Universidad de San Carlos de Guatemala.
Centro Universitario de Occidente.
División de Ciencias de la Ingeniería.
Análisis y Diseño I.
Ing. Daniel Gonzales.

PROYECTO FINAL

Estudiante	Carnet
201930643	Jeffrey Kenneth Menéndez Castillo
201931811	Diego José Maldonado Monterroso
202131284	José Roberto Bautista Rojas

Quetzaltenango, Guatemala.

Miércoles 05 de Noviembre de 2025.

INFORMACIÓN GENERAL BACKEND

Nombre: Park Control S.A
Lenguaje Principal: Java 21
Framework Principal: Spring Boot 3.5.6

ARQUITECTURA DE SISTEMA

Clean Architecture.

Elegimos este tipo de arquitectura ya que permite desaclopar totalmente el sistema en capas, cada una con su propia responsabilidad, lo cual permite que el sistema sea facil de escalar, teniendo la habilidad de implementar nuevas funcionalidades sin afectar al resto del sistema. El sistema implementa la arquitectura dividida en las siguientes capas.

Dominio.

- ***Ubicación:*** domain.
- ***Responsabilidad:*** Contiene la lógica de negocio pura y reglas de dominio.
- ***Componentes:***
    - ***Models:*** Entidades de dominio sin dependencias externas.
    - ***Repositories:*** Interfaces que definen contratos de persistencia.
    - ***Services:*** Servicios de dominio con lógica de negocio compleja.
    - ***Exceptions:*** Excepciones específicas del dominio.

Aplicación.

- ***Ubicación:*** application.
- ***Responsabilidad:*** Orquesta los casos de uso del sistema.
- ***Componentes:***
    - ***Use Cases:*** Implementación de casos de uso específicos.
    - ***DTOs:*** Objetos de transferencia de datos.
    - ***Mappers:*** Conversión entre DTOs y objetos de dominio.
    - ***Ports:*** Interfaces para comunicación con capas externas.

Infraestructura.

- ***Ubicación:*** infrastructure.
- ***Responsabilidad:*** Implementa detalles técnicos y comunicación externa.
- ***Componentes:***
    - ***Persistence:*** Adaptadores JPA, entidades y repositorios.
    - ***Security:*** Implementación de seguridad JWT y autenticación.
    - ***Cache:*** Gestión de caché con Redis.
    - ***Storage:*** Integración con Azure Blob Storage.
    - ***Notification:*** Servicios de notificación por emai.l
    - ***Scheduler:*** Tareas programadas.

Presentación.

- ***Ubicación:*** presentation.
- ***Responsabilidad:*** Maneja la interfaz REST API.
- ***Componentes:***
    - ***Controllers:*** Controladores REST organizados por módulo.
    - ***Filters:*** Filtros de seguridad y autenticación.
    - ***Exception Handlers:*** Manejo global de excepciones.

ARQUITECTURA DE INFRAESTRUCTURA

Contenedores Docker.

- MariaDB 11: Base de datos principal.
- Redis 7: Cache y sesiones.
- Backend App: Aplicación Spring Boot.
- Frontend App: Aplicación Angular.

Servicios Cloud.

- Azure Blob Storage: Almacenamiento de archivos.
- Azure Container Apps: Despliegue en la nube.

TEST-DRIVEN DEVELOPMENT

El proyecto implementa TDD con cobertura mínima del 85% de líneas.

Tipos de Tests:

- Tests Unitarios de Dominio.
- Tests de Casos de Uso.
- Tests de Integración.

Configuración de Testing.

Base de Datos de Test.

- H2 In-Memory Database para tests unitarios.
- Configuración: application-test.properties.

Herramientas de Testing.

- JUnit 5: Framework principal de testing.
- Mockito: Mocking framework.
- AssertJ: Assertions fluidas.
- Spring Boot Test: Tests de integración.

Cobertura de Código.

Configuración JaCoCo:

- Cobertura mínima de líneas: 85%

Estrategias por Capa.

Capa de Dominio.

- Tests de comportamiento de entidades.
- Validación de reglas de negocio.
- Tests de value objects.

Capa de Aplicación.

- Tests de casos de uso con mocks.
- Validación de flujos de negocio.
- Tests de mappers.

Capa de Infraestructura.

- Tests de adaptadores.
- Tests de integración con bases de datos.
- Tests de servicios externos.

VERSIONES Y DEPENDENCIAS

Core.

Componente	Versión	Descripción
Java	21	Lenguaje de programación base
Spring Boot	3.5.6	Framework principal
Maven	3.9.6	Gestión de dependencias y build
MariaDB	11	Base de datos relacional
Redis	7	Cache en memoria

Dependencias Spring Boot.

Dependencia	Propósito
spring-boot-starter-data-jpa	Persistencia con JPA/Hibernate
spring-boot-starter-data-redis	Integración con Redis
spring-boot-starter-security	Seguridad y autenticación
spring-boot-starter-web	APIs REST
spring-boot-starter-mail	Envío de correos
spring-boot-starter-validation	Validación de datos

Dependencias de Seguridad.

Dependencia	Versión	Propósito
jjwt-api	0.12.5	API JWT
jjwt-impl	0.12.5	Implementación JWT
jjwt-jackson	0.12.5	Serialización JWT

Dependencias de Documentación.

Dependencia	Versión	Propósito
springdoc-openapi-starter-webmvc-ui	2.7.0	Documentación OpenAPI/Swagger
spring-restdocs-mockmvc	-	Documentación de APIs

Dependencias de Generación de Reportes.

Dependencia	Versión	Propósito
itext7-core	8.0.2	Generación de PDFs
poi-ooxml	5.2.5	Generación de Excel
jfreechart	1.5.4	Generación de gráficos

Dependencias Azure.

Dependencia	Versión	Propósito
spring-cloud-azure-starter	6.0.0	Integración Azure
azure-storage-blob	12.25.1	Azure Blob Storage

Dependencias de Testing.

Dependencia	Propósito
spring-boot-starter-test	Testing framework completo
mockito-core	Framework de mocking
mockito-junit-jupiter	Integración Mockito-JUnit5
reactor-test	Testing reactivo
h2	Base de datos en memoria para tests

Herramientas de Desarrollo.

Herramienta	Versión	Propósito
Lombok	-	Reducción de código boilerplate
JaCoCo	0.8.11	Cobertura de código
Maven Surefire	3.2.5	Ejecución de tests

Plugins Maven.

Plugin	Versión	Propósito
maven-compiler-plugin	-	Compilación Java 21
jacoco-maven-plugin	0.8.11	Análisis de cobertura
spring-boot-maven-plugin	-	Empaquetado Spring Boot
asciidoctor-maven-plugin	2.2.1	Documentación

DESPLIEGUE E INFRAESTRUCTURA

Containerización Docker.

Dockerfile Multi-stage:

Creación de Dockerfiles para cada app, en este caso, Backend y Frontend, aplicando multi-stage para optimizar tiempos de build.

Docker Compose:

- Orquestación de servicios: MariaDB, Redis, Backend.
- Configuración de redes y volúmenes.
- Variables de entorno centralizadas.

CI/CD Pipeline.

GitHub Actions.

- Testing automático con JaCoCo.
- Build & push a Docker Hub.
- Despliegue automático a Azure.

Configuración de Calidad.

- Cobertura mínima: 85% líneas.
- Tests obligatorios en pull requests.
- Análisis estático de código.

Configuración de Aplicación.

Profiles de Spring.

- Development: Configuración local.
- Test: Configuración para testing.
- Production: Configuración optimizada.

Variables de Entorno.

Gestión centralizada de configuración sensible:

- Credenciales de base de datos.
- Claves JWT y Azure.
- Configuración de correo.
- Parámetros de cache.

INFORMACIÓN GENERAL FRONTEND

Nombre del Proyecto: ayd-proyecto-final-frontend.
Tipo: Aplicación Web de Página Única (SPA).
Framework Principal: Angular 20.3.0.

DEPENDENCIAS

Dependencia	***Versión***	***Propósito***
@angular/core	^20.3.0	Framework principal de Angular
@angular/common	^20.3.0	Módulos comunes (pipes, directivas)
@angular/compiler	^20.3.0	Compilador de plantillas
@angular/platform-browser	^20.3.0	Soporte para navegador
@angular/platform-server	^20.3.0	Soporte para SSR
@angular/router	^20.3.0	Sistema de navegación
@angular/forms	^20.3.0	Manejo de formularios
@angular/ssr	^20.3.2	Configuración SSR
@angular/cli	^20.3.2	CLI de Angular
@angular/build	^20.3.2	Sistema de compilación
@angular/compiler-cli	^20.3.0	Compilador CLI
typescript	~5.9.2	Lenguaje principal
tailwindcss	^3.4.18	Framework CSS utility-first
postcss	^8.5.6	Procesador de CSS
autoprefixer	^10.4.21	Prefijos de navegador
jasmine-core	~5.9.0	Framework de testing
karma	~6.4.0	Test runner
karma-chrome-launcher	~3.2.0	Launcher de Chrome
karma-jasmine	~5.1.0	Adaptador Jasmine/Karma
karma-jasmine-html-reporter	~2.1.0	Reportes HTML
karma-coverage	~2.2.0	Cobertura de código
@types/express	^5.0.1	Tipos TypeScript para Express
@types/jasmine	~5.1.0	Tipos TypeScript para Jasmine
@types/node	^20.17.19	Tipos TypeScript para Node.js

ARQUITECTURA IMPLEMENTADA

- El proyecto utiliza la arquitectura moderna de Angular con componentes standalone.
- No se utilizan NgModules tradicionales.
- Cada componente declara sus propias dependencias.
- Todos los módulos principales se cargan de forma diferida.
- La aplicación implementa SSR para mejorar el SEO y la carga inicial.
- Configurado con Angular Universal.

REPLICACIÓN DE SISTEMA

El sistema corre en una maquina virtual hosteada en Azure, con IP 172.190.44.206. Para acceder a la aplicación solamente debemos acceder al siguiente enlace: http://172.190.44.206:4200/login

Si queremos iniciar el sistema localmente, debemos llenar el .env.template con la información necesaria. Y usar Docker Compose para levantar los servicios de mariadb y redis. Posteriormente, bastaria con iniciar el servidor de Spring.

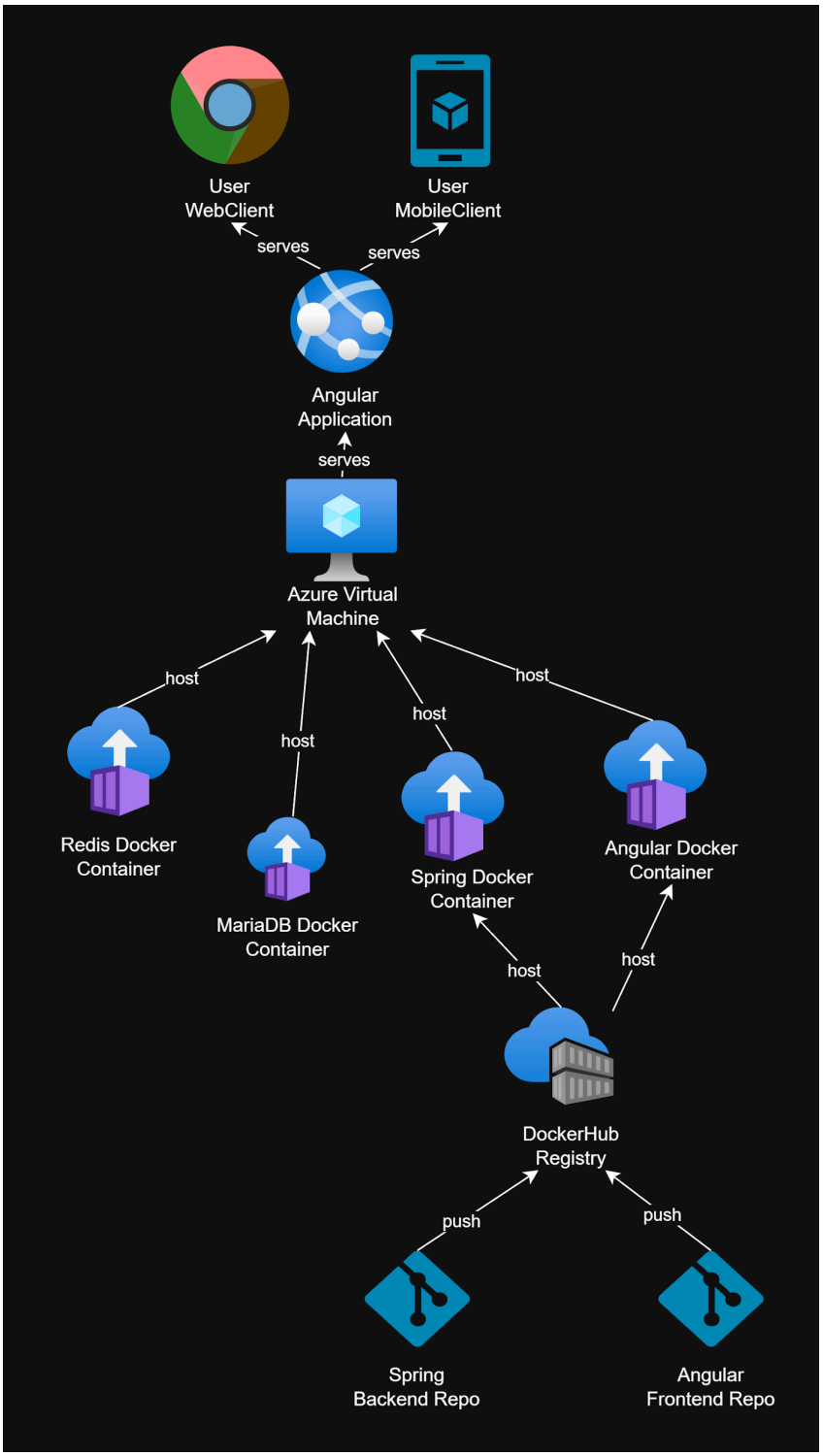
1. docker compose up -d mariadb redis
2. mvn clean compile
3. mvn spring-boot:run

Con estos comando podemos levantar nuestra aplicación localmente, debe tomar en cuenta que deberá ingresar los scripts de Base de Datos dentro del contendor de MariaDB. Usando:

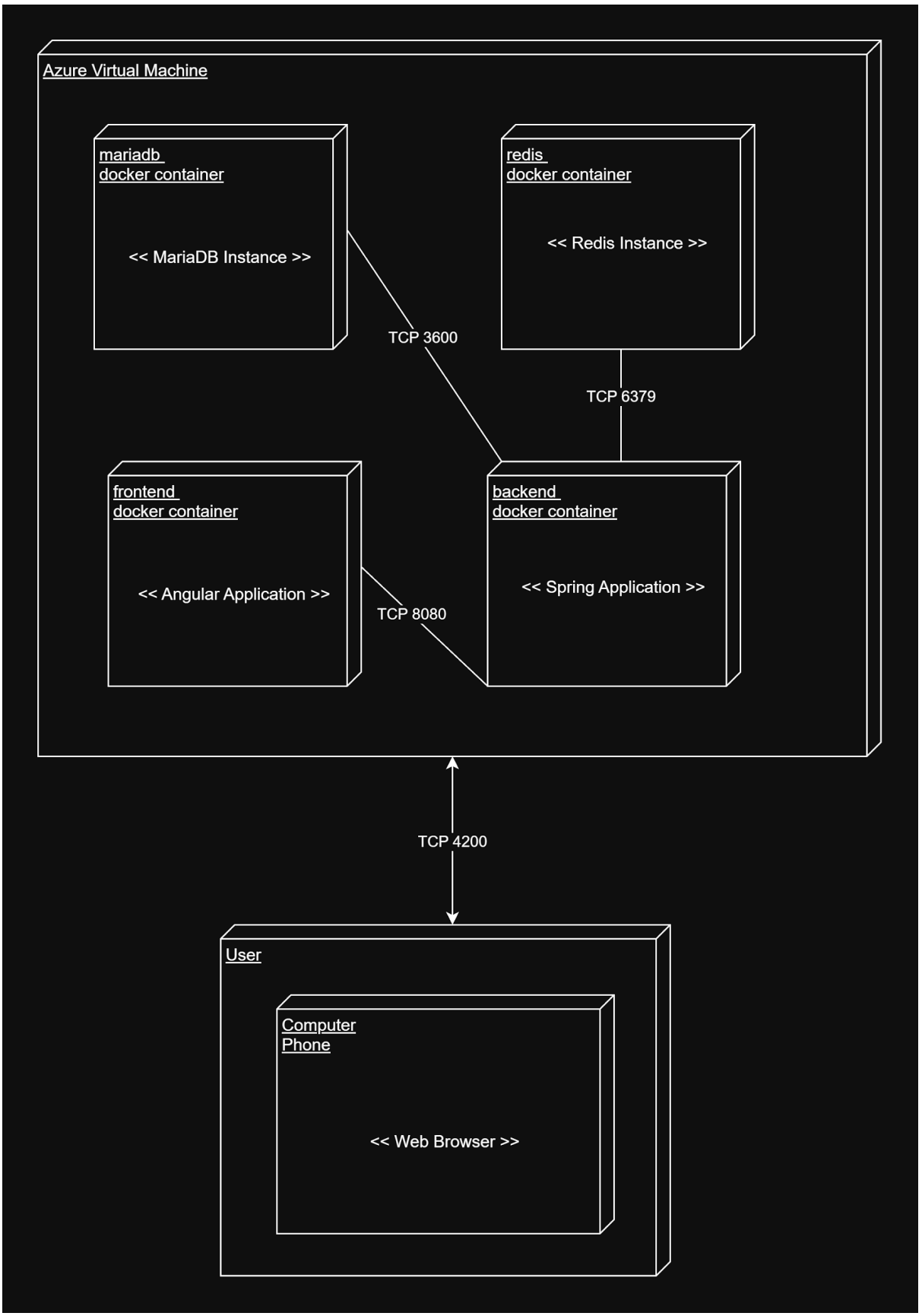
1. docker exec -it mariadb /bin/bash
2. mariadb -u root -p
3. copiar y pegar los scripts

DIAGRAMAS

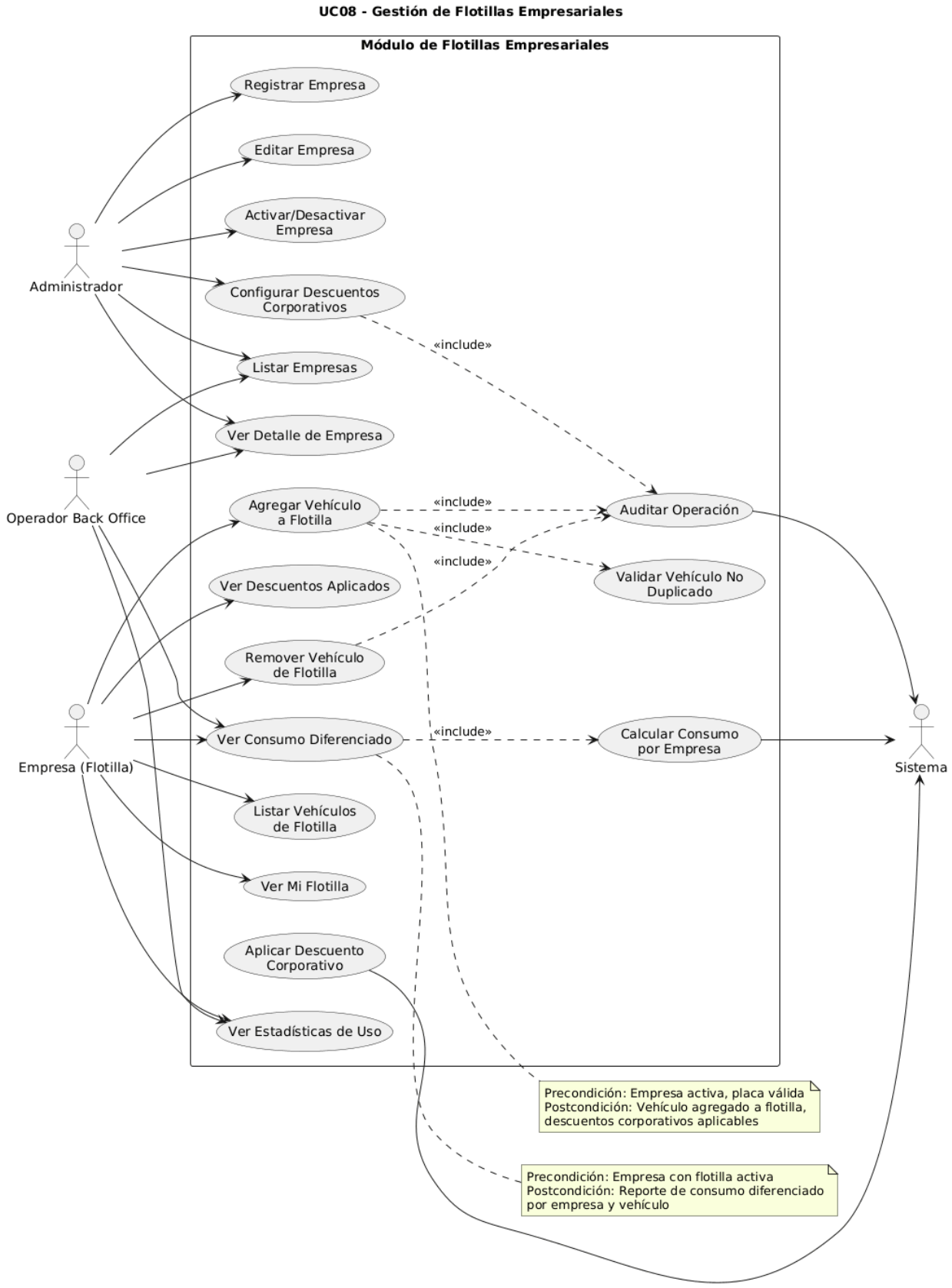
Arquitectura.



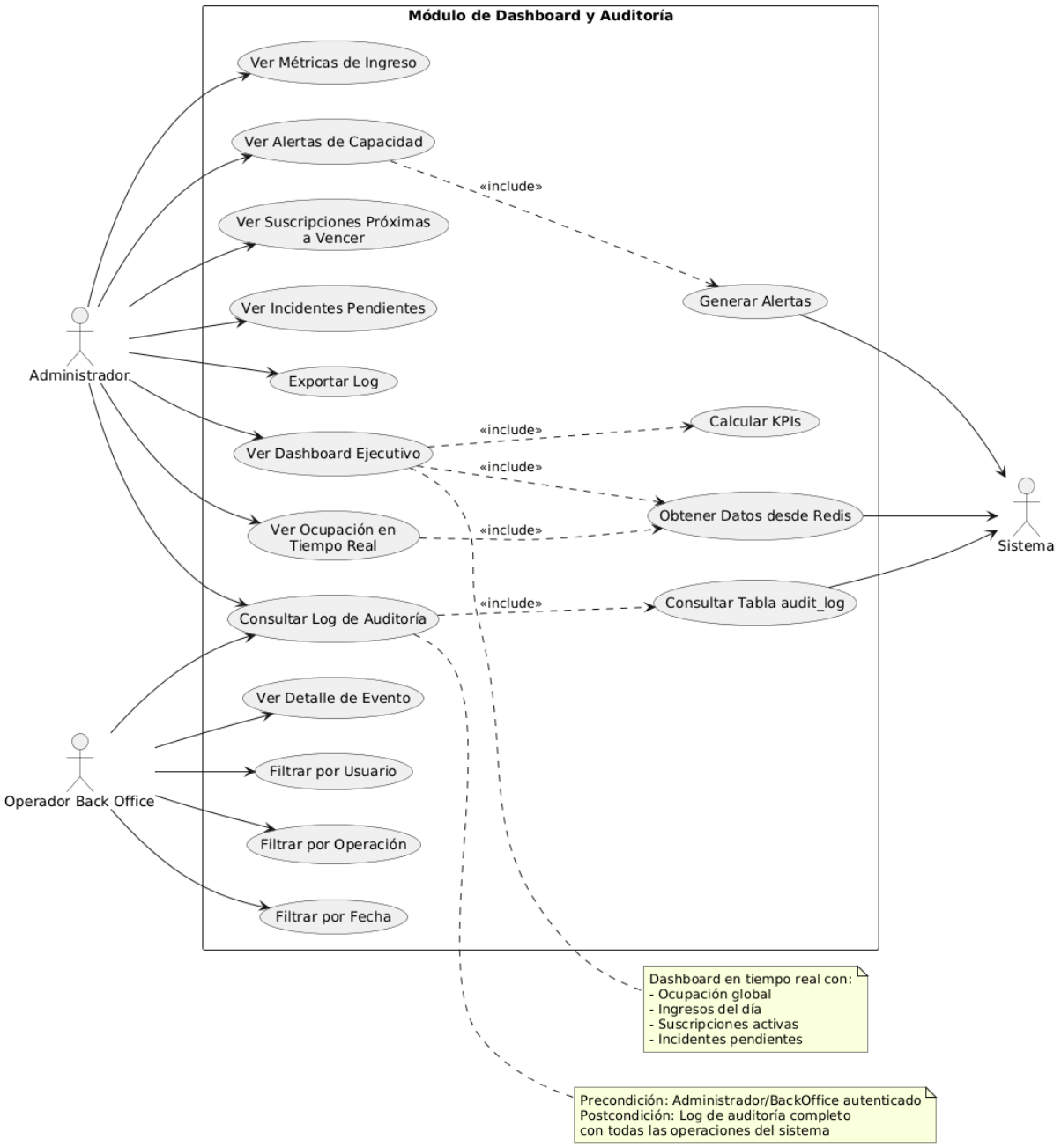
Despliegue.


Azure Virtual Machine
mariadb
docker container
<< MariaDB Instance >>
redis
docker container
<< Redis Instance >>
TCP 3600
TCP 6379
frontend
docker container
<< Angular Application >>
backend
docker container
<< Spring Application >>
TCP 8080
TCP 4200
User
Computer
Phone
<< Web Browser >>

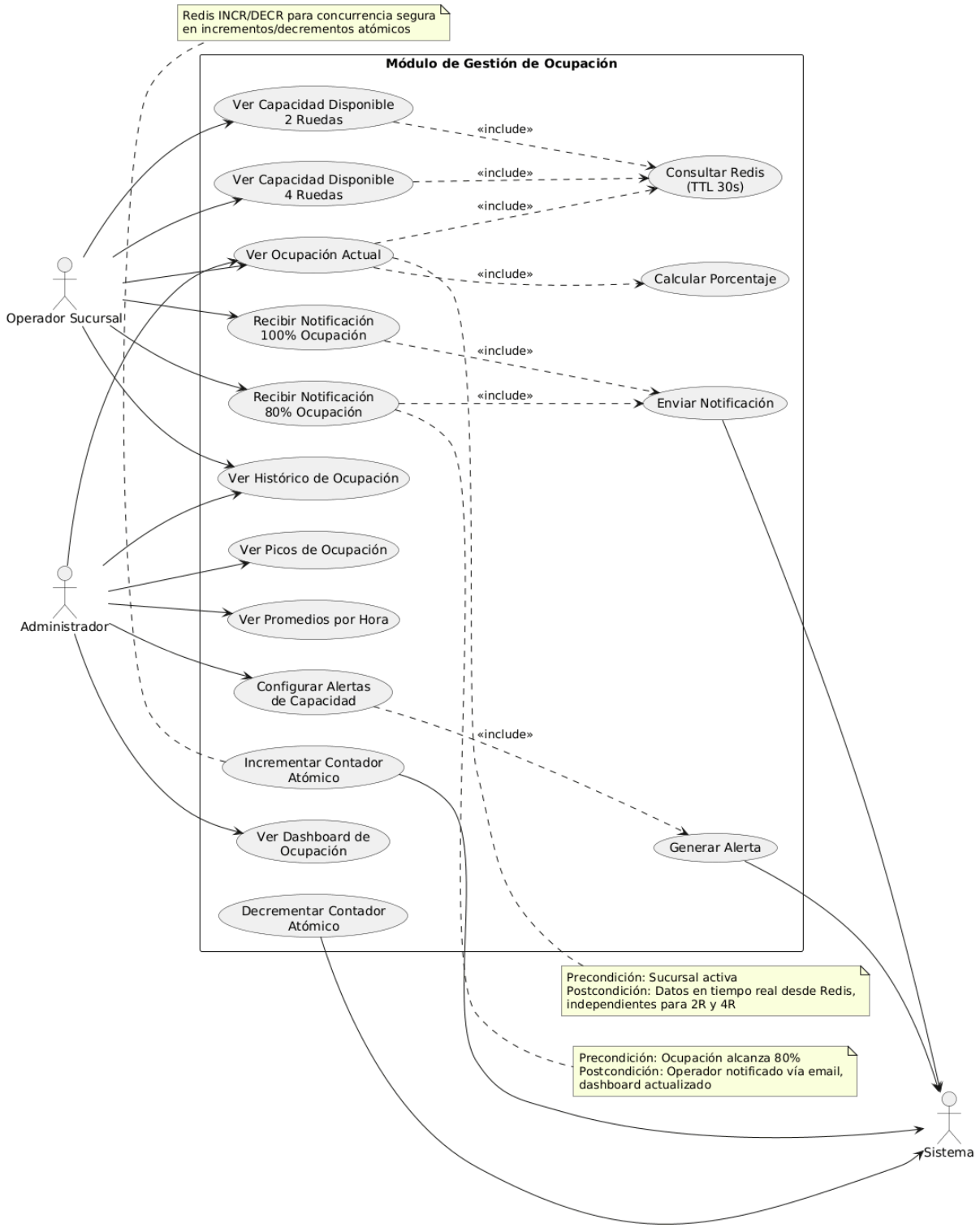
Casos de Uso.


