**ЗВІТ**

**до лабораторної роботи № 5**  
на тему:  
**"Створення програми для адміністрування даних з використанням патерну MVC."**

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

**Мета роботи:** закріпити знання щодо реалізації архітектурного патерну MVC, організації веб-інтерфейсу за допомогою Thymeleaf і виконання CRUD-операцій у Spring Boot. Отримати практичний досвід розробки веб-додатків, працюючи з базами даних через Spring JPA.

**Завдання:**

1. Ознайомитися з принципами реалізації архітектурного патерну Model-View-Controller (MVC).
2. Розробити програму для адміністрування даних (заповнення та редагування бази даних).
3. Забезпечити роботу програми через веб-інтерфейс з використанням Spring MVC і Thymeleaf.
4. Реалізувати CRUD-операції для взаємодії з базою даних за допомогою Spring JPA.

**Хід роботи:**

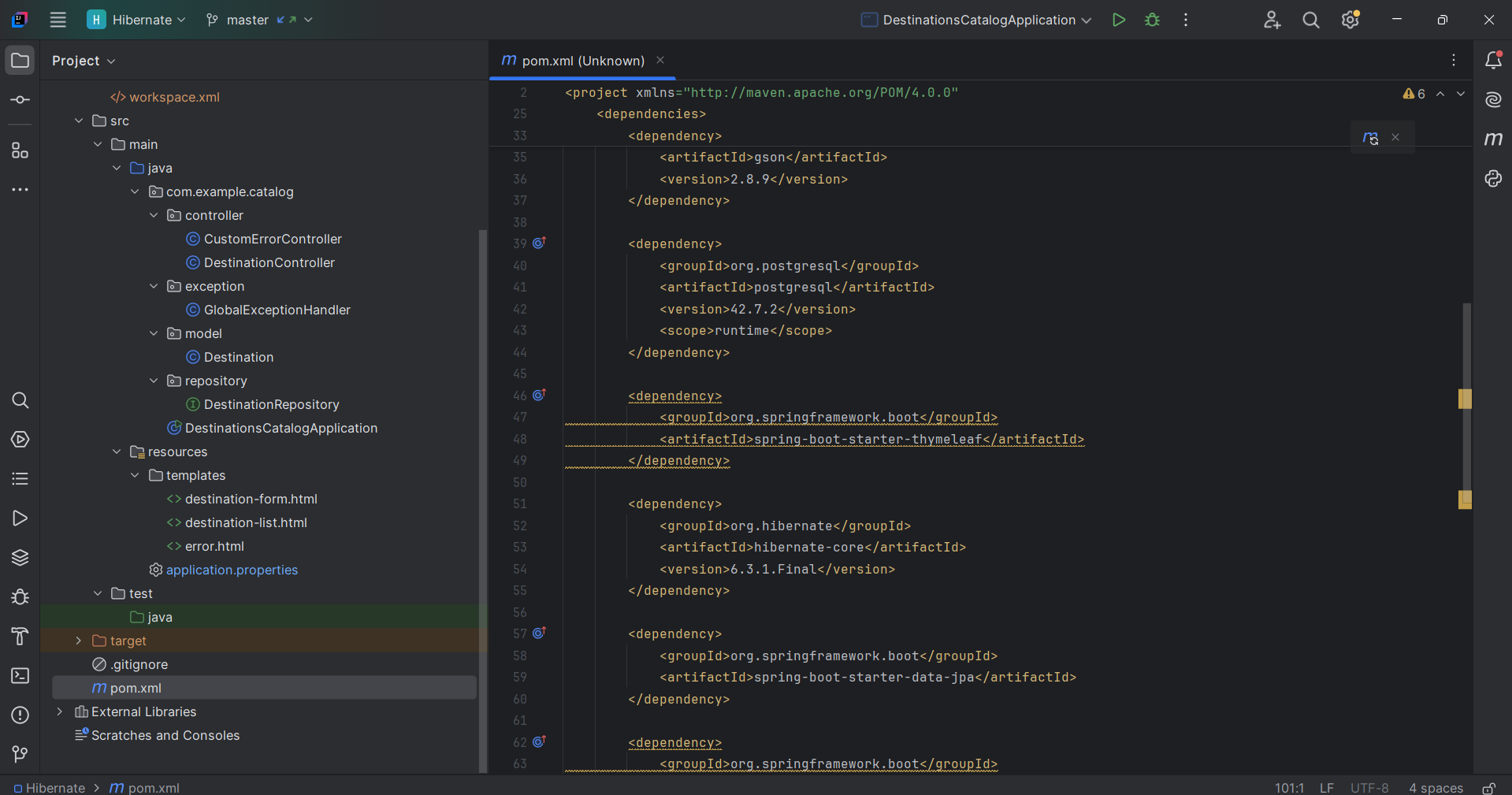
1. **Підготовка репозиторію GitHub**Створити нову гілку **lab5** у репозиторії проєкту.

Організувати структуру проєкту відповідно до концепції MVC.

