Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського» Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота № 3

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування розгортання»

Тема проекту: «26. Download Manager»

Виконала:

студентка групи ІА-34

Мушта Анна

Перевірив:

Мягкий Михайло Юрійович

Дата здачі 11.10.2025

Захищено з балом

Київ 2025

Зміст

Тема: 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)

1.1. Теоретичні відомості

Діаграма розгортання (Deployment Diagram) описує фізичну конфігурацію системи: де саме запускається ПЗ і які середовища залучено. На ній відображаються вузли — як апаратні пристрої (ПК, сервери), так і середовища виконання (ОС, віртуальні машини, веб-сервери), між якими прокладені зв'язки з позначенням протоколів на кшталт HTTP чи внутрішнього IPC. До вузлів "прикріплюються" артефакти: виконувані файли, бібліотеки, скрипти, конфігурації, таблиці БД. Таку діаграму можуть будувати на концептуальному рівні, коли показують типові вузли без конкретного заліза, або як інстанс-схему з реальними машинами та конкретними артефактами.

Діаграма компонентів (Component Diagram) подає систему як композицію модулів із чіткими інтерфейсами та залежностями. Її використовують як для логічного огляду структури коду й меж відповідальності, так і для фіксації фізичних залежностей між артефактами на кшталт .exe/.dll/.jar, конфігурацій та ресурсів. У виконуваному варіанті така діаграма допомагає визначити, які саме модулі входять до складу збірки, як вони повторно використовуються, через які порти чи інтерфейси взаємодіють і як під'єднані до сховищ даних.

Діаграма послідовностей (Sequence Diagram) фокусується на динаміці — хто з ким і в якій черговості взаємодіє. На часовій осі розміщують акторів і об'єкти з "лініями життя", а повідомлення між ними зображаються стрілками викликів і повернень. Активності підсвічують періоди виконання операцій, а умовні та циклічні фрагменти (alt, loop) дозволяють моделювати варіативність сценаріїв. Така нотація зручна для уточнення бізнес-процесів, проєктування сервісних контрактів, прояснення відповідальностей між компонентами і підготовки тестових сценаріїв.

1.2. Хід роботи.

1. **Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи.**

- Use Case покриває ключові дії користувача: додати завантаження, керувати (пауза/відновлення/скасувати), перегляд статусу/статистики, розподіл швидкості, інтеграція з браузером.
- Класи вже розділені на шари «Domain/Application/Integration» з DownloadTask, DownloadService, політиками швидкості, ITaskStore (Repository), ProtocolResolver, BrowserIntegration.
- Тож для цієї ЛР достатньо «підняти» рівень архітектури до компонентів/розгортання та показати взаємодії у часі.

