UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE
DIVISIÓN DE CIENCIAS DE LA INGENIERÍA
INGENIERÍA EN CIENCIAS Y SISTEMAS

ANALISIS Y DISENO DE SISTEMAS 2

MANUAL TÉCNICO BACKEND

Quetzaltenango 15 de mayo de 2025

202030542 - Fernando José Rodríguez Ramírez
202031794 - Luis Antonio Monterroso Guzman
201931012 - Carlos Benjamín Pac Flores

Descripción General

Este documento proporciona una introducción exhaustiva al sistema Backend del Proyecto 2, una robusta aplicación basada en microservicios diseñada para gestionar operaciones empresariales. Describe la arquitectura del sistema, los componentes principales y cómo interactúan para formar una plataforma cohesionada.

Objetivo y alcance

El sistema Backend del Proyecto 2 es una plataforma modular de gestión empresarial construida según los principios de la arquitectura de microservicios. Proporciona funcionalidad para:

- Gestión y autenticación de empleados
- Control de productos e inventarios
- Reservas y seguimiento de juegos
- Generación de facturas
- Informes y análisis empresariales

Esta descripción general se centra en la arquitectura de alto nivel del sistema y en las relaciones entre los servicios. Para obtener información más detallada sobre componentes específicos, consulte las secciones correspondientes de la documentación.

Arquitectura

El sistema Backend del Proyecto 2 sigue un patrón de arquitectura de microservicios, con una clara separación de responsabilidades entre los diferentes componentes funcionales.

Cada microservicio es una aplicación Spring Boot independiente que se puede desarrollar, desplegar y escalar por separado.

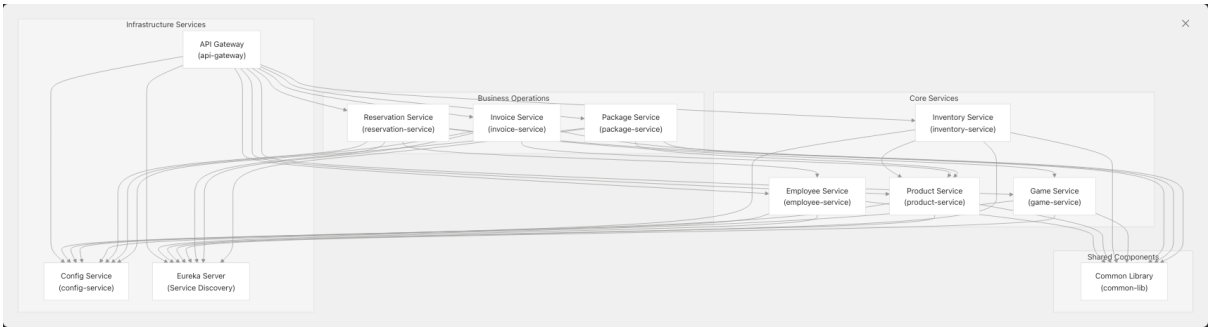


Diagrama de la arquitectura de los microservicios

Tecnologías Usadas

Componente	Tecnología

Runtime	Java 17
Framework	Spring Boot 3.4.5
Microservicios	Spring Cloud 2024.0.1
Descubrimiento	Eureka
Base de datos	MySQL
ORM	Spring Data JPA
Seguridad	Spring Security + JWT
Documentación API	Swagger / OpenAPI
Generación de Código	Lombok, MapStruct
Testing	JUnit, Mockito
Cobertura de Código	JaCoCo
Build Tool	Maven

Componentes Compartidos

- Servicios de Operación del Negocio
- Servicios de Negocio Principal
- Servicios de Infraestructura

Aplicaciones Principales

- ConfigServiceApplication
- EurekaServerApplication
- ApiGatewayApplication
- EmployeeServiceApplication
- ProductServiceApplication
- InventoryServiceApplication
- GameServiceApplication

- ReservationServiceApplication
- InvoiceServiceApplication
- PackageServiceApplication
- ReportsServiceApplication
- Common Library

Organización de Servicios

El sistema está organizado en grupos de servicios que se enfocan en dominios específicos del negocio:

Grupo de Servicio	Descripción	Servicios
Servicios de Infraestructura	Servicios base para descubrimiento, configuración y enrutamiento	Config Service, Eureka Server, API Gateway
Servicios de Negocio Principal	Gestionan entidades principales del negocio	Employee Service, Product Service, Inventory Service, Game Service
Servicios de Operación	Gestionan procesos de negocio complejos que combinan varios servicios base	Reservation Service, Invoice Service, Package Service, Reports Service
Componentes Compartidos	Código reutilizable entre todos los servicios	Common Library

Servicios de Infraestructura

Proveen capacidades fundamentales como descubrimiento de servicios, configuración centralizada y acceso vía API Gateway.

- **Config Service**:

  Administrar configuraciones centralizadas.

  - Almacén central de configuraciones.
  - Permite actualizaciones dinámicas sin reinicio.

    - Soporte para perfiles específicos por entorno.

- **Eureka Server**:

  Descubrimiento de servicios.

    - Registro y descubrimiento dinámico.
    - Monitoreo de salud.
    - Balanceo de carga.

- **API Gateway**:

  Punto de entrada único para solicitudes.

    - Enrutamiento dinámico.
    - Validación de autenticación y autorización.
    - Transformación de peticiones y respuestas.
    - Control de carga y tolerancia a fallos.

Servicios de Negocio Principal

Administran las entidades fundamentales del sistema. Cada servicio tiene su propia lógica y base de datos.

- **Employee Service**:
    - CRUD de empleados.
    - Gestión de roles.
    - Autenticación y emisión de JWT.
    - Historial de acciones.
- **Product Service**:
    - CRUD de productos.
    - Categorización y atributos.

    - Precios y búsqueda avanzada.

- **Inventory Service**:
    - Gestión de inventario y almacenes.
    - Transacciones de entrada/salida.
    - Reportes de stock.
- **Game Service**:
    - CRUD de juegos.
    - Gestión de jugadores.
    - Resultados y programación.

Servicios de Operación del Negocio

Implementan flujos de trabajo complejos, integrando múltiples servicios principales.

- **Reservation Service**:
    - Crear y gestionar reservaciones.
    - Estados: confirmado, cancelado, finalizado.
    - Coordinación con juegos y facturación.
    - Generación de QR.
- **Invoice Service**:
    - Gestión de facturación.
    - Procesamiento de pagos.
    - Cálculo de impuestos.
- **Package Service**:
    - Gestión de combos de productos.

    - Configuración y promociones.

- **Reports Service**:
    - Generación de reportes PDF con QR.
    - Análisis de ingresos y horarios.
    - Visualización de datos.

