Національний технічний університет України

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

Кафедра інформаційних систем та технологій

Лабораторна робота № 2

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування»

Тема проекту: «26. Download Manager»

Виконав:

студент групи ІА-34

Мушта Анна

Перевірив:

Мягкий Михайло Юрійович

Дата здачі 20.09.2025

Захищено з балом

Київ 2025

Зміст

1.1. Теоретичні відомості

2.2.1 Вступ до UML.

UML — універсальна мова візуального моделювання для специфікації, візуалізації, проєктування та документування систем. Дозволяє описувати систему на різних рівнях абстракції (концептуальний, логічний, фізичний) через узгоджені «уявлення» (views) у вигляді діаграм.

2.2.2 Діаграма варіантів використання.

Відправна концептуальна модель вимог: показує межі системи, акторів і які послуги (use cases) система їм надає. Використовується для збору/узгодження вимог і як база для подальших діаграм.

2.2.3 Актори.

Зовнішні до системи ролі (людина/інша система/пристрій), що взаємодіють із системою для досягнення своїх цілей. Імена ролей мають бути зрозумілими предметній області.

2.2.4 Варіанти використання.

Описують послуги системи з точки зору актора (послідовність дій/результат), без деталей реалізації. Ім’я має передавати результат або дію.

2.2.5 Відносини на Use Case.

- Асоціація (зв’язок актор—UC, може бути спрямованою);
- Узагальнення (наслідування між акторами або UC);
- Include (обов’язкове підвикликання спільної поведінки);

- Extend (умовне розширення базового сценарію в точці розширення).

2.2.6 Сценарії використання.

Текстова специфікація UC: передумови, постумови, сторони, короткий опис, основний потік, винятки, примітки. Робить вимоги однозначними та придатними до реалізації/тестування.

2.2.7 Діаграми класів.

Статичний опис структури: класи, атрибути, операції, інтерфейси, видимість, а також зв’язки — асоціації (з множинністю), узагальнення, агрегація/композиція (частина–ціле з різною «щільністю» зв’язку).

2.2.8 Логічна структура БД.

Логічна модель (таблиці, ключі, індекси, зв’язки) відображає зберігання даних незалежно від фізичної реалізації. Нормалізація (1НФ, 2НФ, 3НФ, БКНФ) зменшує надмірність і суперечності.

2.2.9 Проєктування БД.

На основі моделі (класи/зв’язки) створюються таблиці, первинні/зовнішні ключі, індекси; обирається підхід відображення «класи ↔ таблиці». У СУБД інструментах задаються структури й зв’язки (Relationships).

1.2. Хід роботи.

1. Проаналізувати тему та спроєктувати діаграму варіантів використання відповідно до обраної теми лабораторного циклу.

Код:

```
@startuml
left to right direction
skinparam usecase {
  BackgroundColor #FFFFFF
  BorderColor #333333
}
titleUse Case

actor "Користувач" as User

rectangle "Download Manager" {
  usecase "Додати завантаження" as UC_Add
  usecase "Переглянути статус" as UC_Status
  usecase "Керувати завантаженням" as UC_Manage
  usecase "Пауза / Відновлення" as UC_PauseResume
  usecase "Скасувати завантаження" as UC_Cancel
  usecase "Налаштувати розподіл швидкості" as UC_BW
  usecase "Переглянути статистику" as UC_Stats
  usecase "Інтеграція з браузером" as UC_Browser
}

' Зв'язки з користувачем
User --> UC_Add
User --> UC_Status
User --> UC_Manage
User --> UC_BW
User --> UC_Stats
User --> UC_Browser

' Include: додати завантаження завжди включає перегляд статусу
UC_Add ..> UC_Status : <<include>>

UC_Status ..> UC_Stats : <<include>>

UC_Browser ..> UC_Add : <<include>>
' Generalization: керування завантаженням узагальнює паузу/відновлення та скасування
UC_PauseResume -|> UC_Manage
UC_Cancel -|> UC_Manage
@enduml
```

Діаграма:

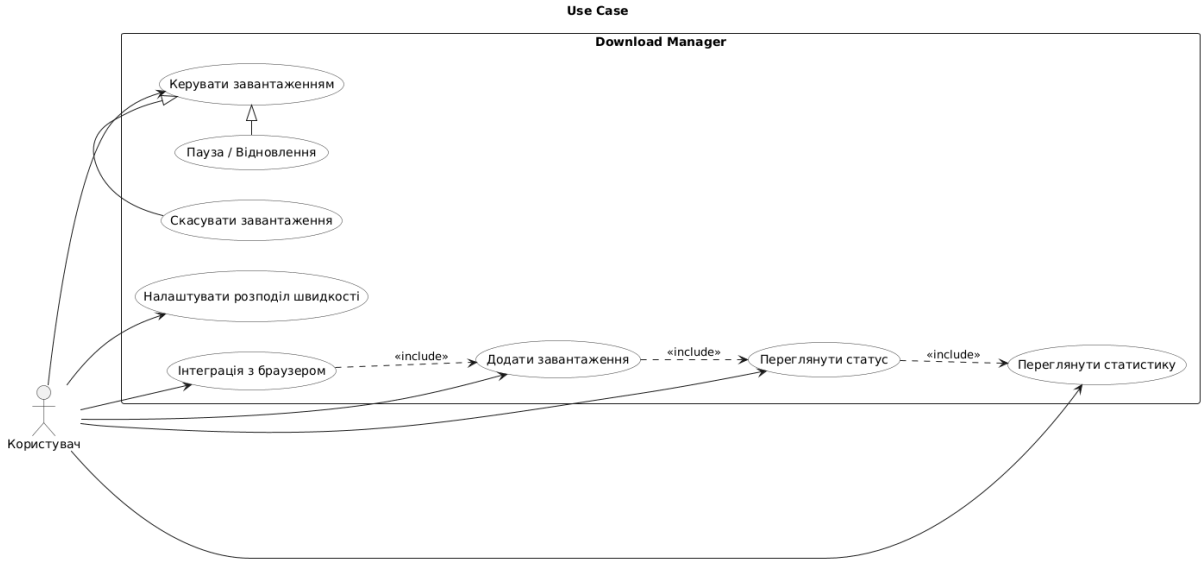

Рисунок 1 – Use case діаграма для користувача

2. Спроєктувати діаграму класів предметної області.

Код:

```
@startuml
title Class Diagram

skinparam class {
  BackgroundColor #FFFFFF
  BorderColor #333333
}
skinparam packageStyle rectangle

package "Domain" {
  class DownloadTask {
    +Id: Guid
    +Url: string
    +FileName: string
    +SavePath: string
```

```
    +Status: DownloadStatus
    +TotalSize: long
    +BytesDownloaded: long

    ' --- Статистика ---
    +StartedAt: DateTime
    +CompletedAt: DateTime
    +AverageSpeedKbps: double
    +Retries: int
    +SourceApp: string   ' "firefox", "chrome", "opera", "ie" або ""
  }

  enum DownloadStatus {
    Queued
    InProgress
    Paused
    Completed
    Error
    Canceled
  }
}

package "Application" {
  class DownloadService {
    +Add(url: string, savePath: string): Guid
    +GetStatus(id: Guid): DownloadStatus
    +List(): List<DownloadTask>
    +Manage(id: Guid, action: IManageAction): void
  }
```

```
interface ITaskStore {
  +Add(task: DownloadTask): void
  +Update(task: DownloadTask): void
  +Get(id: Guid): DownloadTask
  +List(): List<DownloadTask>
}

' Узагальнення керуючих дій
interface IManageAction {
  +Execute(task: DownloadTask): void
}
class PauseResumeAction {
  +Pause(task: DownloadTask): void
  +Resume(task: DownloadTask): void
  +Execute(task: DownloadTask): void
}
class CancelAction {
  +Execute(task: DownloadTask): void
}
PauseResumeAction -|> IManageAction
CancelAction -|> IManageAction

' --- Політика розподілу швидкості ---
class BandwidthPolicy {
  +TotalLimitKbps: int
  +PerTaskLimitKbps: int
  +Apply(activeTasks: List<DownloadTask>): void
}
```

```
  DownloadService ..> BandwidthPolicy : використовує
  DownloadService ..> ITaskStore : зчитує/зберігає
  DownloadService ..> IManageAction : викликає
}

package "Integration" {
  ' --- Інтеграція з браузером ---
  class BrowserIntegration {
    +RegisterExtensions(): void
    +HandleBrowserLink(url: string, sourceApp: string): void
  }

  BrowserIntegration ..> DownloadService : додає через Add(url, ...)
}
' Сховище оперує DownloadTask
ITaskStore o-- DownloadTask
@enduml
```

