Міністерство освіти і науки України
КПІ ім. Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Звіт

З дисципліни “Технології розроблення програмного забезпечення”

Варіант: 23

Виконала
студентка групи ІА-23:
Павленко Анастасія

Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Лабораторна робота №3

Діаграма розгортання. Діаграма компонентів. Діаграма взаємодій та послідовностей.

Мета: Ознайомитися з основними типами діаграм UML, зокрема діаграмою розгортання, діаграмою компонентів, діаграмою взаємодій та послідовностей. Навчитися створювати дані діаграми для відображення структури програмного забезпечення та взаємодії між його компонентами.

Варіант: 23 Project Management software (proxy, chain of responsibility, abstract factory, bridge, flyweight, client-server)

Програмне забезпечення для управління проектами повинно мати наступні функції: супровід завдань/вимог/проектів, списків команд, поточних завдань, планування за методологіями agile/kanban/rup (включаючи дошку завдань, ітерації тощо), мати можливість прикріплювати вкладені файли до завдань та посилатися на конкретні версії програми, зберігати виконувані файли для кожної версії.

Завдання

1. Ознайомитися з короткими теоретичними відомостями.
2. Розробити діаграму розгортання для проектованої системи.
3. Розробити діаграму компонентів для проектованої системи.
4. Розробити діаграму послідовностей для проектованої системи.
5. Скласти звіт про виконану роботу.

Зміст

Короткі теоретичні відомості

Діаграма розгортання (англ. deployment diagram) — діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду.

1. Вузол

Вузлом (node) є деякий наявний фізично елемент системи, що володіє деяким обчислювальним ресурсом. В останній версії мови UML поняття вузла розширене і може включати не тільки обчислювальні пристрої (процесори), але й інші механічні або електронні пристрої, такі як сенсори, принтери, модеми, цифрові камери, сканери і маніпулятори. Графічно на діаграмі розгортання вузол зображається у формі тривимірного куба. Вузол має власне ім'я, яке вказується всередині цього графічного символа. Самі вузли можуть подаватися типами або екземплярами.

2. З’єднання

Окрім власне зображень вузлів на діаграмі розгортання вказуються відношення між ними. Відношеннями виступають фізичні з'єднання між вузлами, залежності між вузлами та компонентами, зображення яких теж можуть бути присутніми на діаграмах розгортання.

З'єднання є різновидом асоціації і зображаються відтинками ліній без стрілок. Наявність такої лінії вказує на необхідність організації фізичного каналу для обміну інформацією між відповідними вузлами.

Окрім з'єднань на діаграмі розгортання можуть бути присутніми відношення залежності між вузлом і розгорненими на ньому компонентами.

Діаграма компонент (англ. Component diagram) — в UML, діаграма, на якій відображаються компоненти, залежності та зв'язки між ними, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Модуль програмного забезпечення може бути

представлено як компоненту. Деякі компоненти існують під час компіляції, деякі — під час компонування, а деякі під час роботи програми.

Компонент (англ. Component) — логічний блок системи, за рівнем абстракції трохи вищий, ніж «клас». Компоненти об'єднуються, разом використовуючи структурні зв'язки, щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Інтерфейс постачання (англ. Provide interface) — набір відкритих атрибутів та операцій, які повинні бути надані класами, що реалізують даний інтерфейс.

Інтерфейс вимоги (англ. Required interface) — набір відкритих атрибутів та операцій, які вимагаються класами, що залежать від даного інтерфейсу.

Структурна взаємодія — зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту.

При використанні діаграми компонент, щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається *з'єднанням делегації*.

Діаграма послідовності (Sequence Diagram) — показує часові особливості передачі і прийому повідомлень об’єктами. Впорядкованість за часом слід розуміти як послідовність дій.

Позначення діаграми послідовності

1. *Лінія життя* починається з об’єкта-прямокутника та зображується вертикальною пунктирною лінією. Вона служить для позначення періоду часу, протягом якого об’єкт існує в системі.