Flujo de Comunicación

El flujo central del sistema involucra múltiples servicios trabajando en conjunto, como se muestra a continuación:

1. Autenticación: JWT emitido por Employee Service
2. Solicitud de reservación con JWT
3. Creación del juego
4. Generación de factura
5. Generación de código QR
6. Confirmación y detalles de reservación

Biblioteca Común

Provee componentes reutilizables como:

Resiliencia y Escalabilidad

La arquitectura incluye características para asegurar resiliencia:

- **Descubrimiento de servicios** dinámico con Eureka
- **API Gateway** como punto de control y balanceo
- **Circuit Breakers** para evitar fallos en cascada
- **Configuración centralizada** para cambiar parámetros sin reiniciar
- **Diseño sin estado (stateless)** para escalar horizontalmente

- **Aislamiento de bases de datos** por servicio

Seguridad

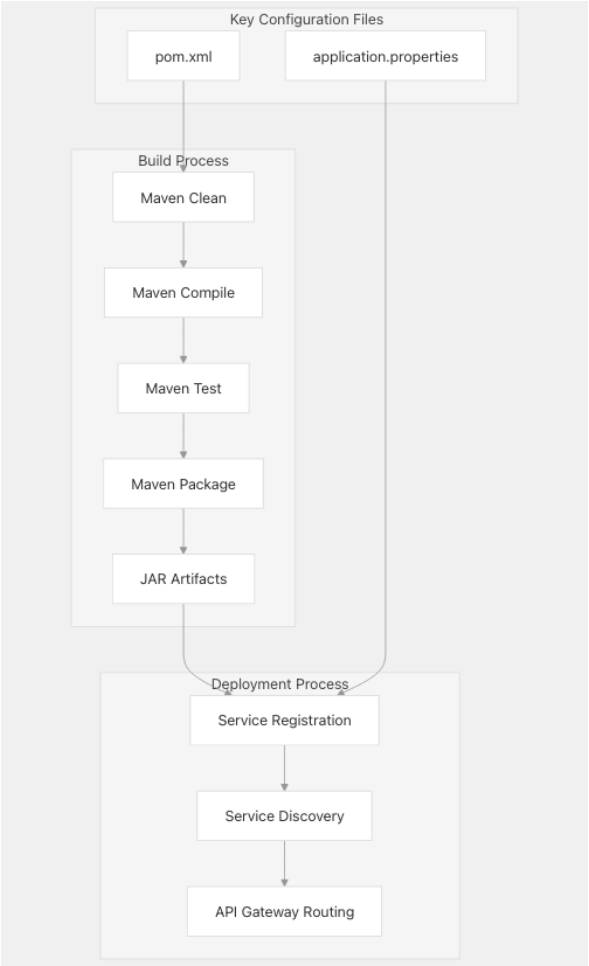
La seguridad se implementa en múltiples capas:

- **Autenticación**: JWT emitido por el Employee Service
- **Autorización**: Control por roles (Spring Security)
- **Seguridad en el API Gateway**: Validación de JWT
- **Seguridad en cada servicio**: Configuración local + compartida
- **Llamadas seguras entre servicios**: uso de JWT

Todas estas configuraciones están centralizadas en la Biblioteca Común.

Build y Despliegue

El sistema está construido usando Apache Maven como herramienta principal de automatización de compilación. Cada microservicio está configurado como un proyecto Maven independiente con su propio ciclo de vida, manteniendo coherencia en todo el sistema mediante configuración compartida, dependencias comunes y gestión de versiones.



Archivos de Configuración Clave

- pom.xml
- application.properties

Procesos Principales

- **Proceso de Build**
    - mvn clean: Limpieza de artefactos previos
    - mvn compile: Compilación del código fuente
    - mvn test: Ejecución de pruebas
    - mvn package: Generación de artefactos JAR

- **Proceso de Despliegue**
    - Registro de servicios
    - Descubrimiento de servicios
    - Enrutamiento vía API Gateway

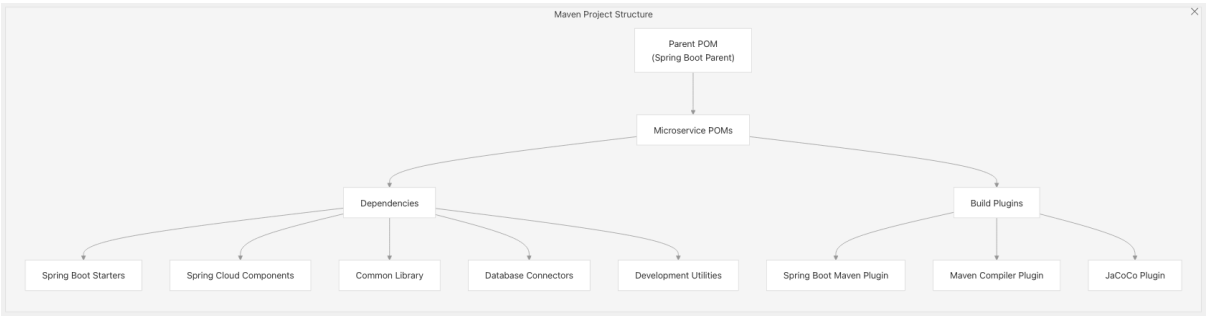
Arquitectura del Sistema de Compilación

Todos los microservicios siguen una estructura común de Maven, que maneja:

- Gestión de dependencias
- Compilación del código
- Pruebas automatizadas
- Empaquetado en JAR ejecutables

Estructura de Proyecto Maven

- **POM padre** (Spring Boot Parent)
- **POM de cada microservicio**
    - Dependencias
    - Plugins de construcción
    - Starters de Spring Boot
    - Componentes de Spring Cloud
    - Conectores a base de datos
    - Utilidades de desarrollo
    - Plugins como Maven Compiler y Spring Boot Plugin
    - Plugin de cobertura (JaCoCo



Requisitos del Entorno

Componente	Versión	Propósito
Java	17	Entorno de ejecución
Maven	3.x	Automatización de construcción
Spring Boot	3.4.5	Framework para aplicaciones
Spring Cloud	2024.0.1	Framework para microservicios
MySQL	Última	Base de datos
Docker	Última	Contenerización (opcional)

Dependencias

Dependencia	Propósito
Spring Boot Starter Data JPA	Acceso a base de datos con JPA
Spring Boot Starter Security	Seguridad y autenticación
Spring Boot Starter Web	Desarrollo de APIs REST
Spring Eureka	Registro y descubrimiento de servicios
MySQL Connector	Conectividad a MySQL
Lombok	Reducción de código repetitivo
MapStruct	Mapeo de objetos

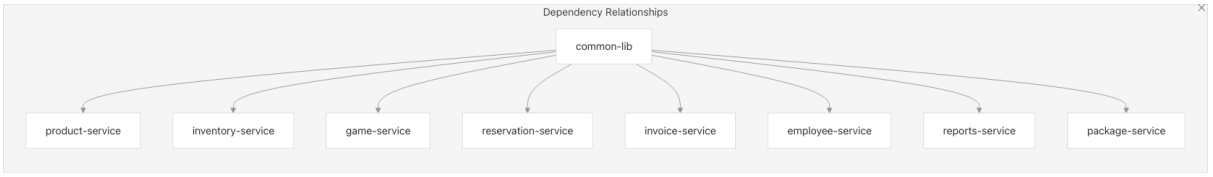
Springdoc OpenAPI	Documentación de APIs
JaCoCo	Cobertura de código

Biblioteca Comun

Una biblioteca común (common-lib) utilizada por todos los servicios para mantener:

- Modelos de datos comunes
- Utilidades compartidas
- Configuraciones estándar

Servicios que la usan: producto, inventario, juegos, reservaciones, facturas, empleados, reportes, paquetes.



Perfiles y Configuración de Maven

Configuración del Compilador

Todos los servicios están configurados para usar **Java 17**, incluyendo soporte para anotaciones de Lombok y MapStruct.

Cobertura de Código con JaCoCo

El plugin JaCoCo está configurado para excluir clases generadas automáticamente como los *MapperImpl.

Flujo de Construcción y Despliegue

Orden de Ejecución:

1. Iniciar el servidor Eureka
2. Iniciar el servicio de configuración

3. Registrar microservicios
4. Configuración centralizada cargada
5. Iniciar API Gateway
6. El sistema está listo para recibir solicitudes

Arquitectura de Despliegue

La arquitectura de despliegue sigue este orden:

1. **Eureka Server**
2. **Config Service**
3. **API Gateway**
4. **Servicios Core** (empleado, producto, inventario, juegos)
5. **Servicios de Negocio** (reservas, facturación, paquetes, reportes)

Pruebas

El proyecto incorpora soporte para pruebas automáticas con:

- Spring Boot Test: pruebas de integración
- Spring Security Test: pruebas de seguridad
- Mockito: pruebas unitarias con objetos simulados
- JaCoCo: informes de cobertura de código

Empaquetado y Despliegue

Todos los servicios se empaquetan como archivos .jar ejecutables usando el plugin spring-boot-maven-plugin.

