**ЗВІТ**

**до лабораторної роботи № 7**  
на тему:  
**"Створення REST-сервісу з використанням анотації @RepositoryRestResource."**

**Виконала:**  
студентка групи **МІТ-21**  
**Йовхимищ Діана**

**Мета роботи:** Ознайомитися з використанням @RepositoryRestResource у Spring Boot для автоматичного створення REST API. Реалізувати CRUD-операції для взаємодії з базою даних PostgreSQL, використовуючи Spring Boot JPA та Lombok.

**Завдання:**

1. В git репозиторії створити гілку lab7.
2. Провести подальше удосконалення front-end частини, створеної у лаб 6.
3. Розробити REST сервіс з використанням анотації @RepositoryRestResource:
   1. забезпечити виконання CRUD операцій для збереження інформації в базі даних Postgres;
   2. для роботи з базою даних використати Spring Boot JPA;
   3. для створення Entity класу використати Lombok.
4. Забезпечити сумісну роботу розроблених у п.п. 2, 3 частин системи.
5. Результати викласти у репозиторій GitHub у гілці lab7.
6. Підготуватися до захисту проекту.

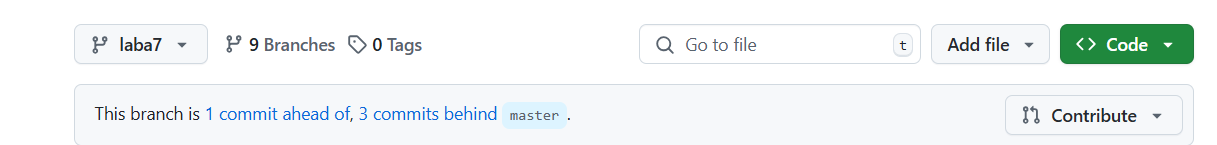
бази даних

**Хід роботи:**

**Крок 1. Створення нової гілки у GitHub**

Перед початком роботи необхідно створити нову гілку lab7 у вашому репозиторії:

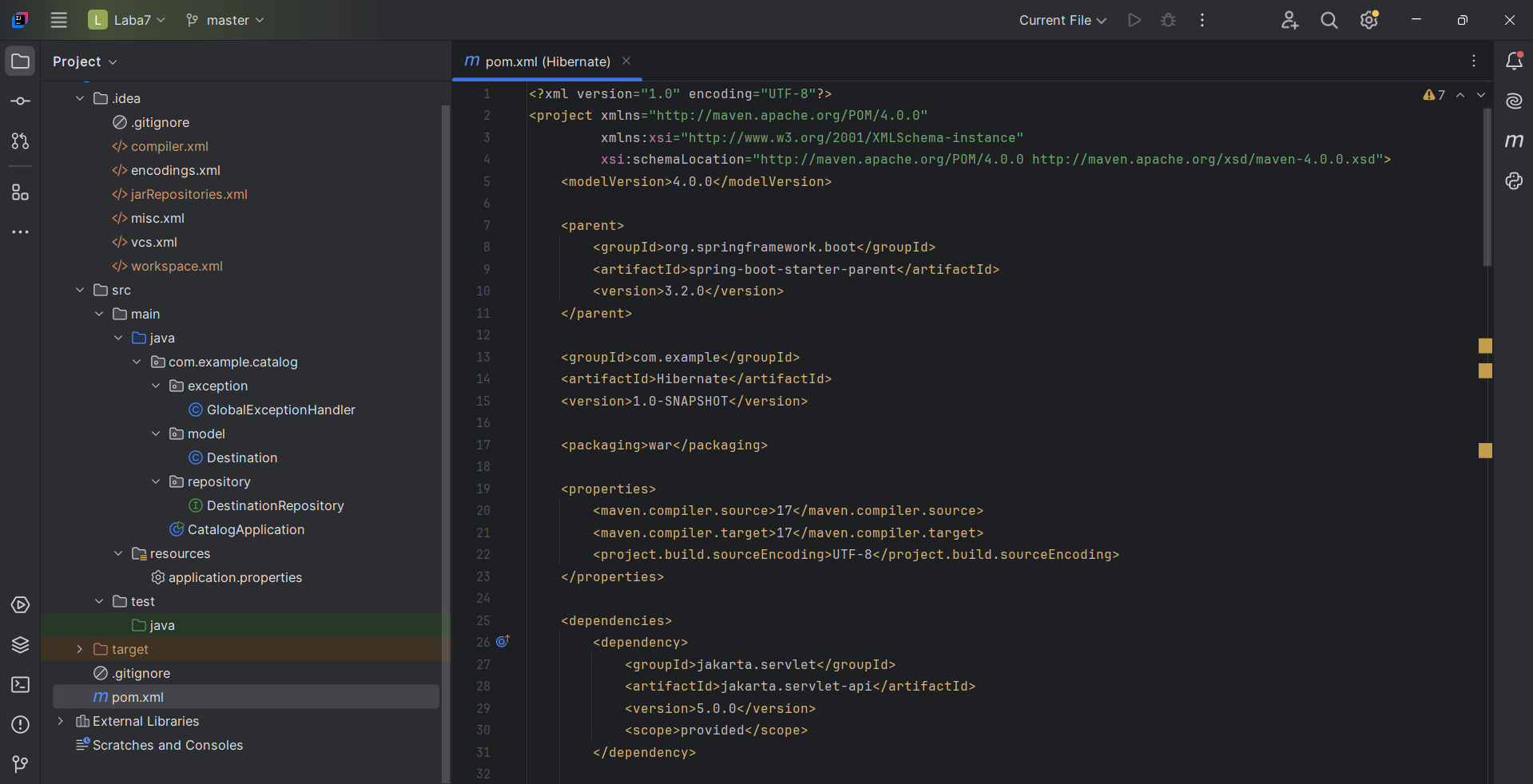
|  |
| --- |
| git checkout -b lab7 git push origin lab7 |