- *Об’єкт (учасник)* — позначення лінії життя та екземпляр класу у горизонтального прямокутника.

- *Актор* — використовується, коли конкретна діаграма послідовності належить варіанту використання.
- *Сутність* — представляє системні дані.
- *Межа/кордон* — вказує на межу системи/ граничний елемент у системі.
- *Управління* вказує на керівну сутність або менеджера. Він організовує та планує взаємодії між кордонами та сутностями та служить посередником між ними.

2. *Смуга активації (фокус управління)* — тонкий прямокутник на лінії життя, протягом якого елемент виконує операцію.

3. *Повідомлення* (виклики) з'являються в послідовному порядку на лінії життя. Повідомлення зображується за допомогою стрілок. Початок стрілки завжди торкається лінії життя відправника та лінії життя об'єкта, що приймає повідомлення.

4. Інші позначення:

Коментар — прямокутник із загнутим кутом. Може бути з'єднаний з об'єктом пунктирною лінією.

Фрагмент використовується, якщо процеси утворюють цикл або вимагають виконання умов для його закінчення. Він складається з вікна та оператора фрагмента в п'ятикутнику зверху зліва.

Альтернативний фрагмент — показує один або декілька альтернативних сценаріїв, де виконується лише один, той, чия умова істинна.

Опціональний фрагмент — виконується, лише якщо вказана умова, істинна. Він має лише одну умову і не поділяється на операнди.

Фрагмент циклу — використовується для послідовності, що повторюється. Може мати межі ітерації, що пишуться біля назви loop

Фрагмент посилання — для повторного використання частини послідовності в іншому місці діаграми.

Хід виконання роботи

Діаграма розгортання (Deployment Diagram)

Для створення UML діаграми розгортання (Deployment Diagram) до проєкту Project Management System, необхідно показати, як програмні компоненти будуть розміщені на фізичній інфраструктурі (сервери, бази даних, клієнтські пристрої тощо). Діаграма відображає, як система буде фізично працювати у реальному світі.

