Національний технічний університет України «КПІ»
Факультет інформатики та обчислювальної техніки
Кафедра Інформаційних систем та технологій

Лабораторна робота №3

з дисципліни « Сучасні технології розробки WEB-застосувань на платформі Microsoft.NET»

на тему: « Проектування REST веб-API»

Виконала:
студентка гр. ІС-11
Гаврильчик Яна
Викладач:
Бардін В.

2023 рік

Завдання:

Теоретична частина:

1. Ознайомитися з основами створення REST веб-API та методологією C4 для відображення архітектури системи.
2. Ознайомитися з основами створення ER-діаграм для представлення структури бази даних.

Практична частина:

1. З дотриманням вимог REST-у спроектувати веб-API для обраної(згідно варіанту) доменної області, використовуючи методологію C4 для створення діаграми архітектури системи.
2. Створити ER-діаграму для DAL (Data Access Layer), яка відображатиме структуру бази даних веб-API.
3. Оформити спроєктоване рішення у вигляді звіту до лабораторної роботи.

Варіант:

4	Гаманець. Керування власним бюджетом та фінансами	1. Власний бюджет складається з декількох рахунків, які поповнюються за заданими статтями прибутку. 2. Гроші цих рахунків можуть бути переведені з одного на інший, можуть витрачатись за заданими статтями витрат. 3. Підсумовуючи витрати та прибутки, можливо отримати інформацію, скільки було витрачено/отримано загалом/за певною статтею по заданому рахунку. **Функціональні вимоги:** 1. Ведення власного бюджету; 2. Отримання звітної інформації по рахунках.

Хід виконання роботи:

1. C4 модель – набір ієрархічних абстракцій (програмні системи, контейнери, компоненти та код).

Тобто у результаті повинно бути чотири види діаграм, хоча доволі популярною практикою є обмеження у використанні тільки діаграм контексту та контейнеру, які дають достатньої цінності вашій архітектурі.

Почнемо з діаграми контексту, яка допомагає побачити загальну картину архітектури:

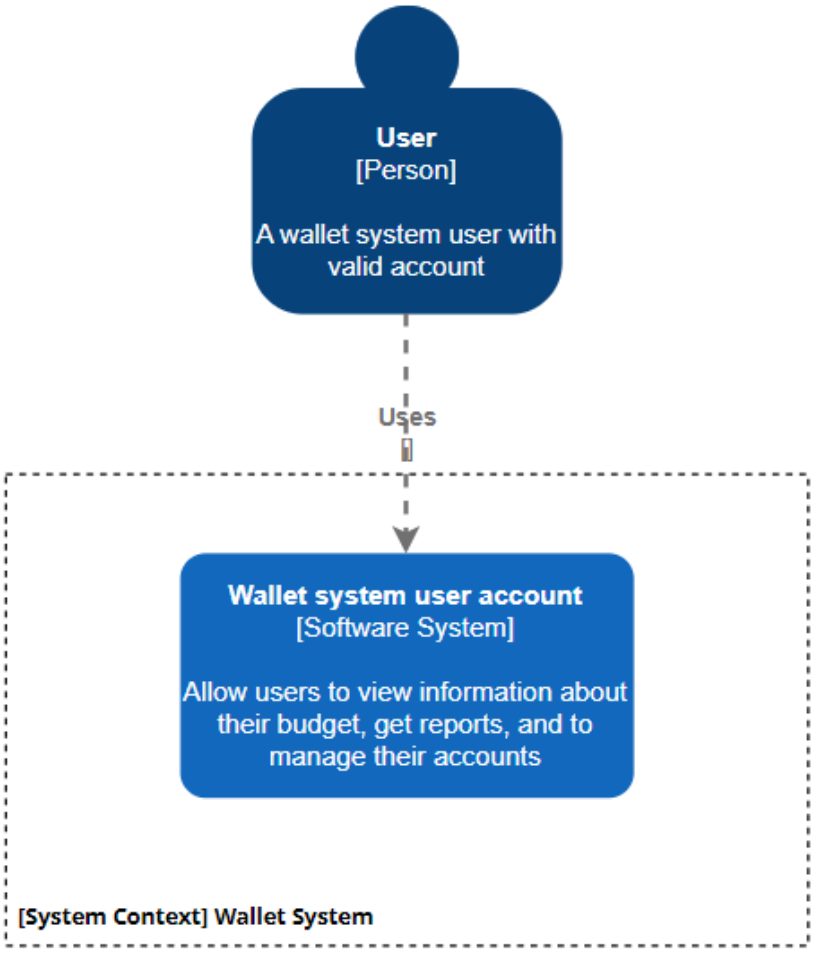


Рис.1 – Context diagram

У системі “Гаманець” буде один тип користувача “Користувач”, який матиме змогу зареєструватись у системі та використовувати її для ведення свого бюджету та керування фінансами.