2. Діаграма розгортання

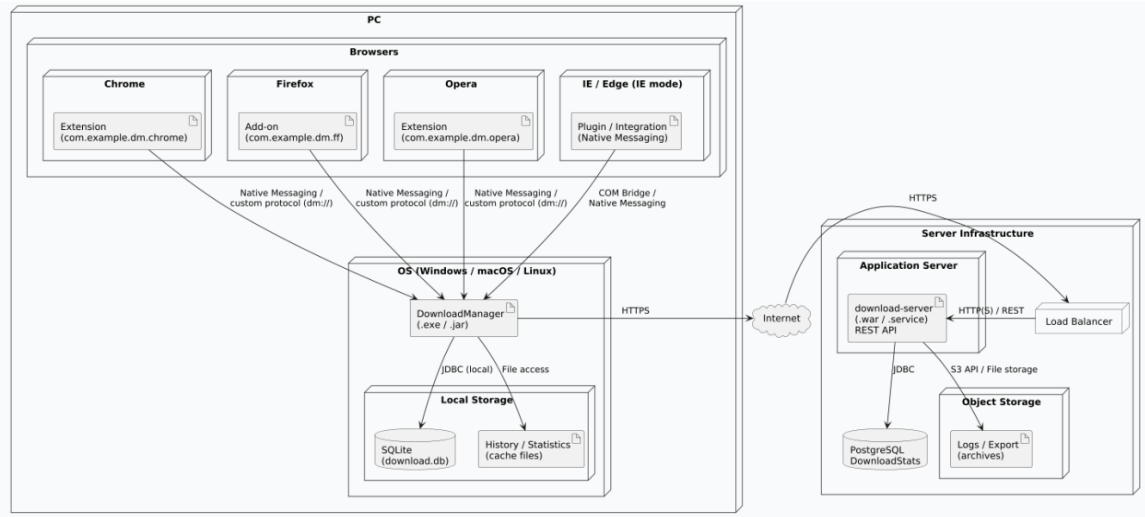


Рисунок 1 – Діаграма розгортання

На діаграмі показано фізичну архітектуру системи Download Manager, побудованої за принципом клієнт–серверної взаємодії з інтеграцією у веббраузери.

Клієнтська частина розгортається на персональному комп’ютері користувача під керуванням Windows / macOS / Linux.
Основний застосунок DownloadManager (.exe / .jar) відповідає за завантаження файлів, керування чергою, паузу/відновлення та ведення статистики.
Дані зберігаються у локальному сховищі, яке включає:

- базу SQLite (download.db) для збереження інформації про завантаження;
- файли історії та кешу для тимчасових даних і статистики.

Інтеграція з браузерами здійснюється через розширення:

- Chrome і Opera – через розширення (*Extension*);
- Firefox – через додаток (*Add-on*);
- Internet Explorer / Edge – через плагін із підтримкою Native Messaging / COM Bridge.
  Ці модулі передають посилання до застосунку за допомогою протоколу dm:// або через Native Messaging API.

Серверна інфраструктура використовується для синхронізації та аналітики. Вона містить:

- Балансувальник навантаження, що розподіляє запити клієнтів;
- Застосунковий сервер із компонентом *download-server (REST API)*;
- Базу даних PostgreSQL, де зберігаються статистичні дані;
- Об’єктне сховище, призначене для логів і архівів.

Обмін між клієнтом і сервером здійснюється через захищений протокол HTTPS, а внутрішні зв’язки серверів реалізовано через JDBC та S3 API

