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»

Виконала:

студентка групи ІА-34

Мушта Анна

Перевірив:

Мягкий Михайло Юрійович

Дата здачі 20.09.2025

Захищено з балом

Київ 2025

Зміст

Тема 3

1.1. Теоретичні відомості 3

2.2.1 Вступ до UML 3

2.2.2 Діаграма варіантів використання 3

2.2.3 Актори 3

2.2.4 Варіанти використання 3

2.2.5 Відносини на Use Case 4

2.2.6 Сценарії використання 4

2.2.7 Діаграми класів 4

2.2.8 Логічна структура БД 4

2.2.9 Проєктування БД 4

1.2. Хід роботи 5

1. Проаналізувати тему та спроєктувати діаграму варіантів використання відповідно до обраної теми лабораторного циклу. 5

Код: 5

Діаграма: 6

2. Спроєктувати діаграму класів предметної області. 6

Код: 6

Діаграма: 10

3. Вибрати 3 варіанти використання та написати за ними сценарії використання. 10

Сценарій 1: Додати завантаження (HTTP/HTTPS) 10

Сценарій 2: Пауза / Відновлення (з урахуванням протоколу) 12

Сценарій 3: Налаштувати розподіл швидкості (незалежно від протоколу) 13

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. 14

1.3. Висновки 15

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

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
}
title Use Case

actor "Користувач" as User

rectangle "Download Manager" {
  usecase "Додати завантаження\n(HTTP/HTTPS)" 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 (залишаємо лише потрібні)
User --> UC_Manage
User --> UC_BW
User --> UC_Browser

' Include-ланцюжок
UC_Browser ..> UC_Add : <<include>>
UC_Add ..> UC_Status : <<include>>
UC_Status ..> UC_Stats : <<include>>

