ЗАТВЕРДЖУЮ
Начальник кафедри інформаційних технологій та систем електронних комунікацій, к.т.н., доцент
Назарій БУРАК
«____» _____ 20___ р.

МЕТОДИЧНА РОЗРОБКА

для проведення лабораторного заняття із здобувачами освіти першого (бакалаврського) рівня вищої освіти спеціальності F3 «Комп'ютерні науки»
з дисципліни «WEB-технології та WEB-програмування»

Лабораторна робота № 8: «Підключення бази даних до застосунку Spring Boot засобами Spring Data JPA. Автентифікація користувачів (Spring Security). Модульне тестування»
(тема 2.4)

Мета: навчитися описувати сутності JPA і зв'язки між ними, створювати репозиторії Spring Data JPA з похідними запитами, JPQL і пагінацією, усувати проблему N+1, керувати схемою БД міграціями Flyway, налаштовувати автентифікацію й авторизацію засобами Spring Security з хешуванням паролів, а також писати модульні та інтеграційні тести.

Час: 4 години.

Місце заняття: комп'ютерна лабораторія.

Матеріальне забезпечення: ПК з доступом до мережі Internet; JDK 21; IntelliJ IDEA; Maven; PostgreSQL; Postman або curl; браузер Chrome; Git; демо-проєкт https://github.com/SidiromUA/shelter-registry (тег `lab-08`).

Після лабораторного заняття здобувачі освіти повинні:

вміти: створювати сутності з `@ManyToOne`/`@OneToMany`; оголошувати репозиторії `JpaRepository` з похідними запитами, `@Query` і `Pageable`; застосовувати `@EntityGraph`/`join fetch`; створювати міграції `V1__…sql`; налаштовувати `SecurityFilterChain`, вхід через форму і HTTP Basic, правила доступу за ролями; зберігати паролі хешами BCrypt; писати тести з Mockito, `@DataJpaTest` і `@WebMvcTest`;

знати: поняття ORM, контексту збереження і dirty checking; сторону-власника зв'язку; ліниве завантаження і проблему N+1; принципи міграцій;

різницю між автентифікацією і авторизацією, кодами 401 і 403; призначення CSRF-токена; піраміду тестування.

I. Підготовка до заняття

Викладач перевіряє на ПК лабораторії наявність JDK 21, Maven, PostgreSQL і Postman. Здобувачі освіти до заняття опрацьовують лекцію 8 і мають власний проєкт з ЛР 7 (сторінки Thymeleaf, REST API, валідація, профілі dev/prod).

II. Структура заняття

Короткі теоретичні відомості

Сутність JPA – клас з `@Entity`, полем `@Id` і конструктором без параметрів. Зв'язок «багато → один» описує `@ManyToOne` з `@JoinColumn` (сторона-власник), зворотний – `@OneToMany(mappedBy=…)`. Зв'язки роблять лінивими (`FetchType.LAZY`), а потрібні дані завантажують `join fetch` або `@EntityGraph`, щоб уникнути **проблеми N+1**. **Spring Data JPA** створює реалізацію інтерфейсу `JpaRepository`, запити будуються з назви методу або задаються `@Query` мовою JPQL; `Pageable` і `Page` реалізують пагінацію. **Flyway** виконує скрипти `db/migration/V<N>__опис.sql` рівно один раз. **Spring Security** – ланцюжок фільтрів: автентифікація (хто ви? – 401) і авторизація (що вам можна? – 403); паролі зберігаються хешами BCrypt; форми захищено CSRF-токеном. **Модульні тести** перевіряють клас із заглушками Mockito, **інтеграційні** – шари застосунку (`@DataJpaTest`, `@WebMvcTest`, `@SpringBootTest`).