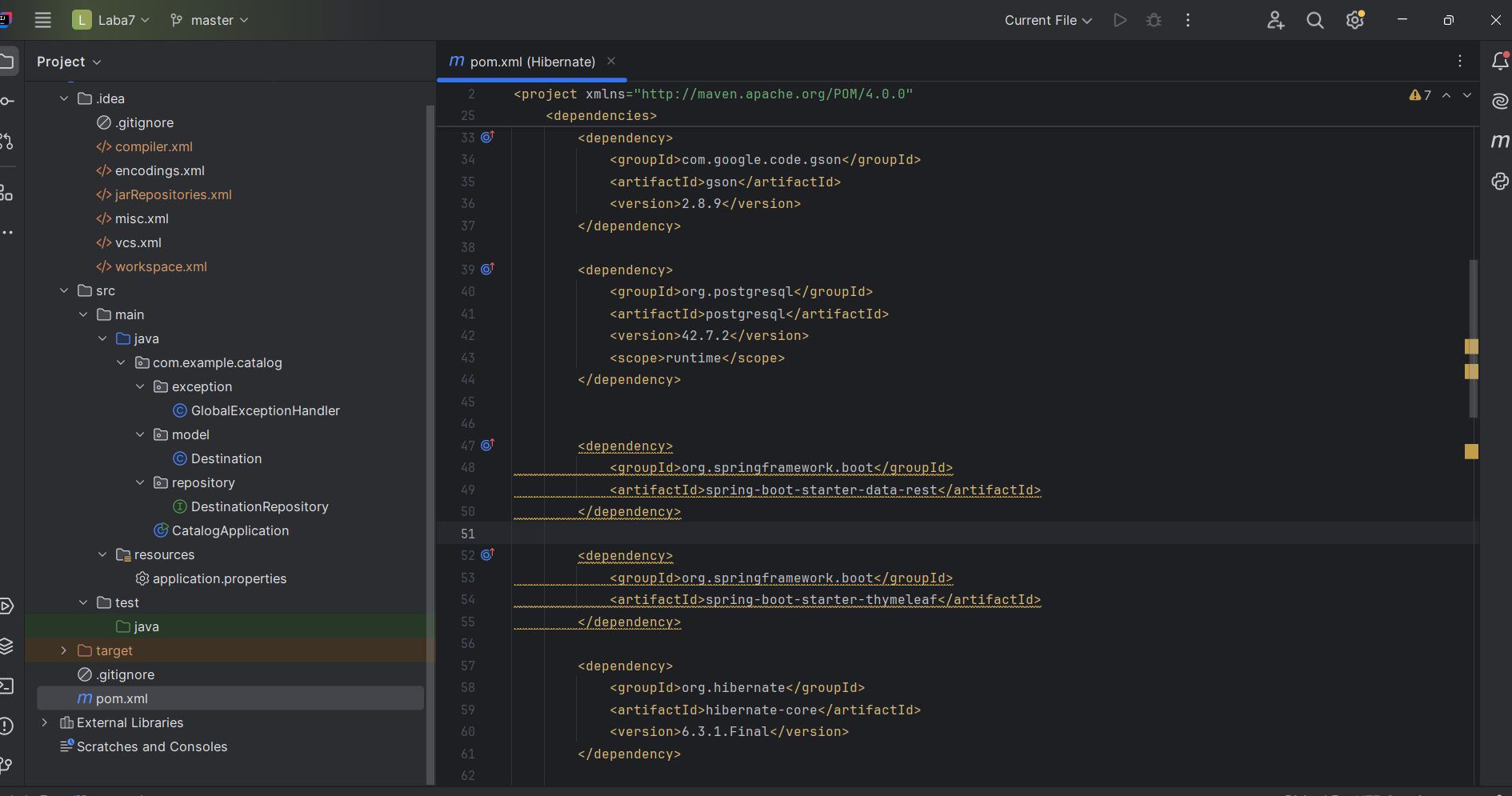


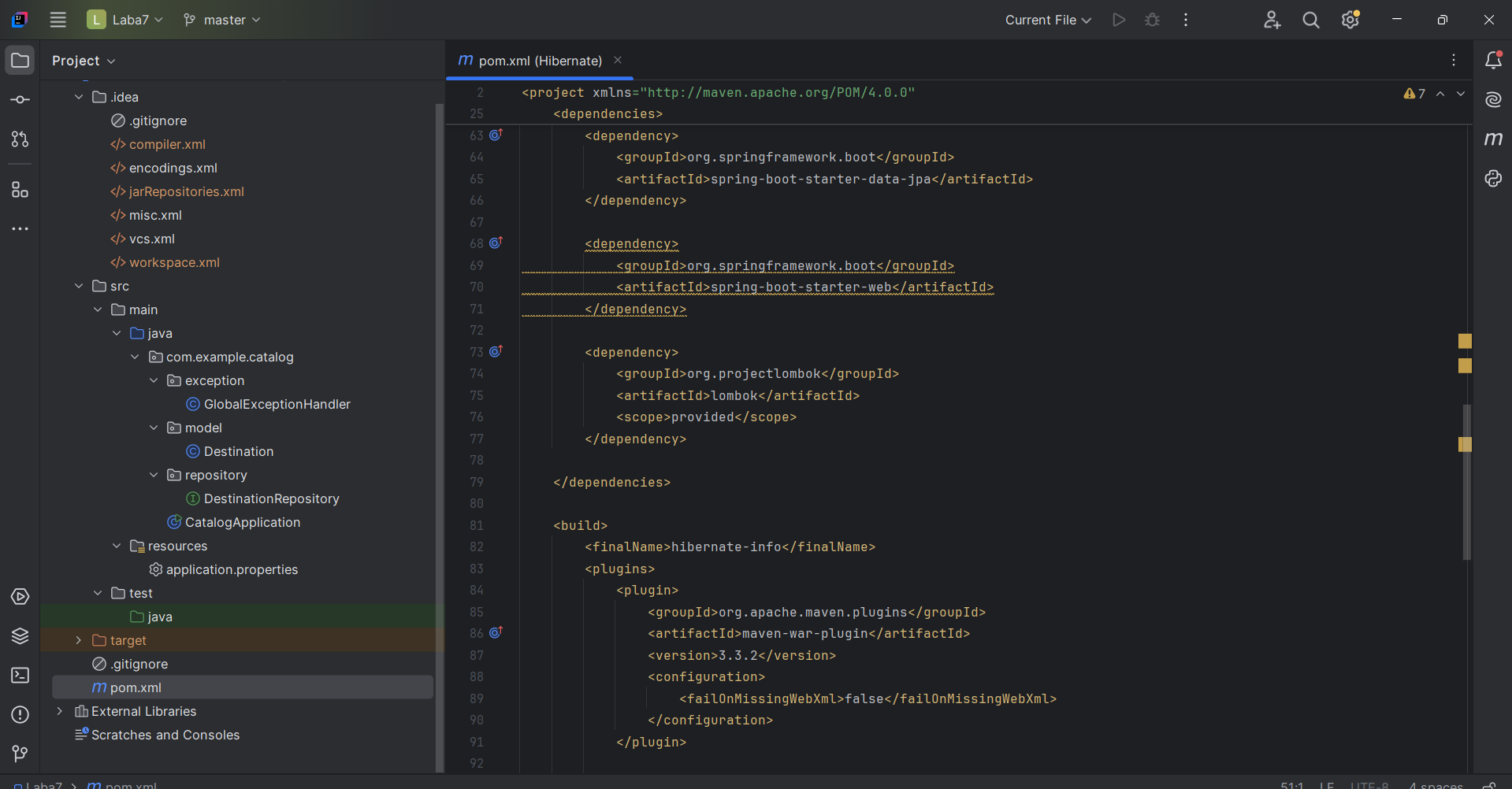
**Крок 2. Налаштування проєкту**

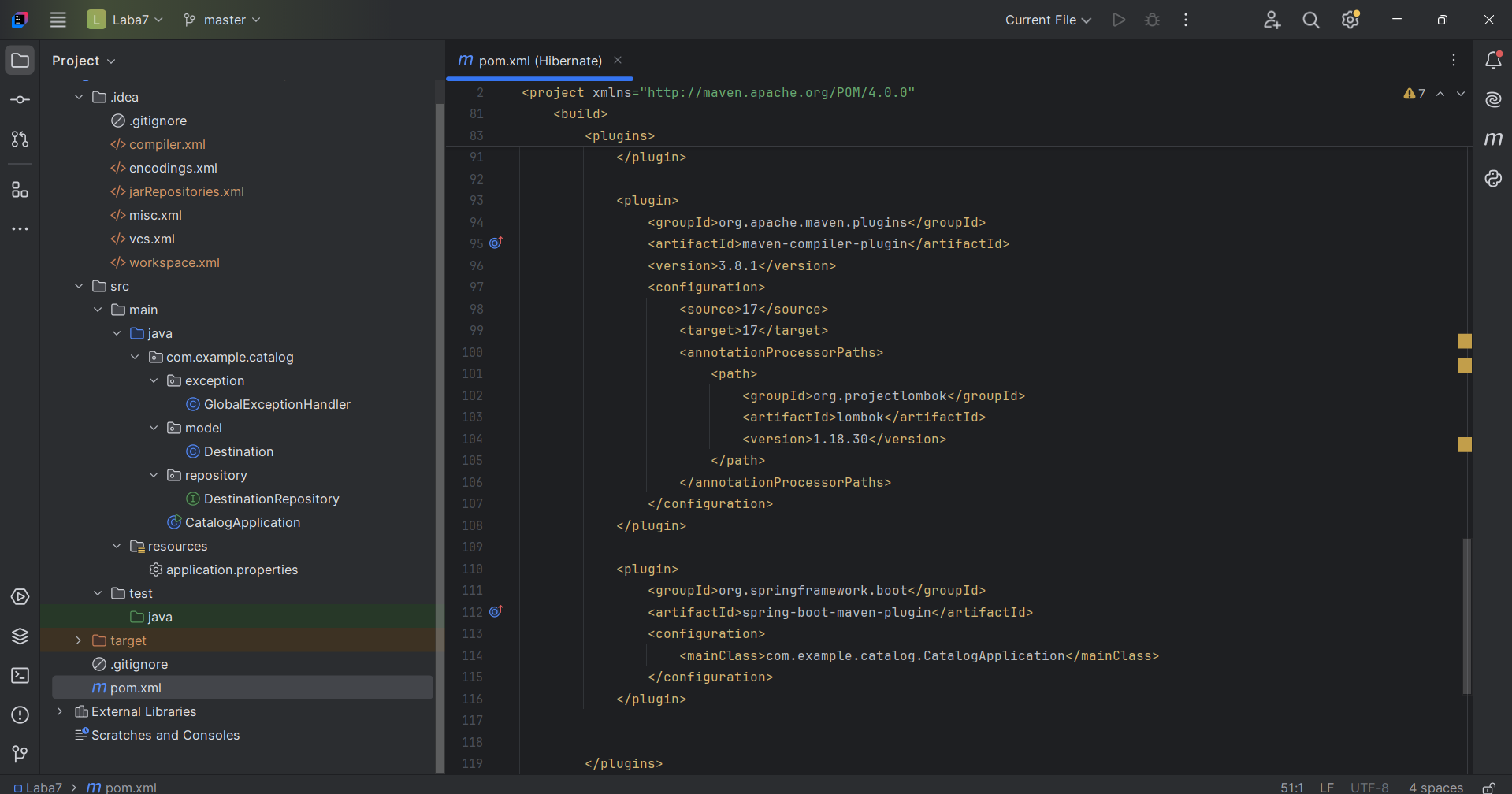
1. **Оновлення pom.xml** – додати залежності:

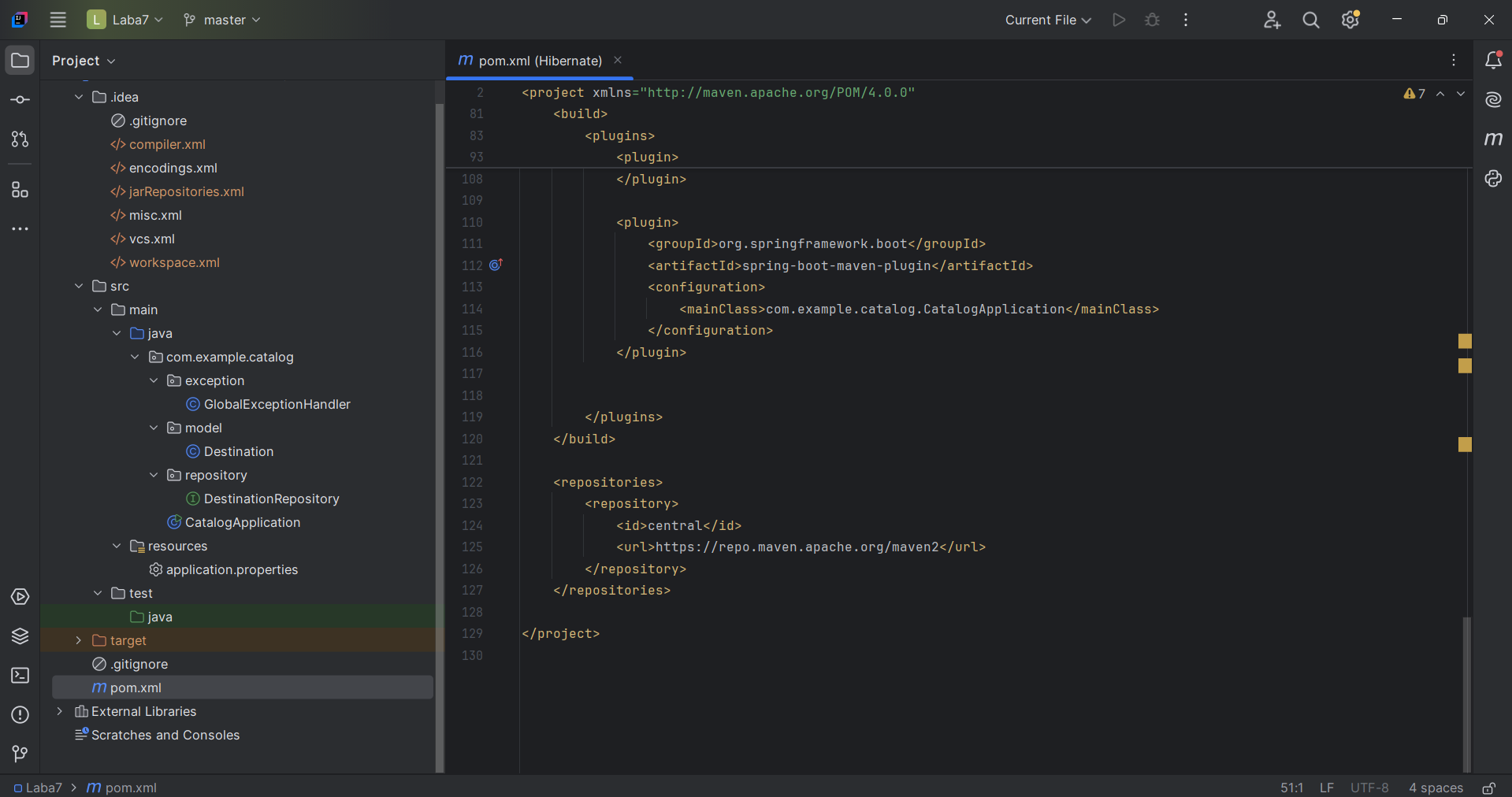
|  |
| --- |
| <dependency>     <groupId>org.springframework.boot</groupId>     <artifactId>spring-boot-starter-data-rest</artifactId> </dependency> <dependency>     <groupId>org.springframework.boot</groupId>     <artifactId>spring-boot-starter-data-jpa</artifactId> </dependency> <dependency>     <groupId>org.springframework.boot</groupId>     <artifactId>spring-boot-starter-web</artifactId> </dependency> <dependency>     <groupId>org.postgresql</groupId>     <artifactId>postgresql</artifactId>     <scope>runtime</scope> </dependency> <dependency>     <groupId>org.projectlombok</groupId>     <artifactId>lombok</artifactId>     <scope>provided</scope> </dependency> |





