en Azure SQL Database, con la intención de centralizar la información financiera y mantener disponibilidad continua de los datos. Para el acceso y manipulación de los datos, se usa Entity Framework Core, facilitando la implementación de la capa de persistencia y promoviendo una adecuada separación entre la lógica de negocio y el acceso a datos. Durante el desarrollo también se usa SQL Server Management Studio para la administración, validación y ejecución de consultas sobre la base de datos.

Infraestructura y Despliegue. La infraestructura principal se despliega sobre servicios cloud de Microsoft Azure. El API se publica utilizando Azure App Service, mientras que el frontend se extiende mediante Azure Static Web Apps. Para centralizar el acceso a ambos componentes bajo un mismo dominio y fortalecer la seguridad de las comunicaciones, se incorpora Azure API Management como puerta de enlace (gateway) entre los usuarios y los servicios del backend. Esta capa adicional unifica el dominio público del sistema, aplica políticas de limitación de solicitudes (*rate limiting*) para mitigar ataques de fuerza bruta y restringir el acceso directo al servicio de backend, de manera que todo el tráfico deba pasar obligatoriamente por el gateway.

Este enfoque reduce la necesidad de infraestructura física dentro de la empresa y simplifica los procesos relacionados con publicación, mantenimiento y disponibilidad del sistema. Adicionalmente, la integración entre Azure y GitHub automatiza los procesos de despliegue y mantendrá el control sobre las versiones publicadas de la aplicación.

Entorno de Desarrollo y Control de Versiones. Para el desarrollo del proyecto, se usa Visual Studio Code como entorno principal de programación, debido a la flexibilidad que ofrece para trabajar tecnologías frontend y backend dentro de un mismo entorno de trabajo. Para el control de versiones, respaldo del código fuente y administración de cambios se utiliza GitHub, con el objetivo de mantener trazabilidad sobre la evolución del proyecto durante todo el proceso de desarrollo.

Seguridad. La autenticación del sistema se implementa mediante JWT, permitiendo validar sesiones y controlar el acceso a funcionalidades según el perfil asignado a cada usuario. La comunicación entre los diferentes componentes de la solución utiliza el protocolo HTTPS,

garantizando la confidencialidad e integridad de la información transmitida entre el frontend, el backend y la base de datos.

Conocimiento Requerido

El sistema financiero-contable está diseñado para usarse por medio de navegador web, por lo que no es necesario que los usuarios posean conocimientos técnicos avanzados en informática o administración de sistemas. De hecho, las personas que operen la plataforma deben poseer conocimientos básicos en el uso de computadoras, navegación web y manejo general de formularios digitales. Esto incluye tareas como iniciar sesión, registrar información, realizar consultas y visualizar reportes dentro del sistema.

Debido a la naturaleza financiera de la aplicación, también es necesario que los usuarios comprendan conceptos básicos relacionados con facturación, ingresos, gastos, cuentas por cobrar y cuentas por pagar. Lo anterior facilita la correcta interpretación de la información y reduce errores durante el registro de operaciones. En el caso de perfiles administrativos o encargados financieros, se requiere conocimiento básico en interpretación de reportes y control de información contable, ya que algunas funcionalidades están orientadas al análisis financiero y seguimiento presupuestario de la empresa. La interfaz del sistema mantendrá una navegación clara y organizada, de ahí que los usuarios pueden adaptarse al uso de la plataforma sin necesidad de procesos extensos de capacitación técnica.

Finalmente, cabe añadir que el sistema financiero-contable es frecuentado por distintos perfiles dentro de la empresa, de acuerdo con las funciones y responsabilidades asignadas a cada usuario. Por tanto, la distribución de accesos mantendrá control sobre la información y limita funcionalidades según el área de trabajo correspondiente. Entre los perfiles se encuentran:

Administrador del Sistema. El administrador tendrá acceso a las configuraciones generales de la plataforma, gestión de usuarios, asignación de permisos y mantenimiento de información crítica del sistema. Este perfil vigila el funcionamiento general de la aplicación y controla aspectos relacionados con seguridad y administración de accesos.

Encargado Financiero. El encargado financiero utiliza principalmente los módulos vinculados con cuentas contables, presupuestos, ingresos, gastos y generación de reportes financieros. Su función es supervisar la información económica de la empresa y validar que los registros financieros mantengan coherencia con las operaciones realizadas.

Asistente Administrativo. El asistente administrativo trabaja principalmente con procesos operativos como facturación, registro de pagos, consultas de información y actualización de datos generales dentro del sistema. Este perfil tendrá acceso limitado únicamente a las funcionalidades necesarias para ejecutar tareas administrativas diarias, reduciendo riesgos relacionados con modificaciones sobre información sensible.

Gerencia. La gerencia emplea el sistema principalmente para consulta de reportes financieros, seguimiento presupuestario y análisis general de la información registrada dentro de la plataforma. El acceso a información consolidada facilita la toma de decisiones y mejora el monitoreo del comportamiento financiero de la empresa.

Base de Datos

La base de datos del sistema se desarrolla utilizando Azure SQL Database como motor principal de almacenamiento. La información financiera, administrativa y contable de la empresa se centraliza en una estructura relacional, permitiendo mantener integridad entre los distintos procesos que componen la plataforma.

La organización de la base de datos se realiza mediante separación por esquemas, la cual distribuye las tablas según su funcionalidad dentro del sistema. Este enfoque facilita la administración de objetos, mejora el orden general de la estructura y ayuda a mantener una mejor separación lógica entre módulos.

Para el proyecto se hace uso de esquemas relacionados con seguridad, datos maestros, procesos contables y facturación. Esto mantendrá agrupadas las entidades de acuerdo con el área funcional a la que pertenecen y reduce el riesgo de inconsistencias entre procesos asociados.

Integridad de la Información. La estructura de la base de datos se diseña por medio de relaciones entre tablas mediante llaves primarias y foráneas, garantizando consistencia entre registros financieros y administrativos. Los módulos del sistema comparten información de manera integrada, por lo que la base de datos tiene un papel fundamental en la sincronización de procesos como facturación, ingresos, gastos, cuentas por cobrar y cuentas contables. Además de almacenar información, la base de datos mantiene trazabilidad sobre operaciones financieras realizadas dentro de la plataforma, facilitando consultas históricas y generación de reportes.

Relación entre los Módulos. La estructura relacional propicia que distintos módulos interactúen automáticamente entre sí. Por ejemplo, la generación de una factura produce registros relacionados en ingresos y cuentas por cobrar, mientras que el registro de gastos actualiza automáticamente movimientos contables y ejecución presupuestaria. Con base en lo anterior, este enfoque evita duplicidad de información y minimiza procesos manuales de conciliación entre módulos financieros.

Seguridad y Administración de Datos. La información almacenada dentro del sistema se protege a través de mecanismos de autenticación y control de acceso definidos desde la aplicación y la infraestructura cloud utilizada para el proyecto. El uso de Azure SQL Database permite aprovechar características administradas vinculadas con respaldo automático, monitoreo, recuperación y disponibilidad de la información. Adicionalmente, la separación entre frontend, backend y base de datos ayuda a reducir exposición directa de información sensible, manteniendo el acceso controlado únicamente mediante el API del sistema.

Escalabilidad y Mantenimiento. La estructura planteada incorpora nuevos módulos o funcionalidades sin necesidad de rediseñar completamente la base de datos. La separación lógica por esquemas y las relaciones normalizadas facilitan futuras ampliaciones y mantenimiento del sistema conforme evolucionen las necesidades de la empresa. Durante el desarrollo, también, se implementaron prácticas orientadas a mantener consistencia en nombres, relaciones y estructuras de tablas, a fin de propiciar la administración y comprensión general de la base de datos del proyecto.

Casos de Uso

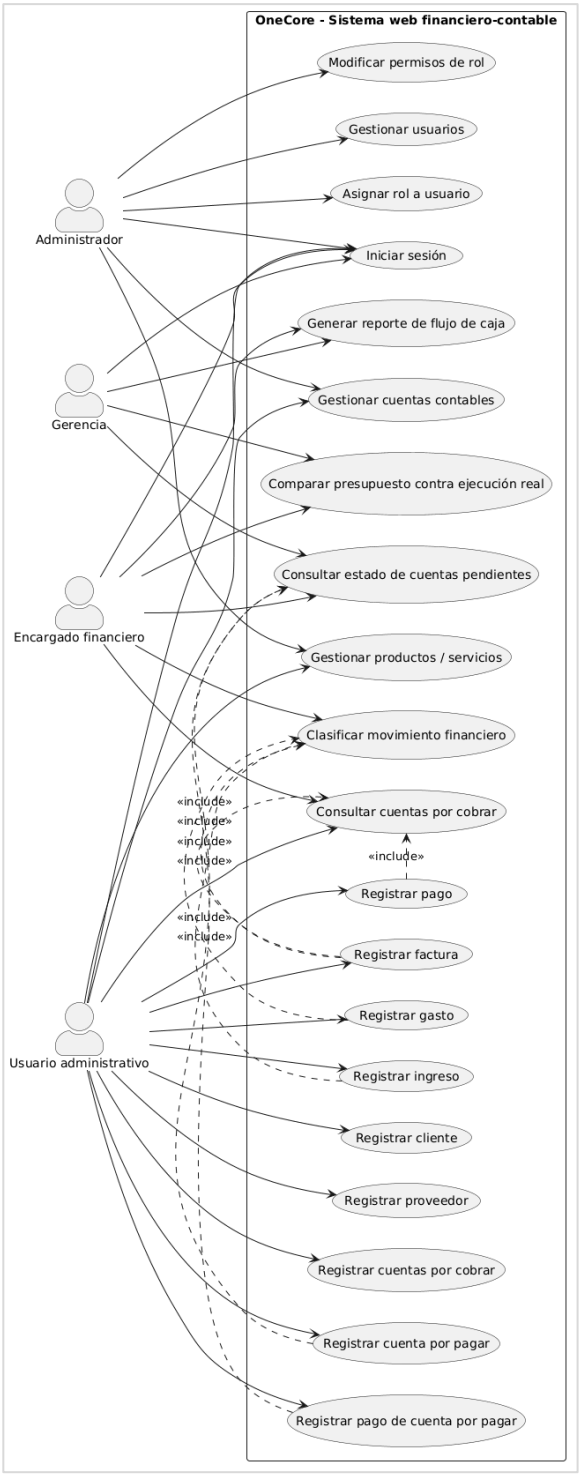
Los casos de uso representan las funcionalidades principales del sistema y la interacción que tienen los usuarios con cada uno de los procesos implementados dentro de la plataforma. A través de los siguientes casos se describen las acciones que realizan los distintos perfiles del sistema, así como el comportamiento esperado de cada módulo según las operaciones ejecutadas.

La definición de los casos de uso facilita la comprensión funcional del sistema, de manera que se identifican con más claridad los procesos involucrados en la gestión financiera y contable de la empresa. Además, sirven como base para el desarrollo, validación y organización de las funcionalidades que conforman la solución propuesta. Dicho esto, a continuación, se presentan los casos de uso correspondientes a los módulos definidos para el sistema financiero-contable de TecniTech Solutions CR, denominado OneCore.

Diagrama de Casos de Uso

Antes de detallar cada caso de uso, se presenta el diagrama general que ilustra las principales funcionalidades del sistema OneCore y la interacción de los distintos perfiles de usuario con cada módulo. Esta representación visualiza de forma general qué operaciones ejecuta cada actor dentro de la plataforma, considerando los permisos asociados a su rol y la relación entre procesos financieros como facturación, cuentas por cobrar, cuentas por pagar, gastos, contabilidad, presupuesto y reportes. A continuación, se detalla cada uno de los 20 casos de uso identificados.

Figura 3.
Diagrama de Casos de uso.



Fuente: Elaboración propia, 2026.

Tabla 10
CU-01 Iniciar sesión.

<table>
<tr><th colspan="2">Prototipo: OneCore - Sistema web para la gestión financiero-contable</th></tr>
<tr><td>Número caso de uso:</td><td>CU-01</td></tr>
<tr><td>Nombre del caso de uso:</td><td>Iniciar sesión</td></tr>
<tr><td>Fecha elaboración:</td><td>08 de mayo de 2026</td></tr>
<tr><td>Descripción caso de uso:</td><td>Este caso de uso permite que los usuarios autorizados ingresen al sistema web financiero-contable mediante el uso de credenciales válidas. El proceso de autenticación garantiza que únicamente usuarios registrados accedan a las funcionalidades del sistema según el rol asignado.</td></tr>
<tr><td>Autor caso de uso:</td><td>Adrián José Fajardo Pérez</td></tr>
<tr><td>Actores relacionados:</td><td>• Administrador
• Usuario administrativo</td></tr>
<tr><td>Precondiciones:</td><td>• El usuario debe encontrarse previamente registrado en el sistema.
• El usuario debe contar con credenciales válidas de acceso.
• El sistema debe encontrarse disponible y conectado a la base de datos.</td></tr>
<tr><th colspan="2">Flujo básico del caso de uso</th></tr>
<tr><td colspan="2">El usuario accede a la pantalla de inicio de sesión, ingresa sus credenciales y el sistema valida la información para permitir el acceso según el rol correspondiente.</td></tr>
<tr><td>1. El usuario accede a la pantalla de inicio de sesión del sistema.</td><td>2. El sistema muestra el formulario de autenticación.</td></tr>
<tr><td>3. El usuario ingresa su correo electrónico y contraseña.</td><td>4. El usuario selecciona la opción “Iniciar sesión”.</td></tr>
<tr><td>5. El sistema valida que los campos requeridos no se encuentren vacíos.</td><td>6. El sistema verifica las credenciales contra la información almacenada en la base de datos.</td></tr>
<tr><td>7. El sistema valida que el usuario se encuentre activo.</td><td>8. El sistema identifica el rol asociado al usuario autenticado.</td></tr>
<tr><td>9. El sistema registra la fecha y hora del acceso.</td><td>10. El sistema concede acceso y redirige al usuario al panel principal correspondiente.</td></tr>
<tr><th colspan="2">Subflujos</th></tr>
</table>

Subflujo S2 – Obtención de permisos del usuario	**1.** El sistema consulta el rol asociado al usuario autenticado. **2.** El sistema obtiene los permisos correspondientes al rol. **3.** El sistema habilita las funcionalidades autorizadas para el usuario.
Flujos alternos	
Flujo alterno A1 – Credenciales incorrectas	**1.** El sistema detecta que las credenciales ingresadas no coinciden con un usuario registrado. **2.** El sistema muestra un mensaje de autenticación inválida. **3.** El sistema registra el intento fallido de acceso. **4.** El usuario permanece en la pantalla de inicio de sesión.
Flujo alterno A2 – Usuario inactivo	**1.** El sistema identifica que el usuario se encuentra inactivo. **2.** El sistema deniega el acceso al usuario. **3.** El sistema muestra un mensaje indicando que el usuario no posee permisos de acceso activos.
Flujo alterno A3 – Fallo de conectividad con la base de datos	**1.** El sistema intenta validar las credenciales del usuario. **2.** Se produce un fallo de conexión con la base de datos. **3.** El sistema muestra un mensaje indicando que el servicio no se encuentra disponible temporalmente. **4.** El acceso al sistema es cancelado hasta restablecer la conexión.
Flujo alterno A4 – Validación de campos vacíos	**1.** El sistema detecta que existen campos obligatorios sin completar. **2.** El sistema muestra un mensaje indicando que los datos son requeridos. El sistema mantiene al usuario en la pantalla de autenticación.
Requerimientos especiales	
1. Las contraseñas deben almacenarse cifradas mediante mecanismos seguros de *hashing*. **2.** El sistema debe utilizar conexión segura HTTPS para el envío de credenciales. **3.** El sistema debe registrar la fecha y hora del último acceso del usuario. **4.** El tiempo de respuesta del proceso de autenticación no debe superar los 5 segundos en condiciones normales de operación. **5.** El sistema debe aplicar control de acceso basado en roles.	

Postcondiciones
1. El usuario autenticado accede al sistema según los permisos asociados a su rol. **2.** El sistema registra el acceso realizado para efectos de auditoría y trazabilidad. **3.** La sesión del usuario queda iniciada hasta el cierre de sesión o expiración de la sesión activa.

Fuente: Elaboración propia, 2026.

Tabla 11
CU-02 Gestionar usuarios.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-02
Nombre del caso de uso:	Gestionar usuarios
Fecha elaboración:	08 de mayo de 2026
Descripción caso de uso:	Este caso de uso administra los usuarios del sistema financiero-contable, incluyendo el registro, modificación, activación e inactivación de cuentas de acceso. El objetivo es garantizar un control adecuado de los permisos y accesos según el rol asignado a cada usuario.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador
Precondiciones:	• El usuario debe haber iniciado sesión previamente. • El usuario autenticado debe poseer permisos administrativos. • Los roles del sistema deben encontrarse previamente definidos.
Flujo básico del caso de uso	
El administrador accede al módulo de usuarios para registrar o administrar cuentas del sistema, asignando roles y estados según corresponda.	
1. El administrador accede al módulo de gestión de usuarios.	**2.** El sistema muestra el listado de usuarios registrados.
3. El administrador selecciona la opción de registrar nuevo usuario.	**4.** El sistema muestra el formulario de registro.
5. El administrador ingresa la información requerida del usuario.	**6.** El administrador asigna el rol correspondiente.
7. El sistema valida que los datos obligatorios se encuentren completos.	**8.** El sistema verifica que el correo electrónico no se encuentre registrado previamente.
9. El sistema almacena la información del nuevo usuario.	**10.** El sistema genera las credenciales de acceso correspondientes.

11. El sistema confirma el registro exitoso del usuario.	
Subflujos	
Subflujo S1 – Asignación de roles	**1.** El sistema consulta los roles disponibles. **2.** El administrador selecciona el rol correspondiente para el usuario. **3.** El sistema asigna los permisos asociados al rol seleccionado.
Subflujo S2 – Actualización de usuario	**1.** El sistema consulta el rol asociado al usuario autenticado. **2.** El administrador selecciona un usuario existente. **3.** El sistema muestra la información actual del usuario. **4.** El administrador modifica los datos requeridos. **5.** El sistema valida los cambios realizados. **6.** El sistema actualiza la información del usuario.
Flujos alternos	
Flujo alterno A1 – Correo electrónico duplicado	**1.** El sistema detecta que el correo ingresado ya pertenece a otro usuario registrado. **2.** El sistema muestra un mensaje indicando que el correo ya existe. **3.** El registro del usuario es cancelado hasta corregir la información.
Flujo alterno A2 – Datos obligatorios incompletos	**1.** El sistema detecta campos requeridos sin completar. **2.** El sistema muestra un mensaje indicando los datos faltantes. **3.** El administrador permanece en el formulario hasta completar la información requerida.
Flujo alterno A3 – Usuario inactivado	**1.** El administrador selecciona la opción de inactivar usuario. **2.** El sistema cambia el estado del usuario a inactivo. **3.** El sistema bloquea futuros accesos de dicho usuario al sistema.
Requerimientos especiales	
1. El sistema debe permitir control de acceso basado en roles. **2.** Las contraseñas deben almacenarse cifradas. **3.** El sistema debe registrar auditoría de creación y modificación de usuarios. **4.** El sistema debe impedir duplicidad de correos electrónicos.	

5. Solo usuarios con rol administrador pueden acceder a este módulo..
Postcondiciones
1. El usuario queda registrado, actualizado o inactivado correctamente. **2.** Los permisos asociados al rol quedan aplicados. **3.** El sistema registra la acción realizada para efectos de auditoría y trazabilidad.

Fuente: Elaboración propia, 2026.

Tabla 12
CU-03 Registrar cliente.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-03
Nombre del caso de uso:	Registrar cliente
Fecha elaboración:	08 de mayo de 2026
Descripción caso de uso:	Este caso de uso registra clientes dentro del sistema financiero-contable, almacenando la información necesaria para procesos de facturación, cuentas por cobrar y generación de reportes financieros.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario administrativo
Precondiciones:	• El usuario debe haber iniciado sesión. • El usuario debe contar con permisos para gestionar clientes. • El sistema debe encontrarse conectado a la base de datos.
Flujo básico del caso de uso	
El usuario accede al módulo de clientes e ingresa la información necesaria para registrar un nuevo cliente dentro del sistema.	
1. El usuario accede al módulo de clientes.	**2.** El sistema muestra el listado de clientes registrados.
3. El usuario selecciona la opción de registrar cliente.	**4.** El sistema muestra el formulario de registro.
5. El usuario ingresa la información general del cliente.	**6.** El usuario registra los datos de identificación y contacto.
7. El usuario selecciona las condiciones de crédito y moneda correspondiente.	**8.** El sistema valida los campos obligatorios.
9. El sistema verifica que el cliente no exista previamente.	**10.** El sistema almacena la información del cliente.
11. El sistema confirma el registro exitoso.	
Subflujos	

Subflujo S1 – Asignación de condiciones de crédito	**1.** El usuario define el límite de crédito permitido. **2.** El usuario establece la cantidad de días de pago. **3.** El sistema almacena las condiciones asociadas al cliente.
Subflujo S2 – Actualización de cliente	**1.** El usuario selecciona un cliente existente. **2.** El sistema muestra la información actual. **3.** El usuario modifica los datos requeridos. **4.** El sistema valida y actualiza la información.
Flujos alternos	
Flujo alterno A1 – Cliente ya registrado	**1.** El sistema detecta que el número de identificación ya existe. **2.** El sistema muestra un mensaje indicando duplicidad. **3.** El registro es cancelado hasta corregir la información.
Flujo alterno A2 – Información incompleta	**1.** El sistema detecta campos obligatorios vacíos. **2.** El sistema muestra los datos pendientes. **3.** El usuario permanece en el formulario hasta completar la información requerida.
Requerimientos especiales	
1. El sistema debe validar unicidad de identificación del cliente. **2.** El sistema debe permitir registro de clientes nacionales y extranjeros. **3.** El sistema debe registrar auditoría de creación y modificación. **4.** El sistema debe asociar moneda y condiciones de crédito al cliente.	
Postcondiciones	
1. El cliente queda registrado correctamente. **2.** La información permanece disponible para facturación y cuentas por cobrar. **3.** El sistema registra la acción realizada para efectos de auditoría.	

Fuente: Elaboración propia, 2026.

Tabla 13
CU-04 Registrar proveedor.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-04
Nombre del caso de uso:	Registrar proveedor
Fecha elaboración:	08 de mayo de 2026
Descripción caso de uso:	Este caso de uso registra proveedores dentro del sistema financiero-contable, almacenando la información necesaria para la gestión de cuentas por pagar, registro de gastos y control de obligaciones financieras.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario administrativo
Precondiciones:	• El usuario debe haber iniciado sesión. • El usuario debe contar con permisos para gestionar proveedores. • El sistema debe encontrarse disponible.
Flujo básico del caso de uso	
El usuario accede al módulo de proveedores e ingresa la información requerida para registrar un nuevo proveedor dentro del sistema.	
1. El usuario accede al módulo de proveedores.	**2.** El sistema muestra el listado de proveedores registrados.
3. El usuario selecciona la opción de registrar proveedor.	**4.** El sistema muestra el formulario correspondiente.
5. El usuario ingresa la información general del proveedor.	**6.** El usuario registra datos de identificación, contacto y condiciones de pago.
7. El sistema valida la información ingresada.	**8.** El sistema verifica que el proveedor no exista previamente.
9. El sistema almacena la información del proveedor.	**10.** El sistema confirma el registro exitoso.
Subflujos	
Subflujo S1 – Configuración de condiciones de pago	**1.** El usuario define la cantidad de días de pago del proveedor.

	2. El usuario registra condiciones adicionales de crédito si corresponde. **3.** El sistema almacena las condiciones asociadas al proveedor.
Subflujo S2 – Actualización de proveedor	**1.** El usuario selecciona un proveedor existente. **2.** El sistema muestra la información registrada. **3.** El usuario modifica los datos necesarios. **4.** El sistema valida y actualiza la información del proveedor.
Flujos alternos	
Flujo alterno A1 – Proveedor duplicado	**1.** El sistema detecta que el proveedor ya se encuentra registrado. **2.** El sistema muestra un mensaje indicando duplicidad de información. **3.** El registro se completa hasta el momento en que corrige los datos.
Flujo alterno A2 – Datos obligatorios incompletos	**4.** El sistema detecta campos requeridos sin completar. **5.** El sistema muestra un mensaje indicando los datos faltantes. **6.** El administrador permanece en el formulario hasta completar la información requerida.
Requerimientos especiales	
1. El sistema debe validar unicidad de identificación del proveedor. **2.** El sistema debe permitir asociar condiciones de pago. **3.** El sistema debe registrar auditoría de creación y modificación de proveedores. **4.** El sistema debe permitir activar o inactivar proveedores.	
Postcondiciones	
1. El proveedor queda registrado correctamente. **2.** La información permanece disponible para cuentas por pagar y registro de gastos. **3.** El sistema registra la acción realizada para efectos de auditoría y trazabilidad.	

Fuente: Elaboración propia, 2026.

Tabla 14
CU-05 Registrar factura.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-05
Nombre del caso de uso:	Registrar factura
Fecha elaboración:	08 de mayo de 2026
Descripción caso de uso:	Este caso de uso registra y genera facturas electrónicas dentro del sistema financiero-contable, asociando automáticamente la transacción con cuentas por cobrar, ingresos y registros contables correspondientes.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario administrativo
Precondiciones:	• El usuario debe haber iniciado sesión. • El cliente debe encontrarse registrado previamente. • Deben existir productos o servicios configurados en el sistema. • El usuario debe contar con permisos para gestionar facturación.
Flujo básico del caso de uso	
El usuario registra una factura electrónica asociada a un cliente, ingresando productos o servicios, montos e impuestos correspondientes. El sistema valida la información y genera los movimientos asociados.	
1. El usuario accede al módulo de facturación.	**2.** El sistema muestra el listado de facturas registradas.
3. El usuario selecciona la opción de registrar factura.	**4.** El sistema muestra el formulario de facturación.
5. El usuario selecciona el cliente correspondiente.	**6.** El usuario agrega los productos o servicios a facturar.
7. El sistema calcula automáticamente subtotal, impuestos y total general.	**8.** El usuario confirma la información registrada.
9. El sistema valida los datos ingresados.	**10.** El sistema genera el consecutivo de factura.

11. El sistema registra la factura en la base de datos.	**12.** El sistema genera automáticamente el registro en cuentas por cobrar.
13. El sistema genera el movimiento correspondiente en ingresos y contabilidad.	**14.** El sistema confirma el registro exitoso de la factura.
Subflujos	
Subflujo S1 – Cálculo automático de impuestos	**1.** El sistema identifica el porcentaje de impuesto asociado al producto o servicio. **2.** El sistema calcula automáticamente el monto de impuesto correspondiente. **3.** El sistema actualiza el total general de la factura.
Subflujo S2 – Generación de cuenta por cobrar	**1.** El sistema identifica las condiciones de crédito del cliente. **2.** El sistema genera automáticamente el saldo pendiente asociado a la factura. **3.** El sistema registra la fecha de vencimiento correspondiente.
Flujos alternos	
Flujo alterno A1 – Cliente no registrado	**1.** El sistema detecta que el cliente seleccionado no existe. **2.** El sistema muestra un mensaje indicando que el cliente debe registrarse previamente. **3.** El proceso de facturación es detenido hasta completar el registro del cliente.
Flujo alterno A2 – Error en validación de datos	**1.** El sistema detecta información incompleta o inconsistente. **2.** El sistema muestra un mensaje indicando los datos incorrectos. **3.** El usuario permanece en el formulario hasta corregir la información.
Flujo alterno A3 – Error en generación de factura	**1.** El sistema detecta un fallo durante el almacenamiento de la factura. **2.** El sistema cancela el proceso de registro. **3.** El sistema muestra un mensaje indicando que la factura no pudo ser generada.
Requerimientos especiales	
1. El sistema debe generar consecutivos automáticos de facturación. **2.** El sistema debe calcular automáticamente impuestos y montos totales. **3.** El sistema debe generar automáticamente cuentas por cobrar asociadas. **4.** El sistema debe mantener trazabilidad de la factura emitida. **5.** El sistema debe permitir futura integración con facturación electrónica ante la Administración Tributaria.	

Postcondiciones
1. La factura permanece registrada correctamente. **2.** La cuenta por cobrar asociada queda generada. **3.** Los movimientos contables correspondientes se almacenan. **4.** El sistema registra la operación para efectos de auditoría y trazabilidad.

Fuente: Elaboración propia, 2026.

Tabla 15
CU-06 Registrar pago.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-06
Nombre del caso de uso:	Registrar pago
Fecha elaboración:	08 de mayo de 2026
Descripción caso de uso:	Este caso de uso registra pagos realizados por clientes sobre facturas pendientes, actualizando automáticamente los saldos de cuentas por cobrar y los movimientos financieros correspondientes.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario administrativo
Precondiciones:	• El usuario debe haber iniciado sesión. • Deben existir facturas pendientes de cobro. • El usuario debe poseer permisos para registrar pagos.
Flujo básico del caso de uso	
El usuario registra un pago asociado a una factura pendiente. El sistema valida la información y actualiza automáticamente los saldos correspondientes.	
1. El usuario accede al módulo de cuentas por cobrar.	**2.** El sistema muestra las facturas pendientes de pago.
3. El usuario selecciona la factura correspondiente.	**4.** El sistema muestra el saldo pendiente asociado.
5. El usuario registra la información del pago realizado.	**6.** El usuario confirma el registro del pago.
7. El sistema valida el monto ingresado.	**8.** El sistema actualiza el saldo pendiente de la factura.
9. El sistema registra el movimiento financiero correspondiente.	**10.** El sistema confirma el registro exitoso del pago.
Subflujos	
Subflujo S1 – Registro de pago parcial	**1.** El usuario registra un monto menor al saldo pendiente. **2.** El sistema actualiza el saldo restante de la factura.

	3. El sistema mantiene la factura en estado pendiente.
Subflujo S2 – Cancelación total de factura	**1.** El usuario registra un monto equivalente al saldo pendiente. **2.** El sistema actualiza el saldo de la factura a cero. **3.** El sistema cambia el estado de la factura a cancelada.
Flujos alternos	
Flujo alterno A1 – Monto superior al saldo pendiente	**1.** El sistema detecta que el monto ingresado excede el saldo pendiente. **2.** El sistema muestra un mensaje de validación. **3.** El pago no puede registrarse hasta corregir el monto.
Flujo alterno A2 – Factura sin saldo pendiente	**1.** El sistema detecta que la factura ya se encuentra cancelada. **2.** El sistema muestra un mensaje indicando que no existen saldos pendientes. **3.** El proceso de registro es cancelado.
Requerimientos especiales	
1. El sistema debe permitir pagos parciales y totales. **2.** El sistema debe actualizar automáticamente cuentas por cobrar. **3.** El sistema debe registrar auditoría de pagos realizados. **4.** El sistema debe mantener historial de pagos asociados a cada factura.	
Postcondiciones	
1. El pago queda registrado correctamente. **2.** El saldo de la factura se actualiza. **3.** Los movimientos financieros asociados permanecen registrados. **4.** El sistema registra la operación para efectos de auditoría y trazabilidad.	

Fuente: Elaboración propia, 2026.

Tabla 16
CU-07 Registrar gasto.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-07
Nombre del caso de uso:	Registrar pago
Fecha elaboración:	08 de mayo de 2026
Descripción caso de uso:	Este caso de uso registra gastos operativos dentro del sistema financiero-contable, clasificándolos según su categoría y generando automáticamente los movimientos contables correspondientes.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario administrativo
Precondiciones:	• El usuario debe haber iniciado sesión. • Deben existir categorías de gastos configuradas. • El usuario debe poseer permisos para registrar gastos.
Flujo básico del caso de uso	
El usuario registra un gasto operativo indicando proveedor, categoría, monto y detalle correspondiente. El sistema valida la información y registra el movimiento financiero asociado.	
1. El usuario accede al módulo de gastos.	**2.** El sistema muestra el listado de gastos registrados.
3. El usuario selecciona la opción de registrar gasto.	**4.** El sistema muestra el formulario correspondiente.
5. El usuario registra la información general del gasto.	**6.** El usuario selecciona el proveedor y la categoría correspondiente.
7. El usuario ingresa el monto y detalle del gasto.	**8.** El sistema valida los datos ingresados.
9. El sistema registra el gasto en la base de datos.	**10.** El sistema genera automáticamente el movimiento contable asociado.
11. El sistema confirma el registro exitoso del gasto.	
Subflujos	

Subflujo S1 – Clasificación de gasto	**1.** El usuario selecciona la categoría correspondiente. **2.** El sistema asocia automáticamente el gasto a la clasificación financiera correspondiente. **3.** El sistema registra la categoría para efectos de reportes.
Subflujo S2 – Asociación de proveedor	**1.** El usuario selecciona un proveedor registrado. **2.** El sistema asocia el gasto al proveedor correspondiente. **3.** El sistema mantiene trazabilidad del gasto registrado.
Flujos alternos	
Flujo alterno A1 – Información incompleta	**1.** El sistema detecta campos obligatorios vacíos. **2.** El sistema muestra un mensaje indicando la información faltante. **3.** El usuario permanece en el formulario hasta completar los datos requeridos.
Flujo alterno A2 – Categoría no disponible	**1.** El sistema detecta que no existe una categoría válida seleccionada. **2.** El sistema muestra un mensaje indicando que debe seleccionarse una categoría. **3.** El registro del gasto no puede completarse hasta corregir la información.
Requerimientos especiales	
1. El sistema debe permitir clasificación de gastos por categoría. **2.** El sistema debe generar automáticamente movimientos contables asociados. **3.** El sistema debe mantener trazabilidad de gastos registrados. **4.** El sistema debe registrar auditoría de creación y modificación de gastos.	
Postcondiciones	
1. El gasto permanece registrado correctamente. **2.** El movimiento contable correspondiente queda generado. **3.** La información se mantiene disponible para reportes y control financiero. **4.** El sistema registra la operación para efectos de auditoría y trazabilidad.	

Fuente: Elaboración propia, 2026.

Tabla 17
CU-08 Consultar cuentas por cobrar.

