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 № 3: «Розроблення об'єктної моделі вебзастосунку за шаблоном MVC. Перевірка доступності вебсторінок на відповідність рекомендаціям WCAG»
(тема 1.3)

Мета: навчитися розкладати код вебзастосунку на шари (модель, репозиторій, сервіс, контролер, представлення), проєктувати сутності й DTO, обробляти помилки валідації власними винятками, а також перевіряти і виправляти доступність вебсторінок відповідно до WCAG 2.2.

Час: 4 години.

Місце заняття: комп'ютерна лабораторія.

Матеріальне забезпечення: ПК з доступом до мережі Internet; JDK 21; IntelliJ IDEA; Maven; PostgreSQL; браузер Chrome з розширенням axe DevTools; Git; демо-проєкт https://github.com/SidiromUA/shelter-registry (тег `lab-03`).

Після лабораторного заняття здобувачі освіти повинні:

вміти: створювати класи сутностей, переліки (enum) і DTO; оголошувати інтерфейс репозиторію та його JDBC-реалізацію; реалізовувати сервіс з валідацією і власними винятками; передавати залежності через конструктор; виконувати аудит доступності засобами Lighthouse і axe та виправляти виявлені порушення;

знати: обов'язки шарів багаторівневої архітектури, компоненти MVC, відмінність сутності від DTO, принципи WCAG (POUR) і основні вимоги до доступних форм і таблиць.

I. Підготовка до заняття

Викладач перевіряє на ПК лабораторії наявність розширення axe DevTools і доступ до БД з ЛР 2. Здобувачі освіти до заняття опрацьовують лекцію 3 і мають власну БД та JDBC-код з ЛР 2.

II. Структура заняття

Короткі теоретичні відомості

Контролер приймає запит і обирає представлення, **сервіс** містить бізнес-логіку і валідацію, **репозиторій** працює з БД. Сервіс залежить від *інтерфейсу* репозиторію, а конкретну реалізацію передають через конструктор. **Сутність** містить дані й бізнес-правила, **DTO** – лише дані для обміну. **WCAG 2.2** вимагає, щоб сторінка була сприйманою, керованою, зрозумілою і надійною: семантична розмітка, підписи полів, контраст, керування з клавіатури.

2.1. Огляд демо-проєкту

```
git checkout lab-03
mvn compile exec:java    # генерує web/shelters.html
                         # і web/shelters-errors.html
```

Розгляньте пакети: `model` (`Shelter`, `District`, `Inspection`, переліки `ShelterType`, `ShelterStatus`), `dto` (`ShelterView`, `CreateShelterRequest`), `repository` (інтерфейси) і `repository/jdbc` (реалізації), `service` (`ShelterService`), `controller`, `view` (`ShelterListPage` формує HTML з екрануванням), `exception`. Простежте шлях виклику `controller.create(...)` → `service.create(...)` → `repository.save(...)`.

```
public ShelterView create(CreateShelterRequest r) {
    List<String> errors = new ArrayList<>();
    if (r.name() == null || r.name().isBlank())
        errors.add("Вкажіть назву укриття");
    if (r.capacity() == null || r.capacity() <= 0)
        errors.add("Місткість має бути додатним числом");
    // ... перевірка адреси, типу, району
    if (!errors.isEmpty()) throw new ValidationException(errors);
    Shelter s = new Shelter(r.name().trim(), r.address().trim(),
            district, type, r.capacity(), r.accessible());
    return ShelterView.from(shelters.save(s));
}
```

Відкрийте `web/shelters-errors.html`: сервіс повернув п’ять помилок одночасно, а сторінка оголошує їх атрибутом `role="alert"`.

2.2. Аудит доступності

```
cd web
jwebserver -p 8000
```

1. Відкрийте http://localhost:8000/inaccessible-example.html. DevTools → **Lighthouse** → категорія *Accessibility* → *Analyze page load*. Запишіть оцінку і перелік порушень.
2. Запустіть **axe DevTools** → *Scan all of my page*. Для кожного порушення знайдіть критерій WCAG.
3. Пройдіть сторінку лише клавіатурою (Tab, Shift+Tab, Enter): чи можна відкрити меню і натиснути «OK»?
4. Повторіть перевірку для `index.html` і `add-shelter.html` та порівняйте результати. Розбір порушень – у файлі `docs/accessibility.md`.