Excluye dependencias como lombok del empaquetado final.

Registro y Descubrimiento de Servicios

Todos los microservicios se registran en **Eureka** al iniciar, y el **API Gateway** los descubre automáticamente para enrutar las solicitudes de los clientes.

Servicios Centrales

Servicio de Configuración

El **Servicio de Configuración** actúa como un componente central de infraestructura del cual dependen todos los demás servicios para obtener su configuración. Está construido con **Spring Boot** e integrado con el **servidor Eureka** para descubrimiento de servicios.

Capas de Arquitectura

- **Capa de Servicio**
- **Capa de Infraestructura**

Cada servicio:

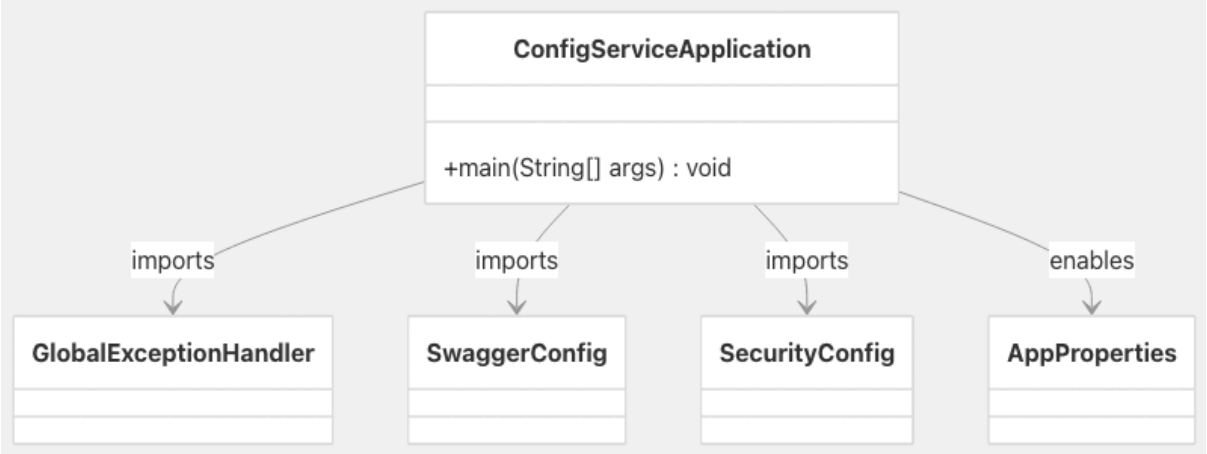
- **se registra** en Eureka
- **carga configuración** desde el Servicio de Configuración
- **descubre** otros servicios a través de Eureka

Servicios involucrados:

- Configuración
- Eureka
- Empleados
- Productos
- Inventario
- Juegos
- Reservas

- Facturación
- Paquetes

Implementación Técnica



El **Servicio de Configuración** es una aplicación Spring Boot con las siguientes características clave:

- **Clase principal**: ConfigServiceApplication
    - Método: main(String[] args)

Importaciones comunes

- GlobalExceptionHandler: Manejo global de excepciones
- SwaggerConfig: Documentación de API
- SecurityConfig: Configuración de seguridad
- AppProperties: Propiedades de configuración compartidas

Componentes Principales

La arquitectura del Servicio de Configuración incluye:

- **Aplicación Spring Boot**: Servicio independiente listo para producción

- **Capa de persistencia JPA**: Usa Spring Data JPA para almacenar y recuperar configuraciones
- **Capa API REST**: Expone endpoints RESTful para acceder a la configuración
- **Capa de seguridad**: Autenticación y autorización con JWT
- **Cliente Eureka**: Se registra para descubrimiento de servicios

Flujo de Configuración

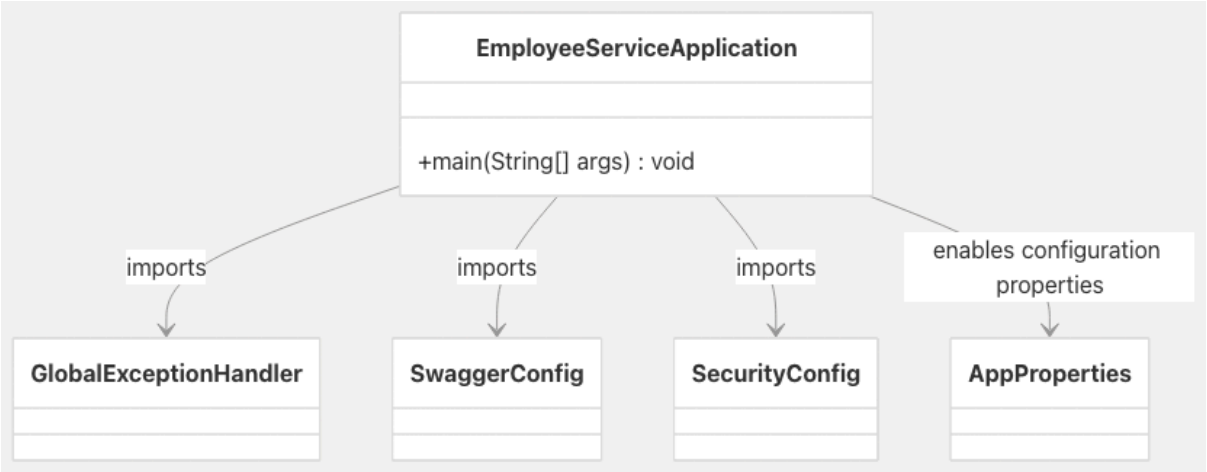
Este es el flujo de cómo se gestiona la configuración:

1. Durante el inicio, cada microservicio se **registra en Eureka**
2. El **Servicio de Configuración** recupera la información de Eureka
3. Se hace una **petición** de configuración al Servicio de Configuración
4. La **base de datos MySQL** proporciona los datos configurados
5. El microservicio **inicializa sus propiedades** usando @EnableConfigurationProperties
6. Si es necesario, se hace una **actualización dinámica** de la configuración

Patrón de Integración

Todos los servicios siguen un patrón consistente para integrarse con el Servicio de Configuración:

1. **Incluir dependencias necesarias** en su archivo pom.xml:
    - Dependencias de Spring Cloud
    - Cliente Eureka para descubrimiento de servicios
2. **Habilitar propiedades de configuración** en la clase principal



Ejemplo de integración

API Gateway

Posición en la Arquitectura

El **API Gateway** se ubica en el límite entre los clientes externos y la infraestructura interna de microservicios. Actúa como proxy inverso, ocultando la topología interna de los microservicios y ofreciendo una interfaz unificada a los clientes.

Responsabilidades Clave del API Gateway

1. **Enrutamiento de Solicitudes**: Dirige las peticiones de los clientes al microservicio correspondiente.

2. **Integración con Descubrimiento de Servicios**: Consulta al Servidor Eureka para localizar instancias de servicio.

3. **Balanceo de Carga**: Distribuye el tráfico entre múltiples instancias de un mismo servicio.

4. **Aplicación de Seguridad**: Gestiona la autenticación y la autorización.

5. **Transformación de Solicitudes/Respuestas**: Modifica peticiones o respuestas según sea necesario.

6. **Circuit Breaker**: Aísla fallos para evitar que se propaguen en cascada.

