**Міністерство освіти і науки України**
**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра обчислювальної техніки**

# Лабораторна робота №4

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

Виконала:

студентка групи ІМ-11
Бащак Ярина Володимирівна

Перевірив:

доц. Крамар Ю. М.

Київ 2023

## Завдання.

**Практична частина**:

1. Використовуючи архітектуру розроблену в попередній роботі Імплементувати REST веб-API, використавши N Layered архітектуру.
2. Покрити модульними та інтеграційними тестами основні компоненти рішення.
3. Експортувати розроблене API до Postman.

## Мій варіант:

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

**Код проекту.**

Посилання на github-репозиторій.

**N-Layered архітектура**.

Є 3 компоненти (проекти):

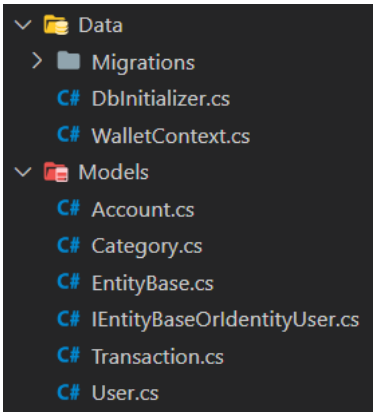
- Presentation Layer – поверхнений шар, який через контролери приймає HTTP-запити і віддає відповідь;
- Business Logic Layer – шар, який виконує логіку додатку, складається з сервісів;
- Data Access Layer – відповідає за взаємодію з базою даних та складається з контексту застосунку, сутностей і репозиторіїв.

**База даних.**

База даних створюється за допомогою Entity Framework. Всі сутності в моєму проекті наслідуються від EntityBase, який містить спільні дані для всіх сутностей (користувача в тому числі): ідентифікатор, час створення, час останнього оновлення.

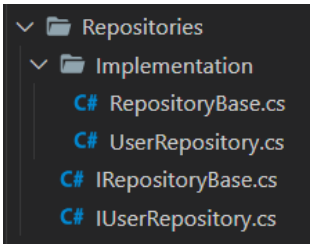
```
public class EntityBase : IEntityBaseOrIdentityUser
{
    public int Id { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime UpdatedAt { get; set; }

    public EntityBase()
    {
        var date = DateTime.Now;
        CreatedAt = date;
        UpdatedAt = date;
    }
}
```

Також використовується DbInitializer для початковго заповнення таблиць даними, якщо вони пусті.



**Автентифікація.**

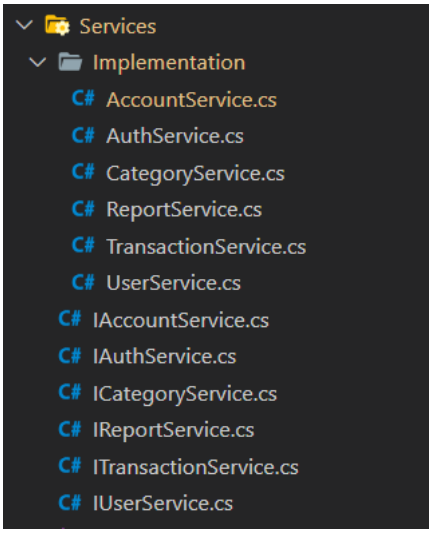
Для реалізації автентифікації було використано Microsoft.AspNetCore.Identity. Спеціально для цього, крім загального репозиторію RepositoryBase<T>, який використовує контекст, було створено репозиторій, що відповідає за створення користувача, пошук по імені та перевірку паролю, який працює на основі UserManager<User>.

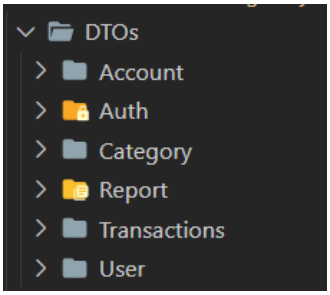



**Сервіси.**

В сервісах відбувається виконання всієї бізнес-логіки, а також перевірка доступу користувача до даних, що він запитує, і мапинг сутностей з бази даних до об'єктів передачі (DTO).

Також є окремо виділений сервіс пов’язатий аутентифікацією Authservice, який імплементує логін, реєстрацію і генерацію jwt-токена.




