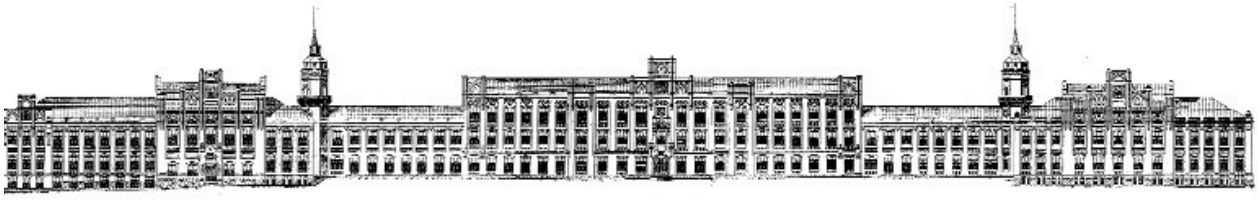

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3
Тема: «Основи проектування розгортання»

Виконала
студентка групи ІА-32:
Іванова Анастасія
Юріївна

Перевірив:
Мягкий Михайло
Юрійович

Зміст

Тема проекту

Варіант: 26
Опис теми: Download manager (iterator, command, observer, template method, composite, p2p) Інструмент для скачування файлів з інтернету по протоколах http або https з можливістю продовження завантаження в зупиненому місці, розподілу швидкостей активним завантаженням, ведення статистики завантажень, інтеграції в основні браузери (firefox, opera, internet explorer, chrome).

Теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Діаграма розгортання показує фізичну структуру системи — на якому обладнанні або середовищі виконуються її компоненти.
Основними елементами є вузли (nodes), які можуть бути:

- пристроями (device) — фізичне обладнання (ПК, сервер);
- середовищем виконання (execution environment) — програмне середовище (операційна система, вебсервер).

Між вузлами встановлюються зв'язки, що позначають канали комунікації, зазвичай із вказівкою протоколу (HTTP, IPC, TCP).
Всередині вузлів розміщуються артефакти (artifacts) — фізичні файли, які розгортаються у системі (виконувані файли, скрипти, таблиці БД, конфігурації).
Розрізняють описові (загальні) та екземплярні (конкретні) діаграми розгортання.

Діаграма компонентів (Component Diagram)

Діаграма компонентів відображає систему як сукупність модулів (компонентів), які взаємодіють між собою.
Компоненти можуть бути:

- логічні — програмні модулі, що реалізують окремі частини логіки;
- фізичні — виконувані файли або бібліотеки;
- виконувані — процеси або служби, що реально запускаються.

Діаграма компонентів дає змогу:

- візуалізувати структуру коду;
- показати залежності між модулями;
- визначити, які частини можна повторно використовувати;
- відобразити зв'язки між клієнтською, серверною та спільною частинами системи.

Діаграма послідовностей (Sequence Diagram)

Діаграма послідовностей моделює взаємодію об'єктів у часі.
Вона показує порядок виклику методів або обміну повідомленнями між об'єктами.
Основні елементи:

- Актори — користувачі або зовнішні системи;
- Об'єкти — елементи системи, що взаємодіють;

- Лінії життя (lifelines) — час існування об’єктів;
- Повідомлення — виклики методів або передачі даних;
- Блоки alt/loop — для умов і циклів.

Діаграми послідовностей деталізують сценарії варіантів використання та показують логіку виконання операцій у системі.

Хід роботи

Створення діаграми розгортання (Deployment Diagram)

