Backend - Sistema de Reservas Lago Escondido

API REST para gestión de reservas del Lago Escondido.

Stack Tecnológico

Tecnología	Versión	Descripción
Java	21 LTS	Lenguaje principal
Spring Boot	3.5.5	Framework backend
PostgreSQL	16	Base de datos
Flyway	10.8.1	Migraciones de BD
JWT (jjwt)	0.12.3	Autenticación
Twilio	10.1.0	Notificaciones WhatsApp
Apache POI	5.2.5	Exportación Excel
SpringDoc	2.8.11	Documentación OpenAPI
Lombok	1.18.30	Reducción de boilerplate
Testcontainers	-	Tests con PostgreSQL real

Quick Start - Desarrollo Local

Opción 1: Docker (Recomendado)

```
# 1. Clonar y entrar al directorio
cd backend-reservas-lago

# 2. Crear archivo .env con credenciales Twilio (opcional)
cp env.example .env
# Editar .env con tus credenciales

# 3. Levantar servicios
docker compose -f docker-compose.dev.yml up -d

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

# 5. Abrir Swagger UI
# http://localhost:8080/docs
```

Servicios disponibles:

- Backend API: http://localhost:8080
- Swagger UI: http://localhost:8080/docs
- PostgreSQL: localhost:5432

Opción 2: Maven Local (Requiere JDK 21)

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

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

# Linux/macOS:
export SPRING_PROFILES_ACTIVE=dev

# 3. Arrancar backend
./mvnw spring-boot:run   # Linux/macOS
.\mvnw.cmd spring-boot:run   # Windows
```

Comandos Docker Útiles

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

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

# Rebuild del backend
docker compose -f docker-compose.dev.yml up -d --build app

# Rebuild sin cache
docker compose -f docker-compose.dev.yml build --no-cache app

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

# Reset completo (elimina datos de BD)
docker compose -f docker-compose.dev.yml down -v
```

Configuración

Variables de Entorno

Ver env.prod.example para la lista completa de variables de producción.

Variables principales:

Variable	Descripción	Requerida
`SPRING_PROFILES_ACTIVE`	Perfil de Spring (dev/prod)	Sí
`SPRING_DATASOURCE_URL`	URL de conexión PostgreSQL	Sí
`SPRING_DATASOURCE_USERNAME`	Usuario BD	Sí
`SPRING_DATASOURCE_PASSWORD`	Password BD	Sí
`JWT_SECRET`	Clave para firmar tokens	Sí (prod)
`DEFAULT_CAPACITY`	Capacidad diaria por defecto	No (default: 30)
`ALLOWED_ORIGINS`	URLs CORS permitidas	Sí
`WHATSAPP_ENABLED`	Habilitar notificaciones	No (default: false)
`TWILIO_ACCOUNT_SID`	Credencial Twilio	Si WhatsApp habilitado
`TWILIO_AUTH_TOKEN`	Credencial Twilio	Si WhatsApp habilitado

Credenciales

> **IMPORTANTE:** Las credenciales de desarrollo están en `docker-compose.dev.yml`. **NUNCA** usar credenciales de desarrollo en producción. Para producción, copiar `env.prod.example` a `.env.prod` y configurar valores seguros.

Conexión directa a BD (desarrollo):

```
docker exec -it lago-postgres-dev psql -U postgres -d lago
```