3. Діаграма компонентів

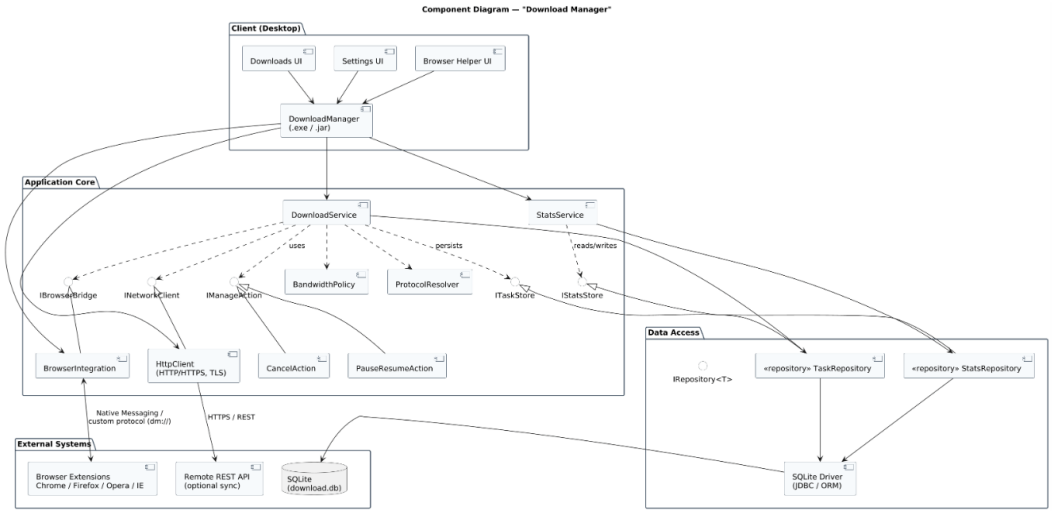


Рисунок 2 – Діаграма компонентів

На діаграмі компонентів представлено логічну архітектуру системи Download Manager, що складається з чотирьох основних частин: *Client (Desktop)*, *Application Core*, *Data Access* та *External Systems*.
Діаграма демонструє взаємозв'язки між модулями, інтерфейсами та зовнішніми компонентами, які забезпечують роботу системи.

Клієнтська частина (Client) містить основний застосунок DownloadManager (.exe/.jar) та інтерфейсні модулі Downloads UI, Settings UI і Browser Helper UI, що забезпечують взаємодію користувача із системою. Вони викликають методи ядра програми для керування завантаженнями, налаштувань і відображення статистики.

Ядро застосунку (Application Core) реалізує головну бізнес-логіку системи. Основні сервіси:

- DownloadService — керує створенням, запуском і контролем завантажень;
- StatsService — опрацьовує статистику завантажень;
- ProtocolResolver — визначає типи протоколів (HTTP, HTTPS) для файлів;
- BandwidthPolicy — регулює швидкість завантаження;
- BrowserIntegration — приймає запити від браузерних розширень через Native Messaging або власний протокол *dm://*;
- HttpClient — відповідає за мережеву взаємодію з віддаленим сервером через HTTPS/REST API.

Додаткові дії, такі як PauseResumeAction та CancelAction, реалізують інтерфейс IManageAction, забезпечуючи гнучке керування чергою завантажень.

Рівень доступу до даних (Data Access) представлено компонентами TaskRepository і StatsRepository, які реалізують відповідно інтерфейси ITaskStore та IStatsStore.
Обидва репозиторії використовують SQLite Driver (JDBC/ORM) для збереження даних у локальній базі SQLite (download.db).

Зовнішні системи (External Systems) включають:

- Browser Extensions (Chrome, Firefox, Opera, IE) — надсилають посилання в клієнтський застосунок;
- Remote REST API — забезпечує опціональну синхронізацію статистики на сервері;
- SQLite DB — основне локальне сховище даних.

Компоненти взаємодіють через визначені інтерфейси (IBrowserBridge, INetworkClient, ITaskStore, IStatsStore), що забезпечує модульність і розширюваність системи.
Таким чином, діаграма демонструє багаторівневу структуру застосунку, де кожен шар має чітко визначену відповідальність і мінімальні залежності від інших.

3. Діаграма розгортання

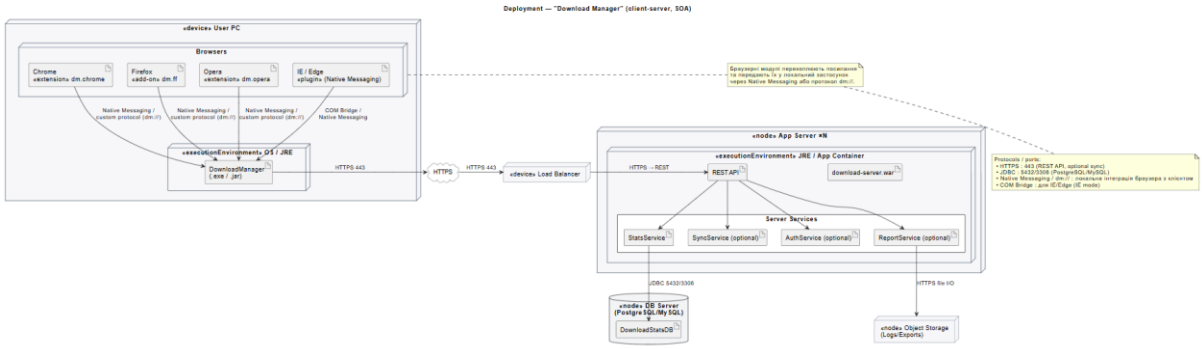


Рисунок 3 – Діаграма розгортання системи “Download Manager”

На діаграмі розгортання показано фізичну архітектуру системи Download Manager, яка працює за принципом клієнт–серверної взаємодії з підтримкою інтеграції у браузери.