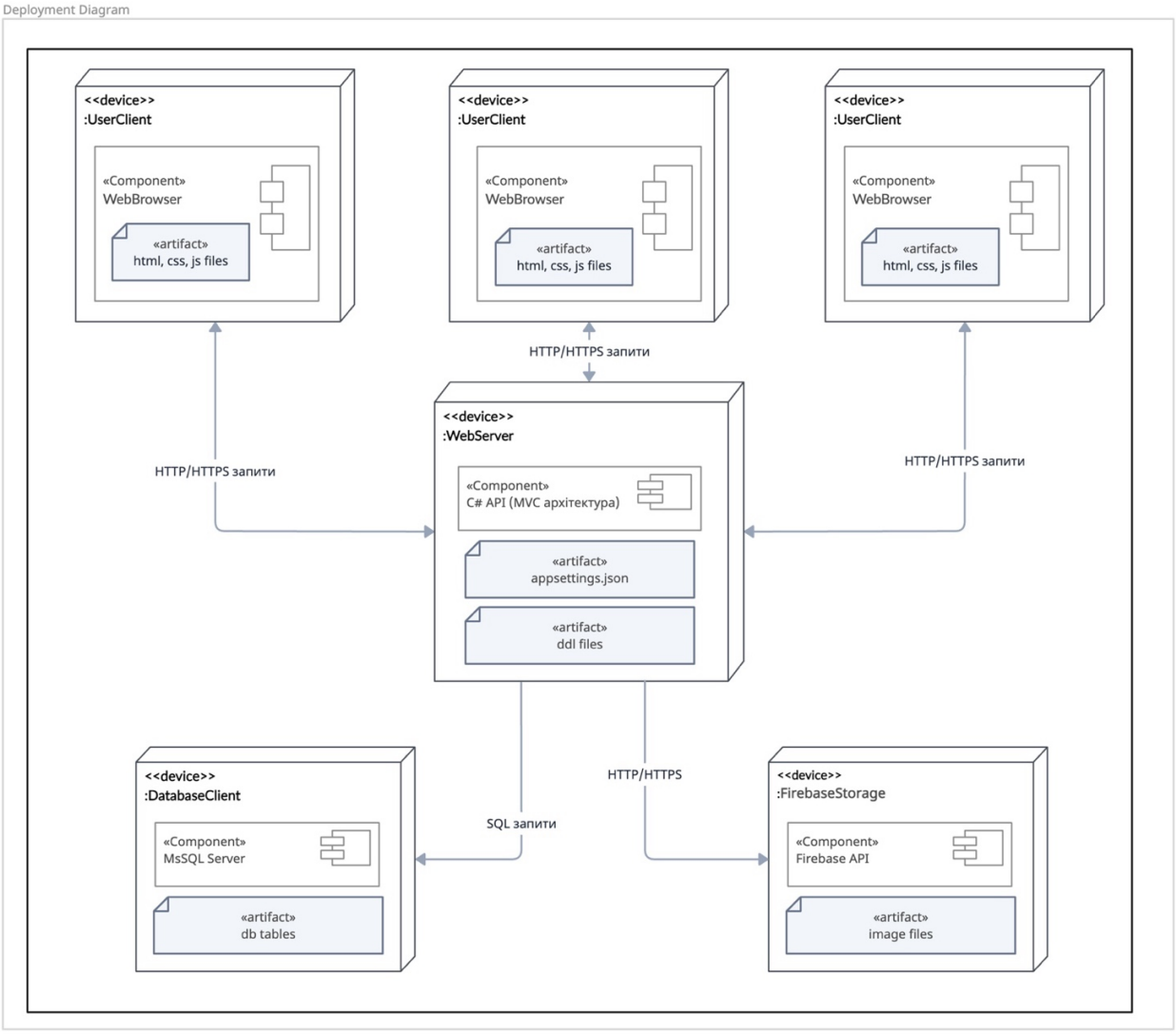

Рисунок 1. Діаграма розгортання

Більш чітке зображення можна переглянути в папці до лабораторної роботи №3 на гітхаб, або за посиланням в графічному редакторі Creatly: https://app.creately.com/d/NZZzSk3LQjz/view

Вузли, компоненти та артефакти:

1. *Вузол: Клієнтський пристрій (браузер користувача)*
   - Компоненти:
     - Веб-додаток (HTML/JavaScript): Клієнтський інтерфейс користувача.
   - Артефакти:
     - HTML/JavaScript/CSS файли: Завантажуються в браузер для відображення UI.
   - Зв’язки:
     - Браузер надсилає HTTP/HTTPS запити до сервера API.
2. *Вузол: Сервер API*
   - Компоненти:
     - C# API (MVC архітектура): Основна логіка обробки запитів від клієнтського інтерфейсу, бізнес-логіка, роутинг запитів.
   - Артефакти:
     - Конфігураційний файл (appsettings.json): Файл налаштувань API.
     - DLL файли: Компіляції C# API.
   - Зв’язки:
     - C# API взаємодіє з MsSQL базою даних для збереження та отримання даних.
     - C# API взаємодіє з Firebase Storage для обробки файлів.
3. *Вузол: MsSQL (Сервер бази даних)*
   - Компоненти:
     - MsSQL Server: База даних для зберігання інформації про проєкти, завдання, користувачів тощо.
   - Артефакти:
     - Таблиці бази даних: Дані про користувачів, проєкти, завдання.
     - SQL скрипти: Операції над даними (вставка, оновлення, видалення).
   - Зв’язки:
     - MsSQL взаємодіє з C# API через SQL запити.

4. *Вузол: Firebase Storage (Хмарне сховище)*

- Компоненти:
  - Firebase API: Інтерфейс для роботи із хмарним сховищем файлів.
- Артефакти:
  - Зображення: Файли, завантажені користувачами, зберігаються у Firebase.
- Зв’язки:
  - Firebase API взаємодіє з C# API для завантаження/отримання зображень.