<table>
<tr><th colspan="2">Prototipo: OneCore - Sistema web para la gestión financiero-contable</th></tr>
<tr><td>Número caso de uso:</td><td>CU-08</td></tr>
<tr><td>Nombre del caso de uso:</td><td>Consultar cuentas por cobrar</td></tr>
<tr><td>Fecha elaboración:</td><td>08 de mayo de 2026</td></tr>
<tr><td>Descripción caso de uso:</td><td>Este caso de uso permite consultar las cuentas por cobrar registradas en el sistema financiero-contable, mostrando información relacionada con facturas pendientes, saldos, fechas de vencimiento y estado de cobro. El objetivo es facilitar el control y seguimiento financiero de los clientes.</td></tr>
<tr><td>Autor caso de uso:</td><td>Adrián José Fajardo Pérez</td></tr>
<tr><td>Actores relacionados:</td><td>• Administrador
• Usuario administrativo</td></tr>
<tr><td>Precondiciones:</td><td>• El usuario debe haber iniciado sesión.
• Deben existir registros de facturas o cuentas por cobrar en el sistema.
• El usuario debe contar con permisos para consultar información financiera.</td></tr>
<tr><th colspan="2">Flujo básico del caso de uso</th></tr>
<tr><td colspan="2">El usuario accede al módulo de cuentas por cobrar para visualizar y consultar la información relacionada con saldos pendientes de clientes registrados.</td></tr>
<tr><td>1. El usuario accede al módulo de cuentas por cobrar.</td><td>2. El sistema muestra el listado de facturas pendientes registradas.</td></tr>
<tr><td>3. El usuario selecciona los criterios de búsqueda o filtrado deseados.</td><td>4. El sistema procesa la consulta solicitada.</td></tr>
<tr><td>5. El sistema muestra la información correspondiente, incluyendo cliente, número de factura, saldo pendiente, fecha de vencimiento y estado de cobro.</td><td>6. El usuario revisa la información mostrada.</td></tr>
<tr><td>7. El sistema permite consultar el detalle individual de una cuenta por cobrar.</td><td></td></tr>
<tr><th colspan="2">Subflujos</th></tr>
</table>

Subflujo S1 – Filtrado de información	**1.** El usuario selecciona criterios de búsqueda, tales como “cliente”, “estado” o “rango de fechas”. **2.** El sistema aplica los filtros correspondientes. **3.** El sistema actualiza el listado de resultados según los criterios seleccionados.
Subflujo S2 – Consulta de detalle	**1.** El usuario selecciona una cuenta por cobrar específica. **2.** El sistema muestra el detalle completo de la factura asociada. **3.** El sistema muestra el historial de pagos relacionados.
Flujos alternos	
Flujo alterno A1 – No existen resultados	**1.** El sistema detecta que no existen registros asociados a los criterios de búsqueda seleccionados. **2.** El sistema muestra un mensaje indicando que no se encontraron resultados. **3.** El usuario puede modificar los filtros de consulta.
Flujo alterno A2 – Error en consulta de información	**1.** El sistema detecta un fallo durante la consulta de información. **2.** El sistema muestra un mensaje indicando que no fue posible completar la operación. **3.** La consulta es cancelada temporalmente hasta restablecer el servicio.
Requerimientos especiales	
1. El sistema debe permitir búsqueda y filtrado de información. **2.** El sistema debe mostrar saldos actualizados en tiempo real. **3.** El sistema debe permitir consultar historial de pagos asociados. **4.** El sistema debe mantener trazabilidad de las operaciones financieras. **5.** Solo usuarios autorizados pueden consultar información financiera.	
Postcondiciones	
1. El usuario visualiza la información de cuentas por cobrar solicitada. **2.** La información consultada permanece disponible para seguimiento financiero. **3.** El sistema registra la consulta realizada para efectos de auditoría y trazabilidad.	

Fuente: Elaboración propia, 2026.

Tabla 18
CU-09 Registrar cuentas por cobrar.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-09
Nombre del caso de uso:	Registrar abono a cuenta por cobrar
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso registra abonos realizados por clientes sobre cuentas por cobrar previamente registradas en el sistema. Este proceso facilita el control de saldos pendientes, actualización automática del estado de la cuenta y mantenimiento del historial de pagos asociados a cada cliente.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario financiero
Precondiciones:	• El usuario debe haber iniciado sesión en el sistema. • El usuario debe contar con permisos correspondientes.
Flujo básico del caso de uso	
1. El usuario ingresa al módulo correspondiente.	**2.** El sistema muestra la información disponible y opciones de gestión.
3. El usuario selecciona la operación correspondiente.	**4.** El sistema solicita la información necesaria.
5. El usuario ingresa los datos requeridos.	**6.** El sistema valida la información ingresada.
7. El sistema procesa la solicitud realizada.	**8.** El sistema muestra un mensaje indicando que la operación fue realizada correctamente.
Subflujos	
Flujos alternos	
Flujo alterno A1 - Campos obligatorios incompletos	**1.** El sistema detecta campos requeridos vacíos. **2.** El sistema solicita completar la información faltante. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Flujo alterno A2 - Información inválida	**1.** El sistema detecta inconsistencias en la información ingresada. **2.** El sistema muestra un mensaje indicando el error. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.

Requerimientos especiales
1. El sistema debe registrar la fecha y el usuario responsable de cada operación. **2.** El sistema debe mantener trazabilidad de los registros realizados.
Postcondiciones
1. La operación queda almacenada en el sistema. **2.** La información permanece disponible para consultas y reportes.

Fuente: Elaboración propia, 2026.

Tabla 19

CU-10 Registrar pago de cuenta por pagar.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-10
Nombre del caso de uso:	Registrar pago de cuenta por pagar
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso registra pagos realizados a proveedores sobre cuentas por pagar almacenadas en el sistema. Este proceso facilita el control de obligaciones financieras, actualización de saldos pendientes y trazabilidad de pagos efectuados.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario financiero
Precondiciones:	• El usuario debe haber iniciado sesión en el sistema. • El usuario debe contar con permisos correspondientes.
Flujo básico del caso de uso	
1. El usuario ingresa al módulo correspondiente.	**2.** El sistema muestra la información disponible y opciones de gestión.
3. El usuario selecciona la operación correspondiente.	**4.** El sistema solicita la información necesaria.
5. El usuario ingresa los datos requeridos.	**6.** El sistema valida la información ingresada.
7. El sistema procesa la solicitud realizada.	**8.** El sistema muestra un mensaje indicando que la operación fue realizada correctamente.
Subflujos	
Flujos alternos	
Flujo alterno A1 - Campos obligatorios incompletos	**1.** El sistema detecta campos requeridos vacíos. **2.** El sistema solicita completar la información faltante. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Flujo alterno A2 - Información inválida	**1.** El sistema detecta inconsistencias en la información ingresada. **2.** El sistema muestra un mensaje indicando el error. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.

Requerimientos especiales
1. El sistema debe registrar la fecha y el usuario responsable de cada operación. **2.** El sistema debe mantener trazabilidad de los registros realizados.
Postcondiciones
1. La operación queda almacenada en el sistema. **2.** La información se mantiene disponible para consultas y reportes.

Fuente: Elaboración propia, 2026.

Tabla **20**

CU-11 Clasificar movimiento financiero.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-11
Nombre del caso de uso:	Clasificar movimiento financiero
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Permite asociar movimientos financieros registrados en el sistema con cuentas contables y categorías financieras específicas para garantizar una correcta organización y control de la información contable.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario financiero
Precondiciones:	• El usuario debe haber iniciado sesión en el sistema. • El usuario debe contar con permisos correspondientes.
Flujo básico del caso de uso	
1. El usuario ingresa al módulo correspondiente.	**2.** El sistema muestra la información disponible y opciones de gestión.
3. El usuario selecciona la operación correspondiente.	**4.** El sistema solicita la información necesaria.
5. El usuario ingresa los datos requeridos.	**6.** El sistema valida la información ingresada.
7. El sistema procesa la solicitud realizada.	**8.** El sistema muestra un mensaje indicando que la operación fue realizada correctamente.
Subflujos	
Flujos alternos	
Flujo alterno A1 - Campos obligatorios incompletos	**1.** El sistema detecta campos requeridos vacíos. **2.** El sistema solicita completar la información faltante. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Flujo alterno A2 - Información inválida	**1.** El sistema detecta inconsistencias en la información ingresada. **2.** El sistema muestra un mensaje indicando el error. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.

Requerimientos especiales
1. El sistema debe registrar la fecha y usuario responsable de cada operación. **2.** El sistema debe mantener trazabilidad de los registros realizados.
Postcondiciones
1. La operación queda almacenada en el sistema. **2.** La información permanece disponible para consultas y reportes.

Fuente: Elaboración propia, 2026.

Tabla 21

CU-12 Comparar presupuesto contra ejecución real.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-12
Nombre del caso de uso:	Comparar presupuesto contra ejecución real
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso compara montos presupuestados con movimientos financieros reales registrados en el sistema para facilitar el análisis financiero y control presupuestario.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario financiero
Precondiciones:	• El usuario debe haber iniciado sesión en el sistema. • El usuario debe contar con permisos correspondientes.
Flujo Básico del caso de uso	
1. El usuario ingresa al módulo correspondiente.	**2.** El sistema muestra la información disponible y opciones de gestión.
3. El usuario selecciona la operación correspondiente.	**4.** El sistema solicita la información necesaria.
5. El usuario ingresa los datos requeridos.	**6.** El sistema valida la información ingresada.
7. El sistema procesa la solicitud realizada.	**8.** El sistema muestra un mensaje indicando que la operación fue realizada correctamente.
Subflujos	
Flujos alternos	
Flujo alterno A1 - Campos obligatorios incompletos	**1.** El sistema detecta campos requeridos vacíos. **2.** El sistema solicita completar la información faltante. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Flujo alterno A2 - Información inválida	**1.** El sistema detecta inconsistencias en la información ingresada. **2.** El sistema muestra un mensaje indicando el error. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Requerimientos especiales	

1. El sistema debe registrar la fecha y el usuario responsable de cada operación. **2.** El sistema debe mantener trazabilidad de los registros realizados.
Postcondiciones
1. La operación queda almacenada en el sistema. **2.** La información se mantiene disponible para consultas y reportes.

Fuente: Elaboración propia, 2026.

Tabla 22
CU-13 Generar reporte de flujo de caja.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-13
Nombre del caso de uso:	Generar reporte de flujo de caja
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso genera reportes financieros relacionados con el flujo de caja de la empresa, utilizando información registrada de ingresos y egresos.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario financiero
Precondiciones:	• El usuario debe haber iniciado sesión en el sistema. • El usuario debe contar con permisos correspondientes.
Flujo básico del caso de uso	
1. El usuario ingresa al módulo correspondiente.	**2.** El sistema muestra la información disponible y opciones de gestión.
3. El usuario selecciona la operación correspondiente.	**4.** El sistema solicita la información necesaria.
5. El usuario ingresa los datos requeridos.	**6.** El sistema valida la información ingresada.
7. El sistema procesa la solicitud realizada.	**8.** El sistema muestra un mensaje indicando que la operación fue realizada correctamente.
Subflujos	
Flujos alternos	
Flujo alterno A1 - Campos obligatorios incompletos	**1.** El sistema detecta campos requeridos vacíos. **2.** El sistema solicita completar la información faltante. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Flujo alterno A2 - Información inválida	**1.** El sistema detecta inconsistencias en la información ingresada. **2.** El sistema muestra un mensaje indicando el error. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Requerimientos especiales	

1. El sistema debe registrar la fecha y el usuario responsable de cada operación. **2.** El sistema debe mantener trazabilidad de los registros realizados.
Postcondiciones
1. La operación queda almacenada en el sistema. **2.** La información se mantiene disponible para consultas y reportes.

Fuente: Elaboración propia, 2026.

Tabla 23

CU-14 Consultar estado de cuentas pendientes.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-14
Nombre del caso de uso:	Consultar estado de cuentas pendientes
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso permite consultar cuentas por cobrar y cuentas por pagar pendientes, vencidas o canceladas, con el objetivo de facilitar el seguimiento financiero de la empresa.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario financiero
Precondiciones:	• El usuario debe haber iniciado sesión en el sistema. • El usuario debe contar con permisos correspondientes.
Flujo básico del caso de uso	
1. El usuario ingresa al módulo correspondiente.	**2.** El sistema muestra la información disponible y opciones de gestión.
3. El usuario selecciona la operación correspondiente.	**4.** El sistema solicita la información necesaria.
5. El usuario ingresa los datos requeridos.	**6.** El sistema valida la información ingresada.
7. El sistema procesa la solicitud realizada.	**8.** El sistema muestra un mensaje indicando que la operación fue realizada correctamente.
Subflujos	
Flujos alternos	
Flujo alterno A1 - Campos obligatorios incompletos	**1.** El sistema detecta campos requeridos vacíos. **2.** El sistema solicita completar la información faltante. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Flujo alterno A2 - Información inválida	**1.** El sistema detecta inconsistencias en la información ingresada. **2.** El sistema muestra un mensaje indicando el error. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Requerimientos especiales	

1. El sistema debe registrar la fecha y el usuario responsable de cada operación. **2.** El sistema debe mantener trazabilidad de los registros realizados.
Postcondiciones
1. La operación queda almacenada en el sistema. **2.** La información permanece disponible para consultas y reportes.

Fuente: Elaboración propia, 2026.

Tabla 24
CU-15 Asignar rol a usuario.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-15
Nombre del caso de uso:	Asignar rol a usuario
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso asigna roles del sistema a usuarios registrados con el objetivo de controlar el acceso a funcionalidades específicas del sistema.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario financiero
Precondiciones:	• El usuario debe haber iniciado sesión en el sistema. • El usuario debe contar con permisos correspondientes.
Flujo básico del caso de uso	
1. El usuario ingresa al módulo correspondiente.	**2.** El sistema muestra la información disponible y opciones de gestión.
3. El usuario selecciona la operación correspondiente.	**4.** El sistema solicita la información necesaria.
5. El usuario ingresa los datos requeridos.	**6.** El sistema valida la información ingresada.
7. El sistema procesa la solicitud realizada.	**8.** El sistema muestra un mensaje indicando que la operación fue realizada correctamente.
Subflujos	
Flujos alternos	
Flujo alterno A1 - Campos obligatorios incompletos	**1.** El sistema detecta campos requeridos vacíos. **2.** El sistema solicita completar la información faltante. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Flujo alterno A2 - Información inválida	**1.** El sistema detecta inconsistencias en la información ingresada. **2.** El sistema muestra un mensaje indicando el error. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Requerimientos especiales	

1. El sistema debe registrar la fecha y el usuario responsable de cada operación. **2.** El sistema debe mantener trazabilidad de los registros realizados.
Postcondiciones
1. La operación queda almacenada en el sistema. **2.** La información permanece disponible para consultas y reportes.

Fuente: Elaboración propia, 2026.

Tabla 25
CU-16 Modificar permisos de rol.

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-16
Nombre del caso de uso:	Modificar permisos de rol
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso permite administrar y modificar los permisos asociados a los roles definidos en el sistema para controlar el acceso a módulos y acciones disponibles.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario financiero
Precondiciones:	• El usuario debe haber iniciado sesión en el sistema. • El usuario debe contar con permisos correspondientes.
Flujo básico del caso de uso	
1. El usuario ingresa al módulo correspondiente.	**2.** El sistema muestra la información disponible y opciones de gestión.
3. El usuario selecciona la operación correspondiente.	**4.** El sistema solicita la información necesaria.
5. El usuario ingresa los datos requeridos.	**6.** El sistema valida la información ingresada.
7. El sistema procesa la solicitud realizada.	**8.** El sistema muestra un mensaje indicando que la operación fue realizada correctamente.
Subflujos	
Flujos alternos	
Flujo alterno A1 - Campos obligatorios incompletos	**1.** El sistema detecta campos requeridos vacíos. **2.** El sistema solicita completar la información faltante. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Flujo alterno A2 - Información inválida	**1.** El sistema detecta inconsistencias en la información ingresada. **2.** El sistema muestra un mensaje indicando el error. **3.** El usuario corrige la información. **4.** El sistema continúa con el proceso.
Requerimientos especiales	

1. El sistema debe registrar la fecha y el usuario responsable de cada operación. **2.** El sistema debe mantener trazabilidad de los registros realizados.
Postcondiciones
1. La operación queda almacenada en el sistema. **2.** La información queda disponible para consultas y reportes.

Fuente: Elaboración propia, 2026.

Tabla 26
CU-17 Registrar cuenta por pagar

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-17
Nombre del caso de uso:	Registrar abono a cuenta por pagar
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso permite registrar una cuenta por pagar a un proveedor dentro del sistema financiero-contable, especificando el proveedor, el monto, la fecha de vencimiento y el detalle correspondiente, previo al registro de los pagos asociados a dicha cuenta.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario administrativo
Precondiciones:	• El usuario debe haber iniciado sesión. • Debe existir el proveedor registrado en el sistema. • El usuario debe poseer permisos para registrar cuentas por pagar.
Flujo básico del caso de uso	
1. El usuario accede al módulo de cuentas por pagar.	**2.** El sistema muestra el listado de cuentas por pagar registradas.
3. El usuario selecciona la opción de registrar cuenta por pagar.	**4.** El sistema muestra el formulario correspondiente.
5. El usuario selecciona el proveedor asociado.	**6.** El usuario ingresa el monto y la fecha de vencimiento.
7. El usuario registra el detalle de la cuenta por pagar.	**8.** El sistema valida la información ingresada.
9. El sistema registra la cuenta por pagar en la base de datos.	**10.** El sistema confirma el registro exitoso de la cuenta por pagar.
Subflujos	
Subflujo S1 – Asociación de proveedor	**1.** El usuario selecciona un proveedor registrado. **2.** El sistema asocia la cuenta por pagar al proveedor correspondiente. **3.** El sistema mantiene trazabilidad de la cuenta registrada.
Subflujo S2 – Cálculo de vencimiento	**1.** El usuario selecciona un proveedor registrado. **2.** El sistema asocia la cuenta por pagar al proveedor correspondiente. **3.** El sistema mantiene trazabilidad de la cuenta registrada.

Flujos alternos	
Flujo alterno A1 - Información incompleta	**1.** El sistema detecta campos obligatorios vacíos. **2.** El sistema muestra un mensaje indicando la información faltante. **3.** El usuario permanece en el formulario hasta completar los datos requeridos.
Flujo alterno A2 - Proveedor no registrado	**1.** El sistema detecta que el proveedor seleccionado no existe en el sistema. **2.** El sistema muestra un mensaje indicando que debe registrarse el proveedor previamente. **3.** El registro de la cuenta por pagar no puede completarse hasta corregir la información.
Requerimientos especiales	
1. El sistema debe calcular automáticamente el estado de vencimiento de la cuenta. **2.** El sistema debe mantener trazabilidad de las cuentas por pagar registradas. **3.** El sistema debe registrar auditoría de creación y modificación de cuentas por pagar.	
Postcondiciones	
1. La cuenta por pagar permanece registrada correctamente. **2.** La cuenta queda disponible para su posterior pago (CU-10). **3.** La información se mantiene disponible para reportes y control financiero. **4.** El sistema registra la operación para efectos de auditoría y trazabilidad.	

Fuente: Elaboración propia, 2026.

Tabla 27
CU-18 Registrar ingreso

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-18
Nombre del caso de uso:	Registrar ingreso
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso registra ingresos operativos dentro del sistema financiero-contable, clasificándolos según su categoría y generando automáticamente los movimientos contables correspondientes.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario administrativo
Precondiciones:	• El usuario debe haber iniciado sesión • Deben existir categorías de ingreso configuradas. • El usuario debe poseer permisos para registrar ingresos.
Flujo básico del caso de uso	
1. El usuario accede al módulo de ingresos.	**2.** El sistema muestra el listado de ingresos registrados.
3. El usuario selecciona la opción de registrar ingreso.	**4.** El sistema muestra el formulario correspondiente.
5. El usuario registra la información general del ingreso.	**6.** El usuario selecciona la categoría correspondiente.
7. El usuario ingresa el monto y detalle del ingreso.	**8.** El sistema valida los datos ingresados.
9. El sistema registra el ingreso en la base de datos.	**10.** El sistema genera automáticamente el movimiento contable asociado.
11. El sistema confirma el registro exitoso del ingreso	
Subflujos	
Subflujo S1 – Clasificación de ingreso	**1.** El usuario selecciona la categoría correspondiente. **2.** El sistema asocia automáticamente el ingreso a la clasificación financiera correspondiente. **3.** El sistema registra la categoría para efectos de reportes.
Flujos alternos	
Flujo alterno A1 - Información incompleta	**1.** El sistema detecta campos obligatorios vacíos. **2.** El sistema muestra un mensaje indicando la información faltante.

	3. El usuario permanece en el formulario hasta completar los datos requeridos.
Flujo alterno A2 - Categoría no disponible	**1.** El sistema detecta que no existe una categoría válida seleccionada. **2.** El sistema muestra un mensaje indicando que debe seleccionarse una categoría. **3.** El registro del ingreso no puede completarse hasta corregir la información.
Requerimientos especiales	
1. El sistema debe permitir clasificación de ingresos por categoría. **2.** El sistema debe generar automáticamente movimientos contables asociados. **3.** El sistema debe mantener trazabilidad de ingresos registrados. **4.** El sistema debe registrar auditoría de creación y modificación de ingresos.	
Postcondiciones	
1. El ingreso permanece registrado correctamente. **2.** El movimiento contable correspondiente queda generado. **3.** La información se mantiene disponible para reportes y control financiero. **4.** El sistema registra la operación para efectos de auditoría y trazabilidad.	

Fuente: Elaboración propia, 2026.

Tabla 28.
CU-19 Gestionar cuentas contables

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-19
Nombre del caso de uso:	Gestionar cuentas contables
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso administra el catálogo de cuentas contables del sistema financiero-contable, incluyendo el registro, modificación y desactivación de cuentas, con el fin de mantener actualizado el plan de cuentas utilizado en el registro de movimientos financieros.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario administrativo
Precondiciones:	• El usuario debe haber iniciado sesión. • El usuario debe poseer permisos para gestionar cuentas contables. • Debe existir una clasificación de tipos de cuenta previamente definida (activo, pasivo, patrimonio, ingreso, gasto).
Flujo básico del caso de uso	
1. El usuario accede al módulo de cuentas contables.	**2.** El sistema muestra el listado de cuentas contables registradas.
3. El usuario selecciona la opción de registrar nueva cuenta.	**4.** El sistema muestra el formulario correspondiente.
5. El usuario ingresa el código y nombre de la cuenta.	**6.** El usuario selecciona el tipo de cuenta correspondiente.
7. El sistema valida que el código no se encuentre duplicado.	**8.** El sistema almacena la información de la nueva cuenta.
9. El sistema confirma el registro exitoso de la cuenta contable.	
Subflujos	
Subflujo S1 – Modificación de cuenta contable	**1.** El usuario selecciona una cuenta contable existente. **2.** El sistema muestra la información actual de la cuenta. **3.** El usuario modifica los datos requeridos. **4.** El sistema actualiza la información de la cuenta contable.
Subflujo S2 – Desactivación de cuenta contable	**1.** El usuario selecciona la opción de desactivar cuenta.

	2. El sistema cambia el estado de la cuenta a inactiva. **3.** El sistema impide el uso de la cuenta en nuevos registros contables, manteniendo el historial de movimientos previos.
Flujos alternos	
Flujo alterno A1 - Código de cuenta duplicado	**1.** El sistema detecta que el código ingresado ya pertenece a otra cuenta registrada. **2.** El sistema muestra un mensaje indicando que el código ya existe. **3.** El registro de la cuenta es cancelado hasta corregir la información.
Flujo alterno A2 - Datos obligatorios incompletos	**1.** El sistema detecta campos requeridos sin completar. **2.** El sistema muestra un mensaje indicando los datos faltantes. **3.** El usuario permanece en el formulario hasta completar la información requerida.
Requerimientos especiales	
1. El sistema debe impedir la duplicidad de códigos de cuenta contable. **2.** El sistema debe mantener el historial de movimientos asociados a cuentas desactivadas. **3.** El sistema debe registrar auditoría de creación, modificación y desactivación de cuentas contables. **4.** Solo usuarios con permisos administrativos pueden desactivar cuentas contables.	
Postcondiciones	
1. La cuenta contable queda registrada, modificada o desactivada correctamente. **2.** El plan de cuentas se mantiene actualizado para su uso en el registro de movimientos financieros. **3.** El sistema registra la operación para efectos de auditoría y trazabilidad.	

Fuente: Elaboración propia, 2026.

Tabla 29.
CU-20 Gestionar productos/servicios

Prototipo: OneCore - Sistema web para la gestión financiero-contable	
Número caso de uso:	CU-20
Nombre del caso de uso:	Gestionar productos/servicios
Fecha elaboración:	24 de mayo de 2026
Descripción caso de uso:	Este caso de uso administra el catálogo de productos y servicios ofrecidos por la empresa dentro del sistema financiero-contable, incluyendo el registro, modificación y desactivación de productos o servicios, con el fin de mantener actualizada la información utilizada en el proceso de facturación.
Autor caso de uso:	Adrián José Fajardo Pérez
Actores relacionados:	• Administrador • Usuario administrativo
Precondiciones:	• El usuario debe haber iniciado sesión. • El usuario debe poseer permisos para gestionar el catálogo de productos y servicios.
Flujo básico del caso de uso	
1. El usuario accede al módulo de productos y servicios.	**2.** El sistema muestra el listado de productos y servicios registrados.
3. El usuario selecciona la opción de registrar nuevo producto/servicio.	**4.** El sistema muestra el formulario correspondiente.
5. El usuario ingresa el código, nombre y descripción.	**6.** El usuario selecciona el tipo (producto o servicio) y define el precio.
7. El sistema valida que el código no se encuentre duplicado.	**8.** El sistema almacena la información del nuevo producto/servicio.
9. El sistema confirma el registro exitoso.	
Subflujos	
Subflujo S1 – Modificación de cuenta contable	**1.** El usuario selecciona un producto o servicio existente. **2.** El sistema muestra la información actual del registro. **3.** El usuario modifica los datos requeridos. **4.** El sistema actualiza la información del producto/servicio.
Subflujo S2 – Desactivación de producto/servicio	**1.** El usuario selecciona la opción de desactivar producto/servicio. **2.** El sistema cambia el estado del registro a inactivo.

	3. El sistema impide su selección en nuevas facturas, manteniendo el historial de facturación previo.
Flujos alternos	
Flujo alterno A1 - Código duplicado	**1.** El sistema detecta que el código ingresado ya pertenece a otro producto/servicio registrado. **2.** El sistema muestra un mensaje indicando que el código ya existe. **3.** El registro es cancelado hasta corregir la información.
Flujo alterno A2 - Datos obligatorios incompletos	**1.** El sistema detecta campos requeridos sin completar. **2.** El sistema muestra un mensaje indicando los datos faltantes. **3.** El usuario permanece en el formulario hasta completar la información requerida.
Requerimientos especiales	
1. El sistema debe impedir la duplicidad de códigos de producto/servicio. **2.** El sistema debe mantener el historial de facturación asociado a productos/servicios desactivados. **3.** El sistema debe registrar auditoría de creación, modificación y desactivación de productos/servicios. **4.** El catálogo debe estar disponible para su selección durante el registro de facturas (CU-05).	
Postcondiciones	
1. El producto o servicio queda registrado, modificado o desactivado correctamente. **2.** El catálogo se mantiene actualizado para su uso en el proceso de facturación. **3.** El sistema registra la operación para efectos de auditoría y trazabilidad.	

Fuente: Elaboración propia, 2026.

Diseño

En esta sección se ilustran los elementos que conforman el diseño del sistema, mediante figuras. También, en este apartado, se muestra el diseño de la arquitectura, la base de datos, las interfaces, entre otros componentes implementados, con el objetivo de que el funcionamiento del prototipo se ejecute de manera óptima, para satisfacer los requisitos identificados durante la investigación.

Arquitectura del Sistema

La arquitectura del sistema describe la forma en que los diferentes componentes tecnológicos interactúan dentro de la solución propuesta. Esta arquitectura contempla los

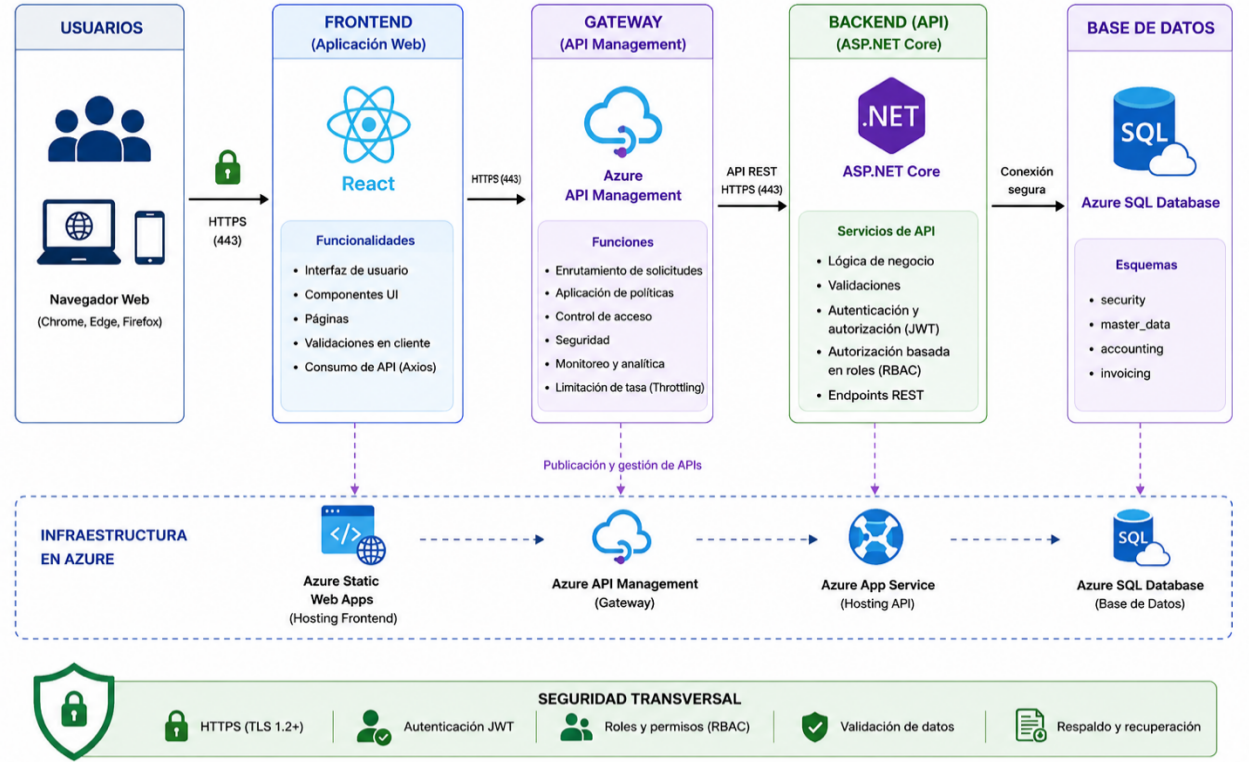
elementos necesarios para el acceso de los usuarios, el procesamiento de la información y el almacenamiento de los datos financieros y administrativos de la empresa. Los usuarios acceden al sistema gracias a un navegador web, utilizando una conexión a internet. La interfaz de usuario es desarrollada mediante React y desplegada empleando Azure Static Web Apps, lo que permite una experiencia de acceso centralizada y accesible desde diferentes ubicaciones.

Las solicitudes realizadas por los usuarios se dirigen a través de Azure API Management, que actúa como puerta de enlace única del sistema, para luego ser procesadas por un API REST desarrollado en ASP.NET Core y publicado por medio de Azure App Service. Esta capa intermedia unifica el dominio público de acceso al sistema, maneja el volumen de solicitudes mediante de políticas de limitación de tasa, y evita que el servicio de backend sea accesible de forma directa desde fuera de la infraestructura de Azure. El API es responsable de gestionar la lógica de negocio, aplicar validaciones, controlar el acceso a las funcionalidades del sistema y coordinar la comunicación con la base de datos.

La información se almacena en Azure SQL Database, donde se mantendrán los registros relacionados con clientes, proveedores, facturación, cuentas por cobrar, cuentas por pagar, presupuestos, ingresos, egresos y demás procesos financieros contemplados dentro del alcance del proyecto. La comunicación entre los componentes se realiza con protocolos seguros utilizando HTTPS, con la intención de proteger la información durante su transmisión.

La siguiente figura muestra la arquitectura general propuesta para el sistema financiero-contable. En ella se observa la interacción entre los usuarios, la aplicación web desarrollada en React, la puerta de enlace implementada mediante Azure API Management, el API desarrollado en ASP.NET Core y la base de datos Azure SQL Database. Esta distribución separa adecuadamente las responsabilidades de cada componente, de modo que facilita el mantenimiento, la escalabilidad y la administración de la solución. Asimismo, el uso de servicios en la nube de Microsoft Azure, junto con la centralización del acceso mediante el gateway, contribuye a garantizar disponibilidad, seguridad y acceso controlado a la información desde diferentes ubicaciones.

Figura 4.
Arquitectura general del sistema.



Fuente: Elaboración propia, 2026.

Arquitectura de Software

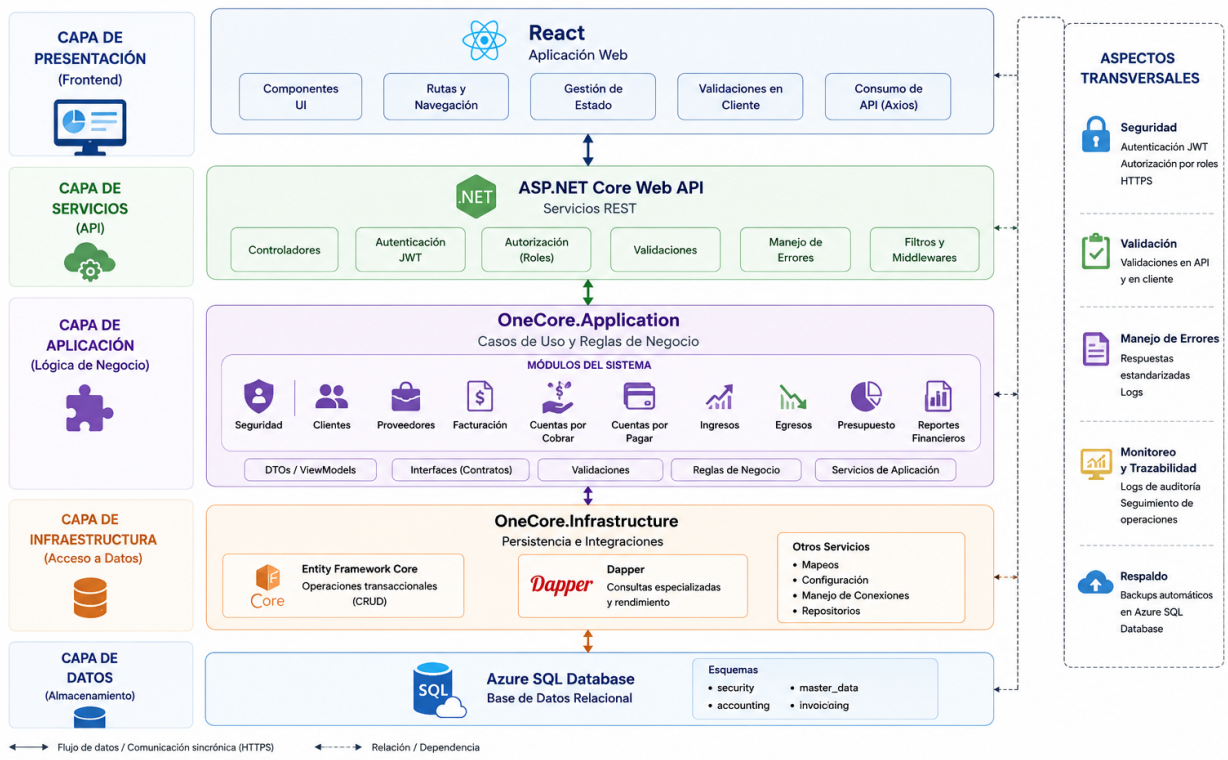
La arquitectura de *software* propuesta para el sistema se basa en una estructura por capas, la cual separa las responsabilidades de cada componente y facilita el mantenimiento, la escalabilidad y la evolución futura de la solución. El acceso externo a esta arquitectura por capas se realiza a través de Azure API Management, que actúa como punto de entrada único hacia el API, sin alterar la organización interna de las capas de presentación, aplicación e infraestructura descritas a continuación.

Sobre la capa de presentación, está conformada por una aplicación web desarrollada en React. Esta capa gestiona la interacción con los usuarios, presenta la información y hace las solicitudes correspondientes a los servicios expuestos por el backend. Por otro lado, la capa de servicios se implementa mediante ASP.NET Core Web API, actuando como punto central de acceso a la lógica de negocio del sistema. Esta capa gestiona las solicitudes provenientes del frontend y coordina la ejecución de los diferentes procesos requeridos por cada módulo funcional.

