UNIVERSIDAD SANTO TOMAS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
T U N J A

**TÍTULO PASANTÍA**

Integración en InvuPos con Ecosistemas Externos: Contabilidad, Delivery y Gestión de Personal


**PROPONENTE(S)**

Jesús Esteban Blanco Sandoval

1051064099

2527220

**DIRECTOR**

Jhon Alexis Piracoca Arcos

**CODIRECTOR**

Martha Susana Contreras Ortiz


Tunja

Fecha de presentación (01, 06, 2026)

# CONTENIDO

1. FICHA TÉCNICA DE LA PASANTÍA ..... 5
2. PLANTEAMIENTO DEL PROBLEMA ..... 8
3. JUSTIFICACIÓN ..... 10
4. OBJETIVOS ..... 11
4.1. OBJETIVO GENERAL ..... 11
4.2. OBJETIVOS ESPECÍFICOS ..... 11
5. DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS ..... 13
5.1. ACTIVIDADES REALIZADAS EN EL PROYECTO ..... 13
6. DESARROLLO DE LOS PROYECTOS ..... 17
6.1. CONTEXTO GENERAL DE LA INTEGRACIÓN CON RAPPI Y ALCANCE DEL DESARROLLO . 17
6.2. CONTEXTO GENERAL DE LA INTEGRACIÓN CON PEDIDOSYA Y ALCANCE DEL DESARROLLO ..... 19
6.3. CONTEXTO GENERAL DE LA INTEGRACIÓN CON SIIGO Y ALCANCE DEL DESARROLLO .. 20
6.4. CONTEXTO GENERAL DE LA INTEGRACIÓN CON QUICKBOOKS Y ALCANCE DEL DESARROLLO ..... 21
6.5. ARQUITECTURA FUNCIONAL DE LAS INTEGRACIONES: COMPONENTES, MÓDULOS Y FLUJO DE INFORMACIÓN ..... 22
6.5.1. COMPONENTES DE LA ARQUITECTURA ..... 23
6.5.2. MÓDULOS DE INTEGRACIÓN INDEPENDIENTES ..... 24
6.6. ARQUITECTURA TÉCNICA Y ESTRUCTURA DEL PROYECTO ..... 31
6.6.1. ESTRUCTURA GENERAL DE LAS INTEGRACIONES ..... 31
6.6.2. PARTICULARIDADES POR TIPO DE INTEGRACIÓN ..... 34
6.6.3. FLUJO TÉCNICO DE FUNCIONAMIENTO ..... 38
6.7. DESARROLLO SEGÚN OBJETIVOS ..... 39
6.7.1. ACTIVIDADES RELACIONADAS CON OBJETIVO ESPECÍFICO 1 ..... 39
6.7.2. ACTIVIDADES RELACIONADAS CON OBJETIVO ESPECÍFICO 2 ..... 42

6.7.3. ACTIVIDADES RELACIONADAS CON OBJETIVO ESPECÍFICO 3 ..... 48
6.7.4. ACTIVIDADES RELACIONADAS CON OBJETIVO ESPECÍFICO 4 ..... 71
6.7.5. ACTIVIDADES EXTRA ..... 83
**6.8. HERRAMIENTAS UTILIZADAS EN EL DESARROLLO DEL PROYECTO ..... 84**

**7. CONCLUSIONES ..... 87**

**8. ANEXOS ..... 90**

**8.1. ANEXO A. REQUERIMIENTOS INTEGRACIÓN RAPPI ..... 90**
**8.2. ANEXO B – REQUERIMIENTOS INTEGRACIÓN PEDIDOSYA ..... 112**
**8.3. ANEXO C – REQUERIMIENTOS INTEGRACIÓN SIIGO ..... 131**
**8.4. ANEXO D – REQUERIMIENTOS INTEGRACIÓN QUICKBOOKS ..... 141**

**9. REFERENCIAS ..... 176**

## Tabla de figuras

Figura 1. Árbol de problemas ..... 9
Figura 2. Proceso de levantamiento de requerimientos y análisis técnico ..... 41
Figura 3. Diseño general de la arquitectura ..... 43
Figura 4. Interfaces de comunicación y seguridad ..... 46
Figura 5. Modelo de datos y persistencia local ..... 47
Figura 6. herramientas backend utilizadas ..... 49
Figura 7. Modelo de datos de la integración con Rappi ..... 51

Figura 8. Modelo de datos de la integración con PedidosYa ..... 52
Figura 9. Modelo de datos de la integración con Siigo ..... 53
Figura 10. Modelo de datos de la integración con QuickBooks ..... 54
Figura 11. arquitectura lógica y estructura de módulos de integración ..... 56
Figura 12. Vista de configuración de la integración con Rappi ..... 57
Figura 13. Configuración de tiendas hijas en Rappi ..... 57
Figura 14. Documento de términos y condiciones previo al uso de la integración ..... 58
Figura 15. Flujo Integración Rappi ..... 58
Figura 16. Vista de configuración de la integración con pedidosYa ..... 59
Figura 17. Previsualización y aprobación de la importación de catálogo en PedidosYa ..... 60
Figura 18. Documento de términos y condiciones previo al uso de la integración ..... 60
Figura 19. Gestión de horarios y categorías en PedidosYa ..... 61
Figura 20. Flujo de la integración con PedidosYa ..... 61
Figura 21. Vista de journals en Siigo ..... 62
Figura 22 Vista Facturas Siigo ..... 63
Figura 23. Vista de órdenes de compra en Siigo ..... 63
Figura 24. Mapeo de proveedores Siigo ..... 64
Figura 25. Mapeo de impuestos Siigo ..... 64
Figura 26. flujo integración con Siigo ..... 65
Figura 27. Vista principal de la integración con QuickBooks ..... 66
Figura 28. Vista de journals en QuickBooks ..... 67
Figura 29. Vista de facturas en QuickBooks ..... 67
Figura 30. Vista de órdenes de compra en QuickBooks ..... 68
Figura 31. Vista de gestión de entidades en QuickBooks ..... 68
Figura 32. Gestión de entidades auxiliares en QuickBooks ..... 69
Figura 33. flujo integración QuickBooks ..... 70

# 1. FICHA TÉCNICA DE LA PASANTÍA

| | |
|---|---|
| Título | Desarrollo integración InvuPos |
| Nombre Estudiante | Jesús Esteban Blanco Sandoval |
| Documento estudiante | 1051064099 |
| Correo electrónico | Jesus.blanco@usantoto.edu.co |

| | |
|---|---|
| Director | Alon Horovitz |
| Entidad o sector de la empresa | InvuPos |
| Lugar de ejecución de la pasantía | Tunja, Boyacá |
| Duración | 6 meses |

Los abajo firmantes confirman que todos los datos incluidos en la presente propuesta son correctos y verídicos, que no incumplen ninguna ley o norma vigente. (Incluir nombres y firmas de estudiantes, director y tutor).

Firma del autor 1

Jesús Esteban Blanco Sandoval

Nombre autor 1

| |
|---|
| Firma del director<br><br>Nombre director de la pasantía |
| |

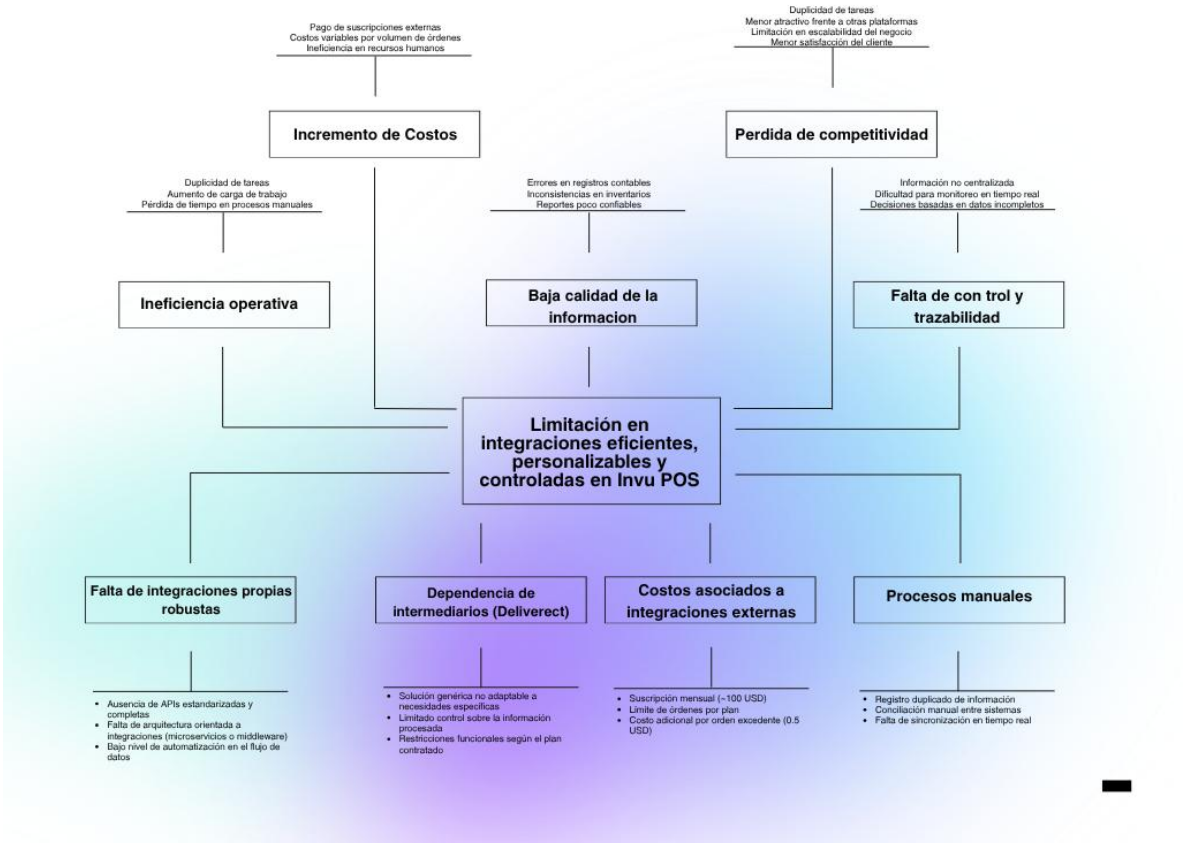
# 2. PLANTEAMIENTO DEL PROBLEMA

INVU POS se ha consolidado como una plataforma integral para la gestión de ventas, inventarios y facturación. Sin embargo, presenta una limitación importante en el área de integraciones con sistemas externos utilizados por sus clientes, especialmente plataformas de delivery y sistemas contables. En muchos casos, la información generada en la operación diaria no fluye de forma automática, precisa y controlada hacia herramientas como Rappi, PedidosYa, Siigo y QuickBooks, lo que obliga a depender de intermediarios o de procesos manuales para registrar, transformar y conciliar los datos.

Esta situación genera problemas relevantes en el funcionamiento de los negocios que utilizan la plataforma. Por una parte, incrementa la carga operativa, ya que los usuarios deben duplicar tareas en distintos sistemas. Por otra, favorece la aparición de errores e inconsistencias en la información, afectando la confiabilidad de pedidos, ventas y registros contables. A ello se suma la falta de trazabilidad en tiempo real, lo que limita el control administrativo y dificulta la toma de decisiones oportunas.

La ausencia de integraciones propias también representa una desventaja competitiva para la empresa. El uso de soluciones intermediarias reduce la flexibilidad para adaptarse a requerimientos particulares de los clientes, incrementa costos operativos y limita el control técnico sobre la información intercambiada. En consecuencia, se hace necesario desarrollar integraciones backend que permitan automatizar el flujo de datos entre INVU POS y plataformas externas, fortaleciendo la interoperabilidad, reduciendo errores y mejorando la eficiencia operativa del sistema.

Figura 1. Árbol de problemas




Fuente: Autor

## 3. Justificación

La presente pasantía se justifica por la necesidad de fortalecer la capacidad de INVUPOS para integrarse de forma directa con plataformas externas utilizadas por sus clientes. En particular, el desarrollo de integraciones con Rappi, PedidosYa, Siigo y QuickBooks permitió abordar procesos que anteriormente dependías de intermediarios o de registros manuales, como la recepción de pedidos, la actualización de catálogos y la sincronización de información contable. De esta manera, el trabajo realizado no solo aporta valor técnico a la empresa, sino que también responde a una necesidad operática concreta dentro de su ecosistema de software

Desde la perspectiva empresarial, el proyecto contribuye a mejorar la propuesta de valor del sistema, al facilitar la conexión con plataformas externas de delivery y contabilidad, reduciendo la dependencia de terceros y disminuyendo la necesidad de procesos manuales. Esto favorece el control de la información, optimiza tiempos de operación y fortalece la confiabilidad de los procesos administrativos y comerciales.

Desde la perspectiva académica y profesional, la pasantía permite aplicar conocimientos de desarrollo backend, arquitectura de software, bases de datos, consumo de APIs, seguridad y validación de sistemas en un contexto real. Asimismo, fortalece competencias en análisis técnico, diseño de soluciones interoperables y resolución de problemas asociados a la integración de plataformas heterogéneas. En este sentido, la pasantía no solo resulta pertinente para la empresa, sino también valiosa para la formación profesional del estudiante.

# 4. OBJETIVOS

## 4.1. Objetivo general

Desarrollar, durante el periodo de la pasantía empresarial, un ecosistema de integraciones backend entre INVU POS y plataformas externas de delivery y contabilidad, mediante el levantamiento de requerimientos, el diseño arquitectónico, la implementación de módulos funcionales y su validación técnica, con el fin de automatizar procesos, optimizar la interoperabilidad y mejorar la eficiencia operativa del sistema.

## 4.2. Objetivos específicos

1. Definir las especificaciones técnicas de las integraciones con Rappi, PedidosYa, Siigo y QuickBooks, mediante el levantamiento de requerimientos funcionales y no funcionales y el análisis de sus APIs externas, con el fin de elaborar un documento formal que sirva como base para el desarrollo.

2. Diseñar la arquitectura de las integraciones y sus interfaces de comunicación, mediante la elaboración de artefactos técnicos, modelos de datos y diagramas de soporte, incorporando principios de diseño modular, seguridad e interoperabilidad, con el fin de garantizar una solución escalable y mantenible.

3. Construir los módulos backend de integración con plataformas externas, mediante la implementación de servicios funcionales, procesos de transformación de datos y mecanismos de automatización, con el fin de disponer de integraciones operativas validadas en un entorno de pruebas.

4. Validar la funcionalidad, el rendimiento y la seguridad de las integraciones desarrolladas, mediante la ejecución de pruebas funcionales, de integración, usabilidad y rendimiento, incluyendo pruebas automatizadas cuando aplique, con el fin de verificar su calidad y documentar los resultados obtenidos.

# 5. DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS

## 5.1. Actividades realizadas en el proyecto

Se debe relacionar el conjunto de actividades que realizó en el proyecto estableciendo fechas de los productos entregados.

**Tabla 1. Actividades**

| Descripción de la actividad | Fecha inicio | Fecha entrega | Producto entregado |
|---|---|---|---|
| capacitación | 01/11/2025 | 07/11/2025 | Comprensión de las actividades realizadas dentro de la empresa, entrega de un menú hecho por el mismo capacitado |
| Extensión editar Marketman | 09/11/2025 | 01/12/2025 | Vista para edición de datos en marketman |
| Extensión editar Deliverect | 04/12/2025 | 08/01/2026 | Vista para edición de datos en Deliverect |
| INTEGRACION PEDIDOSYA | | | |
| Requerimientos de pedidos ya | 09/01/2026 | 13/01/2026 | Comprensión de lo necesario para la integración |

| Descripción de la actividad | Fecha inicio | Fecha entrega | Producto entregado |
|---|---|---|---|
| Lectura ya análisis documentación api Pedidos Ya | 09/01/2026 | 15/01/2026 | Listado de api con sus requisitos para usar |
| Creación base de datos | 15/01/2026 | 20/01/2026 | Base de datos |
| Creación código backend | 20/01/2026 | 30/01/2026 | Código back funcional |
| Creación código frontend | 01/02/2026 | 12/02/2026 | Código front conectado al back y funcional |
| Realización de pruebas | 12/02/2026 | 14/02/2026 | Falencias de código y aspectos a mejorar |
| Correcciones | 12/02/2026 | 15/02/2026 | Correcciones del punto anterior corregidas y realizadas |
| INTEGRACION SIIGO | | | |
| Requerimientos de Siigo | 16/02/2026 | 17/02/2026 | Comprensión de lo necesario para la integración |
| Lectura y análisis documentación api Siigo | 16/02/2026 | 17/02/2026 | Listado de api con sus requisitos para usar |
| Creación de base de datos | 17/02/2026 | 19/02/2026 | Base de datos |

| Descripción de la actividad | Fecha inicio | Fecha entrega | Producto entregado |
|---|---|---|---|
| Creación código backend | 19/02/2026 | 28/02/2026 | Código back funcional |
| Creación código frontend | 28/02/2026 | 06/03/2026 | Código front conectado al back y funcional |
| Realización de pruebas | 06/03/2026 | 08/03/2026 | Falencias de código y aspectos a mejorar |
| Correcciones | 06/03/2026 | 10/03/2026 | Correcciones del punto anterior corregidas y realizadas |
| INTEGRACION RAPPI | | | |
| Requerimientos de Rappi | 12/03/2026 | 12/03/2026 | Comprensión de lo necesario para la integración |
| Lectura y análisis documentación api Rappi | 13/03/2026 | 15/03/2026 | Listado de api con sus requisitos para usar |
| Creación de base de datos | 14/03/2026 | 15/03/2026 | Base de datos |
| Creación código backend | 15/03/2026 | 25/03/2026 | Código back funcional |

| Descripción de la actividad | Fecha inicio | Fecha entrega | Producto entregado |
|---|---|---|---|
| Creación código frontend | 25/03/2026 | 05/04/2026 | Código front conectado al back y funcional |
| Realización de pruebas | 05/04/2026 | 10/04/2026 | Falencias de código y aspectos a mejorar |
| Correcciones | 05/04/2026 | 10/04/2026 | Correcciones del punto anterior corregidas y realizadas |
| INTEGRACION QUICKBOOKS | | | |
| Requerimientos de QuickBooks | 12/04/2026 | 12/04/2026 | Comprensión de lo necesario para la integración |
| Lectura y análisis documentación api QuickBooks | 12/04/2026 | 13/04/2026 | Listado de api con sus requisitos para usar |
| Creación de base de datos | 13/04/2026 | 14/04/2026 | Base de datos |
| Creación código backend | 14/04/2026 | 20/04/2026 | Código back funcional |
| Creación código frontend | 20/04/2026 | 25/04/2026 | Código front conectado al back y funcional |

| Descripción de la actividad | Fecha inicio | Fecha entrega | Producto entregado |
|---|---|---|---|
| Realización de pruebas | 26/04/2026 | 30/04/2026 | Falencias de código y aspectos a mejorar |
| Correcciones | 26/04/2026 | 30/04/2026 | Correcciones del punto anterior corregidas y realizadas |

# 6. DESARROLLO DE LOS PROYECTOS

El desarrollo de los proyectos se llevó a cabo de manera progresiva y organizada, abordando cada uno de los objetivos planteados al inicio de la pasantía. A lo largo del proceso se integraron actividades técnicas, pruebas funcionales y ajustes continuos, lo que permitió un avance constante y garantizó que cada implementación estuviera alineada con las necesidades tanto de los clientes como de la empresa. Este enfoque iterativo facilitó la validación temprana de resultados y la mejora continua de las soluciones desarrolladas.

## 6.1. Contexto general de la integración con Rappi y alcance del desarrollo

Rappi es una plataforma que permite la intermediación tecnológica para conectar a usuarios, comercios locales y repartidores independientes. Aunque comenzó como una aplicación de entregas a domicilio, actualmente se ha consolidado como un ecosistema

digital que ofrece una amplia gama de servicios, incluyendo la gestión de pedidos en línea para restaurantes y comercios. (Rappi, s. f.)

Dentro del contexto de Invu POS, Rappi representa un canal externo de generación de pedidos que deben ser gestionados de manera eficiente dentro del sistema. Sin embargo, en el escenario actual, la integración entre ambas plataformas no se realiza de forma directa, lo que obliga a los clientes a depender de intermediarios como Deliverect o hacer uso del sistema Rappi partes el cual gestiona únicamente los pedidos entrantes por la plataforma y puede afectar o complicar la contabilidad del establecimiento

Esta situación genera problemas como la duplicidad de información, errores en el registro de pedidos y falta de sincronización en tiempo real entre ambas plataformas. Por esta razón, surge la necesidad de desarrollar una integración que permita automatizar el flujo de información entre Rappi e Invu POS, garantizando mayor precisión, rapidez y control de los datos.

El alcance del desarrollo realizado en esta pasantía se enfoca en la implementación de una integración que permita la recepción automática de pedidos generados en Rappi dentro de Invu POS, así como la gestión de la información asociada a cada orden, incluyendo productos, cantidades, precios y datos del cliente. Asimismo, se contempla la actualización del estado de los pedidos, con el fin de mantener sincronizada la información entre ambas plataformas.

Es importante destacar que este desarrollo se limita a la integración a nivel de gestión de pedidos y no contempla la administración de otros servicios ofrecidos por la plataforma Rappi, enfocándose específicamente en mejorar la eficiencia operativa y el control de la información dentro del sistema POS.

## 6.2. Contexto general de la integración con PedidosYa y alcance del desarrollo

PedidosYa es una plataforma de delivery que permite a los usuarios realizar pedidos en línea a diferentes comercios, especialmente restaurantes, facilitando la gestión de órdenes y su posterior entrega. A través de su infraestructura tecnológica, actúa como un canal externo de ventas que genera un alto volumen de pedidos diarios para los negocios afiliados. (PedidosYa Developers, s. f.)

Dentro del contexto de Invu POS, PedidosYa representa una fuente adicional de generación de pedidos que deben ser registrados y gestionados dentro del sistema. No obstante, en el escenario actual, la integración entre ambas plataformas no se realiza de manera directa, lo que obliga a los usuarios a depender de intermediarios o hacer uso del sistema de recepción de ordenes exclusivo de pedidosYa

Esta situación genera problemáticas similares a las identificadas en otras plataformas de delivery, tales como la duplicidad de tareas, errores en el registro de la información y falta de sincronización en tiempo real. Además, se dificulta el control centralizado de las operaciones, lo que impacta negativamente en la eficiencia del negocio.

En este contexto, el desarrollo realizado se enfoca en la implementación de una integración que permita la recepción automática de los pedidos generados en PedidosYa dentro de Invu POS, incluyendo el detalle de productos, cantidades, precios y datos relevantes de la orden. Asimismo, se contempla la gestión del estado de los pedidos, con el objetivo de mantener la coherencia de la información entre ambas plataformas.

El alcance de esta integración se limita a la gestión de pedidos provenientes de la plataforma, sin contemplar otras funcionalidades adicionales, priorizando la automatización de procesos y la mejora en el control de la información.

### 6.3. Contexto general de la integración con Siigo y alcance del desarrollo

Siigo es un software contable que permite a las empresas gestionar sus procesos financieros, incluyendo facturación, contabilidad, impuestos y reportes financieros. Es ampliamente utilizado por pequeñas y medianas empresas para garantizar el cumplimiento de obligaciones fiscales y el control de la información contable. (Siigo API, s. f.)

En el caso de Invu POS, la integración con Siigo es fundamental para asegurar que la información generada en las operaciones de venta se refleje correctamente en el sistema contable. Sin embargo, en el estado actual, esta integración no se encuentra completamente automatizada, lo que obliga a los usuarios a realizar procesos manuales de registro y conciliación de la información.

Como consecuencia, se presentan errores en los registros contables, inconsistencias en los datos y retrasos en la generación de reportes financieros, lo que afecta la confiabilidad de la información y la toma de decisiones.

En este sentido, el desarrollo llevado a cabo se enfoca en la implementación de una integración que permita la transmisión automática de la información de ventas desde Invu POS hacia Siigo, incluyendo datos relevantes como clientes, productos, valores de las transacciones e impuestos. De esta manera, se busca garantizar la consistencia y exactitud de la información contable.

El alcance de esta integración está orientado a la sincronización de información financiera derivada de las ventas, sin incluir otros procesos contables avanzados, con el objetivo de mejorar la eficiencia operativa y reducir errores en el registro de datos.

## 6.4. Contexto general de la integración con QuickBooks y alcance del desarrollo

QuickBooks es un software de contabilidad ampliamente utilizado a nivel internacional, que permite a las empresas gestionar sus finanzas, controlar ingresos y egresos, generar reportes y mantener organizada su información contable. Su uso es común en empresas que requieren soluciones contables robustas con alcance global. (Intuit Developer, s. f.)

Dentro del contexto de Invu POS, la integración con QuickBooks representa una necesidad clave para aquellos clientes que utilizan esta herramienta como su sistema principal de

gestión financiera. Sin embargo, al igual que en otros casos, la ausencia de una integración directa obliga a los usuarios a realizar procesos manuales para transferir la información de ventas, lo que incrementa la carga operativa y el riesgo de errores.

Esta falta de automatización afecta la precisión de los registros contables y limita la disponibilidad de información actualizada para la toma de decisiones, generando ineficiencias en los procesos administrativos.

El desarrollo realizado en el marco de esta pasantía se enfoca en la creación de una integración que permita el envío automático de la información de ventas desde Invu POS hacia QuickBooks, incluyendo detalles como ingresos, clientes, productos y valores asociados a cada transacción. Esto permite mantener sincronizados ambos sistemas y mejorar la calidad de la información financiera.

El alcance de esta integración se centra en la sincronización de datos contables básicos relacionados con las ventas, sin abarcar funcionalidades más avanzadas del sistema, priorizando la automatización y la reducción de errores en los procesos.

## 6.5. Arquitectura funcional de las integraciones: componentes, módulos y flujo de información

En el contexto de las integraciones desarrolladas para InvuPos, la arquitectura funcional del sistema no se centra únicamente en la gestión de usuarios, sino en la organización de los componentes que permiten la comunicación entre la plataforma y los sistemas externos. Esta

Arquitectura está compuesta por módulos de integraciones, servicios de procesamiento y flujos de datos que garantizan la sincronización de la información

El funcionamiento general se basa en la recepción, transformación y envió de datos entre invu POS y plataformas externas como Rappi, PedidosYa, Siigo y QuickBooks. Para ello, se implementan mecanismos que permiten automatizar los procesos y asegurar la consistencia de la información en cada sistema.