Діаграма компонентів (Component diagram)

Щоб створити діаграму компонентів UML для проєкту, потрібно відобразити основні частини системи та їхні зв’язки між собою.

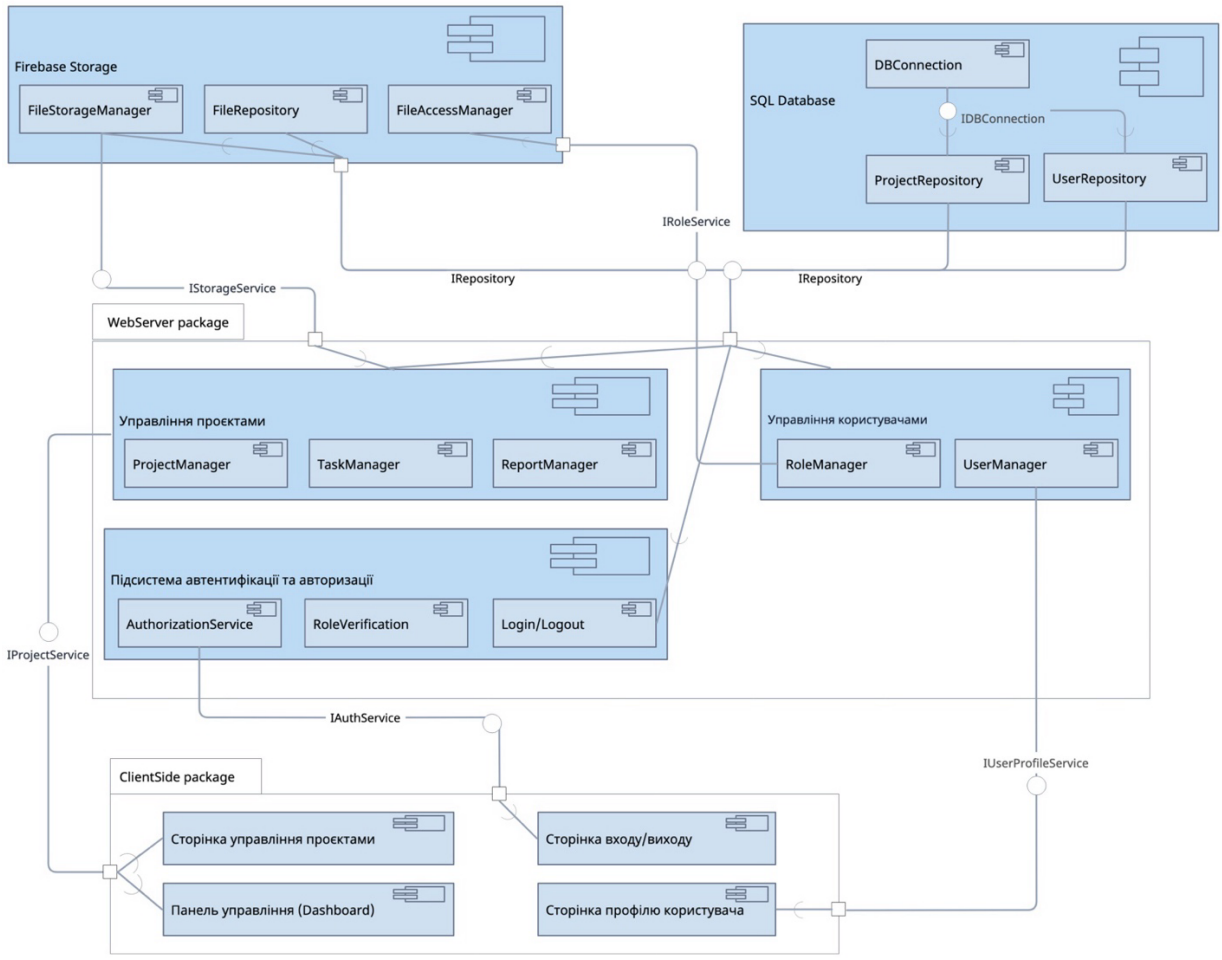

Рисунок 2. Діаграма компонентів

Більш чітке зображення діаграми можна переглянути в папці лабораторної роботи №3 на гітхаб або за посиланням в графічному редакторі Creatly: https://app.creately.com/d/nHTD37Zc7AQ/view