UC08 - Gestión de Flotillas Empresariales
Módulo de Flotillas Empresariales
Registrar Empresa
Editar Empresa
Activar/Desactivar Empresa
Configurar Descuentos Corporativos
Listar Empresas
Ver Detalle de Empresa
Agregar Vehículo a Flotilla
Ver Descuentos Aplicados
Remover Vehículo de Flotilla
Ver Consumo Diferenciado
Listar Vehículos de Flotilla
Ver Mi Flotilla
Aplicar Descuento Corporativo
Ver Estadísticas de Uso
Auditar Operación
Validar Vehículo No Duplicado
Calcular Consumo por Empresa
«include»
«include»
«include»
«include»
«include»
Administrador
Operador Back Office
Empresa (Flotilla)
Sistema
Precondición: Empresa activa, placa válida
Postcondición: Vehículo agregado a flotilla, descuentos corporativos aplicables
Precondición: Empresa con flotilla activa
Postcondición: Reporte de consumo diferenciado por empresa y vehículo

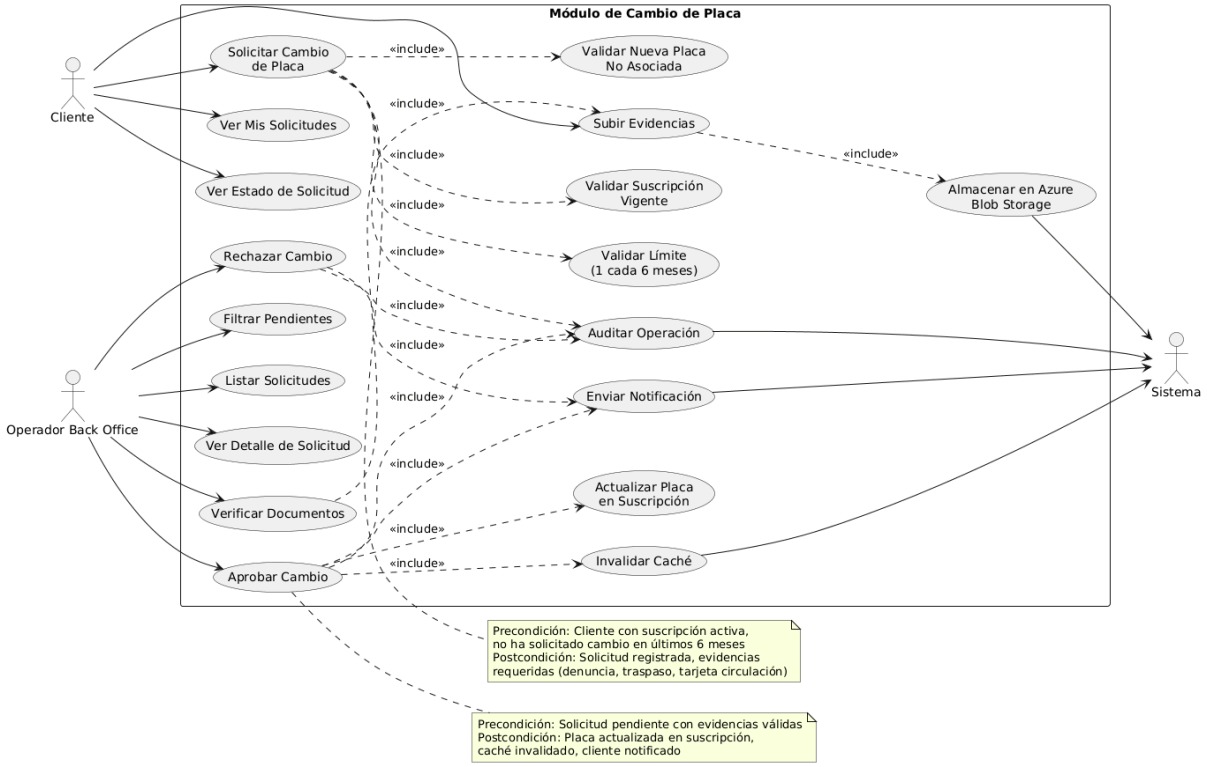
UC12 - Dashboard y Auditoría



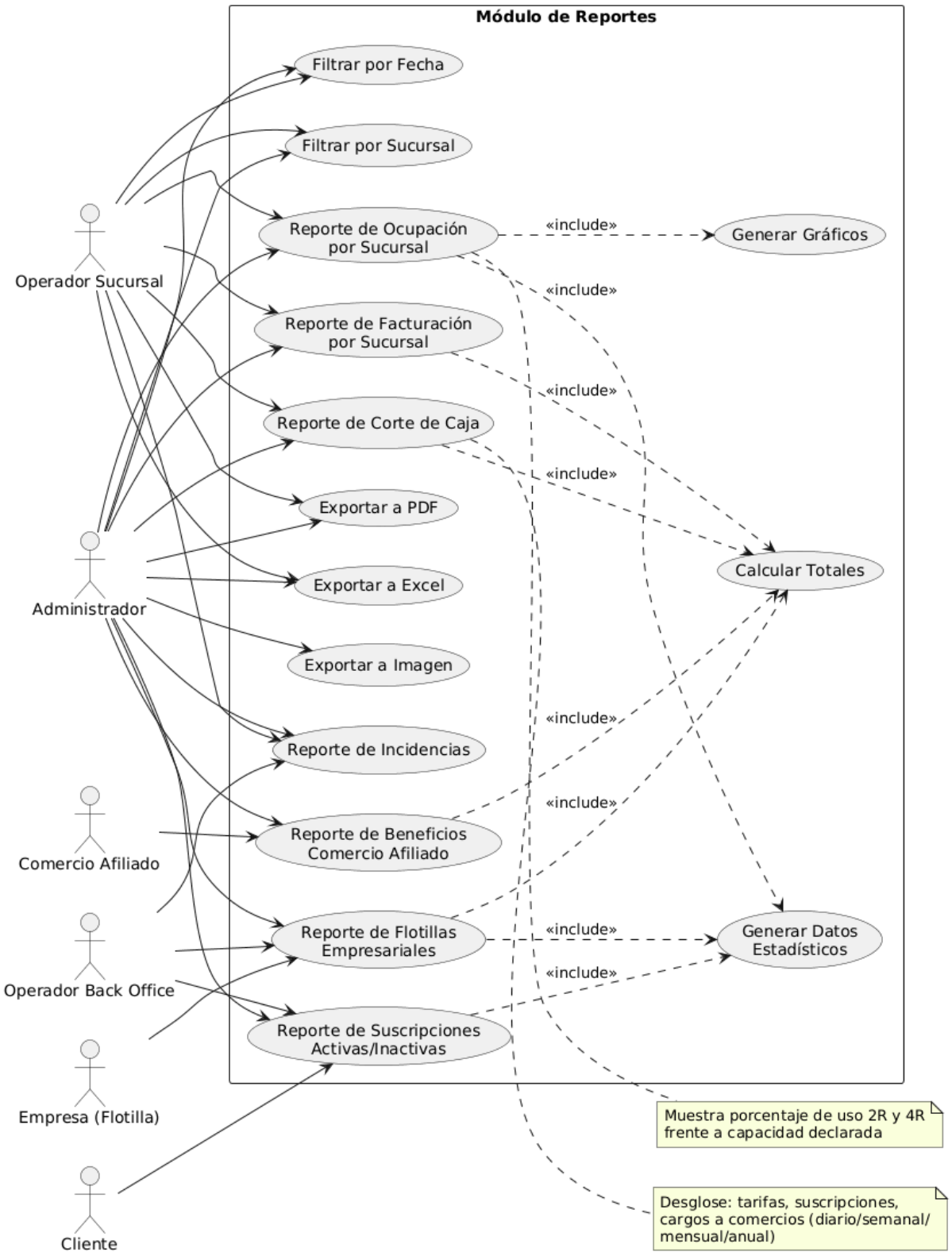
UC15 - Gestión de Ocupación



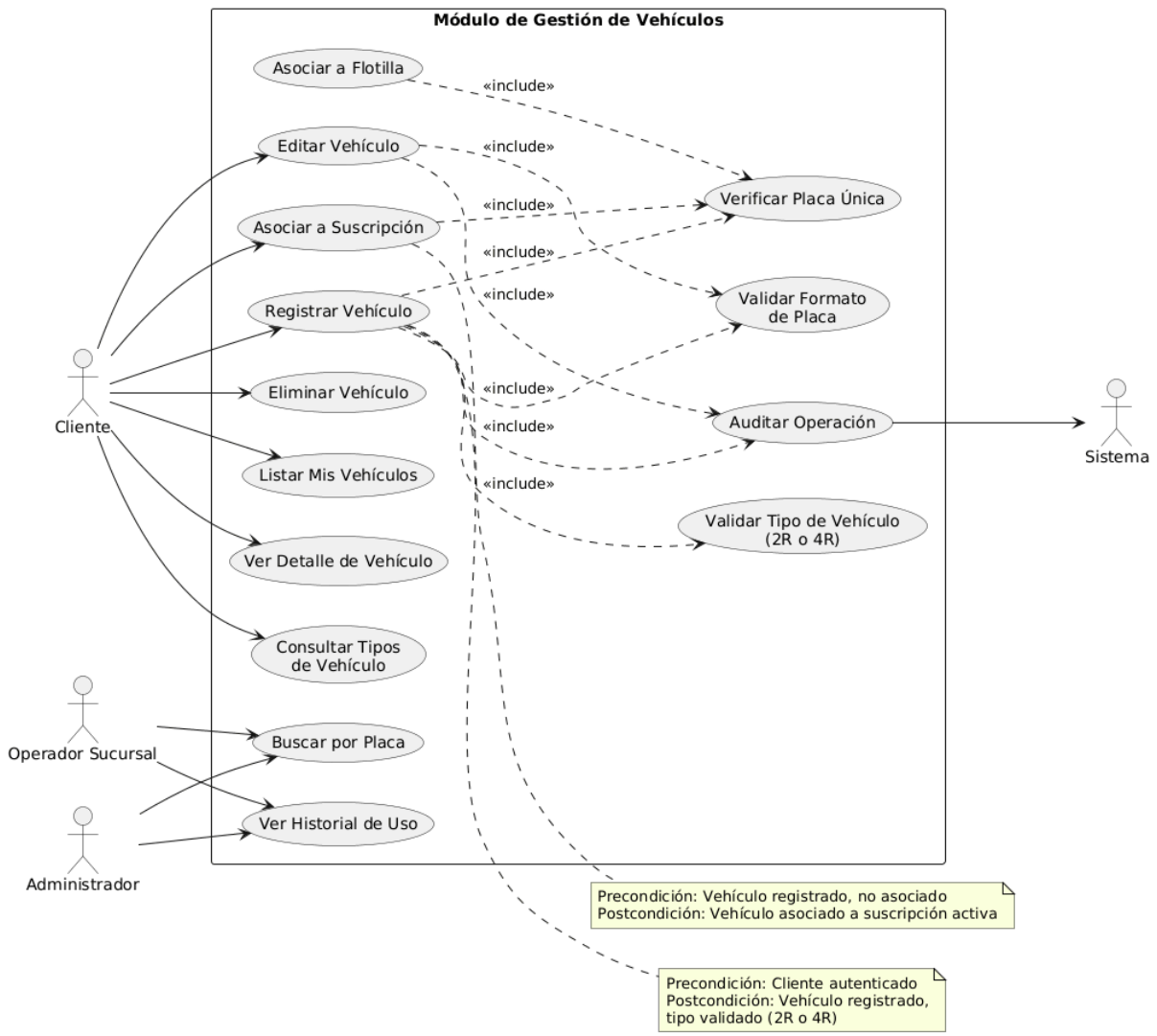
UC09 - Validaciones de Cambio de Placa (Back Office)