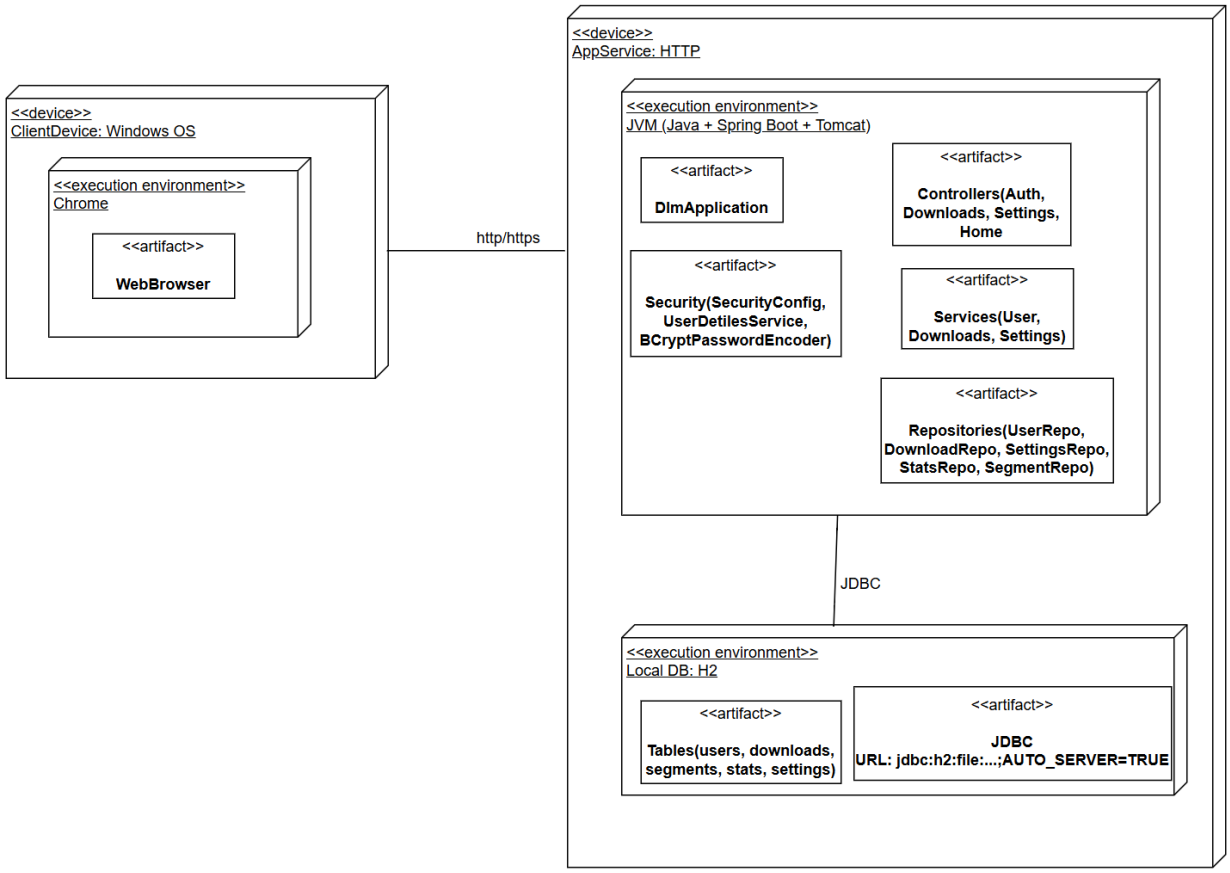

Рис. 1. Діаграма розгортання веб-застосунку Download Manager

На діаграмі розгортання зображено фізичне середовище роботи системи Download Manager. Застосунок реалізовано на основі фреймворку Spring Boot, що виконується у віртуальному середовищі JVM (Java Virtual Machine) з вбудованим веб-сервером Tomcat.

Компоненти системи:

1. ClientDevice (Windows OS)
   На стороні користувача використовується веб-браузер Google Chrome, який забезпечує взаємодію із серверною частиною через протокол HTTP/HTTPS.

Користувач може здійснювати вхід у систему, додавати завантаження за URL, переглядати список, змінювати статус завантажень і редагувати налаштування.

2. AppService (HTTP)
   Це серверна частина застосунку, що реалізує логіку роботи менеджера завантажень.
   У межах виконувального середовища JVM (Java + Spring Boot + Tomcat) розгорнуто такі артефакти:
   - DlmApplication — головний клас застосунку, який ініціалізує Spring-контекст.
   - Controllers (Auth, Downloads, Settings, Home) — приймають HTTP-запити, відображають сторінки.
   - Services (User, Downloads, Settings) — реалізують бізнес-логіку, взаємодіючи з репозиторіями.
   - Repositories (UserRepo, DownloadRepo, SettingsRepo, StatsRepo, SegmentRepo) — здійснюють доступ до даних через Spring Data JPA.
   - Security (SecurityConfig, UserDetailsService, BCryptPasswordEncoder) — відповідає за автентифікацію та захист даних користувача.

3. Local DB (H2)
   У тій же системі зберігається вбудована база даних H2, що працює у file-mode.
   Дані користувачів, завантажень, сегментів і статистики зберігаються у таблицях:
   users, downloads, segments, stats, settings.
   Підключення здійснюється через JDBC, що вказано у властивостях проєкту:
4. jdbc:h2:file:...;AUTO_SERVER=TRUE

Взаємодія:
Користувач надсилає запити через браузер на сервер (HTTP/HTTPS).
Сервер обробляє запити в контролерах, викликає сервіси, які звертаються до репозиторіїв.
Репозиторії через JDBC працюють із локальною базою H2.
Результат обробки повертається користувачеві у вигляді HTML-сторінок.