Опис компонентів, підсистем та інтерфейсів

Підсистема Firebase Storage

Компоненти:

- FileStorageManager: відповідає за завантаження та отримання файлів із Firebase Storage.
  - Інтерфейси:
    - Provided: IStorageService — надає інтерфейс для доступу до зберігання файлів для інших компонентів.
    - Required: IRepository — залежить від цього інтерфейсу для збереження метаданих про файли в базі даних.
- FileRepository: відповідає за збереження метаданих файлів, таких як посилання на файли, власник, тип файлу, і прив'язка до конкретних проектів або користувачів.
  - Інтерфейси:
    - Required: IRepository — для доступу до бази даних.
- FileAccessManager: керує доступом до файлів залежно від ролей користувачів. Цей компонент взаємодіє з RoleManager для перевірки прав доступу.
  - Інтерфейси:
    - Required: IRoleService — для перевірки ролей користувачів через компонент RoleManager у підсистемі авторизації.

Підсистема SQL Database

Компоненти:

- DBConnection: відповідає за встановлення і підтримку з'єднання з базою даних.

- Інтерфейси:
  - Provided: IDBConnection — надає інтерфейс для підключення до бази даних для інших компонентів, таких як ProjectRepository та UserRepository.
- ProjectRepository: Відповідає за збереження і отримання даних про проєкти, зокрема, структуру проєктів, завдання, учасників тощо.
  - Інтерфейси:
    - Required: IDBConnection — використовує інтерфейс для доступу до бази даних через DBConnection.
- UserRepository:
  - Зберігає і управляє даними про користувачів, включаючи їх профілі, ролі та права доступу. Взаємодіє з UserManager та RoleManager.
  - Інтерфейси:
    - Required: IDBConnection — використовує інтерфейс для доступу до бази даних.

Пакет WebServer

Підсистема управління проєктами:

- ProjectManager: відповідає за створення, управління та видалення проєктів.
  - Інтерфейси:
    - Required: IRepository (через ProjectRepository) для взаємодії з базою даних.
    - Required: IStorageService для роботи з файлами через FileStorageManager.
    - Provided: IProjectService для передачі даних проєкту.
- TaskManager: відповідає за управління завданнями всередині проєктів, включаючи створення, редагування та видалення завдань.
  - Інтерфейси:

        - Required: IRepository для збереження та отримання даних .
        - Required: IStorageService для роботи з файлами.
- ReportGenerator: генерує звіти про статус проєктів, які можна завантажити або переглянути всередині системи.
    - Інтерфейси:
        - Required: IRepository для отримання даних через ProjectRepository.
        - Required: IStorageService для роботи з файлами.

Підсистема автентифікації та авторизації:

- Login/Logout: відповідає за вхід/вихід користувачів із системи. Взаємодіє з UserRepository для верифікації користувачів.
    - Інтерфейси:
        - Required: IRepository для перевірки автентичності користувачів через UserRepository.
- RoleVerification: використовується для перевірки ролей користувачів, таких як адміністратор, менеджер або учасник.
- AuthorizationService: відповідає за надання і перевірку прав доступу користувачів до певних функцій і ресурсів системи.

Підсистема управління користувачами:

- UserManager: відповідає за створення, редагування та видалення користувачів у системі. Працює через UserRepository для збереження інформації.
    - Інтерфейси:
        - Required: IRepository для збереження даних про користувачів.
- RoleManager: відповідає за управління ролями користувачів. Використовує UserRepository для збереження і редагування ролей.
    - Інтерфейси:
        - Provided: IRoleService для компонентів системи, які потребують перевірки ролей.

