

Flutter + Spring Boot

A Complete Learning Module

SECTION 1

Overview

Flutter — Frontend (Client-side)

Flutter is Google's open-source UI framework for building natively compiled apps from a single codebase. It runs on mobile, web, and desktop.

Dart language • Cross-platform • Widget-based UI • By Google

Spring Boot — Backend (Server-side)

Spring Boot is a Java framework for building production-ready REST APIs and backend services. It simplifies configuration and deployment of server applications.

Java / Kotlin • REST API • Database access • By VMware/Spring

How They Relate

Flutter and Spring Boot are a full-stack pair. Flutter renders the UI and handles user interaction. Spring Boot manages business logic, data, and security on the server. They communicate over HTTP using REST APIs or WebSockets.

SECTION 2

Architecture

FLUTTER — UI LAYER

- Screens & Widgets
- State management (Riverpod, Bloc)
- HTTP client (http / Dio)
- Local storage (Hive, SharedPreferences)

SPRING BOOT — BACKEND LAYER

- REST Controllers (@RestController)
- Service Layer (business logic)
- Repository Layer (JPA / Hibernate)
- Security (Spring Security / JWT)

↕ HTTP / REST / WebSocket ↕

DATABASE LAYER

- PostgreSQL / MySQL / MongoDB
- Managed by Spring Data JPA
- Invisible to Flutter — accessed only by backend

SECTION 3

Communication

Step 1

Flutter makes a request

Using the http or dio package, Flutter sends an HTTP request (GET, POST, PUT, DELETE) to a Spring Boot URL endpoint.

Step 2

Spring Boot receives it

A @RestController method mapped to that URL receives the request and routes it to the service layer.

Step 3

Backend processes & responds

Spring Boot queries the database, runs business logic, and returns a JSON response with a status code.

Step 4

Flutter parses the JSON

Flutter decodes the JSON into Dart model objects and updates the UI state to reflect the new data.

WebSocket (Real-time)

For chat apps, live dashboards, or notifications. Spring Boot provides a WebSocket endpoint; Flutter uses the web_socket_channel package.

JWT Authentication

Flutter sends login credentials → Spring Boot validates and returns a JWT token → Flutter stores it and sends it in every subsequent request header.

SECTION 4

Code Examples

Spring Boot — REST Controller (Java)

```
@RestController @RequestMapping("/api/products") public class ProductController {
    @Autowired private ProductService service; @GetMapping public List getAll() { return
    service.findAll(); } @PostMapping public Product create(@RequestBody Product p) {
    return service.save(p); } }
```

Flutter — Calling the API (Dart)

```
import 'package:http/http.dart' as http; Future<List> fetchProducts() async { final res =
await http.get( Uri.parse('http://localhost:8080/api/products'), headers:
{'Authorization': 'Bearer $token'}, ); if (res.statusCode == 200) { final data =
jsonDecode(res.body) as List; return data.map((e) => Product.fromJson(e)).toList(); }
throw Exception('Failed to load'); }
```

SECTION 5

The Restaurant Analogy

Flutter = The Dining Room

The part customers see and interact with — tables, menus, waiters. It looks great and is designed for user experience, but it doesn't cook anything itself.

Spring Boot = The Kitchen

Hidden from the customer, but where all the real work happens. Chefs (business logic) take orders, fetch ingredients from the pantry (database), and prepare the meal.

REST API = The Waiter

The communication channel between the two. Flutter places an order (HTTP request), the waiter carries it to the kitchen (Spring Boot), and brings back the finished dish (JSON response).

Key Insight

The customer never enters the kitchen — Flutter never directly touches your database. The kitchen doesn't care who the customer is — Spring Boot can serve Flutter, a web browser, or any third-party app.

SECTION 6

Use Cases

Enterprise Mobile Apps

Companies already using Java/Spring in their backend can add a Flutter front-end without switching languages or rewriting the server.

Academic & Campus Systems

Student portals, evaluation platforms, scheduling apps — Spring Boot handles complex business rules while Flutter delivers a smooth cross-platform experience.

Real-time Collaborative Tools

Using WebSocket support in both Flutter and Spring Boot, you can build chat, live collaboration, or notification systems.

Alternatives to consider

If Java feels heavy for your team, consider pairing Flutter with:

Node.js + Express

Go + Gin

Python + FastAPI

Flutter's HTTP communication pattern is the same regardless of the backend language.