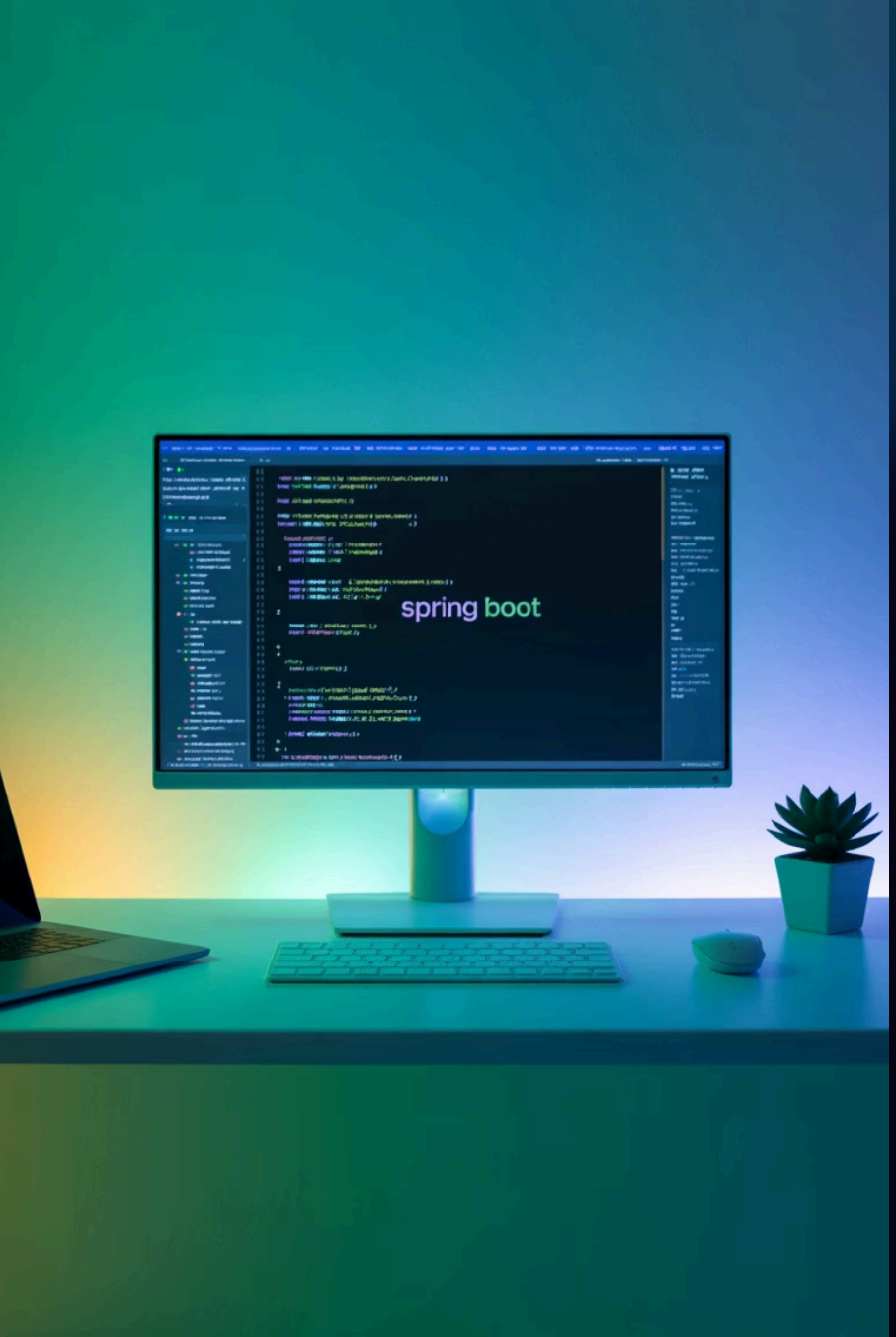
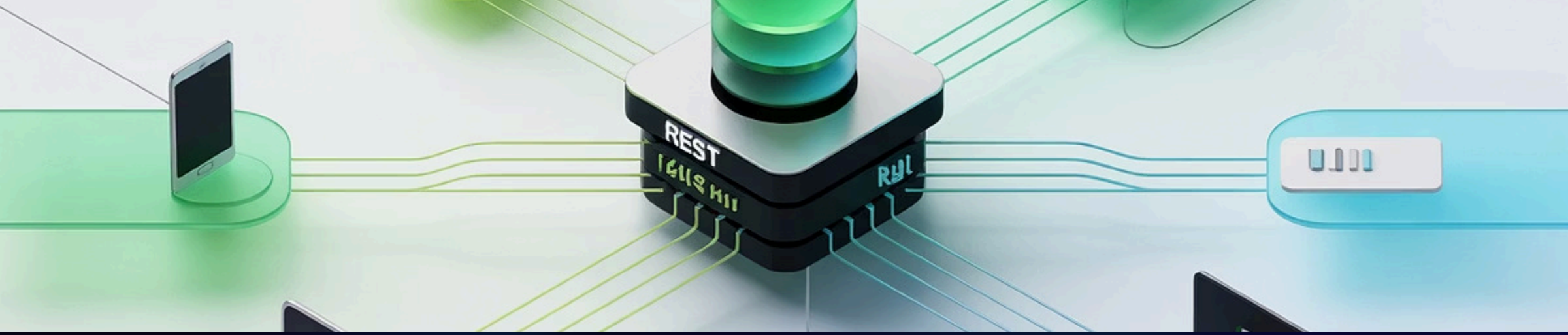


Building Modern APIs with Spring Boot

Spring Boot powers 76% of Java microservices. It streamlines REST API development with minimal configuration.

