Spring Boot CRUD Application: A Deep Dive

This presentation explores a Spring Boot CRUD app using PostgreSQL and JPA. The application demonstrates Create, Read, Update, Delete operations with REST APIs. Code is hosted on GitHub for full access.

GK **by girish kudachi**

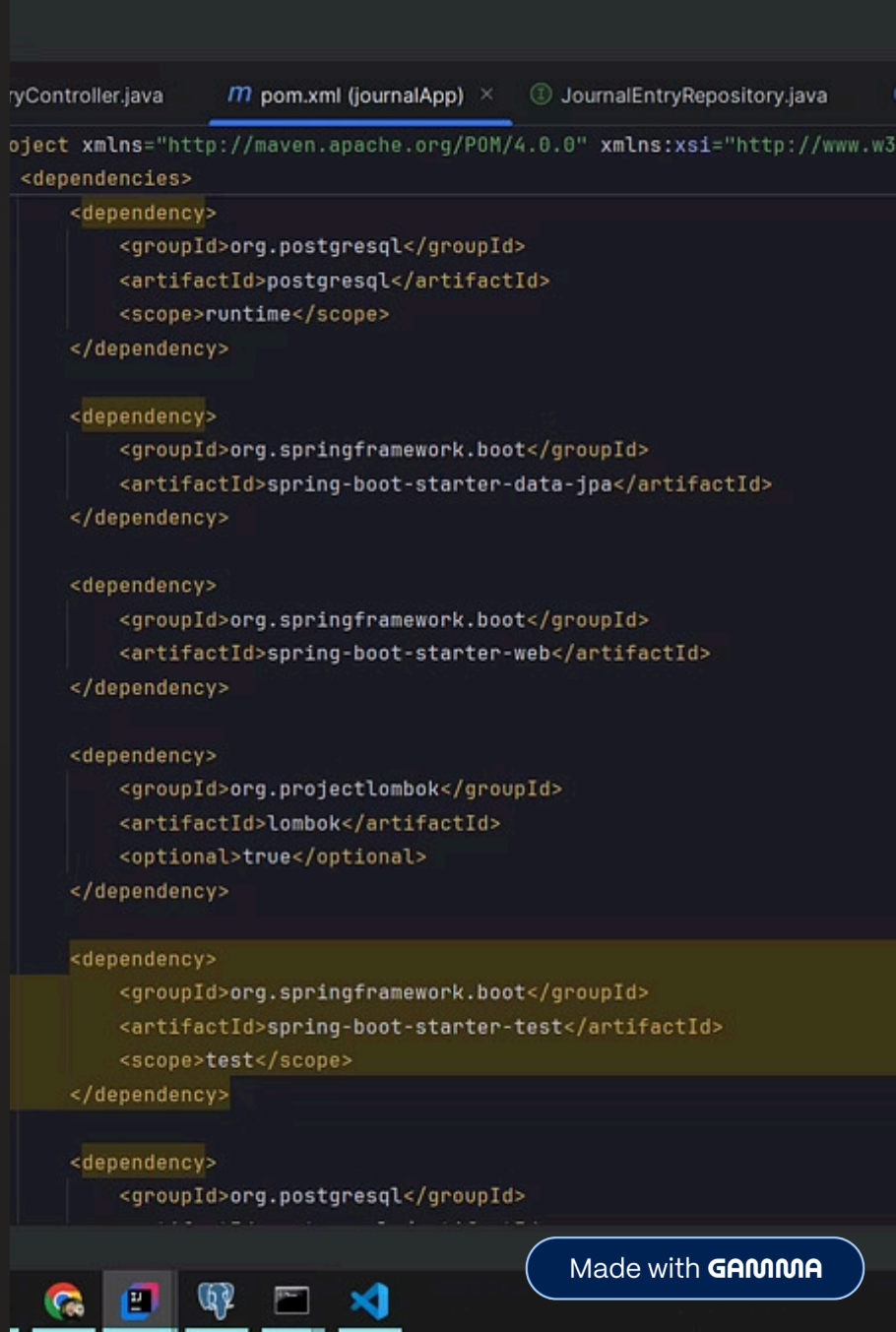
Project Setup and Dependencies

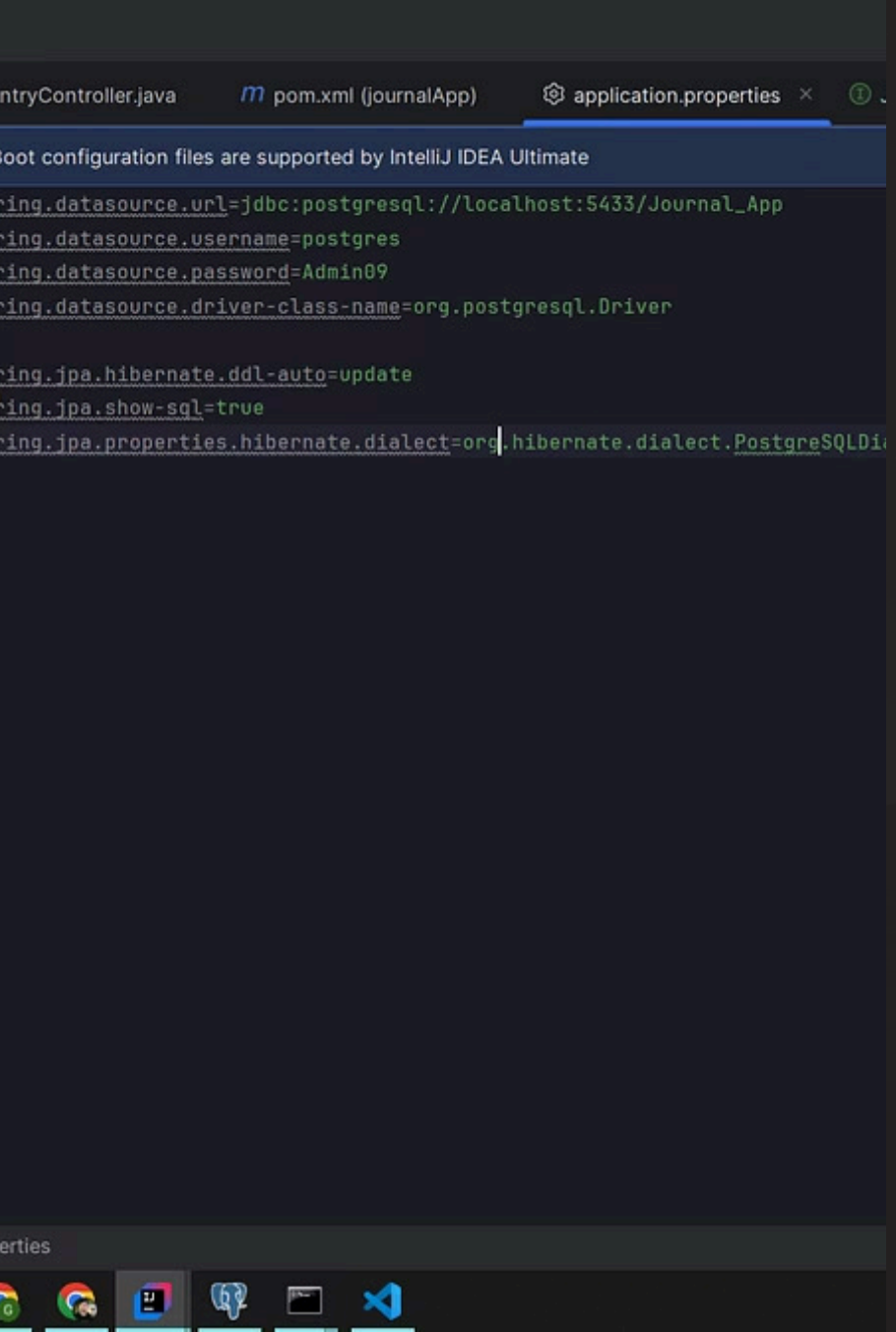
Development Environment

Using IntelliJ IDEA for efficient Java and Spring Boot coding.

Core Dependencies

- spring-boot-starter-data-jpa for JPA integration
- postgresql driver for database connection
- lombok for automatic code generation
- spring-boot-starter-web for REST API functions
- spring-boot-devtools for hot reload





Database Configuration with PostgreSQL

Database Setup

Configured PostgreSQL locally and optionally cloud-based instances.

Key Configuration

Datasource URL, username, password set in properties file.

Schema Management

Used spring.jpa.hibernate.ddl-auto for automatic schema updates.

Entity Definition with JPA

Entity Setup

Defined entities with @Entity and mapped to database tables.

Example: User entity involves id, name, email fields.