Los módulos definidos dentro de esta arquitectura agrupan las funcionalidades principales de integración, mientras que los submódulos corresponden a los procesos específicos que se ejecutan en cada flujo de datos, tales como la recepción de pedidos, validación de información, transformación de datos y envío hacia los sistemas correspondientes.

### 6.5.1. Componentes de la arquitectura

Dentro de la solución desarrollada se identifican los siguientes componentes principales

#### 6.5.1.1 Invu POS

Es el sistema central donde se gestionan las operaciones del negocio, incluyendo ventas, inventarios y facturación. Actúa como el núcleo de la integración, ya que recibe y envía información hacia los sistemas externos

#### 6.5.1.2 APIs de integración

Incluyen las plataformas con las cuales se realiza la integración:

Plataformas de delivery: Rappi y PedidosYa

Sistemas contables: Siigo y QuickBooks

Cada uno de estos sistemas cumple un rol especifico dentro del flujo de información.

### 6.5.2. Módulos de integración independientes

La solución desarrollada se compone de múltiples módulos de integración, cada uno correspondiente a un sistema externo especifico. Estos módulos funcionan de manera independiente y cuentan con sus propios componentes internos

#### 6.5.2.1 Integración con Rappi

El módulo que integra Invu POS con Rappi permite la sincronización bidireccional de información, facilitando tanto la publicación del menú desde Invu POS hacia la plataforma de Rappi, como la recepción y gestión de pedidos generados en esta. Este módulo se compone de cinco submódulos: Recepción de pedidos, Procesamiento de datos, Registro en Invu POS, Configuración de parámetros previos a la integración y Módulo de firma previa a uso, cada uno con funcionalidades específicas que se describen a continuación.

**Tabla 2. Funcionamiento Integración Rappi**

| Submódulo | Funcionamiento |
|---|---|
| Recepción de Pedidos | Recibe las órdenes generadas en Rappi mediante el uso de Webhooks, incluyendo sus actualizaciones de estado (creada, confirmada, cancelada, programada, entre |

| Submódulo | Funcionamiento |
|---|---|
| | otras), permitiendo mantener sincronizada la información en tiempo real. |
| Procesamiento de datos | Se encarga de transformar y adaptar la información entre ambos sistemas. Incluye la conversión de estructuras de menús enviados a Rappi, así como la normalización de los datos de pedidos recibidos para su correcta interpretación dentro de Invu POS. |
| Registro en Invu POS | Permite registrar automáticamente los pedidos recibidos desde Rappi dentro del sistema Invu POS, asegurando que la información de productos, cantidades, precios y datos del cliente quede correctamente almacenada en la base de datos del sistema. |
| Módulo de firma previa a uso | Generar, consultar y administrar enlaces de revisión y firma de reportes de menú y registra la aprobación del cliente sobre el menú mediante sus datos, firma en pantalla y generación de un PDF como soporte |

| Submódulo | Funcionamiento |
|---|---|
| Configuración parámetros previos a integración | Permite definir los parámetros necesarios para el correcto funcionamiento de la integración, tales como tipo de orden, método de pago, productos que no se enviaran a Rappi y otras variables necesarias para adaptar la integración a cada cliente |

#### 6.5.2.2 Integración con PedidosYa

El módulo de integración con PedidosYa permite la sincronización bidireccional de información entre la plataforma y Invu POS, facilitando tanto la gestión de pedidos como la administración del menú desde el sistema. A través de esta integración, es posible centralizar la operación del negocio, evitando la gestión manual directamente en la plataforma externa, los submódulos se explican a continuación en la siguiente tabla

**Tabla 3. Funcionamiento Integración PedidosYa**

| Submódulo | Funcionamiento |
|---|---|
| **Recepción de pedidos** | Recibe las órdenes generadas en PedidosYa mediante Webhooks o endpoints configurados, incluyendo cambios de estado como confirmación, cancelación o entrega, permitiendo la actualización en tiempo real dentro de Invu POS. |
| **Procesamiento de datos** | Transforma la información tanto de pedidos como de menús entre ambos sistemas, asegurando compatibilidad en estructuras de productos, precios, categorías y configuraciones. |
| **Registro en Invu POS** | Permite registrar automáticamente los pedidos recibidos dentro del sistema, integrándolos al flujo operativo del negocio. |
| **Gestión de disponibilidad de ítems** | Permite activar o desactivar productos en tiempo real desde Invu POS, reflejando los cambios directamente en la plataforma de PedidosYa. |
| **Gestión de horarios** | |

| Submódulo | Funcionamiento |
|---|---|
| | Permite definir y asignar horarios de disponibilidad para productos o categorías, facilitando el control de la oferta según franjas horarias. |
| **Configuración de parámetros previos a la integración** | Permite definir identificadores del comercio, mapeo de productos, categorías, credenciales y demás configuraciones necesarias para el correcto funcionamiento de la integración. |

#### 6.5.2.3 Integración con Siigo

El módulo de integración con Siigo permite la sincronización de la información contable generada en Invu POS hacia el sistema Contable, Automatizando el registro de ventas y permitiendo la creación y mapeado de entidades para un correcto traspaso de información

Los submódulos son explicados en la tabla submódulos integración Siigo

**Tabla 4. Submódulos integración Siigo**

| Submódulo | Funcionamiento |
|---|---|
| **Extracción de datos de ventas** | Obtiene la información de las transacciones generadas en Invu POS, incluyendo |

| Submódulo | Funcionamiento |
|---|---|
| | productos, valores, impuestos y datos del cliente. |
| **Procesamiento de datos contables** | Transforma la información al formato requerido por Siigo, adaptando estructuras como cuentas contables, impuestos y terceros. |
| **Envió de información a Siigo** | Realiza el envío de los comprobantes o registros contables al sistema Siigo mediante su API, asegurando la correcta creación de documentos. |
| **Validación de registros** | Verifica que la información haya sido registrada correctamente en Siigo, controlando errores y evitando duplicidad de registros. |
| **Configuración de parámetros** | Permite definir cuentas contables, tipos de comprobantes, impuestos y demás configuraciones necesarias para adaptar la integración a cada cliente. |

#### 6.5.2.4 Integración con QuickBooks

El módulo de integración con QuickBooks permite la sincronización de la información financiera de Invu POS con este sistema contable, considerando estructuras más orientadas a estándares internacionales, el funcionamiento de los submódulos es explicada continuación en la tabla Submódulo integración QuickBooks

**Tabla 5. Submódulo integración QuickBooks**

| Submódulo | Funcionamiento |
|---|---|
| **Extracción de datos de ventas** | Obtiene la información de ventas desde Invu POS, incluyendo ingresos, clientes y detalles de cada transacción. |
| **Procesamiento de datos financieros** | Adapta la información al modelo de QuickBooks, incluyendo cuentas, categorías, impuestos y formatos requeridos por la plataforma. |
| **Envío de información a QuickBooks** | Realiza la creación automática de registros financieros (facturas, ingresos o recibos) mediante la API de QuickBooks. |
| **Sincronización de datos** | Permite mantener actualizada la información entre ambos sistemas, evitando inconsistencias y duplicidad de registros |
| **Configuración de parámetros de integración** | Define credenciales, cuentas contables, monedas y configuraciones específicas |

| | |
|---|---|
| | necesarias para la correcta conexión con QuickBooks. |

## 6.6. Arquitectura técnica y estructura del proyecto

El desarrollo de las integraciones para invu POS se implemento bajo un enfoque de arquitectura modular desacoplada, en el cual cada integración con sistemas externos se construyo como un proyecto independiente. Esta estructura permite aislar la lógica de cada plataforma(Rappi, PedidosYa ,Siigo y QuickBooks), facilitando su mantenimiento escalabilidad y evolución sin afectar el funcionamiento de las demás integraciones

Cada módulo de interacción cuenta con su propia configuración, servicios, rutas, controladores y base de datos, lo que permite adaptar la lógica de negocio según los requerimientos específicos de cada sistema externo

A continuación, se describe la estructura técnica general de los proyectos desarrollados

### 6.6.1. Estructura general de las integraciones

Cada integración sigue una estructura basada en Node.js, organizada en capas funcionales que permiten separar responsabilidades dentro del sistema. Aunque existen variaciones según la plataforma, la estructura general incluye los siguientes componentes

- **Archivos principales**
    - **Server.js/app.js**

- Punto de entrada de la aplicación. Se encarga de levantar el servidor, configurar middlewares y registrar rutas.

- **config.js/. env**
  - Contiene la configuración del sistema, incluyendo credenciales, variables de entorno, claves de APIs y parámetros de integración.
- **Package.json**
  - Define las dependencias del proyecto y scripts de ejecución

- **Carpeta services**
  - Contiene la lógica de negocio y la comunicación con sistemas externos, algunos ejemplos de estos son los siguientes:
    - **Db.js**. permite el manejo de base de datos de cada una de las integraciones
    - **Invu.js** Comunicación con Invu POS
    - **Rappi.js/pedidosYa.js/siigo.js/qb.js** integración con cada plataforma
    - **menuSignature.js** Generación de firmas de seguridad
    - **menuPdfReport.js** Generación de reportes
    - **Crypto.js** encriptación y desencriptación de tokens de acceso que son almacenados en base de datos

- **Carpeta src**

  Organiza la lógica interna de la aplicación.

  Subcomponentes

  - **Controller**

        - Manejan la lógica de entrada de datos, validaciones y flujo de la aplicación
    - **Webhooks (en Rappi)**
        - Gestionan la recepción de eventos externos (pedidos, cambios de estado)
    - **Temp/public**
        - Manejo de archivos temporales, imágenes y reportes generados dinámicamente

- **Carpeta routes**
    - Define los endpoints disponibles del sistema.
        - Ejemplos
            - Gestión de clientes
            - Configuración de horarios
            - Firma de menús
            - Endpoints de integración

- **Carpeta views**
    - Contiene las vistas renderizadas (ETA templates), utilizadas para:
        - Visualización de reportes
        - Simulación de menús
        - Firma de documentos
        - Panales administrativos

- **Carpeta public**

Incluye recursos estáticos como archivos CSS, imágenes y scripts del lado del cliente

### 6.6.2. Particularidades por tipo de integración

Debido a las diferencias entre plataformas, cada integración presenta características específicas:

- **Integraciones con plataformas de delivery (Rappi y PedidosYa)**
    - Uso de webhooks para recepción de pedidos
    - Manejo de menús dinámicos
    - Generación de firmas digitales para validación
    - Soporte para:
        - Activación/desactivación de productos
        - Gestión de horarios
        - Sincronización de categorías
- Integraciones contables (Siigo y QuickBooks)
    - Uso de procesos automáticos (cron Jobs) para envió de información
    - Transformación de datos a estructuras contables
    - Manejo de clientes:
        - Clientes
        - Facturas/comprobantes
        - Impuestos
    - Archivos clave:
        - Cron.js, cronOrders.js, cronSalesReceipt.js

#### 6.6.2.1 Diferencias de cada integración

Las integraciones desarrolladas presentan diferencias importantes asociadas al tipo de plataforma con la que se conectan y al objetivo funcional que cumplen dentro del ecosistema del sistema. En el caso de Rappi y PedidosYa, las integraciones están orientadas principalmente a la operación comercial, incluyendo la publicación de menús, la recepción de pedidos y la actualización de estados en tiempo real. Por su parte, las integraciones con Siigo y QuickBooks están enfocadas en la automatización de procesos contables, mediante la transformación y envío de información proveniente de INVU hacia sistemas externos de gestión financiera. Estas particularidades implicaron el uso de distintos mecanismos de autenticación, sincronización, validación y tratamiento de datos en cada caso.

A continuación, se presenta una comparación de los principales aspectos diferenciales entre las integraciones desarrolladas:

**Tabla 6. Comparaciones entre integraciones realizadas**

| Aspecto | Rappi | PedidosYa | Siigo | QuickBooks |
|---|---|---|---|---|
| **Propósito principal** | Publicación de menú, recepción de pedidos y control operativo de tienda | Publicación de catálogo, recepción de pedidos y gestión operativa del ciclo del pedido | Automatización de procesos contables y administrativos desde INVU hacia Siigo | Automatización de procesos contables desde INVU hacia QuickBooks Online |

| Aspecto | Rappi | PedidosYa | Siigo | QuickBooks |
|---|---|---|---|---|
| **Tipo de sistema** | Plataforma de delivery | Plataforma de delivery | Sistema contable nacional en la nube | Sistema contable internacional en la nube |
| **Dirección del flujo** | Bidireccional | Bidireccional | Unidireccional desde INVU hacia Siigo | Unidireccional desde INVU hacia QuickBooks |
| **Mecanismo de autenticación** | Credenciales client_id y client_secret según ambiente o región | Inicio de sesión con username y password, con generación de token de acceso | Credenciales username y access_key, con token reutilizable | OAuth 2.0 con access_token, refresh_token y realm_id |
| **Mecanismo principal de integración** | Webhooks en tiempo real | Callbacks y consumo de API | Consultas API y procesos automáticos por cron | API, OAuth 2.0 y procesos automáticos por cron |

| Aspecto | Rappi | PedidosYa | Siigo | QuickBooks |
|---|---|---|---|---|
| **Recepción de información** | Webhooks firmados con validación de firma HMAC | Webhooks y Callbacks para pedidos, estados y carga de catálogo | Consulta programada de información contable y comercial | Consulta programada de información contable y comercial |
| **Datos enviados desde INVU** | Menú, disponibilidad de tienda, disponibilidad de productos y respuesta operativa de pedidos | Catálogo, disponibilidad, estados del pedido y modificaciones operativas | Journals, facturas, órdenes de compra, clientes y productos | Journal entries, purchase orders, sales receipts, customers, vendors e items |
| **Gestión de menú** | Si | Si | No | No |
| **Gestión de pedidos** | Si | Si | No | No |
| **Alcance multi-sede** | Soporta relación entre tiendas padre e hijas | Soporta carga centralizada para múltiples vendors | Opera por sucursal | Opera por sucursal |

| Aspecto | Rappi | PedidosYa | Siigo | QuickBooks |
|---|---|---|---|---|
| | | | o external_location_id | o external_location_id |
| **Diferencia funcional más relevante** | Integración altamente orientada a eventos en tiempo real y validación segura de webhooks | Integración orientada a catálogo y operación del pedido, incluyendo horarios y modificaciones posteriores | Integración centrada en reglas contables, mapeos e idempotencia documental | Integración centrada en autenticación avanzada, sincronización contable y validaciones por tipo de cuenta |

### 6.6.3. Flujo técnico de funcionamiento

El flujo técnico general de las integraciones se estructura en una serie de etapas que permiten la recepción, transformación, envío y control de la información entre el sistema origen y las plataformas externas, bajo un modelo de comunicación apoyado en principios REST (Fielding, 2000):

1. Recepción o generación de datos (pedido o venta)
2. Procesamiento en servicios internos
3. Transformación de datos según la plataforma destino
4. Comunicación mediante API externa

5. Registro en base de datos
6. Validación y control de errores

## 6.7. Desarrollo según objetivos

Este apartado presenta el trabajo realizado durante la pasantía con relación a cada uno de los objetivos planteados. En esta fase se desarrolló un ecosistema de integraciones backend orientado a automatizar procesos operativos y contables mediante la conexión con plataformas externas como Rappi, PedidosYa, Siigo y QuickBooks. A lo largo del proceso se abordaron actividades de análisis técnico, diseño arquitectónico, desarrollo de microservicios y validación funcional, con el fin de construir una solución interoperable, modular y orientada a mejorar la eficiencia operativa.

### 6.7.1. Actividades relacionadas con objetivo específico 1

El primer objetivo estuvo orientado a definir las especificaciones técnicas de las integraciones a partir del levantamiento de requerimientos y del análisis de las APIs externas involucradas. Para ello, se revisaron los mecanismos de autenticación, los modelos de datos, los flujos de comunicación y las capacidades funcionales de cada plataforma, estableciendo así una base técnica para el desarrollo posterior.

#### 6.7.1.1 Estado inicial del proceso de análisis

Al inicio de la pasantía se identificó la necesidad de integrar cuatro plataformas externas con comportamientos y propósitos diferentes. Rappi y PedidosYa correspondían a plataformas

de delivery enfocadas en la gestión de menús, disponibilidad y pedidos, mientras que Siigo y QuickBooks se orientaban a la automatización de procesos contables como journals, facturas, órdenes de compra y comprobantes. Esta diferencia funcional hizo necesario estudiar cada API de forma individual, debido a que cada una presentaba particularidades en autenticación, estructura de datos, flujos de actualización y validaciones.

#### 6.7.1.2 Proceso de levantamiento de requerimientos y análisis técnico

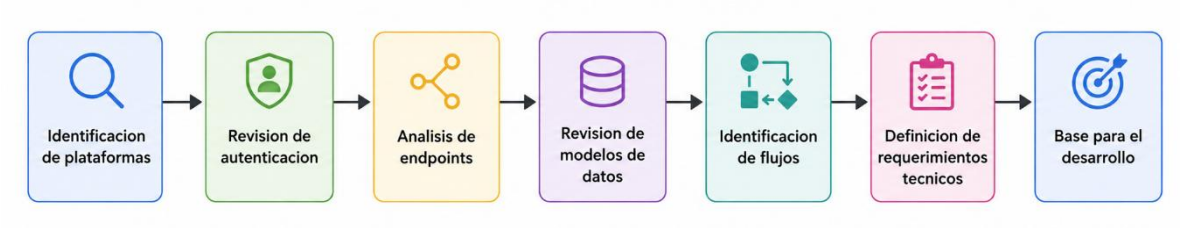
El proceso de definición técnica se desarrolló de forma iterativa. En una primera etapa se analizaron los endpoints disponibles de cada integración, identificando qué operaciones debían ejecutarse desde INVU hacia la plataforma externa y cuáles eventos debían ser recibidos desde dichas plataformas. Posteriormente se documentaron los mecanismos de autenticación requeridos para cada caso: autenticación por credenciales de cliente en Rappi, autenticación mediante usuario y contraseña en PedidosYa, autenticación por usuario y llave de acceso en Siigo y autenticación OAuth 2.0 en QuickBooks (Rappi, s. f.; PedidosYa Developers, s. f.; Siigo API, s. f.; Intuit Developer, s. f.).

En una segunda etapa se revisaron los modelos de datos intercambiados entre sistemas. Este análisis permitió reconocer diferencias importantes entre los catálogos de menú, los pedidos, las estructuras contables y las entidades auxiliares como clientes, proveedores, cuentas, impuestos, métodos de pago y tiendas. A partir de ello se definieron transformaciones

específicas para adaptar la información proveniente de INVU a los formatos exigidos por cada plataforma(ver Figura **2**).

Finalmente, se identificaron los flujos de comunicación necesarios para cada integración. En Rappi se establecieron eventos en tiempo real mediante webhooks para aprobación de menú, rechazo, recepción de pedidos, cancelaciones, tracking y conectividad de tienda. En PedidosYa se identificaron Callbacks para pedidos, cambios de estado, modificación de productos y carga de catálogo. En Siigo y QuickBooks se determinaron flujos basados principalmente en procesos automáticos programados, orientados a sincronizar información contable desde INVU hacia dichos sistemas.

Figura 2. Proceso de levantamiento de requerimientos y análisis técnico




### 6.7.1.3 Hallazgos principales del análisis

El análisis técnico permitió evidenciar que las integraciones de delivery y las contables exigían enfoques muy distintos. En Rappi se encontró una integración fuertemente orientada a eventos en tiempo real y validación de seguridad sobre los webhooks. En PedidosYa se identificó una operación centrada en catálogo, Callback de pedidos, horarios por categoría y gestión de disponibilidad. En Siigo se reconoció una lógica contable basada en mapeos

configurables, control de documentos, auto creación de productos y validación de idempotencia. En QuickBooks se identificó una mayor complejidad en la autenticación, dado que su funcionamiento depende de OAuth 2.0, manejo de realm_id, renovación de tokens y validaciones específicas según el tipo de cuenta contable utilizada (Hardt, 2012; Intuit Developer, s. f.).

#### 6.7.1.4 Resultados obtenidos

Como resultado de esta fase se consolidó una base técnica clara para el desarrollo de las integraciones, permitiendo definir cómo debían construirse los servicios, qué entidades debían persistirse localmente y qué validaciones eran necesarias antes de intercambiar información con plataformas externas. Esta etapa sirvió además como fundamento para la construcción de documentación técnica parcial dentro de los proyectos, reflejada en archivos de guía, reportes y descripciones funcionales asociados a cada integración.

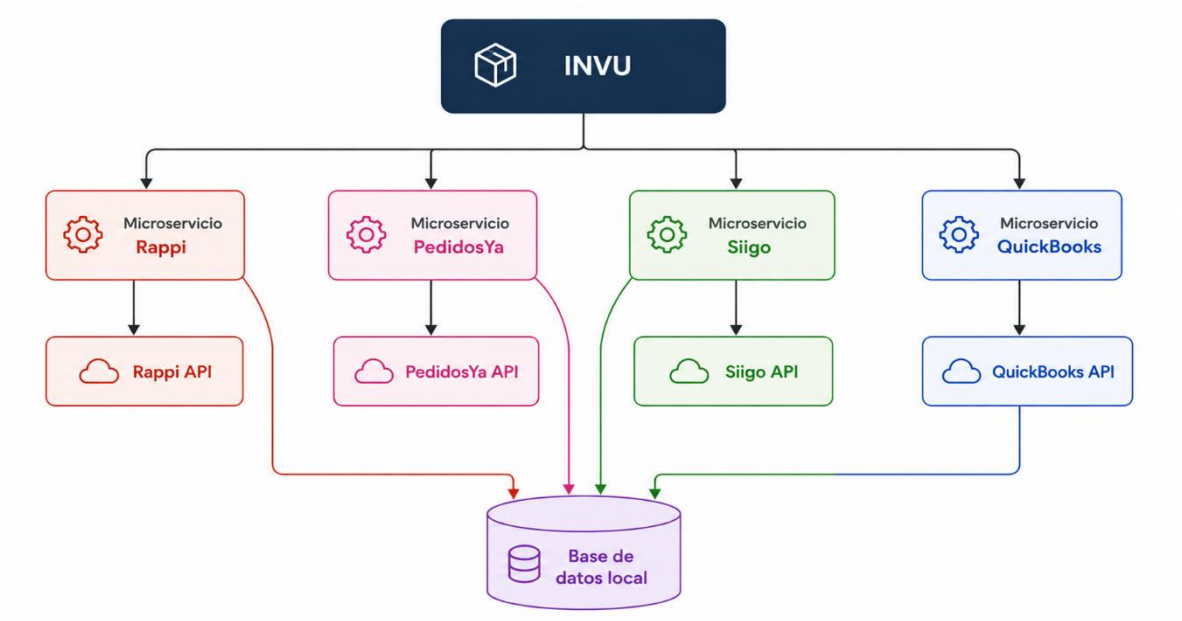
### 6.7.2. Actividades relacionadas con objetivo específico 2

El segundo objetivo se enfocó en diseñar la arquitectura de microservicios y las interfaces de comunicación, incorporando criterios de modularidad, seguridad y escalabilidad. Para ello se estructuraron servicios independientes para cada plataforma, organizando la lógica de negocio, las rutas, los procesos automáticos y la persistencia de datos de manera separada.

#### 6.7.2.1 Diseño general de la arquitectura

La solución fue organizada en servicios diferenciados para Rappi, PedidosYa, Siigo y QuickBooks, cada uno con su propio punto de entrada, dependencias, rutas y servicios internos (ver Figura 3). Esta separación permitió aislar la lógica particular de cada integración y reducir el acoplamiento entre procesos de delivery y procesos contables. A nivel interno, la mayoría de los servicios siguieron una estructura compuesta por rutas, controladores, servicios auxiliares, acceso a base de datos y vistas administrativas o de soporte cuando eran necesarias.

Figura 3. Diseño general de la arquitectura



#### 6.7.2.2 Interfaces de comunicación y seguridad

En el diseño de interfaces se emplearon mecanismos HTTP y API REST para la comunicación con sistemas externos, así como webhooks y Callbacks para eventos en tiempo real. En materia de seguridad se implementaron distintos controles según la naturaleza de cada integración. En Rappi se preservó el raw Body del webhook para validar firmas HMAC. En QuickBooks se utilizó OAuth 2.0, cifrado de tokens almacenados y protección CSRF en la interfaz web. En Siigo se incorporó cache de autenticación, control de tasa de peticiones e idempotencia para evitar documentos duplicados. Adicionalmente, en los procesos programados de Siigo y QuickBooks se implementaron mecanismos de bloqueo mediante cron_locks para prevenir ejecuciones simultáneas.

#### 6.7.2.3 Criterios arquitectónicos aplicados

Durante el diseño de la arquitectura de las integraciones se tuvieron en cuenta criterios técnicos orientados a garantizar que la solución fuera modular, segura, interoperable, escalable y mantenible. Estos conceptos fueron aplicados de manera practica en la organización de los servicios, la separación de responsabilidades, la comunicación con plataformas externas y la persistencia de configuraciones y registros operativos.

La **modularidad** se reflejó en la separación de cada integración como un servicio independiente, permitiendo que Rappi, PedidosYa, Siigo y QuickBooks conservaran su propia lógica, rutas, configuraciones y procesos internos. Esto facilito el desarrollo individual de cada modulo y redujo el acoplamiento entre integraciones de delivery y contabilidad.