Діаграма компонентів

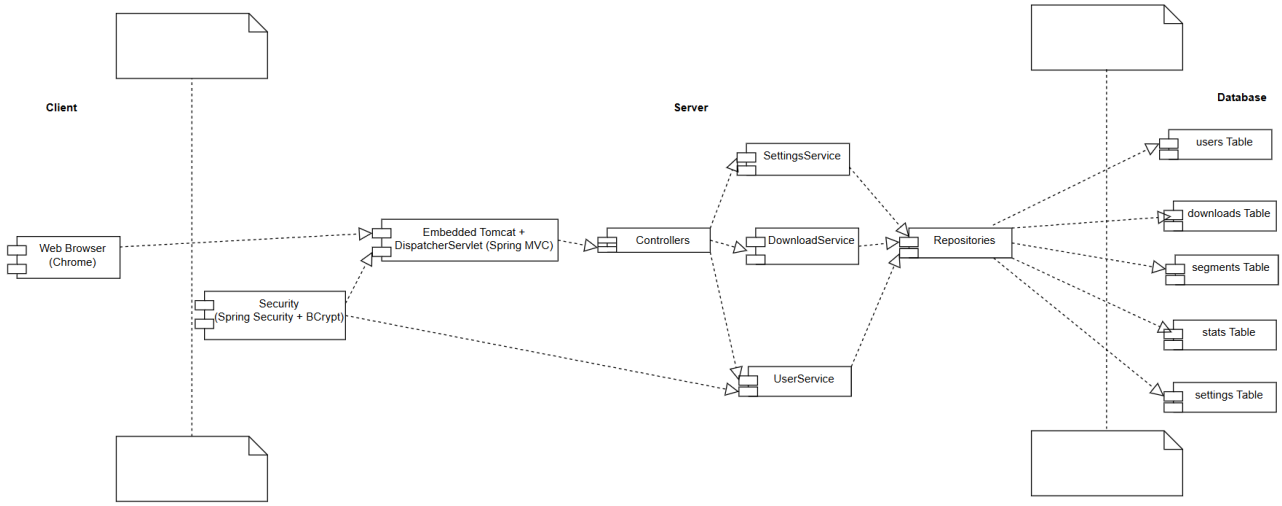

Рис. 2. Діаграма компонентів

Діаграма компонентів демонструє логічну структуру веб-застосунку. Вона показує основні компоненти системи, їхню взаємодію та розподіл між клієнтом, сервером і базою даних.

1. Client

Web Browser (Chrome) — інтерфейс користувача, який взаємодіє із сервером через HTTP/HTTPS.

Відправляє запити до застосунку та відображає згенеровані HTML-сторінки.

2. Server

Embedded Tomcat + DispatcherServlet (Spring MVC) — приймає HTTP-запити, виконує маршрутизацію до відповідних контролерів.

Controllers — реалізують логіку взаємодії з користувачем: обробляють реєстрацію, авторизацію, додавання та керування завантаженнями, зміну налаштувань.

Services — реалізують бізнес-логіку:

UserService — обробка користувачів і реєстрації;

DownloadService — додавання, оновлення статусів, видалення завантажень;

SettingsService — збереження індивідуальних параметрів користувача.

Repositories — інкапсулюють доступ до бази даних через ORM-механізм JPA, з’єднуючись через JDBC.

Security (Spring Security + BCrypt) — забезпечує авторизацію користувачів, фільтрацію запитів і хешування паролів.

3. Database

users — облікові дані користувачів;

downloads — інформація про завантаження;

segments — сегменти файлів для мультисегментного завантаження;

stats — статистика завантажень;

settings — індивідуальні налаштування користувача.

Взаємодія компонентів

1. Користувач взаємодіє із застосунком через веб-браузер, який надсилає HTTP-запити.
2. Вбудований сервер Tomcat приймає запит і передає його в DispatcherServlet.
3. Spring Security перевіряє автентифікацію користувача.
4. Контролери викликають відповідні сервіси, які реалізують бізнес-логіку.
5. Сервіси звертаються до репозиторіїв, які через JPA/JDBC взаємодіють із таблицями бази даних.
6. Результати повертаються через ланцюжок назад до браузера у вигляді HTML-сторінки.