Dentro de la solución, se integra una capa de aplicación encargada de definir las reglas de negocio, validaciones, contratos y casos de uso necesarios para cada uno de los módulos que conforman el sistema. Entre estos módulos se encuentran Seguridad, Clientes, Proveedores, Facturación, Cuentas por Cobrar, Cuentas por Pagar, Ingresos, Egresos, Presupuestos y Reportes Financieros.

La capa de infraestructura es responsable de la persistencia de los datos y de la comunicación con los recursos externos. Para ello, se utilizan Entity Framework Core y Dapper como mecanismos de acceso a datos, con el fin de combinar productividad en las operaciones transaccionales y eficiencia en las consultas especializadas. Finalmente, la información se almacena en Azure SQL Database, garantizando integridad, disponibilidad y administración centralizada de los datos requeridos para la operación del sistema.

Figura 5.
Arquitectura de Software.



Fuente: Elaboración propia, 2026.

La Figura 5 presenta la arquitectura de software propuesta para el sistema. Esta se organiza por medio de una estructura de capas que favorece la separación de responsabilidades y la

mantenibilidad de la solución. La capa de presentación gestiona la interacción con los usuarios, mientras que el API centraliza el acceso a los servicios del sistema. La lógica de negocio se concentra en la capa de aplicación, donde se implementan los diferentes módulos funcionales, y la capa de infraestructura se encarga de la persistencia de datos y la integración con recursos externos. Esta organización desarrolla un sistema más escalable, reutilizable y alineado con buenas prácticas de arquitectura de *software* empresarial.

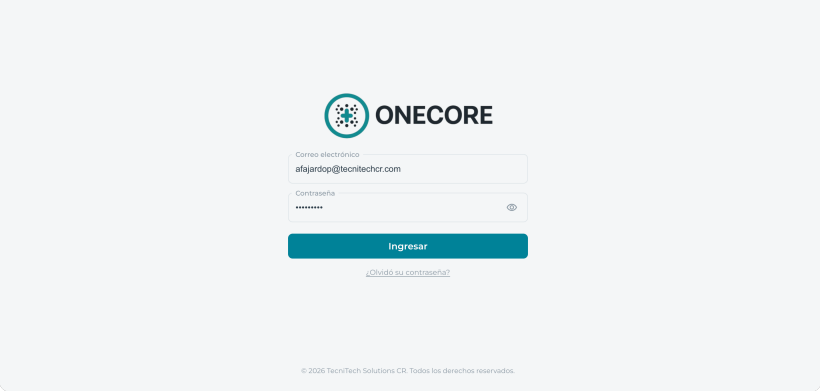
Diseño de entradas

El diseño de entradas corresponde a las interfaces mediante las cuales los usuarios interactúan con el sistema para registrar, consultar y administrar la información requerida por cada uno de los módulos funcionales. Estas pantallas fueron diseñadas con un enfoque orientado a la usabilidad, procurando una navegación intuitiva, una distribución clara de los elementos y una experiencia de usuario que facilite la ejecución de las tareas diarias.

A continuación, se presentan las principales interfaces desarrolladas para el sistema OneCore, las cuales gestionan procesos relacionados con clientes, proveedores, facturación, gastos, cuentas por cobrar, cuentas por pagar, presupuestos, catálogo de cuentas contables y administración de usuarios.

La pantalla de inicio de sesión, mostrada en la Figura 6, constituyó el punto de entrada al sistema. El usuario ingresó su correo electrónico y contraseña para autenticarse, con la opción de mostrar u ocultar la contraseña y de recuperar el acceso en caso de olvido.

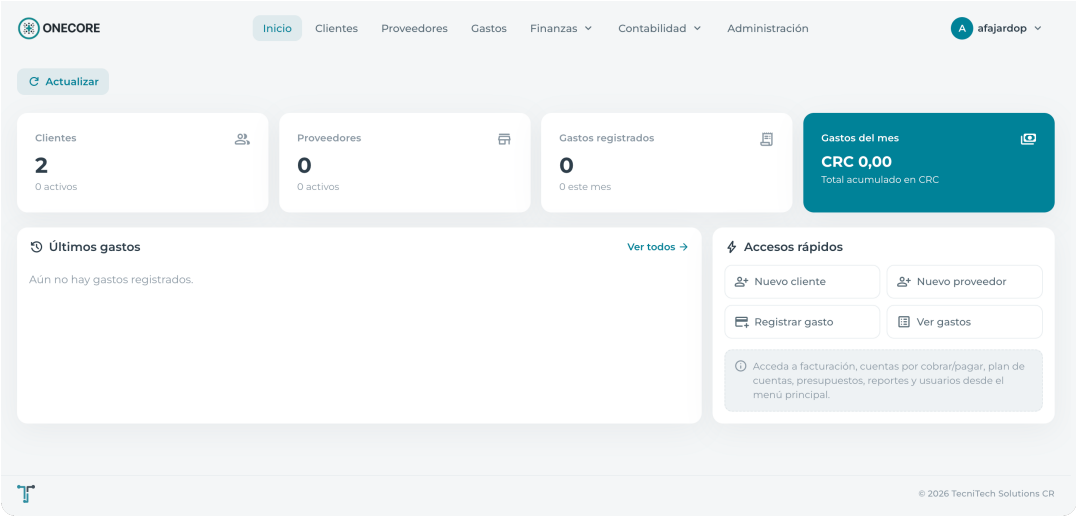
Figura 6.
Pantalla de Inicio de sesión.



Fuente: Elaboración propia, 2026.

Una vez autenticado, el sistema presentó el dashboard principal, ilustrado en la Figura 7, el cual resumió los indicadores clave de la operación (clientes, proveedores, gastos registrados y gastos del mes), junto con un listado de los últimos gastos y un panel de accesos rápidos hacia las funciones más utilizadas del sistema.

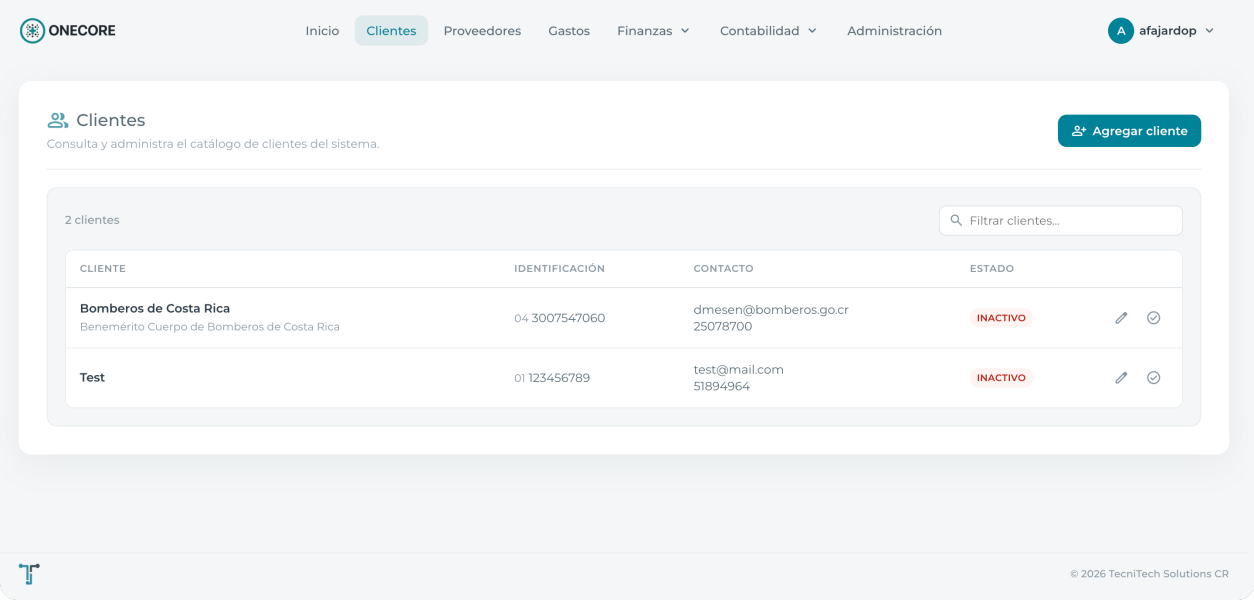
Figura 7.
Pantalla de Dashboard principal.



Fuente: Elaboración propia, 2026.

La Figura 8 mostró la pantalla de gestión de clientes, donde el usuario consultó el catálogo de clientes registrados, filtró la información mediante un buscador y accedió a las opciones de agregar, editar o cambiar el estado de un cliente.

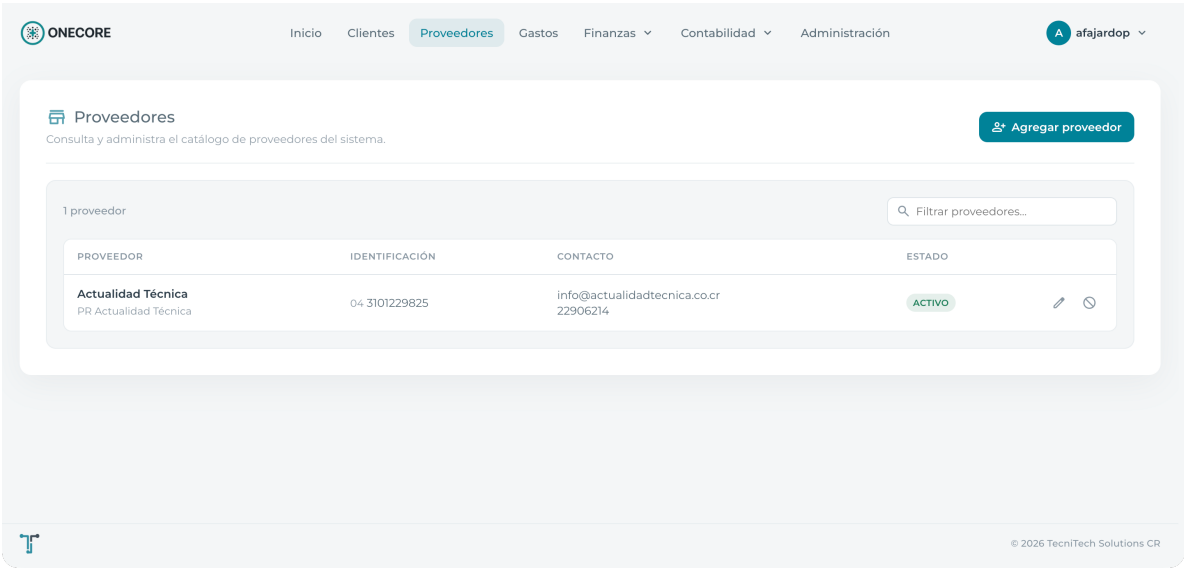
Figura 8.
Pantalla de Gestión de clientes.



Fuente: Elaboración propia, 2026.

De forma equivalente, la Figura 9 presentó la pantalla de gestión de proveedores, que replicó la misma estructura de listado, filtrado y administración utilizada en clientes, adaptada al catálogo de proveedores de la empresa.

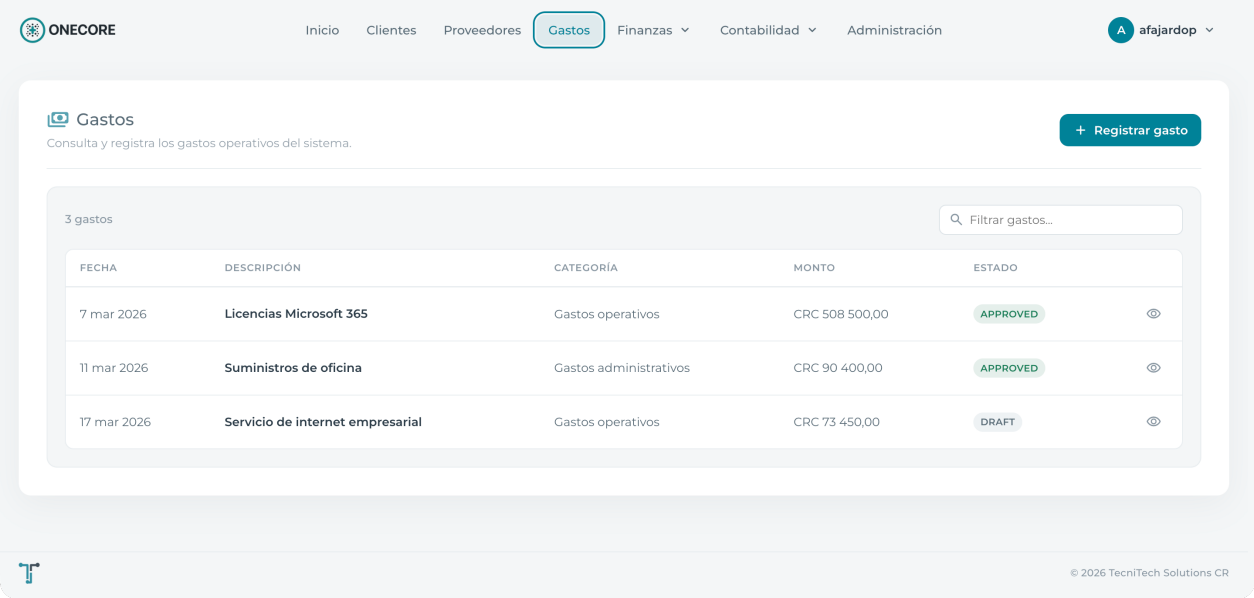
Figura 9.
Pantalla de Gestión de proveedores.



Fuente: Elaboración propia, 2026.

En la Figura 10 se observó la pantalla de gestión de gastos, en la cual el usuario consultó los gastos operativos registrados, con su fecha, descripción, categoría, monto y estado de aprobación, además de la opción de registrar un nuevo gasto.

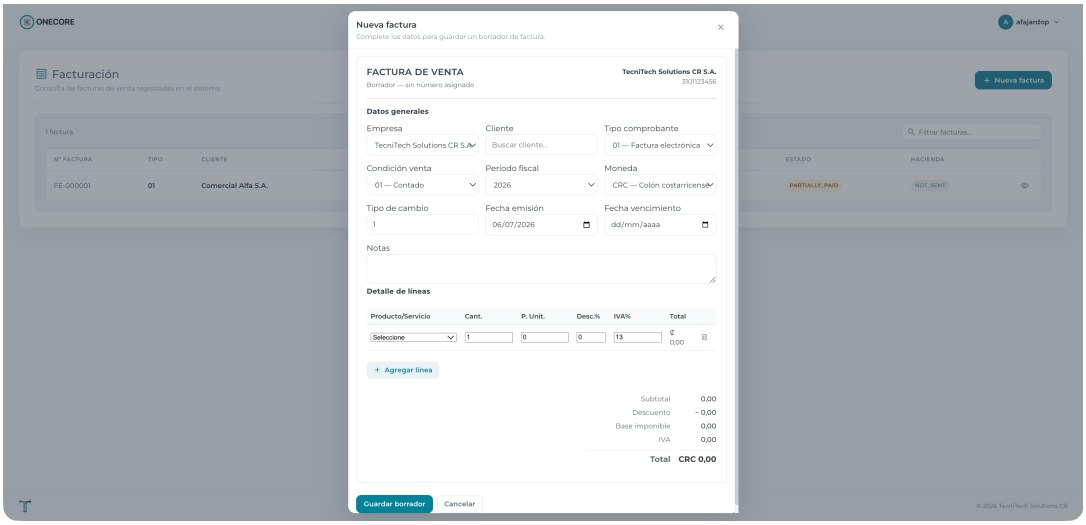
Figura 10.
Pantalla de Gestión de gastos.



Fuente: Elaboración propia, 2026.

La Figura 11 ilustró el formulario de registro de una nueva factura dentro del módulo de facturación. En esta pantalla, el usuario definió los datos generales del comprobante (empresa, cliente, tipo de comprobante, condición de venta, período fiscal, moneda y tipo de cambio), así como el detalle de líneas por producto o servicio, cantidad, precio unitario, descuento e IVA, con el cálculo automático del subtotal, impuestos y total de la factura.

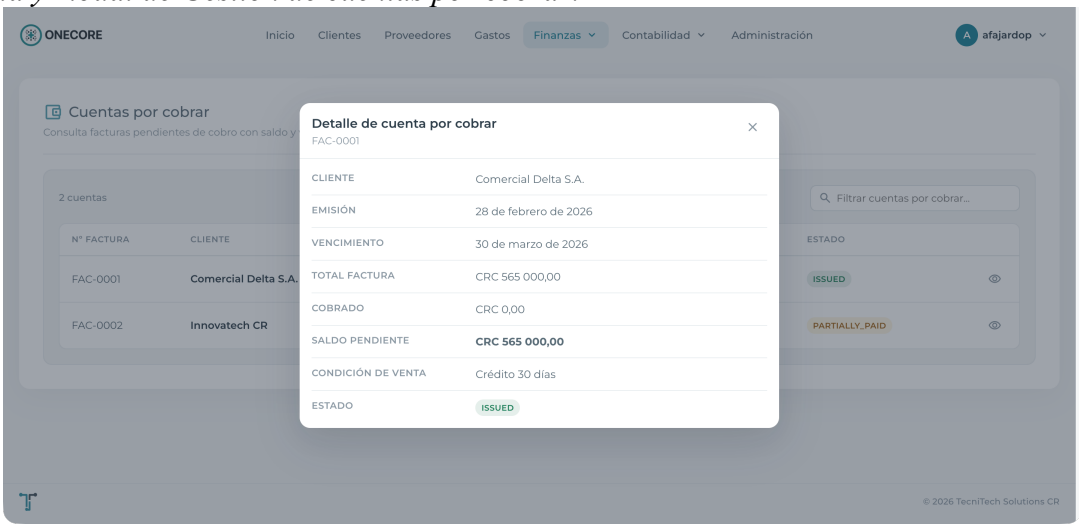
Figura 11.
Pantalla de Gestión de facturación.



Fuente: Elaboración propia, 2026.

En el módulo de cuentas por cobrar, la Figura 12 mostró el listado de cuentas pendientes de cobro junto con el detalle de una cuenta seleccionada, donde se presentó el cliente asociado, las fechas de emisión y vencimiento, el total facturado, el monto cobrado, el saldo pendiente y la condición de venta pactada.

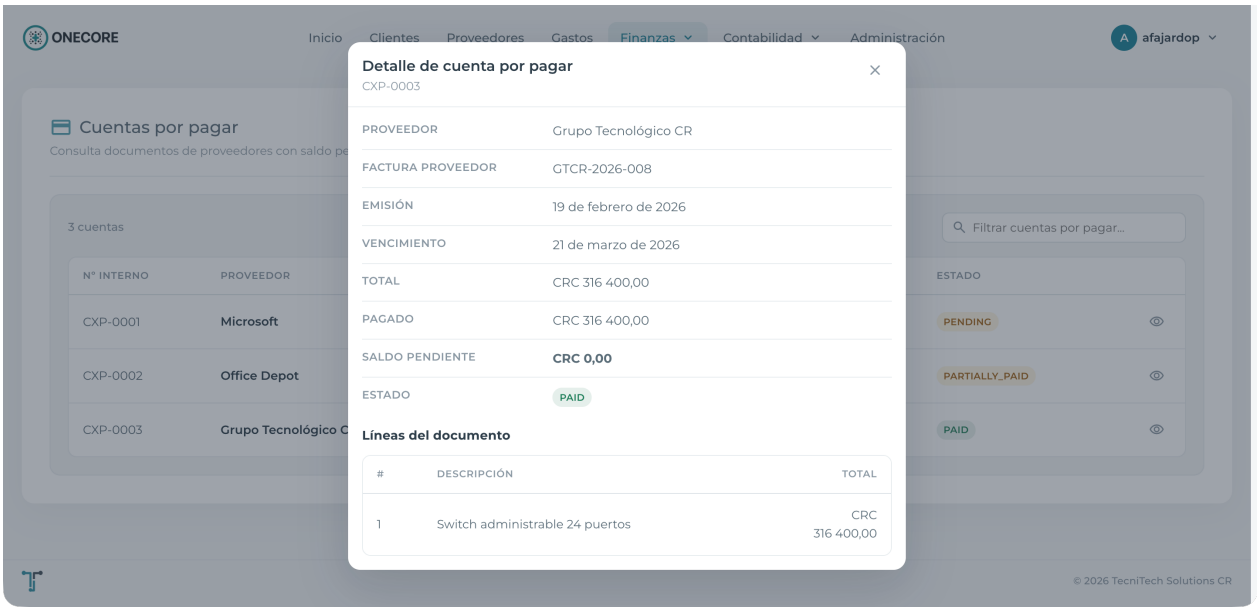
Figura 12.
Pantalla y modal de Gestión de cuentas por cobrar.



Fuente: Elaboración propia, 2026.

De manera similar, la Figura 13 presentó el módulo de cuentas por pagar, donde el usuario consultó el listado de documentos de proveedores con saldo pendiente y el detalle de una cuenta específica, incluyendo el proveedor, la factura de origen, las fechas de emisión y vencimiento, el monto total, lo pagado, el saldo pendiente y las líneas del documento.

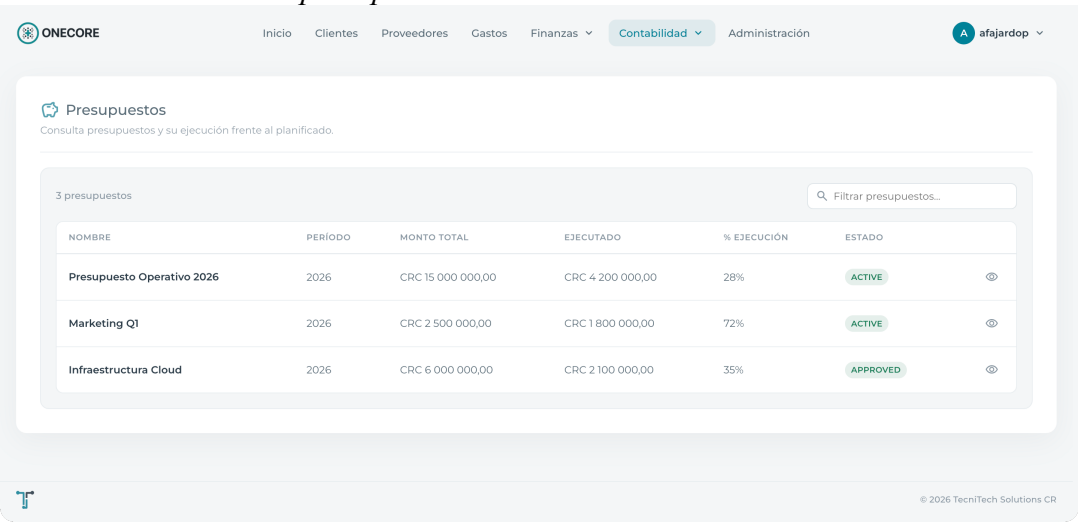
Figura 13.
Pantalla y modal de Gestión de cuentas por pagar.



Fuente: Elaboración propia, 2026.

La Figura 14 mostró la pantalla de gestión de presupuestos, donde el usuario consultó los presupuestos definidos por período, junto con el monto total asignado, el monto ejecutado, el porcentaje de ejecución y el estado de cada presupuesto.

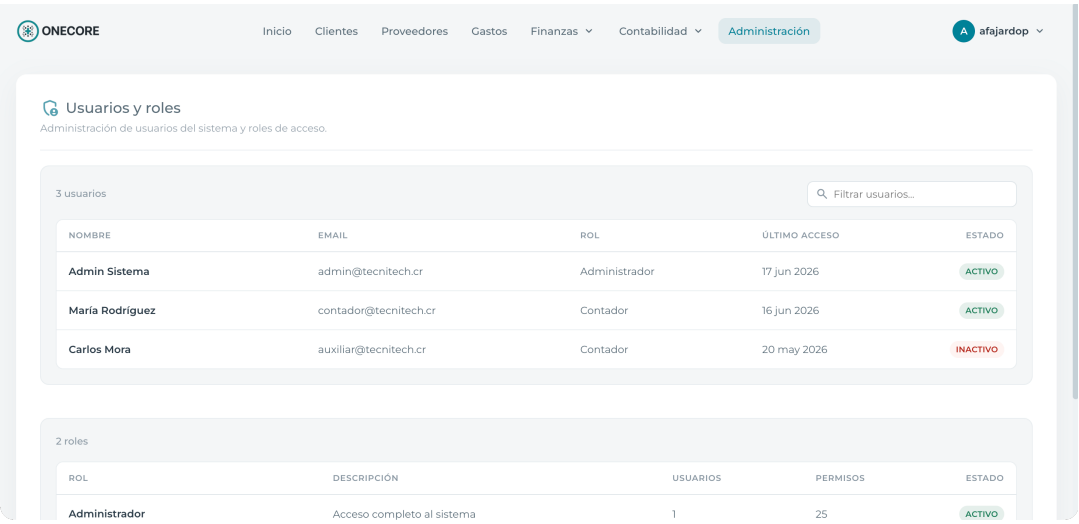
Figura 14.
Pantalla de Gestión de presupuestos.



Fuente: Elaboración propia, 2026.

Finalmente, la Figura 15 presentó la pantalla de administración de usuarios y roles, mediante la cual el usuario administrador consultó los usuarios registrados en el sistema (con su rol y último acceso) y los roles configurados, junto con la cantidad de permisos asociados a cada uno, en concordancia con el modelo de control de acceso basado en roles (RBAC) implementado en el sistema.

Figura 15.
Administración de usuarios y roles.



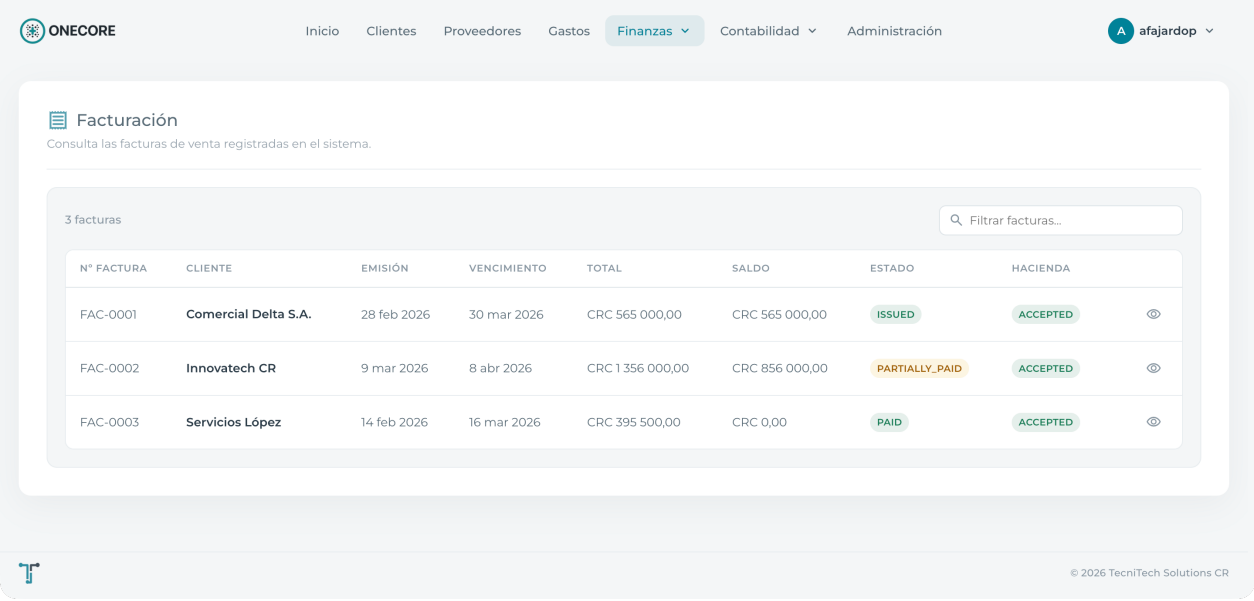
Fuente: Elaboración propia, 2026.

Diseño de Salidas

El diseño de salidas corresponde a las vistas, pantallas y documentos que el sistema genera como resultado del procesamiento de la información. Estas salidas tienen como propósito presentar al usuario la información consolidada de manera clara, estructurada y oportuna, facilitando la toma de decisiones financieras y el control operativo de la empresa. En el sistema OneCore, las salidas se clasifican en tres categorías: listados y consultas, documentos generados y reportes financieros.

Los listados y consultas corresponden a las pantallas donde el usuario visualiza el estado actual de los registros del sistema. En este caso, como se observa en la Figura 16, el listado de facturas muestra al usuario el histórico de documentos emitidos, junto con su estado (borrador, emitida, anulada, pagada, parcialmente pagada), cliente y monto total.

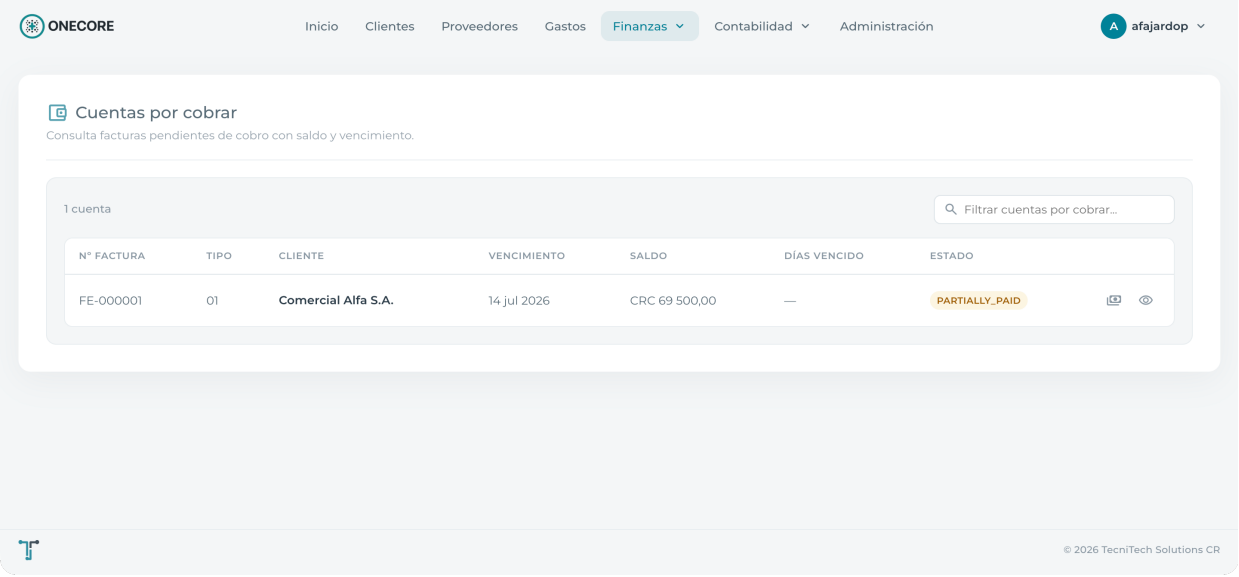
Figura 16.
Pantalla de Listado de facturas.



Fuente: Elaboración propia, 2026.

La Figura 17 ejemplifica el listado de cuentas por cobrar, el cual presenta las facturas con saldo pendiente, permitiendo al usuario identificar cuáles clientes deben y los montos correspondientes.

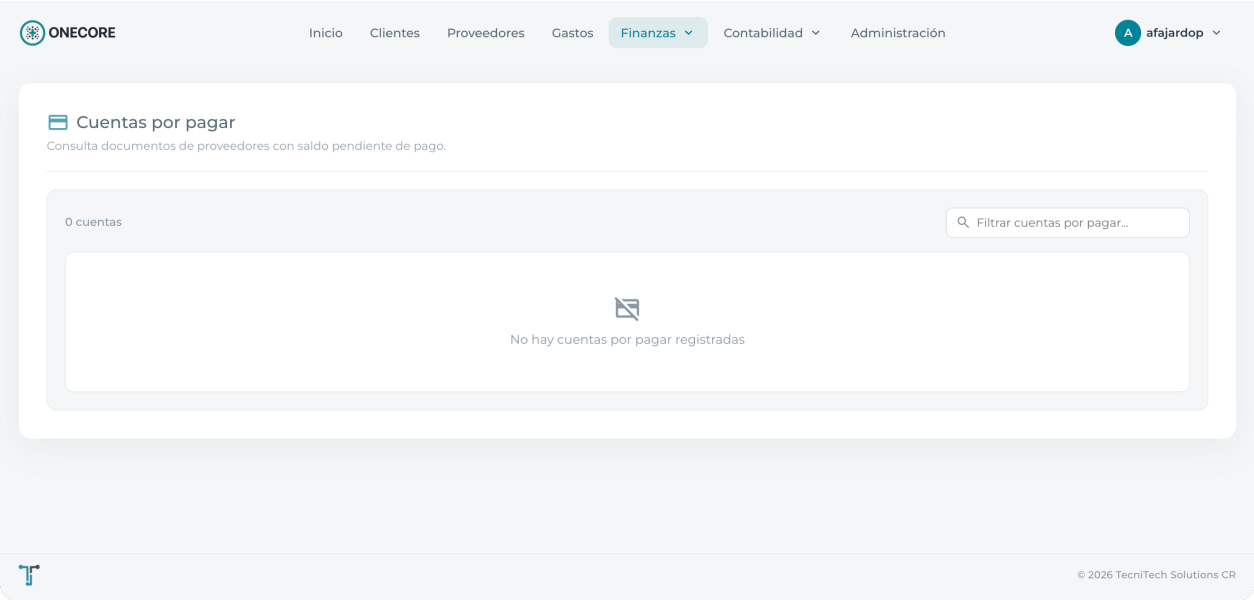
Figura 17.
Pantalla de Saldos pendientes en cuentas por cobrar.



Fuente: Elaboración propia, 2026.

En la Figura 18, el listado de cuentas por pagar muestra el detalle de facturas de proveedor pendientes de pago, con su saldo y fecha de vencimiento.

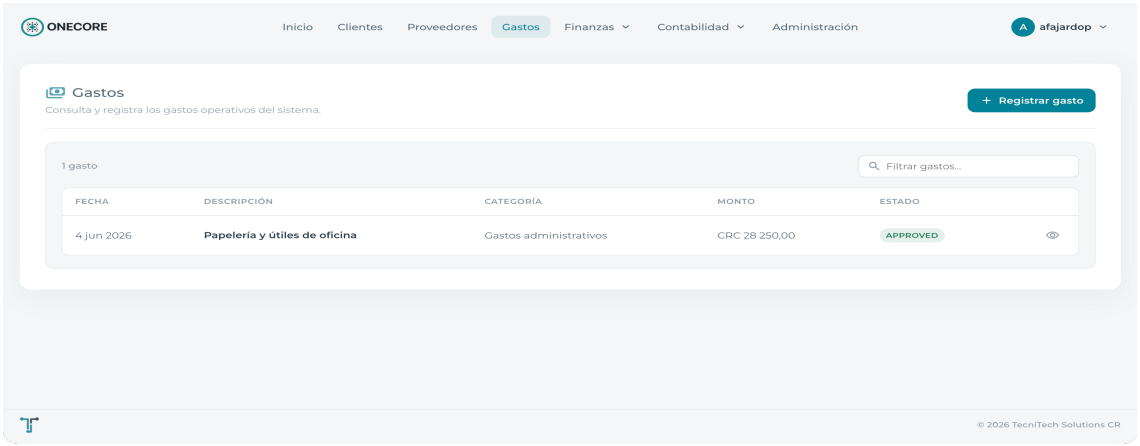
Figura 18.
Pantalla de Saldos pendientes en cuentas por pagar.



Fuente: Elaboración propia, 2026.

El histórico de gastos permite consultar cada transacción registrada, junto con su categoría, fecha, monto y estado (borrador, aprobado, anulado), como se evidencia en la Figura 19.