Пакет ClientSide

Компоненти:

- Login/Logout Page: служить інтерфейсом користувача для входу/виходу із системи.
  - Інтерфейси:
    - Required: IAuthService для автентифікації користувачів.
- UserProfilePage: відповідає за відображення та редагування профілю користувача. Отримує дані через IUserProfileService.
  - Інтерфейси:
    - Required: IUserProfileService для отримання інформації про користувача.
- Dashboard: відображає загальну інформацію про проєкти, завдання, повідомлення та сповіщення для користувачів.
  - Інтерфейси:
    - Required: IProjectService для управління проєктами.
- ProjectManagementPage: служить інтерфейсом для управління проєктами на клієнтській стороні.
  - Інтерфейси:
    - Required: IProjectService для управління проєктами.

Діаграма послідовностей (Sequence diagram)

Для створення діаграми послідовностей розглянемо процес **аутентифікації користувача системи**.

Для цього детально розглянемо сценарій дій користувача для входу у власний вже існуючий аккаунт:

Актори: Адміністратор, Проєкт менеджер, Член команди

Передумови: Користувач не ввійшов в аккаунт застосунку.

Результат: Користувач увійшов в аккаунт.

Сценарій:

1. Користувач натискає кнопку увійти в аккаунт.
2. Система перенаправляє користувача на екран входу.
3. Користувач вводить свою електронну пошту та пароль.
4. Користувач натискає кнопку «Увійти»
5. Система перевіряє надану пошту та пароль.
6. Система перенаправляє користувача на домашню сторінку веб-застосунку.

Альтернативний сценарій:

5. Якщо надана комбінація пошти та паролю невірна – система перериває процес входу в аккаунт.
6. Система перенаправляє користувача на сторінку входу і показує повідомлення про помилку.