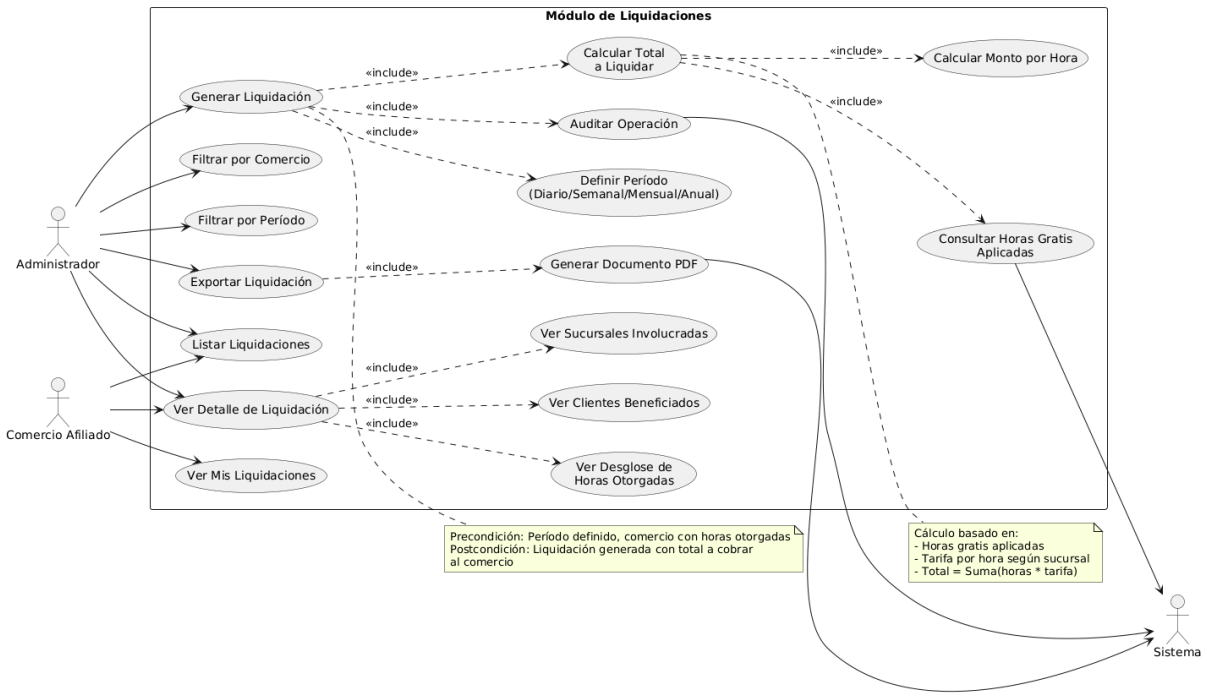
UC11 - Módulo de Reportes del Sistema



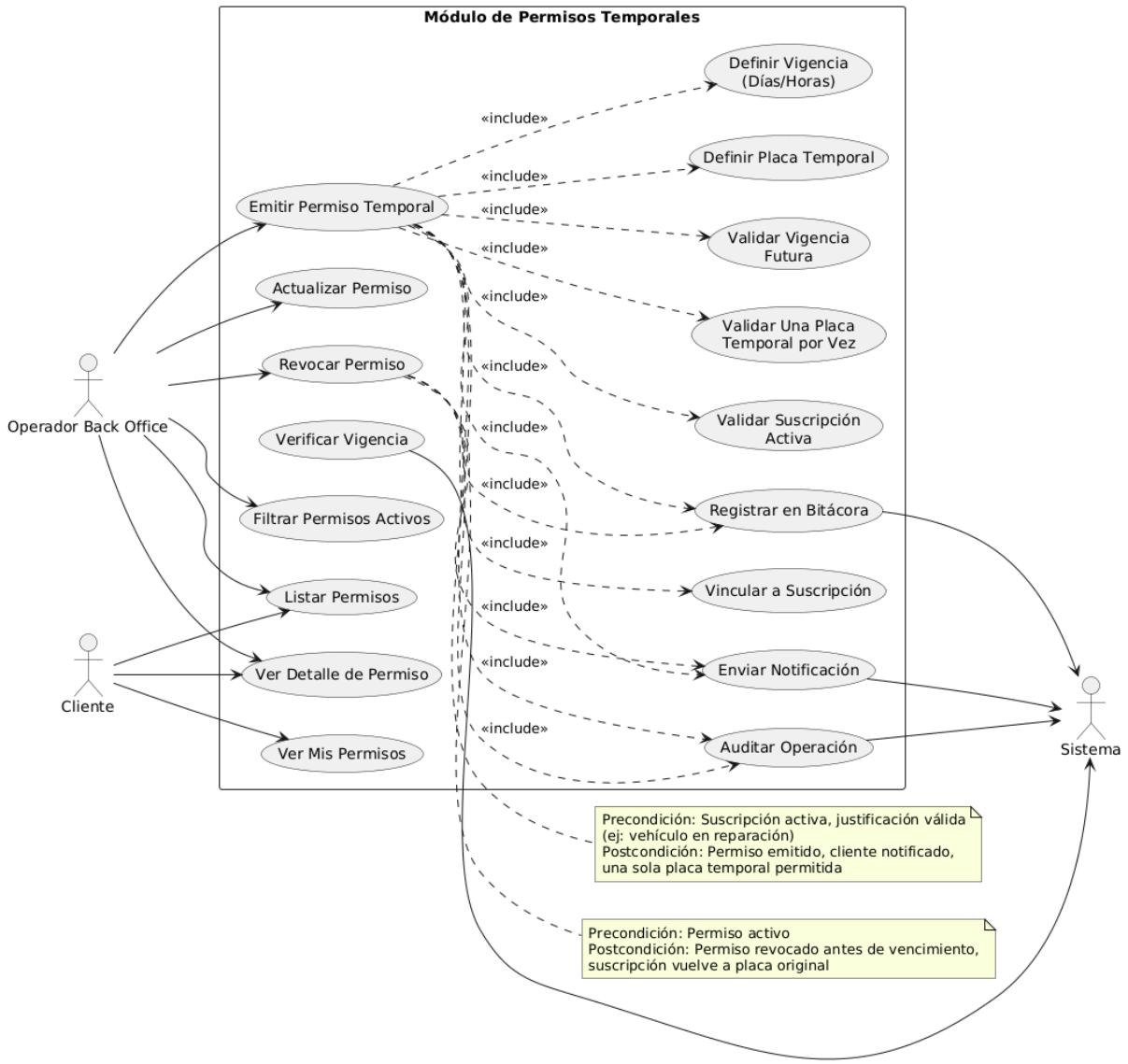
UC13 - Gestión de Vehículos



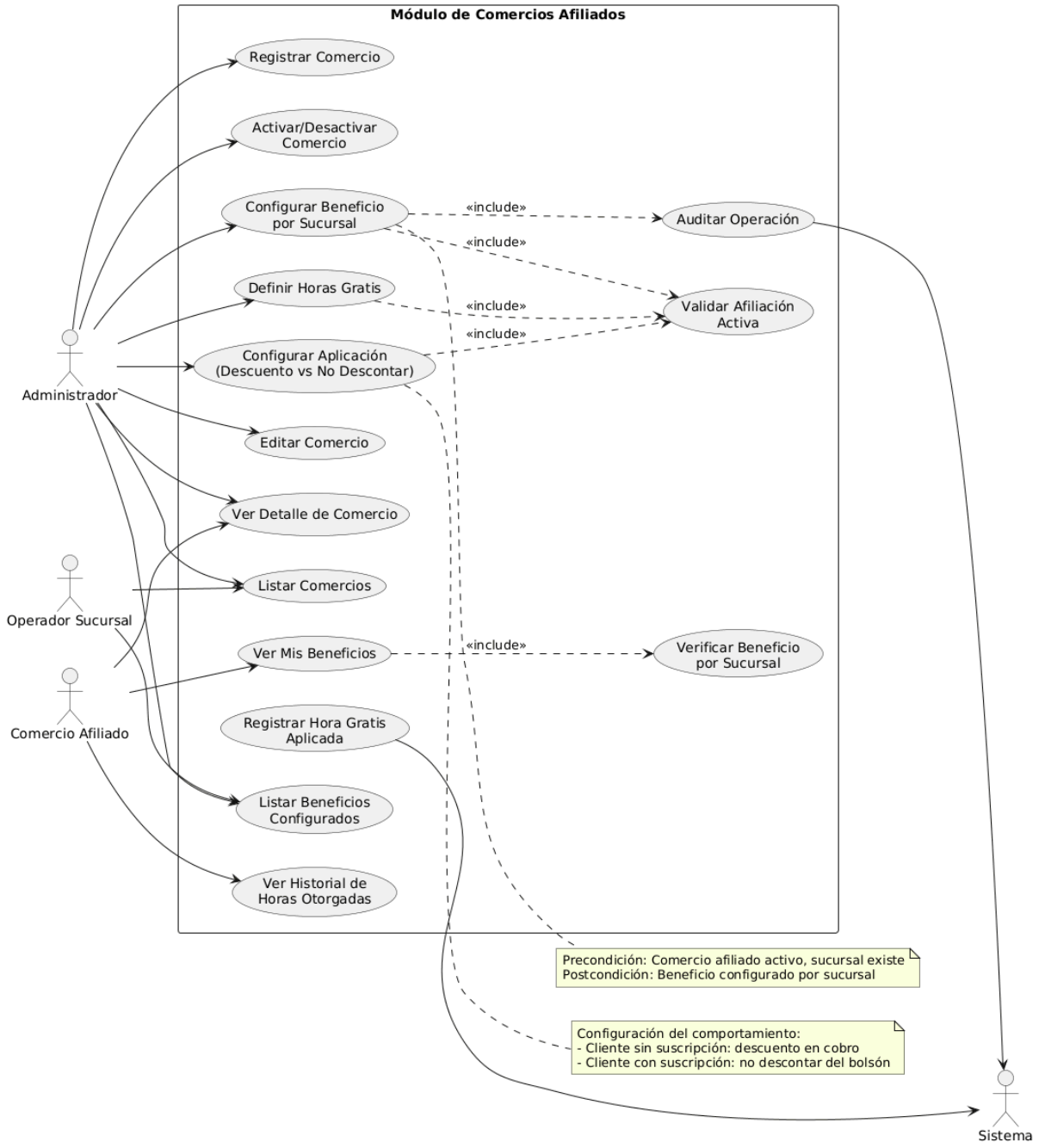
UC14 - Liquidaciones a Comercios Afiliados