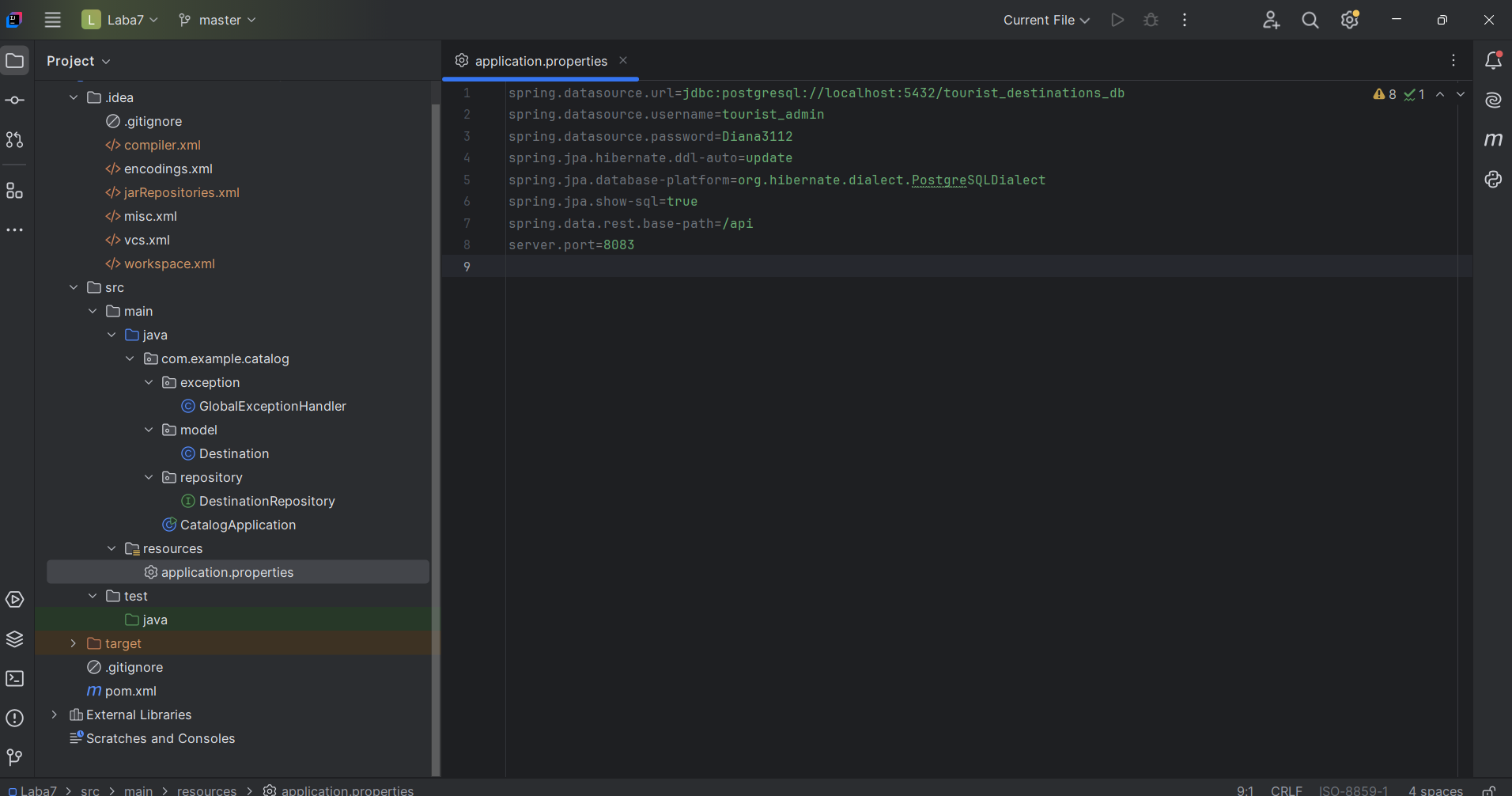






1. **Конфігурація підключення до бази даних у application.properties**:

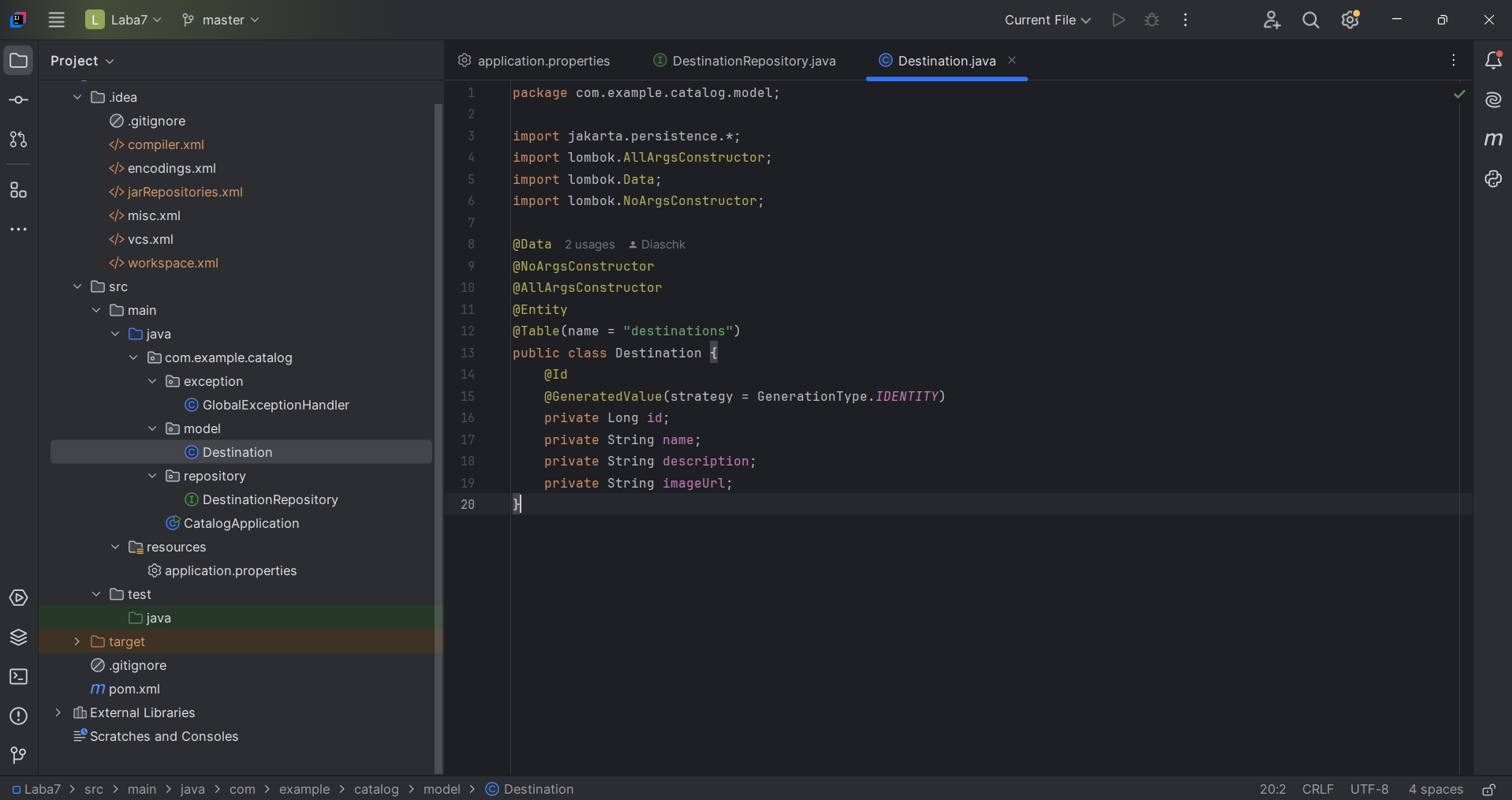
|  |
| --- |
| spring.datasource.url=jdbc:postgresql://localhost:5432/courses\_db spring.datasource.username=courses\_user spring.datasource.password=password spring.jpa.show-sql=true spring.jpa.hibernate.ddl-auto=update spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect spring.data.rest.base-path=/api |