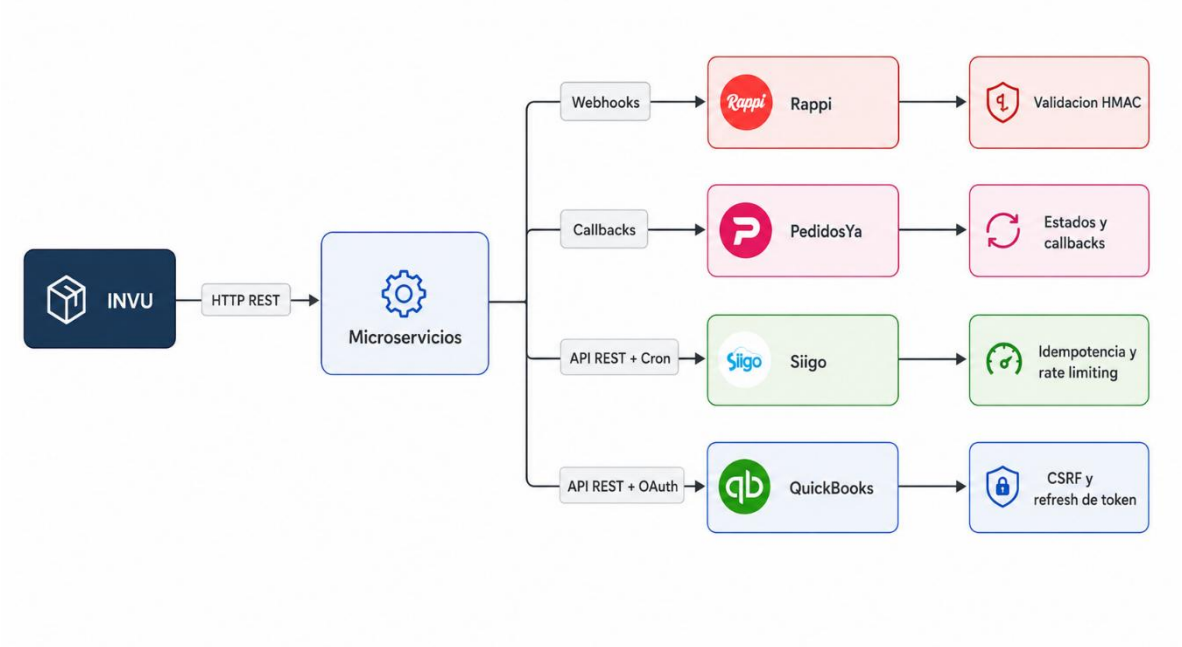
La **seguridad** se abordó mediante mecanismos específicos según la plataforma, como la validación HMAC de webhooks en Rappi, el uso de OAuth 2.0 en QuickBooks, la protección CSRF en formularios web y la gestión de credenciales mediante variables de entorno o configuraciones controladas.

La **interoperabilidad** se garantizó a través del consumo de APIs externas, la transformación de datos entre formatos distintos y la adaptación de la información de INVU POS a los modelos requeridos por cada plataforma. Esto permitió que sistemas con estructuras diferentes pudieran comunicarse de forma controlada.

La **escalabilidad** se consideró al diseñar integraciones separadas por plataforma, con soporte para múltiples tiendas, configuraciones por establecimiento, procesos automáticos y control de concurrencia mediante mecanismos como cron_locks en las integraciones contables.

Finalmente, la **mantenibilidad** se fortaleció mediante la organización del código en rutas, servicios, controladores, modelos y componentes auxiliares, lo que facilita la comprensión, corrección y ampliación futura de cada integración sin afectar directamente a las demás.

Figura 4. Interfaces de comunicación y seguridad



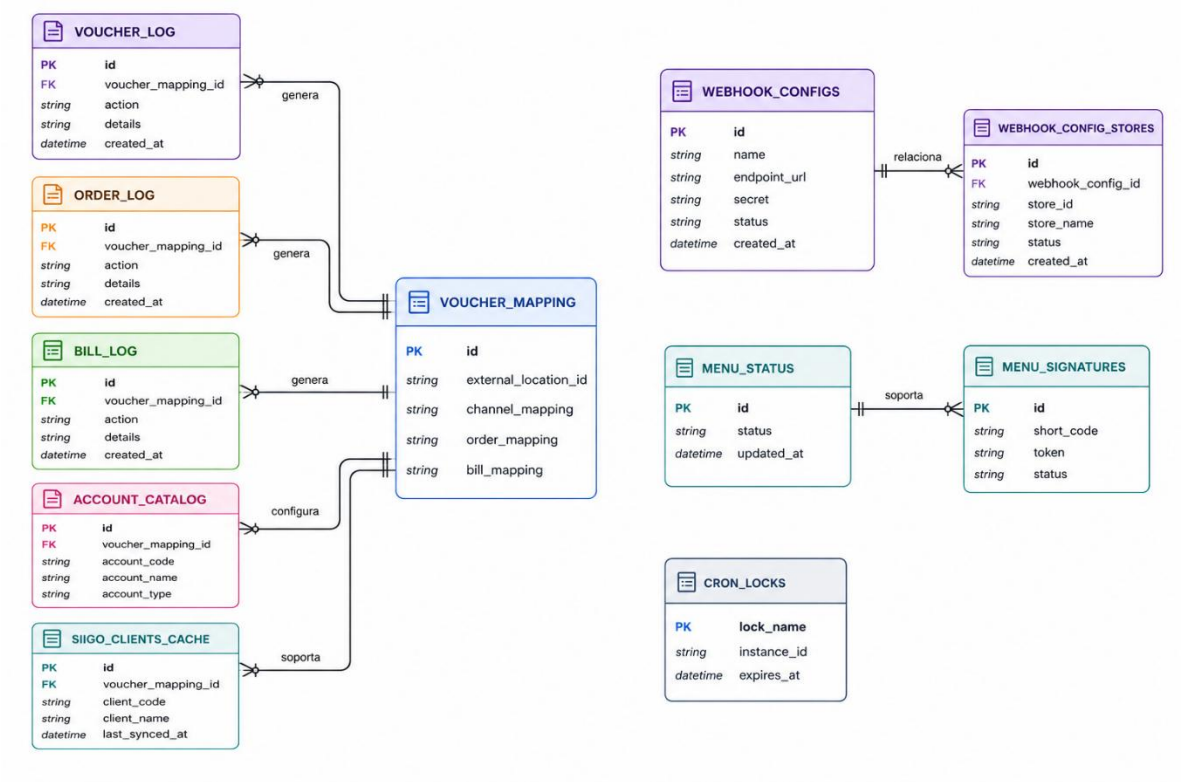

Fuente: autor

#### 6.7.2.4 Modelos de datos y persistencia local

El diseño arquitectónico también incluyó modelos de persistencia para respaldar la operación de las integraciones. Entre las estructuras utilizadas se encuentran tablas para configuraciones de clientes y tiendas, estados de menú, documentos firmados, configuraciones de webhook, mapeos contables, logs de procesamiento, caché de clientes y bloqueos de cron. Estas

estructuras permitieron mantener trazabilidad sobre sincronizaciones, errores, estados de procesamiento, configuraciones de seguridad y operaciones repetibles.

Figura 5. Modelo de datos y persistencia local




### 6.7.2.5 Resultados obtenidos

Como resultado de esta etapa se obtuvo una arquitectura backend organizada por responsabilidades, con servicios especializados por plataforma, mecanismos de comunicación diferenciados según el caso y controles técnicos orientados a mantener seguridad, modularidad y escalabilidad. Esta base facilitó el desarrollo posterior de los

módulos de integración y permitió sostener procesos tanto síncronos como asíncronos dentro del ecosistema construido.

### 6.7.3. Actividades relacionadas con objetivo específico 3

El tercer objetivo consistió en construir los módulos de integración mediante desarrollo backend, implementando servicios funcionales capaces de conectarse con plataformas externas y automatizar operaciones concretas de negocio.

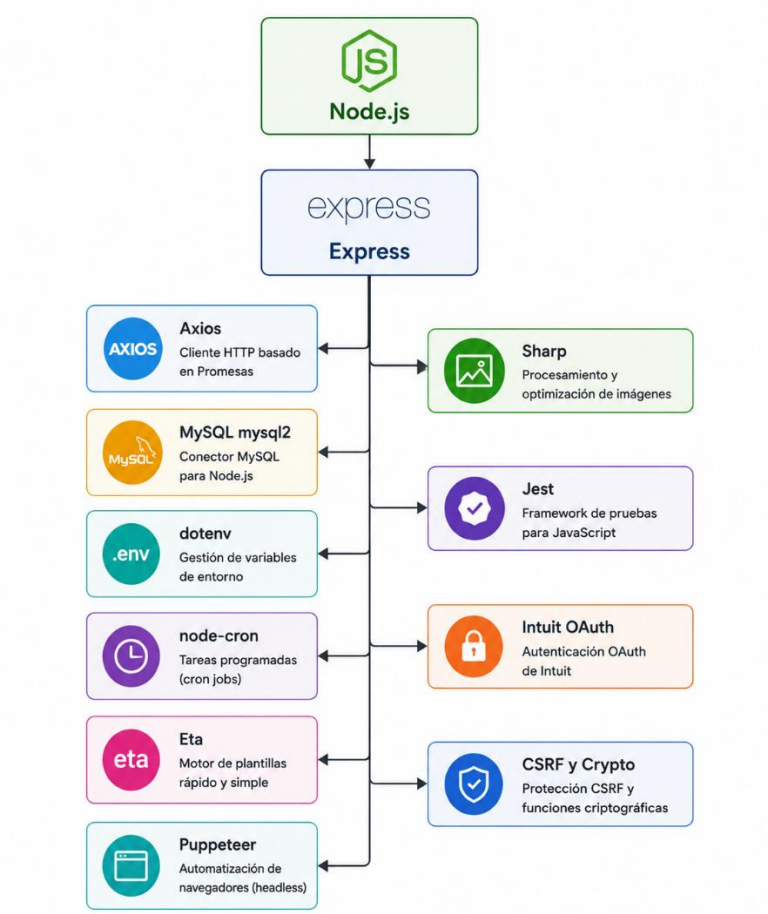
#### 6.7.3.1 Herramientas utilizadas para el desarrollo de los módulos de integración

Para la construcción de los módulos de integración se emplearon principalmente tecnologías del ecosistema JavaScript orientadas al desarrollo backend. Como entorno de ejecución se utilizó Node.js (Node.js, 2026), mientras que Express se usó como framework principal para la definición de rutas, manejo de solicitudes HTTP, recepción de webhooks y construcción de servicios de integración (Express.js, s. f.). Para la comunicación con plataformas externas como Rappi, PedidosYa, Siigo y QuickBooks se utilizó Axios, lo que permitió implementar de forma clara los flujos de autenticación, consulta, envío y recepción de información.

En la capa de persistencia se trabajó con MySQL (Oracle, 2026), mediante los conectores mysql y mysql2, usados para almacenar configuraciones, estados de sincronización, logs, mapeos contables y estructuras auxiliares necesarias para la operación de las integraciones. Adicionalmente, se emplearon herramientas complementarias como dotenv para la gestión de variables de entorno, node-cron para la automatización de procesos

programados, Eta para vistas administrativas, Puppeteer para generación de reportes PDF, Sharp para procesamiento de imágenes, intuit-oauth para la autenticación con QuickBooks, csurf para protección CSRF y Jest para pruebas automatizadas identificadas en la integración con Rappi ver Figura 6.

Figura 6. herramientas backend utilizadas




### 6.7.3.2 Justificación del uso de las tecnologías empleadas

La selección de estas tecnologías responde, en primer lugar, a que una parte importante de la base del proyecto ya se encontraba construida sobre Node.js, Express y MySQL, por lo que mantener este stack permitió dar continuidad al desarrollo existente, conservar coherencia

arquitectónica y facilitar el mantenimiento del código ya implementado. En segundo lugar, se verificó que se tratara de tecnologías vigentes, ampliamente adoptadas en entornos de desarrollo actuales y adecuadas para aplicaciones orientadas a integración de servicios.

El uso continuado de este stack resultó conveniente porque las integraciones requerían consumir múltiples APIs externas, procesar eventos en tiempo real, programar tareas automáticas y mantener trazabilidad de estados, credenciales y respuestas. En este contexto, Node.js y Express ofrecieron un entorno flexible para construir servicios ligeros y modulares (Node.js, 2026; Express.js, s. f.), mientras que MySQL brindó una base relacional sólida para el almacenamiento estructurado de configuraciones y registros operativos (Oracle, 2026).

#### 6.7.3.3 Modelo de base de datos de soporte a las integraciones

El modelo de base de datos implementado para las integraciones es de tipo relacional y cumple una función de soporte operativo. Su objetivo principal no es reemplazar la base de datos del sistema origen, sino almacenar configuraciones técnicas, estados de procesamiento, relaciones auxiliares y registros de auditoría que permiten ejecutar y supervisar correctamente las integraciones.

En las integraciones con Rappi y PedidosYa se identifican estructuras orientadas al control de carga de menús, seguimiento de estados y validación documental, como menu_status y menu_signatures. En el caso de Rappi también se emplean tablas como webhook_configs y webhook_config_stores, destinadas a la administración de configuraciones de webhooks y la asociación de tiendas a eventos específicos.

En las integraciones contables con Siigo y QuickBooks, la tabla voucher_mapping actúa como núcleo de configuración, ya que concentra credenciales, mapeos y parámetros de sincronización. Sobre esta base se apoyan tablas de trazabilidad como voucher_log, order_log y bill_log, además de estructuras complementarias como account_catalog, siigo_clients_cache y cron_locks. Este diseño permitió separar con claridad la lógica de configuración, ejecución, control de concurrencia y auditoría de resultados.

Figura 7. Modelo de datos de la integración con Rappi

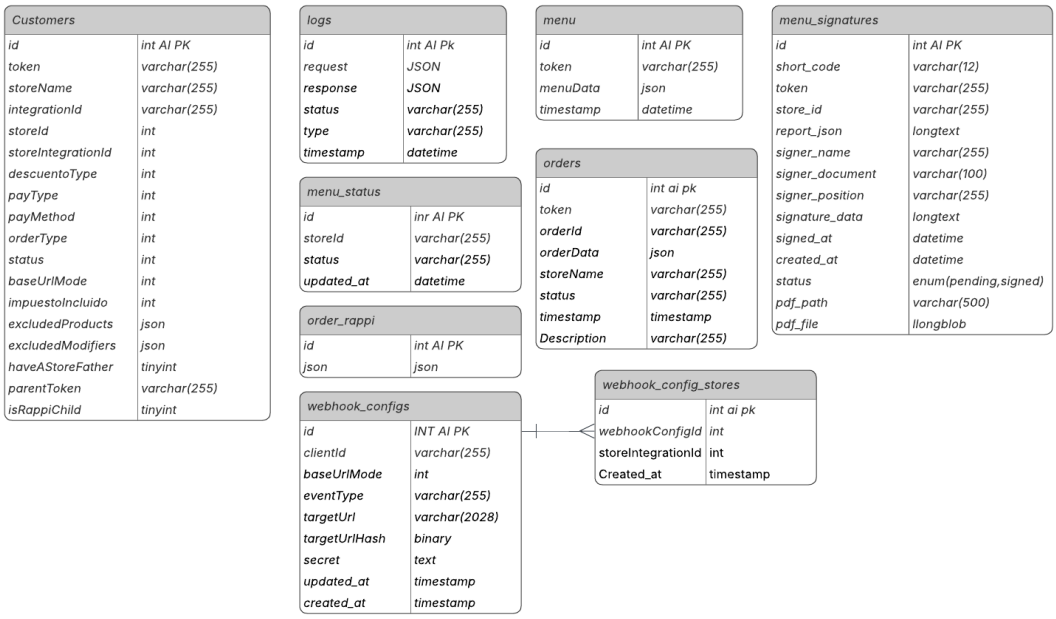


Figura 8. Modelo de datos de la integración con PedidosYa

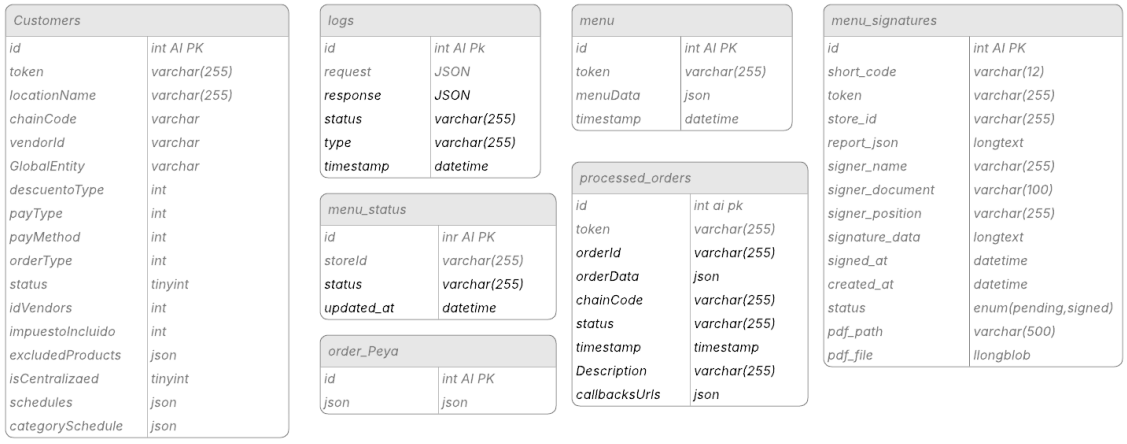


Figura 9. Modelo de datos de la integración con Siigo

Figura 10. Modelo de datos de la integración con QuickBooks


Es importante señalar que los modelos de base de datos empleados en las integraciones desarrolladas no fueron concebidos como estructuras altamente relacionales, en las que todas las tablas dependieran entre sí mediante múltiples llaves foráneas. Por el contrario, se diseñaron como esquemas de soporte orientados a la operación técnica de cada integración, priorizando el almacenamiento de configuraciones, estados, registros de ejecución, auditoría y trazabilidad de la información intercambiada con plataformas externas.

Esta decisión de diseño responde a la naturaleza propia de las integraciones implementadas. En los casos de Rappi y PedidosYa, gran parte de la información es recibida mediante webhooks, callbacks y eventos asíncronos, lo que exige estructuras flexibles capaces de almacenar pedidos, respuestas, estados de menú, firmas y registros JSON completos sin depender de una secuencia estricta de relaciones entre entidades. De forma similar, en Siigo y QuickBooks se requirió almacenar configuraciones contables, mapeos, credenciales,

bitácoras de procesamiento y mecanismos de control para tareas automáticas, por lo que varias tablas cumplen funciones auxiliares o de seguimiento más que de relación transaccional directa.

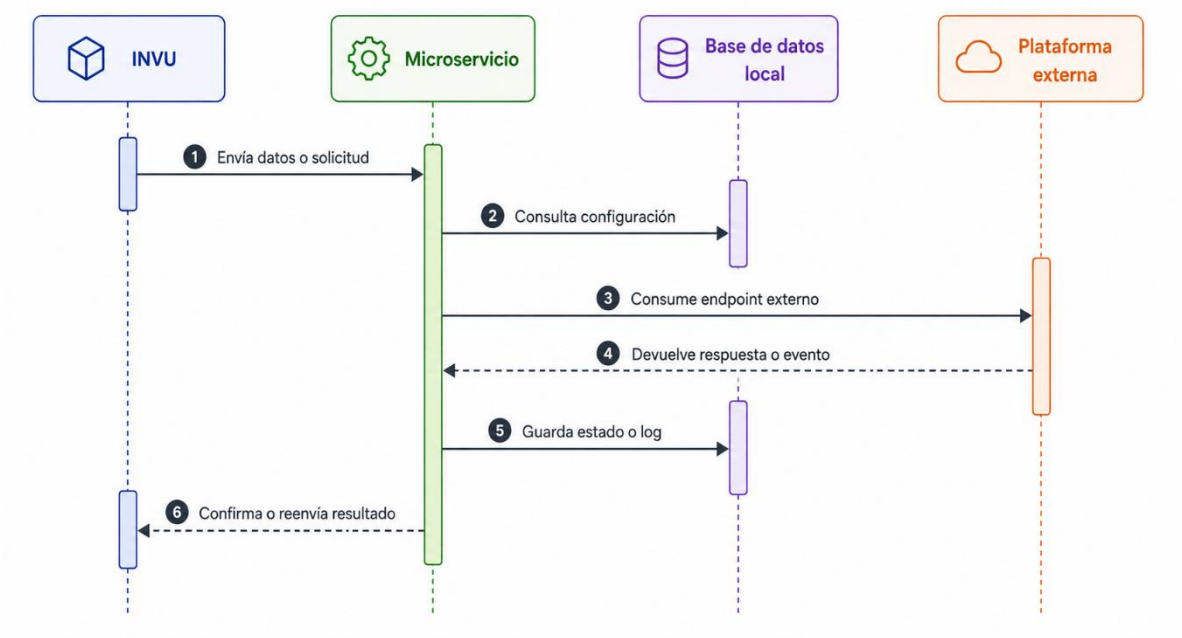
En este sentido, la baja cantidad de conexiones visibles en los diagramas no debe interpretarse como una deficiencia del modelo, sino como una característica coherente con una arquitectura desacoplada de microservicios e integraciones. Este enfoque permitió conservar independencia entre procesos, facilitar el manejo de eventos externos, mejorar la trazabilidad de errores y mantener una mayor flexibilidad operativa en cada módulo desarrollado.

#### 6.7.3.4 Arquitectura lógica y estructura de los módulos de integración

La arquitectura de los módulos desarrollados siguió una organización modular por responsabilidades. Cada integración se estructuró a partir de un punto de entrada principal, un conjunto de rutas, controladores o servicios intermedios, acceso a base de datos y componentes auxiliares especializados según la plataforma. Esta separación permitió aislar la lógica propia de cada sistema externo y facilitar tanto el mantenimiento como la evolución independiente de cada integración.

A nivel funcional, los módulos de delivery se orientaron principalmente a la sincronización de menús, disponibilidad y pedidos, mientras que los módulos contables se enfocaron en el envío automatizado de documentos y el manejo de mapeos financieros. Asimismo, se incorporaron procesos síncronos para solicitudes directas y procesos asíncronos o programados para tareas automáticas, especialmente en las integraciones con Siigo y QuickBooks.

Figura 11. arquitectura lógica y estructura de módulos de integración




### 6.7.3.5 Integración con Rappi

En esta integración se desarrollaron funcionalidades para sincronización de menús, consulta de estado de carga, recepción y procesamiento de pedidos, rechazo de órdenes, actualización de disponibilidad de productos y control de disponibilidad de tienda(ver Figura 12). También se implementó la gestión de webhooks para eventos como aprobación y rechazo de menú, nuevas órdenes, cancelaciones y conectividad(ver Figura 15). De igual forma, se incorporó un flujo adicional para generación de reportes de menú con firma digital, permitiendo validar visualmente la información antes de su operación en plataforma Figura 14.