Наступним кроком буде створення діаграми контейнерів. Вона показує високорівневу архітектуру ПЗ та розподіл відповідальності між нею. Вона також показує основні технологічні рішення та те, як контейнери взаємодіють один з одним:

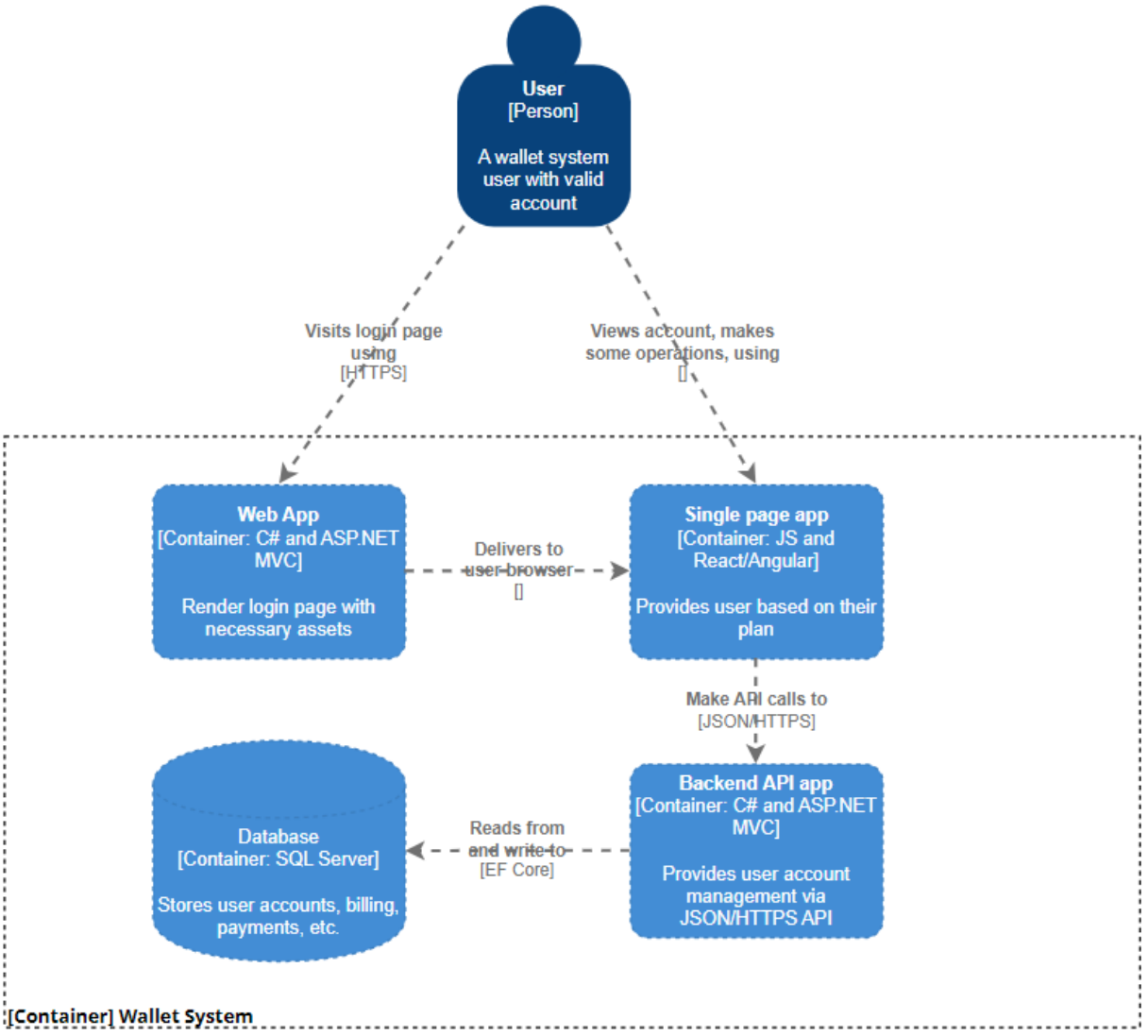


Рис.2 – Container diagram

Далі я збільшую масштаб і розкладаю кожен контейнер далі, щоб визначити основні структурні блоки та їх взаємодію. Діаграма компонентів