Flujo de una Solicitud

1. El cliente envía una **petición HTTP** al API Gateway.
2. El Gateway consulta el **Servidor Eureka** para obtener la ubicación de la instancia del microservicio destino.
3. Lee la **configuración de rutas** desde el Servicio de Configuración.
4. Reenvía la petición al **microservicio** apropiado.
5. Recibe la **respuesta del servicio** y la devuelve al cliente.

Integración con Servicios de Infraestructura

- **Servicio de Configuración**
    - El API Gateway recupera sus ajustes (rutas, balanceo, circuit breakers, políticas de seguridad) del Servicio de Configuración.
    - Puede actualizar dinámicamente su configuración al detectar cambios.
- **Servidor Eureka**
    - El Gateway usa Eureka para descubrir las instancias activas de cada microservicio.
    - Elimina la necesidad de URL codificadas, facilita balanceo y tolerancia a fallos

Seguridad en el API Gateway

- **Autenticación**: Verificación de identidad (normalmente con JWT).
- **Autorización**: Control de permisos según el rol del usuario.
- **Limitación de Tasa**: Previene abusos limitando la frecuencia de peticiones.
- **Filtrado de IP**: Bloquea direcciones no autorizadas.
- **Validación de Solicitudes**: Comprueba formato y contenido de las peticiones entrantes.

Flujo de Seguridad

1. Cliente envía petición.
2. Se valida la autenticación (JWT).
3. Se comprueban roles y permisos.
4. Se aplica limitación de tasa si procede.
5. La petición se reenvía al microservicio destino.

Patrón de Circuit Breaker

Para evitar fallos en cascada, el API Gateway:

- Monitorea las llamadas a servicios.
- **Abre el circuito** cuando se excede el umbral de errores (detiene peticiones).
- Redirige a métodos de reserva o devuelve respuestas por defecto.
- Permite solicitudes de prueba periódicas.
- **Cierra el circuito** cuando el servicio se recupera.

Responsabilidades Transversales

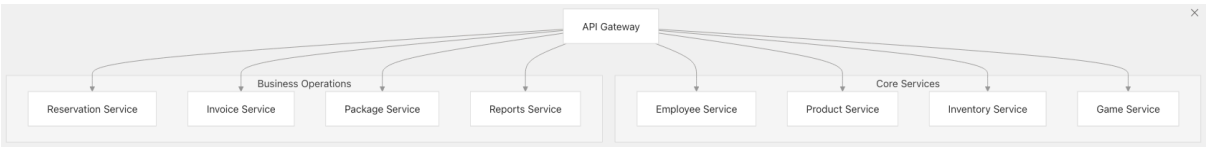
- **Registro y Monitoreo**: Consolidado de logs de todas las peticiones (ruta, tiempos, errores, identidad de usuario).
- **Transformación de Mensajes**:
    - Añade encabezados o contexto.
    - Elimina datos sensibles.
    - Convierte formatos o agrega información de varios servicios.
- **Versionado de API**:

- Versionado en la URI (/v1/...).
- Versionado por cabecera o parámetro.
- Negociación de contenido.

Integración con Servicios de Negocio

El API Gateway conecta y orquesta las interacciones entre los servicios principales del sistema:

- Servicio de Empleados
- Servicio de Productos
- Servicio de Inventario
- Servicio de Juegos
- Servicio de Reservas
- Servicio de Facturación
- Servicio de Paquetes
- Servicio de Reportes



Servicios de la Lógica del Negocio

Administración de Empleados

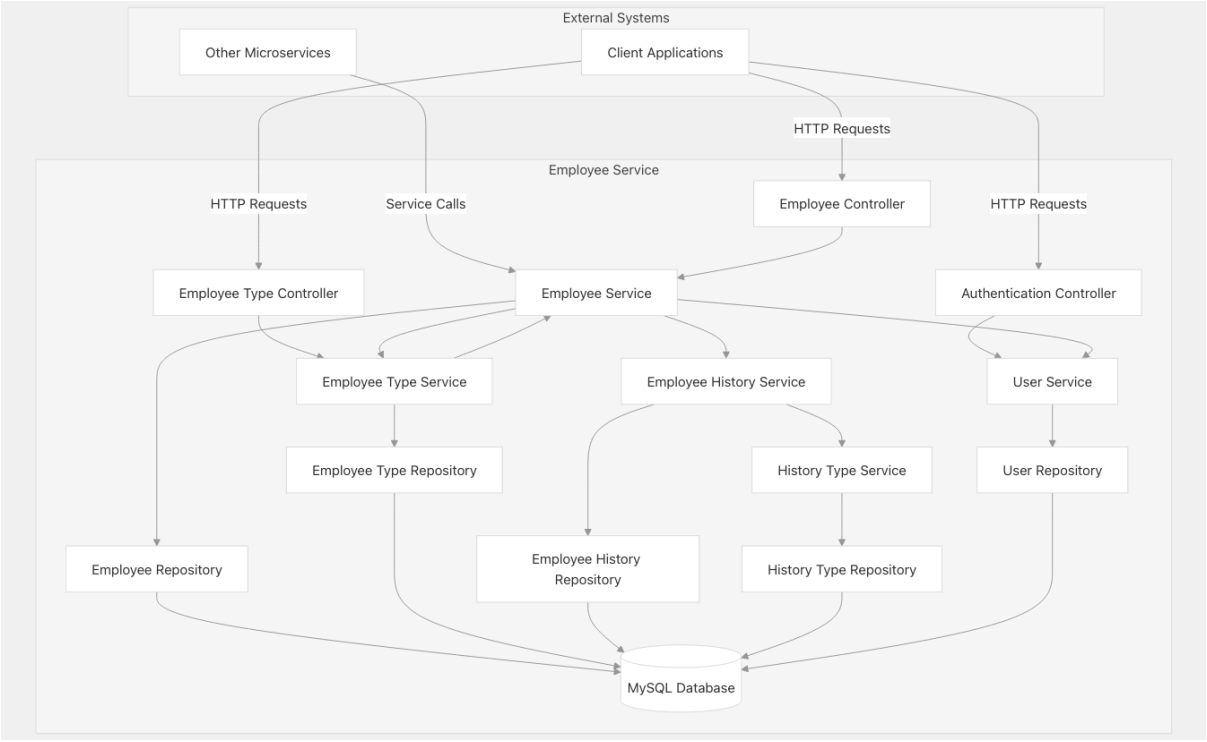
Employee Service

El **Servicio de Empleados** es un componente central del sistema, encargado de gestionar toda la información relacionada con los empleados: datos personales, tipos de empleados, permisos, autenticación e historial laboral. Este microservicio expone APIs para crear, actualizar, consultar y administrar empleados a lo largo del sistema.

Resumen del Servicio

El servicio centraliza la gestión de datos de empleados y tiene varias responsabilidades clave:

- Administración de datos personales y salariales.
- Gestión de tipos de empleados y sus permisos.
- Registro de eventos históricos (contratación, cambios salariales, activación/desactivación).
- Autenticación y autorización de usuarios.
- Provisión de información de empleados a otros servicios del sistema.

