Sistema de Reservas Municipales — Documentación Técnica (Resumen)

Backend Spring Boot + PostgreSQL (Docker) y Cliente JavaFX

1. Introducción

Este documento resume la instalación, ejecución, seguridad, endpoints y pruebas del Sistema de Reservas Municipales, incluyendo el backend (Spring Boot 3 + PostgreSQL) desplegado con Docker y el cliente de escritorio JavaFX. Se cubre además el módulo de métricas administrativas (RF10) y la gestión de horarios de espacios (RF15), así como buenas prácticas aplicadas y lineamientos de pruebas con Postman. El objetivo es ofrecer una guía compacta (4–5 páginas) para docentes y revisores.

2. Arquitectura General

Componentes:

- • Base de datos: PostgreSQL en contenedor Docker; inicialización con script SQL (tablas, índices, datos base).
- • Backend: Spring Boot 3 (REST), empaquetado como JAR; Actuator habilitado y protegido por Spring Security.
- • Cliente: Aplicación JavaFX (Java 21), consume API REST; soporta roles y flujos de QR.
- • Seguridad: Autenticación con JWT; autorización por rol (ADMIN, SUPERVISOR, USER) con @PreAuthorize.
- • Caché: Caffeine (TTL 10 minutos) para métricas del Dashboard.

3. Requisitos

- • Docker y docker-compose instalados y funcionando.
- • Java 21 (o uso de ./mvnw.cmd como wrapper).
- • (Cliente) JDK 21, Maven 3.8+, conectividad a http://localhost:8080.

4. Backend: Despliegue y Ejecución

4.1 Levantar PostgreSQL con docker-compose

Desde la raíz del proyecto:

PowerShell:

- docker-compose up -d

Verificar logs de inicialización de la base (opcional):

- docker-compose logs -f db

4.2 Construir y arrancar la aplicación (perfil docker)

Empaquetar:

- .\mvnw.cmd clean package -DskipTests

Arrancar con perfil docker (forma robusta):

- java -jar -Dspring.profiles.active=docker target\app.jar

Nota: El uso directo de spring-boot:run con perfiles en PowerShell puede fallar; se recomienda package + java -jar.

4.3 Actuator /health protegido

El endpoint /actuator/health existe y por defecto retorna 401 sin credenciales. Si no configuraste usuario/clave, Spring imprime una contraseña aleatoria al iniciar (línea “Using generated security password: <password>”).

4.4 Probar /actuator/health

PowerShell:

- Invoke-WebRequest http://localhost:8080/actuator/health

Esperado: 401 Unauthorized (sin credenciales).

Con credenciales (Git Bash/WSL):

- curl -u user:password http://localhost:8080/actuator/health

4.5 Permitir /actuator/health sin autenticación (opcional)

En application-docker.yml añadir una regla de seguridad que exponga GET /actuator/health públicamente, solo para pruebas locales.

5. Métricas Administrativas (RF10) — Dashboard

El sistema expone un endpoint de métricas administrativas protegido por JWT y roles ADMIN/SUPERVISOR.

Requisitos

- • Autenticación con JWT.
- • Rol: ADMIN o SUPERVISOR.

5.1 Autenticación (obtener token JWT)

PowerShell (con cuerpo JSON) o curl desde Git Bash/WSL:

- curl -X POST http://localhost:8080/api/auth/login -H "Content-Type: application/json" -d "{\"email\":\"admin@test.com\",\"password\":\"<tu_clave>\"}"

La respuesta incluye el campo token. Cópialo.