Figura 12. Vista de configuración de la integración con Rappi

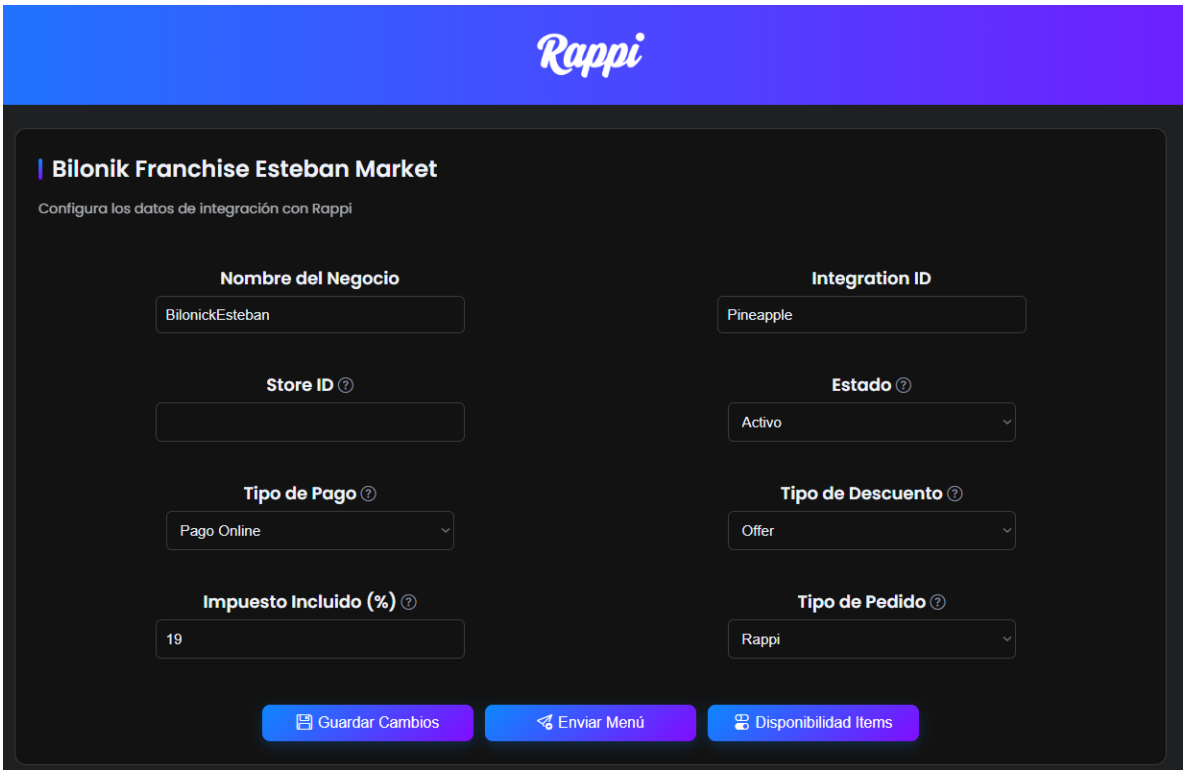



Figura 13. Configuración de tiendas hijas en Rappi

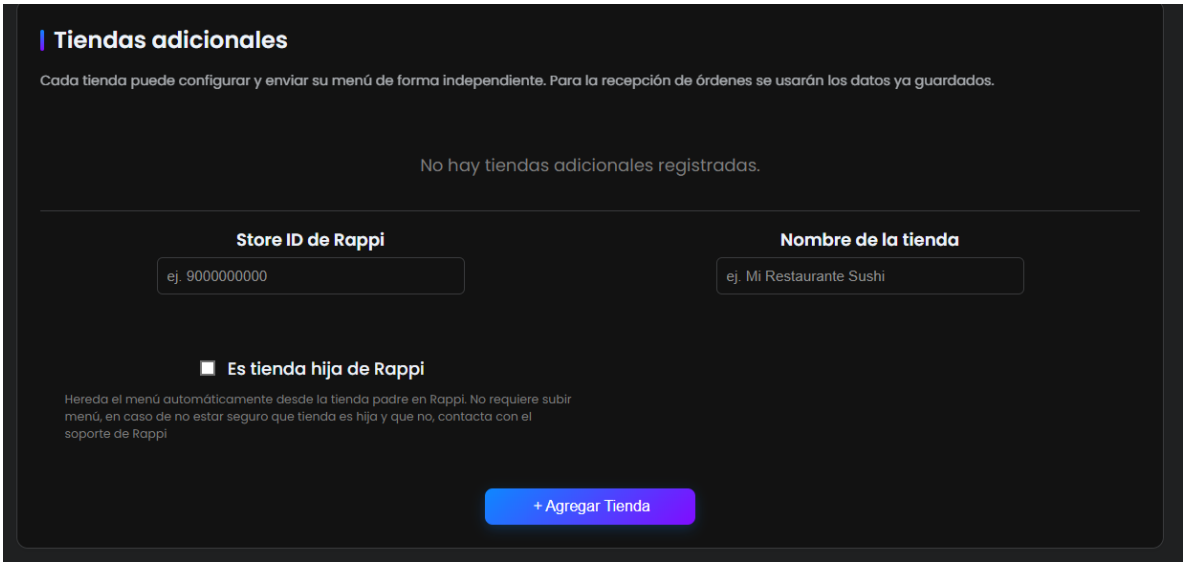


Figura 14. Documento de términos y condiciones previo al uso de la integración

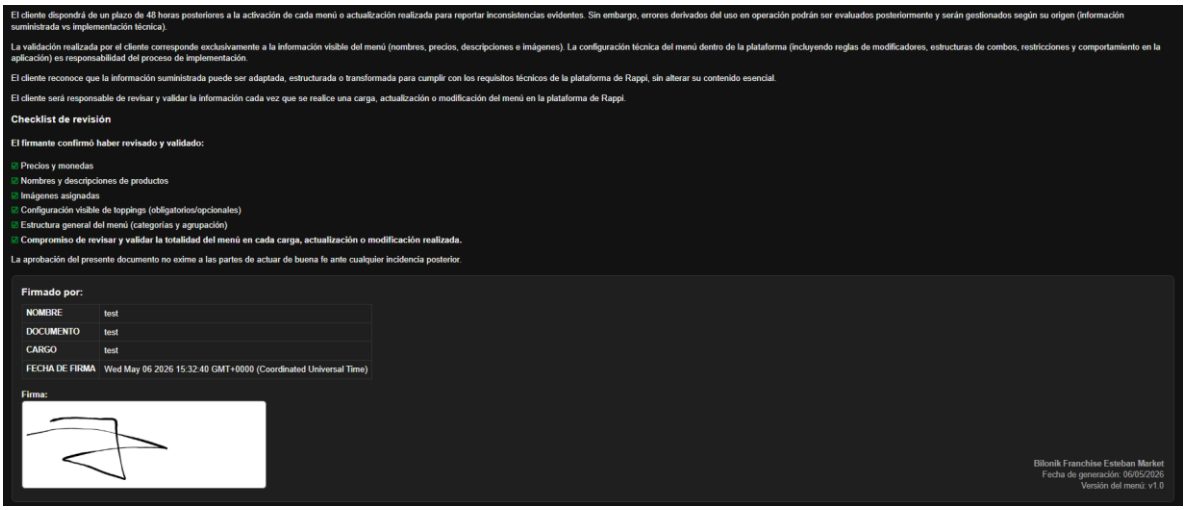
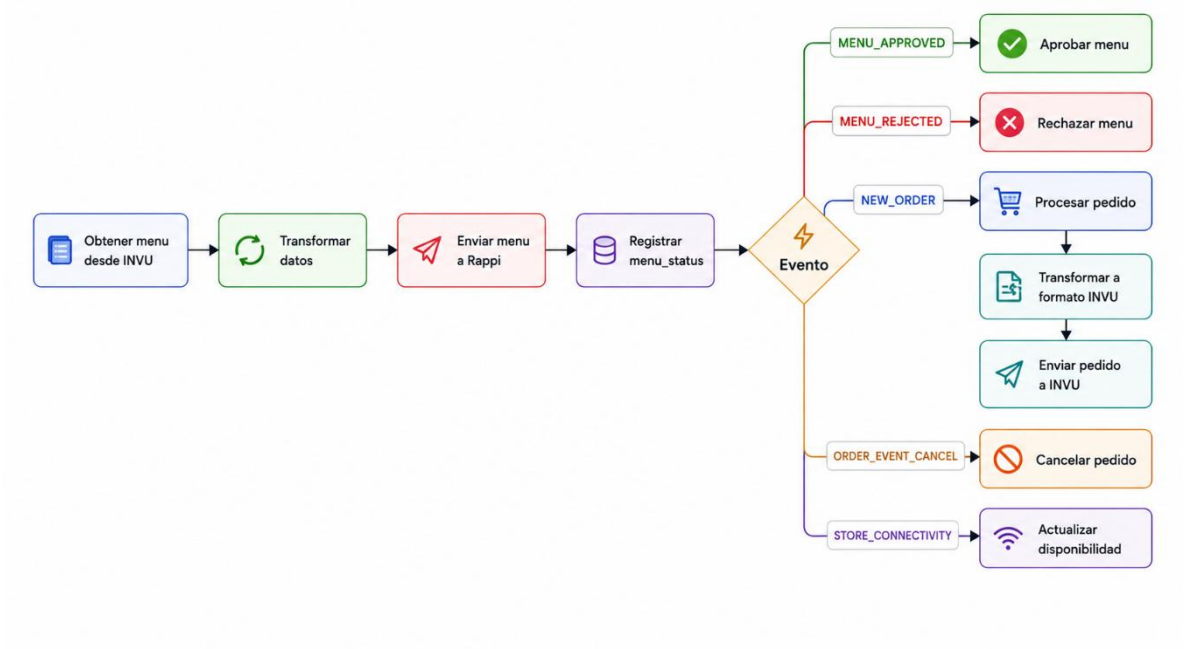



Figura 15. Flujo Integración Rappi



#### 6.7.3.6 Integración con PedidosYa

Para PedidosYa se construyó la lógica de carga de catálogo, tanto en modalidad individual como centralizada, además de funciones para gestión de horarios por categoría, disponibilidad de productos y estados de tienda (ver Figura 16, Figura 17, Figura 19). En el ciclo operativo del pedido se implementaron procesos para aceptación, rechazo, notificación de pedido preparado, pedido recogido (ver Figura 20) y modificación posterior de productos. Al igual que en Rappi, se desarrolló un sistema de reporte y firma digital para validación de menús, incluyendo generación de enlaces, almacenamiento del documento firmado y descarga en PDF Figura 18.

Figura 16. Vista de configuración de la integración con pedidosYa

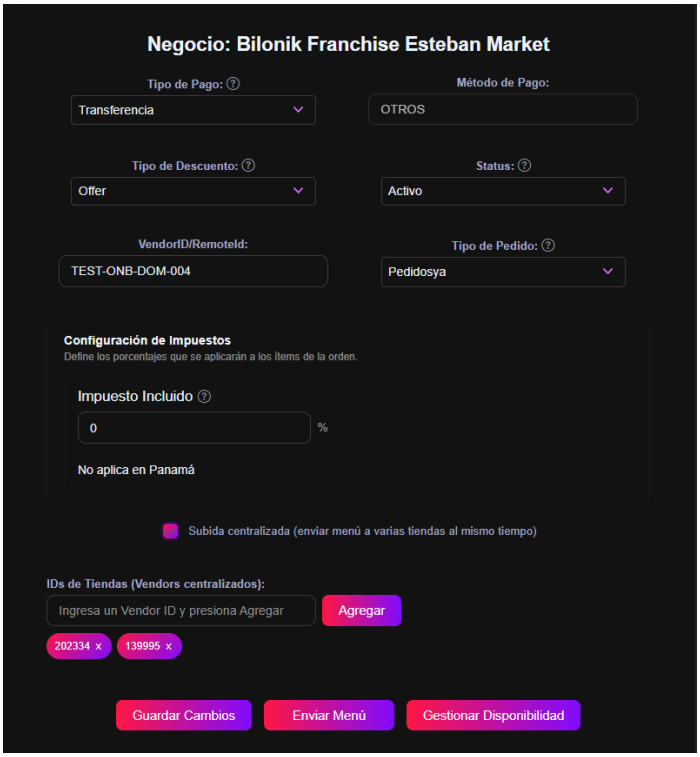


Figura 17. Previsualización y aprobación de la importación de catálogo en PedidosYa

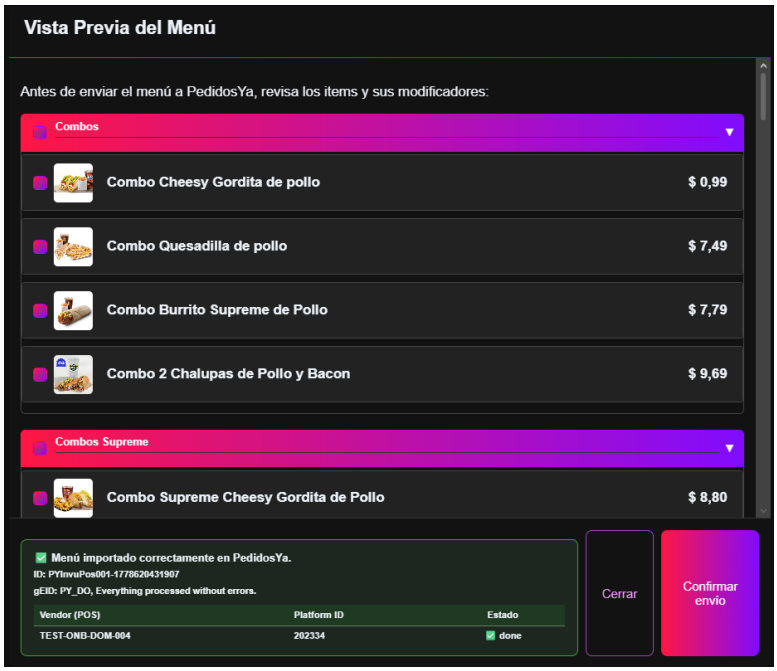



Figura 18. Documento de términos y condiciones previo al uso de la integración

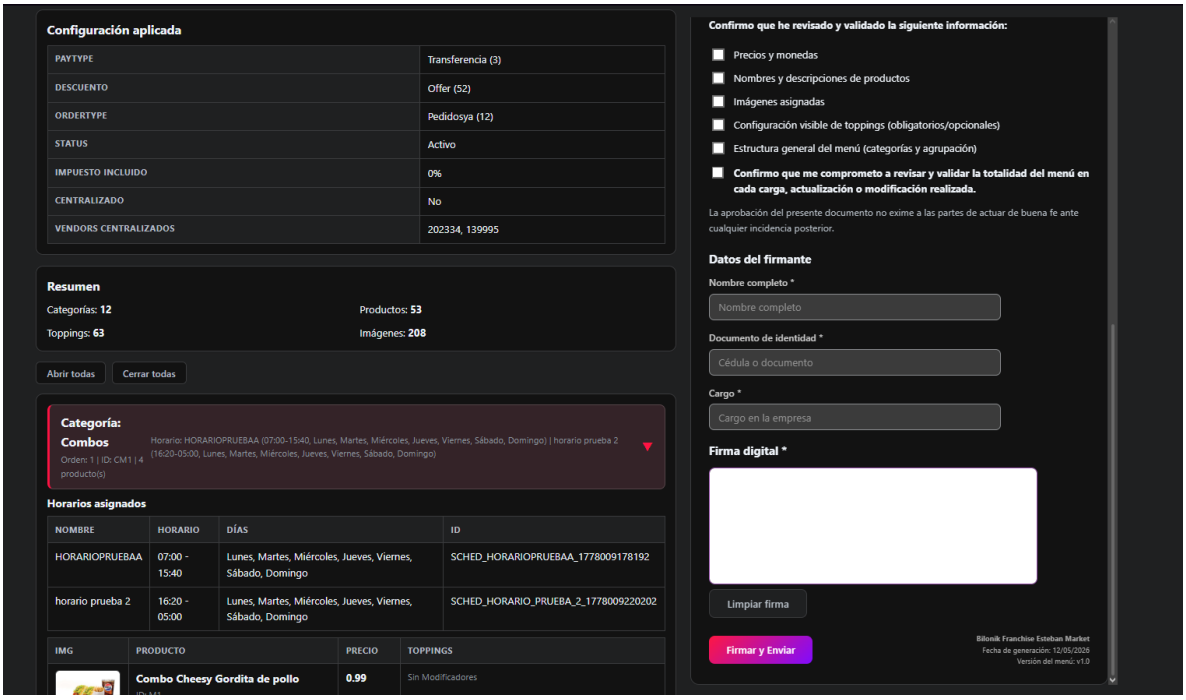


Figura 19. Gestión de horarios y categorías en PedidosYa

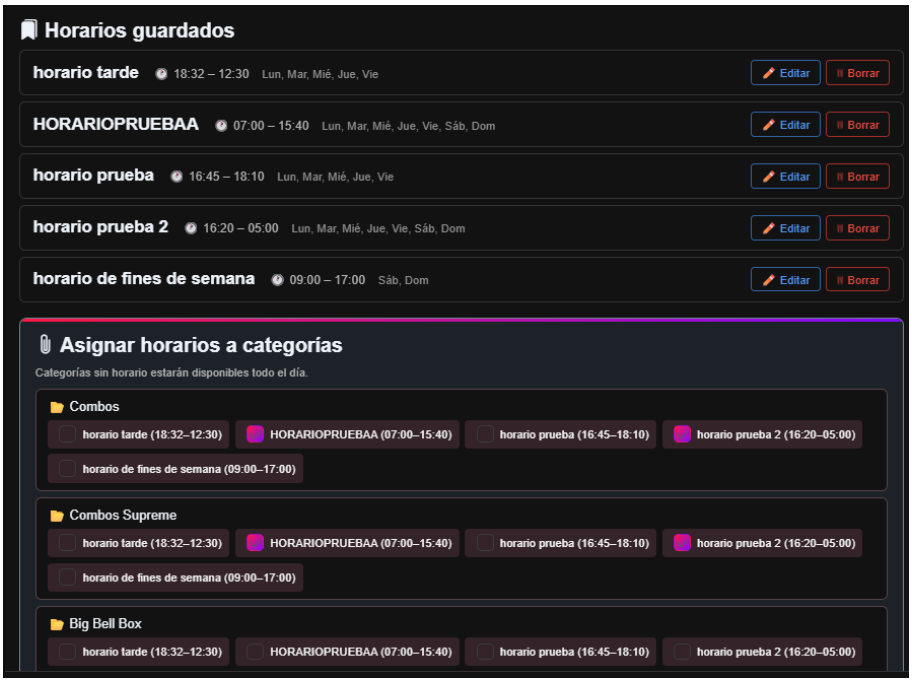
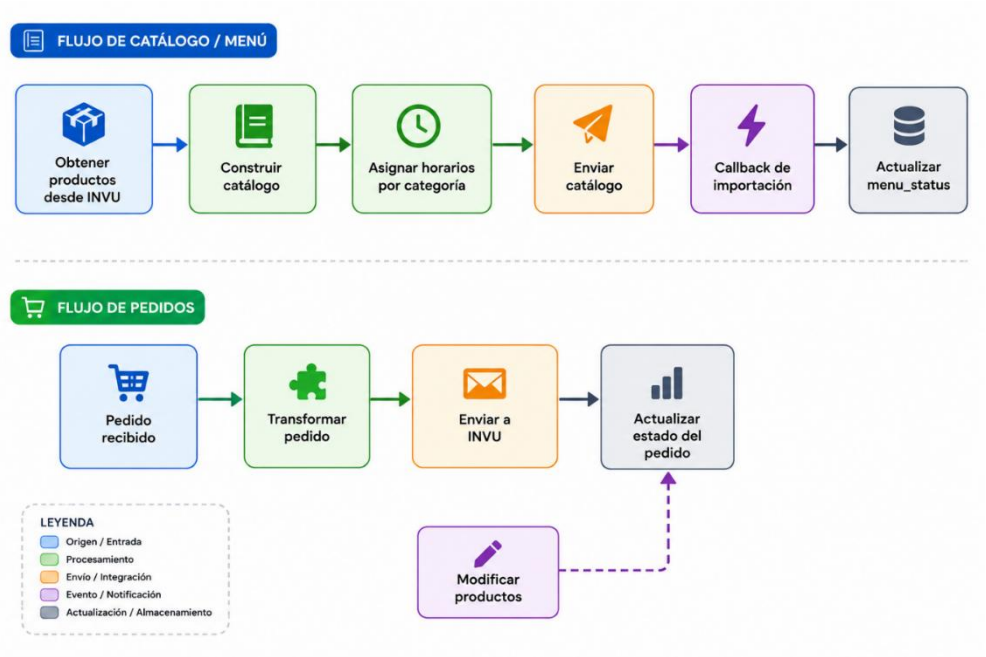



Figura 20. Flujo de la integración con PedidosYa



#### 6.7.3.7 Integración con Siigo

En el caso de Siigo se desarrollaron procesos automáticos orientados a la sincronización contable. Se construyeron flujos para envío de journals (Figura 21), facturas (Figura 22) y órdenes (Figura 23) de compra desde INVU, así como módulos auxiliares para mapeo de cuentas, impuestos, proveedores (Figura 24 , Figura 25 ). Dentro de esta integración se añadieron validaciones para identificación de documentos duplicados, auto creación de productos faltantes, sincronización en segundo plano de clientes y control de dependencias contables antes de emitir un documento.

Figura 21. Vista de journals en Siigo

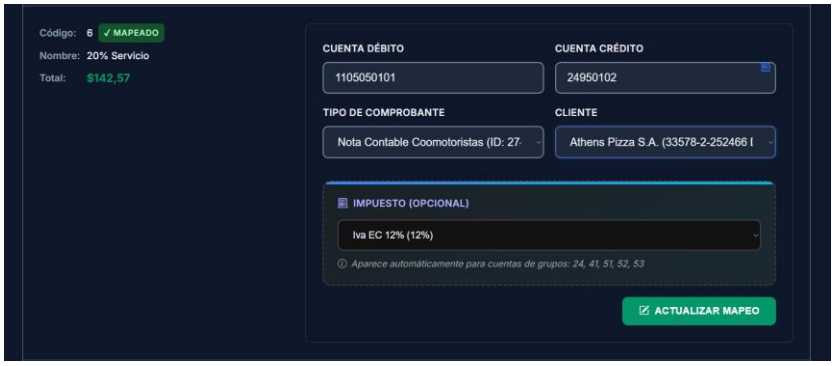


Figura 22 Vista Facturas Siigo

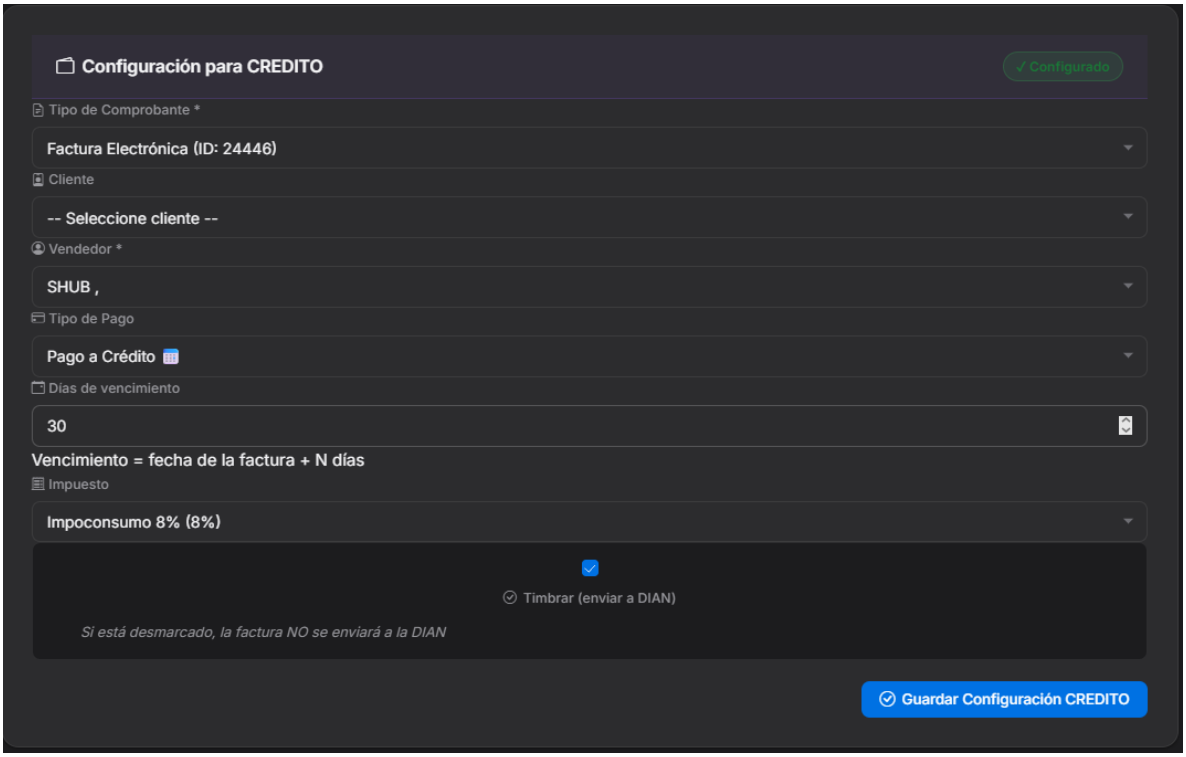



Figura 23. Vista de órdenes de compra en Siigo

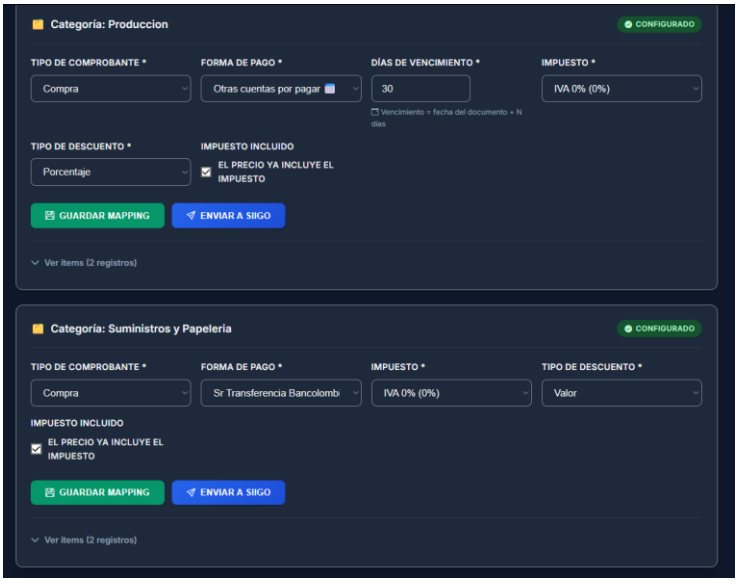


Figura 24. Mapeo de proveedores Siigo

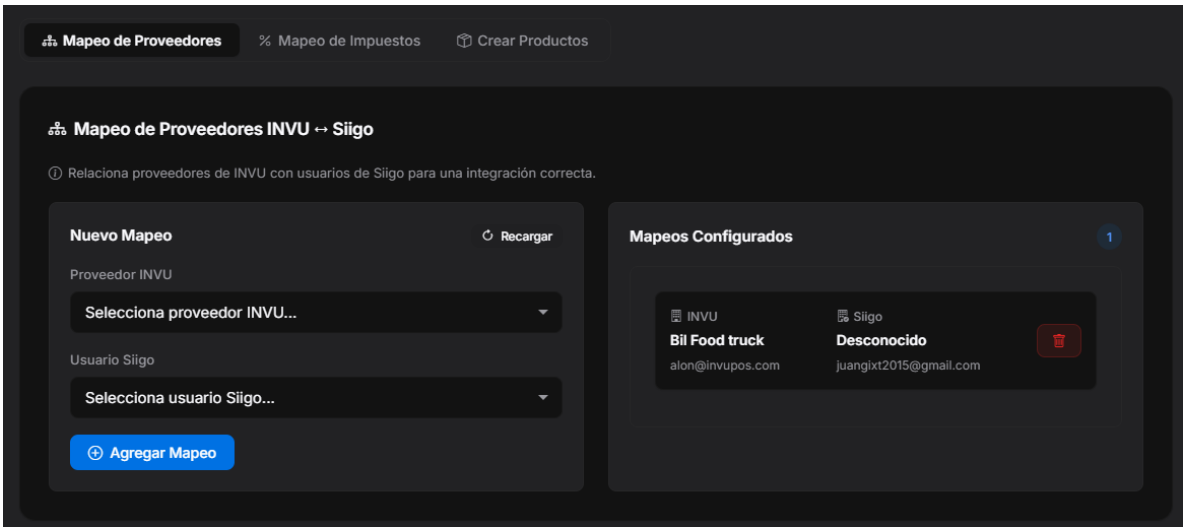



Figura 25. Mapeo de impuestos Siigo

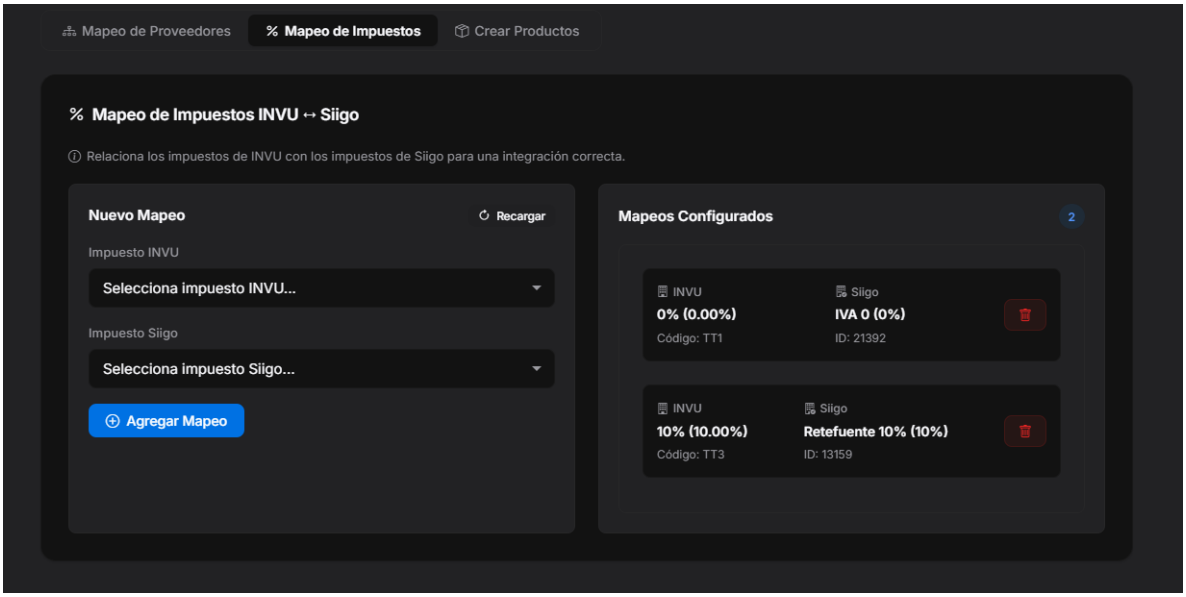
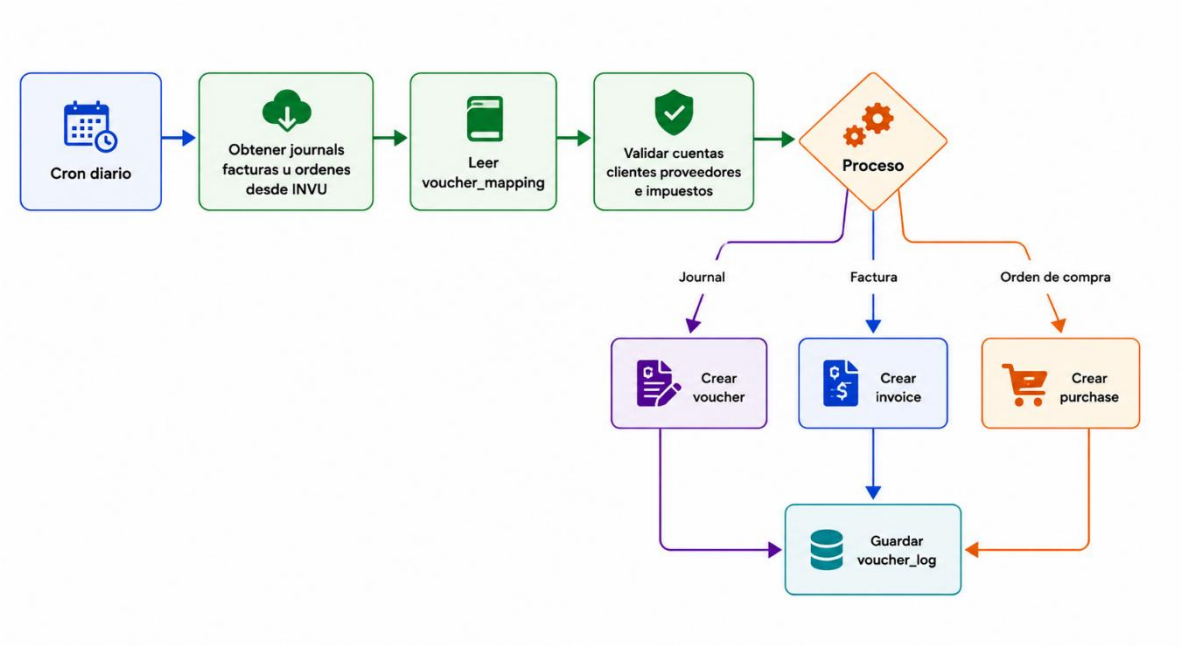


Figura 26. flujo integración con Siigo




### 6.7.3.8 Integración con QuickBooks

En QuickBooks se implementó la autenticación con OAuth 2.0 y la persistencia segura de credenciales. Sobre esta base se desarrollaron procesos para crear journal entries (Figura 28), purchase orders (Figura 30) y sales receipts (Figura 29), además de módulos auxiliares para gestión de cuentas, clientes, proveedores, ítems, métodos de pago e impuestos (Figura 31, Figura 32) . También se incorporaron validaciones contables para escenarios particulares, como el uso de cuentas por cobrar y cuentas por pagar, que exigen la asociación explícita de clientes o proveedores.