Діаграми послідовностей

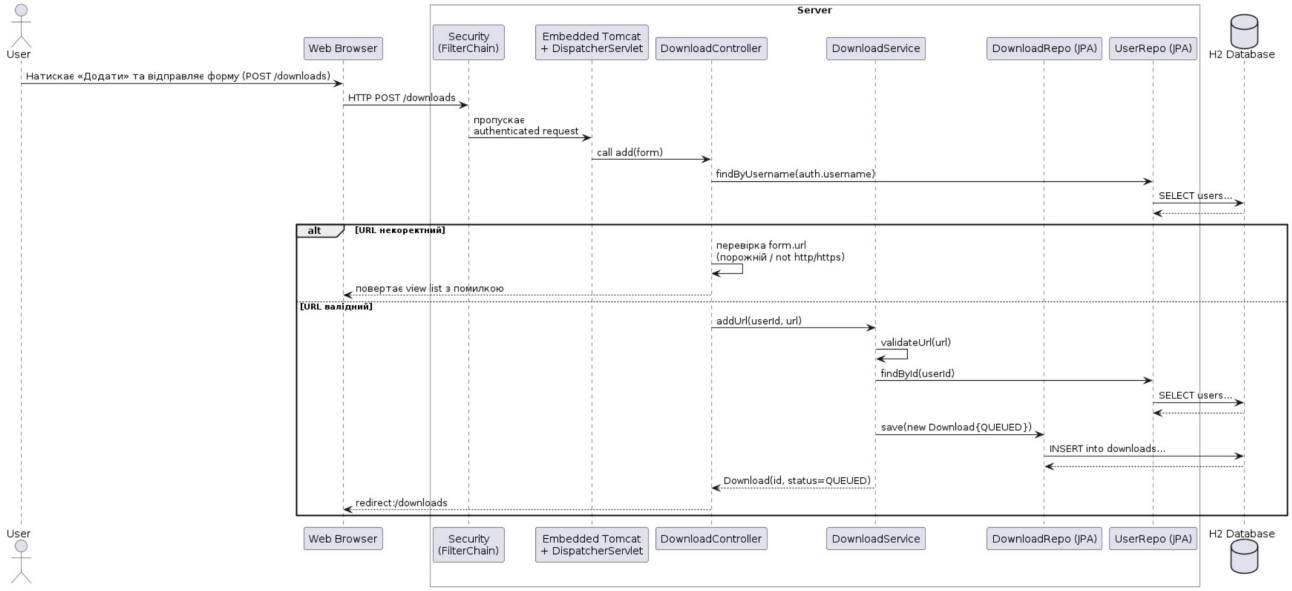

Рис 3. Діаграма послідовностей сценарію «Додати завантаження за URL»

Користувач у веб-браузері натискає кнопку «Додати» і відправляє форму з URL файлу.
Spring Security (FilterChain) перевіряє автентичність користувача та передає запит далі до DispatcherServlet.
DispatcherServlet маршрутизує запит до методу add() у DownloadController.
Контролер звертається до UserRepo, щоб знайти поточного користувача за його ім’ям.
Далі відбувається перевірка введеної адреси:

якщо URL порожній або має неправильний формат — система повертає сторінку зі списком завантажень і повідомленням про помилку;
якщо URL валідний, викликається метод addUrl() сервісу DownloadService.

DownloadService перевіряє правильність URL (validateUrl()), отримує користувача з UserRepo, створює новий об’єкт Download зі статусом QUEUED і передає його до DownloadRepo.
DownloadRepo (JPA) зберігає новий запис у таблиці downloads бази даних H2.

Після успішного збереження система перенаправляє користувача назад на сторінку /downloads, де у списку з’являється нове завантаження.