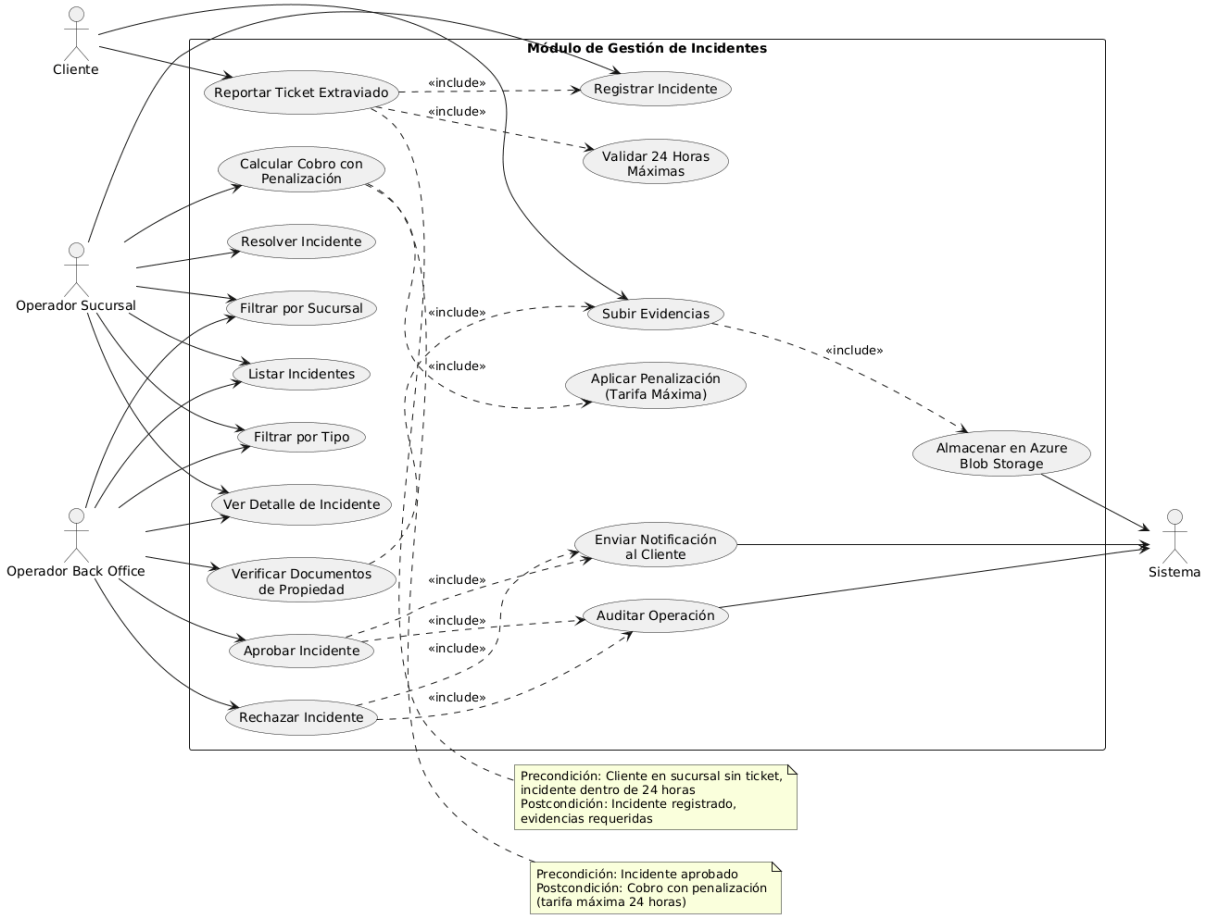
UC10 - Gestión de Permisos Temporales



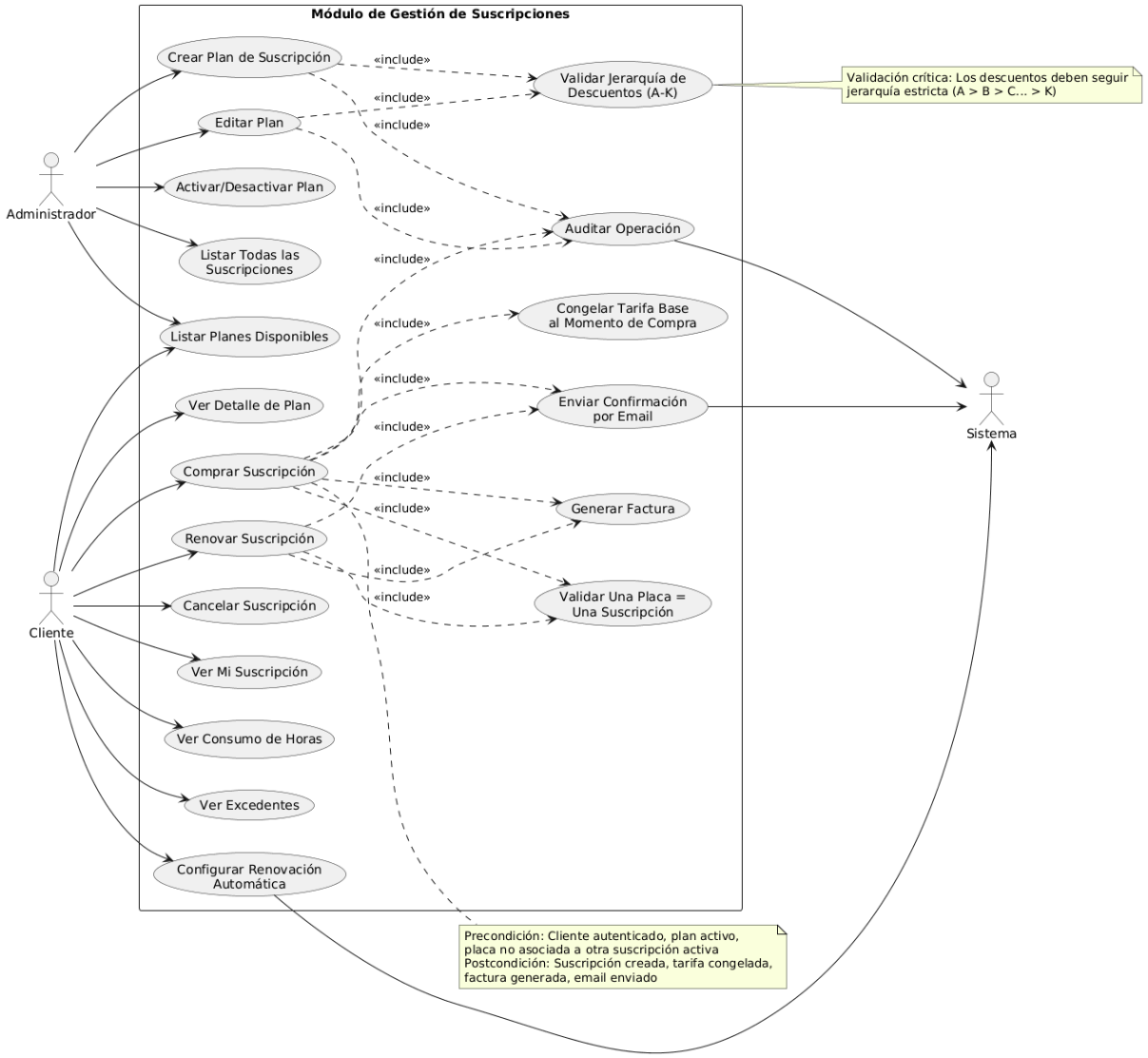
UC07 - Gestión de Comercios Afiliados



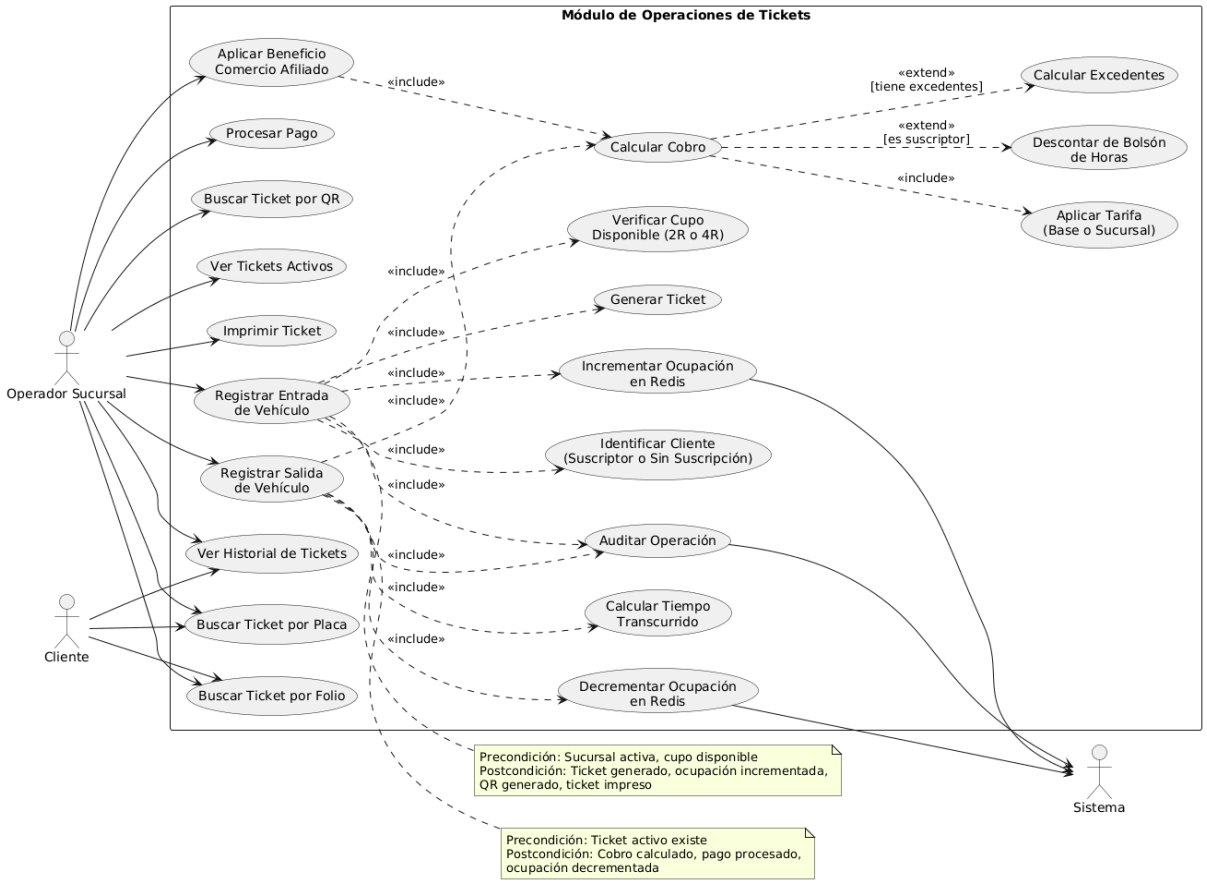
UC06 - Gestión de Incidentes



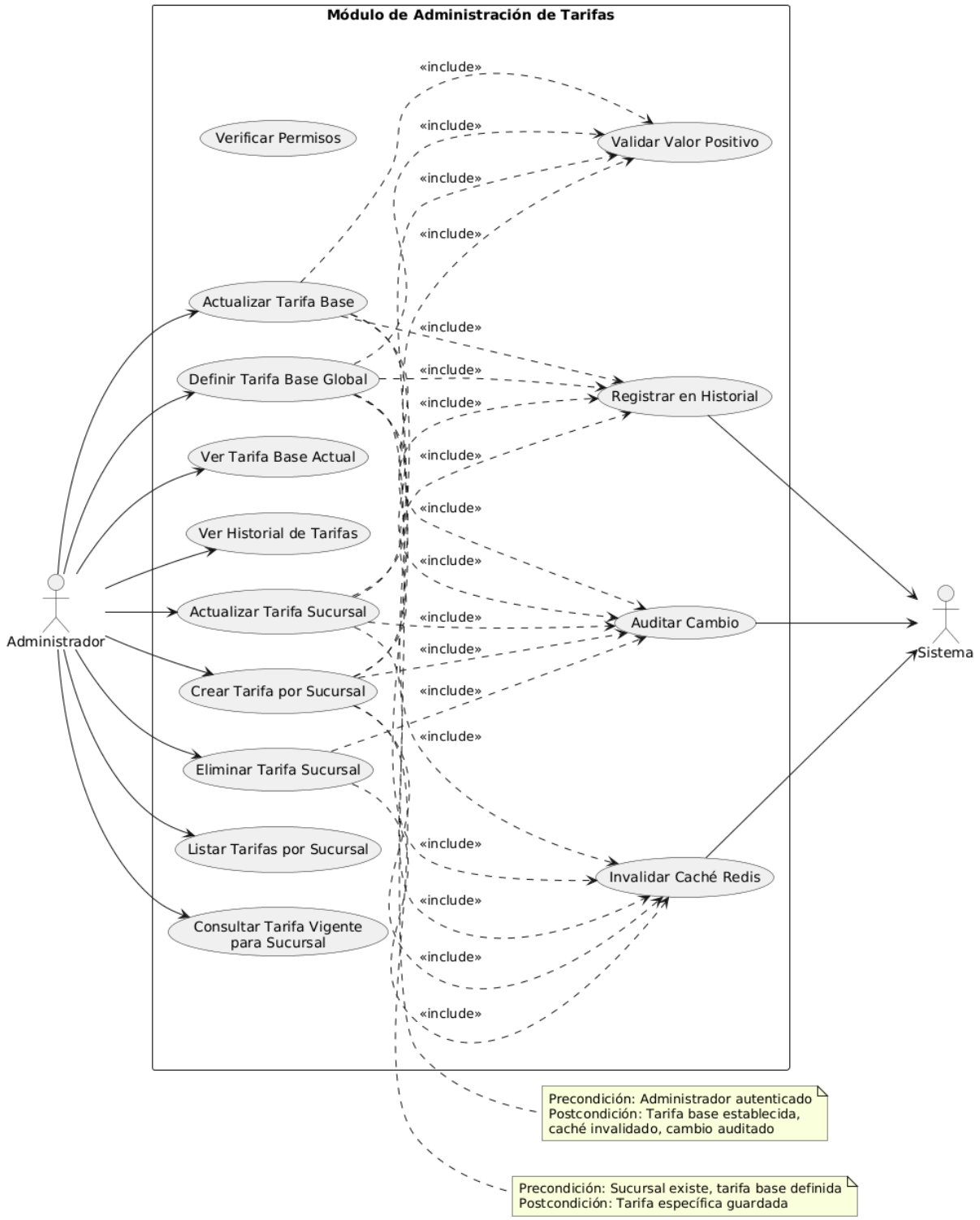
UC04 - Gestión de Suscripciones



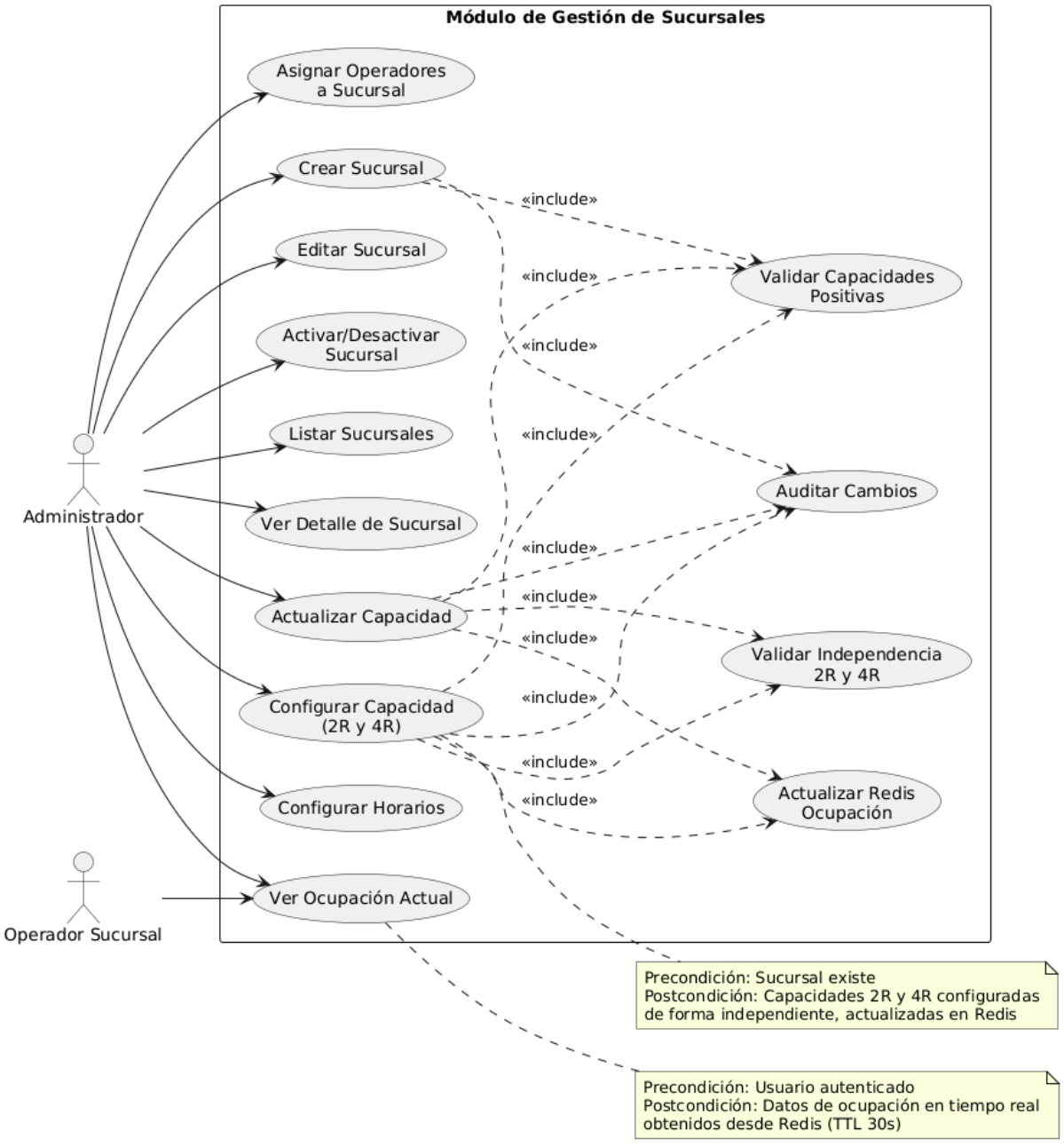
UC05 - Operaciones de Tickets (Entrada/Salida)