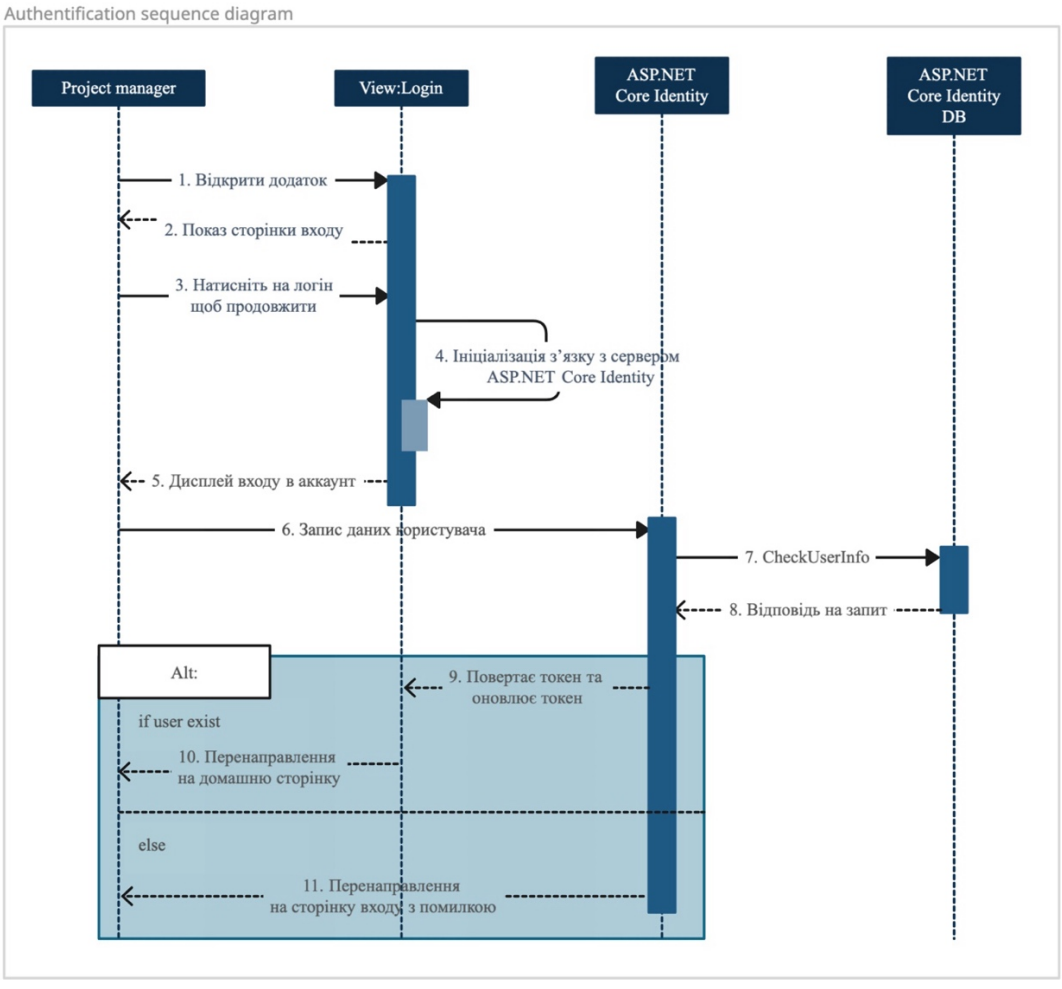

Рисунок 3. Діаграма послідовності: Аутентифікація

Більш чітке зображення можна переглянути в папці до лабораторної роботи №3 на гітхаб, або за посиланням в графічному редакторі Creatly: https://app.creately.com/d/otcJSRxuXXz/view

Додатково створимо діаграму для ілюстрації послідовності **генерації звіту**.