Діаграма:

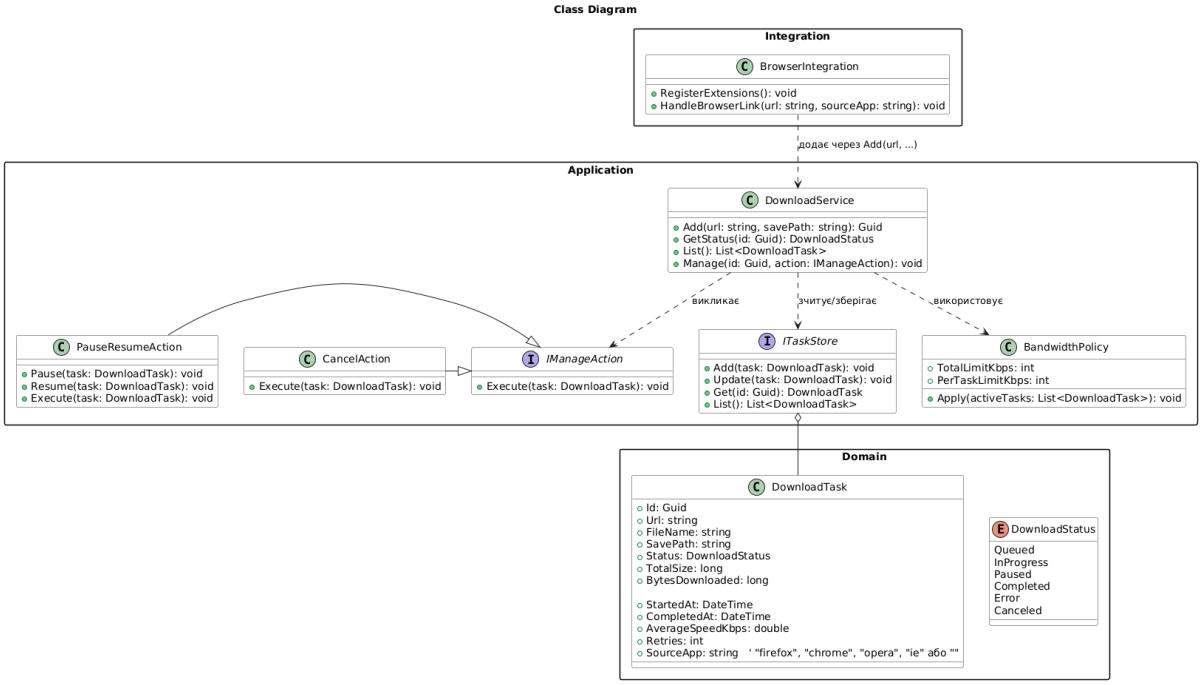

Рисунок 2 –Діаграма класів

3. Вибрати 3 варіанти використання та написати за ними сценарії використання.

Сценарій 1: Додати завантаження

Передумови:

- Застосунок “Download Manager” запущено.
- Є стабільне підключення до інтернету.
- Налаштовано теку збереження за замовчуванням (або користувач її знає).

Постумови:

- У системі створено нове завантаження зі статусом Queued або InProgress.
- Запис з’являється у списку статусів; починається збір статистики (час старту, швидкість, кількість спроб).

Взаємодіючі сторони:
Користувач, Система.

Короткий опис:
Користувач додає нове завантаження за URL (http/https); система перевіряє доступність і запускає процес.

Основний потік подій:

1. Користувач обирає функцію “Додати завантаження”.
2. Система пропонує ввести: URL та (за потреби) теку збереження/назву файлу.
3. Користувач вводить URL (можливо, вставлений із буфера або надісланий із браузера).
4. Система виконує попередню перевірку (HEAD/GET): доступність ресурсу, розмір, підтримка Range.
5. Система створює запис DownloadTask, додає його до списку та запускає завантаження або ставить у чергу.
6. Система оновлює статус і показує елемент у списку завантажень.