**Крок 3. Створення моделі даних**

Файл Course.java:

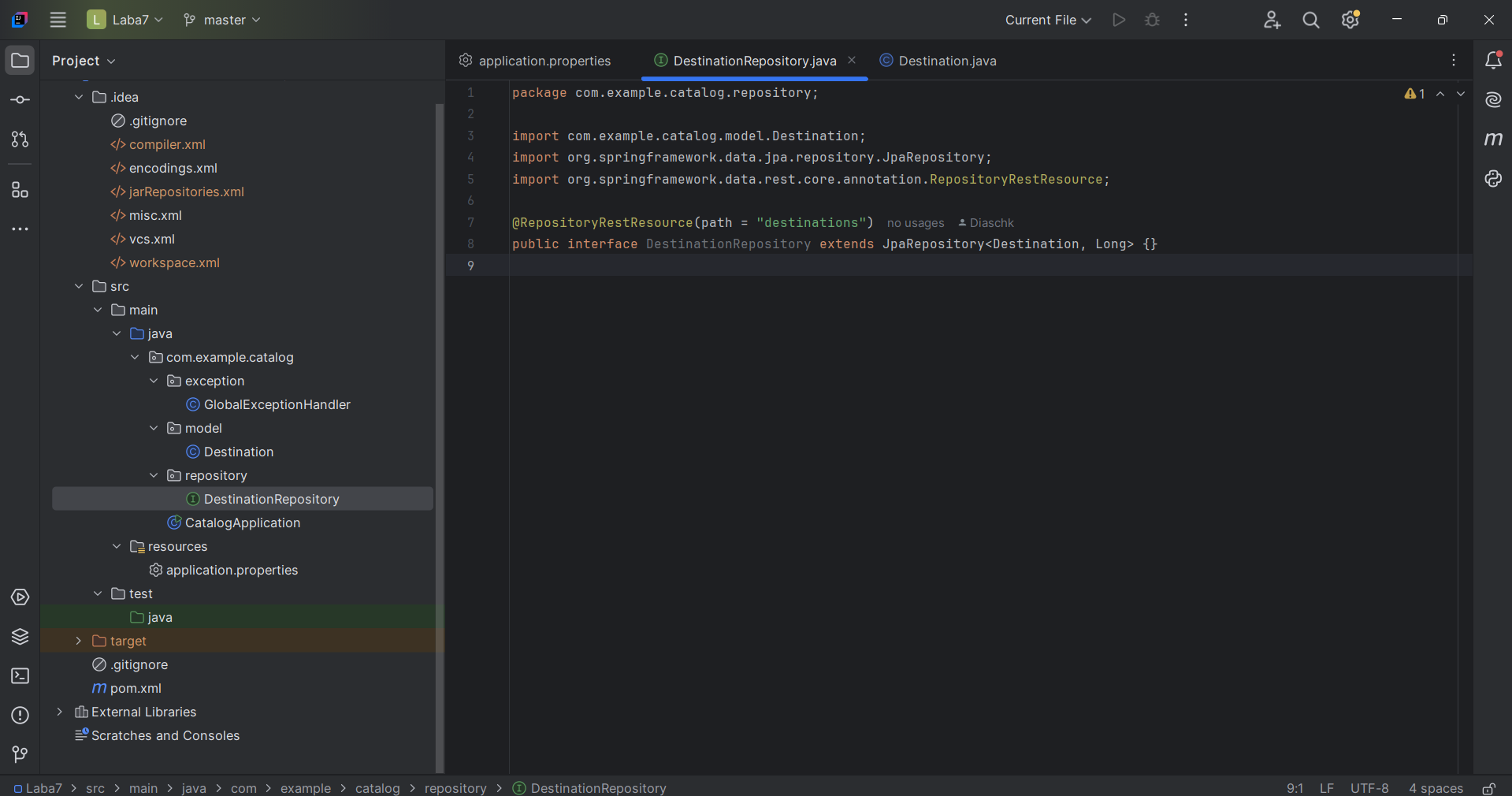
|  |
| --- |
| import jakarta.persistence.\*; import lombok.\*;  @Entity @Table(name = "courses") @Data @NoArgsConstructor @AllArgsConstructor public class Course {     @Id     @GeneratedValue(strategy = GenerationType.IDENTITY)     private Long id;     private String title;     private String description;     private String imageUrl;     private String instructor;     private int duration; } |



**Крок 4. Створення репозиторію**

Файл CourseRepository.java:

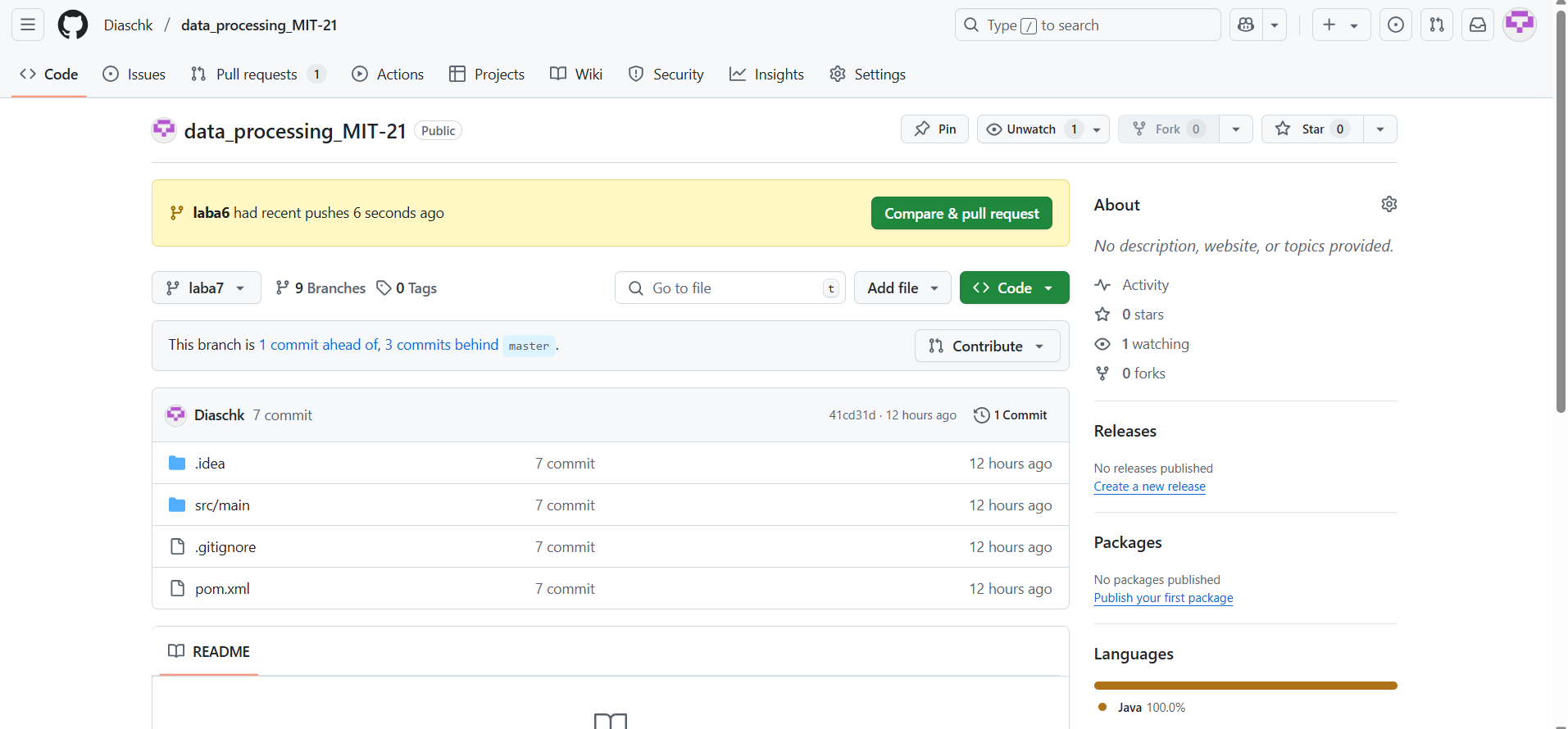
|  |
| --- |
| import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.data.rest.core.annotation.RepositoryRestResource;  @RepositoryRestResource(path = "courses") public interface CourseRepository extends JpaRepository<Course, Long> {} |



Тепер Spring Boot автоматично створить REST API за адресою http://localhost:8083/api/destinations.

**Крок 5. Завантаження у GitHub**

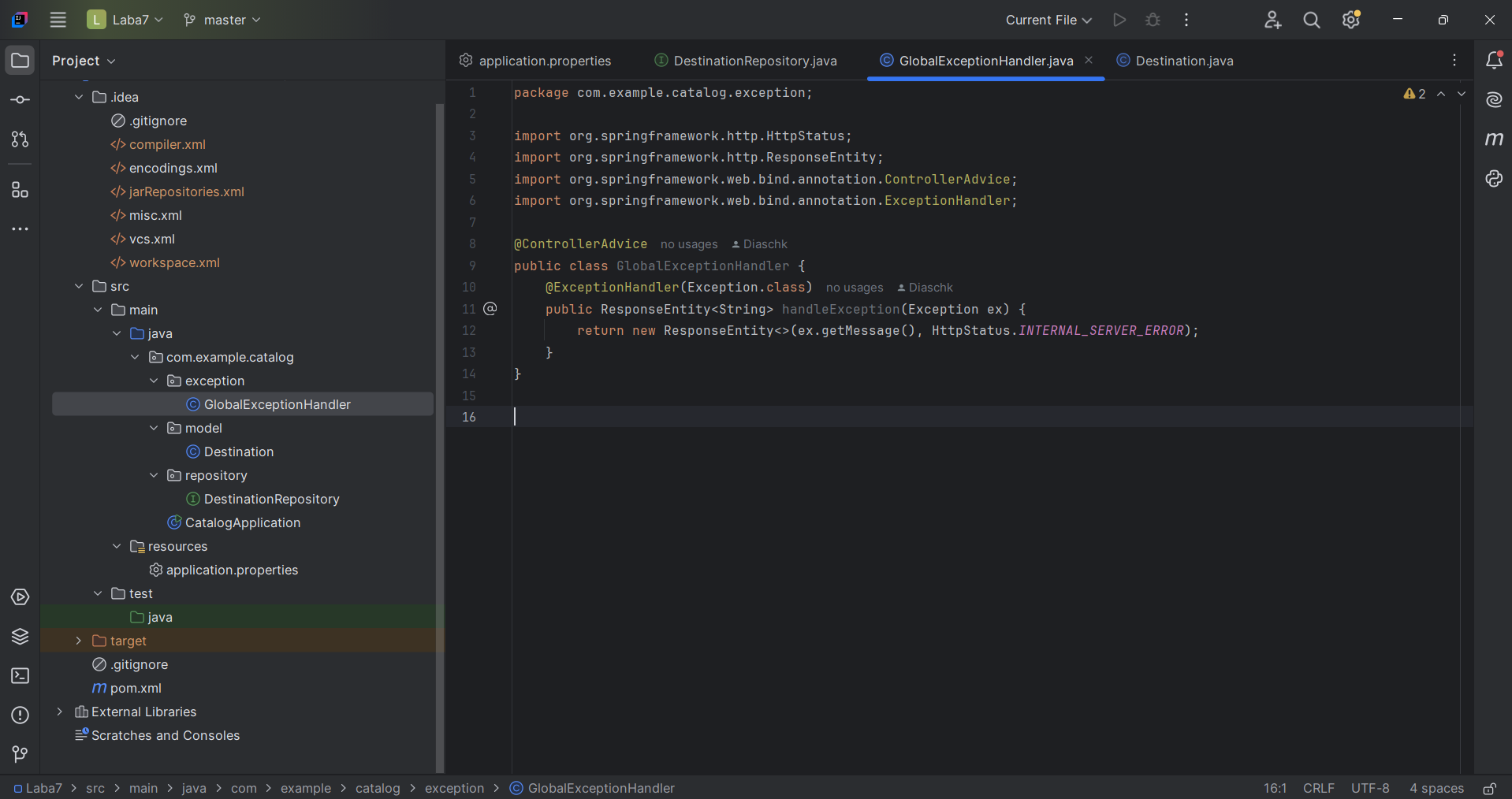
|  |
| --- |
| git add . git commit -m "Lab 7: Implement REST API with @RepositoryRestResource" git push origin lab7 |