Клієнтська частина розгортається на персональному комп’ютері користувача (User PC) під керуванням операційної системи (OS / JRE). Основний застосунок DownloadManager (.exe / .jar) забезпечує функції завантаження файлів, керування чергою, паузи/відновлення та збереження історії.
До нього під’єднуються браузерні модулі (Chrome, Firefox, Opera, IE/Edge), що передають посилання на завантаження через Native Messaging або спеціальний протокол dm://. Для браузера IE/Edge використовується COM Bridge.

З’єднання між клієнтом і сервером здійснюється через захищений протокол HTTPS (порт 443). Запити користувача передаються через Load Balancer до серверної частини.

Серверна частина складається з App Server ×N, у якому розгорнуто застосунок download-server.war в середовищі JRE / App Container.
Основні сервіси:

- StatsService — збір і обробка статистичних даних;
- SyncService — опціональна синхронізація даних користувача;
- AuthService — автентифікація при підключенні до серверу;
- ReportService — формування звітів та експорт статистики.

Дані зберігаються у базі DownloadStatsDB (PostgreSQL або MySQL), доступ до якої здійснюється через JDBC (порти 5432/3306).
Результати експорту та журнали операцій зберігаються в Object Storage (Logs/Exports) з використанням HTTPS file I/O.

Протоколи взаємодії:

- HTTPS : 443 — REST API, синхронізація;
- JDBC : 5432/3306 — робота з базою даних;
- Native Messaging / dm:// — інтеграція браузера з клієнтом;
- COM Bridge — взаємодія з IE/Edge.

Таким чином, діаграма відображає розподіл компонентів системи між клієнтом і сервером та їх взаємозв’язки, що забезпечують надійну й масштабовану роботу системи “Download Manager”.

4. Сценарії

1) Додати завантаження з браузера (через розширення)

Мета: створити задачу завантаження з URL, переданого з браузера, і розпочати скачування.
Актори: Користувач (основний), Browser Extension (вторинний).
Система: DownloadManager (UI + DownloadService + ProtocolResolver + HttpClient + TaskRepository/SQLite).

Передумови

- Розширення браузера встановлене та пов’язане з десктопним застосунком (Native Messaging або dm://).
- Локальна БД SQLite (download.db) доступна.
- Є вільне місце на диску для збереження файлу.