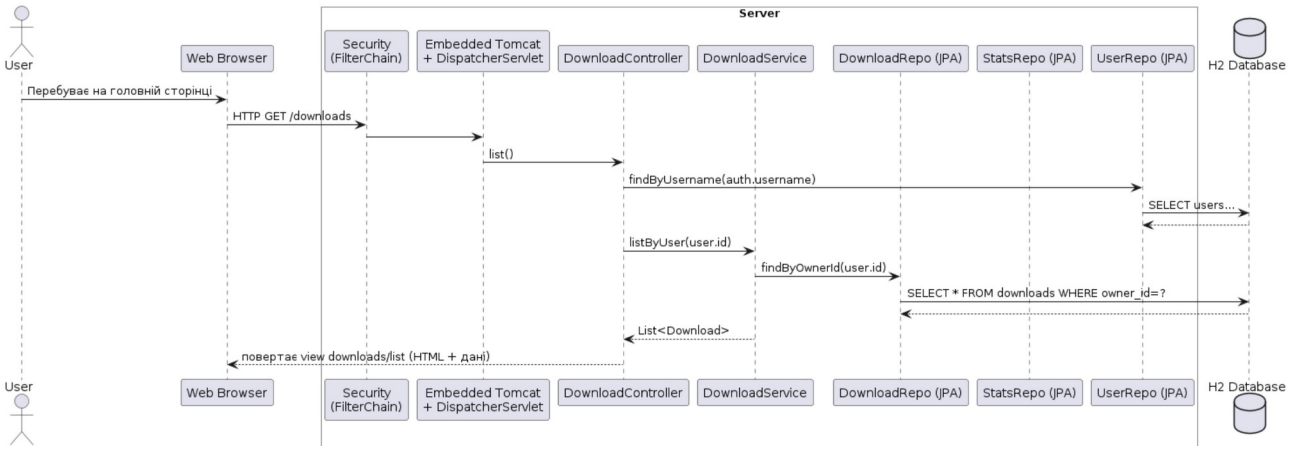

Рис 4. Діаграма послідовностей сценарію «Переглянути статистику завантажень»

Користувач у веб-браузері переходить на головну сторінку.
Браузер надсилає запит HTTP GET /downloads до сервера.
Spring Security (FilterChain) перевіряє автентифікацію користувача і пропускає запит далі до DispatcherServlet.
DispatcherServlet викликає метод list() у DownloadController.
Контролер звертається до UserRepo, щоб знайти поточного користувача за його логіном (findByUsername(auth.username)), і отримує відповідний запис із таблиці users у базі H2.
Потім контролер викликає метод listByUser(user.id) сервісу DownloadService, який звертається до DownloadRepo для отримання всіх завантажень поточного користувача (SELECT * FROM downloads WHERE owner_id=?).
Отримані дані про завантаження повертаються через DownloadService до DownloadController.
DownloadController формує веб-сторінку downloads/list, що містить HTML-таблицю з переліком завантажень та пов’язаними даними (URL, статус, прогрес, тощо).
Веб-браузер відображає сторінку користувачу.