' Generalization для керування
UC_PauseResume -|> UC_Manage
```

```
UC_Cancel -|> UC_Manage
@enduml
```

Діаграма:

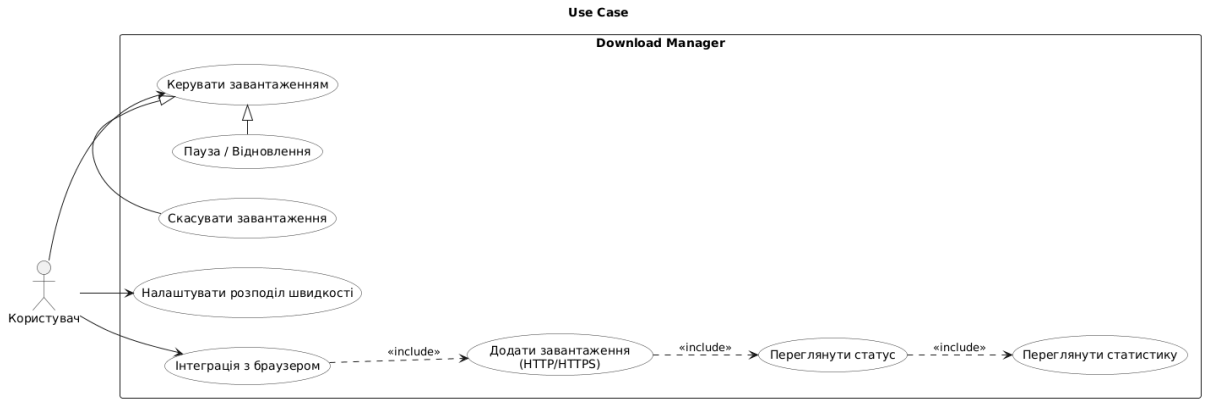

Рисунок 1 – Use case діаграма для користувача

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
```

```
    +Protocol: DownloadProtocol
    +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
  }

  enum DownloadProtocol {
    HTTP
    HTTPS
  }
}

package "Application" {
  class DownloadService {
```

```
  +Add(url: string, savePath: string): Guid
  +GetStatus(id: Guid): DownloadStatus
  +List(): List<DownloadTask>
  +Manage(id: Guid, action: IManageAction): void
}

' Сервіс визначення/перевірки протоколу
class ProtocolResolver {
  +Resolve(url: string): DownloadProtocol
  +IsSupported(protocol: DownloadProtocol): bool
}

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
  DownloadService ..> ProtocolResolver : визначає протокол
}

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:

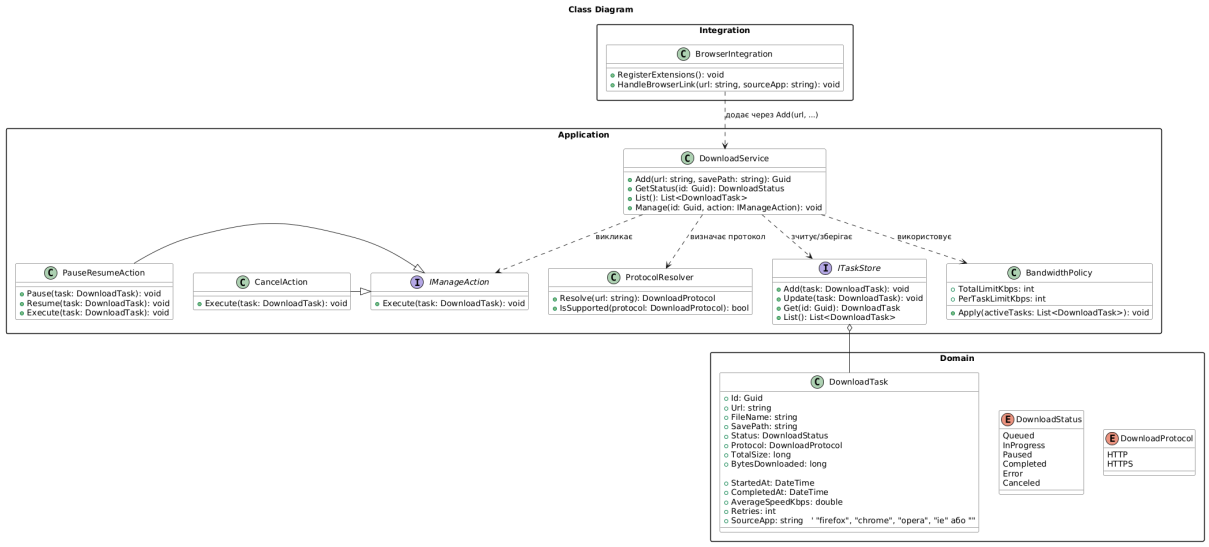

Рисунок 2 –Діаграма класів

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

Сценарій 1: Додати завантаження (HTTP/HTTPS)

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, середня швидкість, кількість спроб).
- Для завдання зафіксовано протокол HTTP або HTTPS.

Взаємодіючі сторони:
Користувач, Система.

Короткий опис:
Користувач додає нове завантаження за URL; система визначає й перевіряє протокол (http/https), доступність ресурсу та 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, із браузера/буфера).
4. Система визначає протокол за схемою URL: http або https; перевіряє, що протокол підтримується.
5. Система виконує попередню перевірку (HEAD/GET): доступність ресурсу, розмір, підтримка Range.
6. У разі HTTP→HTTPS переадресації система слідує редіректу (за політикою безпеки) і фіксує фактичний протокол.
7. Система створює DownloadTask (із полем протоколу), додає його до списку та запускає завантаження або ставить у чергу.