Annotations

- @Id and @GeneratedValue for primary keys
- @Column for column mappings
- Lombok @Data and constructors for boilerplate code

Repository Layer with Spring Data JPA

Repository Interface

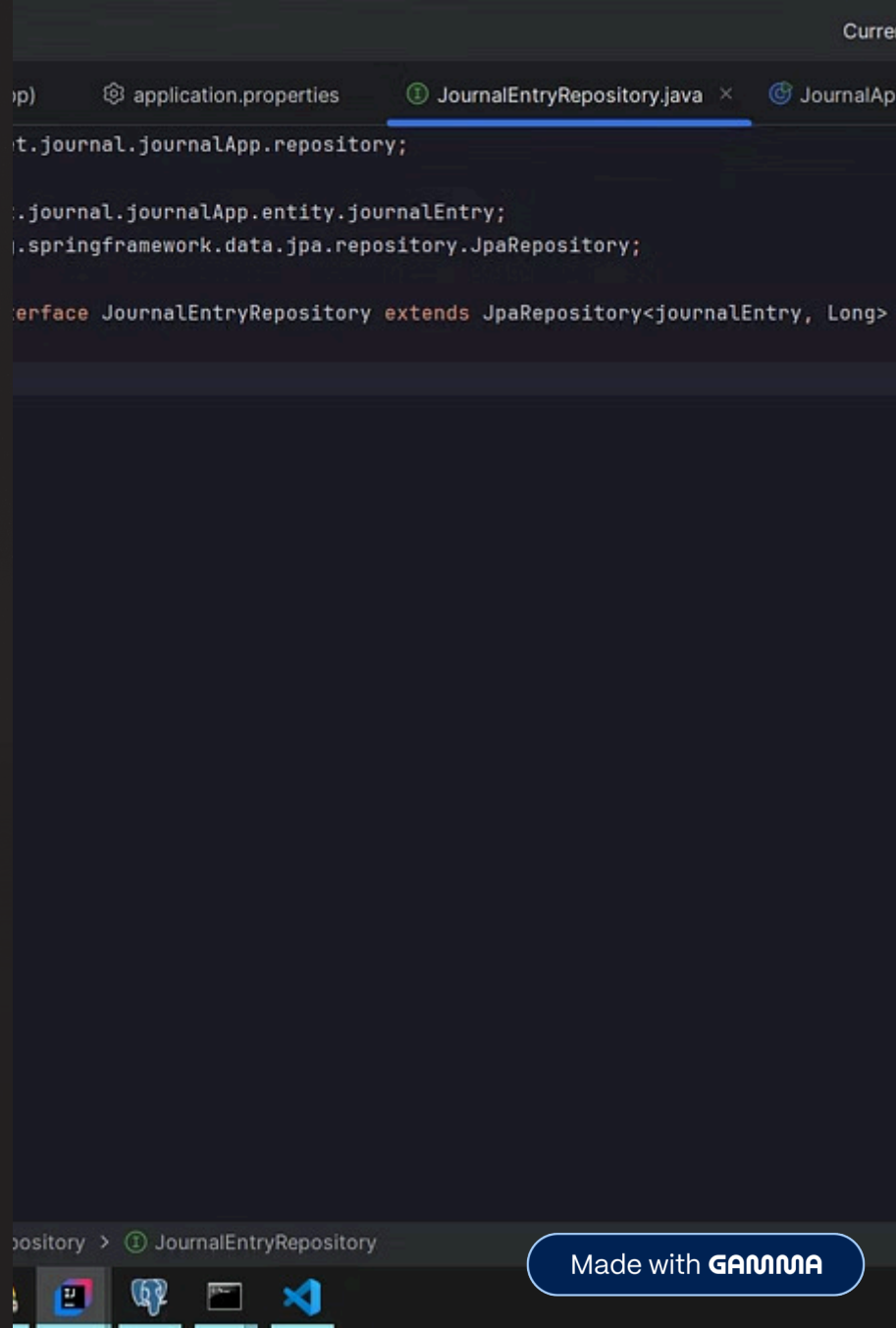
Extended JpaRepository to enable CRUD without extra code.

Built-in Methods

- save(), findById(), findAll(), deleteById()

Custom Queries

Derived query methods by method naming like findByEmail().