Постумови

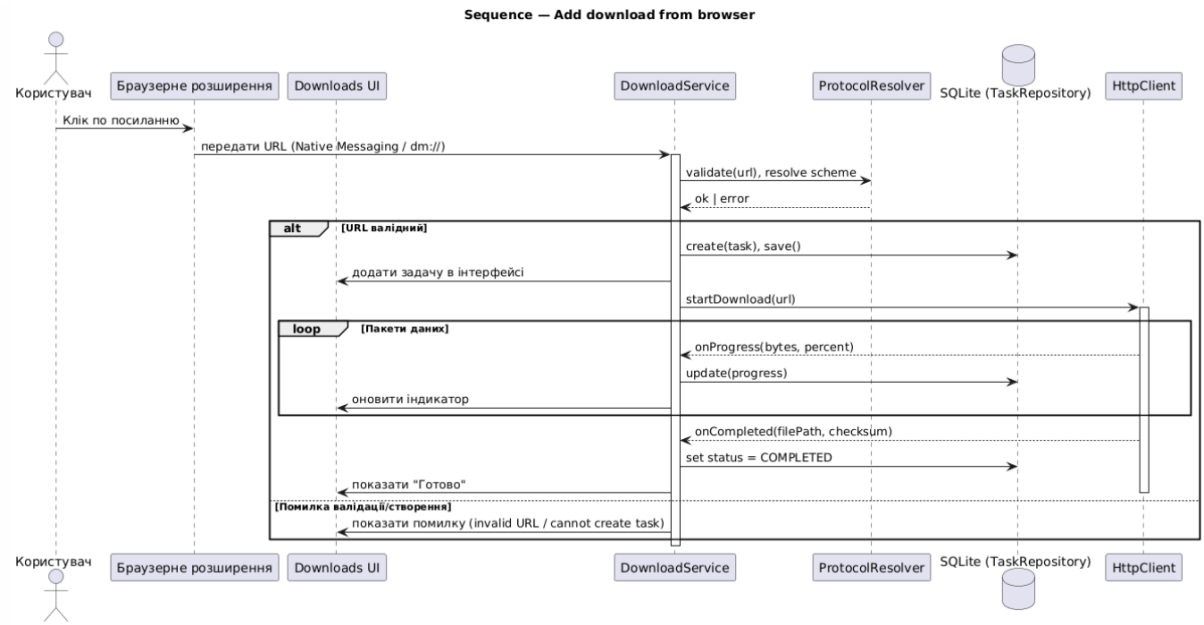
- Створено запис задачі в TaskRepository.
- Задача має статус RUNNING або COMPLETED (якщо файл дуже малий).
- Прогрес відображається в Downloads UI, історія/статистика оновлені.

Основний потік

1. Користувач клацає по посиланню на файл у браузері.
2. Browser Extension перехоплює URL і надсилає його в DownloadManager (Native Messaging / dm://).
3. DownloadService отримує URL, звертається до ProtocolResolver для валідації та визначення схеми (HTTP/HTTPS).
4. Якщо валідно — TaskRepository створює новий запис задачі (статус NEW, шлях збереження, 0 байт).
5. DownloadService додає задачу в чергу й повідомляє Downloads UI про новий елемент.
6. DownloadService → HttpClient: старт завантаження через HTTPS.
7. Цикл прогресу: HttpClient повідомляє про отримані байти → DownloadService оновлює TaskRepository (відсоток/байти) і Downloads UI.
8. По завершенні — HttpClient повертає результат (шлях до файла, хеш/розмір), статус задачі змінюється на COMPLETED, UI показує “Готово”.

Альтернативні та виняткові потоки

- A1. Невалідний URL / непідтримувана схема
  3a. ProtocolResolver повертає помилку → UI показує повідомлення “Некоректне посилання / протокол не підтримується”, задача не створюється.
- A2. Помилка створення задачі (БД)
  4a. TaskRepository недоступний → UI показує “Помилка збереження задачі”, лог у Journal.



2) Пауза і відновлення завантаження (Resume з останньої позиції)

Мета: зупинити активне завантаження з фіксацією останнього байта та відновити його з використанням HTTP Range.
Актори: Користувач.
Система: Downloads UI, DownloadService, BandwidthPolicy, HttpClient, TaskRepository/SQLite.

Передумови

- Існує активна задача зі статусом RUNNING.
- Сервер потенційно підтримує HTTP Range (не обов’язково).

Постумови

- Після “Пауза” — задача має статус PAUSED, у БД збережено lastByte.
- Після “Відновити” — задача або повертається в RUNNING/COMPLETED, або в ERROR (у разі збоїв).

Основний потік (Пауза)

1. Користувач у Downloads UI натискає “Пауза”.
2. UI → DownloadService: pause(taskId).
3. DownloadService → HttpClient: stop() — акуратно розриває з’єднання.
4. HttpClient → DownloadService повертає lastByte.
5. DownloadService → TaskRepository: зберегти status=PAUSED, lastByte.
6. UI відображає статус PAUSED.