We'll explore how to build, secure, and test production-ready APIs using industry best practices.

S by Saratha Natarajan



Core Concepts of REST APIs

Resources

Identified by unique URIs

HTTP Verbs

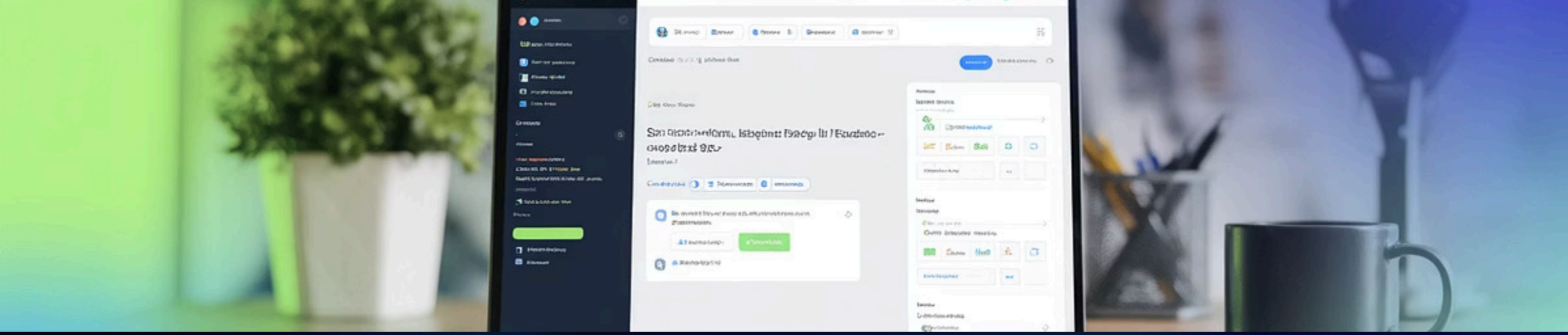
GET, POST, PUT, DELETE

Stateless

No client context stored

Responses

Typically JSON or XML



Setting Up a Spring Boot Project

Generate Project

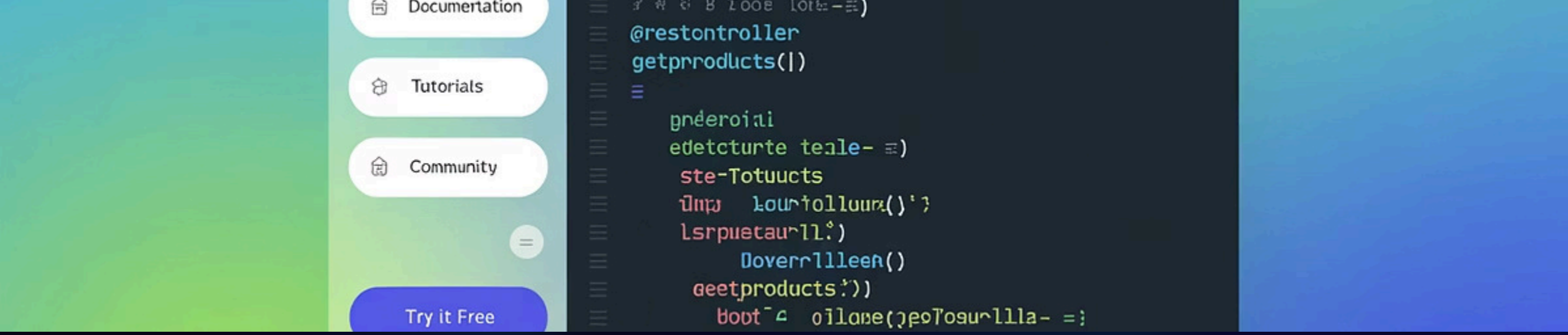
Visit start.spring.io to create your project skeleton.

Add Dependencies

Include Spring Web, Data JPA, and H2 database.

Configure Application

Set up application.properties for your environment.



Creating REST Controllers

Controller Annotation

Mark classes with @RestController to handle HTTP requests.