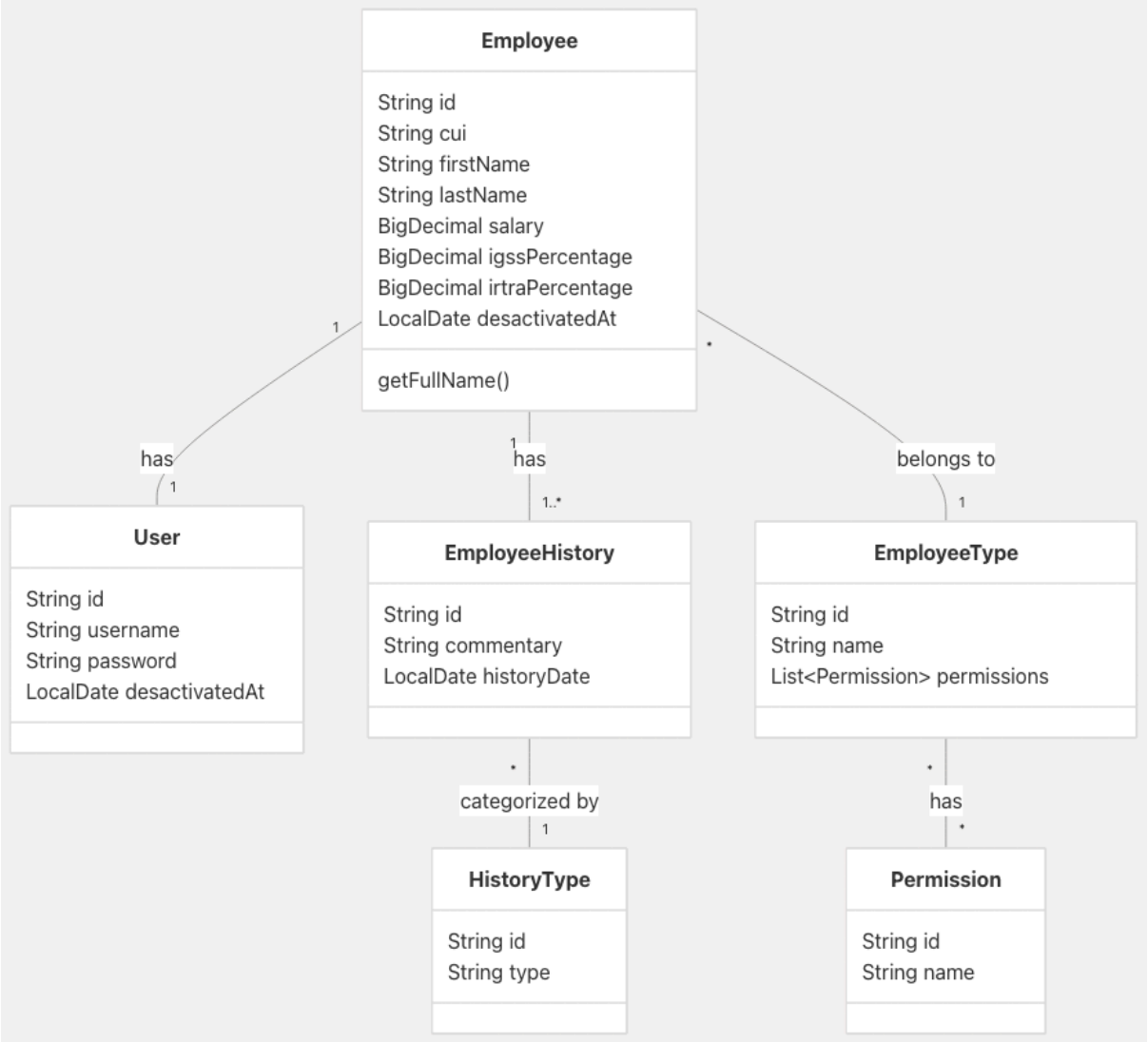


Modelo de Dominio Principal

El dominio del servicio gira alrededor de varias entidades interrelacionadas:

- **Empleado**: Información personal, salario, estado.
- **Usuario**: Cuenta de acceso al sistema (usuario, contraseña).

- **Historial de Empleado**: Registro de eventos laborales.
- **Tipo de Empleado**: Categoría del rol con permisos asociados.
- **Permiso**: Acción específica que puede realizarse.
- **Tipo de Historial**: Clasifica los eventos del historial.



Componentes Clave

Gestión de Empleados

Creación de Empleados

El proceso de creación involucra:

- Verificar que el tipo de empleado exista.

- Crear la cuenta de usuario.
- Crear un registro inicial de contratación.
- Establecer relaciones entre las entidades.
- Guardar el nuevo registro del empleado.

Actualización de Empleados

- **Información básica**: Se puede actualizar nombre, tipo, salario, etc.
- **Salario**: Se registra cada cambio en el historial salarial.
- **Estado laboral**: Activación y desactivación del empleado con historial.
- **Cambio de tipo**: Se puede reasignar el tipo de empleado (rol y permisos).

Gestión de Tipos de Empleado

Los tipos de empleado definen los roles del sistema y los permisos asociados. Operaciones principales:

- Crear tipos con permisos específicos.
- Actualizar nombre y lista de permisos.
- Eliminar tipos (con reasignación de empleados).
- Consultar tipos y permisos asignados.

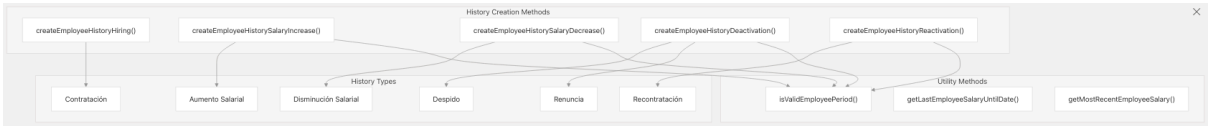
Seguimiento de Historial

El servicio mantiene un historial detallado de cada empleado. Tipos comunes de eventos:

- Contratación
- Despido
- Renuncia
- Recontratación

- Aumento de salario
- Reducción de salario

También incluye validaciones, como asegurarse de que los eventos se registren en períodos válidos de empleo activo.

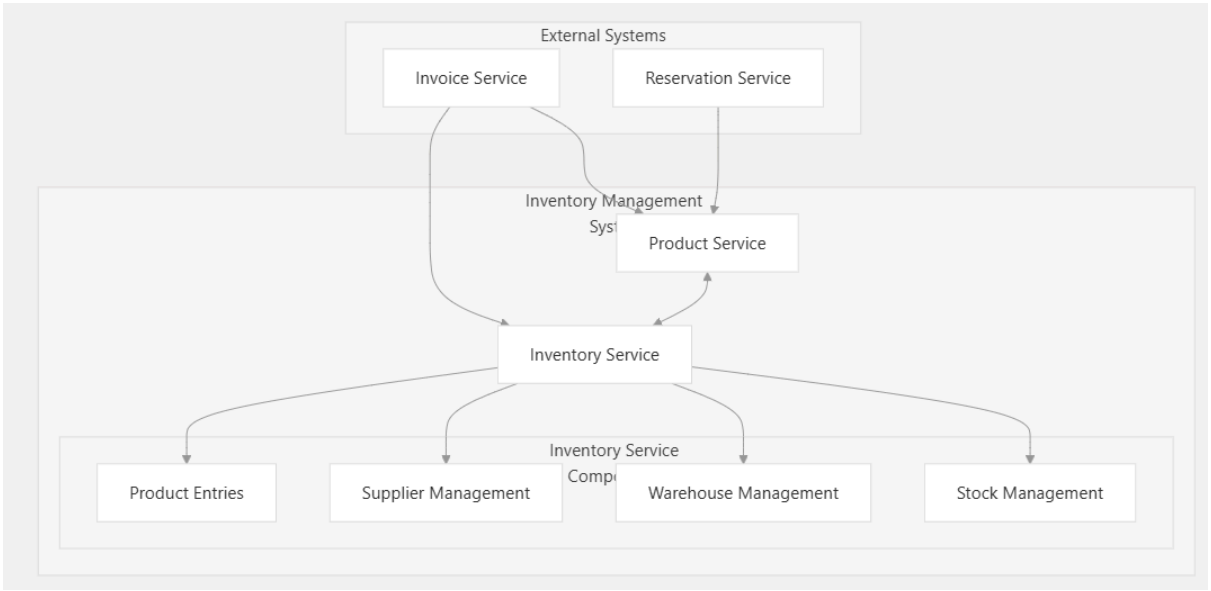


Autenticación y Gestión de Usuarios

- Cada empleado tiene una cuenta de usuario.
- Las credenciales se almacenan de forma segura.
- El estado del usuario refleja el estado del empleado (activo/inactivo).
- Se valida la autenticación por usuario y contraseña.
- Los permisos se basan en el tipo de empleado.