5.2 Consultar dashboard con token

- curl -H "Authorization: Bearer <token>" http://localhost:8080/api/admin/dashboard

5.3 Estructura de respuesta (categorías)

1. 1) General Metrics: totalReservations, totalSpaces, totalUsers, activeReservations
2. 2) Reservations By Status: CONFIRMED, PENDING, CANCELLED, COMPLETED
3. 3) Revenue Metrics: currentMonthRevenue, lastMonthRevenue, percentageChange
4. 4) Top Spaces: Top 5 por reservationCount y totalRevenue
5. 5) Temporal Metrics: reservationsToday, ThisWeek, ThisMonth, ByDayOfWeek, ByHour, mostPopularDay/hour

5.4 Optimización con caché (Caffeine)

- • Primera llamada: ~150–200 ms; posteriores (≤10 min): ~5–10 ms; tras 10 min se recalcula.

5.5 Pruebas con Postman

- • Crear colección “Reservas Municipales”.
- • Request de Login (POST /api/auth/login).
- • Request de Dashboard (GET /api/admin/dashboard) con header Authorization: Bearer <token>.

5.6 Casos de error

- • 403 Forbidden: usuario sin rol ADMIN/SUPERVISOR.
- • 401 Unauthorized: token inválido/expirado.
- • 500 Internal Server Error: revisar logs y configuración.

6. Gestión de Horarios de Espacios (RF15)

Permite configurar disponibilidad por día y múltiples bloques por espacio. La validación de reservas usa estos bloques. Si un espacio no tiene horarios, se mantiene comportamiento backward compatible (permite cualquier horario).

6.1 Estructura

- • Días de la semana: 0=Dom, 1=Lun, …, 6=Sáb.
- • Bloques: (timeFrom, timeTo) por día; múltiples por día.

6.2 Endpoints

6. 1) GET /api/spaces/{id}/schedules — JWT requerido (cualquier usuario autenticado).
7. 2) POST /api/spaces/{id}/schedules — Rol ADMIN o SUPERVISOR.

Validaciones: espacio existe; weekday 0..6; timeFrom < timeTo; sin solapes.

8. 3) DELETE /api/spaces/{id}/schedules/{scheduleId} — ADMIN o SUPERVISOR.
9. 4) DELETE /api/spaces/{id}/schedules — ADMIN (elimina todos).

6.3 Validación al crear reservas

- • Sin horarios: permitir.
- • Con horarios: exigir que toda la reserva esté dentro de un bloque válido del día.
- • Errores típicos: fuera de horario; día sin bloques; solapes al configurar.

7. Cliente JavaFX — Resumen

Interfaz moderna (Material-like), navegación lateral, roles diferenciados y flujo QR.

Tecnologías

Java 21, JavaFX 21, Maven, Apache HttpClient 5.x, Gson 2.11.0, Jackson 2.17.2, ZXing 3.5.3, Webcam Capture 0.3.12.

Funcionalidades por rol

- • USER: Buscar espacios, crear/cancelar reservas, ver QR, reseñas.
- • SUPERVISOR: Validar QR con webcam; historial de validaciones.
- • ADMIN: Gestión integral de espacios/usuarios/reservas, dashboard, reportes.

Buenas prácticas

- • MVC, SOLID, DTOs; patrones: Singleton (UserSession, ApiClient), Factory (diálogos), Observer (listeners JavaFX).
- • Limpieza: nombres descriptivos, métodos pequeños, JavaDoc, sin imports muertos.

8. Solución de Problemas

- • Backend no inicia: revisar puertos, logs, variables de entorno, perfil activo.
- • Actuator 401: usar usuario/clave o exponer /actuator/health localmente.
- • Error conexión cliente: verificar http://localhost:8080 y properties del frontend.
- • Escaneo QR: permisos de cámara; una sola app usando webcam; reiniciar.

9. Conclusión

Se levantó PostgreSQL y la aplicación con el perfil docker; la API responde en localhost:8080. Los endpoints de Actuator están protegidos por configuración por defecto. El Dashboard (RF10) ofrece métricas en tiempo real con caché; la gestión de horarios (RF15) garantiza reservas dentro de la disponibilidad configurada. La arquitectura y las buenas prácticas aplicadas permiten mantener y escalar el sistema con claridad.