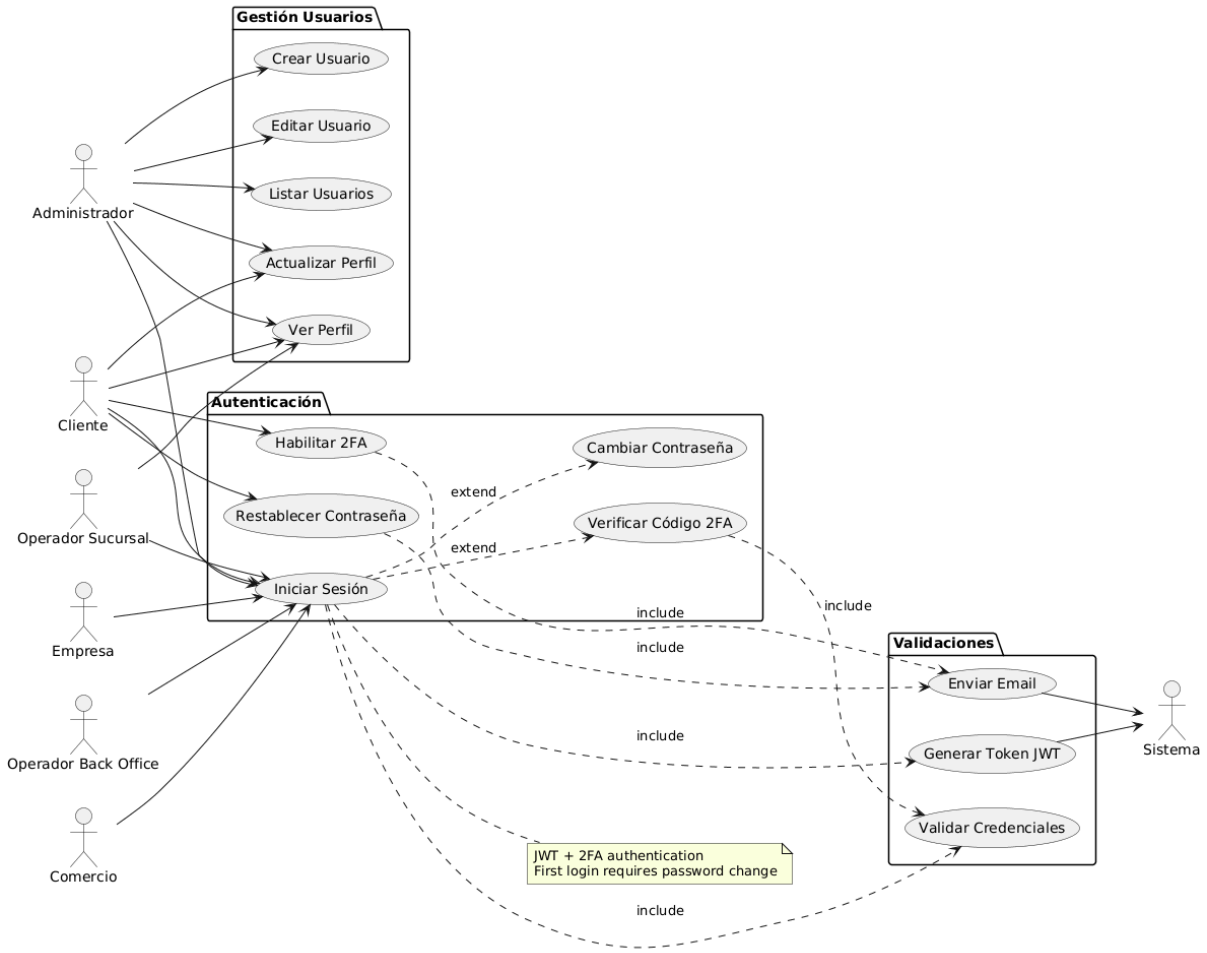
UC02 - Administración de Tarifas



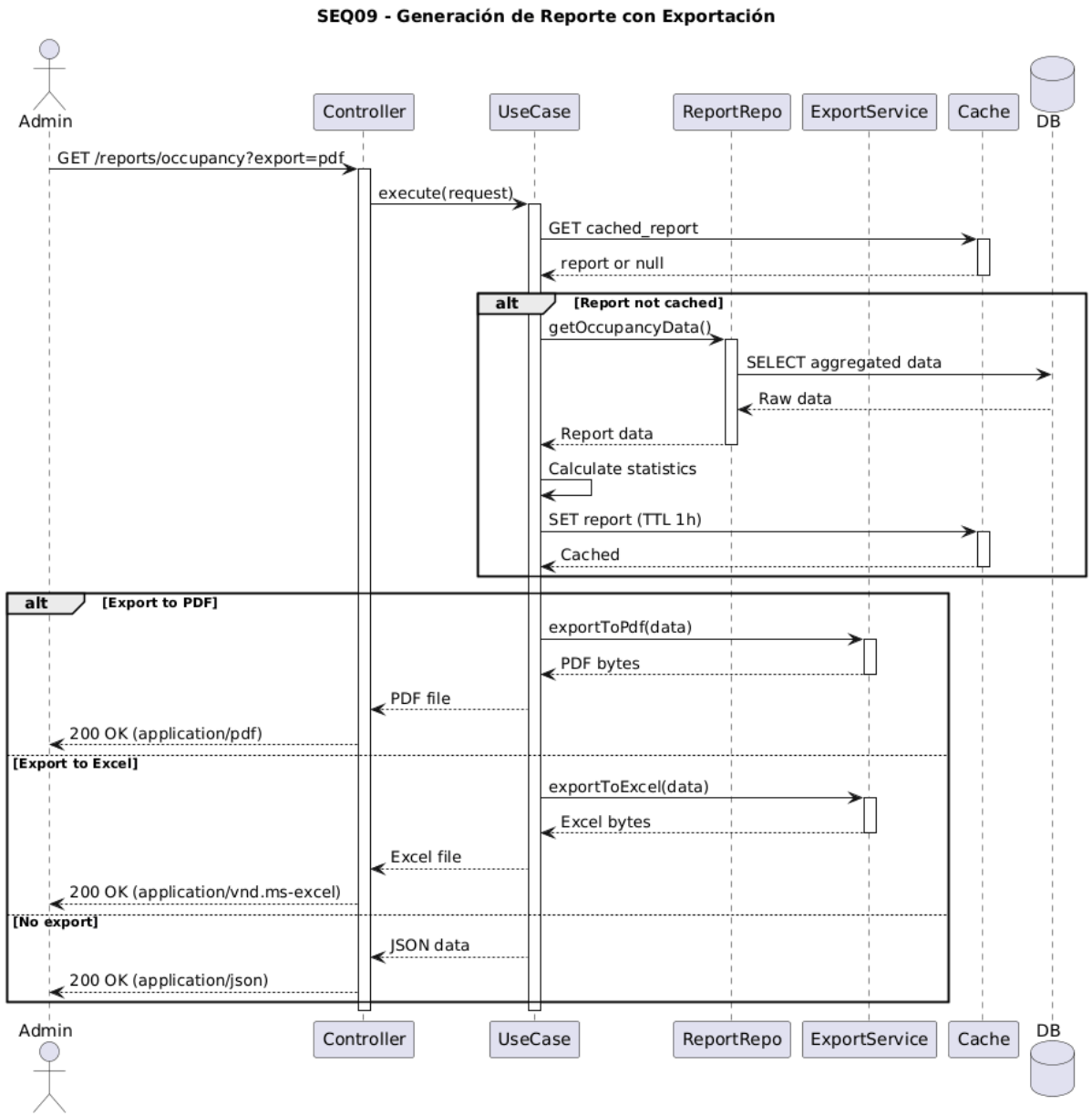
UC03 - Gestión de Sucursales



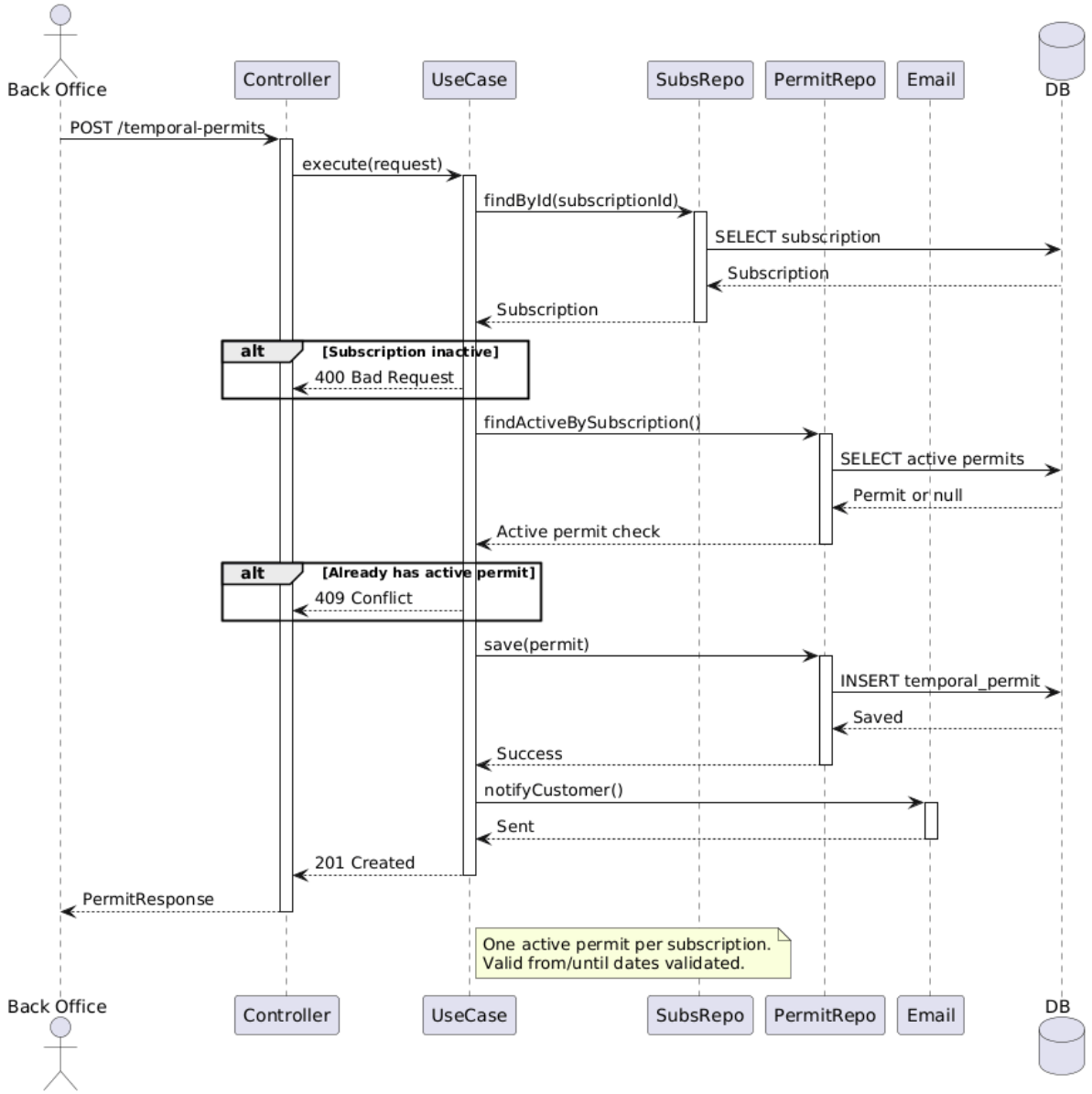
UC01 - Autenticación y Gestión de Usuarios



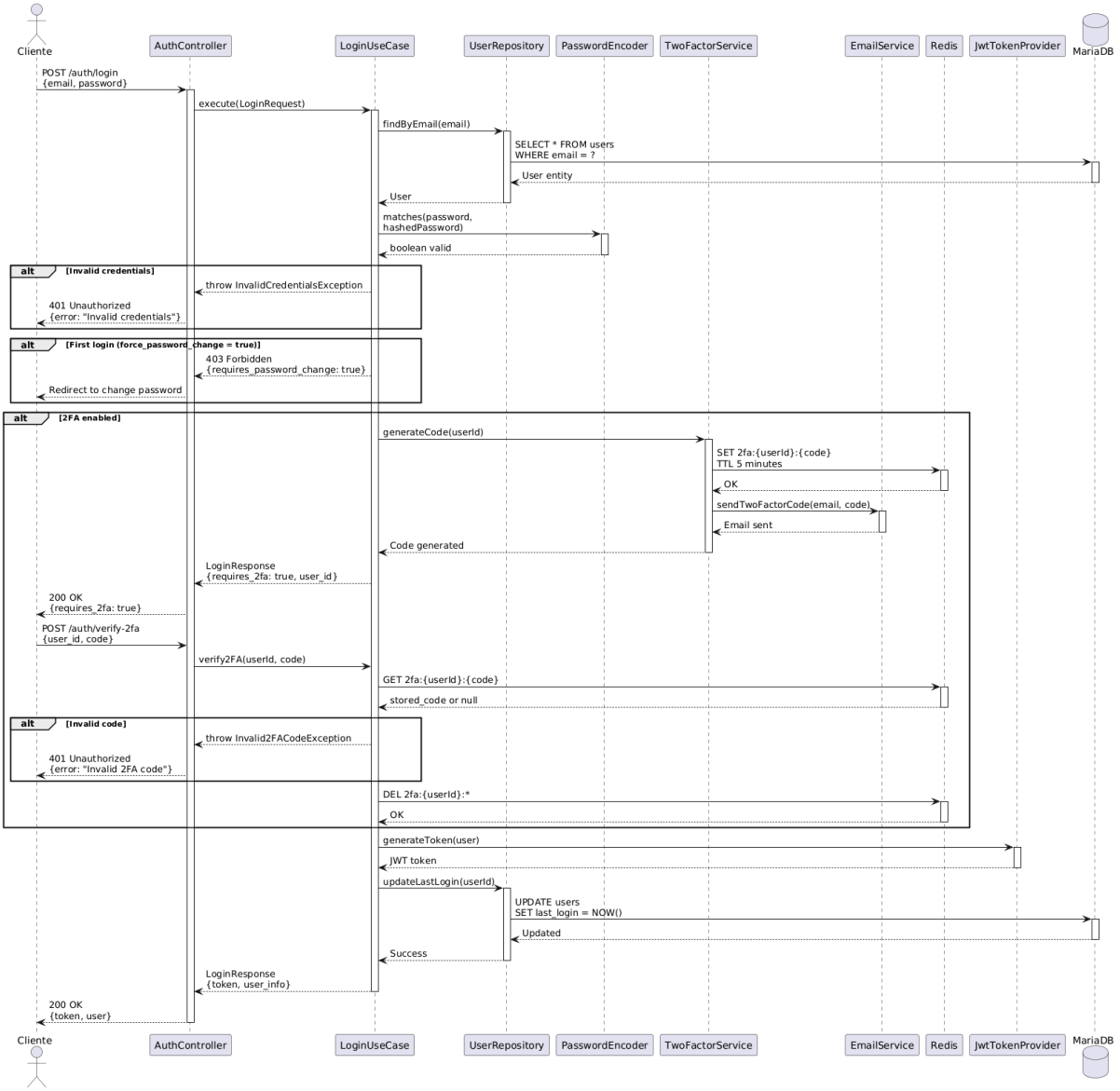
Secuencia.