Administración de Inventario

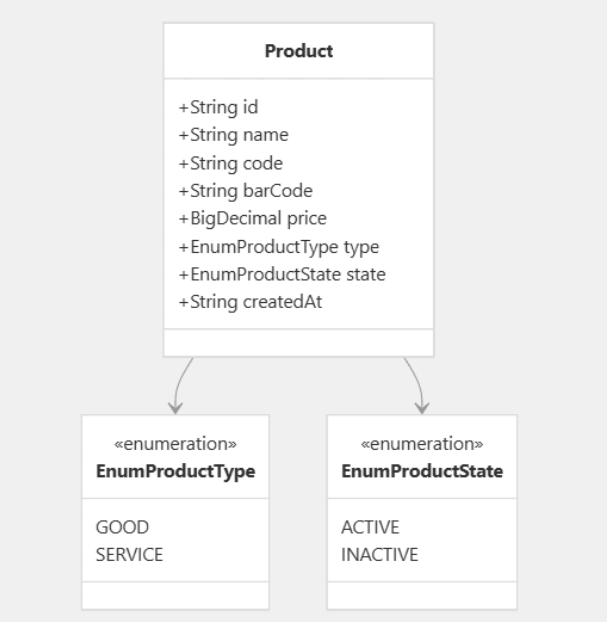
El sistema de gestión de inventario consta de varios componentes interconectados que trabajan en conjunto para administrar el inventario de productos. El sistema está construido como un conjunto de microservicios que manejan diferentes aspectos de la gestión de inventario.



Servicios Principales

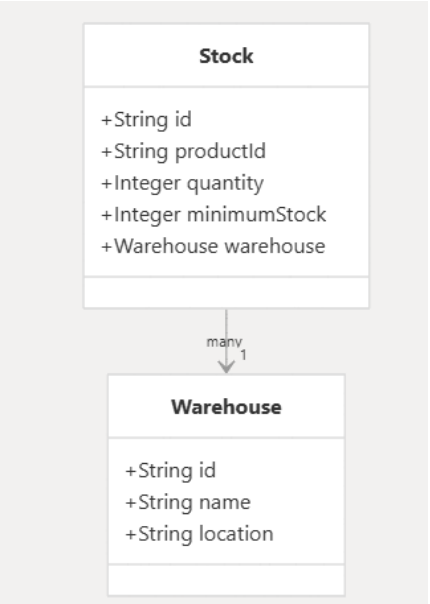
Gestión de Productos

El Servicio de Productos mantiene el catálogo de productos que constituye la base del sistema de inventario. Cada producto está identificado de forma única y contiene información esencial para el seguimiento del inventario.



Gestión de Stock

La Gestión de Stock es responsable de rastrear las cantidades de productos en diferentes almacenes. Incluye funcionalidades para monitorear los niveles de stock, manejar alertas por bajo inventario y mantener umbrales mínimos de existencias.



Operaciones de Stock

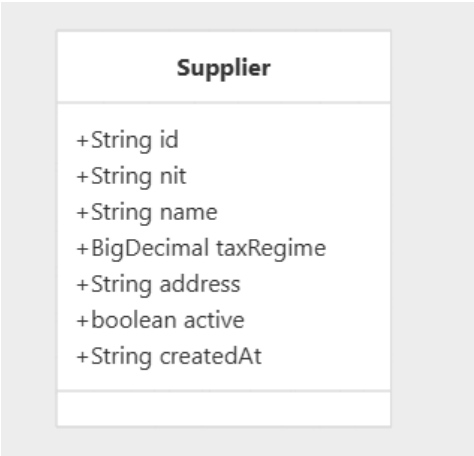
El sistema proporciona una funcionalidad integral para gestionar los niveles de stock:

- **Agregar Stock**: Incrementa la cantidad de un producto en un almacén específico.
- **Eliminar Stock**: Disminuye la cantidad, con validaciones para evitar stock negativo.
- **Monitoreo de Bajo Stock**: Identifica productos que están por debajo de su umbral mínimo.
- **Creación de Registros de Stock**: Cuando un producto se agrega por primera vez a un almacén.
- **Actualización de Niveles Mínimos de Stock**: Ajusta el umbral mínimo de cantidad para realizar reabastecimientos.

Las operaciones de stock aseguran que las cantidades de inventario se mantengan precisas y que las condiciones de bajo stock sean correctamente identificadas.

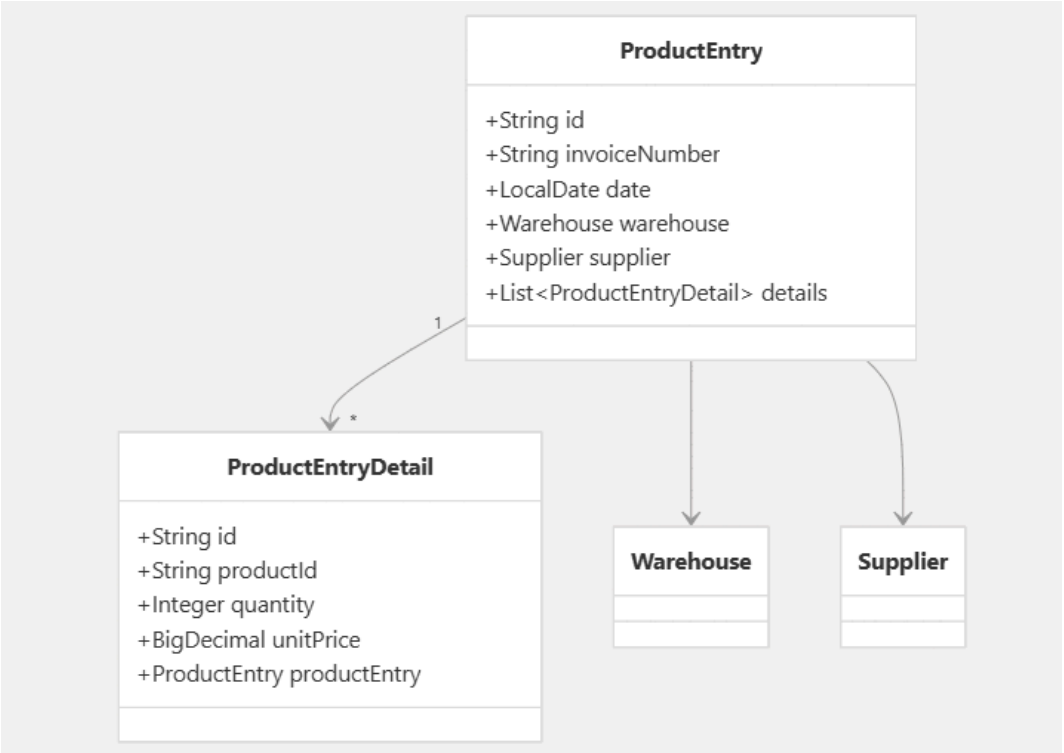
Gestión de Proveedores

El componente de Gestión de Proveedores se encarga de todas las operaciones relacionadas con los proveedores, incluyendo la incorporación de nuevos proveedores, la actualización de su información y la gestión de su estado.