Service Layer Implementation

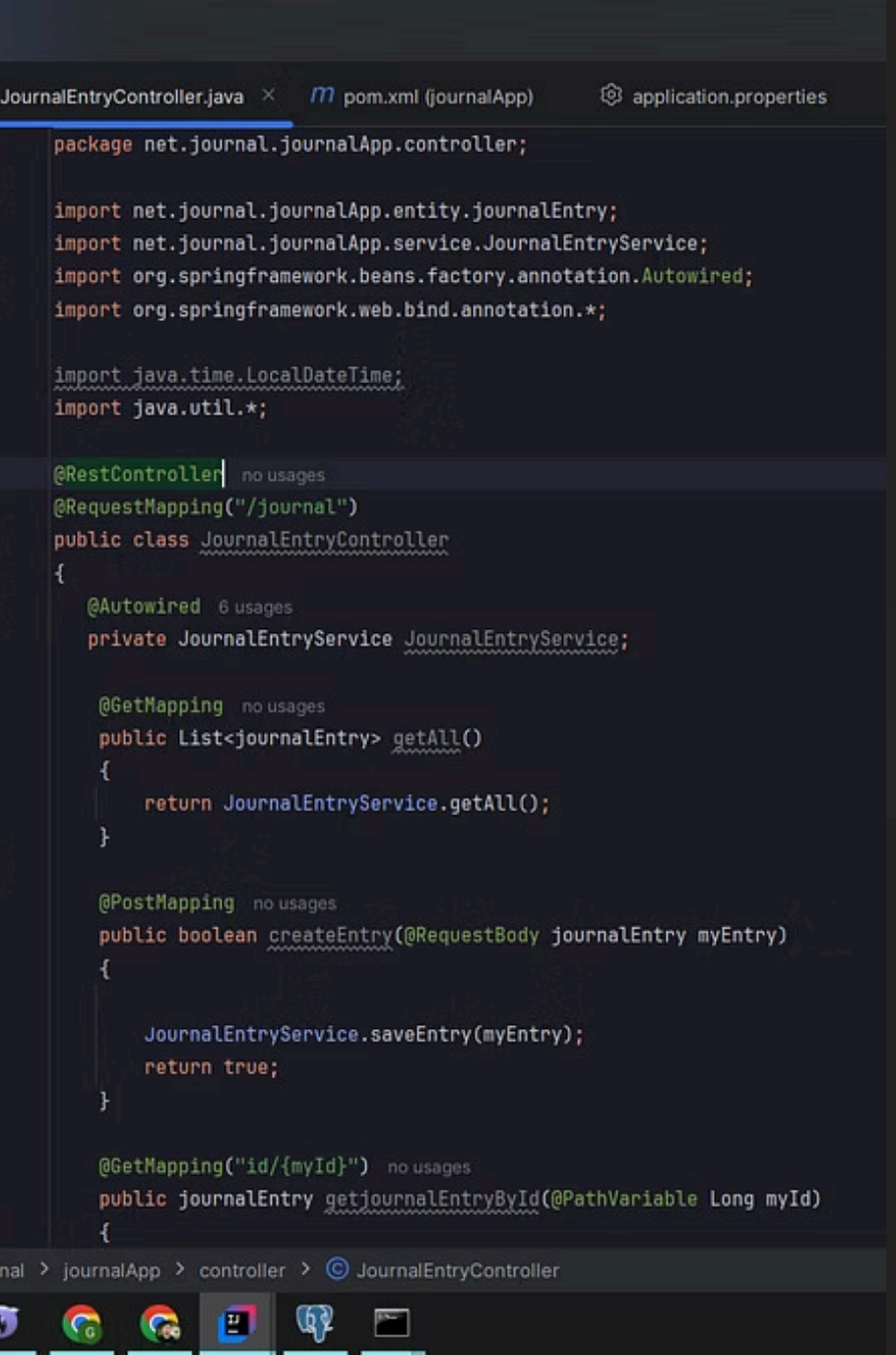
Business Logic

Services handle data manipulation and business rules.

Example methods: createUser, getUserById, updateUser, deleteUser.

Transaction Management

@Transactional ensures data integrity during operations.



REST API Controller Design

1

Controller Annotations

@RestController and @RequestMapping organize API endpoints.