SEQ09 - Generación de Reporte con Exportación
Admin
Controller
UseCase
ReportRepo
ExportService
Cache
DB
GET /reports/occupancy?export=pdf
execute(request)
GET cached_report
report or null
alt
[Report not cached]
getOccupancyData()
SELECT aggregated data
Raw data
Report data
Calculate statistics
SET report (TTL 1h)
Cached
alt
[Export to PDF]
exportToPdf(data)
PDF bytes
PDF file
200 OK (application/pdf)
[Export to Excel]
exportToExcel(data)
Excel bytes
Excel file
200 OK (application/vnd.ms-excel)
[No export]
JSON data
200 OK (application/json)

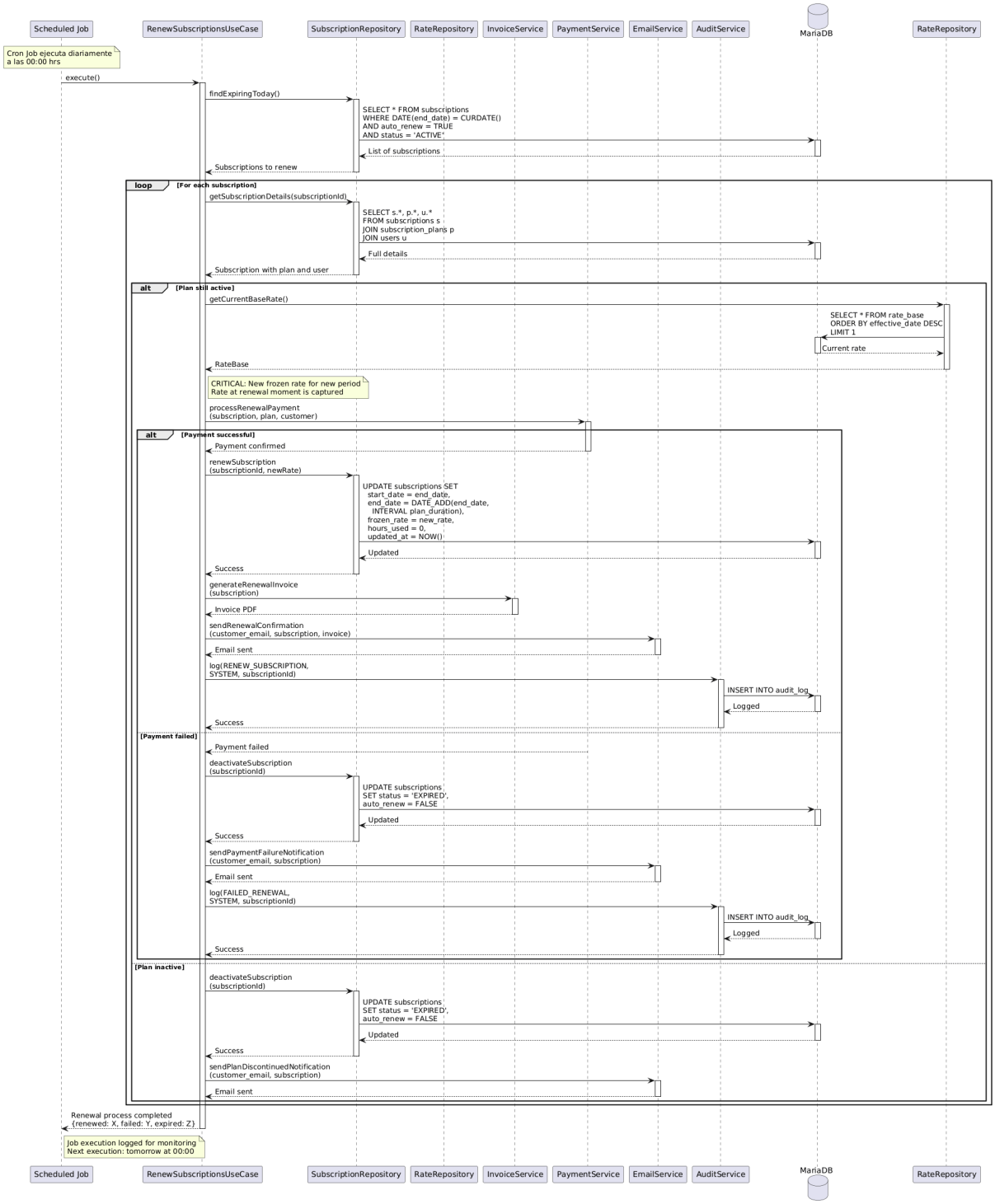
SEQ07 - Emisión de Permiso Temporal



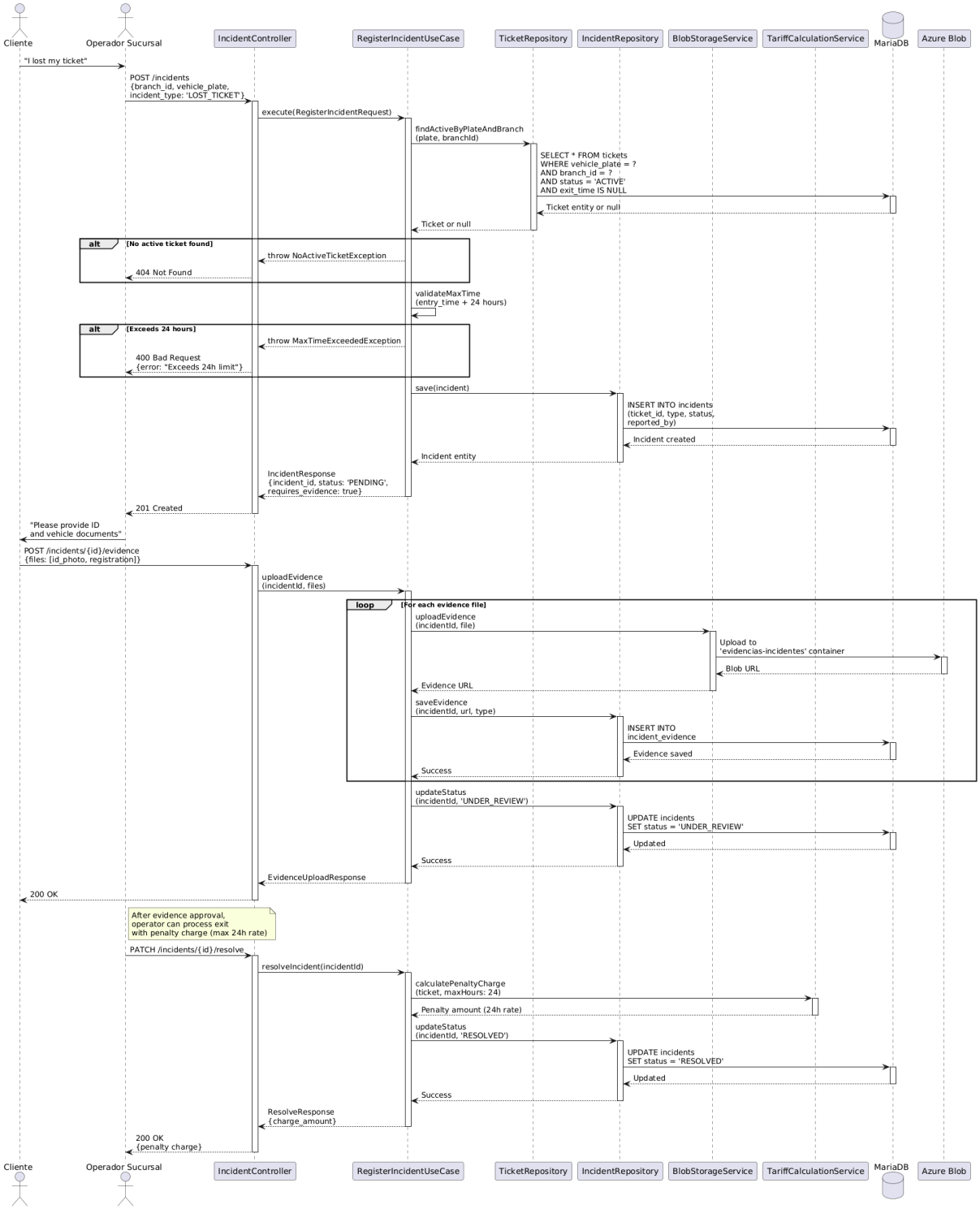
SEQ01 - Autenticación con 2FA



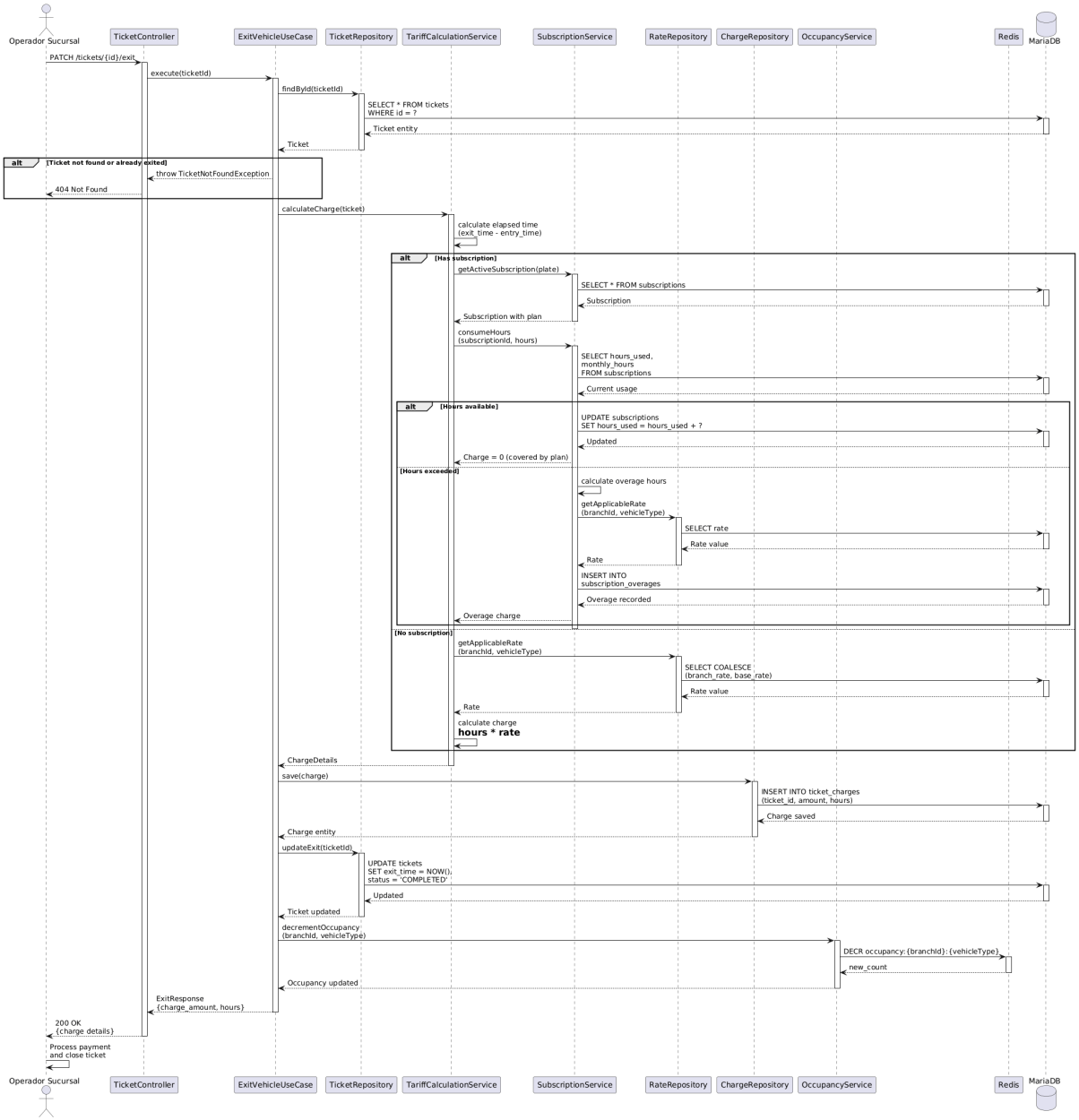
SEQ10 - Renovación Automática de Suscripción