показує, як контейнер складається з кількох компонентів, що таке кожен із цих компонентів, їхні обов’язки та деталі технології/реалізації:

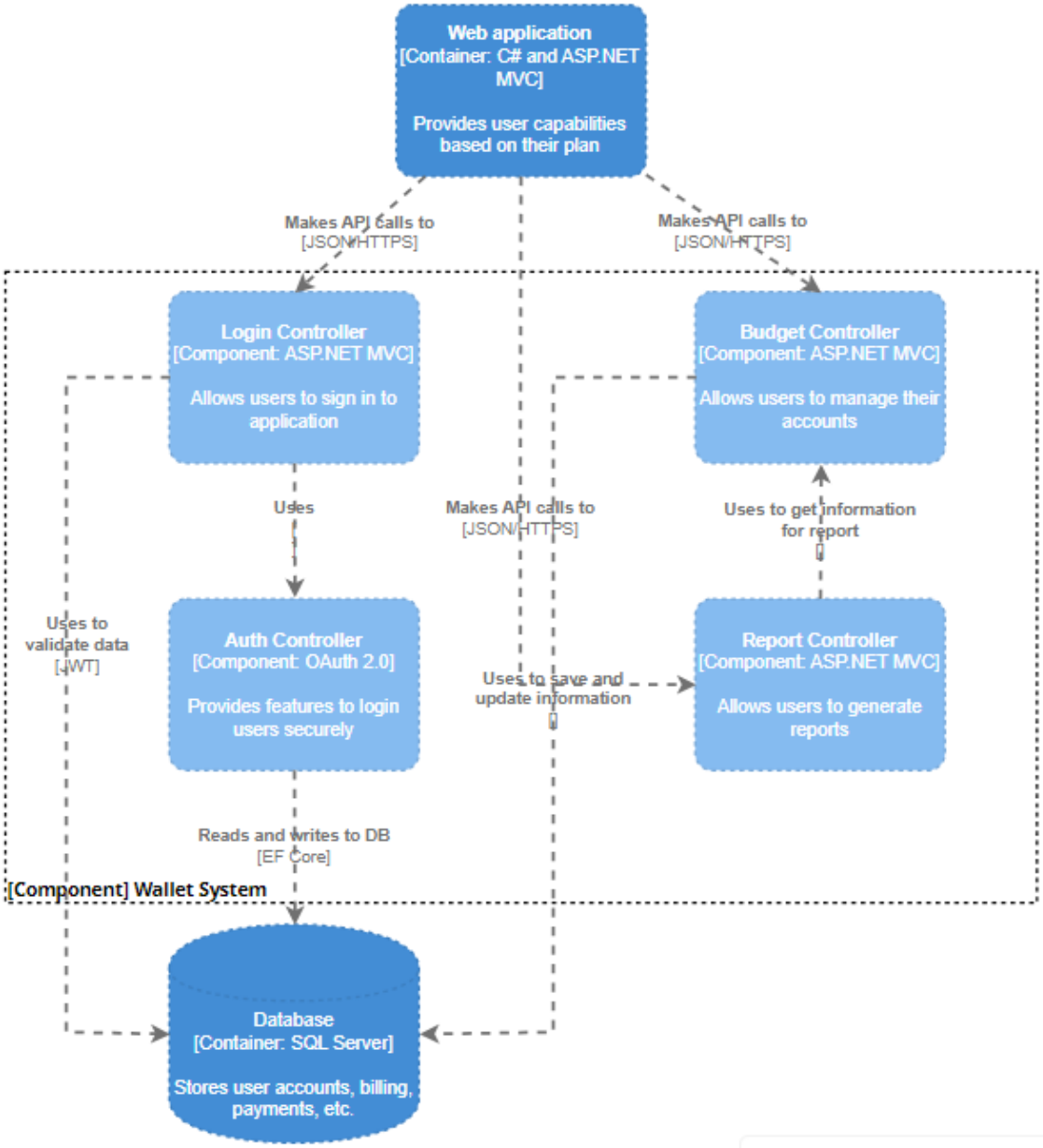


Рис.3 – Component diagram

Для розробки бекенду для застосунку буде застосований архітектурний паттерн MVC (Model-View-Controller), який використовується для розділення компонентів програми на три основні частини:

- **Модель (Model):** реалізація функціональності, пов'язаної з бюджетом та операціями з рахунками. Це включає в себе логіку для

створення рахунків, додавання транзакцій, переказу коштів, обчислення балансу тощо.