Figura 27. Vista principal de la integración con QuickBooks

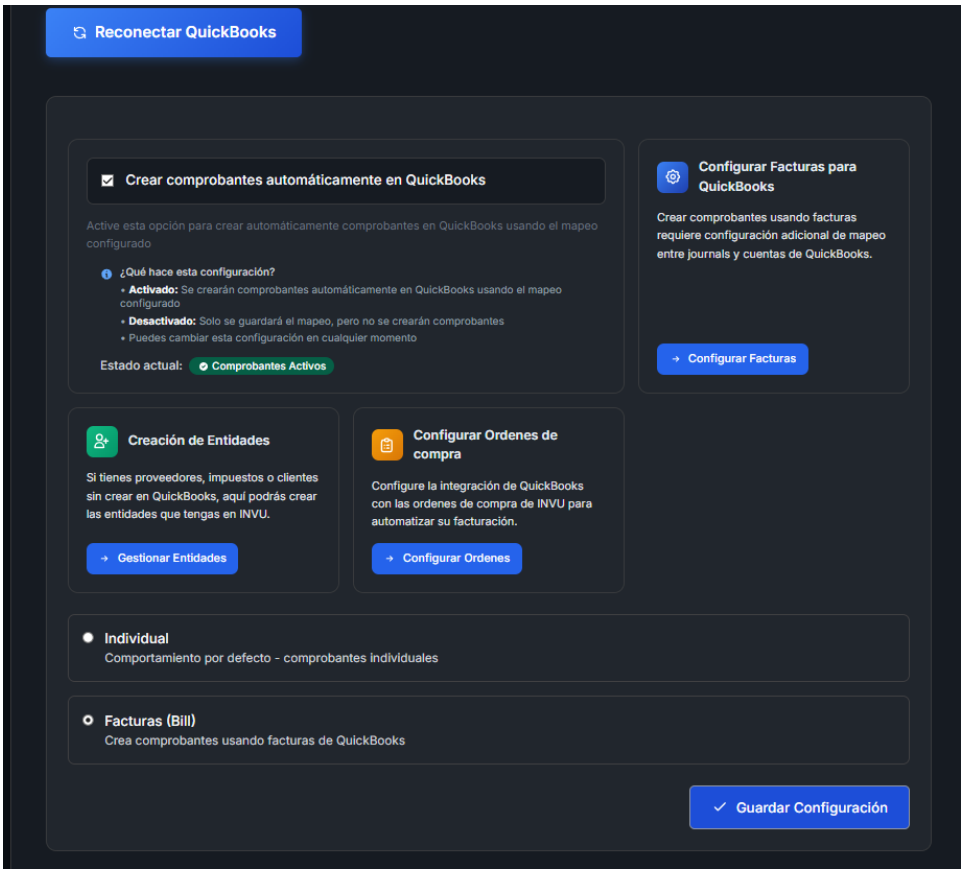


Figura 28. Vista de journals en QuickBooks

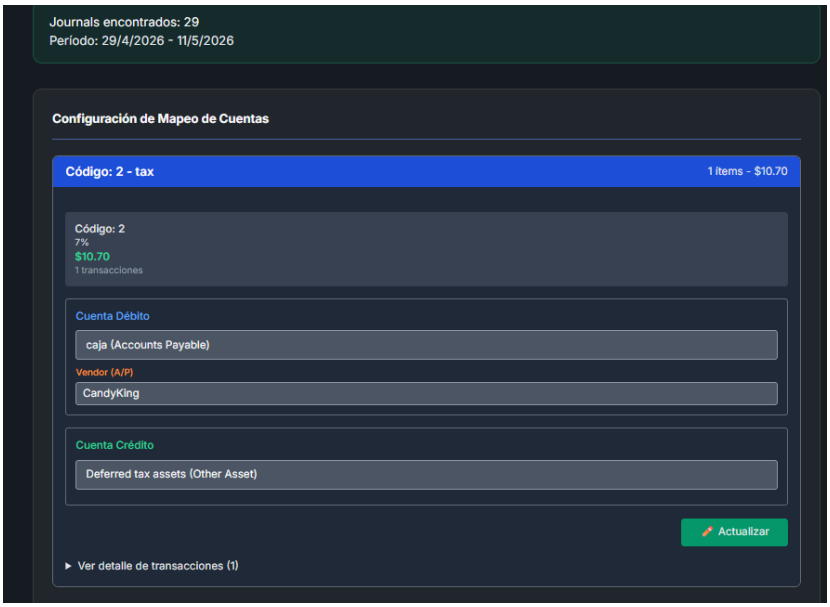



Figura 29. Vista de facturas en QuickBooks

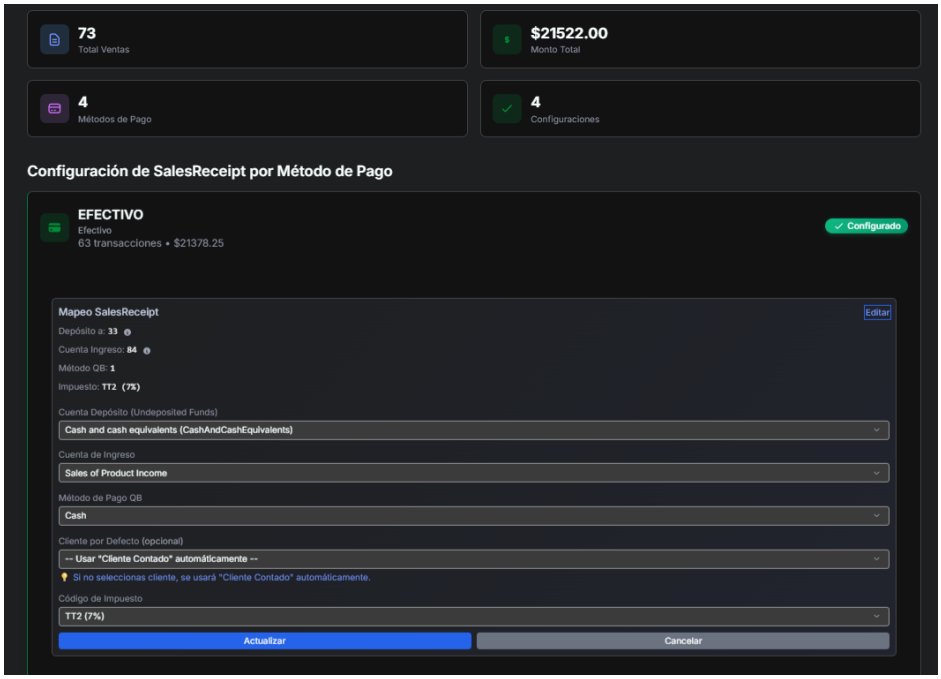


Figura 30. Vista de órdenes de compra en QuickBooks

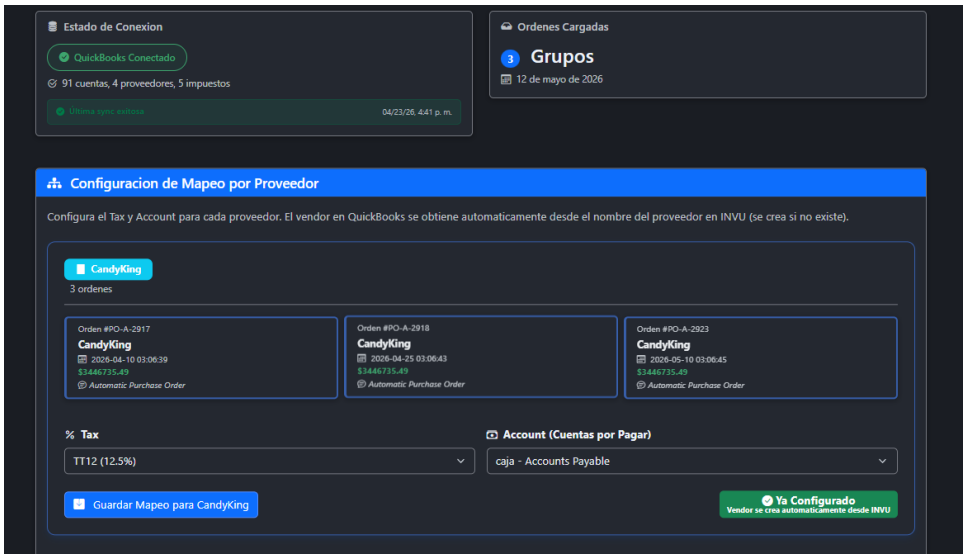



Figura 31. Vista de gestión de entidades en QuickBooks

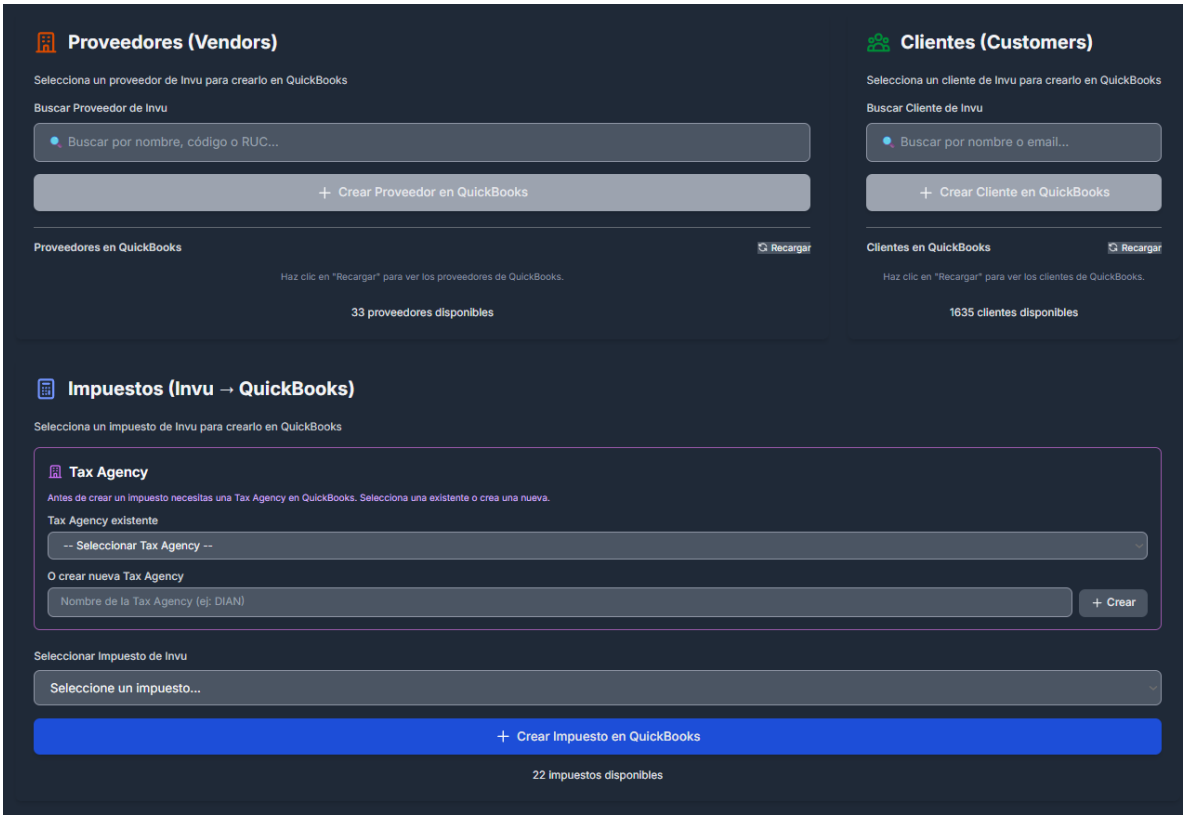


Figura 32. Gestión de entidades auxiliares en QuickBooks

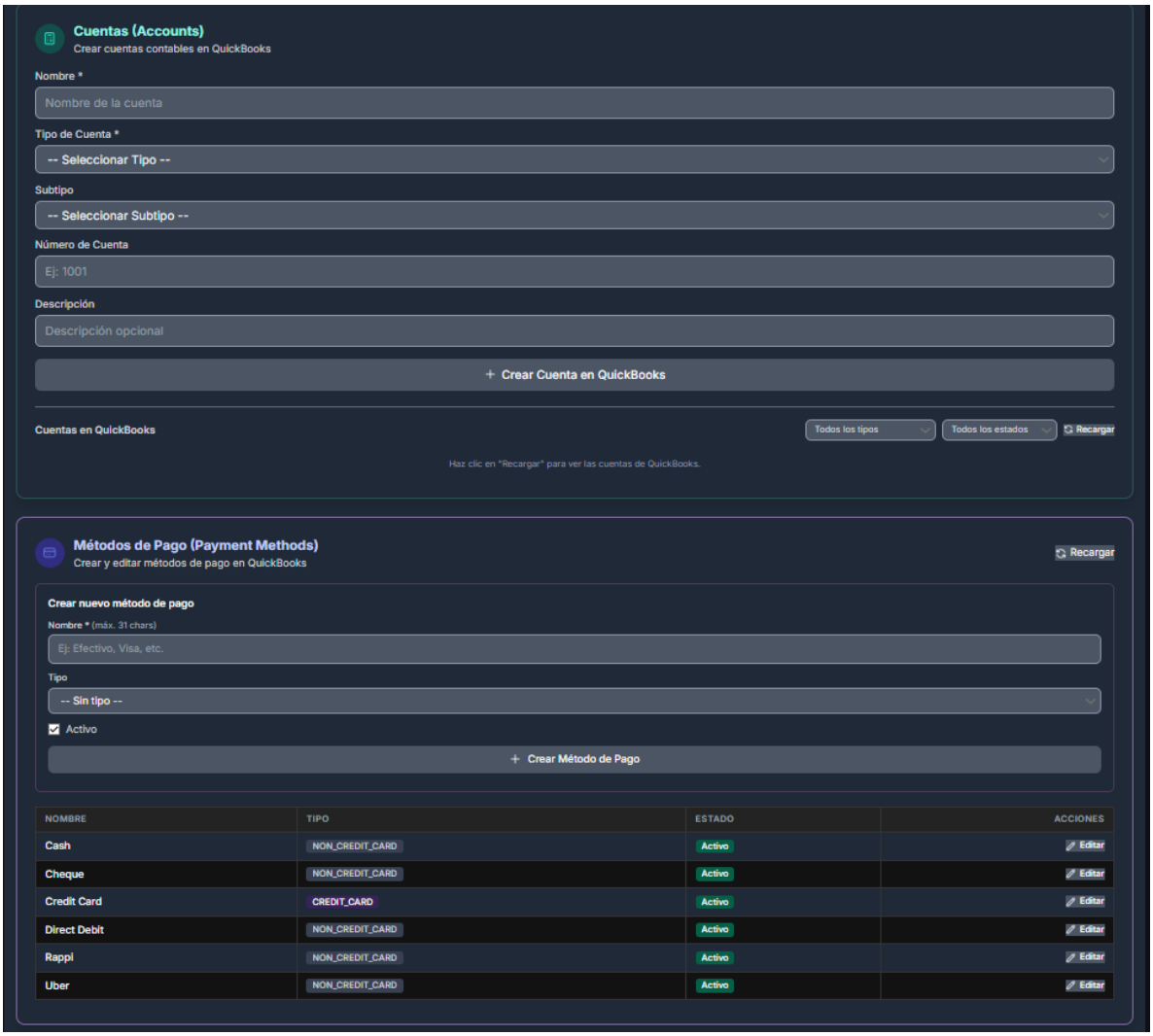
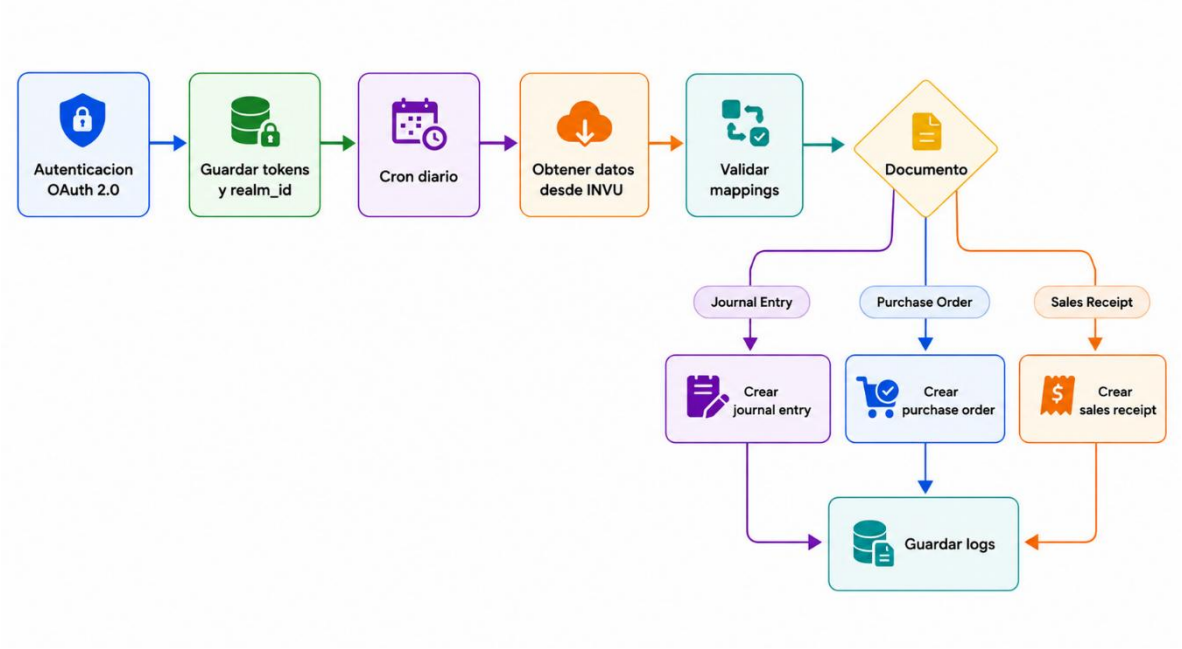


Figura 33. flujo integración QuickBooks




### 6.7.3.9 Consideraciones de despliegue y validación preproductiva

Antes de habilitar una integración para su uso por parte de un cliente, se realizaba una etapa previa de validación en un entorno de pruebas o staging. Esta fase permitía comprobar que el módulo desarrollado funcionara correctamente en condiciones cercanas al entorno real, con el propósito de reducir errores al momento de pasar a producción.

Durante esta etapa se revisaban de manera integral los procesos principales de cada integración. En el caso de Rappi y PedidosYa, se verificaban aspectos como la carga de menús o catálogos, la recepción de pedidos, el manejo de Callbacks o webhooks y la actualización de estados. Por su parte, en Siigo y QuickBooks se validaban los procesos de autenticación, sincronización automática y creación de documentos contables, tales como

facturas, comprobantes y órdenes de compra. También se comprobaba que la información quedara registrada correctamente en la base de datos y que los flujos automáticos operaran sin inconsistencias.

Una vez superadas estas pruebas, la integración se consideraba apta para ser utilizada en producción. De esta forma, el despliegue no consistía únicamente en publicar el servicio, sino en asegurar previamente que su funcionamiento hubiera sido revisado y validado antes de exponerlo al uso real del cliente.

### 6.7.4. Actividades relacionadas con objetivo específico 4

El cuarto objetivo estuvo orientado a validar la funcionalidad, la seguridad y la estabilidad operativa de las integraciones construidas. Esta validación se desarrolló mediante controles en el código, pruebas funcionales sobre flujos específicos y mecanismos de seguimiento en tiempo de ejecución.

#### 6.7.4.1 Enfoque general de validación

La validación de las integraciones se desarrolló de manera progresiva durante la implementación de cada módulo, combinando revisión funcional, verificación técnica y seguimiento de resultados en ejecución. Debido a que se trató de integraciones con plataformas externas, el proceso de prueba no se limitó únicamente a la respuesta visual del sistema, sino que también incluyó autenticación, procesamiento de eventos, almacenamiento local de estados y revisión de logs generados por cada flujo.

En términos generales, la validación buscó comprobar que las integraciones ejecutaran correctamente las operaciones esperadas, que respondieran adecuadamente ante errores o configuraciones incompletas y que mantuvieran consistencia entre la información enviada desde INVU, la respuesta recibida desde la plataforma externa y el registro persistido en la base de datos local.

#### 6.7.4.2 Pruebas de funcionalidad

Las pruebas de funcionalidad se enfocaron en verificar que cada integración cumpliera con las operaciones principales para las cuales fue diseñada. En Rappi se revisaron procesos como la carga de menú, consulta de estados, recepción de pedidos, rechazo de órdenes, actualización de disponibilidad y recepción de eventos por webhook. En PedidosYa se validaron la carga de catálogo, la recepción de pedidos, la actualización de estados operativos, la modificación posterior de productos y la gestión de horarios y disponibilidad. En Siigo y QuickBooks se verificó la correcta generación de documentos contables, la aplicación de mapeos y el registro de resultados en las tablas de log correspondientes.

#### 6.7.4.3 Pruebas de integración

Las pruebas de integración se orientaron a comprobar la correcta comunicación entre los módulos desarrollados y las APIs externas. Esto incluyó validaciones de autenticación, envío y recepción de datos, actualización de estados, manejo de respuestas y almacenamiento local de la información asociada a cada proceso. También se revisó el comportamiento del sistema ante escenarios de error, como credenciales inválidas,

configuraciones incompletas o datos faltantes, verificando que la integración registrara adecuadamente el incidente y evitara inconsistencias operativas.

#### 6.7.4.4 Pruebas de usabilidad

En los módulos que contaban con vistas administrativas o formularios de configuración, se realizaron pruebas de usabilidad orientadas a verificar la claridad del flujo de trabajo, la comprensión de los formularios y la facilidad para consultar estados o corregir configuraciones. Estas validaciones se aplicaron especialmente en las vistas de QuickBooks, Siigo y en los módulos administrativos de reporte y firma digital asociados a Rappi y PedidosYa. El objetivo fue asegurar que la interacción con los formularios de mapeo, configuración y monitoreo fuera comprensible y útil para el usuario técnico encargado de operar la integración.

#### 6.7.4.5 Pruebas de carga y rendimiento

No se identificó dentro de los proyectos una evidencia explícita de pruebas de carga formales o pruebas de estrés ejecutadas sobre las integraciones. Sin embargo, sí se implementaron mecanismos técnicos orientados a mantener la estabilidad operativa y controlar el rendimiento en escenarios de consumo repetitivo de APIs. Entre estos mecanismos se encuentran el uso de retrasos controlados para respetar límites de consumo, llaves de idempotencia para evitar duplicidad documental, caché de autenticación y bloqueos de ejecución mediante cron_locks para prevenir concurrencia entre tareas automáticas programadas.

#### 6.7.4.6 Resultados y evidencias de prueba

Como resultado del proceso de validación, las integraciones quedaron respaldadas por controles funcionales y técnicos que permitieron reducir errores de configuración, duplicidad documental, inconsistencias de autenticación y fallos derivados de concurrencia. En el caso de Rappi se identificó además la existencia de pruebas automatizadas orientadas al controlador de webhooks, lo que aportó una evidencia adicional sobre la validación de escenarios críticos relacionados con seguridad y asociación de tiendas a eventos. En conjunto, estas verificaciones permitieron fortalecer la confiabilidad operativa del ecosistema desarrollado.

