Backend - Sistema de Reservas Lago Escondido

API REST para gestión de reservas del Lago Escondido. Backend desarrollado con Spring Boot 3 + Java 21 + PostgreSQL.

🚀 Quick Start - Desarrollo Local

Opción 1: Docker (Recomendado)

```
# 1. Entrar al directorio del backend
cd backend-reservas-lago

# 2. Levantar servicios (backend + PostgreSQL)
docker compose -f docker-compose.dev.yml up -d

# 3. Verificar que esté corriendo
curl http://localhost:8080/actuator/health

# 4. Abrir Swagger UI en el navegador
# http://localhost:8080/docs
```

Servicios disponibles:

- Backend API: http://localhost:8080
- Swagger UI: http://localhost:8080/docs
- PostgreSQL: localhost:5432 (usuario: `postgres`, password: `postgres`, db: `lago`)

Comandos útiles:

```
# Ver logs en tiempo real
docker compose -f docker-compose.dev.yml logs -f

# Reiniciar solo el backend
docker compose -f docker-compose.dev.yml restart app

# Parar todo
docker compose -f docker-compose.dev.yml down

# Parar y eliminar datos (reset completo de DB)
docker compose -f docker-compose.dev.yml down -v
```

Opción 2: Maven Local (Requiere JDK 21)

```
# 1. Levantar solo PostgreSQL con Docker
docker compose -f docker-compose.dev.yml up -d db

# 2. Configurar perfil de desarrollo
# Windows PowerShell:
$Env:SPRING_PROFILES_ACTIVE="dev"

# Linux/macOS:
export SPRING_PROFILES_ACTIVE=dev

# 3. Arrancar el backend con Maven
# Windows:
.\mvnw.cmd spring-boot:run

# Linux/macOS:
./mvnw spring-boot:run

# 4. Verificar
curl http://localhost:8080/actuator/health
```

🔑 Credenciales de Desarrollo

Usuario Admin por Defecto

- **Email:** `admin@lago-escondido.com`
- **Password:** `admin123`

Base de Datos (PostgreSQL)

- **Host:** `localhost:5432`
- **Database:** `lago`
- **Usuario:** `postgres`
- **Password:** `postgres`

Conexión directa a la DB:

```
# Con Docker corriendo:
docker exec -it lago-postgres-dev psql -U postgres -d lago
```

📱 WhatsApp (Opcional en Desarrollo)

El sistema envía notificaciones por WhatsApp usando Twilio. Es **opcional** para desarrollo local.

Configurar WhatsApp:

1. Crear archivo `.env` en la raíz del backend:

```
TWILIO_ACCOUNT_SID=tu_account_sid_aqui
TWILIO_AUTH_TOKEN=tu_auth_token_aqui
```

2. Obtener credenciales en: https://console.twilio.com

3. Reiniciar el backend:

```
docker compose -f docker-compose.dev.yml restart app
```

Deshabilitar WhatsApp:

Editar docker-compose.dev.yml línea 33:

```
WHATSAPP_ENABLED: "false"
```

📋 Endpoints Principales

Autenticación (Público)

Login

```
POST /api/auth/login
Content-Type: application/json

{
  "email": "admin@lago-escondido.com",
  "password": "admin123"
}
```

Respuesta:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "email": "admin@lago-escondido.com",
  "firstName": "Admin",
  "lastName": "Sistema"
}
```

Usar el token en requests protegidos:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

API Pública (Sin autenticación)

Consultar Disponibilidad de un Día

```
GET /api/availability?date=2025-01-15
```

Respuesta:

```
{
  "date": "2025-01-15",
  "capacity": 30,
  "currentReservations": 12,
  "availableSlots": 18
}
```

Consultar Disponibilidad de un Mes

```
GET /api/availability?month=2025-01
```

Respuesta:

```
[
  {
    "date": "2025-01-15",
    "capacity": 30,
    "currentReservations": 12,
    "availableSlots": 18
  },
  ...
]
```

Crear Reserva