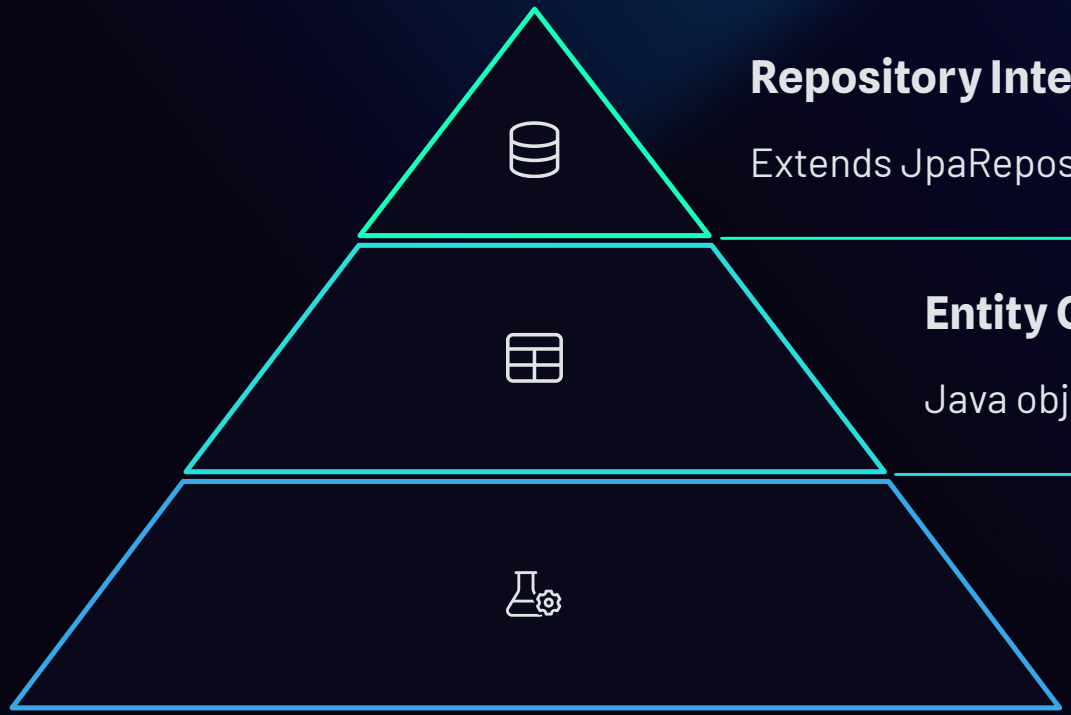
Request Mapping

Use @GetMapping, @PostMapping to route requests to methods.

Response Control

Return ResponseEntity to manage status codes and headers.

Data Persistence with Spring Data JPA



Repository Interface

Extends JpaRepository for CRUD operations

Entity Classes

Java objects mapped to database tables

Configuration

DataSource properties in application.yml

Exception Handling and Validation

Input Validation

Use @Valid with entity constraints

Error Responses

Structured JSON with meaningful codes

Exception Capture

Custom exceptions for business logic

Global Handler

@ControllerAdvice processes all errors

Securing Your REST API

98%

Attack Prevention

Spring Security blocks common exploits

2FA

Authentication

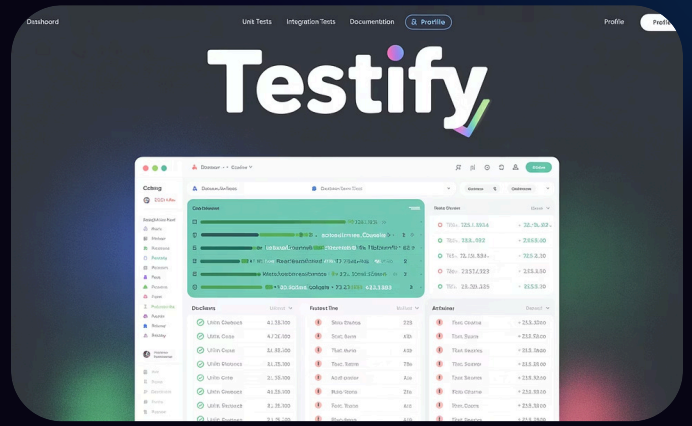
Multiple factors for stronger security

JWT

Authorization

Token-based access control

Testing and Best Practices

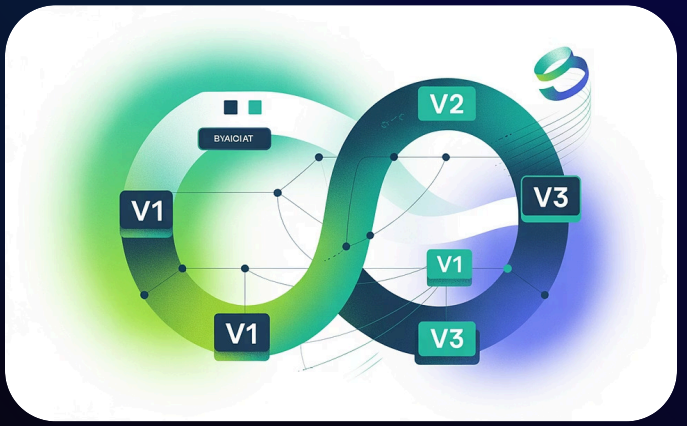


Unit & Integration Tests

Use JUnit, MockMvc and TestRestTemplate for comprehensive coverage.

API Documentation

Generate interactive docs with Swagger/OpenAPI annotations.



API Versioning

Implement v1/v2 paths or content negotiation for evolution.