**Tabla 7. Tabla de Testing de integraciones**

| ID | Integración | Tipo de Prueba | Caso evaluado | Resultado esperado | Resultado obtenido | Estado | Acción |
|---|---|---|---|---|---|---|---|
| **P-01** | Rappi | Funcionalidad | Carga de menú | El menú se envía correctamente y queda registrado su estado | Menú correctamente reflejado en la plataforma de Rappi | Exitoso | No Aplica |
| **P-02** | Rappi | Integración | Recepción de webhook de nueva orden | La orden se procesa y se registra en el sistema | Orden llegaba a precio completo y el POS le aplicaba nuevamente el impuesto | Fallido | Se agrego el campo impuesto incluido para que se descuente el % necesario para que el precio del |

| ID | Integración | Tipo de Prueba | Caso evaluado | Resultado esperado | Resultado obtenido | Estado | Acción |
|---|---|---|---|---|---|---|---|
| | | | | | | | producto cuadre con el pago |
| **P-03** | PedidosYa | Funcionalidad | Carga de catalogo | El catálogo se importa correctamente y se actualiza su estado | Menú subido correctamente junto a horarios, precios y descripciones en la plataforma PedidosYa | Exitoso | No aplica |
| **P-04** | PedidosYa | Integración | Recepción de pedido | El pedido se transforma y se envía a INVU | Datos entrantes en POS correctos sin discrepancias con los datos enviados por PedidosYa | Exitoso | No aplica |

| ID | Integración | Tipo de Prueba | Caso evaluado | Resultado esperado | Resultado obtenido | Estado | Acción |
|---|---|---|---|---|---|---|---|
| **P-05** | Siigo | Funcionalidad | Creación de journal | Se genera el comprobante y se registra en log | Números de cuenta inexistentes en Siigo generaba fallos en la creación del comprobante, junto a campos faltantes según el tipo de cuenta | Fallido | Agregar validaciones según el grupo de cuenta y si existe en Siigo Nube |
| **P-06** | Siigo | Integración | Creación de orden de compra | La compra se envía correctamente a Siigo | Creacion correcta de orden de compra con los datos obtenidos desde una Orden de compra de Invu | Exitoso | No Aplica |
| **P-07** | QuickBooks | Funcionalidad | Autenticación OAuth 2.0 | Se almacenan correctamen | Almacenamiento y encriptación del token | Exitoso | No Aplica |

| ID | Integración | Tipo de Prueba | Caso evaluado | Resultado esperado | Resultado obtenido | Estado | Acción |
|---|---|---|---|---|---|---|---|
| | | | | te los tokens y el realm_id | | | |
| **P-08** | QuickBooks | Integración | Creación de sales receipts | Se genera el documento y se registra el resultado | Creación de factura de venta con datos obtenidos desde Invu | Exitoso | No aplica |

#### 6.7.4.7 Validaciones funcionales y operativas

Durante el desarrollo se incorporaron múltiples validaciones para asegurar el comportamiento correcto de las integraciones. En Rappi se implementó validación de firma de Webhook, prevención de procesamiento duplicado de órdenes y manejo de errores específicos por evento. En PedidosYa se añadieron validaciones de formatos, estados de pedido, disponibilidad y restricciones sobre modificaciones. En Siigo y QuickBooks se construyeron validaciones de mapeos, verificación de existencia de cuentas, clientes y proveedores, así como controles para omitir o registrar procesos cuando la configuración no era suficiente para completar una operación.

#### 6.7.4.8 Mecanismos de seguridad implementados

En materia de seguridad se aplicaron controles diferenciados según la plataforma. Entre ellos se encuentran la validación HMAC de webhooks en Rappi, el uso de CSRF en QuickBooks, el cifrado y renovación segura de tokens, la preservación del cuerpo original del mensaje para verificación criptográfica y la separación de credenciales por ambiente o contexto de integración. Adicionalmente, en los procesos automáticos contables se incorporaron locks distribuidos y controles de idempotencia para evitar ejecuciones repetidas o inconsistentes.

#### 6.7.4.9 Pruebas automatizadas identificadas en el proyecto

Dentro de los proyectos se identificó la presencia de pruebas automatizadas en la integración de Rappi, específicamente sobre el controlador de webhooks. Estas pruebas cubren escenarios relacionados con consulta de secretos, vinculación de múltiples tiendas a una misma configuración de webhook y validación de firmas. Su incorporación permitió verificar partes críticas de la seguridad y del comportamiento esperado en la recepción de eventos.

#### 6.7.4.10 Validación de rendimiento

En la fase de validación de rendimiento se evaluó el comportamiento de las integraciones bajo escenarios de alta volumetría de datos y operación continua, con el propósito de verificar su estabilidad, capacidad de procesamiento y tolerancia frente a cargas elevadas de trabajo.

En la integración con **Rappi** se realizaron pruebas de carga de catálogo con un menú compuesto por aproximadamente **300 productos y 900 modificadores**, logrando su subida correcta a la plataforma sin fallos registrados durante el proceso. Adicionalmente, la integración se encuentra actualmente en uso por 9 tiendas, las cuales han recibido múltiples órdenes de forma operativa, evidenciando estabilidad en el procesamiento continuo de pedidos y en la recepción de eventos desde la plataforma externa.

En el caso de **PedidosYa**, se cuenta con al menos un cliente operando en producción con la integración activa. Sobre esta base se observó el procesamiento correcto de

aproximadamente **500 órdenes**, sin fallos registrados en el flujo de recepción y transformación de pedidos. De igual forma, se validó el manejo de más de un horario por categoría, confirmando el correcto funcionamiento de esta característica dentro del catálogo. En cuanto al rendimiento del proceso de importación, se identificó un **tiempo promedio de 180 segundos** para la aprobación de un catálogo, valor que resulta consistente con la naturaleza asincrónica del proceso de validación en plataforma externa.

Para **Siigo**, la validación de rendimiento se centró en procesos contables de ejecución programada. Se comprobó el **procesamiento correcto de 30 o más facturas sin fallos**, así como la generación de **órdenes de compra con más de 1000 ítems**, las cuales fueron enviadas correctamente. Estos resultados permiten afirmar que la integración soporta cargas altas de información documental, manteniendo consistencia en el envío y registro de resultados.

En **QuickBooks**, se verificó que la integración puede procesar **hasta 1000 ítems en un solo documento** sin colapsar, aunque en este escenario el tiempo aproximado de ejecución fue de **10 minutos**, lo cual refleja un comportamiento estable pero sensible al volumen de datos y a las restricciones de la API externa. También se observó que, cuando es necesario renovar credenciales, el proceso de **Refresh de token tarda menos de 2 segundos**, minimizando el impacto sobre la continuidad de la operación.

Finalmente, tanto en Siigo como en **QuickBooks** se implementaron mecanismos para evitar la creación duplicada de documentos, lo que fortalece el rendimiento lógico del sistema al impedir reprocesamientos innecesarias y conservar la consistencia de la información contable. En conjunto, estos resultados muestran que las integraciones desarrolladas presentan un comportamiento estable frente a escenarios de operación real y cargas elevadas, respondiendo adecuadamente a los requerimientos funcionales del proyecto.

**Tabla 8. Escenarios de validación de rendimiento**

| Integración | Escenario Evaluado | Resultado observado |
|---|---|---|
| **Rappi** | Carga de menú con 300 productos y 900 modificadores | Subida correcta a la plataforma |
| **Rappi** | Operación continua en 9 tiendas | Recepción estable de múltiples órdenes |
| **PedidosYa** | Procesamiento de 500 órdenes | Sin fallos registrados |
| **PedidosYa** | Manejo de múltiples horarios por categoría | Funcionamiento correcto |
| **PedidosYa** | Aprobación de catálogo | Tiempo promedio de 180 segundos |
| **Siigo** | Procesamiento de 30 o más facturas | Ejecución correcta sin fallos |

| Integración | Escenario Evaluado | Resultado observado |
|---|---|---|
| **Siigo** | Orden de compra con más de 1000 ítems | Subida correcta |
| **QuickBooks** | Documento con hasta 1000 ítems | Procesamiento correcto en aproximadamente 10 minutos |
| **QuickBooks** | Renovación de token | Tiempo menor a 2 segundos |
| **Siigo/QuickBooks** | Prevención de duplicidad documental | Control implementado y funcionamiento correcto |

#### 6.7.4.11 Resultados obtenidos

Como resultado de esta fase, las integraciones quedaron respaldadas por validaciones funcionales y controles técnicos orientados a reducir errores de configuración, duplicidad documental, inconsistencias de autenticación y problemas derivados de concurrencia. Aunque cada plataforma exige un tratamiento distinto, el sistema construido logró incorporar mecanismos concretos para sostener la operación de forma más segura, controlada y trazable.

### 6.7.5. Actividades extra

A pesar de que no estaban directamente vinculadas a los objetivos específicos del proyecto, a lo largo de la pasantía se desarrollaron varias actividades complementarias que resultaron fundamentales para el correcto funcionamiento del equipo y la escalabilidad de las integraciones, permitiendo un seguimiento y guía a los procesos en general.

#### 6.7.5.1 Documentación técnica complementaria

Como actividad adicional se generó documentación técnica de apoyo en varios de los proyectos desarrollados. Entre estos artefactos se encuentran reportes funcionales para Rappi y PedidosYa, documentación de integración para Siigo y guías específicas para módulos de QuickBooks. Estos documentos sirvieron para describir flujos, rutas, configuraciones, estructuras de almacenamiento y consideraciones de operación, facilitando el entendimiento y mantenimiento de las integraciones construidas.

## 6.8. Herramientas utilizadas en el desarrollo del proyecto

Durante el desarrollo de la pasantía se utilizaron diversas herramientas tecnológicas orientadas principalmente a la construcción de servicios backend, consumo de APIs externas, persistencia de datos, automatización de procesos y validación funcional de las integraciones. Estas herramientas permitieron implementar una solución modular, escalable y adecuada para la interoperabilidad entre INVU POS y plataformas externas de delivery y contabilidad.

Como entorno principal de desarrollo se empleó Node.js, debido a su capacidad para manejar procesos asincrónicos y peticiones concurrentes, característica especialmente útil en integraciones que dependen del intercambio constante de información con APIs externas. Sobre este entorno se utilizó Express como framework para la creación de rutas, controladores y servicios HTTP, facilitando la organización modular de cada integración.

Para la comunicación con sistemas externos como Rappi, PedidosYa, Siigo y QuickBooks se utilizó Axios, herramienta que permitió consumir endpoints REST, enviar información estructurada y manejar respuestas y errores de forma controlada. En la capa de persistencia se trabajó con MySQL, mediante los conectores mysql y mysql2, para almacenar configuraciones, estados de sincronización, registros de auditoría, mappings contables y estructuras de soporte operativo.

Adicionalmente, se usaron herramientas complementarias como dotenv para la administración de variables de entorno, node-cron para la programación de tareas automáticas, Eta para la construcción de vistas administrativas, Puppeteer y Sharp para la generación y procesamiento de documentos PDF, intuit-oauth para la autenticación con QuickBooks y Jest para pruebas automatizadas identificadas en la integración con Rappi.

En conjunto, estas herramientas ofrecieron una base robusta para el desarrollo del proyecto, permitiendo construir integraciones seguras, flexibles y preparadas para operar sobre flujos de datos tanto síncronos como asíncronos.

**Tabla 9. Herramientas usadas en pasantía**

| Herramientas | Uso principal en el proyecto |
|---|---|
| Node.js | Entorno de ejecución backend |
| Express | Construcción de rutas y servicios HTTP |
| Axios | Consumo de APIs externas |

| Herramientas | Uso principal en el proyecto |
|---|---|
| MySQL | Persistencia de configuraciones, logs y estados |
| Dotenv | Gestión de variables de entorno |
| Node-cron | Automatización de procesos programados |
| Eta | Vistas administrativas |
| Puppeteer | Generación de reportes PDF |
| Sharp | Procesamiento de imágenes |
| intuit-oauth | Autenticación con QuickBooks |
| Jest | Pruebas automatizadas en Rappi |

# 7. CONCLUSIONES

La pasantía permitió cumplir de manera satisfactoria el objetivo general propuesto, orientado a la construcción de un ecosistema integrado de gestión entre INVU POS y plataformas externas de delivery y contabilidad. A través del desarrollo de las integraciones con Rappi, PedidosYa, Siigo y QuickBooks, fue posible avanzar en la automatización de procesos operativos y contables, reduciendo tareas manuales, mejorando la interoperabilidad entre sistemas y fortaleciendo la eficiencia operativa de la solución.

Con respecto al **objetivo específico 1**, se logró definir las especificaciones técnicas de las integraciones mediante el análisis de requerimientos funcionales y no funcionales, así como la revisión de autenticación, modelos de datos, flujos de comunicación y capacidades particulares de cada API externa. Esta fase permitió establecer una base clara para orientar el diseño y la implementación posterior de cada módulo.

En relación con el **objetivo específico 2**, se diseñó una arquitectura de servicios desacoplados que permitió separar la lógica particular de cada integración, diferenciar los flujos de delivery y contabilidad, y estructurar mecanismos de comunicación, persistencia local y seguridad adecuados para cada plataforma. Este diseño favoreció la modularidad, la mantenibilidad y la escalabilidad del ecosistema construido.

Frente al **objetivo específico 3**, se construyeron módulos backend funcionales para las cuatro integraciones, incorporando procesos como sincronización de menús, recepción de pedidos, mapeo contable, creación de documentos financieros, gestión de disponibilidad, manejo de firmas digitales y ejecución de tareas programadas. Esto permitió materializar una solución operativa con aplicación práctica sobre escenarios reales de negocio.

Finalmente, en cuanto al **objetivo específico 4**, se realizaron validaciones funcionales, de integración, usabilidad y rendimiento que permitieron comprobar la estabilidad de los desarrollos construidos. Las pruebas efectuadas demostraron que las integraciones podían operar correctamente bajo condiciones reales de uso, incluyendo procesamiento de pedidos, sincronización documental, manejo de altos volúmenes de ítems y control de duplicidad en procesos contables.

A nivel personal y profesional, la pasantía representó una experiencia de gran valor, ya que permitió aplicar conocimientos de desarrollo backend, arquitectura de software, bases de datos, consumo de APIs y validación de sistemas en un contexto real de trabajo. Entre los principales retos se encontraron la diversidad de comportamientos entre plataformas externas, la necesidad de adaptar modelos de datos heterogéneos y el manejo de flujos asincrónicos. Sin embargo, estos desafíos contribuyeron al fortalecimiento de habilidades técnicas y a una mejor comprensión del desarrollo de integraciones empresariales en entornos productivos.

Finalmente, el impacto del proyecto se evidencia en la posibilidad de llevar las integraciones desarrolladas a escenarios reales de operación. La integración con Rappi permitió soportar menús de alta volumetría y operación en múltiples tiendas; PedidosYa evidenció estabilidad en la recepción de pedidos; y las integraciones contables con Siigo y QuickBooks facilitaron el procesamiento de documentos con altos volúmenes de información. Estos resultados representan un beneficio directo para los clientes, al reducir tareas manuales, mejorar la trazabilidad de la información y disminuir riesgos de error en procesos operativos y

contables. Para INVU POS, el proyecto fortalece su ecosistema tecnológico, amplía su capacidad de integración con plataformas externas y mejora su competitividad frente a soluciones que dependen de intermediarios.

# 8. ANEXOS

## 8.1. Anexo A. Requerimientos Integración Rappi

**Nota de confidencialidad**

Este anexo presenta una versión adaptada para uso académico del levantamiento de requerimientos y del análisis técnico de la integración con Rappi. Se omiten credenciales, dominios públicos, identificadores reales de tiendas, Payload productivos y detalles de infraestructura que puedan comprometer la confidencialidad de la empresa.

### 1.Propósito de la integración

La integración con Rappi tiene como finalidad sincronizar información operativa entre INVU POS y la plataforma externa de delivery. Su alcance comprende la publicación de menú, la recepción y procesamiento de pedidos, la actualización de estados operativos y el manejo de disponibilidad de productos y tiendas.

### 2.Alcance técnico general

La solución fue construida como un servicio backend independiente orientado a eventos. Desde el punto de vista técnico, la integración:

- Consulta configuraciones locales por tienda.
- Obtiene productos, categorías y modificadores desde INVU POS.
- Construye el menú en el formato exigido por Rappi.
- Publica menú y actualiza disponibilidad.
- Recibe webhooks relacionados con menú, pedidos, cancelaciones y conectividad.
- Transforma pedidos al formato requerido por INVU POS.
- Persiste estados, logs y artefactos de soporte documental.

### 3.Requerimientos funcionales oficiales

Los siguientes requerimientos funcionales fueron definidos formalmente en la tabla de requisitos de la pasantía y constituyen la base de esta especificación:

**- RF-01.** Sincronización de menú: el aplicativo deberá permitir sincronizar el menú de productos desde el administrador de INVU hacia la plataforma Rappi, asegurando que la información se mantenga actualizada.

- **RF-02.** Previsualización de menú: el sistema debe permitir visualizar el menú antes de sincronizarlo.

- **RF-03.** Gestión de productos excluidos: el sistema debe permitir excluir productos del menú final.

- **RF-04.** Recepción de webhooks: el sistema debe recibir eventos provenientes de Rappi.

- **RF-05.** Validación de webhooks: el sistema debe validar la autenticidad de los eventos mediante HMAC.

- **RF-06.** Procesamiento de ordenes: el sistema debe procesar ordenes entrantes.

- **RF-07.** Confirmar ordenes: el sistema debe confirmar ordenes en Rappi.

- **RF-08**. Rechazo de ordenes: el sistema debe permitir rechazar pedidos cuando la operación lo requiera.

- **RF-09.** Transformar pedido a INVU: el sistema debe convertir el formato de pedido de Rappi al formato requerido por INVU POS.

- **RF-10.** Enviar pedidos: el sistema debe enviar los pedidos transformados a INVU POS.

- **RF-11.** Estado de menú: el sistema debe actualizar el estado del menú según los eventos recibidos.

- **RF-12.** Consulta estado de menú: el sistema debe permitir consultar el estado actual de sincronización.

- **RF-13.** Gestión de tiendas: el sistema debe permitir administrar tiendas hijas cuando aplique.

- **RF-14.** Actualizar disponibilidad: el sistema debe actualizar la disponibilidad de artículos.

- **RF-15.** Consultar disponibilidad: el sistema debe consultar el estado de productos en la plataforma externa.

- **RF-16.** Generar PDF: el sistema debe generar un reporte de menú.

- **RF-17.** Firma digital: el sistema debe permitir firmar reportes de menú.

- **RF-18.** Simulación de menú: el sistema debe simular menú sin afectar datos reales.

- **RF-19.** Registro de logs: el sistema debe registrar eventos relevantes de la operación.

### 4.Requerimientos no funcionales oficiales

La integración también fue definida a partir de los siguientes requerimientos no funcionales:

- RNF-01. Seguridad de webhooks.

- RNF-02. Autenticación OAuth2.

- RNF-03. Persistencia de datos.

- RNF-04. Manejo de errores.

- RNF-05. Trazabilidad.

- RNF-06. Disponibilidad.

- RNF-07. Integridad de datos.

- RNF-08. Escalabilidad.

- RNF-09. Rendimiento.

- RNF-10. Configuración por entorno.

- RNF-11. Seguridad de credenciales.

- RNF-12. Interoperabilidad.

En términos técnicos, estos requerimientos implican que la integración debe validar criptográficamente los eventos entrantes, autenticarse de forma segura con la plataforma externa, mantener registro de operaciones y errores, soportar múltiples tiendas y operar en distintos ambientes sin modificar el código fuente.

### 5.Componentes técnicos involucrados

La integración se implementó como un servicio backend modular. Entre sus componentes principales se encuentran:

- Servidor HTTP para exposición de rutas.

- Modulo de consumo de APIs externas.

- Modulo de consulta de información desde INVU POS.

- Persistencia local en base de datos relacional.

- Almacenamiento de estados de menú, pedidos y logs.

- Soporte documental para generación y firma de reportes PDF.

### 6.Datos intercambiados

Desde INVU POS hacia Rappi:

- Menú

- Productos

- Categorías

- Modificadores
- Disponibilidad de artículos
- Disponibilidad de tienda
- Respuestas operativas sobre pedidos

Desde Rappi hacia el servicio local:

- Eventos de aprobación o rechazo de menú
- Nuevas ordenes
- Cancelaciones
- Eventos de tracking
- Eventos de conectividad

### 7.Consideraciones de diseño

La integración con Rappi fue concebida como un módulo orientado a eventos en tiempo real. Por ello, la base de datos local no se diseñó como un modelo altamente relacional, sino como un esquema de soporte para configuración, auditoria y trazabilidad. Esta decisión es coherente con la naturaleza asincrónica de los webhooks y con la necesidad de conservar evidencia técnica de lo procesado.

### 8.Resultado del análisis

El levantamiento de requerimientos permitió establecer que la integración con Rappi debía resolverse como un servicio bidireccional enfocado en sincronización de menú y procesamiento de pedidos, con especial énfasis en seguridad de webhooks, configuración flexible por tienda, trazabilidad operativa y soporte para flujos asincrónicos de actualización.

### Tablas de requerimientos de la integración con Rappi

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-01 | Sincronización de menú | | Esencial | |
| Descripción | El aplicativo deberá permitir sincronizar el menú de productos desde el admin de INVU hacia la plataforma Rappi, asegurando que la información este actualizada | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Licencia, StoreId, Productos | Api Invu/BD | Confirmación de sincronización | API Rappi | Requiere autenticación OAuth2 |
| Proceso | El usuario solicita la sincronización, el sistema consulta productos en INVU, transforma la información al formato requerido y la envía a Rappi. | | | |
| Efecto Colateral | Se registra el estado del menú como PENDING en la base de datos | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| RF-02 | Previsualización de menú | Alta |
| Descripción | Permite visualizar el menú antes de sincronizarlo | |

| Entradas | fuente | Salida | Destino | Restricciones |
|---|---|---|---|---|
| Token, StoreId | Usuario | JSON preview | Usuario | No guarda datos |
| Proceso | Construcción del menú sin enviarlo | | | |
| Efecto Colateral | Ninguno | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-03 | Gestión de productos excluidos | | Media | |
| Descripción | Permite excluir productos del menú | | | |
| Entradas | Fuente | Salida | Destino | Restricciones |
| Lista productos | usuario | Confirmación | BD | Debe ser arreglo |
| Proceso | Persistencia de exclusiones | | | |
| Efecto Colateral | Modificación del menú final | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| RF-04 | Recepción de webhooks | Esencial |

| | | | | |
|---|---|---|---|---|
| Descripción | El sistema deberá recibir eventos desde Rappi. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Payload | Rappi | confirmación | Sistema | Ninguna |
| Proceso | Recepción HTTP y enrutamiento | | | |
| Efecto Colateral | Activación de procesos internos | | | |

| | | | | |
|---|---|---|---|---|
| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
| Código | Nombre | | Grado de Necesidad | |
| RF-05 | validación de webhooks | | Esencial | |
| Descripción | Validar autenticidad mediante HMAC | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Headers, payload | Rappi | Ok/ERROR | sistema | Firma obligatoria |
| Proceso | Comparación de firmas | | | |
| Efecto Colateral | Rechazo (401) si falla. | | | |

| | | |
|---|---|---|
| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
| Código | Nombre | Grado de Necesidad |

| RF-06 | Procesamiento de órdenes | | Esencial | |
|---|---|---|---|---|
| Descripción | Procesar órdenes entrantes. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| orden | Rappi | Orden procesada | Sistema | Validación previa |
| Proceso | Transformación del pedido. | | | |
| Efecto Colateral | Registro en BD | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-07 | Confirmar órdenes | | Esencial | |
| Descripción | Confirmar órdenes en Rappi | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| orderId | Sistema | confirmación | Rappi | Orden valida |
| Proceso | Llamada API | | | |
| Efecto Colateral | Cambio de estado. | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales |
|---|

| Código | Nombre | | Grado de Necesidad | |
|---|---|---|---|---|
| RF-08 | Rechazo de órdenes | | Media | |
| Descripción | Permite rechazar pedidos. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| orderId, motivo | Usuario | Confirmación | Rappi | Motivo requerido |
| Proceso | Envió de cancelación. | | | |
| Efecto Colateral | Registro de rechazo. | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-09 | Transformar pedido a INVU | | Esencial | |
| Descripción | Convertir formato Rappi a INVU | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| payload | Rappi | JSON INVU | Sistema | Estructura valida |
| Proceso | Mapeo de datos | | | |
| Efecto Colateral | Ninguno | | | |

| Invu |
|---|

| SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-10 | Enviar pedidos | | Esencial | |
| Descripción | Enviar pedidos a INVU | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| pedido | Sistema | Confirmación | INVU | TOKEN<br>valido |
| Proceso | Consumo API REST | | | |
| Efecto Colateral | Registro en orders. | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-11 | Estado de menú | | Esencial | |
| Descripción | Actualizar estado del menú | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| evento | Rappi | Estado | BD | Evento valido |
| Proceso | Actualización en BD | | | |
| Efecto Colateral | Histórico | | | |

| Invu |
|---|

| SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-12 | Consulta estado de menú | | Media | |
| Descripción | Consultar estado actual | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| StoreId | Usuario | Estado | Usuario | Debe Existir |
| Proceso | Consulta BD | | | |
| Efecto Colateral | Ninguno. | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-13 | Gestión de tiendas | | Media | |
| Descripción | Administrar tiendas Hijas | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Datos tienda | usuario | Confirmación | BD | Padre<br>Requerido |
| Proceso | CRUD en BD | | | |
| Efecto Colateral | Estructura jerárquica | | | |

| Invu |
|---|

| SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-14 | Actualizar disponibilidad | | Media | |
| Descripción | Actualizar disponibilidad | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| ítems | Usuario | Confirmación | Rappi | Datos validos |
| Proceso | Envió API | | | |
| Efecto Colateral | Cambio de stock | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-15 | Consultar disponibilidad | | Media | |
| Descripción | Consultar estado de productos | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| StoreId | Usuario | Estado | Usuario | Ninguna |
| Proceso | Consulta API. | | | |
| Efecto Colateral | Ninguno. | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales |
|---|

| Código | Nombre | | Grado de Necesidad | |
|---|---|---|---|---|
| RF-16 | Generar PDF | | Baja | |
| Descripción | Generar reporte de menú | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Licencia | Usuario | PDF | Usuario | Ninguna |
| Proceso | Renderización documento | | | |
| Efecto Colateral | Almacenamiento. | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-17 | Firma digital | | Baja | |
| Descripción | Permite firmar reportes | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Datos firma | Usuario | Confirmación | BD | Firma única |
| Proceso | Registro firma | | | |
| Efecto Colateral | Bloqueo del documento | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |

| | | | | |
|---|---|---|---|---|
| RF-18 | Simulación de menú | | Baja | |
| Descripción | Simular menú sin afectar datos reales | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| parámetros | Usuario | vista | Usuario | No persistente |
| Proceso | Construcción temporal | | | |
| Efecto Colateral | Ninguno | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RF-19 | Registro de logs | | Esencial | |
| Descripción | Registrar eventos del sistema | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Datos eventos | Sistema | Registro | BD | Ninguna |
| Proceso | Persistencia de logs | | | |
| Efecto Colateral | Auditoria | | | |