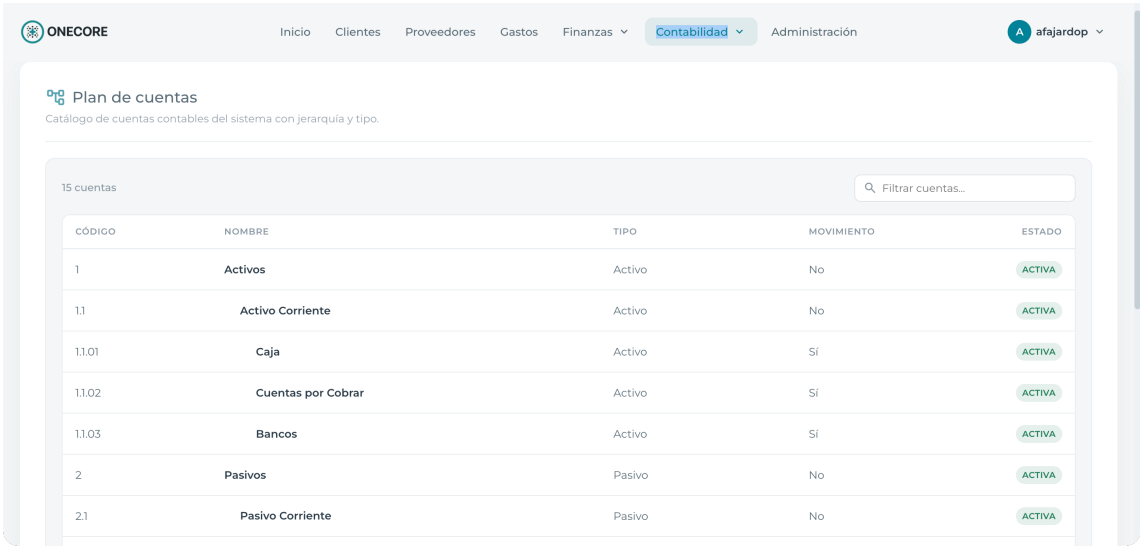
Figura 19.
Pantalla de Histórico de gastos.



Fuente: Elaboración propia, 2026.

La Figura 20 ilustra la consulta del catálogo de cuentas contables, que presenta la estructura jerárquica del plan de cuentas de la empresa.

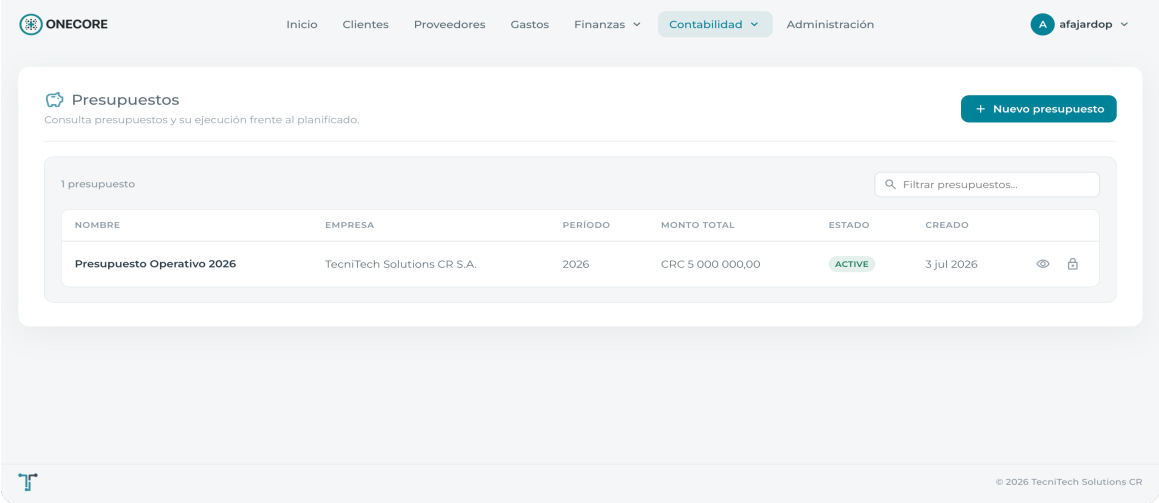
Figura 20.
Pantalla de Consulta del plan de cuentas.



Fuente: Elaboración propia, 2026.

El estado de ejecución presupuestaria compara lo planificado contra lo real por categoría de gasto, así que identifica desviaciones, tal y como se observa en la Figura 21.

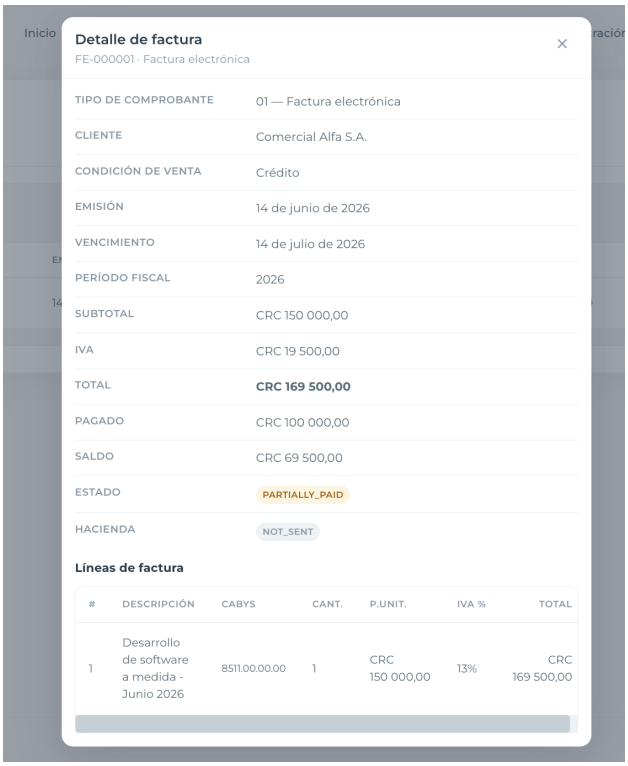
Figura 21.
Pantalla de Listado de presupuestos.



Fuente: Elaboración propia, 2026.

Los documentos generados corresponden a los comprobantes que el sistema produce a partir de la información registrada. La Figura 22 muestra el detalle de una factura, el cual presenta al usuario el desglose completo de la información asociada: cliente, líneas de producto o servicio, subtotal, descuento, impuestos y total.

Figura 22.
Pantalla de Detalle de factura.

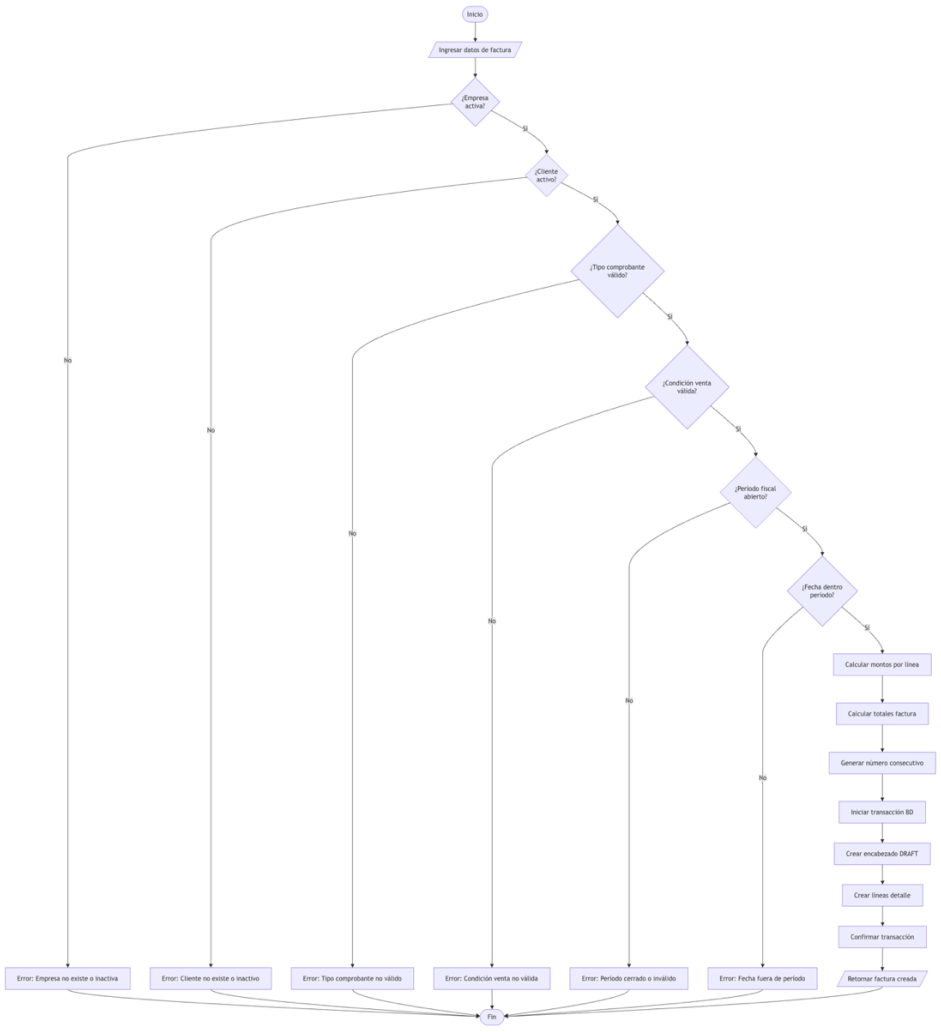


Fuente: Elaboración propia, 2026.

Diseño de Procesos

El diseño de procesos describe, mediante diagramas de flujo, la lógica interna que sigue el sistema al ejecutar las operaciones más relevantes de cada módulo funcional. Estos diagramas visualizan los pasos, las decisiones y las condiciones que determinan el resultado de cada proceso, complementando los diagramas de secuencia UML con una perspectiva orientada al flujo operativo y a las reglas de negocio aplicadas, como se ilustra en la Figura 23.

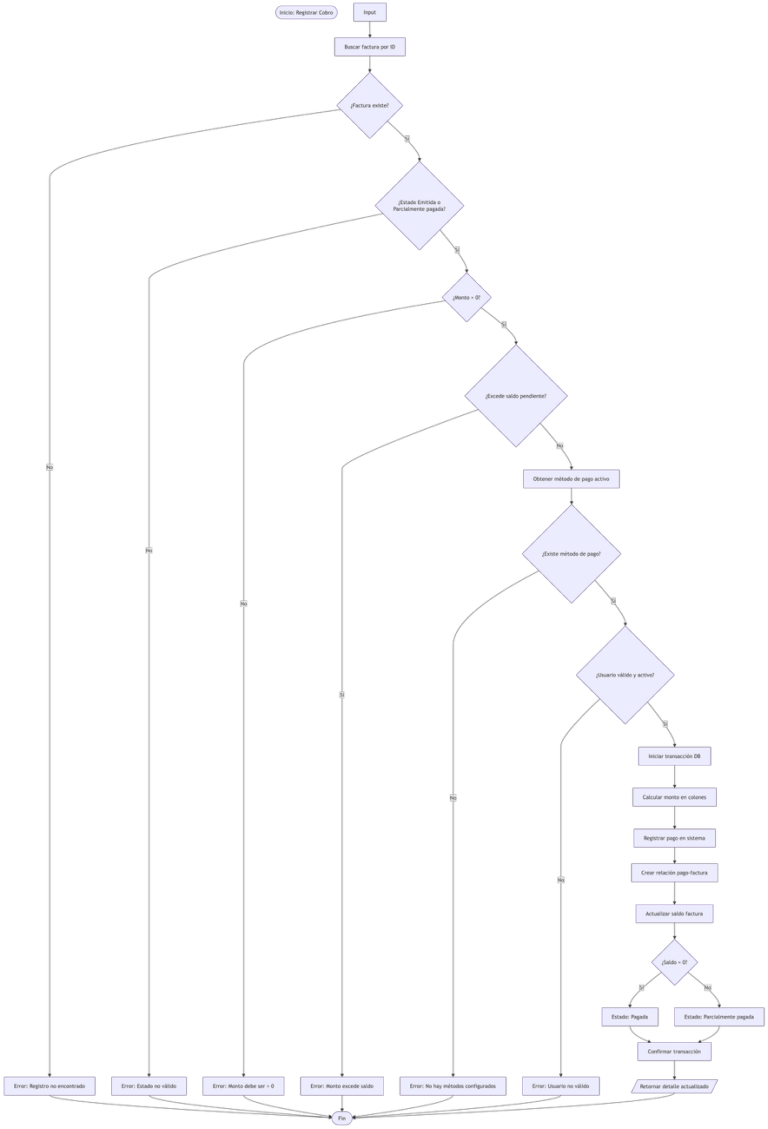
Figura 23.
Diagrama de flujo: Emisión de factura.



Fuente: Elaboración propia, 2026.

El proceso de creación de factura, según la Figura 24, inicia cuando el usuario selecciona un cliente y agrega una o más líneas de producto o servicio. El sistema valida que la empresa esté activa, que el cliente exista y esté activo, que el tipo de comprobante y la condición de venta sean válidos, y que el período fiscal esté abierto y vigente para la fecha de emisión. Calcula los montos por línea (subtotal, descuento, base imponible, impuesto, total), genera el número de factura consecutivo y almacena el encabezado junto a sus líneas dentro de una transacción de base de datos. La factura queda en estado DRAFT (borrador), lista para ser emitida o anulada posteriormente.

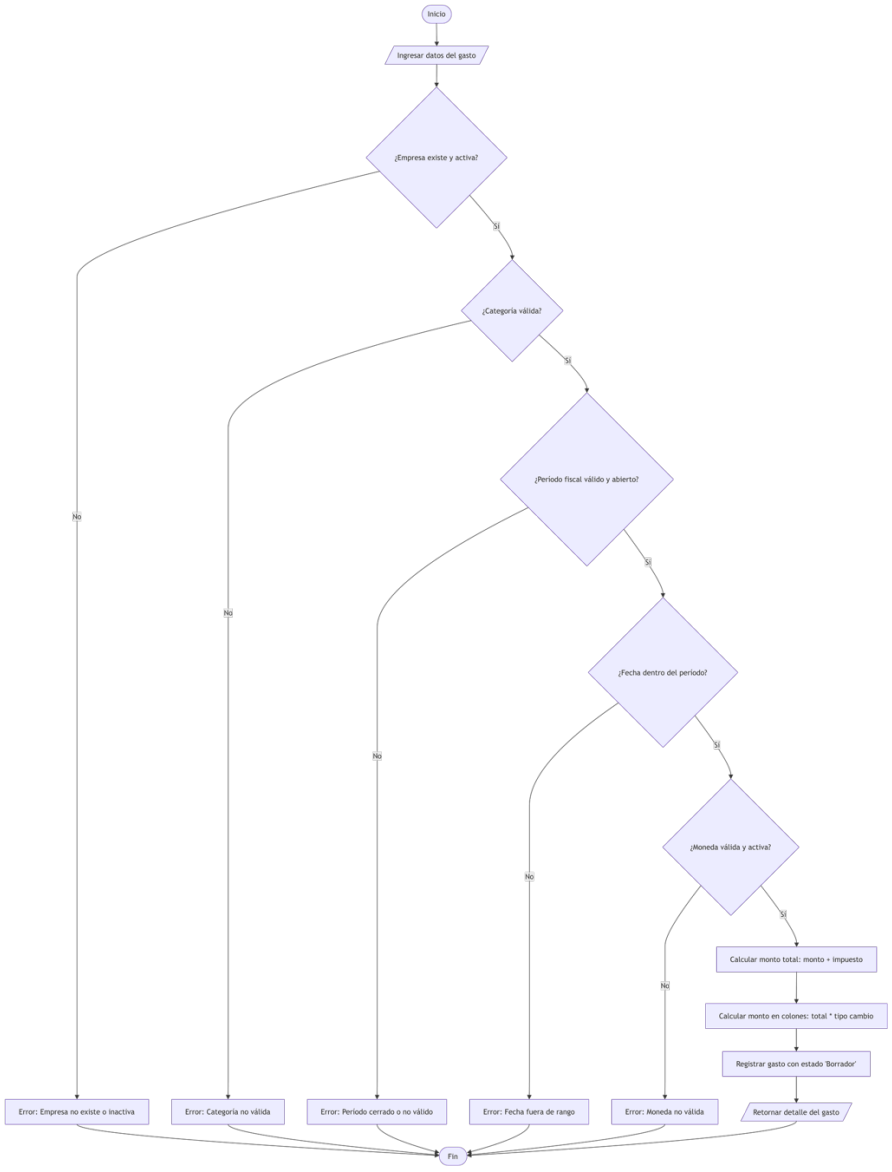
Figura 24.
Diagrama de flujo: Registro de cobro en cuentas por cobrar.



Fuente: Elaboración propia, 2026.

La Figura 25 evidencia que el proceso inicia cuando el usuario selecciona una factura con saldo pendiente. El sistema valida que la factura se encuentre en estado ISSUED o PARTIALLY_PAID, y que el monto del cobro sea mayor a cero y no exceda el saldo pendiente. Si el monto es válido, registra el pago, crea la relación entre el pago y la factura, actualiza el saldo y determina el nuevo estado: PAID, si el saldo llega a cero; en caso contrario, establece que es PARTIALLY_PAID. Toda la operación ocurre dentro de una transacción de base de datos.

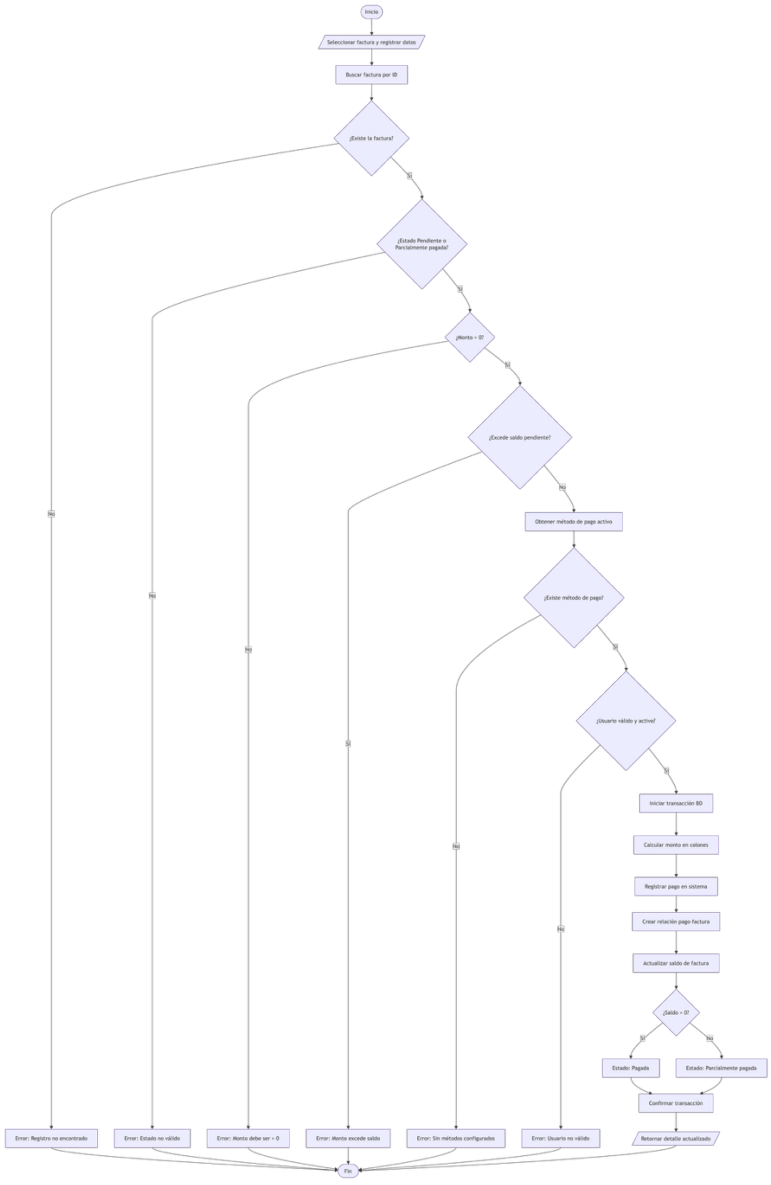
Figura 25.
Diagrama de flujo: Registro de gasto.



Fuente: Elaboración propia, 2026.

El proceso de registro de gasto inicia cuando el usuario ingresa la información del gasto: empresa, categoría, período fiscal, fecha, descripción, moneda y monto. En la Figura 26 se constata que el sistema valida que la empresa, la categoría y el período fiscal sean válidos, que el período esté abierto y que la fecha del gasto esté dentro del rango del período fiscal seleccionado. Si los datos son válidos, el sistema registra el gasto en estado DRAFT. Su aprobación o anulación ocurre en un paso posterior mediante un cambio de estado independiente.

Figura 26.
Diagrama de flujo: Registro de pago a proveedor.



Fuente: Elaboración propia, 2026.

El proceso inicia cuando el usuario selecciona una factura de proveedor con saldo pendiente. El sistema valida que la factura se encuentre en estado PENDING o PARTIALLY_PAID, y que el monto del pago sea mayor a cero y no exceda el saldo adeudado. Si el monto es válido, registra el pago, actualiza el saldo de la factura y determina su nuevo estado: PAID o PARTIALLY_PAID. La operación ocurre dentro de una transacción de base de datos.

Diseño Físico de la Base de Datos

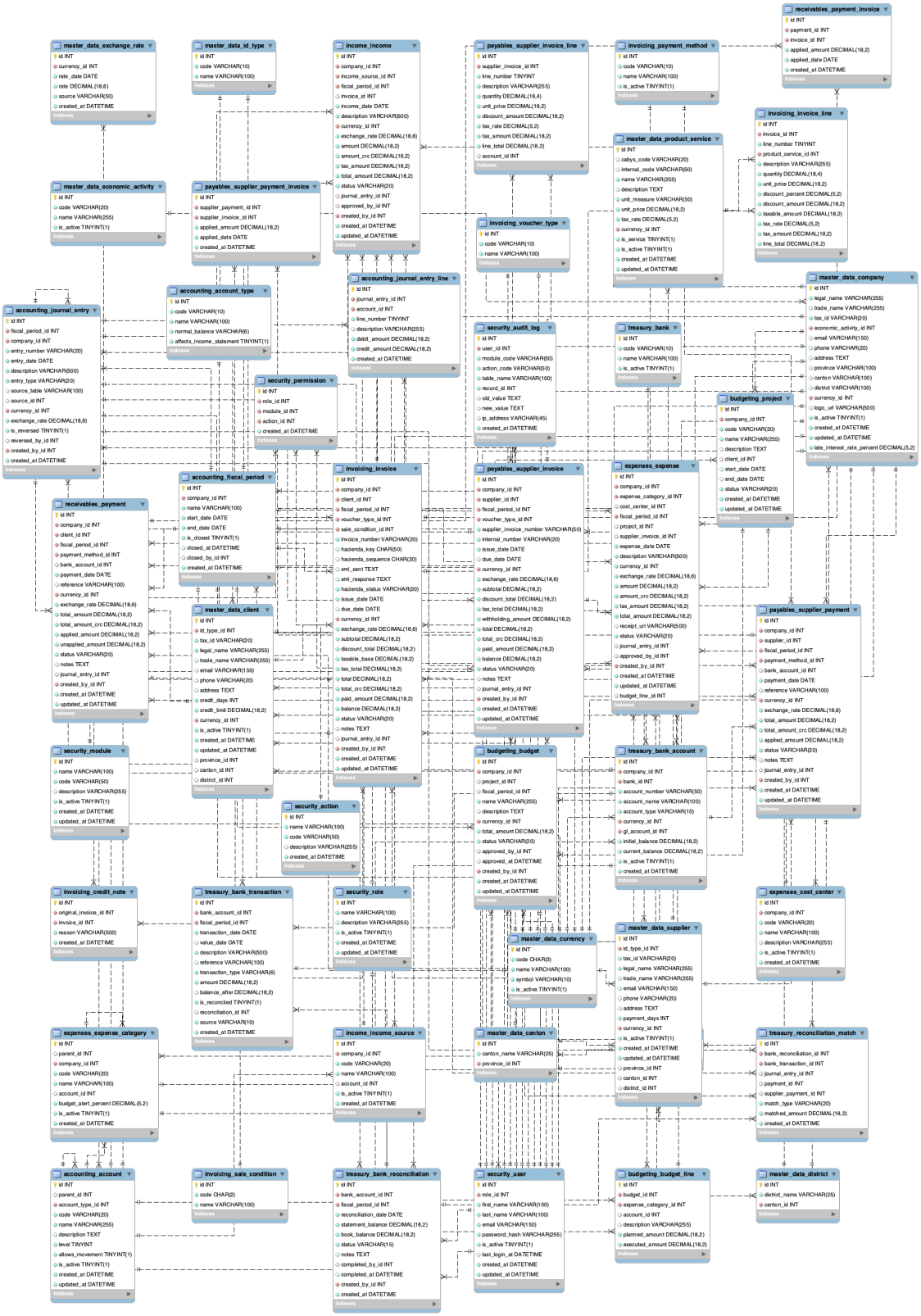
Diagrama Entidad-Relación. El diagrama entidad-relación (ER) tiene como propósito representar de forma estructurada la organización de los datos y las relaciones entre las entidades que componen la solución. Este diagrama visualiza cómo se integran los distintos dominios del sistema —seguridad, datos maestros, contabilidad y facturación—, con el fin de garantizar consistencia, trazabilidad y escalabilidad en la gestión de la información. A través de esta representación, se evidencia la forma en que las entidades clave, como usuarios, roles, cuentas contables, asientos y facturas, interactúan entre sí bajo un modelo relacional normalizado.

Desde un enfoque técnico, el modelo ER fue diseñado siguiendo buenas prácticas de modelado de datos, incluyendo los siguientes aspectos:

- Separación por dominios mediante esquemas.
- Uso de claves primarias y foráneas para asegurar integridad referencial.
- Implementación del modelo de control de acceso basado en roles.
- Aplicación del principio de partida doble en el módulo contable.

El diagrama no solo cumple una función documental, sino que también sirve como fundamento para la implementación física en base de datos, facilitando la validación del diseño, la detección de dependencias críticas y la identificación de posibles riesgos o mejoras en la arquitectura (como se observa en la Figura 26). En el contexto del trabajo final de graduación, este modelo constituye un elemento clave para demostrar la solidez del diseño de datos y su alineación con los requerimientos de un sistema contable empresarial.

Figura 27.
Diagrama Entidad-Relación.



Fuente: Elaboración propia, 2026.

Diccionario de Datos. El diccionario de datos documenta de forma estructurada los elementos que conforman la base de datos, definiendo con claridad cada tabla, sus atributos, tipos de datos y reglas asociadas. Este documento funciona como un punto de referencia técnico que permite comprender cómo está organizada la información dentro del sistema, propiciando tanto su implementación como su mantenimiento y evolución. Además, asegura consistencia en el uso de los datos, por lo que evita ambigüedades entre los distintos componentes del sistema y los equipos involucrados en su desarrollo.

Desde el punto de vista de diseño, el diccionario refleja una arquitectura modular basada en esquemas, donde cada dominio (seguridad, datos maestros, contabilidad y facturación) está claramente delimitado. Esto responde a la necesidad de mantener separación de responsabilidades, mejorar la escalabilidad y facilitar el control de cambios. Adicionalmente, el documento identifica reglas implícitas del negocio, como el control de accesos mediante roles y la correcta ejecución del modelo contable de partida doble, los elementos críticos para garantizar la integridad y confiabilidad del sistema.

En el contexto del trabajo final de graduación, el diccionario de datos no solo cumple una función descriptiva, sino que, también valida la coherencia entre el diseño lógico y la implementación física de la base de datos, evidenciando un enfoque alineado con buenas prácticas de ingeniería de software.

1. SCHEMA: security. Propósito: Seguridad y control de acceso.

Tabla: security.action

- id (INT, PK): Identificador único de la tabla.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- code (NVARCHAR(50), obligatorio): Código identificador del registro.
- description (NVARCHAR(255), opcional): Descripción detallada del registro.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- Tabla: security.audit_log
- id (INT, PK): Identificador único de la tabla.

- user_id (INT, FK): Referencia al usuario asociado (FK → security.user).
- module_code (NVARCHAR(50), obligatorio): Código del módulo donde ocurrió la acción auditada.
- action_code (NVARCHAR(50), obligatorio): Código de la acción ejecutada, registrada para auditoría.
- table_name (NVARCHAR(100), obligatorio): Nombre de la tabla afectada por la acción auditada.
- record_id (INT, obligatorio): Identificador del registro afectado por la acción auditada (referencia polimórfica según table_name; sin restricción FK declarada).
- old_value (NVARCHAR(MAX), opcional): Valor del registro previo al cambio (auditoría).
- new_value (NVARCHAR(MAX), opcional): Valor del registro posterior al cambio (auditoría).
- ip_address (NVARCHAR(45), opcional): Dirección IP desde la cual se ejecutó la acción.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

Tabla: security.module

- id (INT, PK): Identificador único de la tabla.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- code (NVARCHAR(50), obligatorio): Código identificador del registro.
- description (NVARCHAR(255), opcional): Descripción detallada del registro.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

Tabla: security.permission

- id (INT, PK): Identificador único de la tabla.

- role_id (INT, FK): Referencia al rol asociado (FK → security.role).
- module_id (INT, FK): Referencia al módulo del sistema asociado (FK → security.module).
- action_id (INT, FK): Referencia a la acción del sistema asociada (FK → security.action).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

Tabla: security.role

- id (INT, PK): Identificador único de la tabla.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- description (NVARCHAR(255), opcional): Descripción detallada del registro.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

Tabla: security.user

- id (INT, PK): Identificador único de la tabla.
- role_id (INT, FK): Referencia al rol asociado (FK → security.role).
- first_name (NVARCHAR(100), obligatorio): Nombre del usuario.
- last_name (NVARCHAR(100), obligatorio): Apellidos del usuario.
- email (NVARCHAR(150), obligatorio): Correo electrónico de contacto.
- password_hash (NVARCHAR(255), obligatorio): Contraseña del usuario almacenada de forma cifrada.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- last_login_at (DATETIME2, opcional): Fecha y hora del último inicio de sesión.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

2. SCHEMA: master_data. Propósito: Datos maestros.

Tabla: master_data.canton

- id (INT, PK): Identificador único de la tabla.
- canton_name (VARCHAR(25), obligatorio): Nombre del cantón.
- province_id (INT, FK): Referencia a la provincia asociada (FK → master_data.province).

Tabla: master_data.client

- id (INT, PK): Identificador único de la tabla.
- id_type_id (INT, FK): Referencia al tipo de identificación (FK → master_data.id_type).
- tax_id (NVARCHAR(20), obligatorio): Número de identificación tributaria (cédula jurídica o física).
- legal_name (NVARCHAR(255), obligatorio): Razón social del cliente, proveedor o empresa.
- trade_name (NVARCHAR(255), opcional): Nombre comercial del cliente, proveedor o empresa.
- email (NVARCHAR(150), opcional): Correo electrónico de contacto.
- phone (NVARCHAR(20), opcional): Número de teléfono de contacto.
- address (NVARCHAR(MAX), opcional): Dirección física.
- credit_days (INT, obligatorio): Cantidad de días de crédito otorgados.
- credit_limit (DECIMAL(18,2), obligatorio): Límite de crédito autorizado.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.
- province_id (INT, FK): Referencia a la provincia asociada (FK → master_data.province).
- canton_id (INT, FK): Referencia al cantón asociado (FK → master_data.canton).
- district_id (INT, FK): Referencia al distrito asociado (FK → master_data.district).

Tabla: master_data.company

- id (INT, PK): Identificador único de la tabla.
- legal_name (NVARCHAR(255), obligatorio): Razón social del cliente, proveedor o empresa.
- trade_name (NVARCHAR(255), opcional): Nombre comercial del cliente, proveedor o empresa.
- tax_id (NVARCHAR(20), obligatorio): Número de identificación tributaria (cédula jurídica o física).
- economic_activity_id (INT, FK): Referencia a la actividad económica asociada (FK → master_data.economic_activity).
- email (NVARCHAR(150), opcional): Correo electrónico de contacto.
- phone (NVARCHAR(20), opcional): Número de teléfono de contacto.
- address (NVARCHAR(MAX), opcional): Dirección física.
- province (NVARCHAR(100), opcional): Campo province.
- canton (NVARCHAR(100), opcional): Campo canton.
- district (NVARCHAR(100), opcional): Campo district.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- logo_url (NVARCHAR(500), opcional): Enlace al logotipo de la empresa.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.
- late_interest_rate_percent (DECIMAL(5,2), obligatorio): Porcentaje de interés moratorio aplicado por atraso en el pago.

Tabla: master_data.currency

- id (INT, PK): Identificador único de la tabla.
- code (CHAR(3), obligatorio): Código identificador del registro.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- symbol (NVARCHAR(10), obligatorio): Símbolo de la moneda.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).

Tabla: master_data.district

- id (INT, PK): Identificador único de la tabla.
- district_name (VARCHAR(25), obligatorio): Nombre del distrito.
- canton_id (INT, FK): Referencia al cantón asociado (FK → master_data.canton).

Tabla: master_data.economic_activity

- id (INT, PK): Identificador único de la tabla.
- code (NVARCHAR(20), obligatorio): Código identificador del registro.
- name (NVARCHAR(255), obligatorio): Nombre descriptivo del registro.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).

Tabla: master_data.exchange_rate

- id (INT, PK): Identificador único de la tabla.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- rate_date (DATE, obligatorio): Fecha correspondiente al tipo de cambio.

- rate (DECIMAL(18,6), obligatorio): Tasa o tipo de cambio.
- source (NVARCHAR(50), obligatorio): Origen o procedencia del dato.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

Tabla: master_data.id_type

- id (INT, PK): Identificador único de la tabla.
- code (NVARCHAR(10), obligatorio): Código identificador del registro.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.

Tabla: master_data.product_service

- id (INT, PK): Identificador único de la tabla.
- cabys_code (NVARCHAR(20), opcional): Código de clasificación CAByS (Hacienda Costa Rica) del producto o servicio.
- internal_code (NVARCHAR(50), opcional): Código interno asignado por la empresa.
- name (NVARCHAR(255), obligatorio): Nombre descriptivo del registro.
- description (NVARCHAR(MAX), opcional): Descripción detallada del registro.
- unit_measure (NVARCHAR(50), obligatorio): Unidad de medida del producto o servicio.
- unit_price (DECIMAL(18,2), obligatorio): Precio unitario del producto o servicio.
- tax_rate (DECIMAL(5,2), obligatorio): Porcentaje de impuesto aplicado.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- is_service (BIT, obligatorio): Indica si el registro corresponde a un servicio (1) o a un producto (0).
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

Tabla: master_data.province

- id (INT, PK): Identificador único de la tabla.
- province_name (VARCHAR(25), opcional): Nombre de la provincia.

Tabla: master_data.supplier

- id (INT, PK): Identificador único de la tabla.
- id_type_id (INT, FK): Referencia al tipo de identificación (FK → master_data.id_type).
- tax_id (NVARCHAR(20), obligatorio): Número de identificación tributaria (cédula jurídica o física).
- legal_name (NVARCHAR(255), obligatorio): Razón social del cliente, proveedor o empresa.
- trade_name (NVARCHAR(255), opcional): Nombre comercial del cliente, proveedor o empresa.
- email (NVARCHAR(150), opcional): Correo electrónico de contacto.
- phone (NVARCHAR(20), opcional): Número de teléfono de contacto.
- address (NVARCHAR(MAX), opcional): Dirección física.
- payment_days (INT, obligatorio): Cantidad de días de plazo de pago otorgados por el proveedor.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.
- province_id (INT, FK): Referencia a la provincia asociada (FK → master_data.province).
- canton_id (INT, FK): Referencia al cantón asociado (FK → master_data.canton).