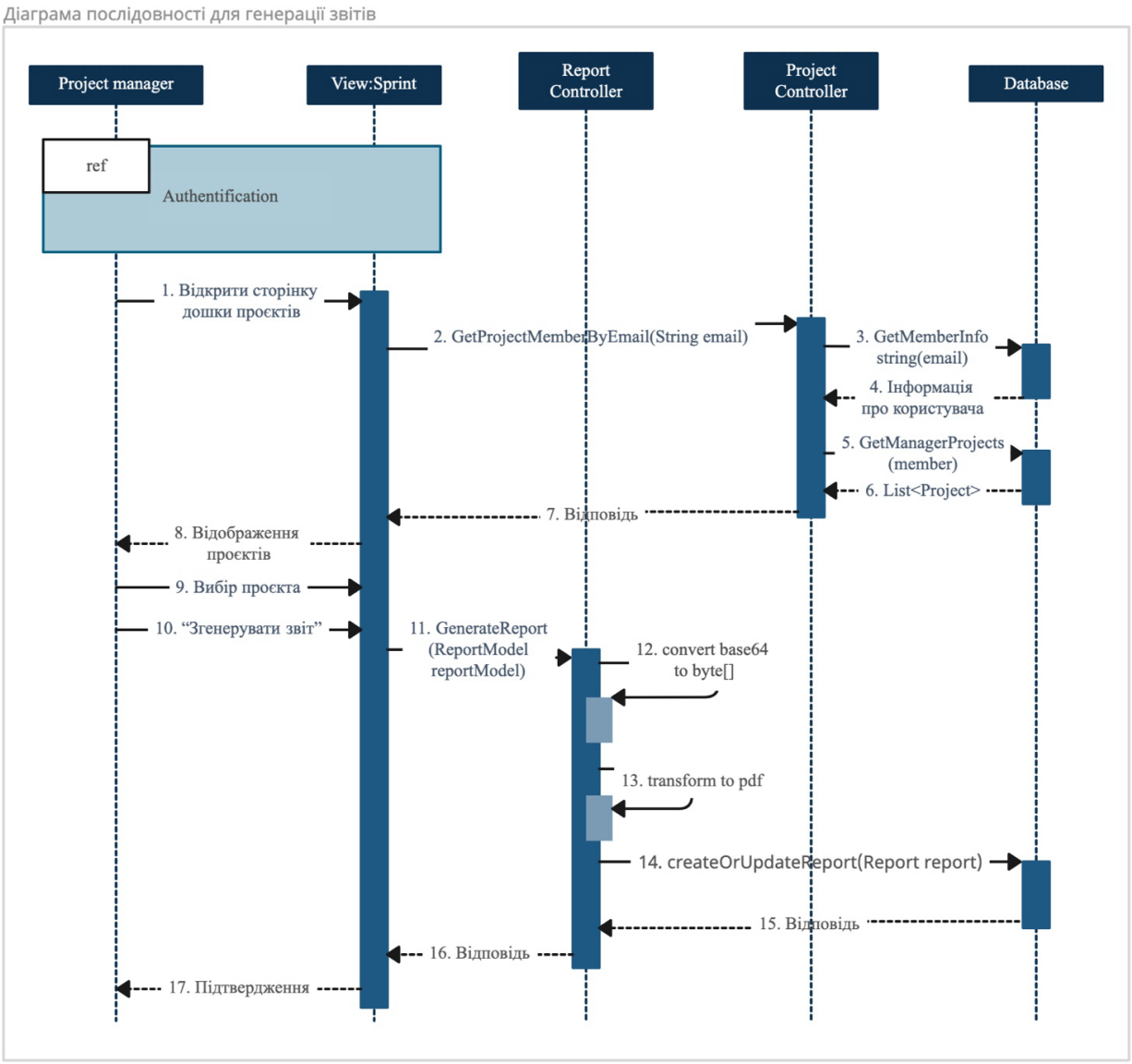

Рисунок 4. Діаграма послідовності: Генерація звітів

Більш чітке зображення можна переглянути в папці до лабораторної роботи №3 на гітхаб, або за посиланням в графічному редакторі Creatly: https://app.creately.com/d/f1SeClq31ni/view

Актори: Адміністратор, Проєкт менеджер

Передумови: Pdf звіт не згенерований.

Результат: Pdf звіт успішно згенерований.

Сценарій:

1. Користувач натискає кнопку меню на дошці.
2. Користувач обирає конкретний проєкт, звіт до якого бажає згенерувати.
3. Користувач обирає опцію згенерувати звіт.
4. З'являється повідомлення про успішне створення звіту і звіт завантажується на пристрій користувача.
5. Користувач натискає кнопку Ok.
6. Модальне вікно закривається.

Висновок:

У процесі виконання лабораторної роботи з технологій розробки програмного забезпечення ми ознайомилися з основними типами діаграм UML, зокрема діаграмою розгортання, діаграмою компонентів, діаграмою взаємодій та послідовностей. Це дало можливість глибше зрозуміти структуру та архітектуру програмного забезпечення, а також способи взаємодії між його компонентами.

Зокрема, для проекту з управління проектами, який реалізує патерни проектування, ми змогли наочно представити:

1. **Діаграму розгортання**, що показує фізичне середовище, в якому працюють компоненти програмного забезпечення, включаючи сервери та клієнтські додатки.
2. **Діаграму компонентів**, що відображає структуру системи, де чітко визначені зв'язки між основними компонентами, такими як модулі для управління завданнями, файлів версій та планування.
3. **Діаграму взаємодій та послідовностей**, яка ілюструє, як користувачі та системи взаємодіють один з одним, виконуючи певні функції програми, такі як створення та оновлення завдань, прикріплення файлів і планування за методологіями Agile/Kanban/RUP.

Таким чином, створення цих діаграм допомогло систематизувати знання про структуру та функціонування проектного програмного забезпечення. Впровадження навчених концепцій у реальний проект дозволить значно покращити його управління, ефективність та зручність використання.