1. **Початок роботи з** [**Thymeleaf**](https://www.thymeleaf.org/) **та** [**Bootstrap**](https://getbootstrap.com/)

Додати залежність у pom.xml:

|  |
| --- |
| <dependency>     <groupId>org.springframework.boot</groupId>     <artifactId>spring-boot-starter-thymeleaf</artifactId> </dependency> |

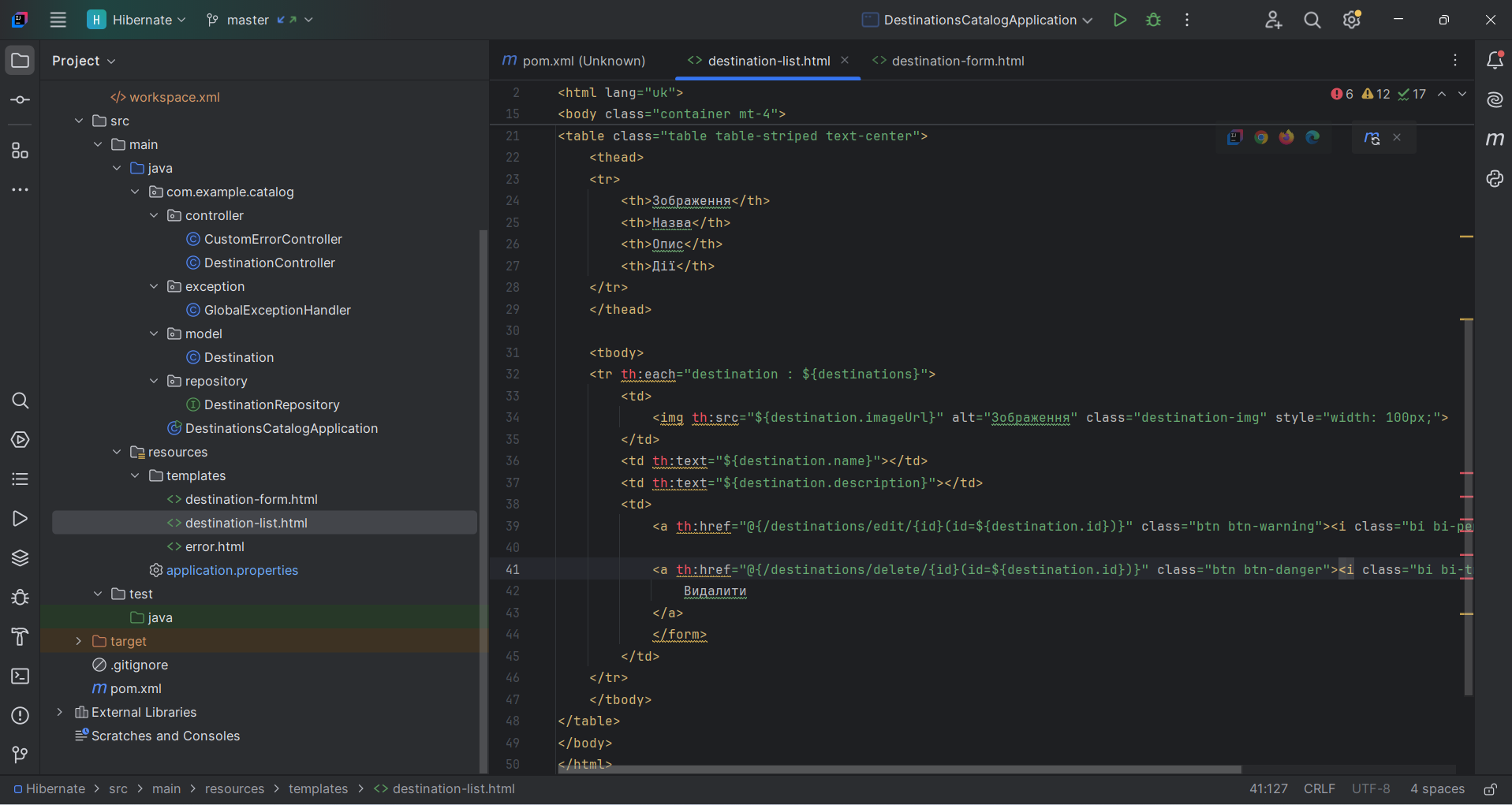


Додати Bootstrap у html:

|  |
| --- |
| <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"> |

Використати Thymeleaf для динамічного відображення даних:

|  |
| --- |
| <table class="table">     <tr th:each="item : ${items}">         <td th:text="${item.name}"></td>         <td th:text="${item.description}"></td>     </tr> </table> |