- district_id (INT, FK): Referencia al distrito asociado (FK → master_data.district).

3. SCHEMA: accounting. Propósito: Contabilidad.

Tabla: accounting.account

- id (INT, PK): Identificador único de la tabla.
- parent_id (INT, FK): Referencia a la entidad padre dentro de la jerarquía (FK → accounting.account).
- account_type_id (INT, FK): Referencia al tipo de cuenta contable (FK → accounting.account_type).
- code (NVARCHAR(20), obligatorio): Código identificador del registro.
- name (NVARCHAR(255), obligatorio): Nombre descriptivo del registro.
- description (NVARCHAR(MAX), opcional): Descripción detallada del registro.
- level (TINYINT, obligatorio): Nivel jerárquico de la cuenta dentro del plan de cuentas.
- allows_movement (BIT, obligatorio): Indica si la cuenta admite movimientos contables directos.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

Tabla: accounting.account_type

- id (INT, PK): Identificador único de la tabla.
- code (NVARCHAR(10), obligatorio): Código identificador del registro.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- normal_balance (NVARCHAR(6), obligatorio): Naturaleza contable normal de la cuenta (débito o crédito).
- affects_income_statement (BIT, obligatorio): Indica si la cuenta afecta el estado de resultados.

Tabla: accounting.fiscal_period

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- start_date (DATE, obligatorio): Fecha de inicio.
- end_date (DATE, obligatorio): Fecha de finalización.
- is_closed (BIT, obligatorio): Indica si el periodo se encuentra cerrado.
- closed_at (DATETIME2, opcional): Fecha y hora de cierre.
- closed_by_id (INT, FK): Usuario que realizó el cierre.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

Tabla: accounting.journal_entry

- id (INT, PK): Identificador único de la tabla.
- fiscal_period_id (INT, FK): Referencia al periodo fiscal asociado (FK → accounting.fiscal_period).
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- entry_number (NVARCHAR(20), obligatorio): Número consecutivo del asiento contable.
- entry_date (DATE, obligatorio): Fecha del asiento contable.
- description (NVARCHAR(500), obligatorio): Descripción detallada del registro.
- entry_type (NVARCHAR(20), obligatorio): Tipo de asiento contable (manual o automático).
- source_table (NVARCHAR(100), opcional): Tabla de origen que generó el asiento automáticamente.
- source_id (INT, opcional): Identificador del registro de origen que generó el asiento (referencia polimórfica según source_table; sin restricción FK declarada, ya que puede apuntar a distintas tablas).

- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- exchange_rate (DECIMAL(18,6), obligatorio): Tipo de cambio aplicado a la operación.
- is_reversed (BIT, obligatorio): Indica si el movimiento fue reversado.
- reversed_by_id (INT, FK): Usuario que realizó la reversión.
- created_by_id (INT, FK): Usuario que creó el registro.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

Tabla: accounting.journal_entry_line

- id (INT, PK): Identificador único de la tabla.
- journal_entry_id (INT, FK): Referencia al asiento contable asociado (FK → accounting.journal_entry).
- account_id (INT, FK): Referencia a la cuenta contable asociada (FK → accounting.account).
- line_number (TINYINT, obligatorio): Número consecutivo de la línea dentro del documento.
- description (NVARCHAR(255), opcional): Descripción detallada del registro.
- debit_amount (DECIMAL(18,2), obligatorio): Monto registrado al debe.
- credit_amount (DECIMAL(18,2), obligatorio): Monto registrado al haber.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

4. SCHEMA: invoicing. Propósito: Facturación.

Tabla: invoicing.credit_note

- id (INT, PK): Identificador único de la tabla.
- original_invoice_id (INT, FK): Referencia a la factura original relacionada (FK → invoicing.invoice).
- invoice_id (INT, FK): Referencia a la factura asociada (FK → invoicing.invoice).
- reason (NVARCHAR(500), obligatorio): Campo reason.

- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

Tabla: invoicing.invoice

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- client_id (INT, FK): Referencia al cliente asociado (FK → master_data.client).
- fiscal_period_id (INT, FK): Referencia al periodo fiscal asociado (FK → accounting.fiscal_period).
- voucher_type_id (INT, FK): Referencia al tipo de comprobante (FK → invoicing.voucher_type).
- sale_condition_id (INT, FK): Referencia a la condición de venta (FK → invoicing.sale_condition).
- invoice_number (NVARCHAR(20), obligatorio): Número de la factura.
- hacienda_key (CHAR(50), opcional): Clave numérica del comprobante electrónico emitida por Hacienda.
- hacienda_sequence (CHAR(20), opcional): Consecutivo del comprobante electrónico según normativa de Hacienda.
- xml_sent (NVARCHAR(MAX), opcional): Contenido XML enviado a Hacienda.
- xml_response (NVARCHAR(MAX), opcional): Contenido XML de respuesta recibido de Hacienda.
- hacienda_status (NVARCHAR(20), obligatorio): Estado del comprobante ante el Ministerio de Hacienda.
- issue_date (DATE, obligatorio): Fecha de emisión del documento.
- due_date (DATE, opcional): Fecha de vencimiento del documento.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- exchange_rate (DECIMAL(18,6), obligatorio): Tipo de cambio aplicado a la operación.
- subtotal (DECIMAL(18,2), obligatorio): Subtotal antes de impuestos y descuentos.

- discount_total (DECIMAL(18,2), obligatorio): Total de descuentos aplicados.
- taxable_base (DECIMAL(18,2), obligatorio): Base imponible sobre la cual se calcula el impuesto.
- tax_total (DECIMAL(18,2), obligatorio): Total de impuestos del documento.
- total (DECIMAL(18,2), obligatorio): Total general del documento.
- total_crc (DECIMAL(18,2), obligatorio): Total general expresado en colones costarricenses.
- paid_amount (DECIMAL(18,2), obligatorio): Monto pagado hasta el momento.
- balance (DECIMAL(18,2), obligatorio): Saldo pendiente del documento.
- status (NVARCHAR(20), obligatorio): Estado actual del registro dentro de su ciclo de vida.
- notes (NVARCHAR(MAX), opcional): Notas u observaciones adicionales.
- journal_entry_id (INT, FK): Referencia al asiento contable asociado (FK → accounting.journal_entry).
- created_by_id (INT, FK): Usuario que creó el registro.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

Tabla: invoicing.invoice_line

- id (INT, PK): Identificador único de la tabla.
- invoice_id (INT, FK): Referencia a la factura asociada (FK → invoicing.invoice).
- line_number (TINYINT, obligatorio): Número consecutivo de la línea dentro del documento.
- product_service_id (INT, FK): Referencia al producto o servicio asociado (FK → master_data.product_service).
- description (NVARCHAR(255), obligatorio): Descripción detallada del registro.
- quantity (DECIMAL(18,4), obligatorio): Cantidad del producto o servicio.
- unit_price (DECIMAL(18,2), obligatorio): Precio unitario del producto o servicio.
- discount_percent (DECIMAL(5,2), obligatorio): Porcentaje de descuento aplicado.

- discount_amount (DECIMAL(18,2), obligatorio): Monto del descuento aplicado.
- taxable_amount (DECIMAL(18,2), obligatorio): Monto sujeto a impuesto.
- tax_rate (DECIMAL(5,2), obligatorio): Porcentaje de impuesto aplicado.
- tax_amount (DECIMAL(18,2), obligatorio): Monto correspondiente al impuesto aplicado.
- line_total (DECIMAL(18,2), obligatorio): Total de la línea del documento.

Tabla: invoicing.payment_method

- id (INT, PK): Identificador único de la tabla.
- code (NVARCHAR(10), obligatorio): Código identificador del registro.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).

Tabla: invoicing.sale_condition

- id (INT, PK): Identificador único de la tabla.
- code (CHAR(2), obligatorio): Código identificador del registro.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.

Tabla: invoicing.voucher_type

- id (INT, PK): Identificador único de la tabla.
- code (NVARCHAR(10), obligatorio): Código identificador del registro.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.

5. SCHEMA: expenses. Propósito: Gastos.

Tabla: expenses.cost_center

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).

- code (NVARCHAR(20), obligatorio): Código identificador del registro.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- description (NVARCHAR(255), opcional): Descripción detallada del registro.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

Tabla: expenses.expense

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- expense_category_id (INT, FK): Referencia a la categoría de gasto asociada (FK → expenses.expense_category).
- cost_center_id (INT, FK): Referencia al centro de costo asociado (FK → expenses.cost_center).
- fiscal_period_id (INT, FK): Referencia al periodo fiscal asociado (FK → accounting.fiscal_period).
- project_id (INT, FK): Referencia al proyecto asociado (FK → budgeting.project).
- supplier_invoice_id (INT, FK): Referencia a payables.supplier_invoice (FK).
- expense_date (DATE, obligatorio): Fecha en que se generó el gasto.
- description (NVARCHAR(500), obligatorio): Descripción detallada del registro.
- currency_id (INT, obligatorio): Referencia a la moneda utilizada (referencia lógica, sin restricción FK declarada en la base).
- exchange_rate (DECIMAL(18,6), obligatorio): Tipo de cambio aplicado a la operación.
- amount (DECIMAL(18,2), obligatorio): Monto de la operación.
- amount_crc (DECIMAL(18,2), obligatorio): Monto de la operación expresado en colones costarricenses.
- tax_amount (DECIMAL(18,2), obligatorio): Monto correspondiente al impuesto aplicado.

- total_amount (DECIMAL(18,2), obligatorio): Monto total de la operación.
- receipt_url (NVARCHAR(500), opcional): Enlace al comprobante o recibo digitalizado.
- status (NVARCHAR(20), obligatorio): Estado actual del registro dentro de su ciclo de vida.
- journal_entry_id (INT, FK): Referencia al asiento contable asociado (FK → accounting.journal_entry).
- approved_by_id (INT, FK): Usuario que aprobó el registro.
- created_by_id (INT, FK): Usuario que creó el registro.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.
- budget_line_id (INT, FK): Referencia a la línea presupuestaria asociada (FK → budgeting.budget_line).

Tabla: expenses.expense_category

- id (INT, PK): Identificador único de la tabla.
- parent_id (INT, FK): Referencia a la entidad padre dentro de la jerarquía (FK → expenses.expense_category).
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- code (NVARCHAR(20), obligatorio): Código identificador del registro.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- account_id (INT, FK): Referencia a la cuenta contable asociada (FK → accounting.account).
- budget_alert_percent (DECIMAL(5,2), obligatorio): Porcentaje de ejecución a partir del cual se genera una alerta.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

6. SCHEMA: income. Propósito: Ingresos.

Tabla: income.income

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- income_source_id (INT, FK): Referencia al origen del ingreso (FK → income.income_source).
- fiscal_period_id (INT, FK): Referencia al periodo fiscal asociado (FK → accounting.fiscal_period).
- invoice_id (INT, FK): Referencia a la factura asociada (FK → invoicing.invoice).
- income_date (DATE, obligatorio): Fecha en que se registró el ingreso.
- description (NVARCHAR(500), obligatorio): Descripción detallada del registro.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- exchange_rate (DECIMAL(18,6), obligatorio): Tipo de cambio aplicado a la operación.
- amount (DECIMAL(18,2), obligatorio): Monto de la operación.
- amount_crc (DECIMAL(18,2), obligatorio): Monto de la operación expresado en colones costarricenses.
- tax_amount (DECIMAL(18,2), obligatorio): Monto correspondiente al impuesto aplicado.
- total_amount (DECIMAL(18,2), obligatorio): Monto total de la operación.
- status (NVARCHAR(20), obligatorio): Estado actual del registro dentro de su ciclo de vida.
- journal_entry_id (INT, FK): Referencia al asiento contable asociado (FK → accounting.journal_entry).
- approved_by_id (INT, FK): Usuario que aprobó el registro.
- created_by_id (INT, FK): Usuario que creó el registro.

- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

Tabla: income.income_source

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- code (NVARCHAR(20), obligatorio): Código identificador del registro.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- account_id (INT, FK): Referencia a la cuenta contable asociada (FK → accounting.account).
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

7. SCHEMA: receivables. Propósito: Cuentas por cobrar.

Tabla: receivables.payment

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- client_id (INT, FK): Referencia al cliente asociado (FK → master_data.client).
- fiscal_period_id (INT, FK): Referencia al periodo fiscal asociado (FK → accounting.fiscal_period).
- payment_method_id (INT, FK): Referencia al método de pago (FK → invoicing.payment_method).
- bank_account_id (INT, FK): Referencia a la cuenta bancaria asociada (FK → treasury.bank_account).
- payment_date (DATE, obligatorio): Fecha en que se realizó el pago.

- reference (NVARCHAR(100), opcional): Referencia externa asociada a la operación.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- exchange_rate (DECIMAL(18,6), obligatorio): Tipo de cambio aplicado a la operación.
- total_amount (DECIMAL(18,2), obligatorio): Monto total de la operación.
- total_amount_crc (DECIMAL(18,2), obligatorio): Monto total expresado en colones costarricenses.
- applied_amount (DECIMAL(18,2), obligatorio): Monto aplicado a la operación relacionada.
- unapplied_amount (DECIMAL(18,2), obligatorio): Monto pendiente de aplicar.
- status (NVARCHAR(20), obligatorio): Estado actual del registro dentro de su ciclo de vida.
- notes (NVARCHAR(MAX), opcional): Notas u observaciones adicionales.
- journal_entry_id (INT, FK): Referencia al asiento contable asociado (FK → accounting.journal_entry).
- created_by_id (INT, FK): Usuario que creó el registro.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

Tabla: receivables.payment_invoice

- id (INT, PK): Identificador único de la tabla.
- payment_id (INT, FK): Referencia al pago asociado (FK → receivables.payment).
- invoice_id (INT, FK): Referencia a la factura asociada (FK → invoicing.invoice).
- applied_amount (DECIMAL(18,2), obligatorio): Monto aplicado a la operación relacionada.
- applied_date (DATE, obligatorio): Fecha en que se aplicó el monto.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

8. SCHEMA: payables. Propósito: Cuentas por pagar.

Tabla: payables.supplier_invoice

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- supplier_id (INT, FK): Referencia al proveedor asociado (FK → master_data.supplier).
- fiscal_period_id (INT, FK): Referencia al periodo fiscal asociado (FK → accounting.fiscal_period).
- voucher_type_id (INT, FK): Referencia al tipo de comprobante (FK → invoicing.voucher_type).
- supplier_invoice_number (NVARCHAR(50), obligatorio): Campo supplier invoice number.
- internal_number (NVARCHAR(20), obligatorio): Campo internal number.
- issue_date (DATE, obligatorio): Fecha de emisión del documento.
- due_date (DATE, opcional): Fecha de vencimiento del documento.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- exchange_rate (DECIMAL(18,6), obligatorio): Tipo de cambio aplicado a la operación.
- subtotal (DECIMAL(18,2), obligatorio): Subtotal antes de impuestos y descuentos.
- discount_total (DECIMAL(18,2), obligatorio): Total de descuentos aplicados.
- tax_total (DECIMAL(18,2), obligatorio): Total de impuestos del documento.
- withholding_amount (DECIMAL(18,2), obligatorio): Monto retenido en la operación.
- total (DECIMAL(18,2), obligatorio): Total general del documento.
- total_crc (DECIMAL(18,2), obligatorio): Total general expresado en colones costarricenses.

- paid_amount (DECIMAL(18,2), obligatorio): Monto pagado hasta el momento.
- balance (DECIMAL(18,2), obligatorio): Saldo pendiente del documento.
- status (NVARCHAR(20), obligatorio): Estado actual del registro dentro de su ciclo de vida.
- notes (NVARCHAR(MAX), opcional): Notas u observaciones adicionales.
- journal_entry_id (INT, FK): Referencia al asiento contable asociado (FK → accounting.journal_entry).
- created_by_id (INT, FK): Usuario que creó el registro.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

Tabla: payables.supplier_invoice_line

- id (INT, PK): Identificador único de la tabla.
- supplier_invoice_id (INT, FK): Referencia a payables.supplier_invoice (FK).
- line_number (TINYINT, obligatorio): Número consecutivo de la línea dentro del documento.
- description (NVARCHAR(255), obligatorio): Descripción detallada del registro.
- quantity (DECIMAL(18,4), obligatorio): Cantidad del producto o servicio.
- unit_price (DECIMAL(18,2), obligatorio): Precio unitario del producto o servicio.
- discount_amount (DECIMAL(18,2), obligatorio): Monto del descuento aplicado.
- tax_rate (DECIMAL(5,2), obligatorio): Porcentaje de impuesto aplicado.
- tax_amount (DECIMAL(18,2), obligatorio): Monto correspondiente al impuesto aplicado.
- line_total (DECIMAL(18,2), obligatorio): Total de la línea del documento.
- account_id (INT, FK): Referencia a la cuenta contable asociada (FK → accounting.account).

Tabla: payables.supplier_payment

- id (INT, PK): Identificador único de la tabla.

- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- supplier_id (INT, FK): Referencia al proveedor asociado (FK → master_data.supplier).
- fiscal_period_id (INT, FK): Referencia al periodo fiscal asociado (FK → accounting.fiscal_period).
- payment_method_id (INT, FK): Referencia al método de pago (FK → invoicing.payment_method).
- bank_account_id (INT, FK): Referencia a la cuenta bancaria asociada (FK → treasury.bank_account).
- payment_date (DATE, obligatorio): Fecha en que se realizó el pago.
- reference (NVARCHAR(100), opcional): Referencia externa asociada a la operación.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- exchange_rate (DECIMAL(18,6), obligatorio): Tipo de cambio aplicado a la operación.
- total_amount (DECIMAL(18,2), obligatorio): Monto total de la operación.
- total_amount_crc (DECIMAL(18,2), obligatorio): Monto total expresado en colones costarricenses.
- applied_amount (DECIMAL(18,2), obligatorio): Monto aplicado a la operación relacionada.
- status (NVARCHAR(20), obligatorio): Estado actual del registro dentro de su ciclo de vida.
- notes (NVARCHAR(MAX), opcional): Notas u observaciones adicionales.
- journal_entry_id (INT, FK): Referencia al asiento contable asociado (FK → accounting.journal_entry).
- created_by_id (INT, FK): Usuario que creó el registro.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

Tabla: payables.supplier_payment_invoice

- id (INT, PK): Identificador único de la tabla.
- supplier_payment_id (INT, FK): Referencia a payables.supplier_payment (FK).
- supplier_invoice_id (INT, FK): Referencia a payables.supplier_invoice (FK).
- applied_amount (DECIMAL(18,2), obligatorio): Monto aplicado a la operación relacionada.
- applied_date (DATE, obligatorio): Fecha en que se aplicó el monto.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

9. SCHEMA: treasury. Propósito: Tesorería y bancos.

Tabla: treasury.bank

- id (INT, PK): Identificador único de la tabla.
- code (NVARCHAR(10), obligatorio): Código identificador del registro.
- name (NVARCHAR(100), obligatorio): Nombre descriptivo del registro.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).

Tabla: treasury.bank_account

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- bank_id (INT, FK): Referencia al banco asociado (FK → treasury.bank).
- account_number (NVARCHAR(50), obligatorio): Número de la cuenta bancaria.
- account_name (NVARCHAR(100), obligatorio): Nombre de la cuenta bancaria.
- account_type (NVARCHAR(10), obligatorio): Tipo de cuenta bancaria.

- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- gl_account_id (INT, FK): Referencia a la cuenta contable del mayor asociada (FK → accounting.account).
- initial_balance (DECIMAL(18,2), obligatorio): Saldo inicial de la cuenta.
- current_balance (DECIMAL(18,2), obligatorio): Saldo actual de la cuenta.
- is_active (BIT, obligatorio): Indica si el registro se encuentra activo (1) o inactivo (0).
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

Tabla: treasury.bank_reconciliation

- id (INT, PK): Identificador único de la tabla.
- bank_account_id (INT, FK): Referencia a la cuenta bancaria asociada (FK → treasury.bank_account).
- fiscal_period_id (INT, FK): Referencia al periodo fiscal asociado (FK → accounting.fiscal_period).
- reconciliation_date (DATE, obligatorio): Fecha en que se realizó la conciliación.
- statement_balance (DECIMAL(18,2), obligatorio): Saldo según el estado de cuenta bancario.
- book_balance (DECIMAL(18,2), obligatorio): Saldo según el registro contable.
- status (NVARCHAR(15), obligatorio): Estado actual del registro dentro de su ciclo de vida.
- notes (NVARCHAR(MAX), opcional): Notas u observaciones adicionales.
- completed_by_id (INT, FK): Usuario que completó el proceso.
- completed_at (DATETIME2, opcional): Fecha y hora de finalización del proceso.
- created_by_id (INT, FK): Usuario que creó el registro.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

Tabla: treasury.bank_transaction

- id (INT, PK): Identificador único de la tabla.

- bank_account_id (INT, FK): Referencia a la cuenta bancaria asociada (FK → treasury.bank_account).
- fiscal_period_id (INT, FK): Referencia al periodo fiscal asociado (FK → accounting.fiscal_period).
- transaction_date (DATE, obligatorio): Fecha en que se realizó la transacción.
- value_date (DATE, opcional): Fecha valor de la transacción bancaria.
- description (NVARCHAR(500), obligatorio): Descripción detallada del registro.
- reference (NVARCHAR(100), opcional): Referencia externa asociada a la operación.
- transaction_type (NVARCHAR(6), obligatorio): Campo transaction type.
- amount (DECIMAL(18,2), obligatorio): Monto de la operación.
- balance_after (DECIMAL(18,2), opcional): Saldo posterior a la transacción.
- is_reconciled (BIT, obligatorio): Indica si el movimiento fue conciliado.
- reconciliation_id (INT, FK): Referencia a la conciliación bancaria asociada (FK → treasury.bank_reconciliation).
- source (NVARCHAR(10), obligatorio): Origen o procedencia del dato.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- Tabla: treasury.reconciliation_match
- id (INT, PK): Identificador único de la tabla.
- bank_reconciliation_id (INT, FK): Referencia a la conciliación bancaria asociada (FK → treasury.bank_reconciliation).
- bank_transaction_id (INT, FK): Referencia a la transacción bancaria asociada (FK → treasury.bank_transaction).
- journal_entry_id (INT, FK): Referencia al asiento contable asociado (FK → accounting.journal_entry).
- payment_id (INT, FK): Referencia al pago asociado (FK → receivables.payment).
- supplier_payment_id (INT, FK): Referencia a payables.supplier_payment (FK).
- match_type (NVARCHAR(20), obligatorio): Tipo de coincidencia identificada en la conciliación.

- matched_amount (DECIMAL(18,2), obligatorio): Monto coincidente identificado en la conciliación.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.

10. SCHEMA: budgeting. Propósito: Presupuestos.

Tabla: budgeting.budget

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- project_id (INT, FK): Referencia al proyecto asociado (FK → budgeting.project).
- fiscal_period_id (INT, FK): Referencia al periodo fiscal asociado (FK → accounting.fiscal_period).
- name (NVARCHAR(255), obligatorio): Nombre descriptivo del registro.
- description (NVARCHAR(MAX), opcional): Descripción detallada del registro.
- currency_id (INT, FK): Referencia a la moneda utilizada (FK → master_data.currency).
- total_amount (DECIMAL(18,2), obligatorio): Monto total de la operación.
- status (NVARCHAR(20), obligatorio): Estado actual del registro dentro de su ciclo de vida.
- approved_by_id (INT, FK): Usuario que aprobó el registro.
- approved_at (DATETIME2, opcional): Fecha y hora de aprobación.
- created_by_id (INT, FK): Usuario que creó el registro.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

Tabla: budgeting.budget_line

- id (INT, PK): Identificador único de la tabla.

- budget_id (INT, FK): Referencia al presupuesto asociado (FK → budgeting.budget).
- expense_category_id (INT, FK): Referencia a la categoría de gasto asociada (FK → expenses.expense_category).
- account_id (INT, FK): Referencia a la cuenta contable asociada (FK → accounting.account).
- description (NVARCHAR(255), opcional): Descripción detallada del registro.
- planned_amount (DECIMAL(18,2), obligatorio): Monto planeado dentro del presupuesto.
- executed_amount (DECIMAL(18,2), obligatorio): Monto efectivamente ejecutado del presupuesto.

Tabla: budgeting.project

- id (INT, PK): Identificador único de la tabla.
- company_id (INT, FK): Referencia a la empresa asociada (FK → master_data.company).
- code (NVARCHAR(20), obligatorio): Código identificador del registro.
- name (NVARCHAR(255), obligatorio): Nombre descriptivo del registro.
- description (NVARCHAR(MAX), opcional): Descripción detallada del registro.
- client_id (INT, FK): Referencia al cliente asociado (FK → master_data.client).
- start_date (DATE, opcional): Fecha de inicio.
- end_date (DATE, opcional): Fecha de finalización.
- status (NVARCHAR(20), obligatorio): Estado actual del registro dentro de su ciclo de vida.
- created_at (DATETIME2, obligatorio): Fecha y hora de creación del registro.
- updated_at (DATETIME2, obligatorio): Fecha y hora de la última actualización del registro.

El modelo implementa separación por esquemas, control de acceso basado en roles, contabilidad de doble partida y soporte para facturación electrónica. Cumple buenas prácticas, con oportunidades de mejora en validaciones globales y optimización.

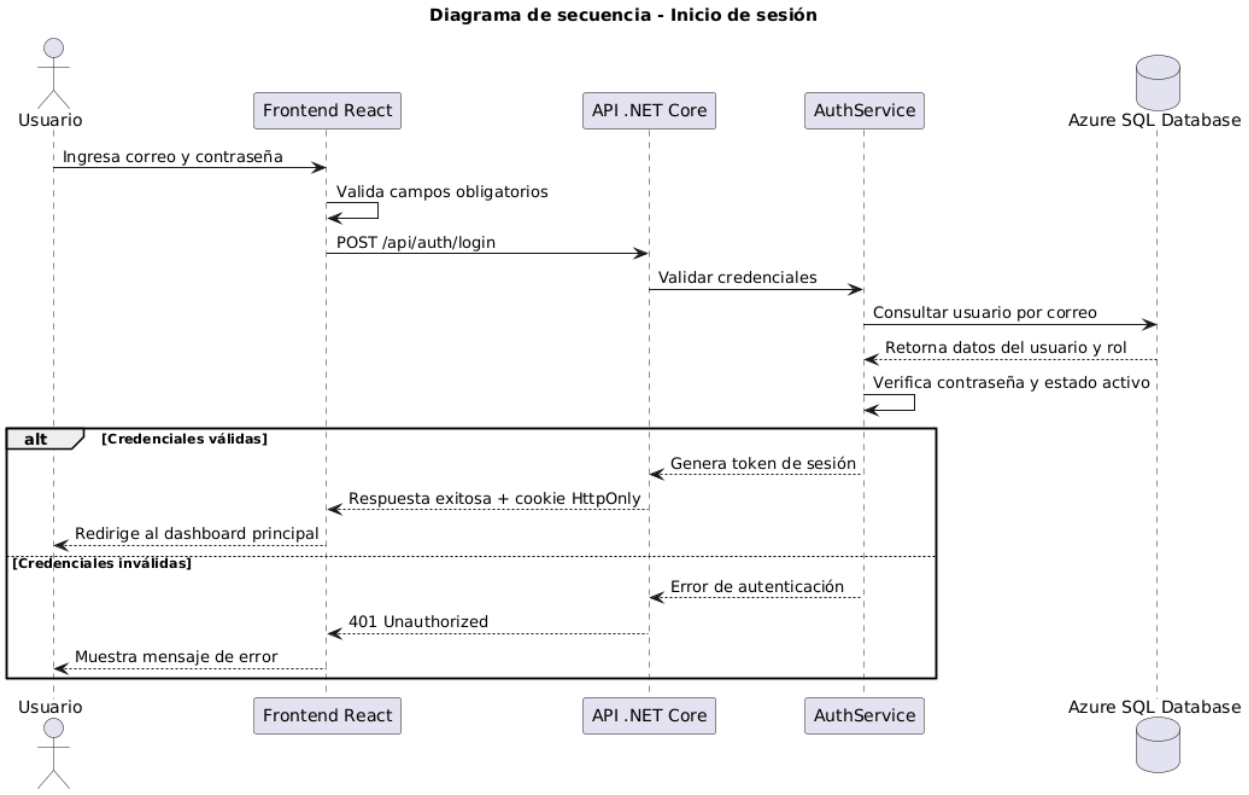
Diagramas UML

Se representan las interacciones entre los usuarios y las funcionalidades del prototipo, con el objetivo de tener una comprensión clara con respecto a los diseños que se plantean en el *software*. De este modo, se simplifica la arquitectura del código y la implementación de los diferentes módulos que conforman la propuesta, con el objetivo de que sean entendibles para los usuarios que no poseen conocimientos técnicos.

Diagramas de Secuencia. Los diagramas de secuencia representan de forma gráfica la interacción entre los diferentes componentes que participan en la ejecución de un proceso dentro del sistema. Estos diagramas muestran el orden cronológico en que se intercambian mensajes entre actores, interfaces de usuario, servicios de negocio y la base de datos, facilitando la comprensión del flujo interno de cada funcionalidad.

En el sistema OneCore, los diagramas de secuencia describen los procesos más relevantes de la solución, incluyendo la autenticación de usuarios, la gestión de facturas, el registro de pagos, el control de gastos y la generación de reportes financieros. Su propósito es complementar los casos de uso y el diagrama de clases, con el fin de proporcionar una visión más detallada del comportamiento dinámico del sistema y de cómo interactúan sus componentes durante la ejecución de las operaciones principales.

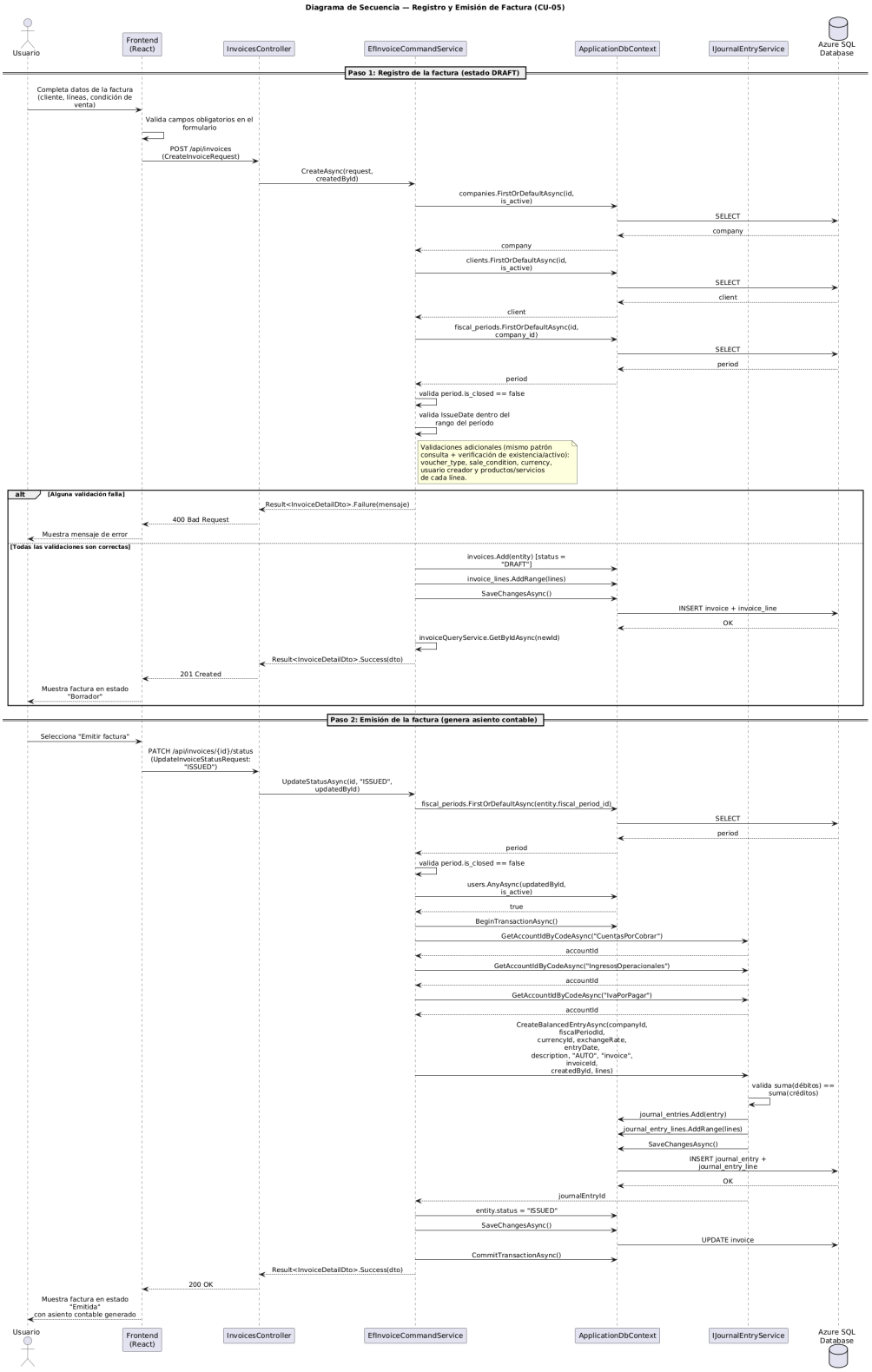
Figura 28.
Diagrama de Secuencia de inicio de sesión.



Fuente: Elaboración propia, 2026.

El diagrama de la Figura 28 detalla el proceso completo de registro y emisión de una factura en el sistema OneCore, dividido en dos pasos que corresponden a llamadas independientes dentro del código: el registro inicial de la factura en estado borrador (EfInvoiceCommandService.CreateAsync), y su posterior emisión (UpdateStatusAsync), momento en el cual el sistema genera automáticamente el asiento contable correspondiente. En el primer paso, el sistema valida la empresa, el cliente, el período fiscal (verificando que se encuentre abierto y que la fecha de emisión esté dentro de su rango), así como el tipo de comprobante, la condición de venta, la moneda y los productos o servicios incluidos en cada línea; solo si todas las validaciones son exitosas, la factura se almacena con estado DRAFT. En el segundo paso, al emitir la factura, el sistema resuelve las cuentas contables involucradas (Cuentas por Cobrar, Ingresos Operacionales e IVA por Pagar) a través de IJournalEntryService, y genera un asiento contable balanceado mediante CreateBalancedEntryAsync, el cual valida que la suma de los débitos sea igual a la suma de los créditos antes de persistirlo, garantizando así la integridad de la partida doble.

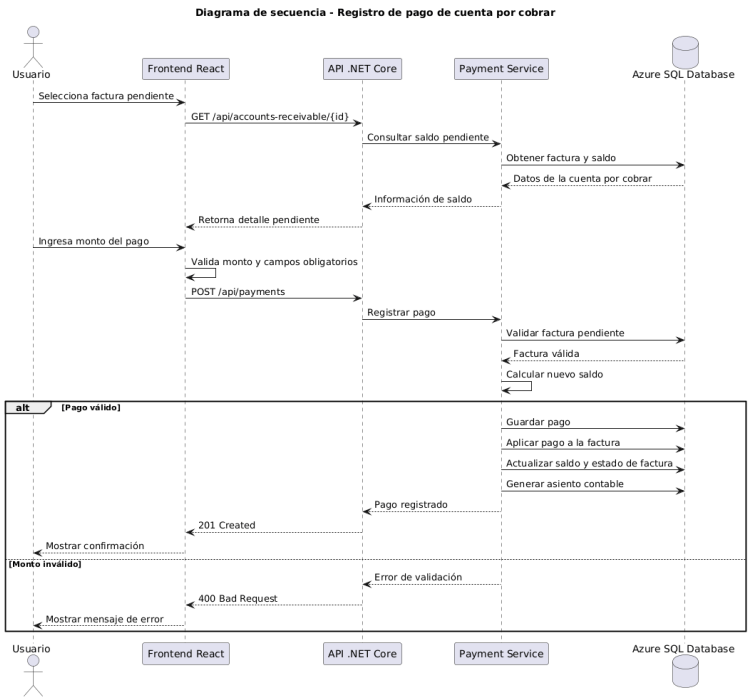
Figura 29.
Diagrama de Secuencia de registro y emisión de factura.