```
POST /api/reservations
Content-Type: application/json

{
  "visitDate": "2025-01-15",
```

```
  "firstName": "Juan",
  "lastName": "Pérez",
  "dni": "12345678",
  "phone": "+5493517734676",
  "email": "juan@email.com",
  "circuit": "A",
  "visitorType": "INDIVIDUAL",
  "adults18Plus": 2,
  "children2To17": 1,
  "babiesLessThan2": 0,
  "reducedMobility": 0,
  "howHeard": "REDES_SOCIALES",
  "acceptedPolicies": true
}
```

Obtener Resumen de Reserva

```
GET /api/reservations/{id}
```

API Administrativa (Requiere JWT)

Todos estos endpoints requieren header `Authorization: Bearer {token}`

Listar Reservas

```
GET /api/admin/reservations

# Con filtros:
GET /api/admin/reservations?date=2025-01-15
GET /api/admin/reservations?status=CONFIRMED
GET /api/admin/reservations?dni=12345678
```

Confirmar Reserva

```
POST /api/admin/reservations/{id}/confirm
```

☑ Envía notificación por WhatsApp al cliente (si está habilitado)

Cancelar Reserva

```
POST /api/admin/reservations/{id}/cancel
```

☑ Envía notificación por WhatsApp al cliente (si está habilitado)

Exportar a Excel

```
GET /api/admin/reservations/export

# Con filtros:
GET /api/admin/reservations/export?month=2025-01
GET /api/admin/reservations/export?year=2025
GET /api/admin/reservations/export?status=CONFIRMED
GET /api/admin/reservations/export?visitorType=INDIVIDUAL
```

Configurar Capacidad de un Día

```
PUT /api/admin/availability/2025-01-15
Content-Type: application/json

{
  "capacity": 50
}
```

Crear Evento Especial

```
POST /api/admin/eventos
Content-Type: application/json

{
  "visitDate": "2025-02-14",
  "organizationName": "Escuela Primaria",
  "contactName": "María González",
  "phone": "+5493517734676",
  "email": "maria@escuela.com",
  "totalPeople": 40,
  "observations": "Grupo escolar - llegada 10am"
}
```

Configuración de Reservas Educativas

```
# Consultar estado
GET /api/admin/config/educational-reservations

# Habilitar/deshabilitar
PUT /api/admin/config/educational-reservations
Content-Type: application/json
```

```
{
  "enabled": true
}
```

Gestión de Usuarios Admin

Crear Usuario Admin

```
POST /api/admin/users
Content-Type: application/json
Authorization: Bearer {token}

{
  "email": "admin2@lago-escondido.com",
  "password": "password123",
  "firstName": "Nuevo",
  "lastName": "Admin"
}
```

Listar Usuarios

```
GET /api/admin/users
Authorization: Bearer {token}
```

Obtener Usuario por Email

```
GET /api/admin/users/admin@lago-escondido.com
Authorization: Bearer {token}
```

Actualizar Usuario

```
PUT /api/admin/users/{userId}
Content-Type: application/json
Authorization: Bearer {token}

{
  "firstName": "Nombre Actualizado",
  "lastName": "Apellido Actualizado",
  "enabled": true
}
```

Eliminar Usuario

```
DELETE /api/admin/users/{userId}
Authorization: Bearer {token}
```

Testing

Ejecutar Tests

```
# Windows
.\mvnw.cmd test

# Linux/macOS
./mvnw test
```

Probar Endpoints con cURL

Login y obtener token:

```
curl -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"admin@lago-escondido.com","password":"admin123"}'
```

Consultar disponibilidad:

```
curl http://localhost:8080/api/availability?date=2025-01-15
```

Listar reservas (con token):