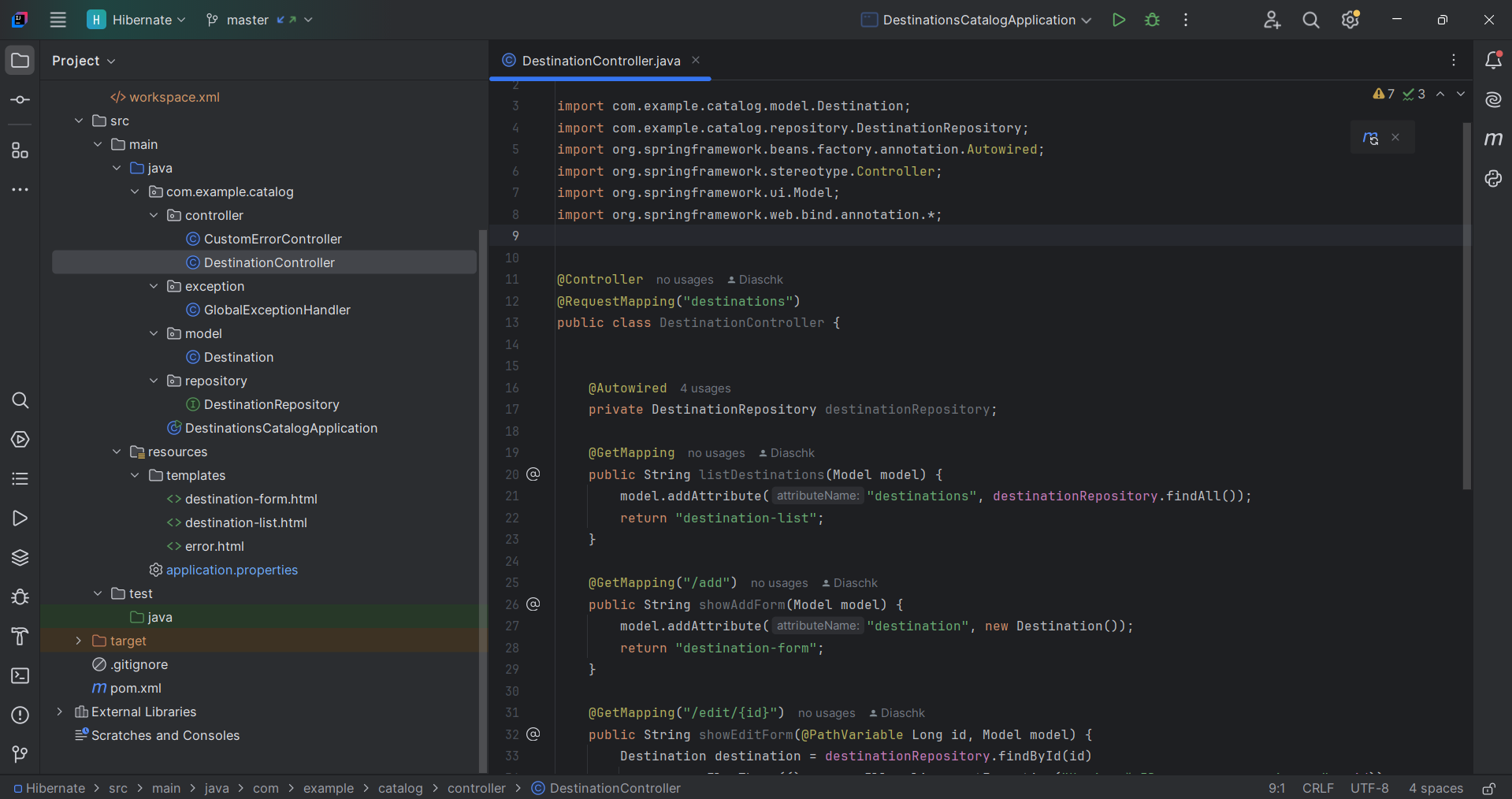
1. **Розробка програми адміністрування даних**