Fuente: Elaboración propia, 2026.

El diagrama de la Figura 29 detalla el proceso completo de registro y emisión de una factura en el sistema OneCore, dividido en dos pasos que corresponden a llamadas independientes dentro del código: el registro inicial de la factura en estado borrador (EfInvoiceCommandService.CreateAsync), y su posterior emisión (UpdateStatusAsync), momento en el cual el sistema genera automáticamente el asiento contable correspondiente. En el primer paso, el sistema valida la empresa, el cliente, el período fiscal (verificando que se encuentre abierto y que la fecha de emisión esté dentro de su rango), así como el tipo de comprobante, la condición de venta, la moneda y los productos o servicios incluidos en cada línea; solo si todas las validaciones son exitosas, la factura se almacena con estado DRAFT. En el segundo paso, al emitir la factura, el sistema resuelve las cuentas contables involucradas (Cuentas por Cobrar, Ingresos Operacionales e IVA por Pagar) a través de IJournalEntryService, y genera un asiento contable balanceado mediante CreateBalancedEntryAsync, el cual valida que la suma de los débitos sea igual a la suma de los créditos antes de persistirlo, garantizando así la integridad de la partida doble..

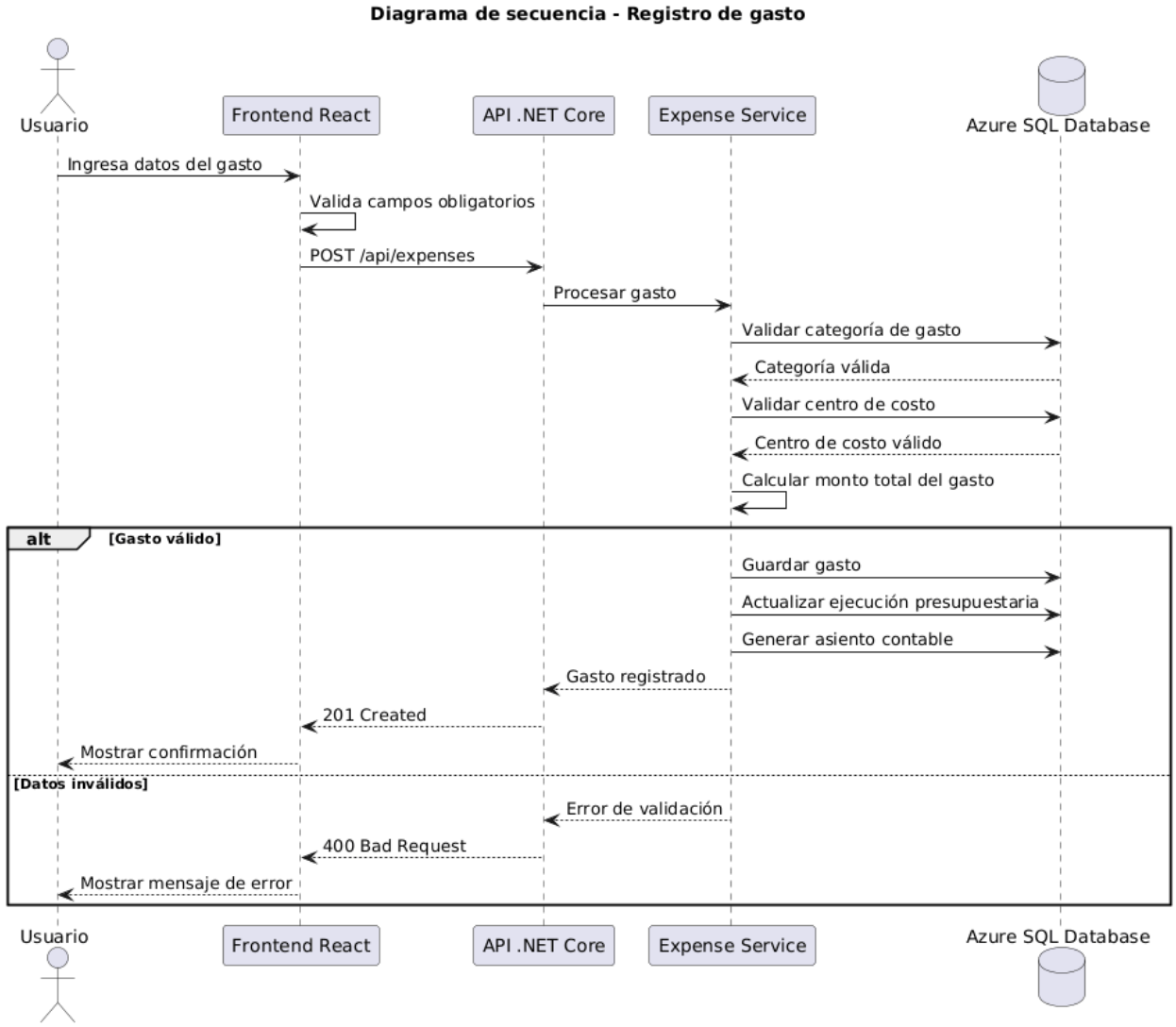
Figura 30.
Diagrama de Secuencia de registro de pago de cuenta por cobrar.



Fuente: Elaboración propia, 2026.

El diagrama de la Figura 30 evidencia el proceso del registro de un pago asociado a una cuenta por cobrar. El flujo inicia cuando el usuario consulta una factura pendiente y el sistema obtiene su saldo actual desde la base de datos. Seguidamente, el usuario ingresa el monto del pago y la API valida que la factura se encuentre pendiente y que el monto sea correcto. Si la información es válida, el sistema registra el pago, lo aplica a la factura correspondiente, actualiza el saldo pendiente y genera el asiento contable asociado. En caso de que el monto ingresado sea inválido, el sistema rechaza la operación y muestra el mensaje de error correspondiente.

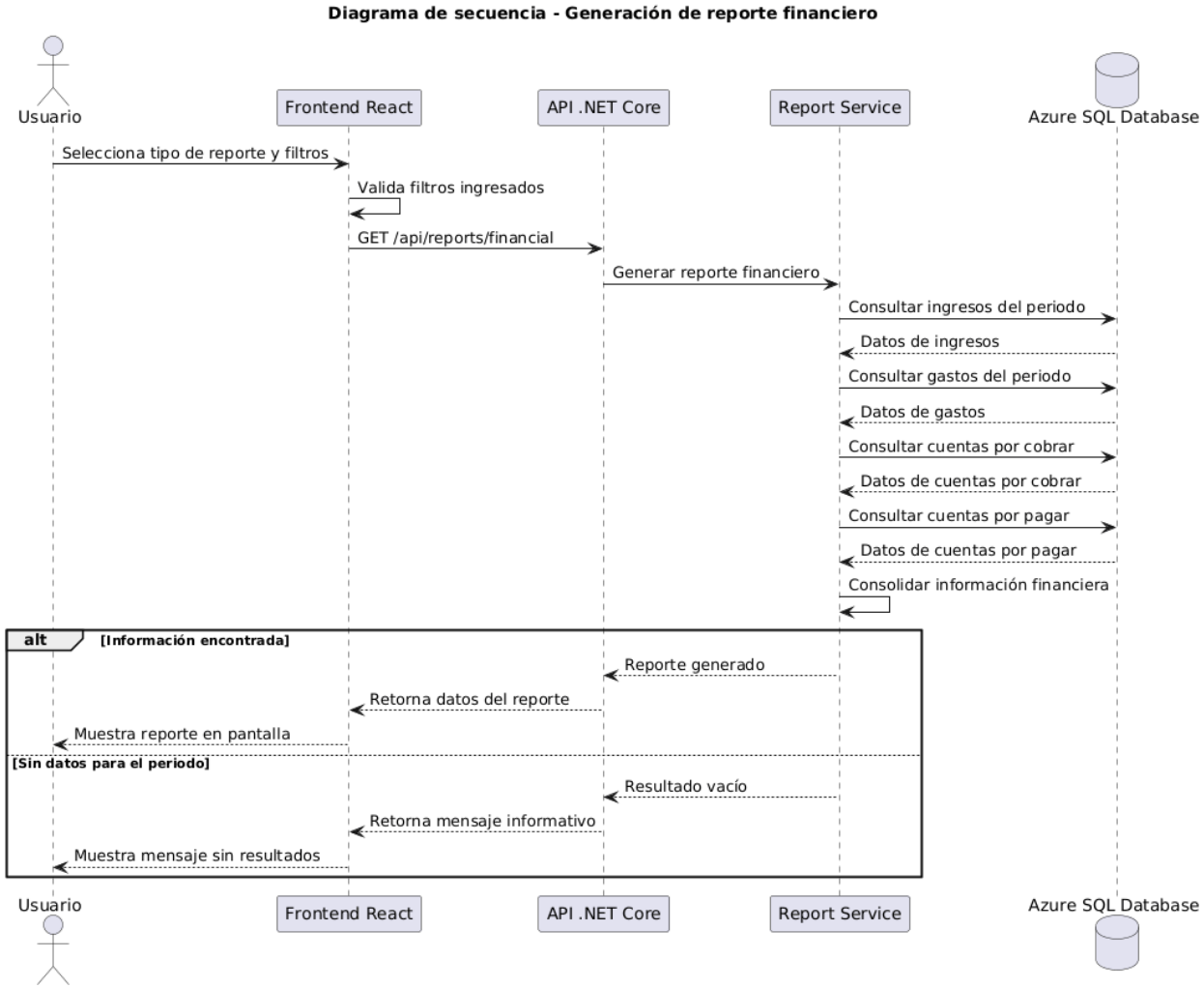
Figura 31.
Diagrama de Secuencia del registro de gasto.



Fuente: Elaboración propia, 2026.

La Figura 31 representa el proceso de registro de un gasto dentro del sistema OneCore. El flujo inicia cuando el usuario ingresa la información del gasto desde la interfaz web. Posteriormente, el frontend valida los campos obligatorios y envía la solicitud a la API. El servicio de gastos verifica la categoría y el centro de costo asociados, calcula el monto correspondiente y registra la información en la base de datos. Si los datos son válidos, el sistema actualiza la ejecución presupuestaria y genera el asiento contable relacionado con la transacción. En caso de errores de validación, la operación es rechazada y se muestra el mensaje correspondiente al usuario.

Figura 32.
Diagrama de secuencia de la generación de reporte financiero.



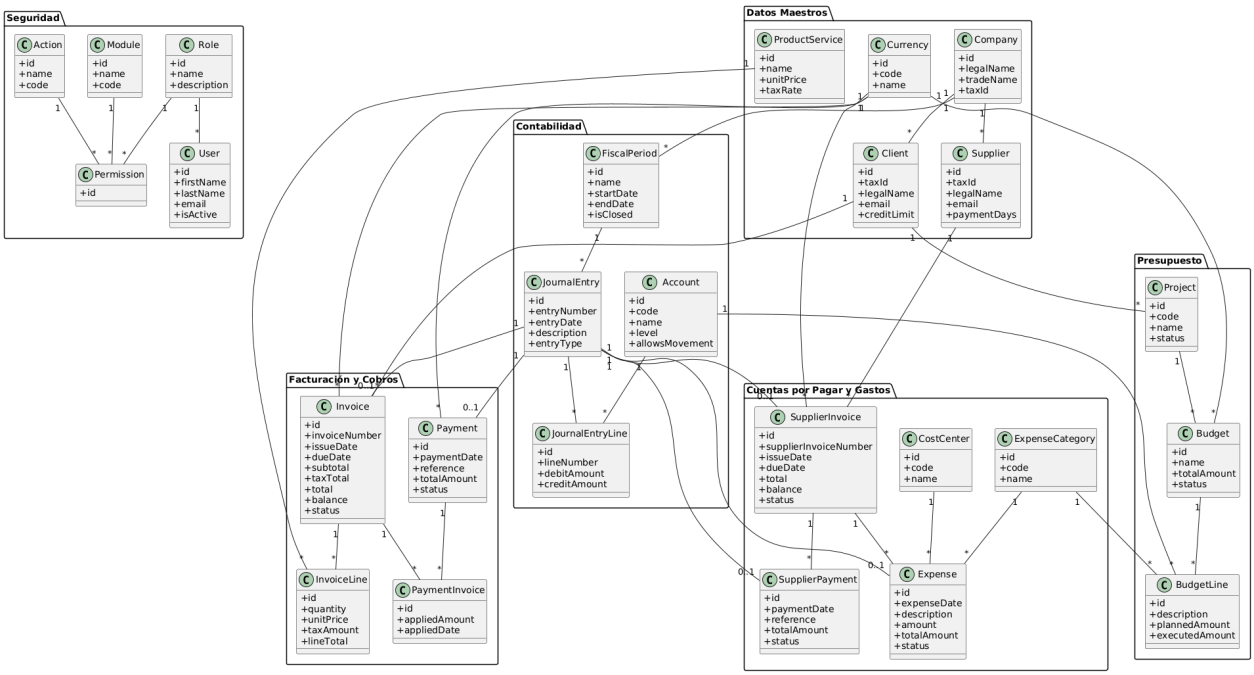
Fuente: Elaboración propia, 2026.

El diagrama de la Figura 32 muestra el proceso de generación de un reporte financiero dentro del sistema OneCore. El flujo inicia cuando el usuario selecciona el tipo de reporte y define los filtros de consulta desde la interfaz web. Como siguiente paso, el frontend valida la información ingresada y envía la solicitud a la API. El servicio de reportes consulta la información relacionada con ingresos, gastos, cuentas por cobrar y cuentas por pagar en la base de datos, con el fin de consolidar los datos financieros del periodo solicitado. Si existe información disponible, el sistema genera el reporte y lo muestra en pantalla; en caso contrario, se presenta un mensaje indicando que no se encontraron resultados para los filtros seleccionados.

Diagrama de Clases. El sistema OneCore sigue el patrón Clean Architecture con separación entre datos y comportamiento: las clases de dominio (entidades) constituyen un modelo de datos anémico, es decir, contienen únicamente atributos, mientras que la lógica de negocio reside en una capa independiente de servicios de aplicación, organizada bajo el patrón Command/Query. Por esta razón, el diagrama de clases se presenta en dos niveles complementarios: primero el modelo de entidades con sus atributos (Figura 33), y luego las clases de servicio con sus métodos, para los módulos financieros centrales del sistema (Figuras 34 a 38).

El diagrama de la Figura 33 representa la estructura principal del sistema OneCore, agrupando las entidades por módulos funcionales: Seguridad, Datos Maestros, Contabilidad, Facturación, Cuentas por Cobrar, Cuentas por Pagar, Gastos y Presupuestos. Este diseño demuestra cómo las clases principales se relacionan entre sí para soportar los procesos financiero-contables del sistema.

Figura 33.
Diagrama de clases de OneCore.

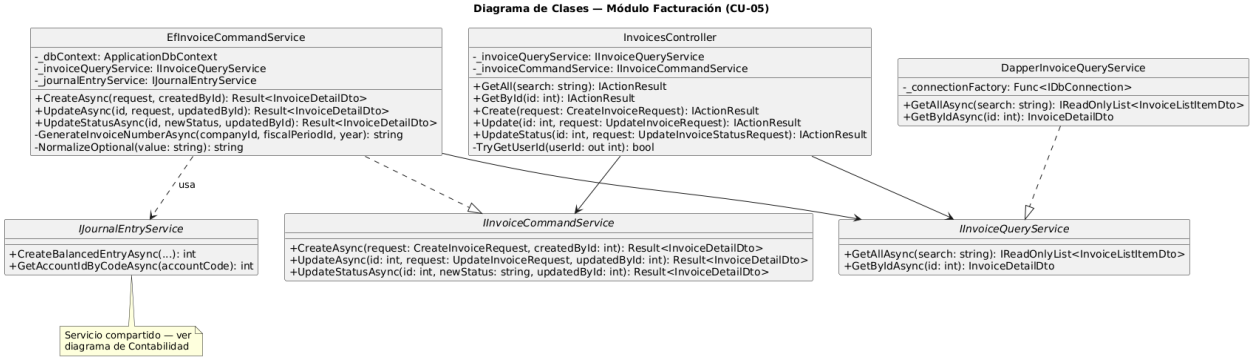


Fuente: Elaboración propia, 2026.

Dado que las clases de dominio no contienen lógica propia, las Figuras 34 a 38 detallan las clases de servicio que operan sobre ellas: los controladores HTTP, los servicios de comando (creación y modificación, con las reglas de negocio y validaciones), los servicios de consulta (lectura optimizada con Dapper), y el servicio compartido de generación de asientos contables (IJournalEntryService), utilizado por los cuatro módulos financieros centrales para mantener la integridad de la partida doble en cada operación.

La Figura 34 detalla las clases de servicio del módulo de Facturación. InvoicesController expone los endpoints HTTP y delega en IInvoiceCommandService, cuya implementación (EfInvoiceCommandService) valida la información de la factura, calcula los montos y, al emitirla, invoca a IJournalEntryService para generar el asiento contable correspondiente. La consulta de facturas se resuelve de forma independiente a través de IInvoiceQueryService, implementado con Dapper para optimizar la lectura.

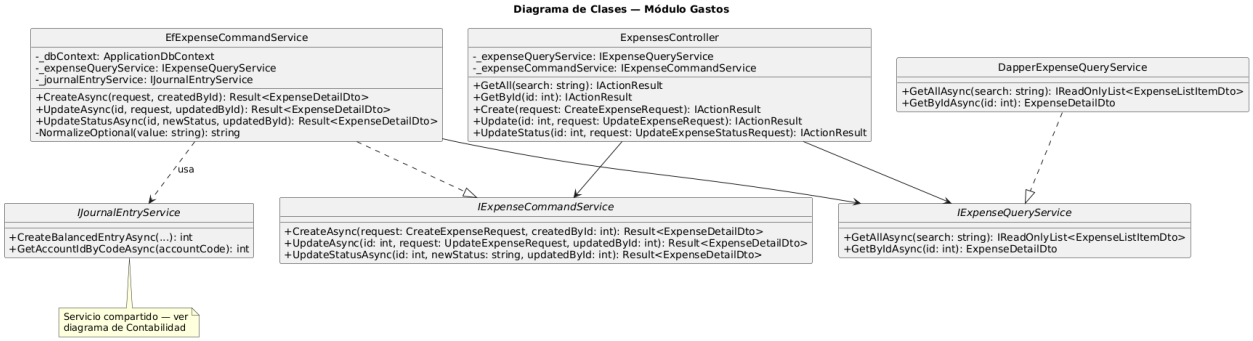
Figura 34.
Diagrama de clases — Servicios de Facturación



Fuente: Elaboración propia, 2026.

La Figura 35 muestra la estructura equivalente para el módulo de Gastos. ExpensesController delega en IExpenseCommandService, cuya implementación valida el período fiscal y la categoría del gasto, y genera el asiento contable únicamente al aprobar el gasto (transición de estado DRAFT a APPROVED), no al crearlo.

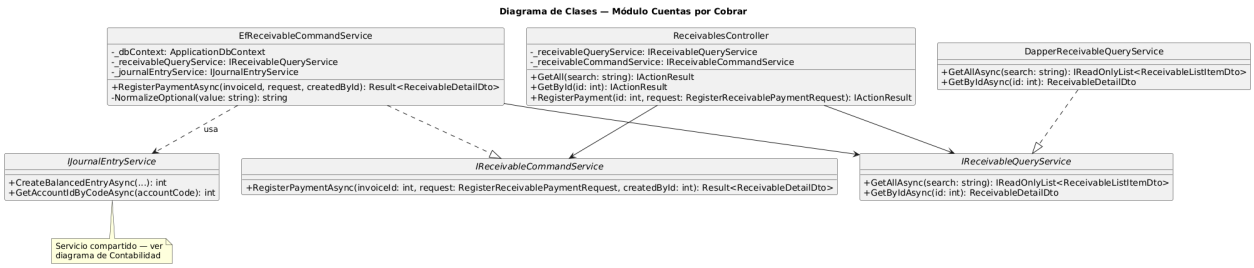
Figura 35.
Diagrama de clases — Servicios de Gastos



Fuente: Elaboración propia, 2026.

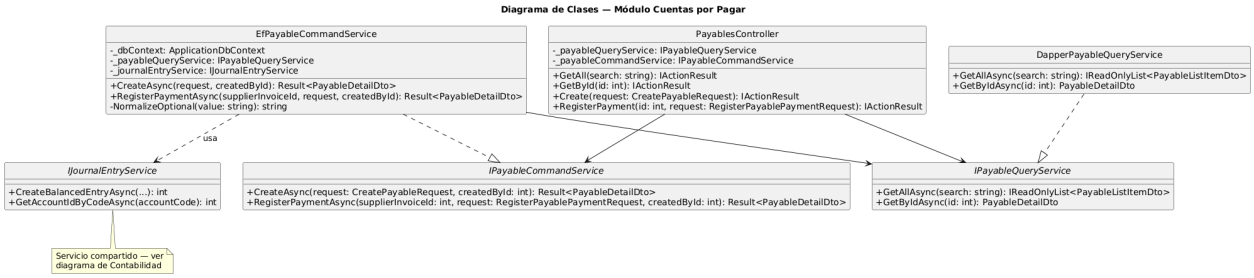
La Figura 36 corresponde al módulo de Cuentas por Cobrar. A diferencia de Facturación y Gastos, este módulo no registra creación directa de documentos, ya que las cuentas por cobrar se generan automáticamente al emitir una factura; su único método de comando es RegisterPaymentAsync, que registra el cobro de una factura existente y genera el asiento contable de la transacción.

Figura 36.
Diagrama de clases — Servicios de Cuentas por Cobrar



La Figura 37 muestra el módulo de Cuentas por Pagar, estructuralmente equivalente al de Cuentas por Cobrar, pero orientado a las facturas de proveedores: IPayableCommandService expone tanto la creación de una cuenta por pagar (CreateAsync) como el registro de su pago (RegisterPaymentAsync), ambos con generación automática de asiento contable.

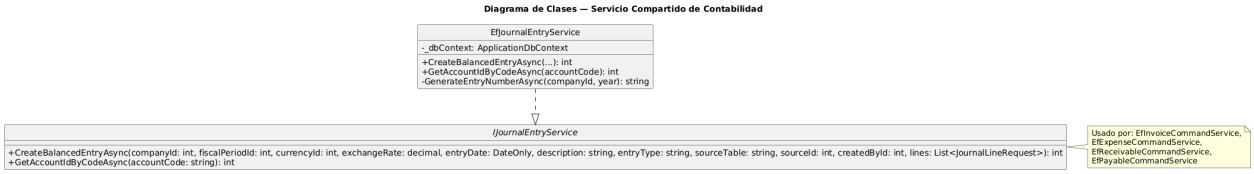
Figura 37.
Diagrama de clases — Servicios de Cuentas por Pagar



Finalmente, la Figura 38 detalla IJournalEntryService, el servicio compartido consumido por los cuatro módulos anteriores. Su método CreateBalancedEntryAsync centraliza la creación de asientos contables de doble partida, validando que la suma de débitos sea igual a la suma de créditos antes de persistir el asiento, mientras que GetAccountIdByCodeAsync resuelve el identificador de una cuenta contable a partir de su código dentro del plan de cuentas.

Figura 38.

Diagrama de clases — Servicio compartido de Contabilidad



CAPITULO VI: PROGRAMACIÓN

En esta sección, se documenta la implementación del sistema OneCore, presentando evidencia del código desarrollado en los módulos funcionales que conforman el prototipo. La programación del sistema se realiza siguiendo los principios de Clean Architecture: se separa la lógica de negocio, la capa de aplicación, la infraestructura y la presentación, con el propósito de garantizar un código mantenible, comprobable y escalable. Los componentes del backend se desarrollan en .NET 10 con ASP.NET Core Web API, mientras que el frontend se implementa con React 19 y TypeScript, comunicados a través de una API REST segura mediante HTTPS y autenticación basada en cookies HttpOnly.

Entradas

Las entradas del sistema corresponden a los objetos de solicitud (request DTOs) y a los métodos de los controladores que reciben los datos enviados por el cliente por medio de una petición HTTP (POST). Cada entrada valida su estructura antes de ser procesada por la capa de aplicación. Como se detalla en la Figura 39, la entrada principal del módulo de facturación es el objeto CreateInvoiceRequest, recibido por el método Create de InvoicesController en el endpoint POST /api/invoices.

Figura 39.
Código InvoicesController.Create.

```
[HttpPost]
[ProducesResponseType(typeof(ApiResponse<InvoiceDetailDto>), StatusCodes.Status201Created)]
[ProducesResponseType(typeof(ApiResponse), StatusCodes.Status400BadRequest)]
public async Task<IActionResult> Create([FromBody] CreateInvoiceRequest request, CancellationToken cancellationToken)
{
    if (!TryGetUserId(out var userId))
        return Unauthorized(ApiMessages.Error.Unauthorized);

    var result = await _invoiceCommandService.CreateAsync(request, userId, cancellationToken);
    if (!result.IsSuccess)
        return BadRequest(result.Error);

    return Created(result.Value!);
}
```

Fuente: Elaboración propia, 2026.

En el módulo de Gastos, según la Figura 40, la entrada corresponde al objeto CreateExpenseRequest, recibido por el método Create de ExpensesController en POST /api/expenses.

Figura 40.
Código ExpensesController.Create.

```
[HttpPost]
[ProducesResponseType(typeof(ApiResponse<ExpenseDetailDto>), StatusCodes.Status201Created)]
[ProducesResponseType(typeof(ApiResponse), StatusCodes.Status400BadRequest)]
public async Task<IActionResult> Create([FromBody] CreateExpenseRequest request, CancellationToken cancellationToken)
{
    if (!TryGetUserId(out var userId))
    {
        return Unauthorized(ApiMessages.Error.Unauthorized);
    }

    var result = await _expenseCommandService.CreateAsync(request, userId, cancellationToken);
    if (!result.IsSuccess)
        return BadRequest(result.Error);

    return Created(result.Value!);
}
```

Fuente: Elaboración propia, 2026.

En el módulo de Cuentas por Cobrar, la entrada corresponde al objeto RegisterReceivablePaymentRequest, recibido por el método RegisterPayment de ReceivablesController en POST /api/receivables/{id}/payments, de acuerdo con la Figura 41.

Figura 41.
Código ReceivablesController.RegisterPayment.

```
[HttpPost("{id:int}/payments")]
[ProducesResponseType(typeof(ApiResponse<ReceivableDetailDto>), StatusCodes.Status201Created)]
[ProducesResponseType(typeof(ApiResponse), StatusCodes.Status400BadRequest)]
[ProducesResponseType(typeof(ApiResponse), StatusCodes.Status404NotFound)]
public async Task<IActionResult> RegisterPayment(
    int id,
    [FromBody] RegisterReceivablePaymentRequest request,
    CancellationToken cancellationToken)
{
    if (!TryGetUserId(out var userId))
        return Unauthorized(ApiMessages.Error.Unauthorized);

    var result = await _receivableCommandService.RegisterPaymentAsync(id, request, userId, cancellationToken);
    if (!result.IsSuccess)
    {
        if (result.Error == "not_found")
            return NotFound(ApiMessages.Error.RecordNotFound);
        return BadRequest(result.Error);
    }

    return Created(result.Value!);
}

private bool TryGetUserId(out int userId)
{
    var claimValue = User.FindFirstValue("userId")
        ?? User.FindFirstValue(ClaimTypes.NameIdentifier);

    return int.TryParse(claimValue, out userId) && userId > 0;
}
```

Fuente: Elaboración propia, 2026.

Salidas

Las salidas del sistema corresponden a los objetos de respuesta (DTOs de lectura) y a los métodos de los controladores y servicios de consulta que los generan. Cada salida es proyectada mediante Dapper y devuelta al frontend en una estructura ApiResponse<T> estandarizada. La salida principal del módulo de Facturación es el objeto InvoiceListItemDto, generado por DapperInvoiceQueryService.GetAllAsync y expuesto en el método GetAll de InvoicesController, como aparece en la Figura 42.

Figura 42.
Código InvoicesController.GetAll.

```
[HttpGet]
[ProducesResponseType(typeof(ApiResponse<IEnumerable<InvoiceListItemDto>>), StatusCodes.Status200OK)]
public async Task<IActionResult> GetAll([FromQuery] string? search, CancellationToken cancellationToken)
{
    var invoices = await _invoiceQueryService.GetAllAsync(search, cancellationToken);
    return OkList(invoices);
}

[HttpGet("{id:int}")]
[ProducesResponseType(typeof(ApiResponse<InvoiceDetailDto>), StatusCodes.Status200OK)]
[ProducesResponseType(typeof(ApiResponse), StatusCodes.Status404NotFound)]
public async Task<IActionResult> GetById(int id, CancellationToken cancellationToken)
{
    var invoice = await _invoiceQueryService.GetByIdAsync(id, cancellationToken);
    if (invoice is null)
    {
        return NotFound(ApiMessages.Error.RecordNotFound);
    }

    return Ok(invoice);
}
```

Fuente: Elaboración propia, 2026.

La Figura 43 demuestra que, en el módulo de Cuentas por Cobrar, la salida principal es el objeto ReceivableListItemDto, expuesto en el método GetAll de ReceivablesController.

Figura 43.
Código ReceivablesController.GetAll.

```
[HttpGet]
[ProducesResponseType(typeof(ApiResponse<IEnumerable<ReceivableListItemDto>>), StatusCodes.Status200OK)]
public async Task<IActionResult> GetAll([FromQuery] string? search, CancellationToken cancellationToken)
{
    var receivables = await _receivableQueryService.GetAllAsync(search, cancellationToken);
    return OkList(receivables);
}

[HttpGet("{id:int}")]
[ProducesResponseType(typeof(ApiResponse<ReceivableDetailDto>), StatusCodes.Status200OK)]
[ProducesResponseType(typeof(ApiResponse), StatusCodes.Status404NotFound)]
public async Task<IActionResult> GetById(int id, CancellationToken cancellationToken)
{
    var receivable = await _receivableQueryService.GetByIdAsync(id, cancellationToken);
    if (receivable is null)
    {
        return NotFound(ApiMessages.Error.RecordNotFound);
    }

    return Ok(receivable);
}
```

Fuente: Elaboración propia, 2026.

Para el módulo de Gastos, de conformidad con la Figura 44, la salida principal es el objeto ExpenseListItemDto, expuesto en el método GetAll de ExpensesController.

Figura 44.
Código ExpensesController.GetAll.

```
[HttpGet("{id:int}")]
[ProducesResponseType(typeof(ApiResponse<ExpenseDetailDto>), StatusCodes.Status200OK)]
[ProducesResponseType(typeof(ApiResponse), StatusCodes.Status404NotFound)]
public async Task<IActionResult> GetById(int id, CancellationToken cancellationToken)
{
    var expense = await _expenseQueryService.GetByIdAsync(id, cancellationToken);
    if (expense is null)
    {
        return NotFound(ApiMessages.Error.RecordNotFound);
    }

    return Ok(expense);
}
```

Fuente: Elaboración propia, 2026.

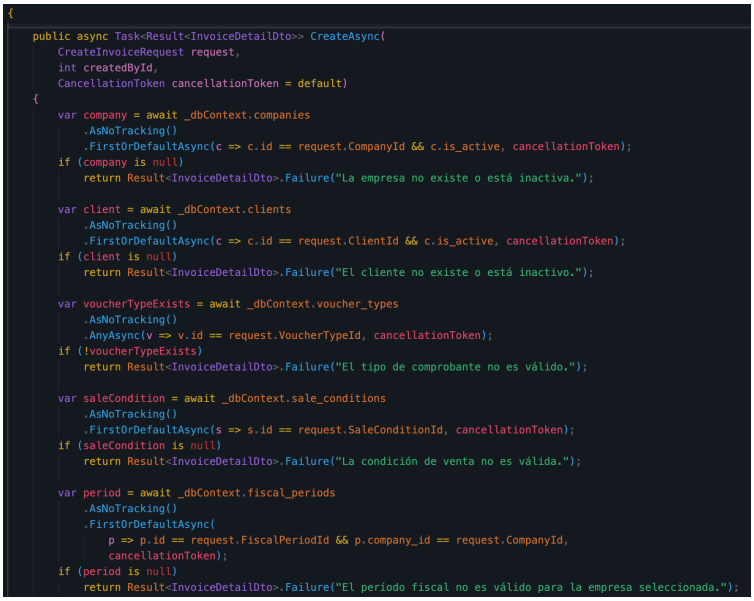
Procesos

Los procesos corresponden a la lógica de negocio que el sistema ejecuta internamente al recibir una entrada, antes de generar una salida. En OneCore, los procesos principales se implementan como servicios de comando (*command services*) dentro de la capa de infraestructura, siguiendo el patrón Query/Command de Clean Architecture: las lecturas se resuelven con Dapper (Dapper*QueryService) y las escrituras con Entity Framework Core (Ef*CommandService). Cada servicio de comando encapsula una operación del negocio, aplica las validaciones correspondientes y coordina el acceso a datos dentro de transacciones explícitas cuando la operación involucra múltiples tablas.

La creación de factura es el proceso de mayor complejidad del sistema, como se evidencia en la Figura 45, ya que integra múltiples validaciones y cálculos en una sola transacción. El método CreateAsync de la clase EfInvoiceCommandService recibe el comando, valida que la empresa esté habilitada, que el cliente exista y esté activo, que el tipo de comprobante y la condición de venta sean válidos, y que el período fiscal esté abierto y vigente para la fecha de emisión.

Además, calcula los montos de cada línea (subtotal, descuento, base imponible, impuesto y total), genera el número de factura consecutivo por empresa y período, y almacena el encabezado junto a sus líneas de detalle. Todo el proceso ocurre dentro de una transacción explícita de Entity Framework (BeginTransactionAsync), garantizando que el encabezado y las líneas se registren de forma atómica.

Figura 45.
Proceso de Creación de factura.

