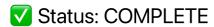


# 👋 Start Here - Spring Boot Backend Setup

### What You Asked For

You wanted a Kotlin Spring Boot backend that:

- V Handles database operations via JPA
- Uses SQLite with Flyway migrations
- 🔽 Is launched by Rust in development mode via Gradle
- Runs as embedded JAR with bundled JRE in production
- Supports both macOS and Windows (with platform annotations)



Everything has been set up and documented. The project is ready to use!

### Where to Start

Option 1: Quick Start (5 minutes)

```
cd backend-spring
./gradlew bootRun
```

Then read: 4\_QUICK\_START.md

Option 2: Complete Understanding (30 minutes)

Read in this order:

- 1. 2\_OVERVIEW.md High-level overview
- 2. 3\_COMPLETE\_SETUP\_GUIDE.md Complete setup guide
- 3. 5\_FLYWAY\_WORKFLOW.md Database migrations
- 4. 6\_RUST\_INTEGRATION.md Rust integration

## Documentation Structure



### **Solution** Your Three Questions Answered

Q1: How do JPA entity changes create SQL files for Flyway migration?

Answer: They don't automatically. You manually create SQL migration files.

### Workflow:

- 1. Change JPA entity (e.g., add a field)
- 2. Write SQL migration manually (recommended) OR use Hibernate temporarily to see what SQL would be generated
- 3. Create V{version}\_\_Description.sql in backendspring/src/main/resources/db/migration/
- 4. Restart app Flyway auto-applies the migration

Detailed guide: 5\_FLYWAY\_WORKFLOW.md

Q2: How to apply SQL file to actual schema?

**Answer**: Automatically when the Spring Boot app starts.

#### Process:

- 1. Put SQL file in src/main/resources/db/migration/
- 2. Name it correctly: V1\_\_Initial.sql, V2\_\_Add\_field.sql, etc.
- 3. Start/restart Spring Boot: ./gradlew bootRun
- 4. Flyway detects new migration and applies it
- 5. Check logs: "Successfully applied 1 migration"

**Configuration**: Already set in application.yml - Flyway is enabled and configured.

Q3: How to launch Spring Boot from Rust in dev and embed JRE in production?

#### Answer:

**Development** (Rust launches Gradle):

```
Command::new("./gradlew")
        arg("bootRun")
        current_dir("./backend-spring")
        spawn()?;
```

Production (Rust launches embedded JAR):

Complete code examples: 6\_RUST\_INTEGRATION.md

# 🚀 Quick Test

Verify everything works:

```
# 1. Build the backend
cd backend-spring
./gradlew build

# 2. Run the backend
./gradlew bootRun

# 3. In another terminal, test the API
curl http://localhost:8080/api/folders
curl http://localhost:8080/api/app-state

# 4. Success! You should see JSON responses
```

### What's Been Created

### **Project Structure**

- Complete Spring Boot project in backend-spring/
- JPA entities for all 4 database tables
- REST API controllers with full CRUD operations
- Flyway migration for initial schema
- · Gradle build configuration

Documentation (6 files in docs/ folder)

- 1. 1\_START\_HERE.md This file
- 2. 2\_OVERVIEW.md Overview + summary

- 3. **3\_COMPLETE\_SETUP\_GUIDE.md** Complete setup guide (31 sections)
- 4. 4\_QUICK\_START.md Quick reference
- 5. 5\_FLYWAY\_WORKFLOW.md Migration workflow with examples
- 6. 6\_RUST\_INTEGRATION.md Full Rust integration code

## Learning Path

### Today (30 minutes)

- 1. Read this file (you're doing it!)
- 2. Read 2\_OVERVIEW.md
- 3. Run / gradlew bootRun and test API
- 4. Skim 3\_COMPLETE\_SETUP\_GUIDE.md

### This Week

- 1. T Study 5\_FLYWAY\_WORKFLOW.md
- 2. The Practice: Add a field to an entity + create migration
- 3. Tstudy 6\_RUST\_INTEGRATION.md
- 4. Implement HTTP client in Rust

### Next Week

- 1. Z Auto-launch Spring Boot from Rust
- 2. Z Convert Prisma operations to HTTP calls
- 3. Test end-to-end integration
- 4. Z Prepare for production (download JRE, build JAR)

## Checklist

### Setup (Do Now)

- Read 2 OVERVIEW.md
- Run \_/gradlew bootRun successfully
- Test API endpoints with curl
- Explore code in IDE (IntelliJ IDEA or VS Code)

### Understanding (This Week)

- Read 3\_COMPLETE\_SETUP\_GUIDE.md sections 1-6
- Read 5\_FLYWAY\_WORKFLOW.md
- Practice creating a migration
- Understand the entity → SQL workflow

### Integration (Next Week)

- Read 6\_RUST\_INTEGRATION.md
- Add request and tokio to Cargo.toml

- Create HTTP client module in Rust
- Test launching Spring Boot from Rust

### Production (When Ready)

- Read 3\_COMPLETE\_SETUP\_GUIDE.md section 8
- Download JRE 17 for macOS ARM64
- Build production JAR: \_/gradlew bootJar
- Test embedded deployment
- (Optional) Add Windows support



1. JPA Entities ≠ Schema Updates

```
hibernate:
ddl-auto: validate # ← Only validates, NEVER auto-creates/updates
```

All schema changes must go through Flyway migrations.

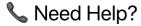
2. Flyway Migrations Are Sequential

```
V1__Initial_schema.sql ← Initial setup
V2__Add_description.sql ← Add field
V3__Create_user_table.sql ← Add table
V4__Add_index.sql ← Add index
```

Version numbers must be sequential. Never skip or modify existing migrations.

3. Development vs Production

Mode	How Spring Boot Runs
Development	./gradlew bootRun (launched by Rust or manually)
Production	Embedded JAR with bundled JRE (no Java install needed)



Problem: Port 8080 already in use

```
lsof -ti:8080 | xargs <mark>kill</mark> -9
```

Problem: Build failed

cd backend-spring
./gradlew clean build --no-daemon

Problem: Don't understand Flyway workflow

• Read: 5\_FLYWAY\_WORKFLOW.md

• It has detailed examples and step-by-step instructions

Problem: Don't know how to integrate with Rust

• Read: 6\_RUST\_INTEGRATION.md

• It has complete working code examples

# You're Ready!

Everything is set up. Your next step:

Read 2\_OVERVIEW.md

Then:

Good luck! 🚀

Created: October 30, 2025

Status: ✓ Complete and tested

**Build Status**: **✓** ./gradlew build succeeded