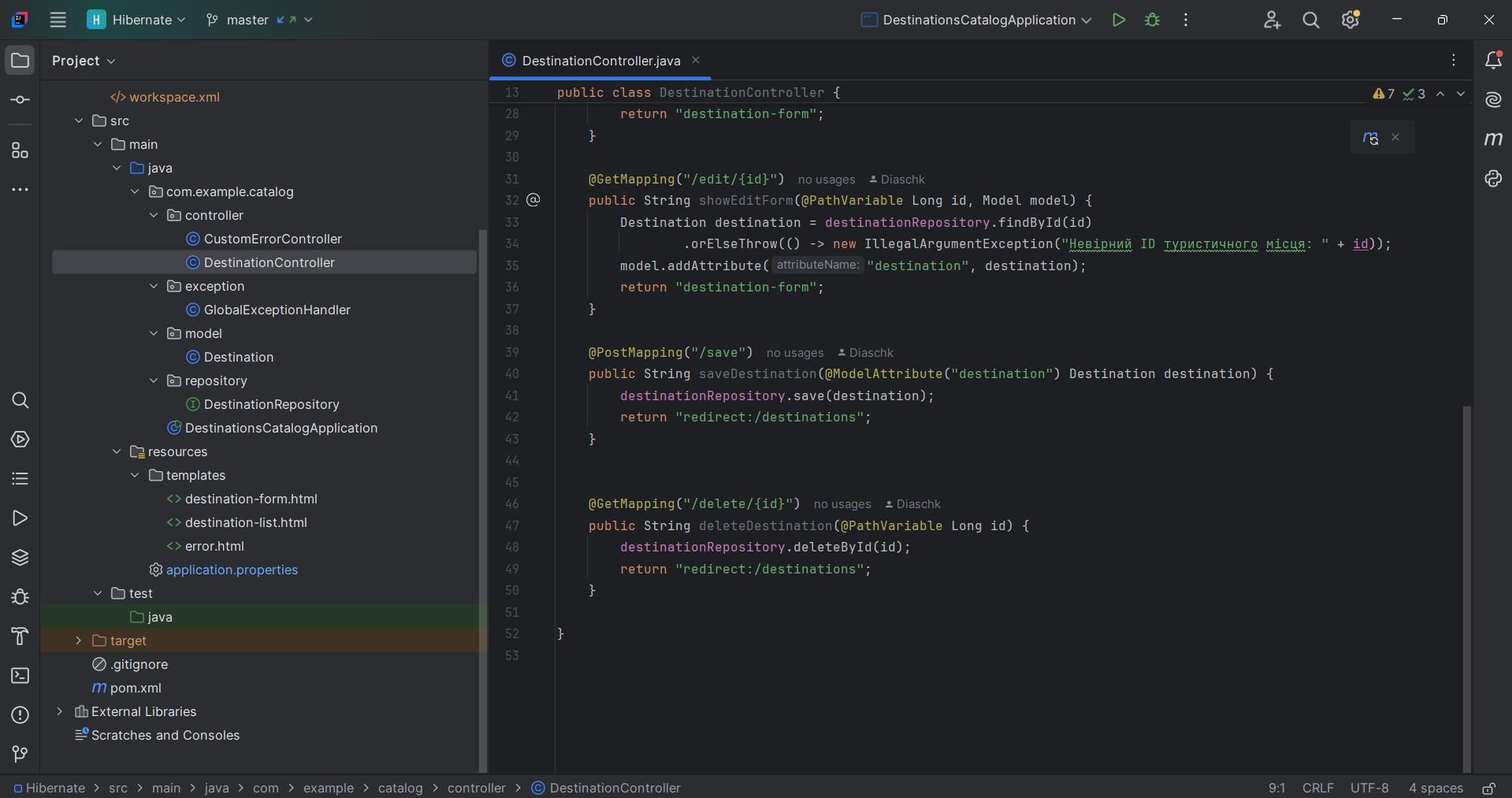
**Front-end (View):** реалізувати сторінки для CRUD-операцій.

**Back-end (Model, Controller):** створити контролер для роботи з даними:

|  |
| --- |
| @Controller @RequestMapping("/items") public class ItemController {     @Autowired     private ItemRepository repository;      @GetMapping     public String listItems(Model model) {         model.addAttribute("items", repository.findAll());         return "index";     } } |