2.1. Огляд демо-проєкту

```
git checkout lab-08
git diff lab-07 lab-08 --stat
mvn test                 # 20 тестів: сервіс (Mockito), репозиторій,
API, контекст
mvn spring-boot:run      # http://localhost:8080/shelters, вхід:
admin/admin, inspector/inspector
```

Компонент	Що змінилося
`model.*`	сутності JPA `Shelter`, `District`, `Inspection`, `User` зі зв'язками
`repository.*`	інтерфейси `JpaRepository`; класи `Jdbc*Repository` видалено
`db/migration/V1__schema.sql`, `V2__data.sql`	міграції Flyway; паролі – хеші `{bcrypt}`
`config.SecurityConfig`	два ланцюжки: сторінки (форма, сесія, CSRF), API (HTTP Basic, без сесії)
`templates/login.html`	власна сторінка входу; ім'я користувача і кнопка виходу в шапці
`src/test/java/…`	`ShelterServiceTest`, `ShelterRepositoryTest`, `ShelterApiControllerTest`

Таблиця 1 – Зміни демо-проєкту (тег lab-08)

2.2. Дослідження

1. Запустіть застосунок у профілі dev і знайдіть у журналі SQL, згенерований Hibernate, для сторінки списку. Тимчасово видаліть `@EntityGraph` з `findAll()` у `ShelterRepository` і порівняйте кількість запитів (проблема N+1). Поверніть анотацію.
2. Відкрийте H2-консоль і перегляньте таблицю `flyway_schema_history` і хеші паролів у `users`.
3. Без входу спробуйте відкрити `/shelters/new` – має відкритися сторінка входу. Увійдіть як `inspector`: зареєструйте перевірку укриття; спроба відкрити `/shelters/new` має дати 403. Увійдіть як `admin` і створіть укриття.
4. У DevTools знайдіть приховане поле `_csrf` у формі. Видаліть його через DevTools і надішліть форму – очікується 403.
5. Виконайте запити до API (табл. 2).
6. Запустіть `mvn test`; відкрийте звіт `target/surefire-reports`. Навмисно зламайте правило в сервісі (наприклад, приберіть перевірку унікальності адреси) і переконайтеся, що тест «червоніє».

Запит	Очікувана відповідь
`GET /api/shelters?page=0&size=4&sort=capacity,desc`	200, сторінка з `totalElements`, `totalPages`
`POST /api/shelters` без автентифікації	401
`POST /api/shelters, -u inspector:inspector`	403
`POST /api/shelters, -u admin:admin`	201 + `Location`
`POST /api/shelters/4/inspections, -u inspector:inspector`	200, стан укриття змінено; інспектор – за входом

Таблиця 2 – Запити для перевірки безпеки і пагінації

2.3. Індивідуальне завдання

Доповніть власний проєкт з ЛР 7:

1. Замініть стартер `jdbc` на `data-jpa`, додайте `flyway`, `security` і тестові стартери (див. `pom.xml` демо-проєкту). Встановіть `ddl-auto: none` і `open-in-view: false`.
2. Перенесіть SQL-скрипти в міграції `V1__schema.sql` і `V2__data.sql`; додайте третю міграцію з реальною зміною схеми (новий стовпець або таблиця) і використайте її в коді.
3. Перетворіть класи моделі на сутності JPA зі зв'язками `@ManyToOne` (LAZY) і `@OneToMany(mappedBy)`; переліки – `EnumType.STRING`.
4. Замініть JDBC-репозиторії на інтерфейси `JpaRepository`: щонайменше три похідні запити, один запит `@Query` JPQL (з `join fetch` або агрегатом), пагінація в REST API. Перевірте журнал SQL на відсутність N+1.
5. Додайте таблицю користувачів з ролями (щонайменше дві) і паролями BCrypt; реалізуйте `UserDetailsService` через репозиторій.
6. Налаштуйте `SecurityConfig`: власна сторінка входу і вихід для сторінок, HTTP Basic без сесії для API, правила доступу за ролями (перегляд – усім, зміни – за ролями), CSRF для форм. У шапці покажіть ім'я користувача і кнопку виходу.
7. Визначайте автора дій (хто створив, хто перевірив) з `Principal`, а не з даних запиту.
8. Напишіть тести: щонайменше 4 модульні тести сервісу з Mockito (успіх і порушення бізнес-правил), 3 тести репозиторію (`@DataJpaTest`), 4

тести API (`@WebMvcTest`: 200, 400, 401/403, 201). Усі тести мають проходити командою `mvn test`.