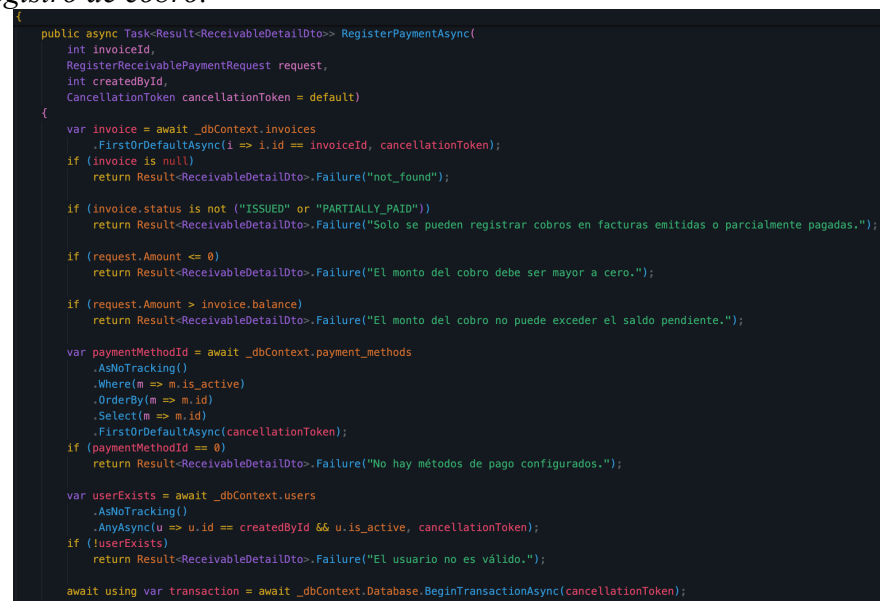


```
{
    public async Task<Result<InvoiceDetailDto>> CreateAsync(
        CreateInvoiceRequest request,
        int createdById,
        CancellationToken cancellationToken = default)
    {
        var company = await _dbContext.companies
            .AsNoTracking()
            .FirstOrDefaultAsync(c => c.id == request.CompanyId && c.is_active, cancellationToken);
        if (company is null)
            return Result<InvoiceDetailDto>.Failure("La empresa no existe o está inactiva.");

        var client = await _dbContext.clients
            .AsNoTracking()
            .FirstOrDefaultAsync(c => c.id == request.ClientId && c.is_active, cancellationToken);
        if (client is null)
            return Result<InvoiceDetailDto>.Failure("El cliente no existe o está inactivo.");

        var voucherTypeExists = await _dbContext.voucher_types
            .AsNoTracking()
            .AnyAsync(v => v.id == request.VoucherTypeId, cancellationToken);
        if (!voucherTypeExists)
            return Result<InvoiceDetailDto>.Failure("El tipo de comprobante no es válido.");

        var saleCondition = await _dbContext.sale_conditions
            .AsNoTracking()
            .FirstOrDefaultAsync(s => s.id == request.SaleConditionId, cancellationToken);
        if (saleCondition is null)
            return Result<InvoiceDetailDto>.Failure("La condición de venta no es válida.");

        var period = await _dbContext.fiscal_periods
            .AsNoTracking()
            .FirstOrDefaultAsync(
                p => p.id == request.FiscalPeriodId && p.company_id == request.CompanyId,
                cancellationToken);
        if (period is null)
            return Result<InvoiceDetailDto>.Failure("El período fiscal no es válido para la empresa seleccionada.");
```

Fuente: Elaboración propia, 2026.

En la Figura 46, se observa que el proceso de registro de cobro en cuentas por cobrar es ejecutado por el método RegisterPaymentAsync de la clase EfReceivableCommandService. Este proceso recupera la factura por su identificador, valida que se encuentre en estado ISSUED o PARTIALLY_PAID, y verifica que el monto del cobro no supere el saldo pendiente. Si las validaciones son superadas, registra el pago en la tabla receivables.payment, crea la relación en payment_invoice, actualiza el saldo de la factura y determina su nuevo estado. Toda la operación ocurre dentro de una transacción de base de datos.

Figura 46.
Proceso de Registro de cobro.

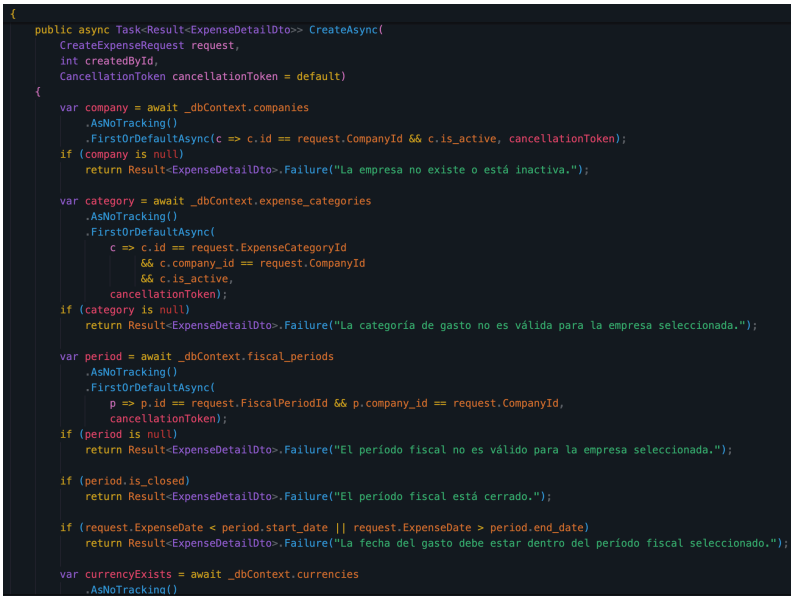


```
{
    public async Task<Result<ReceivableDetailDto>> RegisterPaymentAsync(
        int invoiceId,
        RegisterReceivablePaymentRequest request,
        int createdById,
        CancellationToken cancellationToken = default)
    {
        var invoice = await _dbContext.invoices
            .FirstOrDefaultAsync(i => i.id == invoiceId, cancellationToken);
        if (invoice is null)
            return Result<ReceivableDetailDto>.Failure("not_found");

        if (invoice.status is not ("ISSUED" or "PARTIALLY_PAID"))
            return Result<ReceivableDetailDto>.Failure("Solo se pueden registrar cobros en facturas emitidas o parcialmente pagadas.");

        if (request.Amount <= 0)
            return Result<ReceivableDetailDto>.Failure("El monto del cobro debe ser mayor a cero.");

        if (request.Amount > invoice.balance)
            return Result<ReceivableDetailDto>.Failure("El monto del cobro no puede exceder el saldo pendiente.");

        var paymentMethodId = await _dbContext.payment_methods
            .AsNoTracking()
            .Where(m => m.is_active)
            .OrderBy(m => m.id)
            .Select(m => m.id)
            .FirstOrDefaultAsync(cancellationToken);
        if (paymentMethodId == 0)
            return Result<ReceivableDetailDto>.Failure("No hay métodos de pago configurados.");

        var userExists = await _dbContext.users
            .AsNoTracking()
            .AnyAsync(u => u.id == createdById && u.is_active, cancellationToken);
        if (!userExists)
            return Result<ReceivableDetailDto>.Failure("El usuario no es válido.");

        await using var transaction = await _dbContext.Database.BeginTransactionAsync(cancellationToken);
```

Fuente: Elaboración propia, 2026.

El proceso de registro de gasto es gestionado por el método CreateAsync de la clase EfExpenseCommandService, y su edición y cambio de estado por los métodos UpdateAsync y UpdateStatusAsync de la misma clase. Tal como se muestra en la Figura 47, al recibir el comando, valida que la empresa, la categoría de gasto y el período fiscal sean válidos, que el período esté abierto y que la fecha del gasto esté dentro del rango del período fiscal. El gasto se crea en estado DRAFT, su transición a APPROVED o CANCELLED ocurre mediante UpdateStatusAsync, y respeta que solo los gastos en borrador puedan editarse o cambiar de estado.

Figura 47.
Proceso de registro de gasto.

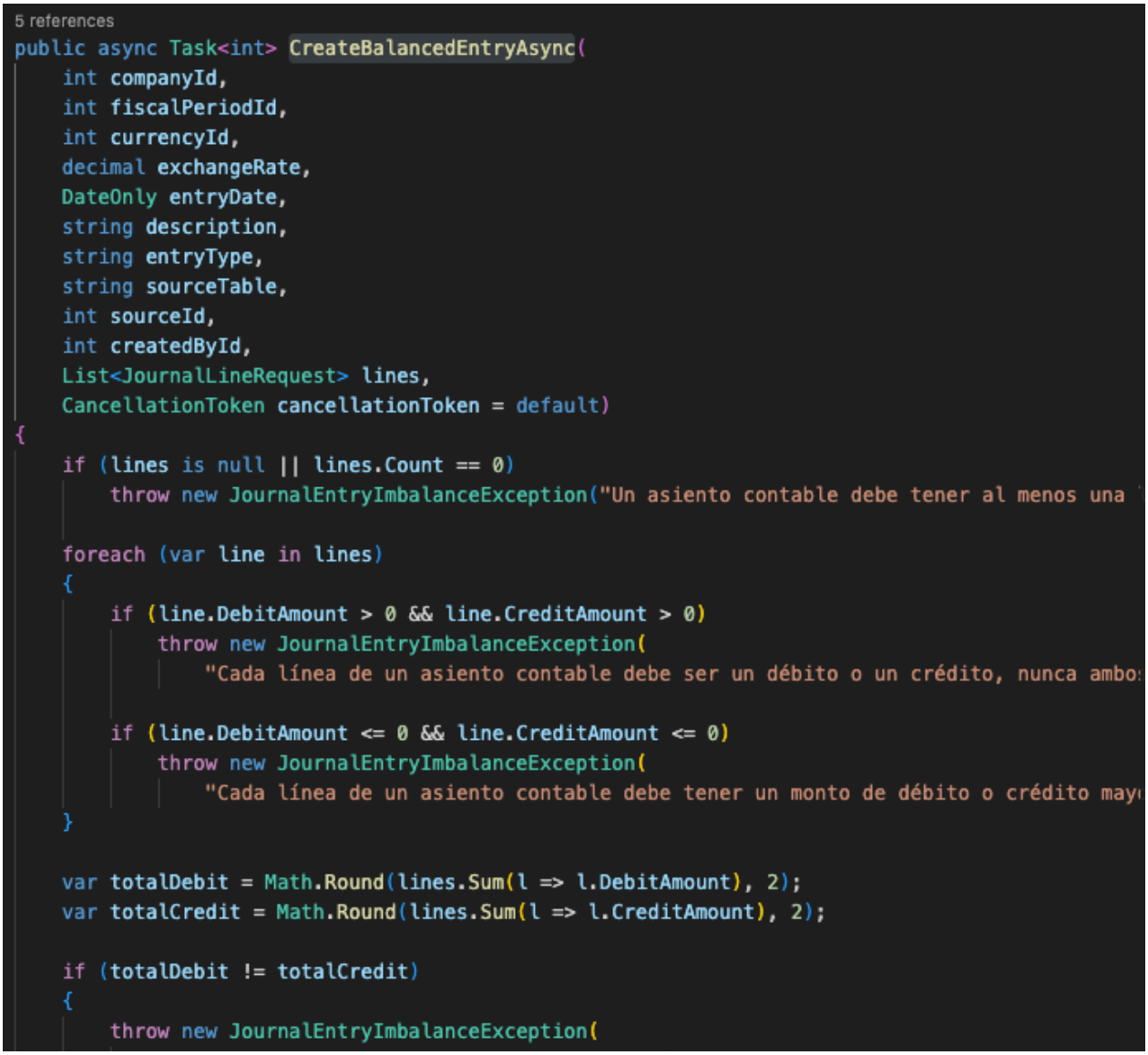


```
{
    public async Task<Result<ExpenseDetailDto>> CreateAsync(
        CreateExpenseRequest request,
        int createdById,
        CancellationToken cancellationToken = default)
    {
        var company = await _dbContext.companies
            .AsNoTracking()
            .FirstOrDefaultAsync(c => c.id == request.CompanyId && c.is_active, cancellationToken);
        if (company is null)
            return Result<ExpenseDetailDto>.Failure("La empresa no existe o está inactiva.");

        var category = await _dbContext.expense_categories
            .AsNoTracking()
            .FirstOrDefaultAsync(
                c => c.id == request.ExpenseCategoryId
                     && c.company_id == request.CompanyId
                     && c.is_active,
                cancellationToken);
        if (category is null)
            return Result<ExpenseDetailDto>.Failure("La categoría de gasto no es válida para la empresa seleccionada.");

        var period = await _dbContext.fiscal_periods
            .AsNoTracking()
            .FirstOrDefaultAsync(
                p => p.id == request.FiscalPeriodId && p.company_id == request.CompanyId,
                cancellationToken);
        if (period is null)
            return Result<ExpenseDetailDto>.Failure("El período fiscal no es válido para la empresa seleccionada.");

        if (period.is_closed)
            return Result<ExpenseDetailDto>.Failure("El período fiscal está cerrado.");

        if (request.ExpenseDate < period.start_date || request.ExpenseDate > period.end_date)
            return Result<ExpenseDetailDto>.Failure("La fecha del gasto debe estar dentro del período fiscal seleccionado.");

        var currencyExists = await _dbContext.currencies
            .AsNoTracking()
```

Fuente: Elaboración propia, 2026.

Según la Figura 48, para garantizar la integridad de la información contable generada por las operaciones del sistema, se implementa un motor central de generación de asientos contables bajo el principio de partida doble. El método CreateBalancedEntryAsync de la clase EfJournalEntryService recibe el tipo y el identificador de la operación de origen junto con una lista de líneas de asiento, valida que la suma de los montos débito sea igual a la suma de los montos crédito, y rechaza la operación mediante una excepción de dominio (JournalEntryImbalanceException), si el asiento no cuadra. El registro del encabezado (journal_entry) y sus líneas (journal_entry_line) ocurre dentro de la misma transacción de la operación que lo origina, de modo que un asiento descuadrado revierte también el cambio en el registro de origen.

Figura 48.
Código EfJournalEntryService.CreateBalancedEntryAsync.



```
5 references
public async Task<int> CreateBalancedEntryAsync(
    int companyId,
    int fiscalPeriodId,
    int currencyId,
    decimal exchangeRate,
    DateOnly entryDate,
    string description,
    string entryType,
    string sourceTable,
    int sourceId,
    int createdById,
    List<JournalLineRequest> lines,
    CancellationToken cancellationToken = default)
{
    if (lines is null || lines.Count == 0)
        throw new JournalEntryImbalanceException("Un asiento contable debe tener al menos una

    foreach (var line in lines)
    {
        if (line.DebitAmount > 0 && line.CreditAmount > 0)
            throw new JournalEntryImbalanceException(
                "Cada línea de un asiento contable debe ser un débito o un crédito, nunca ambo

        if (line.DebitAmount <= 0 && line.CreditAmount <= 0)
            throw new JournalEntryImbalanceException(
                "Cada línea de un asiento contable debe tener un monto de débito o crédito may
    }

    var totalDebit = Math.Round(lines.Sum(l => l.DebitAmount), 2);
    var totalCredit = Math.Round(lines.Sum(l => l.CreditAmount), 2);

    if (totalDebit != totalCredit)
    {
        throw new JournalEntryImbalanceException(
```

Fuente: Elaboración propia, 2026.

Este motor se integró en los cuatro puntos del sistema donde se generan efectos contables: la aprobación de un gasto (débito a la cuenta de la categoría del gasto, crédito a Bancos o a Cuentas por Pagar, según si el gasto está vinculado a una factura de proveedor existente), la emisión de una factura (débito a Cuentas por Cobrar, crédito a Ingresos y a Impuesto al Valor Agregado por Pagar), el cobro de una cuenta por cobrar (débito a Caja o Bancos, crédito a Cuentas por Cobrar) y el pago de una cuenta por pagar (débito a Cuentas por Pagar, crédito a Caja o Bancos). En los cuatro casos,

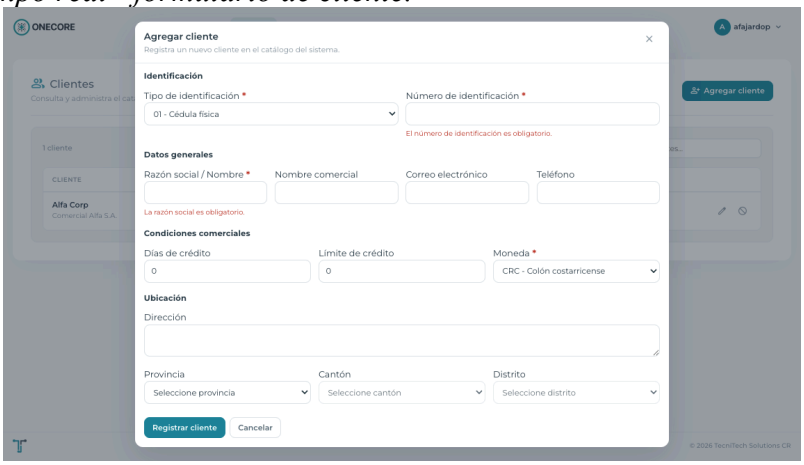
el identificador del asiento generado se almacena en el registro de origen mediante la columna journal_entry_id.

Validaciones

Las validaciones del sistema se aplican en tres niveles complementarios: validación en tiempo real en el frontend, validación de estructura y formato antes del envío, y validación de reglas de negocio en el backend; lo anterior posibilita la verificación de estados, relaciones entre entidades y condiciones del dominio antes de ejecutar una operación.

En el frontend, los formularios de creación y edición implementan validación en vivo por campo, sin necesidad de que el usuario intente enviar el formulario. Cada campo se valida al perder el foco (onBlur) y, una vez marcado como "tocado", se revalida en cada cambio de valor (onChange), esto propicia que el usuario corrija el error de forma inmediata mientras edita. Esta lógica se centraliza en la función validatePartyField, reutilizada por los formularios de Clientes y Proveedores.

Figura 49.
Validación en tiempo real - formulario de cliente.



Fuente: Elaboración propia, 2026.

El método UpdateAsync de la clase EfExpenseCommandService ilustra, en la Figura 50, la validación de reglas de negocio sobre el estado del recurso: únicamente permite modificar un gasto cuando su estado es DRAFT y comprueba adicionalmente que el período fiscal asociado no se encuentre cerrado antes de aplicar los cambios. Si alguna de estas condiciones no se cumple, el

proceso se detiene y retorna un Result de fallo con el mensaje correspondiente, sin alcanzar la capa de persistencia.

Figura 50.
Validación de reglas de negocio.

```
2 references
public async Task<Result<ExpenseDetailDto>> UpdateAsync(
    int id,
    UpdateExpenseRequest request,
    int updatedById,
    CancellationToken cancellationToken = default)
{
    var entity = await _dbContext.expenses
        .FirstOrDefaultAsync(e => e.id == id, cancellationToken);
    if (entity is null)
        return Result<ExpenseDetailDto>.Failure("not_found");

    if (entity.status != "DRAFT")
        return Result<ExpenseDetailDto>.Failure("Solo se pueden editar gastos en estado borrador.");

    var period = await _dbContext.fiscal_periods
        .AsNoTracking()
        .FirstOrDefaultAsync(p => p.id == entity.fiscal_period_id, cancellationToken);
    if (period is null)
        return Result<ExpenseDetailDto>.Failure("El período fiscal no es válido.");

    if (period.is_closed)
        return Result<ExpenseDetailDto>.Failure("El período fiscal está cerrado.");

    if (request.ExpenseDate < period.start_date || request.ExpenseDate > period.end_date)
        return Result<ExpenseDetailDto>.Failure("La fecha del gasto debe estar dentro del período fiscal seleccionado.");

    var category = await _dbContext.expense_categories
        .AsNoTracking()
        .FirstOrDefaultAsync(
            c => c.id == request.ExpenseCategoryId
                && c.company_id == entity.company_id
                && c.is_active,
            cancellationToken);
    if (category is null)
```

Fuente: Elaboración propia, 2026.

La Figura 51 muestra que el método UpdateStatusAsync de la clase EfInvoiceCommandService valida la transición de estado de una factura antes de ejecutarla: solo aprueba el cambio desde DRAFT hacia ISSUED o CANCELLED, verifica que el período fiscal siga abierto y confirma que el usuario que ejecuta la acción exista y esté activo. Cualquier transición fuera de este conjunto permitido es rechazada de forma explícita.

Figura 51.
Validación de transición de estado.

```
2 references
public async Task<Result<InvoiceDetailDto>> UpdateStatusAsync(
    int id,
    string newStatus,
    int updatedById,
    CancellationToken cancellationToken = default)
{
    var entity = await _dbContext.invoices
        .FirstOrDefaultAsync(i => i.id == id, cancellationToken);
    if (entity is null)
        return Result<InvoiceDetailDto>.Failure("not_found");

    var normalizedStatus = newStatus.Trim().ToUpperInvariant();
    if (entity.status != "DRAFT")
        return Result<InvoiceDetailDto>.Failure("Solo se puede cambiar el estado de facturas en borrador.");

    var isValidTransition = normalizedStatus switch
    {
        "CANCELLED" => true,
        "ISSUED" => true,
        _ => false
    };
    if (!isValidTransition)
        return Result<InvoiceDetailDto>.Failure("Transición de estado no válida. Solo se permite DRAFT → CANCELLED o DRAFT → ISSUE

    var period = await _dbContext.fiscal_periods
        .AsNoTracking()
        .FirstOrDefaultAsync(p => p.id == entity.fiscal_period_id, cancellationToken);
    if (period is null)
        return Result<InvoiceDetailDto>.Failure("El período fiscal no es válido.");

    if (period.is_closed)
        return Result<InvoiceDetailDto>.Failure("El período fiscal está cerrado.");
```

Fuente: Elaboración propia, 2026.

El sistema implementa un esquema de autorización granular sobre el modelo de roles y permisos ya existente en el esquema security. En la Figura 52 se constata que la clase PermissionAuthorizationHandler valida, para cada solicitud, que el usuario autenticado posea el consentimiento específico requerido por el recurso —expresado como una combinación de módulo y acción (por ejemplo, SECURITY.CREATE o EXPENSES.CHANGESTATUS)— en lugar de validar únicamente que el usuario esté autenticado. Los permisos se derivan dinámicamente como políticas de autorización mediante la clase PermissionPolicyProvider, y se aplican a los controladores por medio del atributo RequirePermission. Este mecanismo se aplicó a los quince *endpoints* de mayor sensibilidad del sistema, entre ellos la creación y modificación de usuarios, el cambio de estado de gastos, facturas y presupuestos, y el registro de pagos.

Figura 52.
Código PermissionAuthorizationHandler.

```
using Microsoft.AspNetCore.Authorization;

namespace OneCore.API.Authorization;

1 reference
public sealed class PermissionAuthorizationHandler : AuthorizationHandler<PermissionRequirement>
{
    0 references
    protected override Task HandleRequirementAsync(
        AuthorizationHandlerContext context,
        PermissionRequirement requirement)
    {
        var hasPermission = context.User.Claims.Any(c =>
            c.Type == "permissions" &&
            string.Equals(c.Value, requirement.Permission, StringComparison.OrdinalIgnoreCase));

        if (hasPermission)
            context.Succeed(requirement);

        return Task.CompletedTask;
    }
}
```

Fuente: Elaboración propia, 2026.

Módulos Señalados en el Alcance

En este apartado se presenta, para cada uno de los módulos que conforman el prototipo, evidencia de su implementación funcional. Los módulos de Facturación, Gastos y Cuentas por Cobrar ya fueron ilustrados en las secciones de Entradas, Procesos y Validaciones; por consiguiente, en esta sección, se completan los módulos restantes. Adicionalmente, se indica en qué módulo real quedó resuelta cada funcionalidad descrita en el Alcance Funcional bajo los rubros de Ingresos y Consultas, dado que no constituyen pantallas independientes, sino comportamiento transversal de otros módulos.

Módulo de Gestionar Clientes

En la Figura 53, se observa que el método CreateAsync de la clase EfClientCommandService valida la unicidad de la identificación tributaria (tax_id) antes de

registrar un nuevo cliente en el esquema master_data. Este módulo, junto con Proveedores, corresponde al rubro Mantenimientos del Alcance Funcional.

Figura 53.
Código EfClientCommandService.CreateAsync.

```
2 references
public async Task<ClientDetailDto> CreateAsync(CreateClientRequest request, CancellationToken cancellationToken = default)
{
    var entity = new client
    {
        id_type_id = request.IdTypeId,
        tax_id = TaxId.Normalize(request.TaxId),
        legal_name = request.LegalName.Trim(),
        trade_name = NormalizeOptional(request.TradeName),
        email = NormalizeOptional(request.Email),
        phone = NormalizeOptional(request.Phone),
        address = NormalizeOptional(request.Address),
        province_id = request.ProvinceId,
        canton_id = request.CantonId,
        district_id = request.DistrictId,
        credit_days = request.CreditDays,
        credit_limit = request.CreditLimit,
        currency_id = request.CurrencyId,
        is_active = true,
        created_at = DateTime.UtcNow,
        updated_at = DateTime.UtcNow
    };

    _dbContext.clients.Add(entity);
    await _dbContext.SaveChangesAsync(cancellationToken);

    return (await _clientQueryService.GetByIdAsync(entity.id, cancellationToken))!;
}
```

Fuente: Elaboración propia, 2026.

Módulo de Gestionar Proveedores

El método CreateAsync de la clase EfSupplierCommandService replica el mismo criterio de unicidad aplicado a clientes y adaptado a la entidad de proveedor, como se muestra en la Figura 54.

Figura 54.
Código EfSupplierCommandService.CreateAsync.

```
2 references
public async Task<SupplierDetailDto> CreateAsync(CreateSupplierRequest request, CancellationToken cancellationToken = default)
{
    var entity = new supplier
    {
        id_type_id = request.IdTypeId,
        tax_id = TaxId.Normalize(request.TaxId),
        legal_name = request.LegalName.Trim(),
        trade_name = NormalizeOptional(request.TradeName),
        email = NormalizeOptional(request.Email),
        phone = NormalizeOptional(request.Phone),
        address = NormalizeOptional(request.Address),
        province_id = request.ProvinceId,
        canton_id = request.CantonId,
        district_id = request.DistrictId,
        payment_days = request.PaymentDays,
        currency_id = request.CurrencyId,
        is_active = true,
        created_at = DateTime.UtcNow,
        updated_at = DateTime.UtcNow
    };

    _dbContext.suppliers.Add(entity);
    await _dbContext.SaveChangesAsync(cancellationToken);

    return (await _supplierQueryService.GetByIdAsync(entity.id, cancellationToken))!;
}
```

Fuente: Elaboración propia, 2026.

Módulo de Cuentas por Pagar

El módulo de Cuentas por Pagar gestiona tanto el registro como el pago de facturas de proveedor. De acuerdo con la Figura 55, el método CreateAsync de la clase EfPayableCommandService valida que el proveedor exista y esté activo, que el período fiscal se encuentre abierto y que la fecha de vencimiento no sea anterior a la fecha de emisión, antes de calcular el total de la factura a partir de sus líneas de detalle y generar el número interno consecutivo con el prefijo CP-{año}. El método RegisterPaymentAsync de la misma clase registra el pago de una factura de proveedor ya existente, validando que esta se encuentre en estado PENDING o PARTIALLY_PAID y que el monto no exceda el saldo pendiente, antes de actualizar el balance y el estado correspondiente.

Figura 55.
Código EfPayableCommandService.RegisterPaymentAsync.

```
2 references
public async Task<Result<PayableDetailDto>> RegisterPaymentAsync(
    int supplierInvoiceId,
    RegisterPayablePaymentRequest request,
    int createdById,
    CancellationToken cancellationToken = default)
{
    var invoice = await _dbContext.supplier_invoices
        .FirstOrDefaultAsync(i => i.id == supplierInvoiceId, cancellationToken);
    if (invoice is null)
        return Result<PayableDetailDto>.Failure("not_found");

    if (invoice.status is not ("PENDING" or "PARTIALLY_PAID"))
        return Result<PayableDetailDto>.Failure("Solo se pueden registrar pagos en facturas pendientes o parcialmente pagadas.");

    if (request.Amount <= 0)
        return Result<PayableDetailDto>.Failure("El monto del pago debe ser mayor a cero.");

    if (request.Amount > invoice.balance)
        return Result<PayableDetailDto>.Failure("El monto del pago no puede exceder el saldo pendiente.");

    var paymentMethodId = await _dbContext.payment_methods
        .AsNoTracking()
        .Where(m => m.is_active)
        .OrderBy(m => m.id)
        .Select(m => m.id)
        .FirstOrDefaultAsync(cancellationToken);
    if (paymentMethodId == 0)
        return Result<PayableDetailDto>.Failure("No hay métodos de pago configurados.");

    var userExists = await _dbContext.users
        .AsNoTracking()
        .AnyAsync(u => u.id == createdById && u.is_active, cancellationToken);
    if (!userExists)
```

Fuente: Elaboración propia, 2026.

Módulo de Plan de Cuentas

Según la Figura 56, la consulta expuesta por IAccountQueryService recupera las cuentas contables del esquema accounting junto con su tipo asociado (account_type). Este módulo corresponde al rubro Cuentas Contables del Alcance Funcional.

Figura 56.
Código IAccountQueryService (consulta de plan de cuentas).

```
2 references
public async Task<IReadOnlyList<AccountListItemDto>> GetAllAsync(
    string? search,
    CancellationToken cancellationToken = default)
{
    const string sql = """
        SELECT
            a.id              AS Id,
            a.parent_id       AS ParentId,
            a.code            AS Code,
            a.name            AS Name,
            a.level           AS Level,
            a.allows_movement AS AllowsMovement,
            a.is_active       AS IsActive,
            act.code          AS AccountTypeCode,
            act.name          AS AccountTypeName,
            act.normal_balance AS NormalBalance
        FROM accounting.account a
        INNER JOIN accounting.account_type act ON act.id = a.account_type_id
        WHERE (
            @search IS NULL OR @search = '' OR
            a.code LIKE @searchPattern OR
            a.name LIKE @searchPattern OR
            act.name LIKE @searchPattern
        )
        ORDER BY a.code;
        """;

    using var connection = _connectionFactory();
    var rows = await connection.QueryAsync<AccountListItemDto>(new CommandDefinition(
        commandText: sql,
        parameters: new
        {
            search,
            searchPattern = string.IsNullOrWhiteSpace(search) ? null : $"%{search.Trim()}%"
```

Fuente: Elaboración propia, 2026.

Módulo de Presupuestos

El método CreateAsync de la clase EfBudgetCommandService valida que la empresa esté activa y el período fiscal abierto, y calcula el monto total del presupuesto como la suma de sus líneas de detalle, de conformidad con la Figura 57.

Figura 57.
Código EfBudgetCommandService.CreateAsync.

```
2 references
public async Task<Result<BudgetDetailDto>> CreateAsync(
    CreateBudgetRequest request,
    int createdById,
    CancellationToken cancellationToken = default)
{
    if (string.IsNullOrWhiteSpace(request.Name))
        return Result<BudgetDetailDto>.Failure("El nombre del presupuesto es obligatorio.");

    if (request.Lines is null || request.Lines.Count == 0)
        return Result<BudgetDetailDto>.Failure("Debe incluir al menos una línea de presupuesto.");

    var company = await _dbContext.companies
        .AsNoTracking()
        .FirstOrDefaultAsync(c => c.id == request.CompanyId && c.is_active, cancellationToken);
    if (company is null)
        return Result<BudgetDetailDto>.Failure("La empresa no existe o está inactiva.");

    var period = await _dbContext.fiscal_periods
        .AsNoTracking()
        .FirstOrDefaultAsync(
            p => p.id == request.FiscalPeriodId && p.company_id == request.CompanyId,
            cancellationToken);
    if (period is null)
        return Result<BudgetDetailDto>.Failure("El período fiscal no es válido para la empresa seleccionada.");

    if (period.is_closed)
        return Result<BudgetDetailDto>.Failure("El período fiscal está cerrado.");

    var userExists = await _dbContext.users
        .AsNoTracking()
        .AnyAsync(u => u.id == createdById && u.is_active, cancellationToken);
    if (!userExists)
        return Result<BudgetDetailDto>.Failure("El usuario que registra el presupuesto no es válido.");
```

Fuente: Elaboración propia, 2026.

Módulo de Reportes

El servicio IReportQueryService ejemplificado en la Figura 58, a través del endpoint GET /api/reports/cash-flow, consolida los pagos de cuentas por cobrar, los pagos de cuentas por pagar y

los gastos aprobados de un período fiscal, agrupándolos por mes para generar el reporte de flujo de caja. Este módulo, junto con las operaciones GetAll ya mostradas en Salidas, resuelve el rubro Consultas del Alcance Funcional: la visualización estructurada de la información financiera se ejecuta mediante estos mismos *endpoints* de consulta, sin requerir un módulo separado.

Figura 58.
Código IReportQueryService.

```
using Dapper;
using System.Data;
using OneCore.Application.Reports;

namespace OneCore.Infrastructure.Reports;

2 references
internal sealed class DapperReportQueryService : IReportQueryService
{
    2 references
    private readonly Func<IDbConnection> _connectionFactory;

    2 references
    private static readonly string[] SpanishMonthNames =
    [
        "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
        "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"
    ];

    0 references
    public DapperReportQueryService(Func<IDbConnection> connectionFactory)
    {
        _connectionFactory = connectionFactory;
    }

    2 references
    public async Task<CashFlowReportDto> GetCashFlowAsync(
        CashFlowReportRequest request,
        CancellationToken cancellationToken = default)
    {
        const string periodSql = """
            SELECT name AS PeriodName
            FROM accounting.fiscal_period
            WHERE id = @FiscalPeriodId AND company_id = @CompanyId;
            """;

        const string inflowsSql = """
            SELECT MONTH(p.payment_date) AS Month, SUM(pi.applied_amount) AS Amount
```

Fuente: Elaboración propia, 2026.

Módulo de Administración de Usuarios

El método CreateAsync de la clase EfUserAdminCommandService, como se evidencia en la Figura 59, valida la unicidad del correo electrónico y la existencia de un rol activo, y aplica el *hash* de la contraseña mediante BCrypt antes de almacenar el nuevo usuario en el esquema security. Este módulo, junto con Autenticación, cubre el rubro Seguridad del Alcance Funcional.

Figura 59.
Código EfUserAdminCommandService.CreateAsync.

```
2 references
public async Task<Result<UserAdminDetailDto>> CreateAsync(
    CreateUserRequest request,
    CancellationToken cancellationToken = default)
{
    if (string.IsNullOrWhiteSpace(request.Email))
        return Result<UserAdminDetailDto>.Failure("El correo electrónico es obligatorio.");

    if (string.IsNullOrWhiteSpace(request.FirstName))
        return Result<UserAdminDetailDto>.Failure("El nombre es obligatorio.");

    if (string.IsNullOrWhiteSpace(request.LastName))
        return Result<UserAdminDetailDto>.Failure("El apellido es obligatorio.");

    if (string.IsNullOrWhiteSpace(request.Password) || request.Password.Length < 8)
        return Result<UserAdminDetailDto>.Failure("La contraseña debe tener al menos 8 caracteres.");

    var normalizedEmail = request.Email.Trim().ToLowerInvariant();

    var emailExists = await _dbContext.users
        .AsNoTracking()
        .AnyAsync(u => u.email.ToLower() == normalizedEmail, cancellationToken);
    if (emailExists)
        return Result<UserAdminDetailDto>.Failure(DuplicateEmail);

    var role = await _dbContext.roles
        .AsNoTracking()
        .FirstOrDefaultAsync(r => r.id == request.RoleId && r.is_active, cancellationToken);
    if (role is null)
        return Result<UserAdminDetailDto>.Failure("El rol seleccionado no es válido o está inactivo.");

    var now = DateTime.UtcNow;
    var entity = new user
    {
        email = request.Email.Trim(),
```

Fuente: Elaboración propia, 2026.

Módulo de Dashboard

La Figura 60 muestra que el servicio DapperDashboardQueryService recupera, por medio de consultas Dapper sobre múltiples esquemas, los conteos generales del sistema y los cinco gastos más recientes, para su presentación consolidada en el panel principal.