8. Система оновлює статус і показує елемент у списку завантажень.

Винятки:

- Некоректний/порожній URL -> система просить виправити.
- Непідтримуваний протокол (наприклад, ftp:) -> повідомлення про помилку, запропонувати http/https-посилання.
- Сервер недоступний / помилка мережі -> статус Error, пропозиція Повторити.
- Для HTTPS: помилка TLS/сертифіката (прострочений/недійсний) -> повідомлення та зупинка (безпека).
- Недостатньо місця/немає доступу до теки -> запропонувати інший шлях.
- Сервер не підтримує Range -> попередження, що відновлення буде лише “з початку”.

Примітки:
Виклик із браузера (розширення/контекстне меню) передає URL зі схемою http/https; система зберігає джерело (chrome/firefox/opera).

Сценарій 2: Пауза / Відновлення (з урахуванням протоколу)

Передумови:

- Існує активне завантаження зі статусом InProgress.
- Бажано, щоб сервер підтримував Range (часткові запити) — для коректного resume.

Постумови:

- Після “Пауза” завантаження переходить у Paused, збережено BytesDownloaded.
- Після “Відновлення” завантаження повертається в InProgress і продовжується з позиції зупинки; використовується той самий протокол (HTTP або HTTPS), що й на старті/після редіректу.

Взаємодіючі сторони:
Користувач, Система.

Короткий опис:
Користувач зупиняє та відновлює завантаження без повторного отримання вже завантажених даних; протокол не змінюється в процесі.

Основний потік подій:

1. Користувач у списку обирає завантаження й натискає “Пауза”.
2. Система коректно зупиняє IO/мережеві операції, фіксує BytesDownloaded.
3. Статус змінюється на Paused.
4. Користувач натискає “Відновити”; система формує запит із заголовком Range від збереженої позиції, використовуючи той самий протокол.
5. Завантаження триває; система оновлює статус і статистику (середня швидкість, час).

Винятки:

- Сервер не підтримує Range -> попередження: відновлення лише “з початку”.
- Обрив мережі під час відновлення -> Error або Paused, пропозиція повторити.