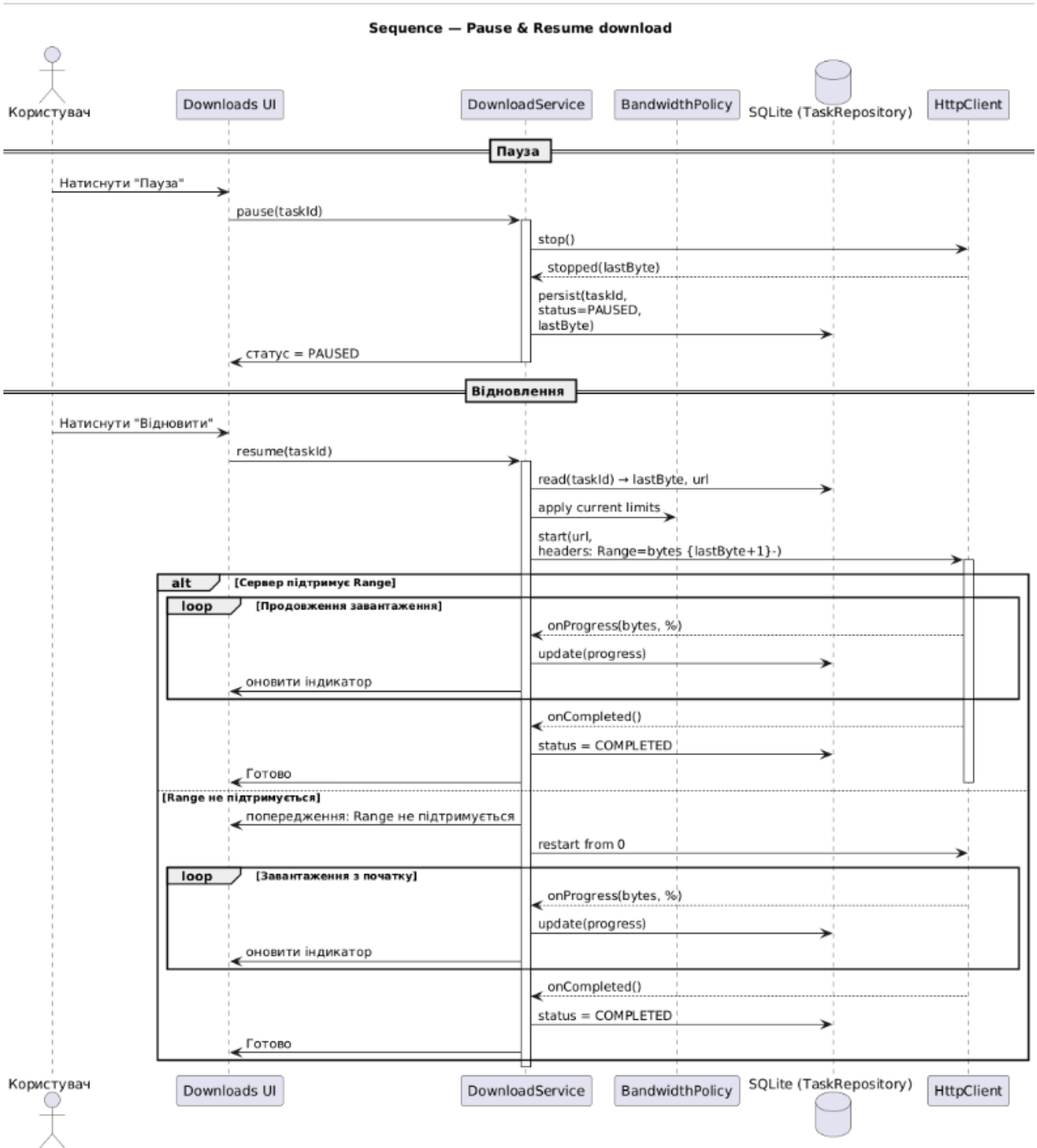
Основний потік (Відновлення)

7. Користувач натискає “Відновити”.
8. UI → DownloadService: resume(taskId).
9. DownloadService → TaskRepository: прочитати url, lastByte.
10. DownloadService → BandwidthPolicy: застосувати поточні обмеження (за потреби).
11. DownloadService → HttpClient: старт з Range=bytes {lastByte+1}-.
12. Цикл прогресу: HttpClient надсилає onProgress → DownloadService оновлює TaskRepository і UI.
13. Після onCompleted — статус COMPLETED, UI показує “Готово”.