9. Оновіть `README.md` (облікові записи для перевірки, ролі, запуск тестів), зафіксуйте зміни, створіть тег `lab-08` і надішліть на GitHub.

Додаткові завдання (за бажанням): налаштуйте GitHub Actions для запуску `mvn test` під час кожного `push`; використайте Testcontainers з PostgreSQL замість H2 у тестах; додайте `@PreAuthorize` на методи сервісу; реалізуйте реєстрацію нового користувача з перевіркою складності пароля.

2.4. Вимоги до звіту

Звіт (PDF або DOC, до 6 сторінок) містить: мету; ER-діаграму або перелік сутностей зі зв'язками; фрагменти коду сутності, репозиторію (похідні запити, JPQL), `SecurityConfig` і по одному тесту кожного виду; вміст каталогу міграцій і таблиці `flyway_schema_history`; журнал SQL до і після усунення N+1; знімки сторінки входу, відмови 403 і відповідей API 401, 403, 201; результат `mvn test` («Tests run: …, Failures: 0»); посилання на репозиторій з тегом `lab-08`; відповіді на 3 контрольні запитання; висновки. У ВУ в завданні «ЛР 8» у поле «Текстова відповідь» вставте посилання на репозиторій, файлом додайте звіт.

2.5. Контрольні запитання

1. Що таке ORM, JPA і Hibernate? Що таке dirty checking?
2. Що таке сторона-власник зв'язку? Для чого `mappedBy` і `cascade`?
3. Як Spring Data JPA будує запит з назви методу? Чим JPQL відрізняється від SQL?
4. Що таке проблема N+1 і як ви її усунули?
5. Як працює Flyway? Чому не можна змінювати застосовану міграцію?
6. Чим автентифікація відрізняється від авторизації? Коли повертається 401, а коли 403?
7. Як зберігаються паролі у вашому застосунку і чому саме так?
8. Що таке CSRF і чому для API з HTTP Basic захист вимкнено?
9. Чим модульний тест відрізняється від інтеграційного? Для чого Mockito?

2.6. Критерії оцінювання (максимально 5 балів)

Бали	Критерій
5	сутності зі зв’язками; репозиторії з похідними запитами, JPQL і пагінацією без N+1; три міграції Flyway; вхід і ролі для сторінок і API, BCrypt, CSRF; автор дій з `Principal`; усі вимоги до тестів виконано, `mvn test` зелений; тег `lab-08` на GitHub; здобувач вільно пояснює роботу JPA, Spring Security і тестів
4	робота виконана повністю, є незначні недоліки (напр. менше тестів, ніж потрібно, або N+1 в одному місці)
3	виконано не менше 60 % (напр. немає тестів API або міграцій) або є суттєві помилки (паролі без хешування)
0–2	виконано менше 60 % або здобувач не може пояснити поданий код

Робота, подана після встановленого терміну без поважних причин, оцінюється з пониженням на 1 бал.

III. Підведення підсумків заняття

Викладач перевіряє роботу застосунків і тестів, звертає увагу на типові помилки (EAGER-зв’язки і N+1, `EnumType.ORDINAL`, зміна застосованої міграції, `ddl-auto: update` у prod, паролі `{noop}`, неправильний порядок правил доступу, тести, що залежать від поточної дати чи порядку запуску), підсумовує лабораторний цикл і нагадує, що проєкт ЛР 1–8 є основою курсової роботи.

Методичну розробку розробив:

Доцент кафедри ІТтаСЕК, к.т.н. Роман ГОЛОВАТИЙ

Методична розробка обговорена на засіданні кафедри ІТтаСЕК
Протокол № ____ від «____» _____ 20___ р.