- **Вид (View):**реалізація відображення інтерфейсу користувача для роботи з бюджетом та звітами. Відображення складається з графічного інтерфейсу, який відображає дані та дозволяє користувачеві взаємодіяти з ними.
- **Контролер (Controller):** Контролер відповідає за обробку вхідних запитів користувача та виклик відповідних методів Моделі. Наприклад, коли користувач створює новий рахунок, Контролер обробляє цей запит і викликає метод Моделі для створення рахунку. Контролер також обробляє відображення звітів та взаємодію із Моделлю для отримання даних для відображення.

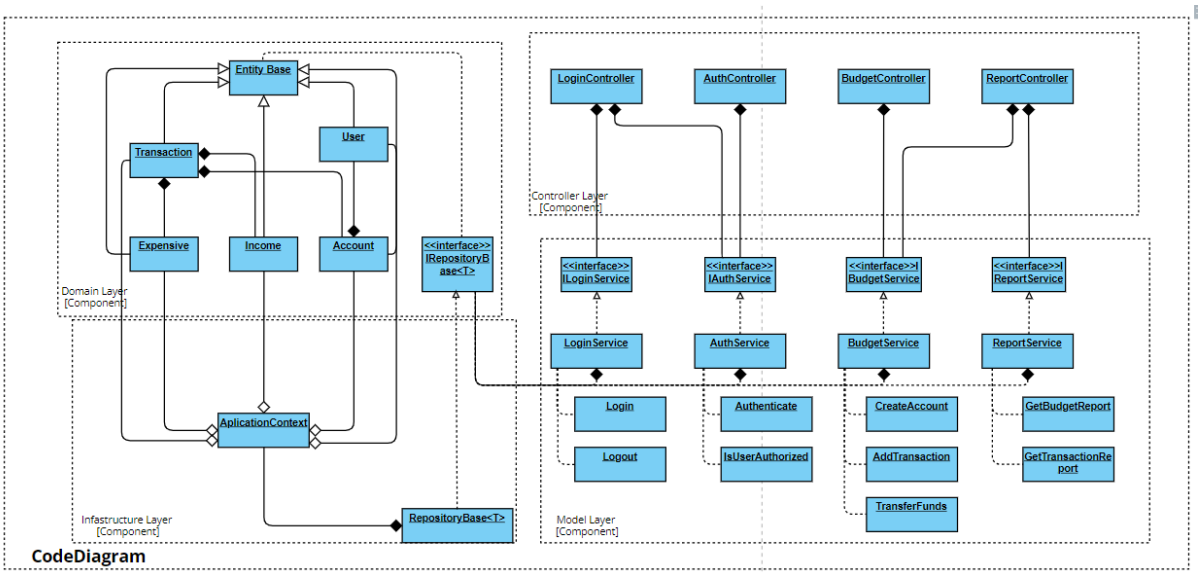
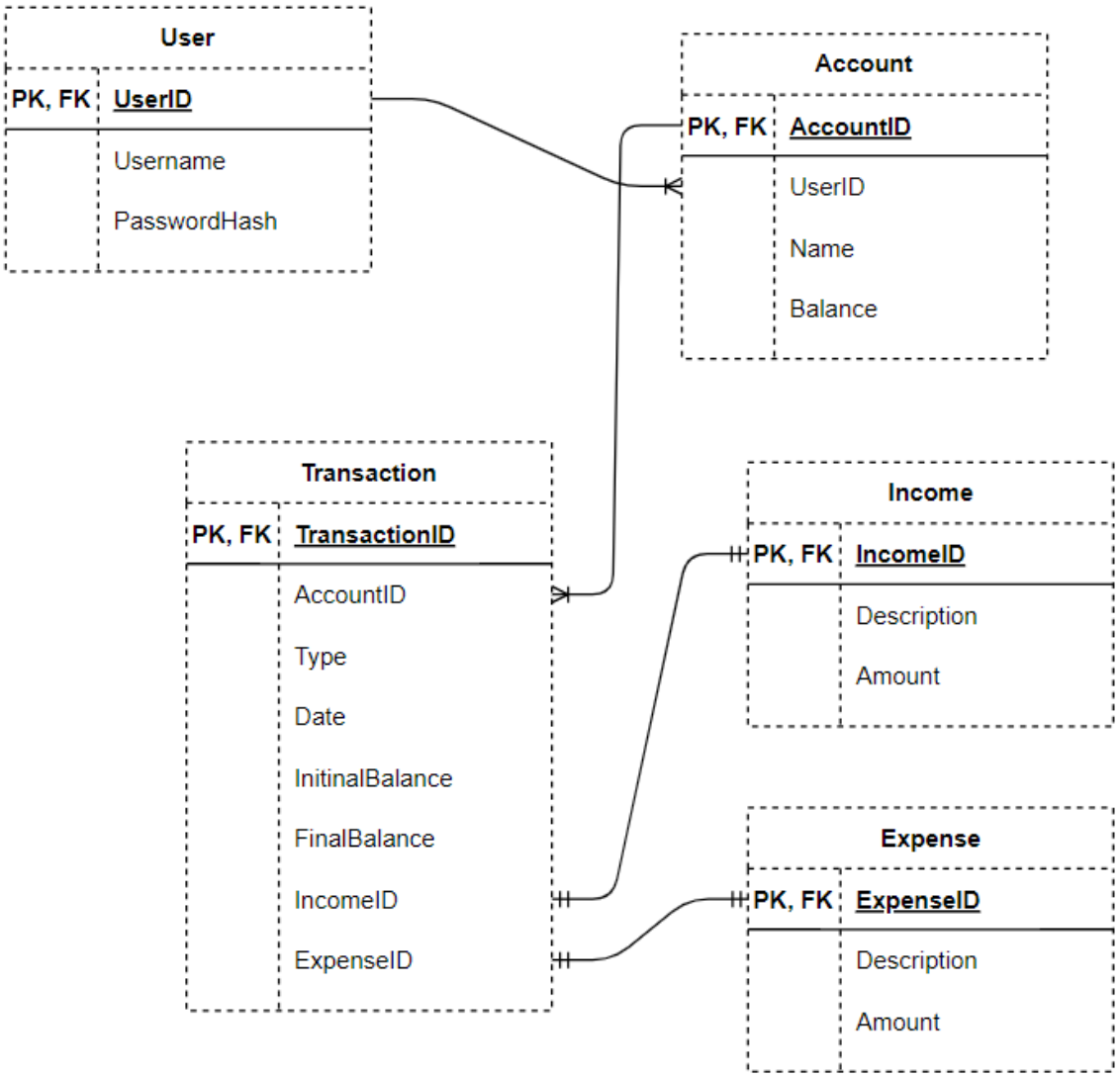