**Requerimientos No funcionales de la integración con Rappi**

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RNF-01 | Seguridad de webhooks | | Esencial | |
| Descripción | El sistema deberá garantizar la autenticidad e integridad de los webhooks mediante validación criptográfica | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Headers, payload | Rappi | Validación OK/ERROR | Sistema | Uso obligatorio de HMAC SHA-256 |
| Proceso | El sistema genera un hash basado en el payload y lo compara con la firma recibida | | | |
| Efecto Colateral | Rechazo automático de solicitudes no validas (HTTP 401) | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| RNF-02 | Autenticación OAuth2 | Esencial |

| Descripción | El sistema deberá autenticarse contra la API de Rappi utilizando el protocolo OAuth2 | | | |
|---|---|---|---|---|
| Entradas | fuente | Salida | Destino | Restricciones |
| Client_id, client_secret | Variables de entorno | Access_token | API Rappi | Credenciales validas |
| Proceso | Se solicita un token de acceso y se utiliza en las peticiones | | | |
| Efecto Colateral | Renovación periódica de tokens | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RNF-03 | Persistencia de datos | | Esencial | |
| Descripción | El sistema deberá almacenar información en una base de datos relacional | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Datos del sistema | Aplicación | Registros almacenados | MySQL | Integridad de datos |
| Proceso | Inserción, actualización y consultar de registros | | | |
| Efecto Colateral | Disponibilidad de información histórica | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RNF-04 | Manejo de errores | | Esencial | |
| Descripción | El sistema deberá manejar errores y responder con códigos HTTP adecuados | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Solicitudes invalidas | Usuario/API | Códigos HTTP | Cliente | Uso de estándares HTTP |
| Proceso | Validación de datos y remoto de errores controlados | | | |
| Efecto Colateral | Mejora de depuración | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RNF-05 | Trazabilidad | | Esencial | |
| Descripción | El sistema deberá registrar todas las operaciones relevantes | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Eventos del sistema | Aplicación | logs | BD | Registro obligatorio |

| | |
|---|---|
| Proceso | Captura de eventos y almacenamiento |
| Efecto Colateral | Auditoria del sistema |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RNF-06 | Disponibilidad | | Esencial | |
| Descripción | El sistema deberá estar disponible para recibir eventos en tiempo real | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Solicitudes externas | Rappi | Respuesta del sistema | Cliente | Alta disponibilidad |
| Proceso | Atención continua de endpoints | | | |
| Efecto Colateral | Dependencia de infraestructura | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| RNF-07 | Integridad de datos | Esencial |
| Descripción | El sistema deberá validar la consistencia de los datos antes de procesarlos. | |

| Entradas | Fuente | Salida | Destino | Restricciones |
|---|---|---|---|---|
| Datos de entrada | Usuario/API | Datos validados | Sistema | Validación Obligatoria |
| Proceso | Validación previa a persistencia o envió | | | |
| Efecto Colateral | Prevención de errores | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RNF-08 | Escalabilidad | | Media | |
| Descripción | El sistema deberá soportar múltiples tiendas y alto volumen de solicitudes | | | |
| Entradas | Fuente | Salida | Destino | Restricciones |
| Múltiples solicitudes | Usuarios | Procesamiento eficiente | Sistema | Arquitectura escalable |
| Proceso | Gestión concurrente de peticiones | | | |
| Efecto Colateral | Incremento de recursos | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |

| RNF-09 | Rendimiento | | Esencial | |
|---|---|---|---|---|
| Descripción | El sistema deberá procesar ordenes en tiempos adecuados | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| ordenes | Rappi | Respuesta rápida | Sistema | Tiempo mínimo de respuesta |
| Proceso | Optimización de procesos | | | |
| Efecto Colateral | Mejor Experiencia de integración | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RNF-10 | Configuración por entorno | | Media | |
| Descripción | El sistema deberá soportar múltiples entornos (DEV, PROD) | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Variables de entorno | Sistema | Configuración aplicada | Aplicación | Variables definidas |
| Proceso | Carga de configuración según entorno | | | |
| Efecto Colateral | Flexibilidad de despliegue | | | |

| Invu |
|---|

| SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RNF-11 | Seguridad de credenciales | | Esencial | |
| Descripción | El sistema deberá proteger credenciales sensibles | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Credenciales | Variables de entorno | Uso seguro | Sistema | No exponer datos |
| Proceso | Lectura segura desde entorno | | | |
| Efecto Colateral | Reducción de riesgos | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| RNF-12 | Interoperabilidad | | Esencial | |
| Descripción | El sistema deberá integrarse correctamente con APIs externas | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Request/responses | APIs externas | Datos integrados | Sistema | Cumplir contratos API |
| Proceso | Consumo de adaptación de APIs | | | |
| Efecto Colateral | Dependencia de servicios externos | | | |

## 8.2. Anexo B – Requerimientos Integración PedidosYa

### Nota de confidencialidad

Este anexo resume la especificación técnica de la integración con PedidosYa a partir de la tabla oficial de requisitos y del análisis del servicio implementado. Se omiten credenciales, callbacks reales, identificadores operativos y detalles internos de infraestructura.

### 1. Propósito de la integración

La integración con PedidosYa tiene como objetivo centralizar la administración de catálogos, pedidos, estados operativos y reglas de disponibilidad entre INVU POS y la plataforma externa.

### 2. Alcance técnico general

La solución desarrollada permite:

- Sincronizar catálogos desde INVU POS.
- Administrar exclusiones de productos.
- Recibir pedidos desde PedidosYa.
- Transformar pedidos al formato requerido por INVU POS.
- Confirmar, rechazar, preparar y completar el ciclo operativo del pedido.
- Manejar disponibilidad de tienda y productos.
- Gestionar horarios por categoría.
- Generar reportes PDF de menú y soportar firma digital.

### 3. Requerimientos funcionales oficiales

La especificación técnica de esta integración se encuentra sustentada en los siguientes requerimientos funcionales:

- PYF-01. Sincronización de menú.
- PYF-02. Previsualización de menú.
- PYF-03. Recepción de órdenes.
- PYF-04. Procesar órdenes.
- PYF-05. Transformación a INVU.
- PYF-06. Envió de pedidos.
- PYF-07. Confirmar órdenes.
- PYF-08. Rechazar órdenes.
- PYF-09. Disponibilidad de tienda.
- PYF-10. Consulta disponibilidad.
- PYF-11. Productos excluidos.
- PYF-12. Estado de catálogo.
- PYF-13. Generación de PDF.
- PYF-14. Firma digital.
- PYF-15. Gestión de horarios.
- PYF-16. Simulación de menú.
- PYF-17. Registro de logs.

En conjunto, estos requerimientos definen una integración bidireccional orientada tanto al catálogo como al flujo completo del pedido.

**4. Requerimientos no funcionales oficiales**

Los requerimientos no funcionales definidos para esta integración son:

- PYNF-01. Autenticación con PedidosYa.
- PYNF-02. Protección de credenciales.
- PYNF-03. Persistencia en base de datos.
- PYNF-04. Manejo de errores HTTP.
- PYNF-05. Registro de logs.
- PYNF-06. Disponibilidad del sistema.
- PYNF-07. Validación de datos.
- PYNF-08. Rendimiento del sistema.
- PYNF-09. Escalabilidad.
- PYNF-10. Configuración por entorno.
- PYNF-11. Integración con APIs externas.

Desde el punto de vista técnico, esto exige que la integración mantenga disponibilidad para recepción de pedidos y callbacks, valide entradas antes de procesarlas, preserve la trazabilidad de errores y pueda operar sobre distintos ambientes sin cambios estructurales en el código.

**5. Componentes técnicos involucrados**

La solución se implementó como un servicio backend con:

- Capa HTTP para rutas operativas y administrativas.

- Módulos de consumo de API externa y de consulta de datos desde INVU POS.

- Persistencia local de clientes, catálogos, estados y pedidos.

- Almacenamiento de callbacks y registros de log.

- Soporte de simulación y documentación PDF del menú.

## 6. Datos intercambiados

Desde INVU POS hacia PedidosYa:

- Catálogos de menú

- Categorías

- Productos

- Toppings

- Horarios

- Disponibilidad de tienda

- Disponibilidad de productos

- Actualizaciones operativas de la orden

Desde PedidosYa hacia el servicio local:

- Pedidos

- Resultados de importación de catalogo

- Callbacks de estados

- Eventos operativos sobre ordenes

## 7. Consideraciones de diseño

La integración con PedidosYa combina procesos sincrónicos y asincrónicos. El catálogo se genera y se envía desde el servicio local, pero su estado final depende de callbacks externos. Asimismo, los pedidos son recibidos de forma asincrónica y luego requieren transformación, almacenamiento y notificación posterior de estados. Por ello, la base de datos local se organizó alrededor de configuración, trazabilidad y soporte operativo, y no como un modelo transaccional central del negocio.

## 8. Resultado del análisis

El análisis formal de requerimientos permitió concluir que la integración con PedidosYa debía implementarse como un módulo bidireccional con énfasis en sincronización de catálogo, recepción de pedidos, gestión de horarios, control de disponibilidad y seguimiento detallado de callbacks y estados operativos.

**Tablas de requerimientos de la integración con PedidosYa**

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYF-01 | Sincronización de menú | | Esencial | |
| Descripción | El sistema deberá sincronizar el catálogo de productos desde InvuPos hacia PedidosYa | | | |
| Entradas | fuente | Salida | Destino | Restricciones |

| | | | | |
|---|---|---|---|---|
| Licencia, restaurant Id | Usuario/BD | Confirmación de carga | Api PedidosYa | Requiere autenticación |
| Proceso | Consulta catálogo de Invu, construye estructura menú. ítems y lo envía a pedidosYa | | | |
| Efecto Colateral | Registro en menu_status | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYF-02 | Previsualización de menú | | Media | |
| Descripción | Permite visualizar el menú antes de sincronizarlo | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| token | usuario | JSON menú | Usuario | No persistente |
| Proceso | Construcción del menú sin envió | | | |
| Efecto Colateral | Ninguno | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| PYF-03 | Recepción de ordenes | Esencial |
| Descripción | El sistema deberá recibir pedidos desde PedidosYa | |

| Entradas | fuente | Salida | Destino | Restricciones |
|---|---|---|---|---|
| Payload orden | PedidosYa | Orden recibida | Sistema | Debe existir VendorId |
| Proceso | Recepción vía Webhook | | | |
| Efecto Colateral | Registro en orderPeya | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYF-04 | Procesar ordenes | | Esencial | |
| Descripción | Procesar órdenes y prepararlas para Invu | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| orden | Sistema | Orden procesada | Sistema | validación previa |
| Proceso | Transformación de datos | | | |
| Efecto Colateral | Registro en processed_orders | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| PYF-05 | Transformación a Invu | Esencial |

| | | | | |
|---|---|---|---|---|
| Descripción | Convertir pedidos al formato de InvuPos | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Payload PedidosYa | Sistema | JSON INVU | Sistema | Estructura valida |
| Proceso | Mapeo de campos | | | |
| Efecto Colateral | Ninguno | | | |

| | | | | |
|---|---|---|---|---|
| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
| Código | Nombre | | Grado de Necesidad | |
| PYF-06 | Envió de pedidos | | Esencial | |
| Descripción | Enviar pedidos al POS | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| pedido | Sistema | Confirmación | API Invu | TOKEN requerido |
| Proceso | Consumo API REST | | | |
| Efecto Colateral | Registro o retry | | | |

| | | |
|---|---|---|
| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
| Código | Nombre | Grado de Necesidad |

| | | | | |
|---|---|---|---|---|
| PYF-07 | Confirmar ordenes | | Esencial | |
| Descripción | Confirmar automáticamente pedidos en PedidosYa | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| orderId | Sistema | Confirmación | PedidosYa | Orden valida |
| Proceso | Callback a API | | | |
| Efecto Colateral | Cambio de estado | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYF-08 | Rechazar ordenes | | Baja | |
| Descripción | Permite rechazar pedidos con motivo | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| rejectionCode | Usuario | Confirmación | PedidosYa | Campo Obligatorio |
| Proceso | Envió de rechazo | | | |
| Efecto Colateral | Registro | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |

| | | | | |
|---|---|---|---|---|
| PYF-09 | Disponibilidad de tienda | | Esencial | |
| Descripción | Disponibilidad de productos | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| ítems | Usuario | Confirmación | PedidosYa | Lista valida |
| Proceso | Envió Api | | | |
| Efecto Colateral | Actualización de stock | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYF-10 | Consulta disponibilidad | | Media | |
| Descripción | Consultar productos no disponibles | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| token | Usuario | Lista ítems | Usuario | Ninguna |
| Proceso | Consulta API | | | |
| Efecto Colateral | Ninguno | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| PYF-11 | Productos excluidos | Media |

| Descripción | Gestionar exclusión de productos | | | |
|---|---|---|---|---|
| Entradas | fuente | Salida | Destino | Restricciones |
| Productos | Usuario | Confirmación | BD | Debe ser arreglo |
| Proceso | Persistencia | | | |
| Efecto Colateral | Modificación Menú | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYF-12 | Estado de catalogo | | Media | |
| Descripción | Consultar estado de importación | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| VendorID | Usuario | estado | Usuario | Ninguna |
| Proceso | Consulta BD | | | |
| Efecto Colateral | Ninguno | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| PYF-13 | Generación de PDF | Baja |

| Descripción | Generar PDF del menú | | | |
|---|---|---|---|---|
| Entradas | fuente | Salida | Destino | Restricciones |
| licencia | Usuario | PDF | Usuario | Ninguna |
| Proceso | Renderización | | | |
| Efecto Colateral | Almacenamiento | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYF-14 | Firma digital | | Baja | |
| Descripción | Permite firmar reportes | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Datos firma | Usuario | Confirmación | BD | Una sola firma |
| Proceso | Registro de firma | | | |
| Efecto Colateral | Bloqueo documento | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| PYF-15 | Gestión de horarios | Media |

| | | | | |
|---|---|---|---|---|
| Descripción | Crear, actualizar y eliminar horarios | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| horarios | usuario | confirmación | BD | Formato<br>Valido |
| Proceso | CRUD de horarios | | | |
| Efecto Colateral | Asignación a categorías | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYF-16 | Simulación de menú | | Baja | |
| Descripción | Simular menú para pruebas | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| parámetros | Usuario | vista | Usuario | No persistente |
| Proceso | Construcción temporal | | | |
| Efecto Colateral | Ninguno | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| PYF-17 | Registro de logs | Esencial |

| Descripción | Registrar eventos del sistema | | | |
|---|---|---|---|---|
| Entradas | fuente | Salida | Destino | Restricciones |
| eventos | Sistema | logs | BD | Ninguna |
| Proceso | Persistencia | | | |
| Efecto Colateral | Auditoria | | | |

Requerimientos no funcionales de la integración con PedidosYa

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYNF-01 | Autenticación con pedidosYa | | Esencial | |
| Descripción | El sistema deberá autenticarse contra los servicios de PedidosYa mediante token Bearer | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| credenciales | Variables de entorno | Access_token | API PedidosYa | Credenciales validas |
| Proceso | Solicitud de token y uso en llamadas API | | | |
| Efecto Colateral | Renovación periódica del token | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYNF-02 | Protección de credenciales | | Esenciales | |
| Descripción | Las credenciales del sistema deberán almacenarse de forma segura | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Credenciales | Variables de entorno | Uso seguro | Sistema | No exponer datos |
| Proceso | Carga desde entorno | | | |
| Efecto Colateral | Reducción de riesgos | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYNF-03 | Persistencia en base de datos | | Esencial | |
| Descripción | El sistema deberá almacenar información operativa en base de datos MySQL | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Datos sistema | Aplicación | registros | BD | Integridad |
| Proceso | Inserción y consulta | | | |
| Efecto Colateral | Histórico disponible | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYNF-04 | Manejo errores HTTP | | Esencial | |
| Descripción | El sistema deberá manejar errores y devolver códigos HTTP adecuados | | | |
| Entradas | Fuente | Salida | Destino | Restricciones |
| Solicitudes invalidas | Usuario/API | Códigos HTTP | Cliente | Estándar HTTP |
| Proceso | validación y respuesta | | | |
| Efecto Colateral | Mejor depuración | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYNF-05 | Registro de logs | | Esencial | |
| Descripción | El sistema deberá registrar todas las operaciones relevantes | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Eventos | Sistema | logs | BD | Registro obligatorio |

| | |
|---|---|
| Proceso | Persistencia de logs |
| Efecto Colateral | Auditoria |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYNF-06 | Disponibilidad del sistema | | Esencial | |
| Descripción | El sistema deberá estar disponible para recibir órdenes y eventos | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| solicitudes | PedidosYa | respuesta | Sistema | Alta disponibilidad |
| Proceso | Atención continua | | | |
| Efecto Colateral | Dependencia de infraestructura | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYNF-07 | validación de datos | | Esencial | |
| Descripción | El sistema deberá validar datos antes de procesarlos | | | |
| Entradas | fuente | Salida | Destino | Restricciones |

| | | | | |
|---|---|---|---|---|
| Datos entrada | Usuario/API | Datos validos | Sistema | validación obligatoria |
| Proceso | Verificación previa | | | |
| Efecto Colateral | Prevención de errores | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYNF-08 | Rendimiento del sistema | | Esencial | |
| Descripción | El sistema deberá procesar ordenes en tiempo adecuado | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| ordenes | PedidosYa | Respuesta rápida | sistema | Baja latencia |
| Proceso | Optimización de procesos | | | |
| Efecto Colateral | Mejor experiencia | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| PYNF-09 | Escalabilidad | Media |
| Descripción | El sistema deberá soportar múltiples tiendas y pedidos concurrentes | |

| Entradas | Fuente | Salida | Destino | Restricciones |
|---|---|---|---|---|
| Múltiples solicitudes | usuarios | procesamiento | Sistema | Arquitectura escalable |
| Proceso | Gestión concurrente | | | |
| Efecto Colateral | Mayor consumo de recursos | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| PYNF-10 | Configuración por entorno | | Media | |
| Descripción | El sistema deberá soportar múltiples entornos | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Variables entorno | sistema | configuración | aplicación | Variables definidas |
| Proceso | Carga de configuración | | | |
| Efecto Colateral | Flexibilidad | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| PYNF-11 | Integración con APIs externas | Esencial |

| Descripción | El sistema deberá integrarse correctamente con Apis Externas | | | |
|---|---|---|---|---|
| Entradas | fuente | Salida | Destino | Restricciones |
| Request/responses | APIs | Datos integrados | sistema | Cumplir contratos |
| Proceso | Consumo API | | | |
| Efecto Colateral | Dependencia externa | | | |

## 8.3. Anexo C – Requerimientos Integración Siigo

**Nota de confidencialidad**

La presente versión del anexo resume los requerimientos y lineamientos técnicos de la integración con Siigo en una forma apta para presentación académica. Se excluyen credenciales, identificadores reales, estructuras internas de infraestructura y detalles sensibles de despliegue.

**1. Propósito de la integración**

La integración con Siigo busca automatizar la transmisión de información contable y administrativa desde INVU POS hacia el sistema externo, mediante la configuración de mapeos y la ejecución de procesos automáticos programados.

**2. Alcance técnico general**

La solución permite:

- Configurar credenciales por establecimiento.

- Parametrizar mapeos de journals, facturas, compras, proveedores e impuestos.

- Sincronizar clientes.

- Consultar información operativa desde INVU POS.

- Consultar catálogos auxiliares desde Siigo.

- Generar comprobantes, facturas y órdenes de compra.

- Ejecutar procesos programados con control de concurrencia.

**3. Requerimientos funcionales oficiales**

La tabla de requisitos de la pasantía definió los siguientes requerimientos funcionales para esta integración:

- SFG-01. Configurar integración.

- SFG-02. Mapear journals.

- SFG-03. Configurar facturación.

- SFG-04. Mapear compras.

- SFG-05. Mapear proveedores.

- SFG-06. Mapear impuestos.

- SFG-07. Sincronizar clientes.

- SFG-08. Procesos automáticos.

Estos requerimientos reflejan que la integración con Siigo no se centra en pedidos o catálogos, sino en configuración contable, sincronización documental y automatización de tareas batch.

**4. Requerimientos no funcionales oficiales**

Los requerimientos no funcionales definidos para esta integración son:

- SGNF-01. Seguridad.

- SGNF-02. Disponibilidad.

- SGNF-03. Rendimiento.

Técnicamente, estos requisitos implican que el servicio debe proteger la configuración sensible, mantenerse disponible para la ejecución de tareas programadas y soportar el procesamiento de documentos de manera estable y controlada.

**5. Componentes técnicos involucrados**

La solución se implementó como un servicio backend con:

- Configuración por establecimiento.

- Persistencia local de mapeos y credenciales.

- Lógica de consulta de documentos desde INVU POS.

- Servicios de consumo de API de Siigo.

- Cache local de clientes.

- Logs de ejecución documental.

- Cron jobs con control de concurrencia.

**6. Datos intercambiados**

Desde INVU POS hacia Siigo:

- Journals

- Facturas

- Órdenes de compra

- Clientes

- Proveedores

- Productos

- Impuestos

Desde Siigo hacia el servicio local:

- Tokens de autenticación

- Tipos de documento

- Impuestos

- Usuarios

- Medios de pago

- Respuestas de validación y creación documental

**7. Consideraciones de diseño**

La integración con Siigo fue pensada como un módulo de automatización contable orientado a configuración y procesamiento programado. Por ello, la persistencia local se concentró en tablas de mapping, logs, cache de clientes y control de ejecución, en lugar de relaciones transaccionales complejas. Este diseño favorece la flexibilidad operativa, la trazabilidad y la tolerancia a reprocesos controlados.

**8. Resultado del análisis**

El levantamiento de requerimientos mostro que la integración con Siigo debía desarrollarse como un servicio unidireccional de soporte contable, con fuerte dependencia de mapeos parametrizables, ejecución automática, trazabilidad de documentos y control de rendimiento en procesos batch.

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| SFG-01 | Configurar integración | | Esencial | |
| Descripción | Registrar credenciales de Siigo | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Credenciales | Usuario | Confirmación | BD | Autorizado |
| Proceso | Guardar configuración | | | |
| Efecto Colateral | Activa integración | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| SFG-02 | Mapear journals | | Esencial | |
| Descripción | Mapeo contable | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Datos Mapping | Usuario | Guardado | BD | Código valido |
| Proceso | Registrar cuentas | | | |
| Efecto Colateral | Envía journals | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| SFG-03 | Configurar facturación | | Esencial | |
| Descripción | Configurar facturas | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Parámetros | Usuario | Guardado | BD | Datos completos |
| Proceso | Guardar config | | | |
| Efecto Colateral | Genera facturas | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| SFG-04 | Mapear compras | Esencial |
| Descripción | Mapeo Compras | |

| Entradas | fuente | Salida | Destino | Restricciones |
|---|---|---|---|---|
| Datos | Usuario | Guardado | BD | Categoría valida |
| Proceso | Guardar mapping | | | |
| Efecto Colateral | Envía compras | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| SFG-05 | Mapear proveedores | | Esencial | |
| Descripción | Relación proveedores | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Correos | Usuario | Guardado | BD | Único |
| Proceso | Guardar mapping | | | |
| Efecto Colateral | Mejora relación | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| SFG-06 | Mapear impuestos | Esencial |
| Descripción | Relación impuestos | |

| Entradas | fuente | Salida | Destino | Restricciones |
|---|---|---|---|---|
| IDs | Usuario | Guardado | BD | Permisos |
| Proceso | Guardar mapping | | | |
| Efecto Colateral | Aplica impuestos | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| SFG-07 | Sincronizar clientes | | Esencial | |
| Descripción | Sync clientes | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| ID | Sistema | Lista | BD | Config valida |
| Proceso | Consulta API | | | |
| Efecto Colateral | Mejora rendimiento | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| SFG-08 | Procesos automáticos | Esencial |

| Descripción | Automatización | | | |
|---|---|---|---|---|
| Entradas | fuente | Salida | Destino | Restricciones |
| Datos | Sistema | resultado | Siigo | Sin concurrencia |
| Proceso | Ejecutar cron | | | |
| Efecto Colateral | Logs generados | | | |

No funcionales

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| SGNF-01 | Seguridad | | Esencial | |
| Descripción | Requisito no funcional de seguridad | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| ninguna | Sistema | Cumplimiento | Sistema | General |
| Proceso | Aplicar seguridad | | | |
| Efecto Colateral | Sistema estable | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| SGNF-02 | Disponibilidad | | Esencial | |
| Descripción | Requisito no funcional de disponibilidad | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| ninguna | Sistema | Cumplimiento | Sistema | General |
| Proceso | Aplicar disponibilidad | | | |
| Efecto Colateral | Sistema estable | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| SGNF-03 | Rendimiento | | Esencial | |
| Descripción | Requisito no funcional de rendimiento | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| ninguna | Sistema | Cumplimiento | Sistema | General |

| | |
|---|---|
| Proceso | Aplicar rendimiento |
| Efecto Colateral | Sistema estable |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| SGNF-01 | Seguridad | | Esencial | |
| Descripción | Requisito no funcional de seguridad | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| ninguna | Sistema | Cumplimiento | Sistema | General |
| Proceso | Aplicar seguridad | | | |
| Efecto Colateral | Sistema estable | | | |

## 8.4. Anexo D – Requerimientos integración QuickBooks

**Nota de confidencialidad**

Este anexo resume la especificación técnica de la integración con QuickBooks a partir de los requerimientos oficiales definidos durante la pasantía y del análisis del servicio

implementado. Se omiten credenciales, identificadores reales de empresa, tokens, callbacks reales y detalles internos de infraestructura.