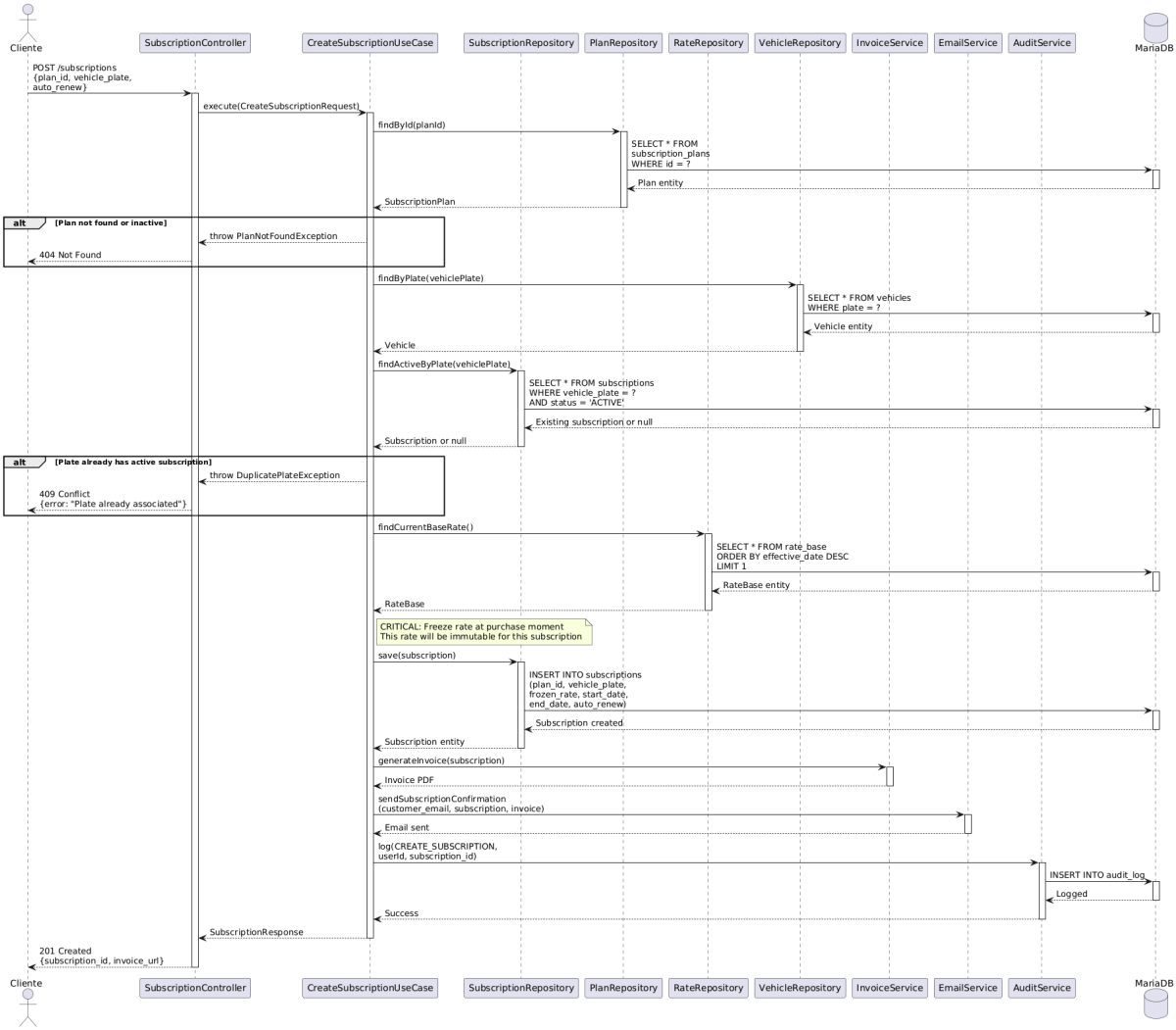
SEQ08 - Registro de Incidente por Ticket Extraviado



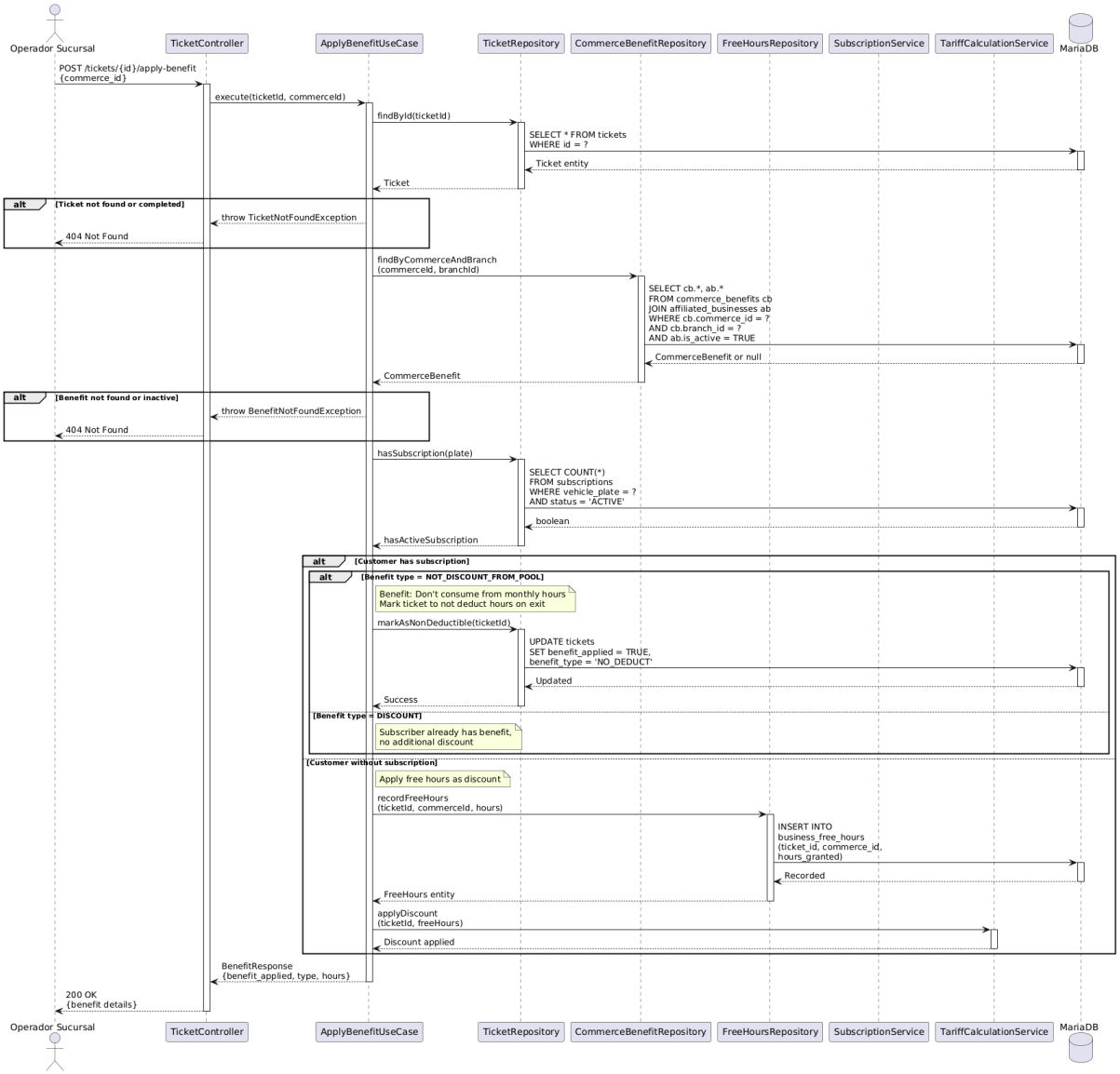
SEQ04 - Salida de Vehículo con Cobro



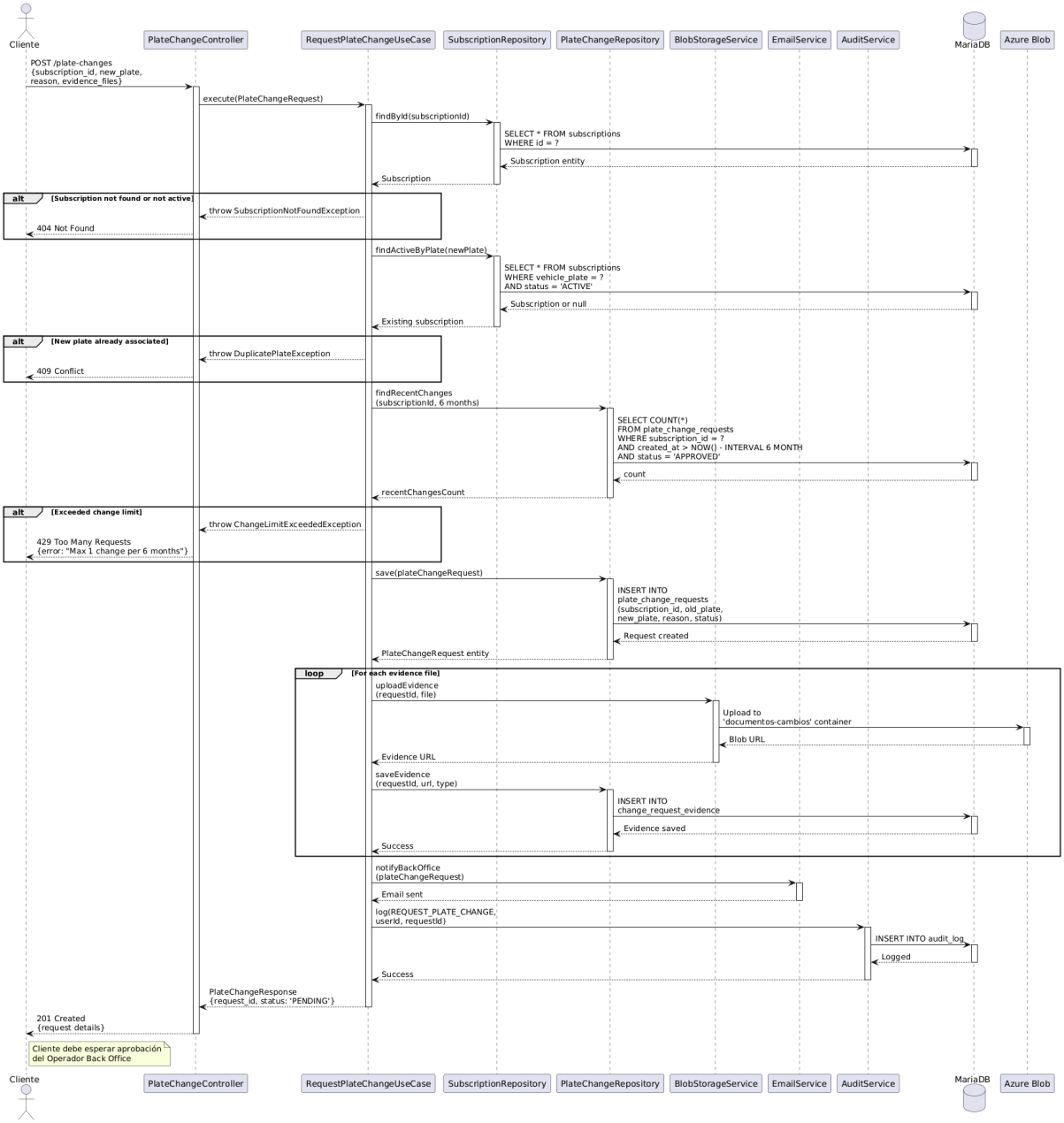
SEQ02 - Compra de Suscripción



SEQ05 - Aplicación de Beneficio de Comercio Afiliado



SEQ06 - Solicitud de Cambio de Placa



SEQ03 - Entrada de Vehículo

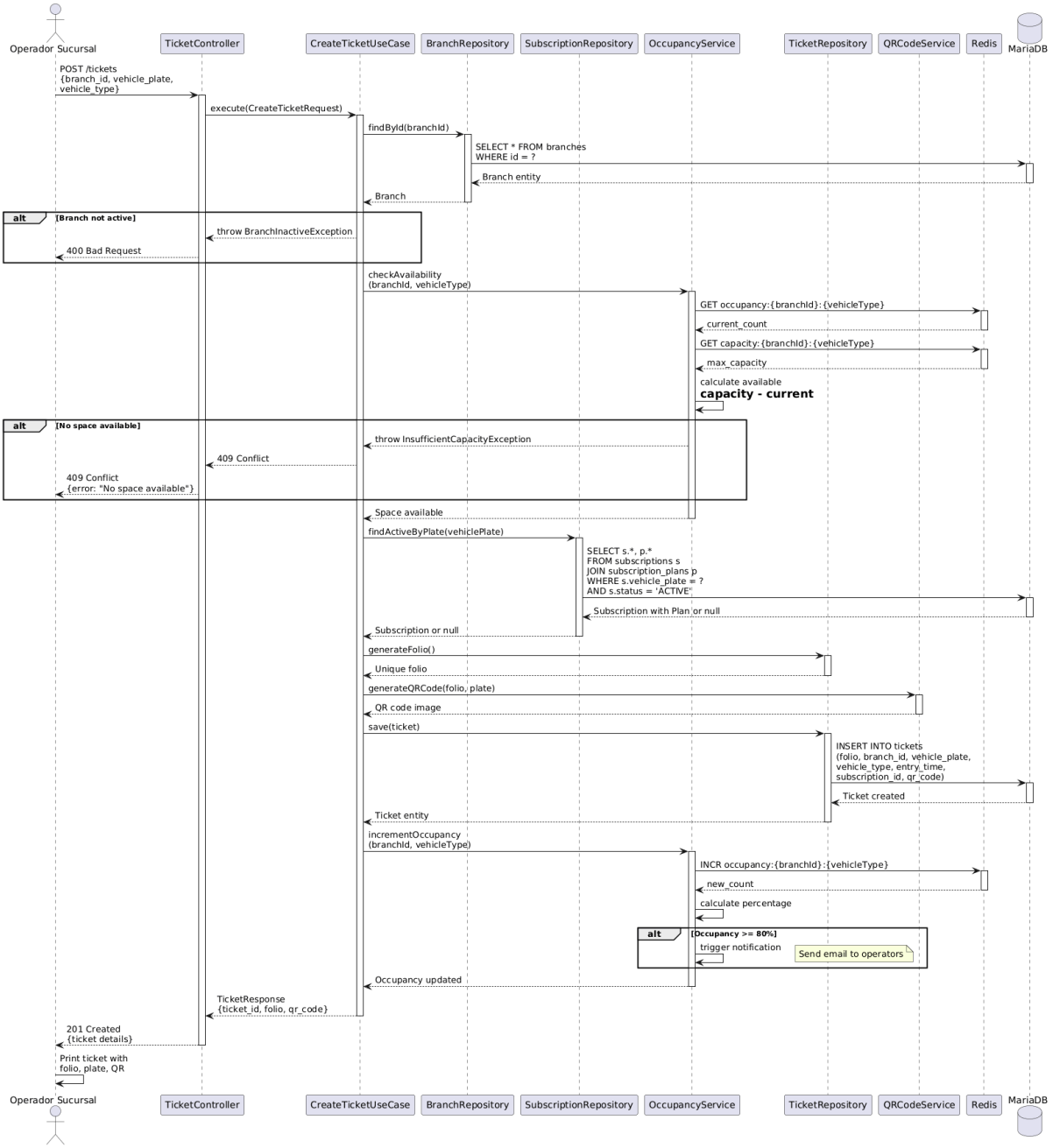


Diagrama Entidad Relación.

ANEXOS

- *Carpeta Google Drive:* Carpeta
- *Repositorio Backend:* https://github.com/DiegoMaldonado19/ayd-proyecto-final-backend
- *Repositorio Frontend:* https://github.com/DiegoMaldonado19/ayd-proyecto-final-frontend