2

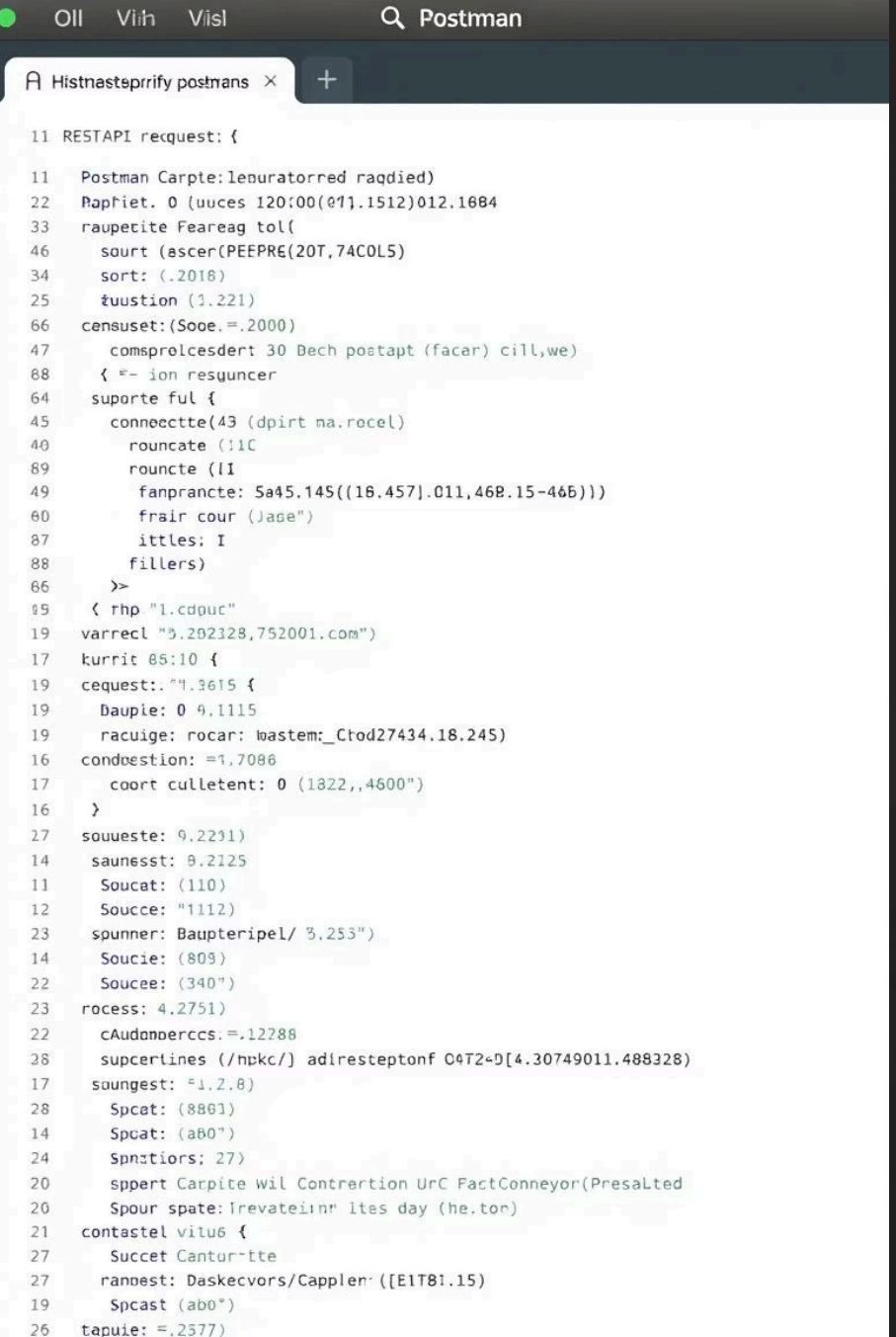
HTTP Methods

- @PostMapping for create
- @GetMapping for read
- @PutMapping for update
- @DeleteMapping for delete

3

Parameter Binding

@PathVariable and @RequestBody simplify data handling.



Testing the APIs with Postman

1 Create User

Sent POST request with JSON payload to /api/users.

2 Retrieve User

Tested GET endpoint using user ID parameter.

3 Update and Delete

Confirmed PUT and DELETE operations via API calls.

Code Structure and Best Practices

Project Layout

- src/main/java structured by layers
- resources hold config files

Standards

Consistent naming and error handling improve maintainability.

Conclusion and Future Enhancements

Summary

CRUD app leverages Spring Boot with PostgreSQL efficiently.

Next Steps

- Add pagination and sorting support
- Implement request validation
- Enable authentication and authorization
- Deploy to cloud platforms like AWS or Heroku

Discussion

Open floor for questions and future collaboration ideas.