Figura 60.
Código DapperDashboardQueryService.

```
using Dapper;
using System.Data;
using OneCore.Application.Dashboard;
using OneCore.Application.Expenses;

namespace OneCore.Infrastructure.Dashboard;

2 references
internal sealed class DapperDashboardQueryService : IDashboardQueryService
{
    2 references
    private readonly Func<IDbConnection> _connectionFactory;

    0 references
    public DapperDashboardQueryService(Func<IDbConnection> connectionFactory)
    {
        _connectionFactory = connectionFactory;
    }

    2 references
    public async Task<DashboardDto> GetSummaryAsync(CancellationToken cancellationToken = default)
    {
        using var connection = _connectionFactory();

        const string countsSql = """
            SELECT
                (SELECT COUNT(1) FROM master_data.client) AS ClientsTotal,
                (SELECT COUNT(1) FROM master_data.client WHERE is_active = 1) AS ClientsActive,
                (SELECT COUNT(1) FROM master_data.supplier) AS SuppliersTotal,
                (SELECT COUNT(1) FROM master_data.supplier WHERE is_active = 1) AS SuppliersActive,
                (SELECT COUNT(1) FROM expenses.expense) AS ExpensesTotal,
                (
                    SELECT COUNT(1)
                    FROM expenses.expense
                    WHERE YEAR(expense_date) = @year
                      AND MONTH(expense_date) = @month
                ) AS ExpensesMonthCount,
                (
```

Fuente: Elaboración propia, 2026.

Módulo de Autenticación

La interacción entre ILoginQueryService y JwtTokenService valida las credenciales del usuario contra el esquema security y, en caso de ser correctas, emite un token JWT dentro de una cookie HttpOnly (como aparece en la Figura 61). En cuanto al rubro Ingresos del Alcance Funcional, en la implementación actual, el registro de ingresos no constituye un módulo independiente, sino que se resuelve a través del ciclo Facturación → Cuentas por Cobrar , donde

la emisión de una factura y el registro de su cobro generan el movimiento de ingreso correspondiente.

Figura 61.
Código ILoginQueryService / JwtTokenService.

```
using Dapper;
using System.Data;
using OneCore.Application.Auth;

namespace OneCore.Infrastructure.Auth;

2 references
internal sealed class DapperLoginQueryService : ILoginQueryService
{
    2 references
    private readonly Func<IDbConnection> _connectionFactory;

    0 references
    public DapperLoginQueryService(Func<IDbConnection> connectionFactory)
    {
        _connectionFactory = connectionFactory;
    }

    2 references
    public async Task<LoginUserQueryResult?> FindByUsernameOrEmailAsync(string usernameOrEmail, CancellationToken cancellationToken =
    {
        const string sql = """
            SELECT
                u.id                    AS UserId,
                u.role_id               AS RoleId,
                r.name                  AS RoleName,
                u.email                 AS Email,
                u.password_hash         AS PasswordHash,
                u.is_active             AS IsActive,
                m.code                  AS ModuleCode,
                a.code                  AS ActionCode
            FROM security.[user] u
            INNER JOIN security.role r
                ON r.id = u.role_id
            LEFT JOIN security.permission p
                ON p.role_id = r.id
            LEFT JOIN security.module m
                ON m.id = p.module_id
```

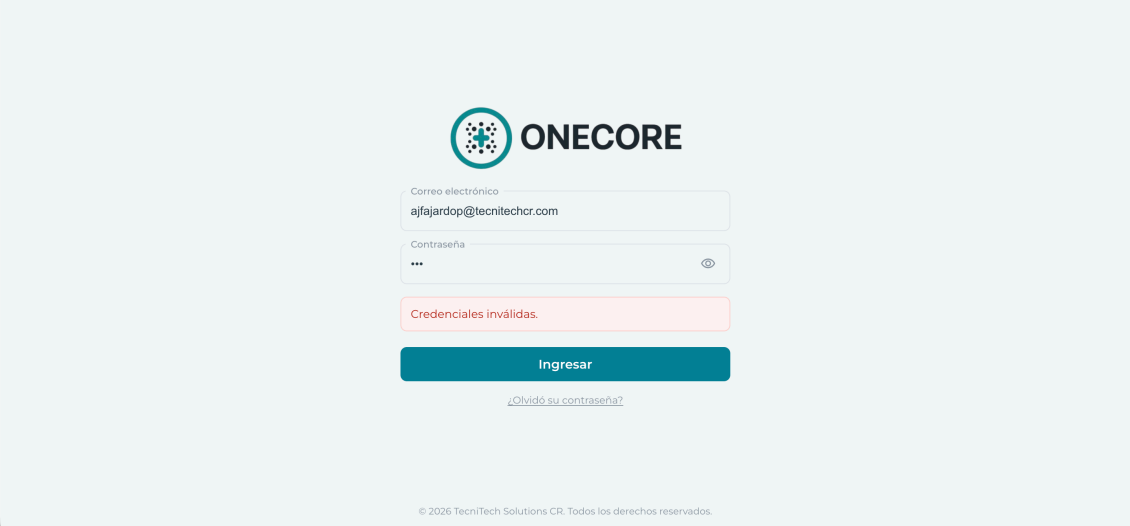
Fuente: Elaboración propia, 2026.

Pruebas

En este apartado se presenta la evidencia de las pruebas funcionales ejecutadas sobre los flujos críticos del sistema, validando que las reglas de negocio y controles de seguridad documentados en las secciones anteriores operan correctamente ante escenarios reales de uso y de error.

Tabla 30.

Caso de prueba 1: Inicio de sesión con credenciales inválidas

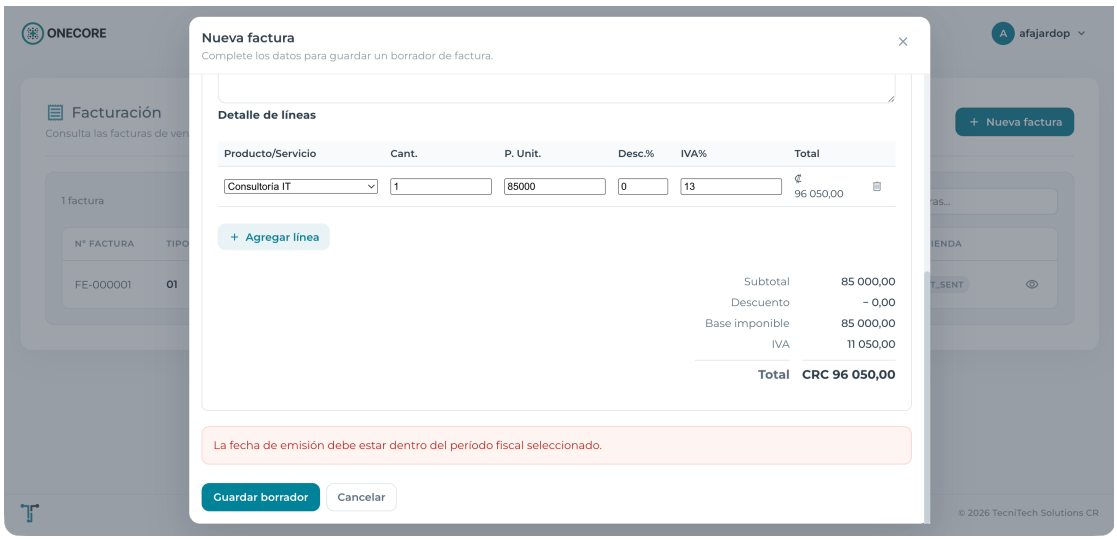
Caso de prueba 1: OneCore - Sistema web para la gestión financiero contable			
Nombre de prueba	Inicio de sesión con credenciales inválidas		
Fecha de prueba	12 de julio de 2026		
Realizada por	Adrián Fajardo		
Caso por probar	**Resultado esperado**	**Resultado obtenido**	**Estado de la prueba**
Validar el inicio de sesión al sistema, confirmando que solo los usuarios registrados con credenciales correctas puedan acceder.	Si las credenciales ingresadas no son válidas, el sistema debe rechazar el acceso y mostrar un mensaje de error indicando que las credenciales son incorrectas, sin conceder acceso al panel principal.	Al ingresar el correo ajfajardop@tecnitechcr.com con una contraseña incorrecta, el sistema mostró el mensaje "Credenciales inválidas" y mantuvo al usuario en la pantalla de inicio de sesión, sin conceder acceso.	Correcta
Evidencia del resultado obtenido			
			

Fuente: Elaboración propia, 2026.

Tabla 31.

Caso de prueba 2: Emisión de factura con período fiscal cerrado

Caso de prueba 2: OneCore - Sistema web para la gestión financiero contable			
Nombre de prueba	Emisión de factura con período fiscal cerrado		
Fecha de prueba	12 de julio de 2026		
Realizada por	Adrián Fajardo		
Caso por probar	**Resultado esperado**	**Resultado obtenido**	**Estado de la prueba**
Validar que el sistema impida la emisión de una factura cuya fecha corresponda a un período fiscal cerrado.	El sistema debe rechazar la operación mostrando un mensaje indicando que la fecha de emisión debe estar dentro del período fiscal seleccionado, sin generar el encabezado ni las líneas de la factura.	El sistema mostró el mensaje "La fecha de emisión debe estar dentro del período fiscal seleccionado" y no permitió guardar el borrador de la factura.	Correcta
Evidencia del resultado obtenido			

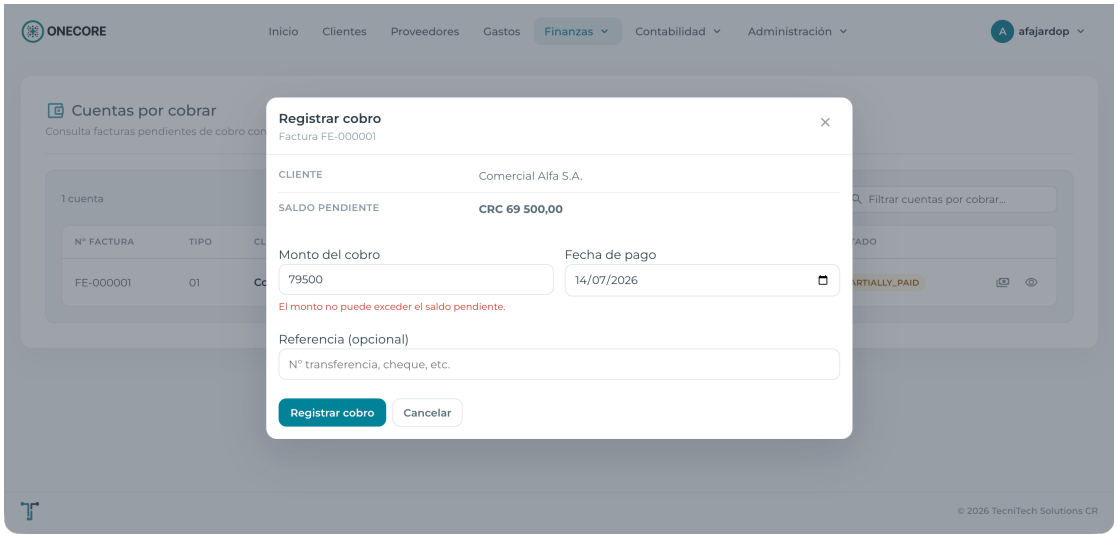


Fuente: Elaboración propia, 2026.

Tabla 32.

Caso de prueba 3: Registro de cobro con monto mayor al saldo pendiente

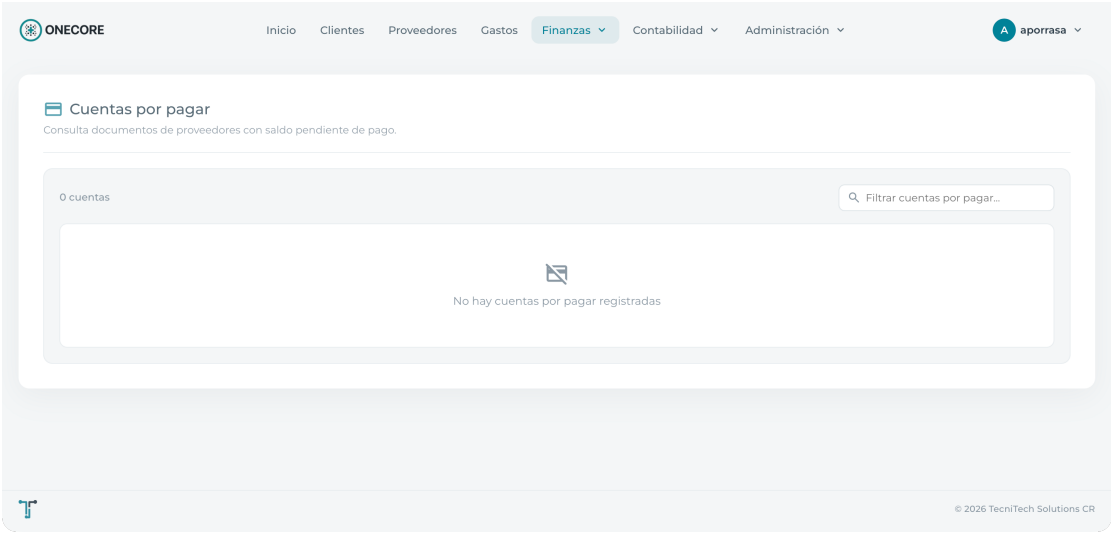
Caso de prueba 3: OneCore - Sistema web para la gestión financiero contable			
Nombre de prueba	Registro de cobro con monto mayor al saldo pendiente		
Fecha de prueba	12 de julio de 2026		
Realizada por	Adrián Fajardo		
Caso por probar	**Resultado esperado**	**Resultado obtenido**	**Estado de la prueba**
Validar que el sistema impida registrar un cobro cuyo monto exceda el saldo pendiente de la factura.	El sistema debe rechazar la operación indicando que el monto excede el saldo pendiente, sin actualizar el estado ni el balance de la factura.	Sobre la factura FE-000001 (saldo pendiente ₡69 500), al ingresar un monto de cobro de ₡79 500 el sistema mostró el mensaje "El monto no puede exceder el saldo pendiente" y no permitió registrar el cobro.	Correcta
Evidencia del resultado obtenido			



Fuente: Elaboración propia, 2026.

Tabla 33.

Caso de prueba 4: Registro de pago a proveedor con factura ya cancelada

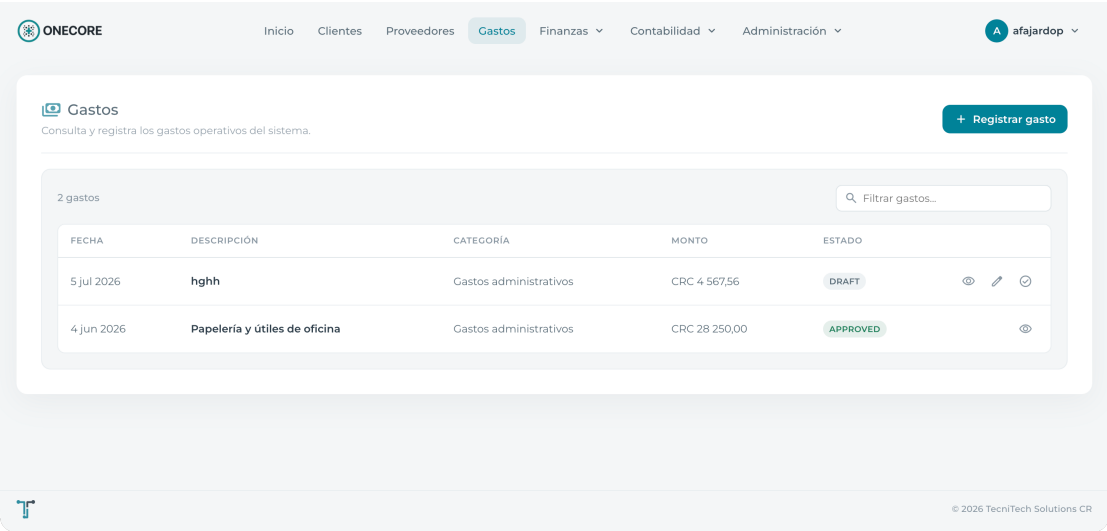
Caso de prueba 4: OneCore - Sistema web para la gestión financiero contable			
Nombre de prueba	Registro de pago a proveedor con factura ya cancelada		
Fecha de prueba	12 de julio de 2026		
Realizada por	Adrián Fajardo		
Caso por probar	**Resultado esperado**	**Resultado obtenido**	**Estado de la prueba**
Validar que el sistema impida registrar un pago sobre una factura de proveedor que ya no se encuentra en estado PENDING o PARTIALLY_PAID.	El sistema debe impedir el registro de un nuevo pago sobre una factura de proveedor ya cancelada, ya sea ocultando el documento del listado de cuentas pendientes o rechazando la operación si se intenta forzar la solicitud.	El listado de "Cuentas por pagar" únicamente muestra documentos con saldo pendiente. Una vez que una factura de proveedor queda completamente pagada, deja de aparecer en el listado, por lo que no existe una vía en la interfaz para intentar un nuevo pago sobre ella.	Correcta
Evidencia del resultado obtenido			
			

Fuente: Elaboración propia, 2026.

Tabla 34.

Caso de prueba 5: Edición de gasto en estado distinto a DRAFT

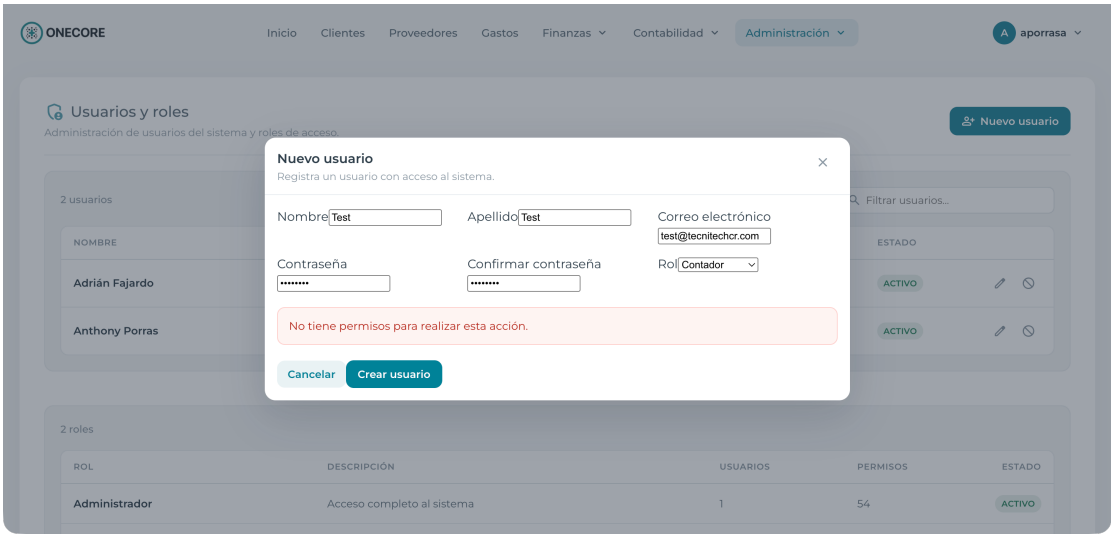
Caso de prueba 5: OneCore - Sistema web para la gestión financiero contable			
Nombre de prueba	Edición de gasto en estado distinto a DRAFT		
Fecha de prueba	12 de julio de 2026		
Realizada por	Adrián Fajardo		
Caso por probar	**Resultado esperado**	**Resultado obtenido**	**Estado de la prueba**
Validar que el sistema impida modificar un gasto que ya fue aprobado o cancelado.	El sistema debe impedir la edición de un gasto que no se encuentre en estado DRAFT, ya sea ocultando el control de edición en la interfaz o rechazando la operación en el backend si se intenta forzar la solicitud.	En el listado de gastos, el gasto "Papelería y útiles de oficina" (estado APPROVED) no muestra el ícono de edición, mostrando únicamente la opción de ver detalle. El gasto "hghh" (estado DRAFT) sí muestra las opciones de editar y aprobar.	Correcta
Evidencia del resultado obtenido			



Fuente: Elaboración propia, 2026.

Tabla 35.

Caso de prueba 6: Acceso a endpoint sensible sin el permiso requerido

Caso de prueba 6: OneCore - Sistema web para la gestión financiero contable			
Nombre de prueba	Acceso a endpoint sensible sin el permiso requerido		
Fecha de prueba	12 de julio de 2026		
Realizada por	Adrián Fajardo		
Caso por probar	**Resultado esperado**	**Resultado obtenido**	**Estado de la prueba**
Validar que un usuario autenticado, pero sin el permiso específico requerido, no pueda ejecutar una acción sensible (ej. creación de usuarios en el módulo de Administración).	El sistema debe rechazar la solicitud indicando que el usuario no posee permisos para realizar la acción, sin crear el registro.	El usuario aporrasa, autenticado, pero sin el permiso SECURITY.CREATE, intentó crear un nuevo usuario desde el módulo "Usuarios y roles". El sistema mostró el mensaje "No tiene permisos para realizar esta acción" y no completó el registro.	Correcta
Evidencia del resultado obtenido			
			

Fuente: Elaboración propia, 2026.

CAPÍTULO VII: CONCLUSIONES Y RECOMENDACIONES

Conclusiones

A lo largo del desarrollo del presente proyecto se diseñó, construyó y validó OneCore, un sistema web de gestión financiero-contable para TecniTech Solutions CR. A partir del trabajo realizado en las distintas fases del ciclo incremental, y de los resultados obtenidos durante las pruebas funcionales documentadas en el Capítulo VI, se presentan a continuación las principales conclusiones del proyecto.

El desarrollo de OneCore permitió atender de forma directa la ausencia de un sistema automatizado e integrado para la gestión financiero-contable de TecniTech Solutions CR, sustituyendo registros dispersos y procesos de facturación desconectados del control de cobros por un flujo único, trazable y validado en cada transacción.

La arquitectura Clean Architecture adoptada, con separación entre dominio, aplicación, infraestructura y presentación, y el patrón Query/Command (Dapper para lecturas, Entity Framework Core para escrituras), permitió construir un backend mantenible y escalable, capaz de absorber la complejidad contable del sistema sin acoplar la lógica de negocio a la persistencia.

La integración de un motor central de partida doble, aplicado de forma transversal en la aprobación de gastos, la emisión de facturas y el registro de cobros y pagos, garantiza que la información contable generada por el sistema sea internamente consistente, rechazando de forma automática cualquier asiento que no cuadre entre débitos y créditos.

El esquema de autorización granular basado en permisos (PermissionAuthorizationHandler / PermissionPolicyProvider), aplicado a los endpoints de mayor sensibilidad, junto con las validaciones en tres niveles (frontend, estructura y reglas de negocio), asegura que el acceso y la modificación de la información financiera estén controlados de acuerdo con el rol de cada usuario.

Las pruebas de QA realizadas durante el ciclo de desarrollo —corrección de comportamiento de modales, validación en tiempo real, ordenamiento de middleware CORS, límite de intentos de inicio de sesión y patrones anti-XSS— confirmaron que el prototipo no solo cumple visualmente con los módulos definidos en el alcance, sino que opera de forma funcional y segura de extremo a extremo.

En conjunto, el prototipo desarrollado demuestra ser una solución viable para resolver las problemáticas identificadas en el diagnóstico inicial, sentando una base técnica sólida sobre la cual TecniTech Solutions CR puede continuar ampliando la gestión financiera de la empresa.

Recomendaciones

A partir del diagnóstico realizado durante el desarrollo del proyecto y de los hallazgos documentados en los capítulos anteriores, se plantean las siguientes recomendaciones dirigidas a TecniTech Solutions CR. Estas buscan orientar la adopción, el mantenimiento y la evolución del sistema OneCore una vez finalizado el presente trabajo, atendiendo tanto aspectos técnicos de infraestructura y seguridad como aspectos operativos relacionados con su uso por parte del personal de la empresa. Se presentan ordenadas por prioridad: primero las de seguridad, seguidas por las de continuidad operativa, capacitación del personal y, finalmente, crecimiento funcional del sistema.

Restringir el acceso al App Service exclusivamente a través de Azure API Management

Actualmente el backend de OneCore puede recibir tráfico directo sin pasar por el gateway de APIM, lo que deja una superficie de ataque innecesaria expuesta. Se recomienda al personal de infraestructura de TecniTech Solutions CR configurar Access Restrictions en el App Service para aceptar únicamente el rango de IPs correspondiente a APIM, de forma que toda solicitud quede obligada a pasar por las políticas de limitación y monitoreo ya definidas en el gateway. Esta acción debe ejecutarse antes de que el sistema entre en uso productivo, ya que corresponde a una brecha de seguridad activa desde el primer día de operación; se trata de una tarea breve, de uno a dos días, que no requiere cambios en el código de la aplicación ni representa un costo adicional, dado que se utiliza infraestructura ya contratada y el único recurso involucrado es el tiempo interno de configuración.

Renovar el certificado Let's Encrypt antes de su expiración

El certificado del dominio api.tecnitechcr.com fue emitido de forma manual y expira en octubre de 2026, ya que la capa Consumption de APIM no soporta renovación automática. Se recomienda al personal técnico de TecniTech Solutions CR documentar el proceso de renovación (certbot más carga del archivo PFX) y programar un recordatorio recurrente cada 90 días, de manera que la tarea no dependa de la memoria de una sola persona. La documentación del proceso debe completarse dentro del primer mes posterior a la entrega, y la primera renovación efectiva debe ejecutarse a más tardar en septiembre de 2026, con un mes de margen antes del vencimiento real; al ser Let's Encrypt un servicio gratuito, esta recomendación no implica costo monetario alguno, más allá del tiempo que el personal técnico invierta en la renovación periódica. Un olvido

en este punto provocaría la caída del servicio para todos los usuarios del sistema, incluyendo los módulos de facturación y cobros en producción.

Definir una política de respaldo y monitoreo continuo de la base de datos

OneCore centraliza la información financiero-contable de la empresa, por lo que su continuidad operativa depende de algo más que la infraestructura de despliegue ya configurada. Se recomienda a la gerencia y al personal de infraestructura de TecniTech Solutions CR configurar respaldos automáticos verificados de la base de datos, junto con alertas de disponibilidad del servicio. Esta configuración debe quedar activa antes de que el sistema comience a almacenar información financiera real de la operación diaria, es decir, en el mismo periodo de puesta en producción; la configuración inicial tomaría entre dos y tres días, y a partir de ahí el monitoreo es continuo, pero de bajo esfuerzo operativo. En cuanto al costo, Azure SQL Database incluye respaldos automáticos sin cargo adicional hasta el tamaño de la base de datos, por lo que la configuración básica no representa un gasto nuevo; únicamente si se requiere retención extendida más allá de 35 días el costo adicional dependería del volumen de datos almacenado, y debería cotizarse en la calculadora de precios de Azure. El costo de no implementar esta recomendación, en cambio, sí es alto: cualquier pérdida de datos contables afectaría directamente la operación financiera de la empresa.

Establecer un plan de capacitación para los usuarios finales

La adopción real del sistema depende de que el personal administrativo y contable de TecniTech Solutions CR domine los flujos de facturación, cobros, pagos y aprobación de gastos. Se recomienda a las jefaturas administrativa y contable designar a un usuario clave (key user) que lidere una semana de capacitación práctica sobre el sistema, cubriendo el registro de operaciones y la interpretación de los reportes generados. Esta capacitación debe ejecutarse en la semana previa al inicio del uso productivo del sistema, una vez resueltas las recomendaciones de seguridad e infraestructura anteriores, de forma que el personal aprenda directamente sobre el entorno ya asegurado; al tratarse de tiempo interno del propio personal, no representa un desembolso monetario directo, sino un costo de oportunidad durante esa semana. Sin una capacitación formal existe el riesgo de que el personal subutilice los controles de validación ya implementados, o

busque atajos fuera del sistema que reintroduzcan los mismos problemas que OneCore fue diseñado para resolver.

Habilitar el módulo de tesorería como siguiente iteración del sistema

El diseño físico de la base de datos ya contempla un esquema treasury con soporte para movimientos bancarios y conciliación, pero esta funcionalidad no fue expuesta a nivel de aplicación ni incluida como módulo dentro del alcance del presente proyecto. Se recomienda a la gerencia de TecniTech Solutions CR y al equipo de desarrollo priorizar su implementación en una próxima iteración, siguiendo la misma metodología incremental aplicada durante este proyecto. Al tratarse de una mejora funcional y no de un riesgo de seguridad o continuidad, esta recomendación se ubica al final de la lista de prioridades: se sugiere planificarla para el primer trimestre posterior a la puesta en producción del sistema, una vez que el personal esté familiarizado con los módulos actuales. Formalizar este módulo permitiría cerrar el ciclo de conciliación entre las cuentas por cobrar y por pagar con los movimientos bancarios reales de la empresa, eliminando la dependencia de procesos manuales para esa validación; se estima que su implementación tomaría entre uno y dos sprints de desarrollo interno, sin costo de licenciamiento adicional, dado que el modelo de datos ya está definido y la arquitectura ya está en su lugar.

REFERENCIAS

Nota: Las entradas correspondientes a Ley 8968 de 2011 y Decreto 37554 de 2012 se citan por su año de promulgación oficial, dado que corresponden a normativa vigente en Costa Rica y no a fuentes académicas sujetas al criterio de actualidad editorial.

Bolaños, Y., Lazo, Y. y Ruiz, R. (2021). *Propuesta de implementación de un sistema contable y financiero para la Asociación Solidarista de Empleados de Costa Rica* (Tesis de Licenciatura). https://repositorio.utn.ac.cr/items/f02e42d8-27e0-4ba3-92ff-be1c11b6add5

Chinchilla, D. (2020). *¿Cuál sería el sistema de información contable disponible en Costa Rica que permita la adecuada gestión financiero-contable para MangoTropic?* (Tesis de Licenciatura). https://repositoriotec.tec.ac.cr/handle/2238/12424

Decreto 37554 de 2012. [Poder Ejecutivo de la República de Costa Rica]. Reglamento a la Ley de Protección de la Persona frente al Tratamiento de sus Datos Personales. 5 de marzo de 2013. *La Gaceta* No. 45. https://pgrweb.go.cr/scij/Busqueda/Normativa/Normas/nrm_texto_completo.aspx?nValor1=1&nValor2=76084

Hernández-Sampieri, R. y Mendoza, C. (2023). Metodología de la investigación: Las rutas cuantitativa, cualitativa y mixta (2.ª ed.). McGraw-Hill Interamericana.

Ley 8968 de 2011. Ley de Protección de la Persona Frente al Tratamiento de sus Datos Personales. 7 de julio de 2011. *La Gaceta* No. 170. https://pgrweb.go.cr/scij/Busqueda/Normativa/Normas/nrm_texto_completo.aspx?nValor1=1&nValor2=70975

Ministerio de Economía, Industria y Comercio, Dirección General de Apoyo a la Pequeña y Mediana Empresa. (2021). *Estado de situación de la PYME en Costa Rica 2021. Serie 2015-2019*. https://taxescr.com/wp-content/uploads/2022/07/DIGEPYME-INF-038-2021.pdf

Patiño, A., Cabello, S., Ros, D. y Urtasun, M. (2025). Transformación digital productiva: Análisis de políticas, estrategias e instrumentos para impulsarla en América Latina (LC/TS.2025/97). Comisión Económica para América Latina y el Caribe (CEPAL).

https://repositorio.cepal.org/server/api/core/bitstreams/678e47d8-cc9a-4691-8f38-eca4b74e0bb3/content

Schwaber, K. y Sutherland, J. (2020). *The Scrum Guide: The definitive guide to Scrum: The rules of the game*. Scrum Guides. https://scrumguides.org/scrum-guide.html

Apéndice A

UNIVERSIDAD INTERNACIONAL DE LAS AMÉRICAS
ESCUELA DE INGENIERÍA INFORMÁTICA
PROYECTO FINAL DE GRADUACIÓN

GUÍA DE ENTREVISTA

Entidad:	
Nombre del entrevistado:	
Puesto del entrevistado:	
Nombre del estudiante:	
Fecha de la entrevista:	
Lugar o medio de la entrevista:	

Preguntas:

Contexto del negocio

¿Qué tipo de servicios ofrece actualmente la empresa?
¿Qué tipos de clientes atiende principalmente la empresa?
¿De qué forma se realizan actualmente los cobros a los clientes?
¿Qué tipo de información financiera considera importante registrar para el control del negocio?

Facturación e ingresos

¿Cómo se realiza actualmente el proceso de facturación a los clientes?
¿Qué datos se registran normalmente en una factura?
¿Se utiliza algún control de consecutivos o numeración para las facturas?
¿Cómo se manejan actualmente los impuestos, descuentos u otros cargos en las facturas?

Cuentas por cobrar

¿Cómo se lleva actualmente el control de los pagos pendientes de los clientes?
¿Se registran pagos parciales o abonos de los clientes?
¿Cómo se identifican las facturas vencidas o clientes morosos?
¿Qué tipo de seguimiento se realiza para recuperar pagos pendientes?

Proveedores y cuentas por pagar
¿Cómo se registran actualmente las facturas de proveedores?
¿Cómo se controla el pago de esas facturas?
¿Se registran fechas de vencimiento para las cuentas por pagar?
¿Se realizan controles para evitar pagos duplicados o errores en los registros?

Registro de gastos
¿Cómo se registran actualmente los gastos de la empresa?
¿Existen categorías de gastos definidas?
¿Se utilizan centros de costo o algún tipo de clasificación adicional?
¿Existen controles o límites para ciertos tipos de gastos?

Conciliación y control financiero
¿Cómo se verifica que los registros financieros coincidan con la información real del negocio?
¿Se realizan conciliaciones de ingresos o gastos periódicamente?
¿Cómo se detectan errores o diferencias en los registros financieros?
¿Se realiza algún proceso de cierre financiero mensual?

Presupuesto y planificación
¿La empresa realiza algún tipo de planificación financiera o presupuesto?
¿Cómo se comparan los resultados reales con lo planificado?
¿Se utilizan alertas o controles cuando los gastos superan lo esperado?

Seguridad y control de acceso
¿Quiénes deberían tener acceso al sistema financiero?
¿Qué tipo de acciones deberían poder realizar los distintos usuarios (ver, registrar, modificar información)?
¿Considera importante llevar un registro de las acciones realizadas dentro del sistema?

Reportes y consultas
¿Qué tipo de reportes financieros considera necesarios para el control del negocio?
¿Con qué frecuencia se deberían generar estos reportes?
¿Qué nivel de detalle debería tener la información mostrada en los reportes?

Prioridades del sistema
¿Qué funcionalidades considera más importantes para el sistema que se desarrolla?
¿Qué módulo o funcionalidad debería implementarse primero?
¿Qué criterios utilizaría para considerar que el sistema funciona correctamente?

Apéndice B

UNIVERSIDAD INTERNACIONAL DE LAS AMÉRICAS
ESCUELA DE INGENIERÍA INFORMÁTICA
PROYECTO FINAL DE GRADUACIÓN

GUÍA DE OBSERVACIÓN

Entidad:	
Dirección física de la entidad:	
Fecha de la actividad de observación:	
Nombre del estudiante:	

Tabla de control de aspectos observados:

No	Aspectos por Observar	Cumple	No Cumple	Oportunidad de Mejora	Detalle de Observación
1	Las facturas emitidas contienen número consecutivo				
2	Las facturas incluyen datos completos del cliente				
3	Las facturas registran fecha de emisión				

No	Aspectos por Observar	Cumple	No Cumple	Oportunidad de Mejora	Detalle de Observación
4	Las facturas detallan los servicios prestados				
5	Las facturas incluyen subtotal, impuestos y total				
6	Existe registro organizado de facturas emitidas				
7	Se registran los pagos recibidos de los clientes				
8	Se identifican claramente las facturas pendientes de cobro				
9	Existe registro de facturas de proveedores				
10	Se registran las fechas de vencimiento de cuentas por pagar				
11	Los gastos se registran con algún tipo de categoría				

No	Aspectos por Observar	Cumple	No Cumple	Oportunidad de Mejora	Detalle de Observación
12	Se cuenta con comprobantes de respaldo para los gastos				
13	Existen registros de movimientos bancarios				
14	Se realizan conciliaciones entre registros y movimientos bancarios				
15	Existe algún tipo de control o resumen financiero del negocio				