Estructura del Proyecto

```
src/main/java/com.luismunozse.reservalago/
├── config/                          # Configuración
│   ├── CorsConfig.java                  # CORS
│   ├── SecurityConfig.java              # Spring Security + JWT
│   ├── JwtAuthenticationFilter.java     # Filtro JWT
│   └── OpenApiConfig.java               # Swagger/OpenAPI
│
├── controller/                      # Endpoints REST
│   ├── PublicController.java            # API pública (reservas, disponibilidad)
│   ├── AuthController.java              # Login/autenticación
│   ├── AdminController.java             # API administrativa
│   ├── UserController.java              # Gestión de usuarios
│   └── ApiExceptionHandler.java         # Manejo global de errores
│
├── service/                         # Lógica de negocio
```

```
│   ├── ReservationService.java         # Gestión de reservas
│   ├── ReservationMapper.java          # Mapeo entidad - DTO
│   ├── ReservationExcelExporter.java   # Exportación a Excel
│   ├── AvailabilityService.java        # Disponibilidad y capacidad
│   ├── WhatsAppService.java            # Notificaciones WhatsApp
│   ├── JwtService.java                 # Generación/validación JWT
│   ├── UserService.java                # CRUD usuarios admin
│   ├── UserDetailsServiceImpl.java     # Spring Security UserDetails
│   └── SystemConfigService.java        # Configuración del sistema
│
├── repo/                           # Repositorios JPA
│   ├── ReservationRepository.java
│   ├── ReservationSpecifications.java  # Criterios dinámicos
│   ├── AvailabilityRuleRepository.java
│   ├── UserRepository.java
│   └── SystemConfigRepository.java
│
├── model/                          # Entidades JPA
│   ├── Reservation.java
│   ├── ReservationVisitor.java         # Acompañantes
│   ├── AvailabilityRule.java
│   ├── User.java
│   ├── SystemConfig.java
│   ├── Circuit.java                    # Enum: A, B, C, D
│   ├── VisitorType.java                # Enum: INDIVIDUAL, EDUCATIONAL, EVENT
│   ├── ReservationStatus.java          # Enum: PENDING, CONFIRMED, CANCELLED
│   └── HowHeard.java                   # Enum: cómo conoció el lugar
│
├── dto/                            # Data Transfer Objects
│   ├── CreateReservationRequest.java
│   ├── ReservationSummaryDTO.java
│   ├── AdminReservationDTO.java
│   ├── VisitorDTO.java
│   ├── AdminVisitorDTO.java
│   ├── CreateEventRequest.java
│   ├── CapacityRequest.java
│   ├── ExportReservationsFilter.java
│   ├── EducationalReservationsRequest.java
│   ├── LoginRequest.java
│   ├── LoginResponse.java
│   ├── CreateUserRequest.java
│   ├── UpdateUserRequest.java
│   └── UserResponse.java
│
└── ReservalagoApplication.java     # Clase principal
```

API Endpoints

Documentación Interactiva

- **Swagger UI:** http://localhost:8080/docs

- **OpenAPI JSON:** http://localhost:8080/v3/api-docs

Importar a Postman:

1. Abrir Postman
2. Import - Link - `http://localhost:8080/v3/api-docs`

Autenticación

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

{
  "email": "<EMAIL>",
  "password": "<PASSWORD>"
}
```

Respuesta:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "email": "usuario@dominio.com",
  "firstName": "Nombre",
  "lastName": "Apellido"
}
```

Usar token en requests protegidos:

```
Authorization: Bearer <TOKEN>
```

Endpoints Públicos (Sin autenticación)

Método	Endpoint	Descripción
GET	/api/availability?date=YYYY-MM-DD	Disponibilidad de un día
GET	/api/availability?month=YYYY-MM	Disponibilidad del mes
POST	/api/reservations	Crear reserva
GET	/api/reservations/{id}	Obtener resumen de reserva

Ejemplo - Consultar disponibilidad:

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

Respuesta:

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

Ejemplo - 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
}
```

Endpoints Administrativos (Requieren JWT)

Header requerido: `Authorization: Bearer <token>`

Método	Endpoint	Descripción
GET	/api/admin/reservations	Listar reservas (paginado)
GET	/api/admin/reservations?status=PENDING	Filtrar por estado
GET	/api/admin/reservations?date=YYYY-MM-DD	Filtrar por fecha
GET	/api/admin/reservations?dni=12345678	Buscar por DNI
GET	/api/admin/reservations?name=Juan	Buscar por nombre
POST	/api/admin/reservations/{id}/confirm	Confirmar reserva (envía WhatsApp)

Método	Endpoint	Descripción
POST	/api/admin/reservations/{id}/cancel	Cancelar reserva (envía WhatsApp)
GET	/api/admin/reservations/export	Exportar a Excel
GET	/api/admin/reservations/export?month=2025-01	Exportar mes específico
PUT	/api/admin/availability/{date}	Configurar capacidad de un día
GET	/api/admin/availability/state?year=2025&month=1	Estado del calendario
PUT	/api/admin/availability/state?year=2025&month=1	Habilitar/deshabilitar mes
POST	/api/admin/eventos	Crear evento especial
GET	/api/admin/config/educational-reservations	Estado reservas educativas
PUT	/api/admin/config/educational-reservations	Toggle reservas educativas

Gestión de Usuarios Admin

Método	Endpoint	Descripción
GET	/api/admin/users	Listar usuarios
POST	/api/admin/users	Crear usuario
GET	/api/admin/users/{email}	Obtener usuario por email
PUT	/api/admin/users/{id}	Actualizar usuario
DELETE	/api/admin/users/{id}	Eliminar usuario

Base de Datos

Migraciones Flyway

Las migraciones se ejecutan automáticamente al iniciar la aplicación.

Versión	Archivo	Descripción
V4	V4__init.sql	Tablas principales (reservations, availability_rules)
V5	V5__unique_reservation_per_day.sql	Constraint único DNI por día
V6	V6__create_users_table.sql	Tabla de usuarios admin
V7	V7__add_user_names.sql	Nombres en usuarios
V8	V8__create_system_config_table.sql	Configuración del sistema
V9	V9__create_reservation_visitors.sql	Tabla de acompañantes

Versión	Archivo	Descripción
V10	`V10__drop_allergies_column.sql`	Limpieza columnas obsoletas
V11	`V11__partial_unique_index_exclude_cancelled.sql`	Índice parcial (excluye canceladas)
V12	`V12__add_phone_to_users.sql`	Teléfono en usuarios
V13	`V13__add_phone_to_reservation_visitors.sql`	Teléfono en acompañantes

Consultar historial de migraciones:

```
SELECT * FROM flyway_schema_history;
```

Enums y Valores Permitidos

Circuit (Circuitos de visita)

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

VisitorType (Tipo de visitante)

- `INDIVIDUAL` - Visitante individual o grupo familiar
- `EDUCATIONAL_INSTITUTION` - Institución educativa
- `EVENT` - Evento especial

ReservationStatus (Estado de reserva)

- `PENDING` - Pendiente de confirmación
- `CONFIRMED` - Confirmada (envía WhatsApp)
- `CANCELLED` - Cancelada (envía WhatsApp)

HowHeard (Cómo conoció el lugar)

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

Notificaciones WhatsApp

El sistema envía notificaciones automáticas al confirmar/cancelar reservas usando Twilio.

Configuración

1. Crear cuenta en Twilio Console
2. Habilitar WhatsApp Sandbox

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

```
TWILIO_ACCOUNT_SID=<tu_account_sid>
TWILIO_AUTH_TOKEN=<tu_auth_token>
```

4. Reiniciar el backend

Deshabilitar

En `docker-compose.dev.yml`:

```
WHATSAPP_ENABLED: "false"
```

Testing

Ejecutar tests

```
# Windows
.\mvnw.cmd test

# Linux/macOS
./mvnw test
```

Los tests usan **Testcontainers** para levantar PostgreSQL real automáticamente.

Probar con cURL

```
# Consultar disponibilidad (público)
curl http://localhost:8080/api/availability?date=2025-01-15

# Login (reemplazar credenciales)
curl -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"<EMAIL>","password":"<PASSWORD>"}'

# Listar reservas (con token)
curl http://localhost:8080/api/admin/reservations \
  -H "Authorization: Bearer <TOKEN>"
```

Troubleshooting

El backend no inicia

```
# Verificar contenedores
docker compose -f docker-compose.dev.yml ps

# Ver logs del backend
docker compose -f docker-compose.dev.yml logs app

# Ver logs de PostgreSQL
docker compose -f docker-compose.dev.yml logs db
```

Error de conexión a BD

```
# Reset completo
docker compose -f docker-compose.dev.yml down -v
docker compose -f docker-compose.dev.yml up -d
```

Puerto 8080 ocupado

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

# Linux/macOS
lsof -i :8080

# Cambiar puerto en docker-compose.dev.yml:
ports:
  - "8081:8080"
```

Error CORS

Verificar que `ALLOWED_ORIGINS` en `docker-compose.dev.yml` incluya la URL exacta del frontend (protocolo + puerto).

Migraciones Flyway fallan

```
# Ver error específico
docker compose -f docker-compose.dev.yml logs app | grep -i flyway

# Reset de schema (BORRA TODOS LOS DATOS)
docker exec -it lago-postgres-dev psql -U postgres -d lago -c "DROP SCHEMA public
CASCADE; CREATE SCHEMA public;"

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

WhatsApp no envía mensajes

1. Verificar archivo `.env` existe con credenciales Twilio
2. Verificar `WHATSAPP_ENABLED: "true"` en docker-compose
3. Ver logs: `docker compose -f docker-compose.dev.yml logs -f app | grep -i whatsapp`

Seguridad

Buenas prácticas

1. **NUNCA** subir archivos `.env` con credenciales reales al repositorio
2. Usar passwords seguros (mínimo 16 caracteres)
3. `JWT_SECRET` debe ser aleatorio y único por entorno
4. Rotar credenciales periódicamente
5. Usar HTTPS en producción

Generar valores seguros

```
# JWT_SECRET (mínimo 64 caracteres)
openssl rand -base64 64

# Database password
openssl rand -base64 32

# Admin password
openssl rand -base64 24
```

Documentación Adicional

Documento	Descripción
docker-compose.dev.yml	Configuración Docker desarrollo
docker-compose.prod.yml	Configuración Docker producción
env.prod.example	Plantilla variables de entorno
POSTMAN_README.md	Colección Postman para testing

Autor

Desarrollado por **Luis Muñoz**

Versión: 1.0.0 **Última actualización:** Diciembre 2025