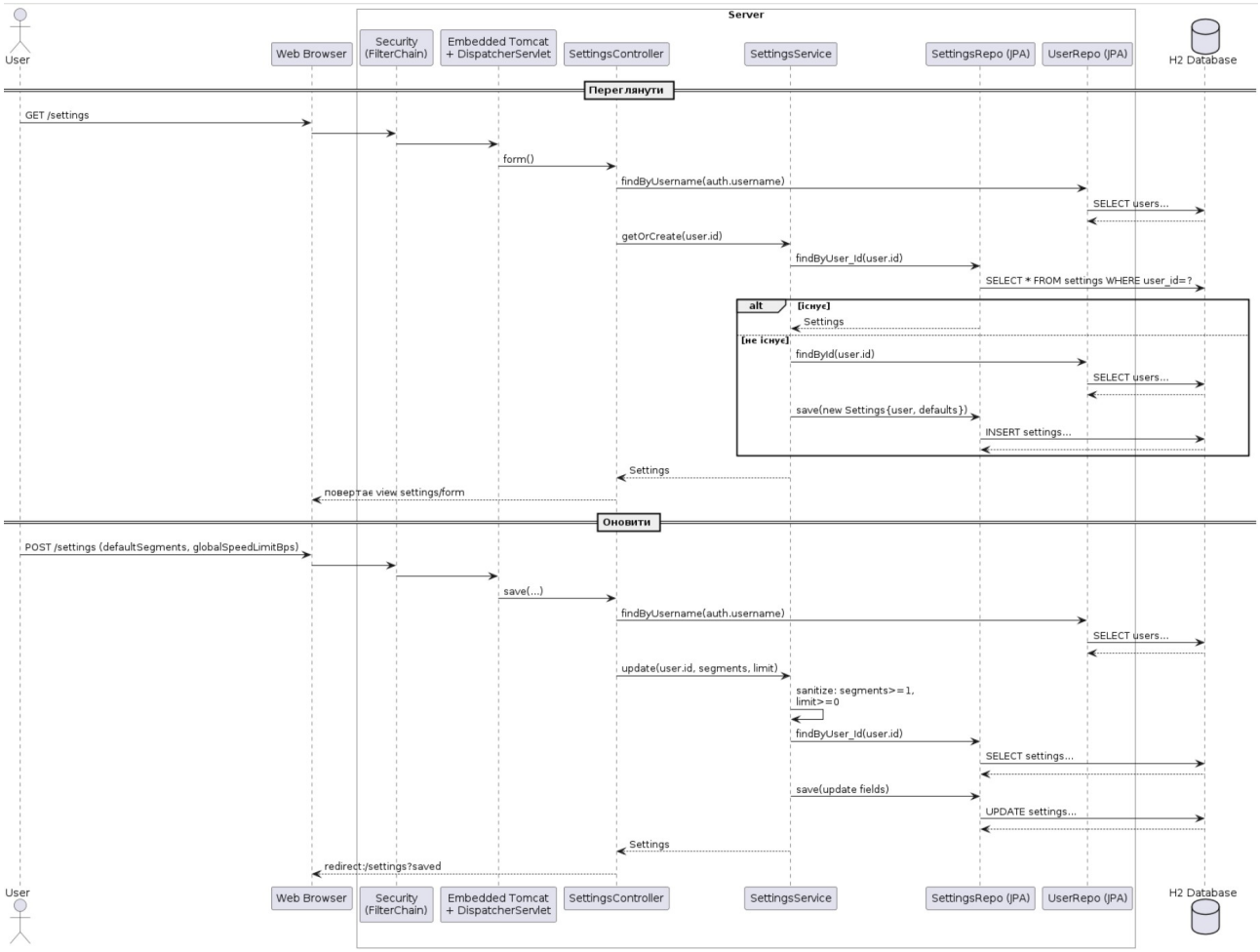

Рис 5. Діаграма послідовностей сценарію «Налаштувати параметри»

1. Перегляд поточних налаштувань
    1. Користувач у веб-браузері відкриває сторінку /settings.
    2. Відправляється запит GET /settings, який проходить через Spring Security (FilterChain) і потрапляє до DispatcherServlet.
    3. DispatcherServlet викликає метод form() у SettingsController.
    4. Контролер отримує поточного користувача через UserRepo (findByUsername(auth.username)).
    5. Потім викликається метод getOrCreate(user.id) сервісу SettingsService, який шукає налаштування користувача через SettingsRepo (findByUser_Id(user.id)).
    6. Якщо запис уже існує — він повертається у відповідь.
       Якщо запису немає, створюється новий об’єкт Settings із параметрами за замовчуванням і зберігається в базі даних (INSERT settings).
    7. Контролер повертає користувачу сторінку settings/form із поточними або новими параметрами.

2. Оновлення налаштувань
    1. Користувач змінює значення у формі (кількість сегментів, ліміт швидкості) й натискає кнопку «Зберегти».
    2. Відправляється запит POST /settings з новими параметрами.
    3. Spring Security пропускає запит, і DispatcherServlet викликає метод save() у SettingsController.

4. Контролер знову отримує користувача через UserRepo і викликає метод update(user.id, segments, limit) сервісу SettingsService.
5. Сервіс перевіряє коректність даних (segments >= 1, limit >= 0) і знаходить існуючий запис користувача (findByUser_Id(user.id)).
6. Поля оновлюються, і виконується команда UPDATE settings у базі даних.
7. Система повертає користувача на сторінку /settings?saved, де відображається повідомлення “Налаштування збережено”.