- Для HTTPS: помилка TLS під час відновлення (наприклад, змінився сертифікат) -> повідомлення про безпеку, пауза/помилка.
- Файл заблоковано іншою програмою -> запропонувати закрити доступ/змінити шлях.

Примітки:
Якщо при відновленні сервер робить примусовий редірект HTTP→HTTPS, система оновлює фактичний протокол у завданні й продовжує (за політикою безпеки).

Сценарій 3: Налаштувати розподіл швидкості (незалежно від протоколу)

Передумови:

- Застосунок запущено; є активні або заплановані завантаження (HTTP/HTTPS).

Постумови:

- Застосовано нові обмеження швидкості (загальні/не завдання); активні завантаження підлаштувались під політику.
- Налаштування збережено для наступних сеансів.

Взаємодіючі сторони:
Користувач, Система.

Короткий опис:
Користувач задає правила розподілу пропускної спроможності; політика застосовується однаково до HTTP та HTTPS завантажень.

Основний потік подій:

1. Користувач відкриває “Налаштування” -> “Швидкість завантаження”.
2. Система показує поля: “Загальний ліміт (kbps)” і “Ліміт на завдання (kbps)”.

3. Користувач вводить значення й натискає “Застосувати”.
4. Система перераховує активні швидкості з урахуванням обох лімітів, незалежно від протоколу кожного завдання.
5. Система зберігає налаштування та повідомляє про успішне застосування.

Винятки:

- Нуль/негативні значення -> система просить ввести коректний ліміт.
- “Ліміт на завдання” > “Загальний ліміт” -> запропонувати скоригувати параметри.
- Конфлікт із системними обмеженнями (фаєрвол/QoS) -> підказка перевірити налаштування ОС/мережі.

Примітки:
Просте правило — рівний розподіл у межах загального ліміту з верхньою межею на завдання; надалі можна додати ваги/пріоритети. Протокол (HTTP/HTTPS) не впливає на обчислення лімітів, лише на мережевий стек і безпеку

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.

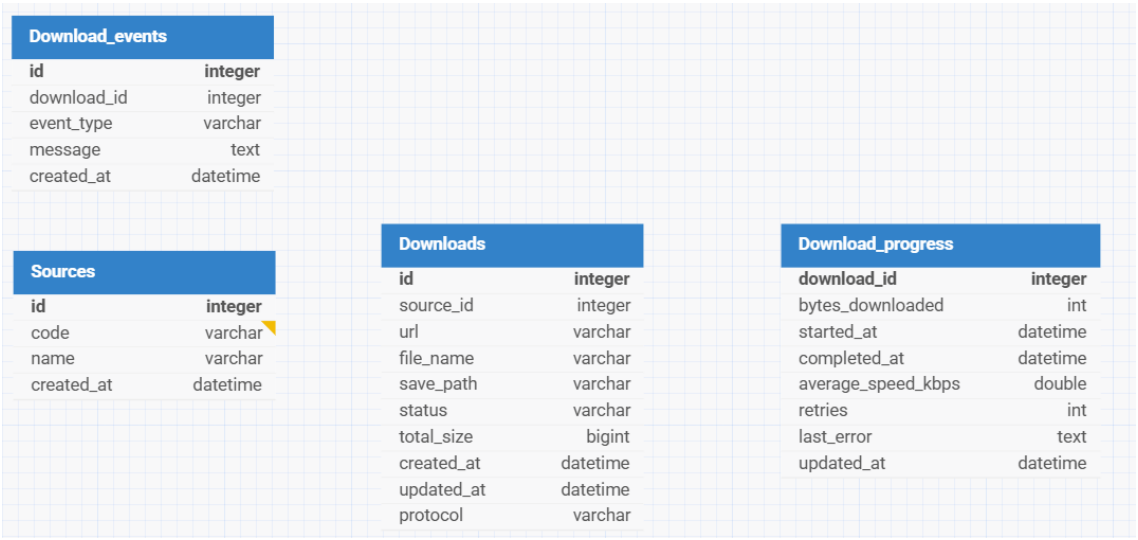

Рисунок 3 – Основні класи бази даних

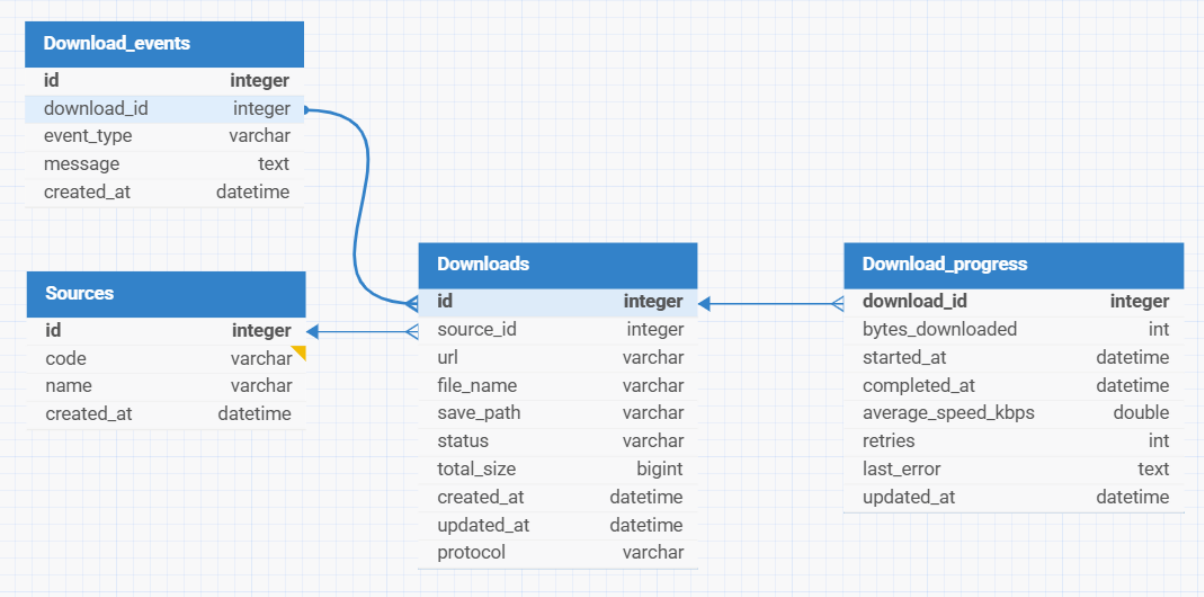

Рисунок 4 – структура бази даних системи

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).