**1. Propósito de la integración**

La integración con QuickBooks tiene como objetivo automatizar el registro financiero y comercial de operaciones generadas en INVU POS, mediante autenticación segura, configuración de mapeos y procesamiento automático de documentos.

**2. Alcance técnico general**

La solución desarrollada permite:

- Conectar una empresa a través de OAuth 2.0.
- Renovar tokens de acceso automáticamente.
- Consultar journals, ventas POS y órdenes de compra desde INVU POS.
- Configurar mapeos contables y comerciales.
- Generar Journal Entries, Purchase Orders y Sales Receipts.
- Consultar catálogos auxiliares de QuickBooks.
- Crear proveedores, clientes e impuestos cuando sea necesario.
- Registrar resultados y errores de sincronización.

**3. Requerimientos funcionales oficiales**

La tabla de requisitos formal definió para esta integración los siguientes requerimientos funcionales:

- QBF-01. Autenticación OAuth 2.0.
- QBF-02. Renovación automática de tokens de acceso.
- QBF-03. Consulta de journals de INVU POS.
- QBF-04. Configuración de mapeo contable de journals.
- QBF-05. Creación automática de Journal Entries en QuickBooks.
- QBF-06. Configuración de mapeo de órdenes de compra.
- QBF-07. Creación automática de Purchase Orders en QuickBooks.
- QBF-08. Configuración de mapeo de ventas POS (SalesReceipt).
- QBF-09. Creación automática de Sales Receipts en QuickBooks.
- QBF-10. Consulta de catálogos de QuickBooks.
- QBF-11. Creación de Vendor en QuickBooks desde INVU.
- QBF-12. Creación de Customer en QuickBooks desde INVU.
- QBF-13. Sincronización de impuestos INVU a QuickBooks.
- QBF-14. Verificación de estado de conexión con QuickBooks.
- QBF-15. Registro de trazabilidad de operaciones.

Estos requerimientos muestran que la integración fue concebida como un módulo contable y comercial con fuerte dependencia de autenticación segura, configuración de mapping y procesos automáticos controlados.

**4. Requerimientos no funcionales oficiales**

Los requerimientos no funcionales definidos formalmente para esta integración son:

- QBNF-01. Protección CSRF en formularios web.
- QBNF-02. Rate limiting y reintentos hacia QuickBooks.
- QBNF-03. Control de concurrencia en procesos programados.
- QBNF-04. Gestión segura de credenciales.
- QBNF-05. Disponibilidad del servicio de integración.
- QBNF-06. Manejo estandarizado de errores HTTP.
- QBNF-07. Compatibilidad con entornos sandbox y producción.
- QBNF-08. Persistencia relacional en MySQL.
- QBNF-09. Rendimiento en procesamiento batch.

Desde el punto de vista técnico, estos requisitos implican que la integración debe proteger formularios y credenciales, respetar límites de consumo de la API externa, sostener procesos automáticos sin ejecuciones concurrentes y mantener persistencia confiable de tokens, mapping y logs.

**5. Componentes técnicos involucrados**

La integración fue implementada como un servicio backend con:

- Interfaz administrativa web.
- Rutas HTTP para configuración y consulta.
- Servicios de autenticación y consumo de QuickBooks.
- Servicios de consulta de información desde INVU POS.
- Persistencia relacional para mapping, tokens y logs.

- Procesos programados para journals, órdenes de compra y ventas POS.

- Mecanismos de bloqueo para prevenir concurrencia en cron Jobs.

**6. Datos intercambiados**

Desde INVU POS hacia QuickBooks:

- Journals

- Ventas POS

- Órdenes de compra

- Clientes

- Proveedores

- Impuestos

- Ítems auxiliares

Desde QuickBooks hacia el servicio local:

- Tokens de autenticación

- Catálogos de cuentas

- Clientes

- Proveedores

- Métodos de pago

- Impuestos

- Respuestas de validación y creación documental

**7. Consideraciones de diseño**

La integración con QuickBooks presenta una complejidad superior en materia de autenticación, mantenimiento de sesión y control de volumen de procesamiento. Por esta razón, el diseño local se apoyó en tablas de configuración, mapping, logs y control de cron Jobs, privilegiando la trazabilidad y la flexibilidad operativa frente a un modelo altamente relacional.

**8. Resultado del análisis**

El levantamiento de requerimientos permitió concluir que la integración con QuickBooks debía desarrollarse como un módulo unidireccional de automatización financiera, con autenticación avanzada, compatibilidad por entorno, control de concurrencia, soporte para procesamiento batch y persistencia confiable de configuración y resultados.

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBF-01 | Autenticación OAuth 2.0 | | Esencial | |
| Descripción | El sistema debe permitir iniciar y completar el flujo de autenticación OAuth 2.0 contra la plataforma Intuit para obtener acceso autorizado a una empresa de QuickBooks Online | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Licencia, credenciales usuario qb | Usuario administrador atraves de la interfaz web | Token de acceso, Refresh token, | BD | Requiere credenciales del App |

| | | | | |
|---|---|---|---|---|
| | | Realm_id<br>Fecha de expiración de los tokens | | creada en Intuit |
| Proceso | El usuario accede a la vista parametrizada por Licencia.<br>Si no existe conexión vigente, el sistema redirige a /auth/invu/: Licencia.<br>QuickBooks presenta pantalla de autorización.<br>Tras autorizar, QuickBooks redirige al callback.<br>5. El sistema intercambia el código por tokens y los persiste. | | | |
| Efecto Colateral | Creación de usuario en BD | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| QBF-02 | Renovación automática de tokens de acceso | Esencial |

| | | | | |
|---|---|---|---|---|
| Descripción | El sistema debe renovar automáticamente el Access_token de QuickBooks antes de que expire mediante el uso del Refresh token almacenado | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Refresh token almacenado en BD | BD | Token de acceso y Refresh actualizados | BD | En caso de que el Refresh expire se requiere autenticación manual |
| Proceso | Antes de cada llamada a la API de QuickBooks, el sistema verifica la vigencia del token.<br>Si está próximo a expirar, se refresca.<br>Almacena los nuevos tokens en base de datos.<br>4. Si falla, registra el error en BD. | | | |
| Efecto Colateral | Tokens anteriores quedan invalidados | | | |

| Invu |
|---|

<table>
<tr><th colspan="5">SRS – Especificaciones de Requerimientos Funcionales</th></tr>
<tr><td>Código</td><td colspan="2">Nombre</td><td colspan="2">Grado de Necesidad</td></tr>
<tr><td>QBF-03</td><td colspan="2">Consulta de journals de INVU POS</td><td colspan="2">Esencial</td></tr>
<tr><td>Descripción</td><td colspan="4">El sistema debe consultar los asientos contables (journals) generados por INVU POS dentro de un rango de fechas especificado.</td></tr>
<tr><td>Entradas</td><td>fuente</td><td>Salida</td><td>Destino</td><td>Restricciones</td></tr>
<tr><td>Licencia, fecha inicial y final</td><td>APIs INVU</td><td>Listado json con datos solicitados</td><td>Interfaz web del administrador o proceso de sincronización automática</td><td>Rango de fechas valido, existir en INVU</td></tr>
<tr><td>Proceso</td><td colspan="4">El usuario o proceso automático solicita journals vía /api/invu/: licencia/journals.<br>El sistema traduce parámetros y consulta la API de INVU.<br>3. Se retorna la respuesta normalizada.</td></tr>
<tr><td>Efecto Colateral</td><td colspan="4">Ninguno</td></tr>
</table>

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBF-04 | Configuración de mapeo contable de journals | | Esencial | |
| Descripción | El sistema debe permitir al usuario configurar la correspondencia entre códigos de journal de INVU y cuentas contables de QuickBooks (débito y crédito), incluyendo asignación de clientes y proveedores. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Licencia, mapeo por código de cada journal, y configuración relacionada a este | Formulario de vista | BD | BD | Cuentas existentes y activas en QB |
| Proceso | El usuario accede a la vista de integración.<br>Selecciona cuentas de débito y crédito para cada código de journal.<br>Envía el formulario vía POST a /mapping/: licencia.<br>4. El sistema serializa y persiste el mapeo. | | | |

| | |
|---|---|
| Efecto Colateral | Sobre escribe configuración en BD para esta licencia |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBF-05 | Creación automática de Journal Entries en QuickBooks | | Esencial | |
| Descripción | El sistema debe generar automáticamente asientos contables (JournalEntry) en QuickBooks Online a partir de los journals de INVU, utilizando la configuración de mapeo definida. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Journals de INVU, configuración guardada en BD, tokens de QB vigentes | API de INVU, BD | JournalEntry creado correctamente en QB | QuickBooks Online, BD | Monto débito y crédito debe estar balanceado, las licencias deben haber habilitado la opción para |

| | | | | |
|---|---|---|---|---|
| | | | | crear comprobantes de forma automática, se usa rate limiting |
| Proceso | El proceso cron procesar Diario se ejecuta según programación. 2. Consulta ubicaciones habilitadas. Obtiene journals de INVU. Transforma cada journal según channel_mapping. Valida balance contable. Envía JournalEntry a QuickBooks. 7. Registra resultado en voucher_log. | | | |
| Efecto Colateral | Se crean transacciones contables en QuickBooks, se generar registros de auditoria en BD | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| QBF-06 | Configuración de mapeo de órdenes de compra | Alta |

| Descripción | El sistema debe permitir configurar la correspondencia entre proveedores/categorías de INVU y entidades de QuickBooks para la generación de órdenes de compra (PurchaseOrder). | | | |
|---|---|---|---|---|
| Entradas | fuente | Salida | Destino | Restricciones |
| Licencia, mapeo por categoría con id del cliente, impuesto y cuenta | Interfaz | Actualización campo de mapeo de orden en BD | BD campo de mapeo para órdenes de compra | Datos de entrada deben existir en QuickBooks |
| Proceso | El usuario accede a /orders/: licencia.<br>Configura mapeo por categoría de proveedor.<br>Envía vía POST a /orders/: licencia/mapping.<br>4. El sistema persiste el mapeo. | | | |
| Efecto Colateral | Se sobrescribe la configuración previa del campo de mapeo de ordenes | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |

| | | | | |
|---|---|---|---|---|
| QBF-07 | Creación automática de Purchase Orders en QuickBooks | | Alta | |
| Descripción | El sistema debe generar automáticamente órdenes de compra en QB a partir de los mapeos configurados y las ordenes de compras obtenidas desde INVU | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Órdenes de compra de INVU, campo de mapeo configurado y tokens de acceso vigentes | API INVU | PurchaseOrder creada en QuickBooks | QuickBooks Online | Requiere campo de mapeo configurado |
| Proceso | El proceso cronOrder.js se ejecuta periódicamente.<br>Consulta órdenes de INVU.<br>Transforma según mapeo de órdenes de compra.<br>4. Crea PurchaseOrder en QuickBooks. | | | |
| Efecto Colateral | Creación de órdenes de compra en QuickBooks | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBF-08 | Configuración de mapeo de ventas POS (SalesReceipt) | | Esencial | |
| Descripción | El sistema debe permitir configurar la correspondencia entre métodos de pago de INVU y parámetros de QuickBooks para la generación de recibos de venta (SalesReceipt). | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Licencia, método de pago, cuenta de depósito, cuenta de ingreso, cliente e impuesto de QB | Interfaz | Mapeo guardado en BD | Bd campo mapeo de facturas | Las entidades guardadas en el mapeo deben existir previamente en QB |
| Proceso | El usuario accede a /bill/: licencia.<br>Selecciona un tipo de pago.<br>Configura cuentas, método de pago, cliente e impuesto. | | | |

| | |
|---|---|
| | 4. Guarda la configuración. |
| Efecto Colateral | Se crea/actualizada la configuración del método de pago indicado |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBF-09 | Creación automática de Sales Receipts en QB | | Esencial | |
| Descripción | El sistema debe generar automáticamente recibos de venta (SalesReceipt) en QuickBooks Online a partir de las ventas POS de INVU. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Ventas POS de INVU, configuración del campo de mapeos de facturas, | API de INVU, BD campo mapeo facturas | Factura creada en QB con referencia al ID de orden de invu | QuickBooks online (API de intuit) | Requiere configuración completa para el tipo de pago de la venta, se |

| | | | | |
|---|---|---|---|---|
| tokens de QB vigentes | | | | aplica rate limiting |
| Proceso | El proceso cronSalesReceipt.js se ejecuta periódicamente. Consulta ventas de INVU. 3. Transforma según el campo de mapeo de facturas. Construye payload de SalesReceipt. 5. Envía a QuickBooks. | | | |
| Efecto Colateral | Se crean transacciones de venta en QuickBooks, Se incluye PrivateNote con id de la orden para trazabilidad | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBF-10 | Consulta de catálogos de QuickBooks | | Esencial | |
| Descripción | El sistema debe consultar y presentar los catálogos maestros de QuickBooks (cuentas, clientes, proveedores, métodos de pago, ítems e impuestos) para facilitar la configuración de mapeos. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |

| | | | | |
|---|---|---|---|---|
| Licencia con conexión QB vigente | Api de QB Online | Listados normalizados de cuentas, clientes, proveedores, etc.… | Interfaz de mapeo | Requiere token de acceso vigente. Solo se retornan entidades activas |
| Proceso | El sistema recibe solicitud GET en el endpoint correspondiente.<br>2. Verifica/renueva token.<br>Consulta el catálogo en QuickBooks.<br>4. Normaliza la respuesta. 5. Retorna JSON. | | | |
| Efecto Colateral | Ninguna | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| QBF-11 | Creación de Vendor en QuickBooks desde INVU | Alta |

| | | | | |
|---|---|---|---|---|
| Descripción | El sistema debe permitir crear un proveedor (Vendor) en QuickBooks Online a partir de los datos de un proveedor existente en INVU. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Licencia, datos del proveedor INVU | Interfaz | Proveedor creado en QuickBooks | QuickBooks online | No debe existir un proveedor con los mismos datos |
| Proceso | El usuario selecciona un proveedor de INVU.<br>Envía solicitud POST a /others/vendor/create/: licencia.<br>3. El sistema transforma los datos y crea el Vendor en QuickBooks. 4. Retorna confirmación. | | | |
| Efecto Colateral | Creación nueva entidad Vendor en QuickBooks | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |

| | | | | |
|---|---|---|---|---|
| QBF-12 | Creación de Customer en QuickBooks desde INVU | | Alta | |
| Descripción | El sistema debe permitir crear un cliente(customer) en QB a partir de los datos existentes en invu | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Licencia, datos del cliente INVU | Interfaz | Cliente creado en QB | QB online | No debe existir un cliente con los mismos datos en QB |
| Proceso | El usuario selecciona un cliente de INVU.<br>Envía solicitud POST a /others/customer/create/: licencia.<br>El sistema transforma los datos y crea el Customer en QuickBooks.<br>4. Retorna confirmación. | | | |
| Efecto Colateral | Creación nueva entidad Customer en QB | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBF-13 | Sincronización de impuestos INVU a QuickBooks | | Alta | |
| Descripción | El sistema debe permitir sincronizar un código de impuesto de INVU como TaxCode/TaxService en QuickBooks Online. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Licencia, datos impuestos INVU | Interfaz web | Tax code creado en QuickBooks | QuickBooks Online | Si el impuesto ya existe en QuickBooks, se retorna el existente sin duplicar |
| Proceso | El usuario selecciona un impuesto de INVU.<br>Envía solicitud POST a /others/tax/create/: licencia.<br>3. El sistema verifica existencia y crea o retorna el TaxCode. | | | |
| Efecto Colateral | Creación entidad TaxCode en QuickBooks | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBF-14 | Verificación de estado de conexión con QuickBooks | | Esencial | |
| Descripción | El sistema debe verificar si existe un token vigente de QuickBooks para una ubicación externa, indicando el estado de la conexión. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Licencia | Solicitud GET del cliente web o proceso interno | JSON con connected, realm_id y expiresAt | Interfaz | Ninguna |
| Proceso | Se recibe solicitud en /api/qb/status/: licencia.<br>Se consulta BD.<br>Se evalúa vigencia del token.<br>4. Se retorna estado. | | | |
| Efecto Colateral | Ninguno | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBF-15 | Registro trazabilidad de operaciones | | Esencial | |
| Descripción | El sistema debe registrar en tablas los resultados de cada operación de sincronización, incluyendo éxitos y errores | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Resultado de las operaciones, identificador de entidad, detalles del error | Procesos automáticos de sincronización(cron) | Registro insertado en la tabla de log | BD, tablas de registros | Ninguna |
| Proceso | Al finalizar cada operación de sincronización, el sistema construye el registro de log.<br>2. Inserta en la tabla correspondiente con timestamp. | | | |

| | |
|---|---|
| Efecto Colateral | Crecimiento de tablas de log que debe gestionarse periódicamente. |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBNF-01 | Protección CSRF en formularios web | | Esencial | |
| Descripción | Todas las solicitudes POST originadas desde formularios web deben estar protegidas contra ataques Cross-Site Request Forgery (CSRF) mediante tokens de validación. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Token CSRF generado por el servidor e | Middleware csurf de Express. | Validación exitosa o rechazo con estado | Todas las rutas POST del | Las cookies deben configurarse con httpOnly, sameSite=lax y secure en producción. |

| | | | | |
|---|---|---|---|---|
| incluido en cada formulario. | | HTTP 403. | sistema. | |
| Proceso | El servidor genera un token CSRF por sesión.<br>El token se incluye en formularios como campo oculto.<br>En cada POST, el middleware valida el token.<br>4. Si es inválido, se rechaza con 403. | | | |
| Efecto Colateral | Solicitudes sin token CSRF válido son rechazadas automáticamente. | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| QBNF-02 | Rate limiting y reintentos hacia QuickBooks | Esencial |
| Descripción | El sistema debe respetar los límites de tasa de la API de QuickBooks Online, implementando estrategias de retry con | |

| | | | | |
|---|---|---|---|---|
| | backoff exponencial ante respuestas HTTP 429 (Too Many Requests). | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Respuesta HTTP de QuickBooks con código 429 o 5xx. | API de QuickBooks Online. | Reintento automático tras espera o registro de error si se agotan los intentos. | Proceso invocante y tablas de log. | Máximo número de reintentos configurable. El backoff debe ser exponencial para evitar saturación. |
| Proceso | Se envía solicitud a QuickBooks.<br>Si retorna 429, el sistema espera según backoff.<br>Reintenta hasta agotar intentos.<br>4. Si persiste el error, registra y notifica. | | | |
| Efecto Colateral | Puede incrementar la latencia de operaciones individuales durante períodos de throttling. | | | |

| Invu |
|---|
| SRS – Especificaciones de Requerimientos Funcionales |

| Código | Nombre | | Grado de Necesidad | |
|---|---|---|---|---|
| QBNF-03 | Control de concurrencia en procesos programados | | Esencial | |
| Descripción | Los procesos programados (cron) deben implementar mecanismos de bloqueo distribuido para evitar ejecuciones concurrentes del mismo proceso | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Nombre del proceso e identificador de instancia | Proceso cron al iniciar ejecución | Bloqueo adquirido o ejecución omitida | Tabla de bloqueo de instancia | Los bloqueos deben tener tiempo de expiración para evitar bloqueos huérfanos |
| Proceso | El proceso cron intenta adquirir bloqueo en cron_locks.<br>Si el bloqueo existe y no ha expirado, la ejecución se omite.<br>Si se adquiere, se ejecuta el proceso.<br>4. Al finalizar, se libera el bloqueo. | | | |
| Efecto Colateral | En caso de fallo sin liberación, el bloqueo expira automáticamente | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBNF-04 | Gestión segura de credenciales | | Esencial | |
| Descripción | Las credenciales de acceso a servicios externos (QuickBooks, base de datos) deben gestionarse mediante variables de entorno y no estar presentes en el código fuente. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Variables de entorno<br>Client_id<br>client_secret<br>Redirect_uri | Archivo de variables de entorno | Configuración del entorno de despliegue | Servicios que requieren autenticación | Los tokens no deben exponerse en logs de producción |
| Proceso | Al iniciar la aplicación, dotenv carga variables de entorno.<br>2. Los servicios acceden a credenciales a través de process.env. | | | |
| Efecto Colateral | Si las variables no están definidas, los servicios dependientes fallan al iniciar. | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBNF-05 | Disponibilidad del servicio de integración | | Esencial | |
| Descripción | El servicio de integración debe estar disponible de forma continua (24/7) para garantizar la ejecución oportuna de los procesos de sincronización programados y el acceso a la interfaz administrativa | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| ninguno | Infraestructura de despliegue | Servicio activo y respondiendo solicitudes HTTP | Usuarios administrativos y procesos cron | La disponibilidad depende del servidor Node.js, la base de datos MySQL y la conectividad con las |

| | | | | |
|---|---|---|---|---|
| | | | | APIs externas. |
| Proceso | El servicio Express se inicia y escucha en el puerto configurado. Los procesos cron se registran al arrancar la aplicación. | | | |
| Efecto Colateral | Una caída del servicio detiene la sincronización automática hasta su reinicio. | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBNF-06 | Manejo estandarizado de errores HTTP | | Alta | |
| Descripción | El sistema debe retornar códigos HTTP estándar y mensajes descriptivos para todos los escenarios de error: 400 (validación), 403 (no autorizado/CSRF), 404 (no encontrado) y 500 (error interno). | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Excepciones o condiciones de error durante el procesamiento. | Cualquier capa del sistema (rutas, | Respuesta HTTP con código apropiado | Cliente HTTP (navegador o proceso). | Los mensajes de error no deben |

| | | | | |
|---|---|---|---|---|
| | servicios, base de datos). | y mensaje descriptivo en JSON o redirección con parámetro de error. | | exponer información sensible del sistema (stack traces, credenciales, rutas internas). |
| Proceso | Se detecta error en cualquier capa.<br>Se clasifica el tipo de error.<br>3. Se construye respuesta con código HTTP y mensaje apropiados. 4. Se retorna al cliente. | | | |
| Efecto Colateral | Ninguno | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |
| QBNF-07 | Compatibilidad con entornos sandbox y producción | Alta |

| | | | | |
|---|---|---|---|---|
| Descripción | El sistema debe soportar la configuración de entorno (sandbox o producción) de QuickBooks Online mediante parámetros, sin requerir cambios en el código fuente. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Variable de entorno environment (sandbox o production) | Configuración del entorno de despliegue. | URLs base de la API de QuickBooks apuntando al entorno correcto. | Cliente OAuth y llamadas a la API de Intuit. | Las URLs base de sandbox y producción deben estar correctamente parametrizadas. |
| Proceso | Al iniciar, el sistema lee QB_ENVIRONMENT.<br>Configura el cliente OAuth con el entorno indicado.<br>3. Todas las llamadas usan la URL base correspondiente. | | | |
| Efecto Colateral | Un valor incorrecto de entorno dirige las llamadas al endpoint equivocado. | | | |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | |
|---|---|---|
| Código | Nombre | Grado de Necesidad |

| QBNF-08 | Persistencia relacional en MySQL | | Esencial | |
|---|---|---|---|---|
| Descripción | Toda la configuración, tokens, mapeos y registros de auditoría deben persistirse en una base de datos MySQL con soporte para conexiones asíncronas mediante mysql2/promise | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Datos de configuración, tokens, resultados de operaciones. | Capa de servicios. | Registros almacenados en las tablas correspondientes. | Base de datos MySQL | La base de datos debe estar accesible desde el servidor de la aplicación. Se debe usar pool de conexiones para eficiencia. |
| Proceso | Las operaciones de lectura/escritura se realizan mediante consultas parametrizadas a través del módulo services/db.js. | | | |

| | |
|---|---|
| Efecto Colateral | Indisponibilidad de MySQL impide el funcionamiento completo del sistema. |

| Invu<br>SRS – Especificaciones de Requerimientos Funcionales | | | | |
|---|---|---|---|---|
| Código | Nombre | | Grado de Necesidad | |
| QBNF-09 | Rendimiento en procesamiento batch | | alta | |
| Descripción | Los procesos de sincronización automática deben ser capaces de procesar lotes de transacciones de forma eficiente, respetando los límites de tasa de las APIs externas sin comprometer la completitud del procesamiento. | | | |
| Entradas | fuente | Salida | Destino | Restricciones |
| Lote de transacciones de INVU para un período determinado. | API de INVU POS. | Transacciones procesadas en QuickBooks con registro de éxitos y fallos. | QuickBooks Online y tablas de log. | Debe respetar rate limits de QuickBooks (máximo ~500 solicitudes por minuto). El procesamiento |

| | | | | |
|---|---|---|---|---|
| | | | | de un lote no debe bloquear la interfaz web. |
| Proceso | Los procesos cron procesan transacciones secuencialmente con pausas entre llamadas a la API para respetar rate limits. | | | |
| Efecto Colateral | Lotes grandes pueden extender el tiempo de ejecución del proceso cron. | | | |

# 9. Referencias

Express.js. (s. f.). Installing. Recuperado el 13 de mayo de 2026, de https://expressjs.com/en/starter/installing.html

Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures. University of California, Irvine. https://roy.gbiv.com/pubs/dissertation/top.htm

Hardt, D. (2012). The OAuth 2.0 authorization framework (RFC 6749). Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/rfc6749

Intuit Developer. (s. f.). Set up OAuth 2.0. Recuperado el 13 de mayo de 2026, de https://developer.intuit.com/app/developer/qbpayments/docs/develop/authentication-and-authorization/oauth-2.0

Intuit Developer. (s. f.). What you can do with the QuickBooks Online Accounting API. Recuperado el 13 de mayo de 2026, de https://developer.intuit.com/app/developer/qbo/docs/learn/explore-the-quickbooks-online-api

Node.js. (2026). About this documentation. https://nodejs.org/api/documentation.html

Oracle. (2026). MySQL reference manual. https://dev.mysql.com/doc/refman/en/

PedidosYa Developers. (s. f.). Manage incoming orders - Real-time order processing with the Partner API. Recuperado el 13 de mayo de 2026, de https://developer.pedidosya.com/en/documentation/manage-incoming-orders

Rappi. (s. f.). Introducción. Recuperado el 13 de mayo de 2026, de https://dev-portal.rappi.com/es/api-reference/

Siigo API. (s. f.). Introducción. Recuperado el 13 de mayo de 2026, de https://developers.siigo.com/docs/siigoapi