Розширена реалізація системи

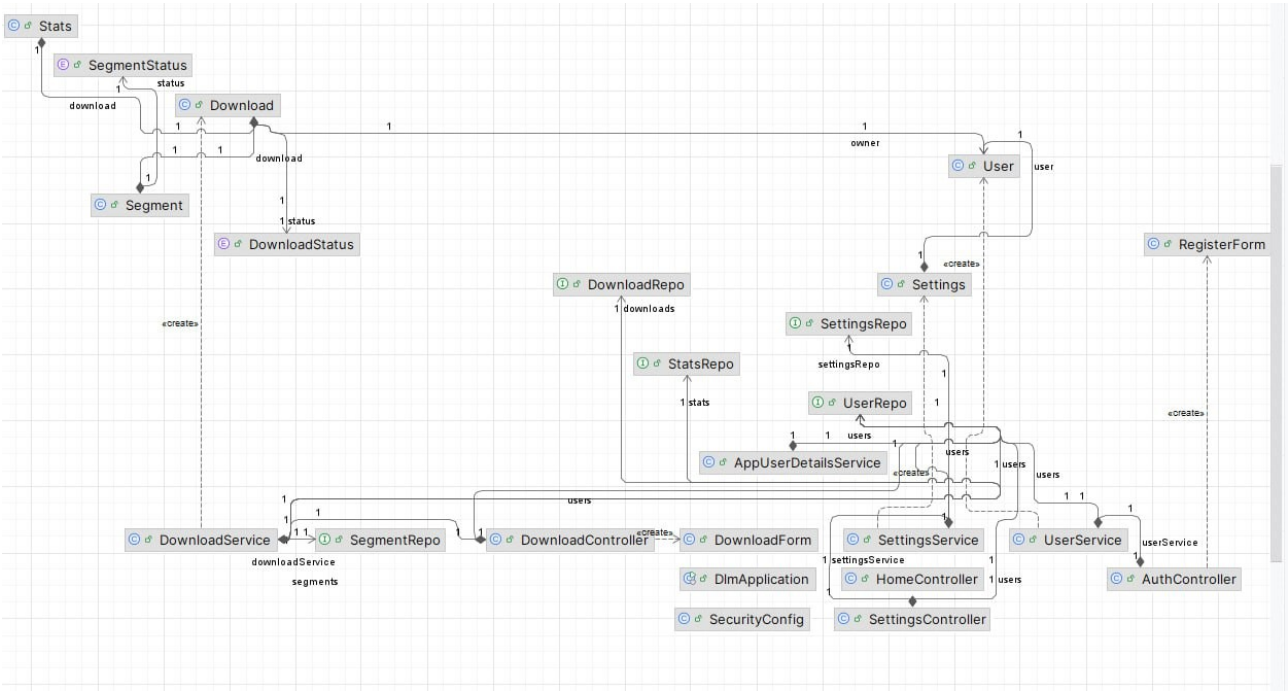

Рис. 6. Діаграма класів розширеної системи

У цій лабораторній роботі до діаграми класів було додано рівень бізнес-логіки та веб-інтерфейсу: сервіси (DownloadService, SettingsService, UserService), контролери (DownloadController, SettingsController, AuthController, HomeController) і компоненти безпеки (SecurityConfig, AppUserDetailsService).

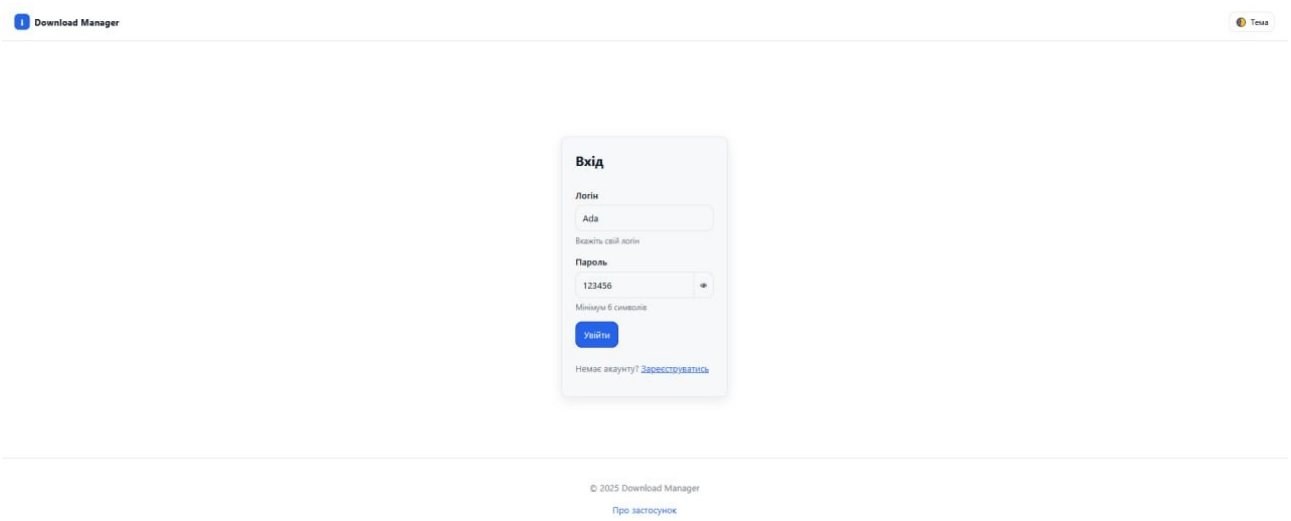

Рис. 7. Сторінка входу

Тут відбувається вхід користувача в систему.