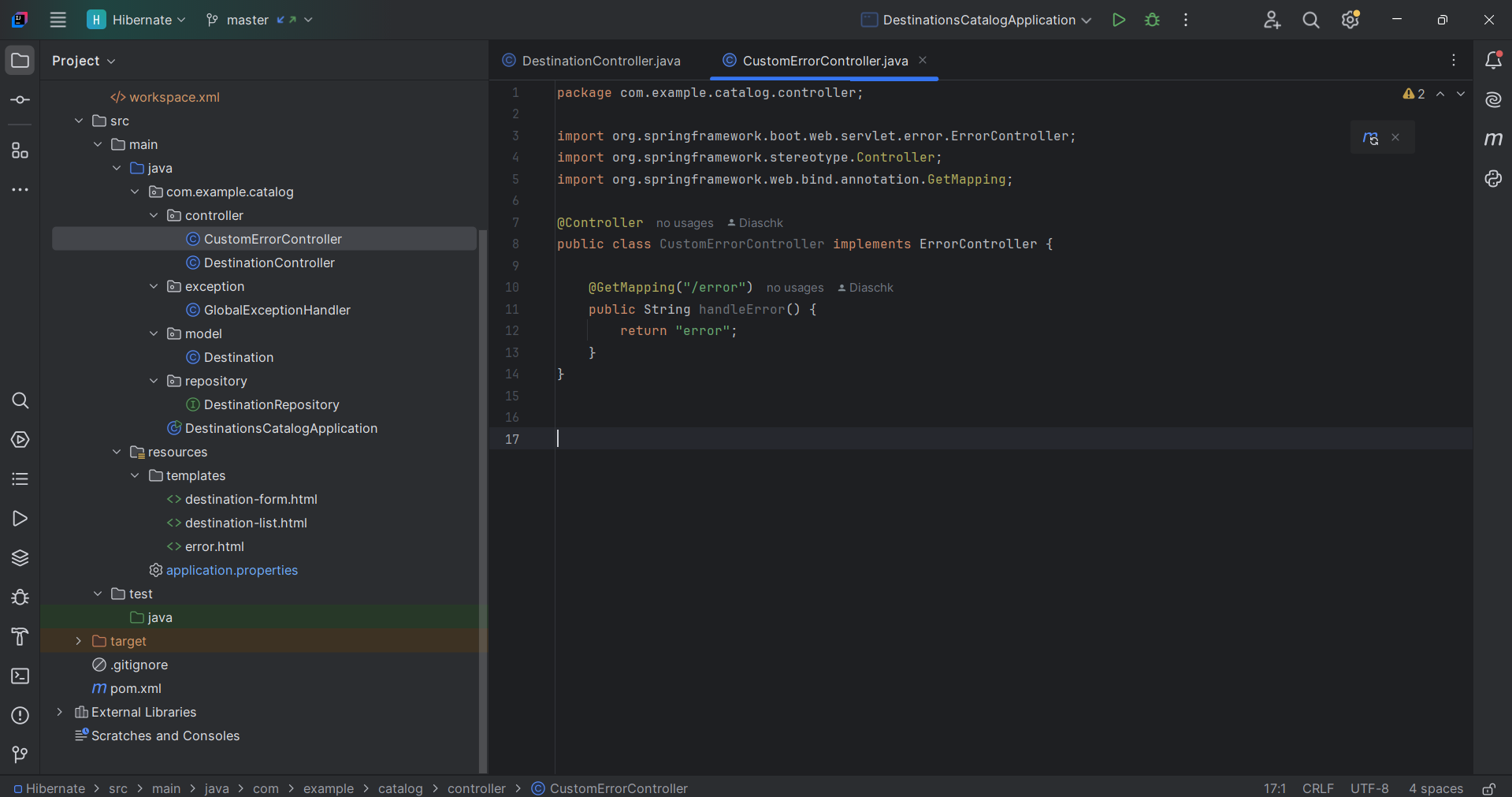
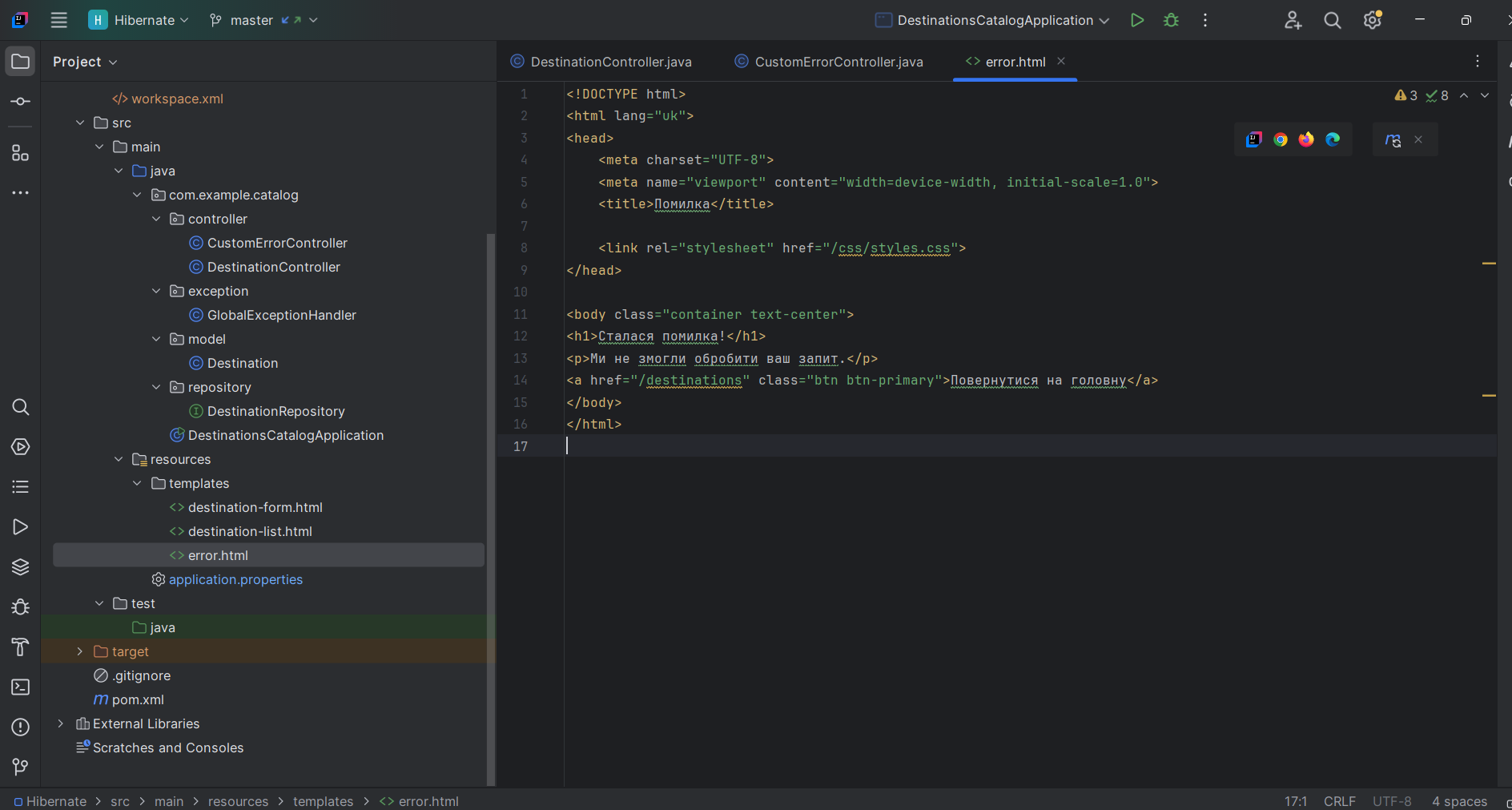
Реалізувати CRUD-операції через Spring JPA.