Винятки:

- Некоректний або порожній URL -> система просить виправити.
- Сервер недоступний -> статус Error, пропозиція Повторити.
- Недостатньо вільного місця/немає доступу до теки -> система повідомляє й пропонує обрати іншу теку.
- Сервер не підтримує Range -> завантаження можливе лише з початку (система попереджає).

Примітки:
Може бути виклик “із браузера” (розширення/меню «Завантажити через Download Manager»); система фіксує поле джерела (наприклад, *chrome/firefox/opera*).

Сценарій 2: Пауза / Відновлення

Передумови:

- Існує активне завантаження зі статусом InProgress.
- Для коректного продовження бажано, щоб сервер підтримував Range -часткові запити.

Постумови:

- Після “Пауза” завантаження переходить у Paused, збережено значення BytesDownloaded.
- Після “Відновлення” завантаження переходить у InProgress і продовжується з позиції зупинки.

Взаємодіючі сторони:
Користувач, Система.

Короткий опис:
Користувач тимчасово зупиняє завантаження і згодом відновлює його без повторного скачування вже отриманих даних.

Основний потік подій:

1. Користувач у списку обирає завантаження і натискає “Пауза”.
2. Система коректно зупиняє поточні операції та фіксує BytesDownloaded.
3. Статус завантаження змінюється на Paused.
4. Коли користувач натискає “Відновити”, система формує запит із заголовком Range від збереженої позиції.
5. Завантаження триває, система оновлює статус і статистику (середню швидкість, час).

Винятки:

- Сервер не підтримує Range -> система попереджає, що відновлення можливе лише з початку.
- Раптовий обрив мережі під час відновлення -> статус Error або Paused, пропозиція повторити.
- Файл заблоковано іншою програмою -> система просить закрити доступ/обрати інший шлях збереження.

Примітки:
Статистика (середня швидкість, час у паузі, кількість спроб) оновлюється автоматично та відображається у вікні статусу.

Сценарій 3: Налаштувати розподіл швидкості

Передумови:

- Застосунок запущено, наявні активні або заплановані завантаження.

Постумови:

- Застосовано нові обмеження швидкості (загальне/на завдання), активні завантаження підлаштувалися під політику.
- Налаштування збережено для наступних сеансів.

Взаємодіючі сторони:
Користувач, Система.

Короткий опис:
Користувач встановлює правила розподілу пропускної спроможності між активними завантаженнями (наприклад, загальний ліміт та ліміт на кожне завдання).

Основний потік подій:

1. Користувач відкриває “Налаштування” -> “Швидкість завантаження”.
2. Система показує поля для введення: “Загальний ліміт” та “Ліміт на завдання”.
3. Користувач вводить значення та натискає “Застосувати”.
4. Система перераховує активні швидкості згідно (наприклад, не перевищувати загальний ліміт і ліміт на завдання).
5. Система зберігає налаштування та повідомляє про успішне застосування.

Винятки:

- Введено нуль/негативне значення -> система просить ввести коректний ліміт.
- “Ліміт на завдання” перевищує “Загальний ліміт” -> система пропонує скоригувати параметри.

Примітки:
Просте правило розподілу — рівний розподіл у межах загального ліміту з верхньою межею на завдання.

4. На основі спроєктованої діаграми класів предметної області розробити основні класи та структуру бази даних системи. Класи даних повинні реалізувати шаблон Repository для взаємодії з базою даних.