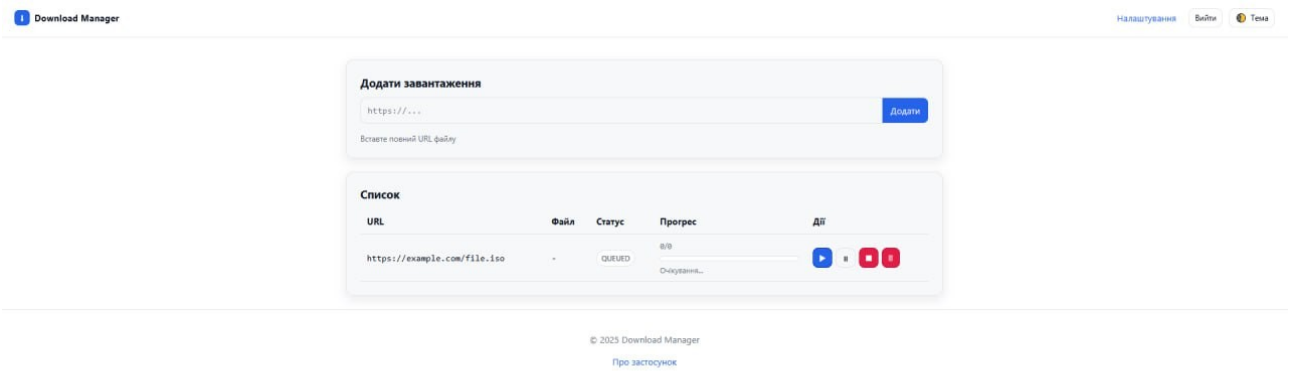

Рис. 8. Головна сторінка

Тут додається URL, а також відображається список доданих завантажень та дані про конкретне завантаження. Саме тут можна поставити завантаження на паузу, скасувати завантаження або повністю його видалити.

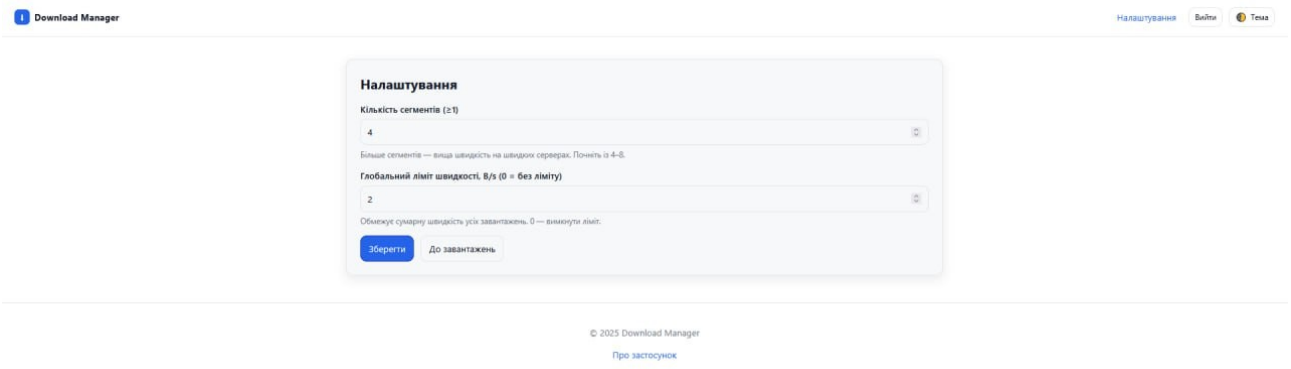

Рис. 9. Сторінка редагування налаштувань

Тут можна налаштовувати параметри завантажень.

Висновки

На основі спроєктованих діаграм розгортання та компонентів була доопрацювана програмна асистеми. Реалізація системи, додатково до попередньої реалізації, містить візуальні форми. В системі вже повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).

Контрольні питання

1. Що собою становить діаграма розгортання?

Це діаграма, яка показує фізичну структуру системи — на яких пристроях і середовищах виконуються компоненти програмного забезпечення.

2. Які бувають види вузлів на діаграмі розгортання?

Пристрій (Device) — фізичне обладнання (ПК, сервер, мобільний пристрій).

Середовище виконання (Execution Environment) — програмна платформа, у якій запускаються компоненти (ОС, вебсервер, JVM тощо).

3. Які бувають зв’язки на діаграмі розгортання?

Зв’язки між вузлами відображають канали комунікації (HTTP, TCP, IPC тощо) і можуть мати назви, що вказують технологію або протокол взаємодії.

4. Які елементи присутні на діаграмі компонентів?

Компоненти, залежності між ними, інтерфейси, а також виконувані файли або бібліотеки (.exe, .dll, .jar тощо).

5. Що становлять собою зв’язки на діаграмі компонентів?

Це залежності між компонентами, які показують, що один модуль використовує або викликає функції іншого.

6. Які бувають види діаграм взаємодії?

Діаграма послідовностей (Sequence Diagram)

Діаграма кооперації (Communication Diagram)

7. Для чого призначена діаграма послідовностей?

Для відображення взаємодії між об’єктами у часі, показуючи порядок викликів і обміну повідомленнями.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

Актори, об’єкти, лінії життя, повідомлення (виклики методів), активності, а також блоки alt (альтернатива) і loop (цикл).

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

Вони деталізують сценарії варіантів використання, показуючи покрокову взаємодію акторів і об’єктів системи.

10. Як діаграми послідовностей пов’язані з діаграмами класів?
Вони показують динамічну взаємодію об’єктів, створених із класів, зображених на діаграмі класів, і допомагають перевірити правильність зв’язків між ними.