1. **Обробка помилок у Spring MVC** Для обробки помилок потрібно створити глобальний обробник:

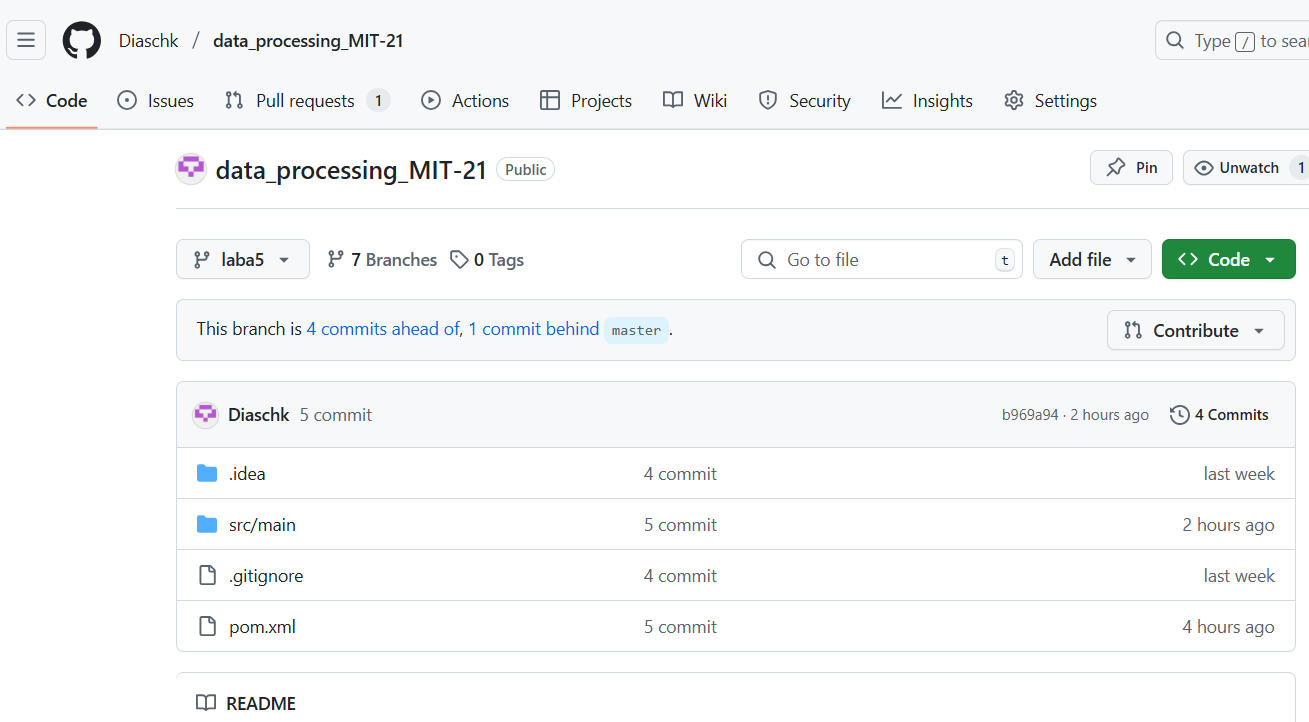
|  |
| --- |
| @ControllerAdvice public class GlobalExceptionHandler {     @ExceptionHandler(Exception.class)     public String handleException(Exception ex, Model model) {         model.addAttribute("message", ex.getMessage());         return "error";     } } |

Також потрібно додати сторінку error.html, яка буде відображатися у разі виникнення помилок.  
  