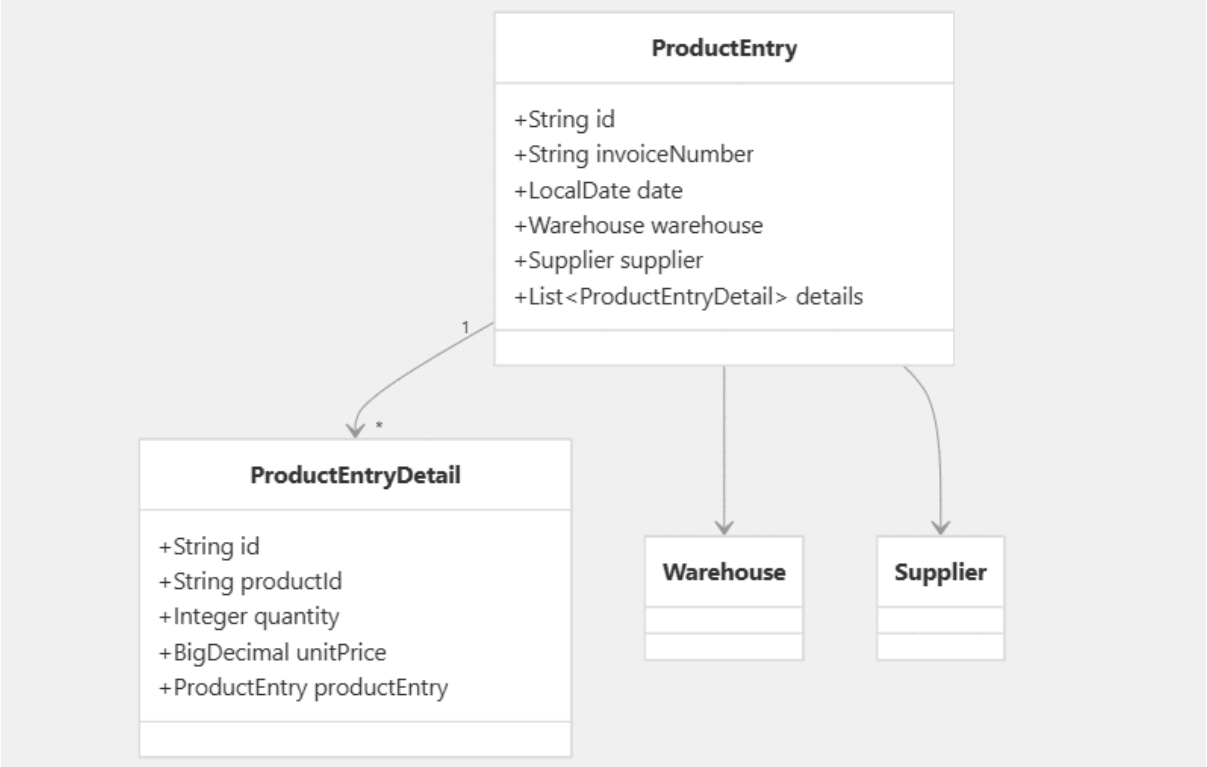
Gestión de Ingresos de Productos

Los Ingresos de Productos registran los productos que ingresan al inventario, incluyendo su origen, el almacén de destino y detalles sobre cantidades y precios.

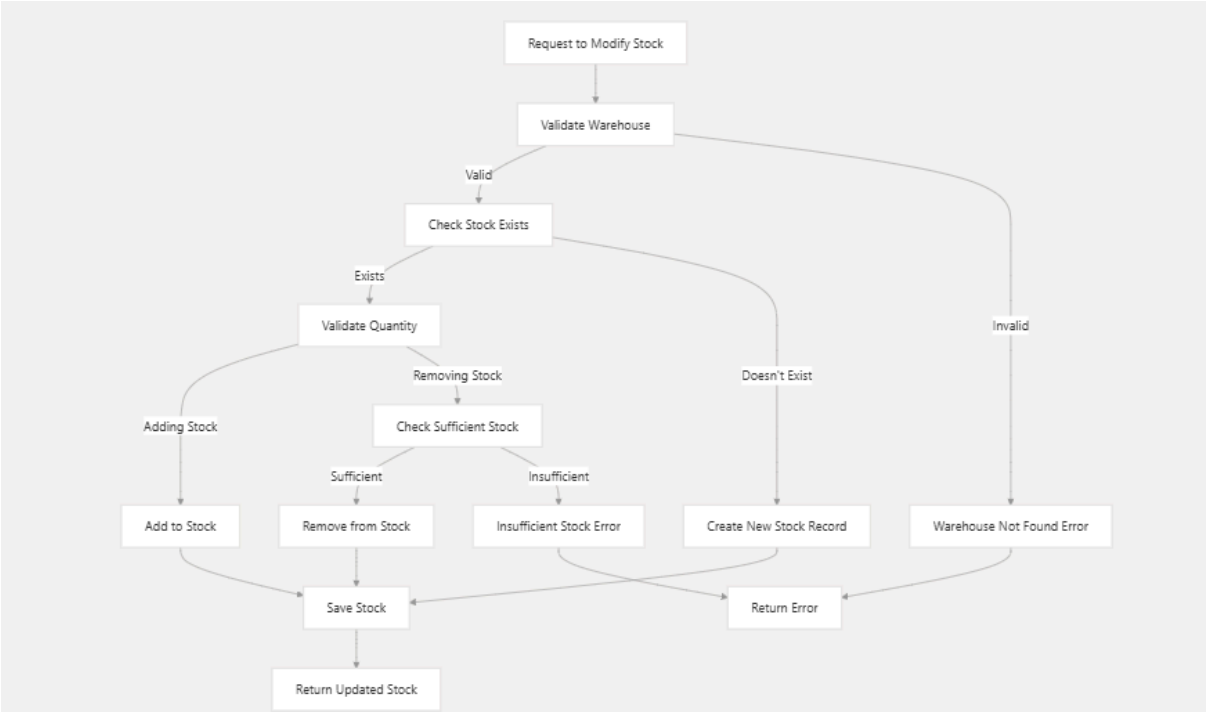


Gestión de Ingresos de Productos

Los ingresos de productos registran los productos que entran al inventario, incluyendo su origen, el almacén de destino y la información detallada sobre cantidades y precios.



Proceso de Modificación de Stock



Administración de Reservaciones

El Sistema de Reservas es un componente central de la aplicación que gestiona la creación, el pago y el seguimiento de las reservas tanto en línea como presenciales. Este sistema se coordina con el Servicio de Juegos para la gestión de jugadores, el Servicio de Facturación para el procesamiento de pagos, y el Servicio de Reportes para la generación de documentación y códigos QR.

Arquitectura del Sistema

El Sistema de Reservas se compone de varios componentes clave que trabajan en conjunto para gestionar el ciclo de vida de una reserva.