Альтернативні та виняткові потоки

- B1. Сервер не підтримує Range
  11a. Відповідь без Accept-Ranges або код 200 замість 206.
  11b. DownloadService показує попередження й виконує fallback: перезапустити завантаження з нуля (з підтвердженням користувача або автоматично).

- B2. Файл було змінено на сервері
  11c. Змінився ETag/Last-Modified → система пропонує перезапустити з початку або зберегти окремо (уникнути пошкодження).
- B3. Недоступна БД
  9a/12a. Помилка запису прогресу → запис у журнал, статус RUNNING, UI показує попередження “Проблема з історією”, завантаження не переривається.



5. Реалізація системи

- Додавання завантаження (сценарій 1).
  Команда: add http://speed.hetzner.de/10MB.bin C:\Temp\10MB.bin.
  Результат list: створено задачу #13, статус COMPLETED — файл успішно завантажено в C:\Temp\10MB.bin (показано фактичний/очікуваний розмір).

```
> add http://speed.hetzner.de/10MB.bin C:\Temp\10MB.bin
Task created: #13
> list
#13 [COMPLETED] http://speed.hetzner.de/10MB.bin (1059/1059) -> C:\Temp\10MB.bin
```

- Створення великого завантаження та обмеження швидкості.
  Команди: add http://speed.hetzner.de/100MB.bin C:\Temp\100MB.bin, потім limit 200000.
  Результат list: задача #14 з’явилась і завершилась (для демонстрації швидкість було обмежено до ~200 КБ/с).

```
> add http://speed.hetzner.de/100MB.bin C:\Temp\100MB.bin
Task created: #14
> limit 200000
Limit set to 200000 B/s (0 = unlimited)
> list
#14 [COMPLETED] http://speed.hetzner.de/100MB.bin (1059/1059) -> C:\Temp\100MB.bin
```

- Пауза завантаження (сценарій 2).
  Команда: pause 14.
  Результат list: задача #14 переходить у статус PAUSED (збережено поточний прогрес — відновлення можливе з останнього байта).

```
> pause 14
Paused #14
> list
#14 [PAUSED] http://speed.hetzner.de/100MB.bin (1059/1059) -> C:\Temp\100MB.bin
```

- Відновлення завантаження (сценарій 2).
  Команда: resume 14.
  Результат list: задача #14 завершується зі статусом COMPLETED — підтверджує коректну роботу механізму *Pause/Resume* із докачкою.

```
> resume 14
Resumed #14
> list
#14 [COMPLETED] http://speed.hetzner.de/100MB.bin (1059/1059) -> C:\Temp\100MB.bin
```

1.3. Висновки.

У ході роботи я спроєктувала спрощений Download Manager: побудувала Use Case-діаграму з одним актором і зв’язками include та узагальненням, узгодила її з діаграмою класів і перетворила модель на структуру бази даних із кількох таблиць та чіткими зв’язками. Реалізувала основні класи домену та сервіс із шаблоном Repository для взаємодії з БД, охопивши ключові вимоги теми: додавання, пауза/відновлення, статуси, базова статистика, керування швидкістю та інтеграція з браузера. Я навчилася формулювати варіанти використання й переносити їх у об’єктну модель, задавати індекси й зовнішні ключі, а також користуватися PlantUML (зокрема правильними коментарями, параметрами оформлення та нотаціями include/generalization).