2.3. Індивідуальне завдання

Розвиньте власний проєкт з ЛР 2:

1. Пакет `model`: класи сутностей вашої предметної області; щонайменше один перелік `enum` (стан, тип, категорія) з українськими підписами; щонайменше один метод бізнес-правила в сутності.
2. Пакет `dto`: DTO для відображення (`record`) і DTO для створення об’єкта з форми.
3. Пакет `repository`: інтерфейс репозиторію головної сутності і його реалізація на JDBC (перенесіть код DAO з ЛР 2); `SQLException` перетворюйте на власний неперевірюваний виняток.
4. Пакет `service`: сервіс з методами `findAll`, `findById`, `create`, `delete`; повна валідація з переліком усіх помилок (`ValidationException`), `NotFoundException` для відсутніх об’єктів. Залежності – лише через конструктор.
5. Пакети `controller` і `view`: контролер з методами `list()` і `create(...)`; представлення формує доступну HTML-сторінку (мова, `<main>`, таблиця з `<caption>` і `<th scope>`, текстове позначення стану, `role="alert"` для помилок, екранування даних).
6. `App.java` «збирає» шари і зберігає дві сторінки: зі списком і з помилками валідації.
7. Створіть статичну сторінку з формою додавання об’єкта вашої предметної області з урахуванням WCAG (`label`, `fieldset`/`legend`,

підказки `aria-describedby`, видимий фокус). Досягніть оцінки Lighthouse Accessibility **не нижче 90** для всіх своїх сторінок.

8. Зафіксуйте зміни, створіть тег `lab-03` і надішліть на GitHub.

2.4. Вимоги до звіту

Звіт (PDF або DOC, до 6 сторінок) містить: мету; схему пакетів і класів (знімок структури проєкту в IDE або UML-діаграму); фрагменти коду сервісу з валідацією і конструктора з передаванням залежностей; знімки згенерованих сторінок (список, помилки); знімки звітів Lighthouse для `inaccessible-example.html` і для ваших сторінок; таблицю «порушення – критерій WCAG – виправлення» (щонайменше 5 рядків); посилання на репозиторій з тегом `lab-03`; відповіді на 3 контрольні запитання; висновки. У ВУ в завданні «ЛР 3» у поле «Текстова відповідь» вставте посилання на репозиторій, файлом додайте звіт.

2.5. Контрольні запитання

1. Які обов’язки має контролер, сервіс і репозиторій?
2. Чому сервіс отримує репозиторій через конструктор і залежить від інтерфейсу?
3. Чим сутність відрізняється від DTO? Наведіть приклад з вашого проєкту.
4. Навіщо перетворювати `SQLException` на власний виняток?
5. Що таке XSS і як його запобігає екранування?
6. Назвіть чотири принципи WCAG і по одному порушенню, яке ви знайшли.
7. Чому автоматичної перевірки доступності недостатньо?

2.6. Критерії оцінювання (максимально 5 балів)

Бали	Критерій
5	усі шари реалізовано коректно, залежності через конструктор, повна валідація з власними винятками; сторінки доступні (Lighthouse ≥ 90), таблиця виправлень повна; тег `lab-03` на GitHub; здобувач вільно пояснює архітектуру
4	робота виконана повністю, є незначні недоліки (напр. неповна валідація або оцінка Lighthouse 80–89)
3	виконано не менше 60 % (напр. змішано шари або немає аудиту доступності) або є суттєві помилки
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 структуру проєктів і результати аудиту, звертає увагу на типові помилки (SQL у контролері, створення залежностей усередині класу, поля форми без підписів), нагадує про необхідність погодити тему курсової роботи.

Методичну розробку розробив:

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

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