Componentes Principales

Componente	**Responsabilidad**
ReservationController	Expone endpoints de la API REST para operaciones de reserva
ReservationService	Implementa la lógica de negocio para la gestión de reservas

Componente	Responsabilidad
ReservationRepository	Proporciona operaciones de acceso a datos para la persistencia de reservas
ReservationSpecification	Permite filtrar reservas según distintos criterios
ForReservationPort	Define la interfaz entre el controlador y el servicio
ForGameClientPort	Permite la integración con el Servicio de Juegos
ForInvoiceClientPort	Permite la integración con el Servicio de Facturación
ForReportClientPort	Permite la integración con el Servicio de Reportes

Endpoints de la API

Método HTTP	Endpoint	Descripción	Permiso
POST	/api/v1/reservations/presential	Crear reserva presencial	CREATE_PRESENTIAL_RESERVATION
POST	/api/v1/reservations/online	Crear reserva en línea	—
PATCH	/cancel/{reservationId}	Cancelar reserva	CANCEL_RESERVATION
PATCH	/pay	Marcar como pagada	PAY_RESERVATION
GET	/{reservationId}	Obtener por ID	—
GET	/get-reservation-qr/{reservationId}	Obtener QR de reserva	—
GET	/all	Obtener todas con filtros opcionales	—
GET	/getReservationsBetweenDates	Consultar por rango de fechas	—

Método HTTP	Endpoint	Descripción	Permiso
POST	/get-popular-hours-between-dates	Obtener estadísticas de horarios populares	—
DELETE	/{reservationId}	Eliminar reserva	DELETE_RESERVATION

Integración con Otros Servicios

Servicio de Juegos

- Crea juegos vinculados a reservas
- Maneja jugadores (reservas en línea)
- Usa ForGameClientPort y DTOs como CreateGameRequestDTO

Servicio de Facturación

- Genera facturas al pagar
- Usa ForInvoiceClientPort y CreateInvoiceRequestDTO

Servicio de Reportes

- Genera PDFs con datos de la reserva
- Incluye QR para autenticación
- Usa ForReportClientPort

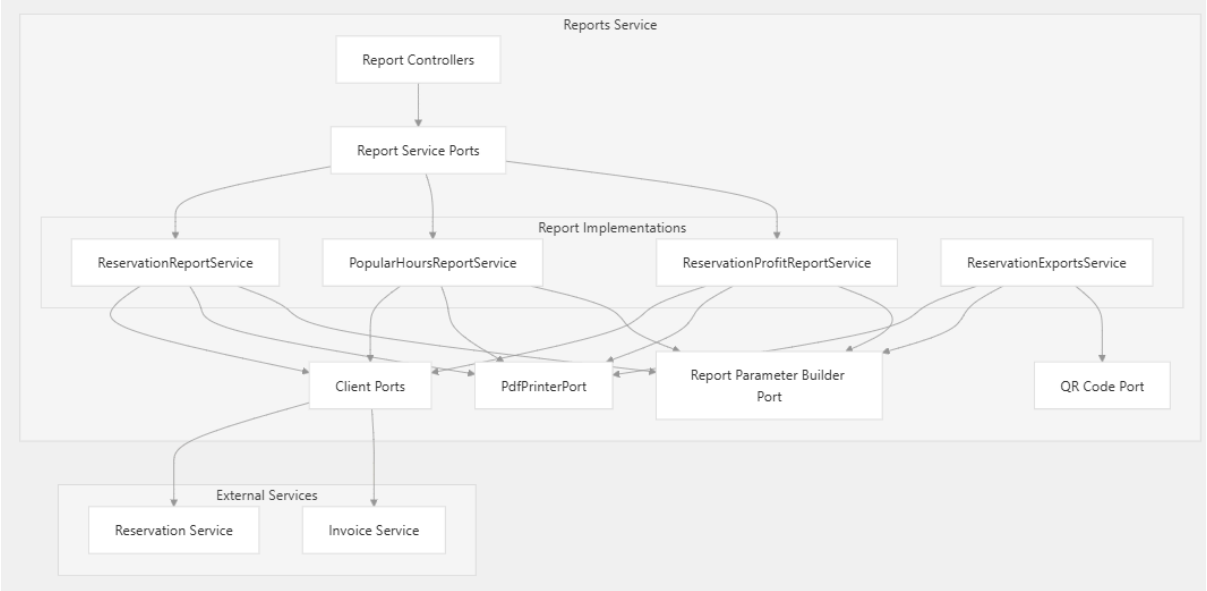
Capacidades de Reporte y Estadísticas

- Consultas como findReservationByDateBetween()
- Estadísticas de horarios más reservados
- Reportes generados en PDF y con código QR

Administración de Reportes

El Sistema de Reportes está diseñado como un microservicio que agrega datos desde otros servicios para generar reportes. Implementa una arquitectura puerto-adaptador (port-adapter) para mantener una separación clara de responsabilidades y facilitar las pruebas.

Diagrama de Componentes del Sistema

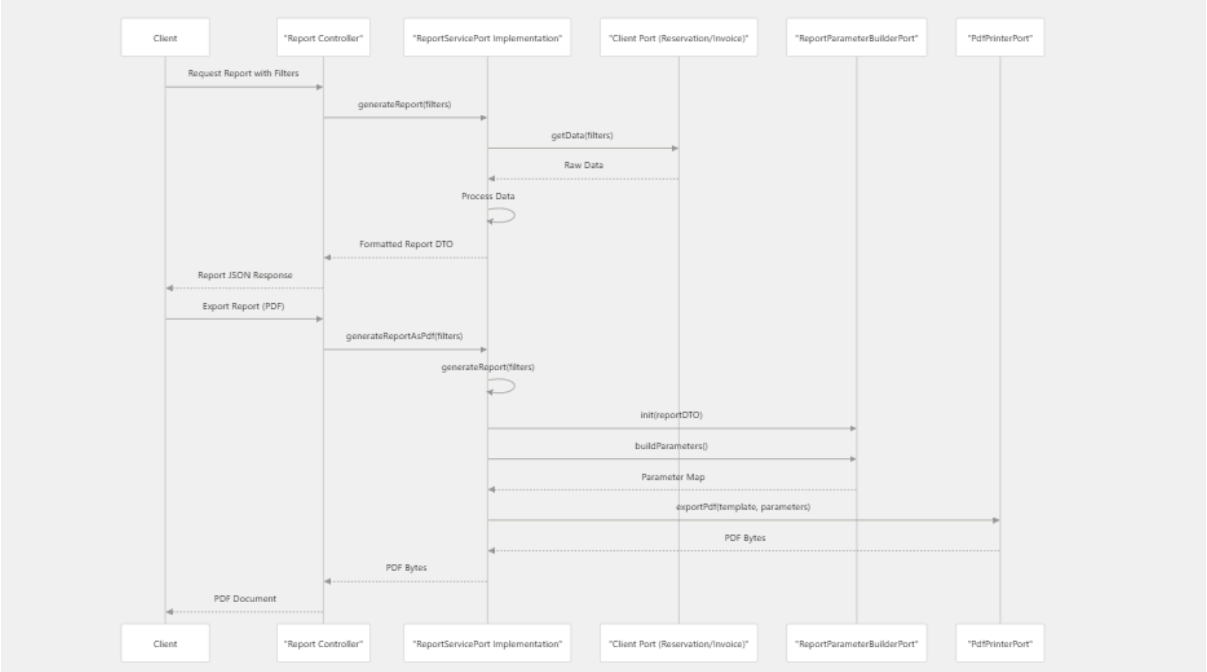


Tipos de Reporte

El sistema soporta múltiples tipos especializados de reportes:

Tipo de Reporte	**Propósito**	**Endpoint del Controlador**	**Implementación**
Reportes de Reservas	Genera listas de reservas	/api/v1/reservation-reports	ReservationReportService
Horarios Populares	Identifica franjas horarias más reservadas	/api/v1/popular-hours-report	PopularHoursReportService
Ganancia por Reservas	Calcula las ganancias obtenidas	(No especificado en los archivos)	ReservationProfitReportService
Exportación de Reservas	Exporta tickets y facturas con códigos QR	/api/v1/reservations-exports	ReservationExportsService

Flujo de Generación de Reportes



Biblioteca Común

La Librería Común es un componente compartido que proporciona objetos de transferencia de datos reutilizables y utilidades comunes para todo el ecosistema de microservicios. Esta librería facilita la comunicación estandarizada entre servicios, reduce la duplicación de código y garantiza estructuras de datos consistentes a lo largo de toda la aplicación.

Organización de paquetes