Рис.4 – Code diagram

2. Створена ER-діаграма для DAL має наступні сутності: User, Account, Transaction, Income, Expense. В свою чергу між сутностями присутні наступні зв’язки: User → Account (one to many), Account → Transaction (one to many), Transaction → Income (one to one), Transaction → Expense (one to one).



Таблиця: User

Призначення: містить інформацію про користувача системи та використовує дані таблиці для аутентифікації.

Поля:

- UserID: GUID - унікальний ідентифікатор користувача
- UserName: VARCHAR(20) - Ім’я користувача
- PasswordHash: VARCHAR(30) - Хеш паролю користувача

Таблиця: Account

Призначення: зберігає інформацію про рахунок користувача та використовуєтьсь для веденя бюджету.

Поля:

- AccountID: GUID - унікальний ідентифікатор рахунку
- UserID: GUID - унікальний ідентифікатор користувача
- Name: VARCHAR(20) - Ім’я (опис) рахунку
- Balance: DECIMAL - Баланс рахунку

Таблиця: Transaction

Призначення: зберігає історію фінансових транзакцій, які включають прибуток та витрати на рахунку користувача.

Поля:

- TransactionID: GUID - унікальний ідентифікатор транзакції
- AccountID: GUID - унікальний ідентифікатор рахунку
- Type: ENUM - тип транзакції (прибуток або витрати)
- Date: DateTime - дата транзакції
- InitialBalance: DECIMAL - баланс до здійснення транзакцій
- FinalBalance: DECIMAL - балан опісля здійснення транзакції
- IncomeID: GUID - унікальний ідентифікатор доходу
- ExpensiveID: GUID - унікальний ідентифікатор витрати

Таблиця: Income

Призначення: містить інформацію про фінансові доходи користувача та використовується для їх відстеження.

Поля:

- IncomeID: GUID - унікальний ідентифікатор доходу
- Description: VARCHAR (50) - опис статті доходу
- Amount: DECIMAL - сума доходу

Таблиця: Expensive

Призначення: містить інформацію про фінансові витрати користувача та використовується для їх відстеження.

Поля:

- ExpensiveID: GUID - унікальний ідентифікатор витрати
- Description: VARCHAR (50) - опис статті витрати
- Amount: DECIMAL - сума витрати