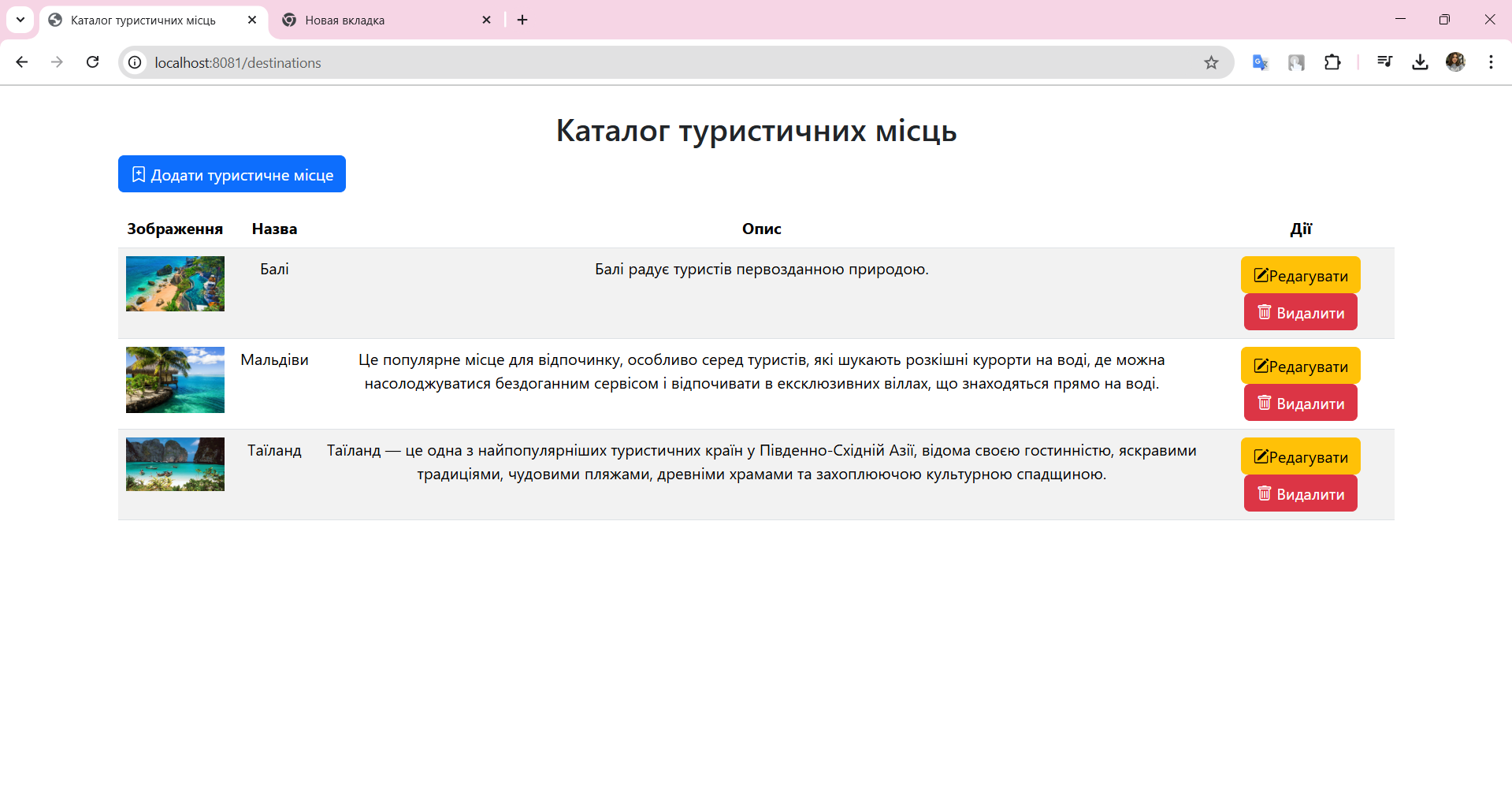
1. **Інтеграція частин**Забезпечити взаємодію між фронтендом і контролерами.

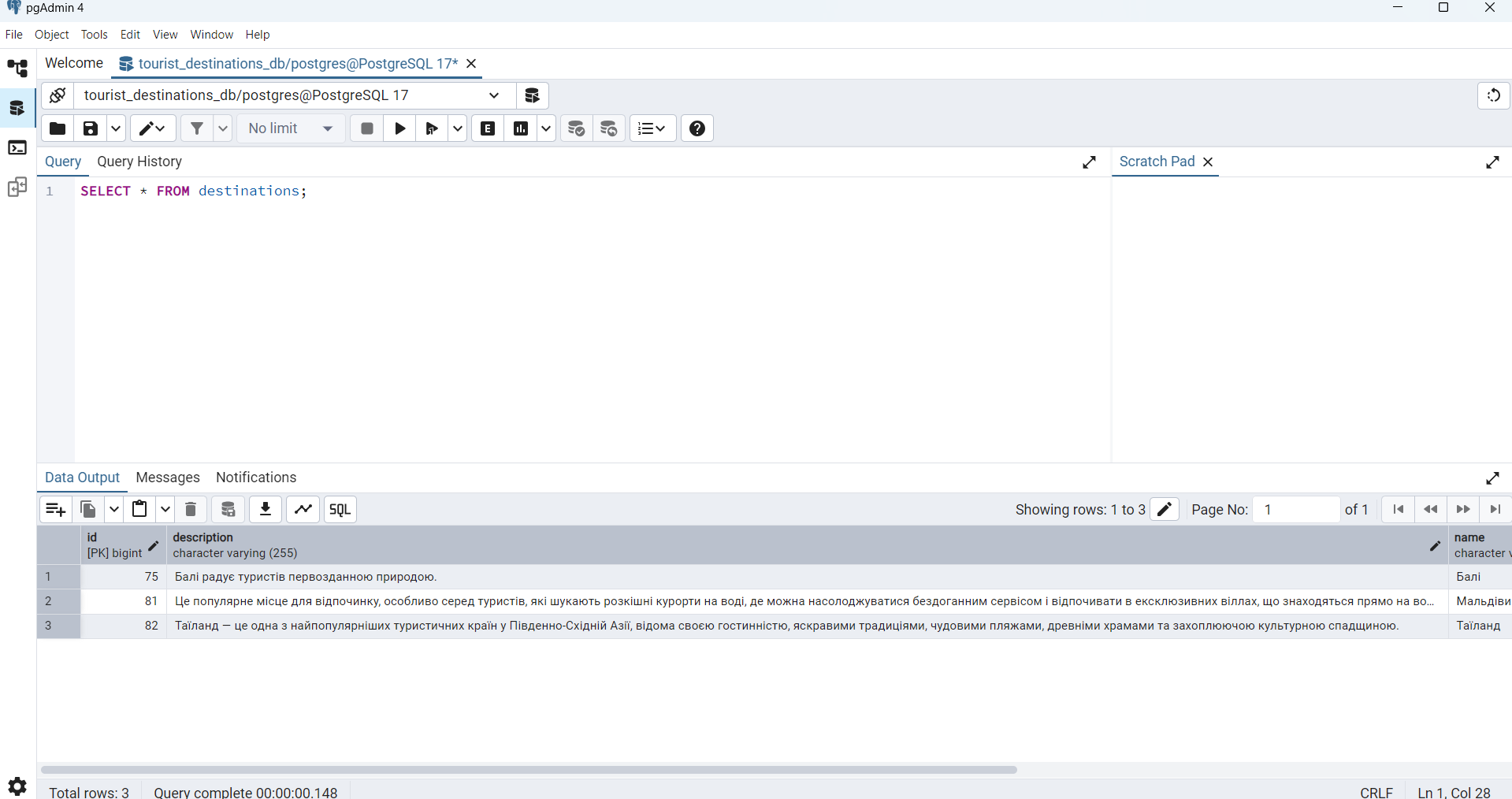
Перевірити коректність виконання CRUD-операцій.