```
TOKEN="tu_token_aqui"
curl http://localhost:8080/api/admin/reservations \
  -H "Authorization: Bearer $TOKEN"
```

Documentación API

Swagger UI (Solo Desarrollo)

- URL: http://localhost:8080/docs
- Interfaz visual interactiva para probar todos los endpoints

- Autenticación JWT integrada

OpenAPI JSON

- URL: http://localhost:8080/v3/api-docs
- Definición completa de la API en formato OpenAPI 3

Importar a Postman:

1. Abrir Postman
2. Import → Link → `http://localhost:8080/v3/api-docs`
3. Listo para usar

También hay colección Postman documentada: POSTMAN_README.md

🏗 Arquitectura

Stack Tecnológico

- **Framework:** Spring Boot 3.5.5
- **Java:** 21 (LTS)
- **Base de Datos:** PostgreSQL 16
- **Migraciones:** Flyway
- **Autenticación:** JWT (jjwt 0.12.3)
- **Notificaciones:** WhatsApp (Twilio 10.1.0)
- **Exportación:** Excel (Apache POI 5.2.5)
- **Documentación:** Swagger/OpenAPI (SpringDoc 2.8.11)

Estructura del Proyecto

```
src/main/java/com.luismunozse.reservalago/
├── config/                    # Configuración (Security, CORS, JWT, OpenAPI)
├── controller/                # Endpoints REST
│   ├── PublicController.java      # API pública
│   ├── AuthController.java        # Login JWT
│   ├── AdminController.java       # API administrativa
│   └── UserController.java        # Gestión usuarios
├── service/                   # Lógica de negocio
│   ├── ReservationService.java    # Gestión de reservas
│   ├── AvailabilityService.java   # Disponibilidad y capacidad
│   ├── WhatsAppService.java       # Notificaciones WhatsApp
│   ├── JwtService.java            # JWT tokens
│   └── UserService.java           # Usuarios admin
├── repo/                      # Acceso a datos (JPA)
├── model/                     # Entidades JPA
│   ├── Reservation.java
│   ├── AvailabilityRule.java
│   ├── User.java
│   └── SystemConfig.java
└── dto/                       # DTOs (Request/Response)
```

Base de Datos

Migraciones Flyway (se ejecutan automáticamente al iniciar):

- V4: Tablas principales (reservations, availability_rules)
- V5: Constraint único (1 reserva por DNI por día)
- V6: Tabla de usuarios admin
- V7: Nombres de usuarios
- V8: Configuración del sistema
- V9: Visitantes adicionales
- V10: Limpieza de columnas obsoletas

Consultar historial de migraciones:

```
SELECT * FROM flyway_schema_history;
```

🔧 Configuración Avanzada

Variables de Entorno (Desarrollo)

Estas ya están configuradas en docker-compose.dev.yml:

```
SPRING_PROFILES_ACTIVE: dev
SPRING_DATASOURCE_URL: jdbc:postgresql://db:5432/lago
SPRING_DATASOURCE_USERNAME: postgres
SPRING_DATASOURCE_PASSWORD: postgres
DEFAULT_CAPACITY: 30
ALLOWED_ORIGINS: http://localhost:3000,http://localhost:3002
WHATSAPP_ENABLED: "true"
TWILIO_ACCOUNT_SID: ${TWILIO_ACCOUNT_SID}
TWILIO_AUTH_TOKEN: ${TWILIO_AUTH_TOKEN}
```

Cambiar Puerto del Backend

Editar docker-compose.dev.yml línea 38:

```
ports:
  - "8081:8080"  # Cambia 8081 por el puerto que prefieras
```

Cambiar Capacidad por Defecto

Editar docker-compose.dev.yml línea 29:

```
DEFAULT_CAPACITY: 50  # Cambia 30 por el valor deseado
```

Configurar CORS para Otro Frontend

Editar docker-compose.dev.yml línea 30:

```
ALLOWED_ORIGINS:
http://localhost:3000,http://localhost:5173,http://192.168.1.100:3000
```

🐛 Troubleshooting

El backend no inicia

Verificar que PostgreSQL esté corriendo:

```
docker compose -f docker-compose.dev.yml ps
```

Ver logs del backend:

```
docker compose -f docker-compose.dev.yml logs app
```

Logs de PostgreSQL:

```
docker compose -f docker-compose.dev.yml logs db
```

Error de conexión a la base de datos

Resetear completamente la DB:

```
# Parar y eliminar volúmenes
docker compose -f docker-compose.dev.yml down -v

# Levantar de nuevo
docker compose -f docker-compose.dev.yml up -d
```

Puerto 8080 ocupado

Ver qué proceso usa el puerto:

```
# Windows
netstat -ano | findstr :8080

# Linux/macOS
lsof -i :8080
```

Cambiar puerto del backend: Editar `docker-compose.dev.yml` o usar:

```
# Sin Docker
SERVER_PORT=8081 ./mvnw spring-boot:run
```

Error CORS desde el frontend

Verificar CORS configurado en docker-compose.dev.yml:

```
ALLOWED_ORIGINS: http://localhost:3000,http://localhost:3002
```

Debe incluir la URL **exacta** desde donde el frontend hace las peticiones (incluyendo protocolo y puerto).

WhatsApp no envía mensajes

Verificar configuración:

1. Archivo `.env` existe en la raíz del backend
2. Contiene `TWILIO_ACCOUNT_SID` y `TWILIO_AUTH_TOKEN`
3. `WHATSAPP_ENABLED: "true"` en docker-compose.dev.yml

Ver logs del servicio:

```
docker compose -f docker-compose.dev.yml logs -f app | grep WhatsApp
```

Deshabilitar temporalmente:

```
# docker-compose.dev.yml línea 33
WHATSAPP_ENABLED: "false"
```

Migraciones Flyway fallan

Ver error específico:

```
docker compose -f docker-compose.dev.yml logs app | grep Flyway
```

Resetear Flyway (⚠ BORRA TODOS LOS DATOS):

```
# Conectar a la DB
docker exec -it lago-postgres-dev psql -U postgres -d lago

# En psql:
DROP SCHEMA public CASCADE;
CREATE SCHEMA public;
\q

# Reiniciar backend (aplica migraciones de nuevo)
docker compose -f docker-compose.dev.yml restart app
```

📖 Documentación Adicional

Documento	Descripción
README-DOCKER.md	Guía completa de docker-compose (dev vs prod)
ARQUITECTURA.md	Decisiones técnicas y arquitectura detallada
SSL-SETUP.md	Configuración de certificados SSL (producción)
PLAN_DESPLIEGUE.md	Guía completa de despliegue en producción
env.prod.example	Plantilla de variables de entorno para producción
POSTMAN_README.md	Colección Postman para testing

📝 Enums y Valores Permitidos

Circuit (Circuitos)

- `A`, `B`, `C`, `D`

VisitorType (Tipo de Visitante)

- `INDIVIDUAL`: Visitante individual o grupo familiar
- `EDUCATIONAL`: Grupo educativo (escuelas, colegios)
- `EVENT`: Evento especial

ReservationStatus (Estado de Reserva)

- `PENDING`: Pendiente de confirmación

- `CONFIRMED`: Confirmada (envía WhatsApp)
- `CANCELLED`: Cancelada (envía WhatsApp)

HowHeard (Cómo se enteró)

- `REDES_SOCIALES`
- `RECOMENDACION`
- `WEB`
- `OTRO`

🚀 Próximos Pasos

1. **Desarrollo Local:** Ya está listo ☑
2. **Testing de Features:** Usa Swagger UI http://localhost:8080/docs
3. **Integración con Frontend:** El frontend debe apuntar a `http://localhost:8080`
4. **Despliegue en Producción:** Ver PLAN_DESPLIEGUE.md (próxima semana)

🆘 Soporte

- **Arquitectura técnica:** ARQUITECTURA.md
- **Docker Compose:** README-DOCKER.md
- **Despliegue:** PLAN_DESPLIEGUE.md

Versión: 1.0.0 **Última actualización:** Diciembre 2024