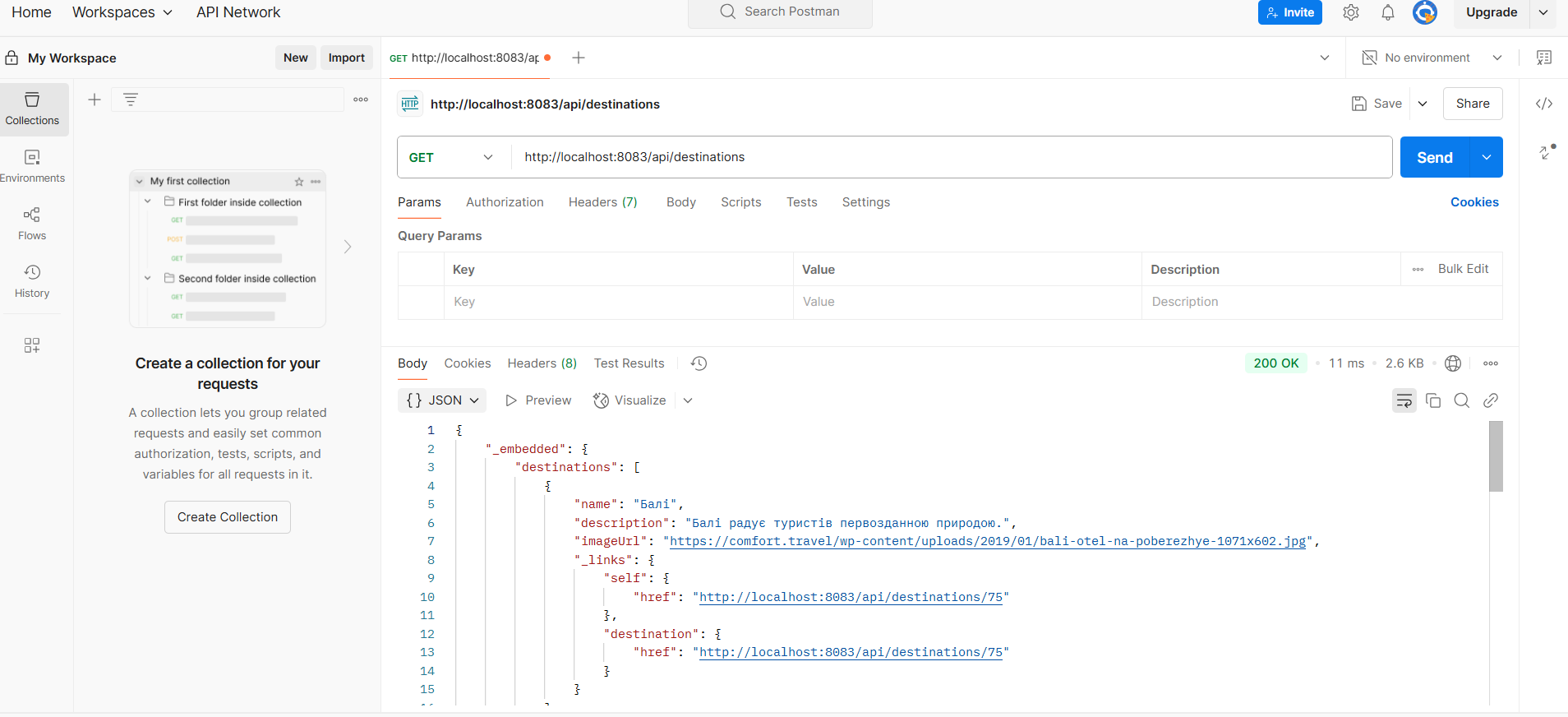
**Обробка помилок у Spring Boot**

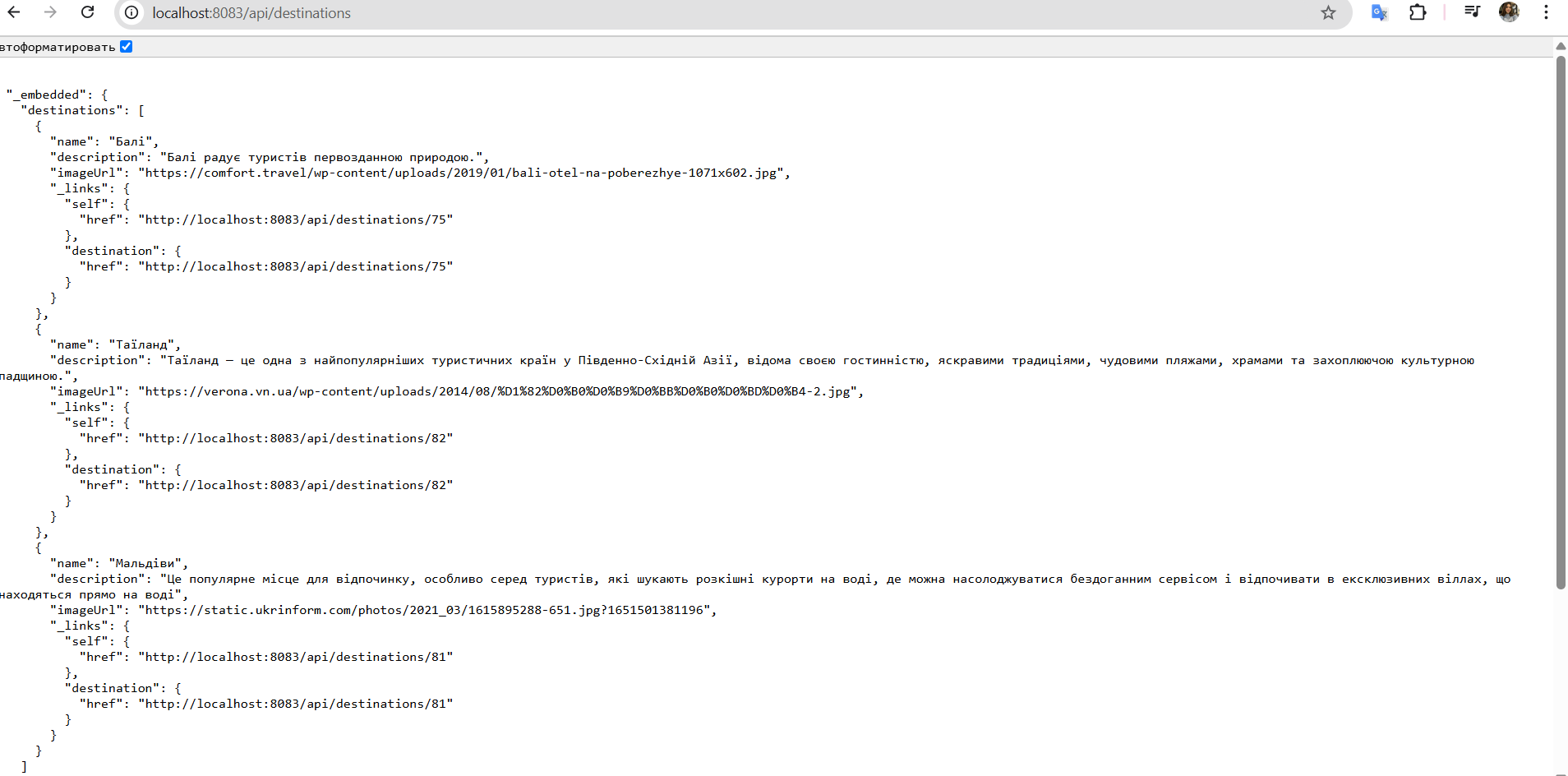
Щоб коректно обробляти помилки, слід створити GlobalExceptionHandler:

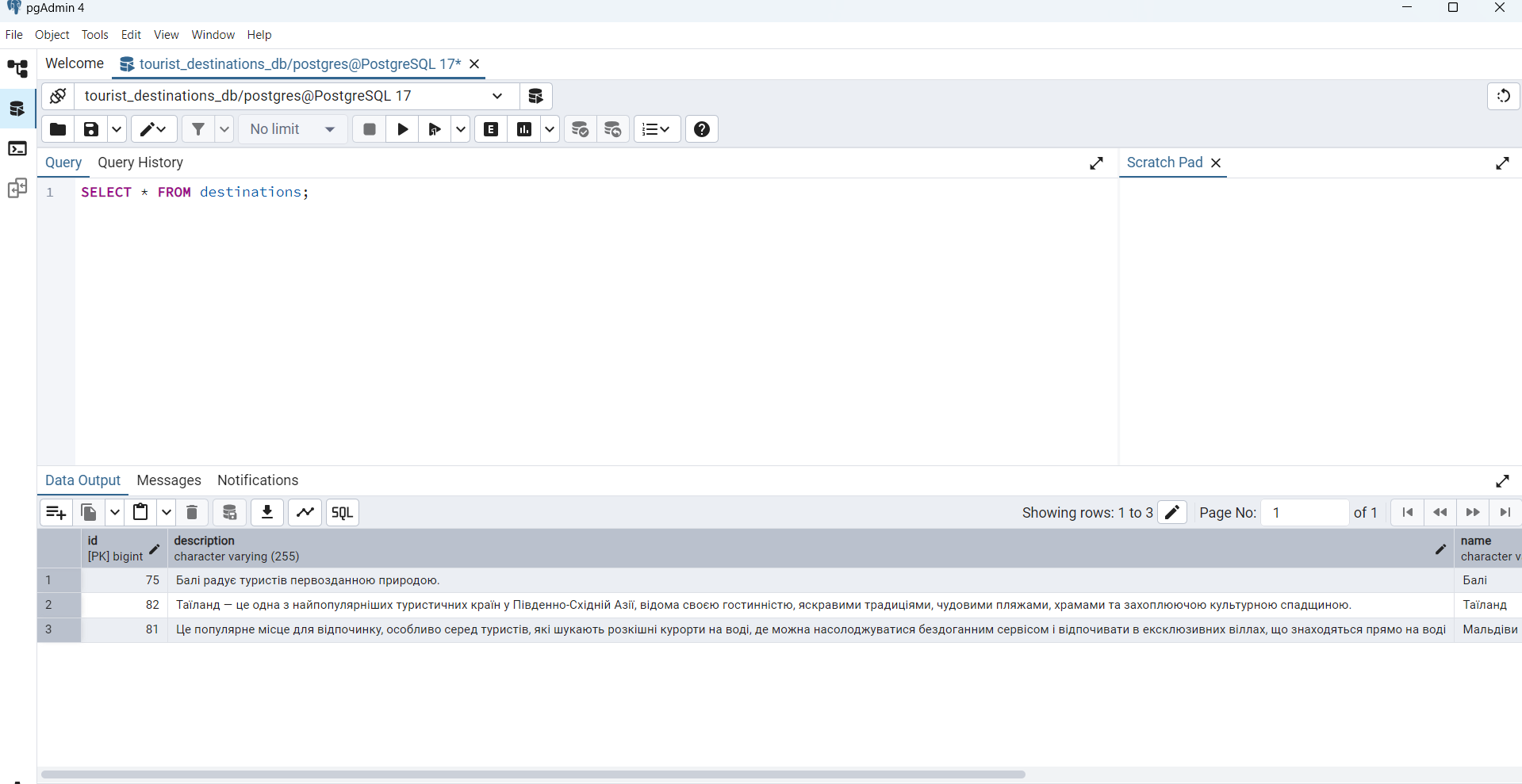
|  |
| --- |
| @ControllerAdvice public class GlobalExceptionHandler {     @ExceptionHandler(Exception.class)     public ResponseEntity<String> handleException(Exception ex) {         return new ResponseEntity<>(ex.getMessage(), HttpStatus.INTERNAL\_SERVER\_ERROR);     } } |



**Результати роботи:**

****

****

****

**Запитання для самоперевірки:**

1. Що автоматично створює Spring Data REST?

Spring Data REST автоматично створює REST API для JPA-репозиторіїв. Це включає ендпоінти для виконання CRUD-операцій, підтримку сортування, пагінації та фільтрації без написання додаткового коду контролерів.

1. Як налаштувати базовий шлях (base-path) до API?

Базовий шлях API можна налаштувати у файлі application.properties або application.yml, використовуючи параметр:

spring.data.rest.base-path=/apiє

Це означає, що всі автоматично згенеровані ендпоінти будуть доступні за адресою /api/{entityName}.

1. У яких випадках @RepositoryRestResource не підходить?
2. Якщо потрібно реалізувати складну бізнес-логіку, яка виходить за межі простих CRUD-операцій.
3. Якщо необхідно контролювати структуру запитів і відповідей (наприклад, формат JSON).
4. Якщо потрібна детальніша обробка помилок або складна авторизація/аутентифікація.
5. Якщо потрібно об'єднати кілька сутностей в один API-запит.
6. Як забезпечити сортування та пагінацію у Spring Data REST?

Spring Data REST підтримує вбудовану пагінацію та сортування через параметри запиту:

1. ?page=0&size=10 – вибір сторінки та кількості елементів на сторінці.
2. ?sort=title,asc – сортування за полем title у порядку зростання.
3. ?sort=title,desc – сортування за полем title у порядку спадання.
4. Як обмежити доступ до ендпоінтів, створених через @RepositoryRestResource?

Доступ можна обмежити за допомогою:

1. Використання анотації @PreAuthorize разом із Spring Security, наприклад:

@PreAuthorize("hasRole('ADMIN')")

@RepositoryRestResource(path = "courses")

public interface CourseRepository extends JpaRepository<Course, Long> {}

1. Використання @RestResource(exported = false) для відключення окремих методів:

@RestResource(exported = false)

void deleteById(Long id);

1. Використання HttpSecurity у конфігурації Spring Security для налаштування доступу до конкретних маршрутів.

**Висновок**

У цій лабораторній роботі я розглянула використання анотації @RepositoryRestResource у Spring Boot для автоматичного створення REST API. Це дозволило швидко розгорнути веб-сервіс, що взаємодіє з базою даних PostgreSQL за допомогою Spring Data JPA. Крім того, я налаштувала Lombok для спрощення роботи з модельними класами та реалізувала глобальну обробку помилок.