1. **Завантаження у GitHub**Викласти реалізований код у репозиторій у гілку **lab5**.



**Результат лабораторної роботи:**





**Запитання для самоперевірки**

1. Що таке патерн MVC і які його основні компоненти?

**Патерн MVC (Model-View-Controller)** — це архітектурний шаблон, який розділяє застосунок на три компоненти:

1. **Model** — логіка роботи з даними та бізнес-логіка.
2. **View** — інтерфейс користувача.
3. **Controller** — обробник запитів, що керує взаємодією між Model і View.
4. Як Spring MVC реалізує взаємодію між Model, View та Controller?

Spring MVC використовує контролери (@Controller), які отримують HTTP-запити, взаємодіють із моделлю (сервіси та репозиторії), а потім передають дані у вигляді Model до представлення (Thymeleaf).

1. Які можливості надає Thymeleaf у Spring-застосунку?

Thymeleaf дозволяє створювати динамічні HTML-сторінки, підставляти змінні в шаблони, використовувати умови та цикли, працювати з фрагментами та локалізацією.

1. Як працює Spring JPA у взаємодії з базою даних?

Spring JPA забезпечує абстракцію над рівнем взаємодії з базою даних, дозволяючи працювати з нею через репозиторії (інтерфейси, що розширюють JpaRepository).

1. Як забезпечити виконання CRUD-операцій у Spring MVC?

Використовуючи Spring JPA та контролери:

1. @GetMapping — отримання даних;
2. @PostMapping — створення записів;
3. @PutMapping або @PatchMapping — оновлення даних;
4. @DeleteMapping — видалення записів.
5. Як забезпечити валідацію даних у формах на веб-сторінках?

Використовуючи анотації Spring Validation (@NotNull, @Size, @Email) у моделях та перевірку в контролерах із BindingResult.

1. Як налаштувати підключення PostgreSQL у Spring Boot-проєкті?

Додати залежність spring-boot-starter-data-jpa та вказати налаштування в application.properties:

spring.datasource.url=jdbc:postgresql://localhost:5432/my\_db

spring.datasource.username=postgres

spring.datasource.password=password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

1. Які HTTP-запити використовуються для CRUD-операцій у Spring MVC?
   1. GET — отримання даних;
   2. POST — створення нового запису;
   3. PUT / PATCH — оновлення запису;
   4. DELETE — видалення запису.
2. Як організована робота контролерів у Spring MVC?

Контролери позначаються анотацією @Controller або @RestController, обробляють HTTP-запити через @GetMapping, @PostMapping тощо та передають дані у Model або як JSON-відповіді.

1. Які засоби використовуються для стилізації веб-інтерфейсу у Thymeleaf?

Bootstrap, CSS, SCSS, а також можливості Thymeleaf для динамічного підключення стилів (th:class, th:style).

**Висновок**

У ході виконання лабораторної роботи я закріпила знання щодо архітектурного патерну MVC та його реалізації у Spring Boot. Також розробила веб-застосунок для адміністрування даних, який використовує Spring MVC, Thymeleaf та Spring JPA для виконання CRUD-операцій.

На фронтенді реалізувала динамічне відображення даних за допомогою Thymeleaf і застилізувала інтерфейс із використанням Bootstrap. Для взаємодії з базою даних застосувала Spring JPA, що забезпечив ефективне виконання операцій із збереження, редагування та видалення даних. Також опрацювала механізми обробки помилок у Spring MVC, що дозволило покращити стабільність і надійність застосунку.

У результаті я отримала практичний досвід роботи зі Spring Boot, налаштування PostgreSQL та організації структури проєкту за принципами MVC.