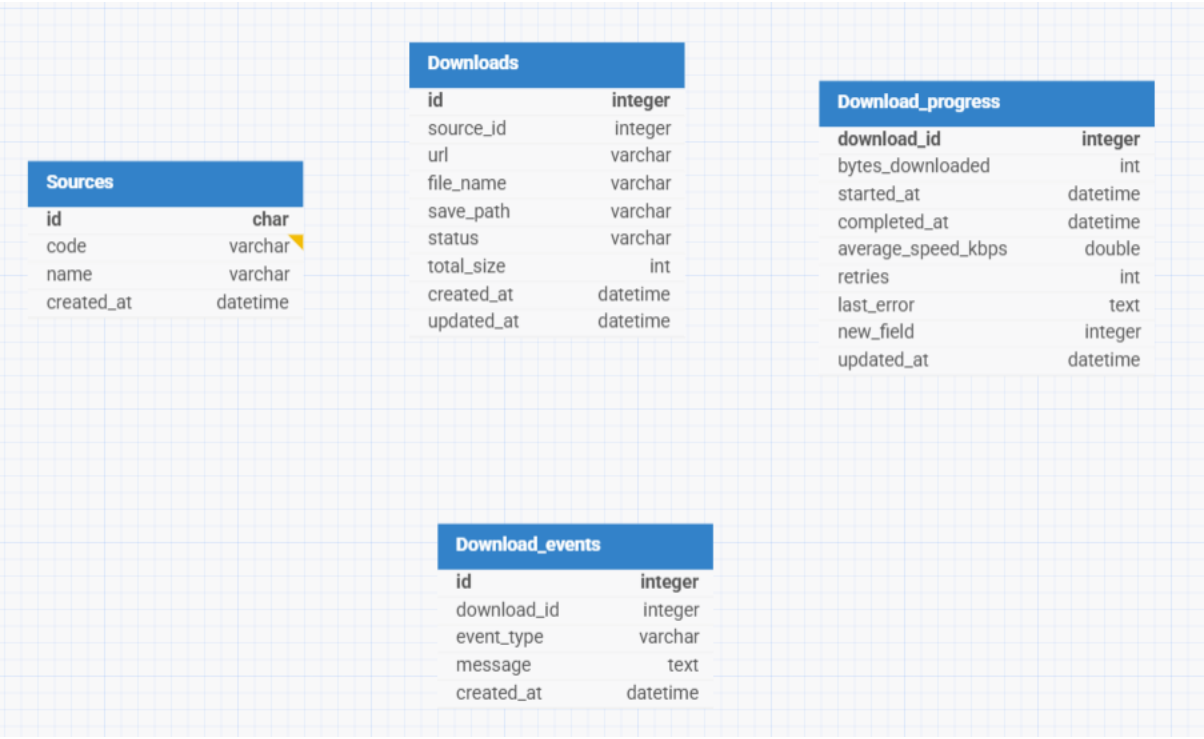

Рисунок 3 – Основні класи бази даних

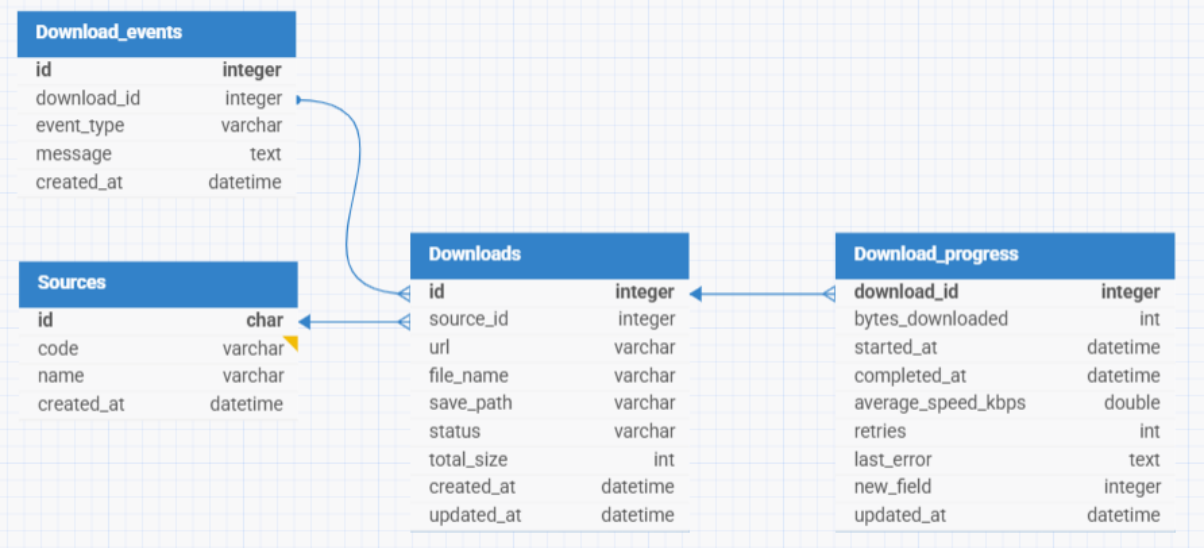

Рисунок 4 – структура бази даних системи

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).