**Звіт для користувача**.

Звіт по доходах і витратах за категорією і загалом генерується в окремому сервісі для звітів ReportService, а сама структура звіту виглядає задається за допомогою DTOs:

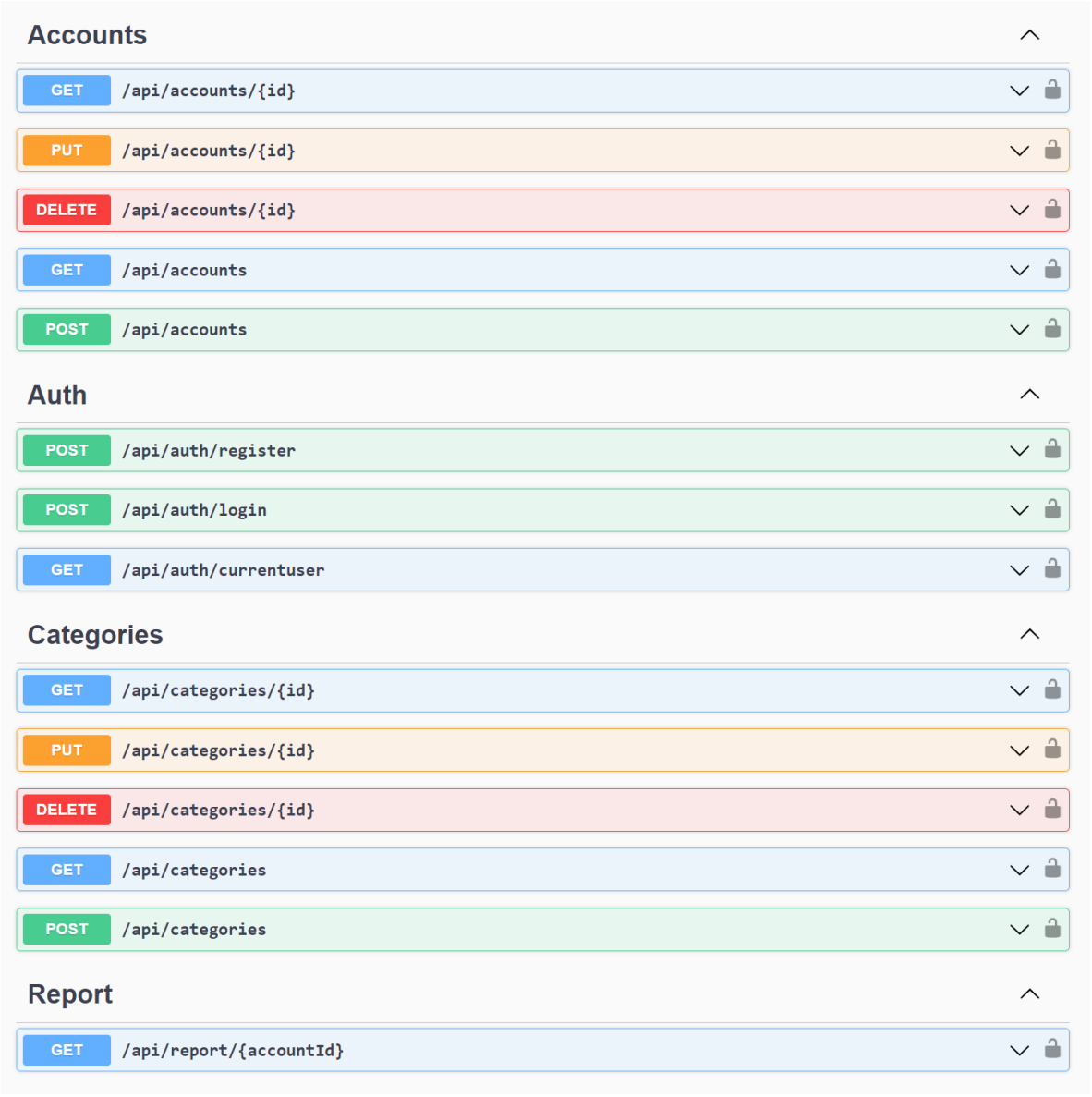
```
public class ReportDto
{
    public int AccountId { get; set; }
    public string AccountName { get; set; }
    public decimal Total { get; set; }
    public decimal Balance { get; set; }
    public SummaryByCategoryType Income { get; set; }
    public SummaryByCategoryType Expenses { get; set; }
```

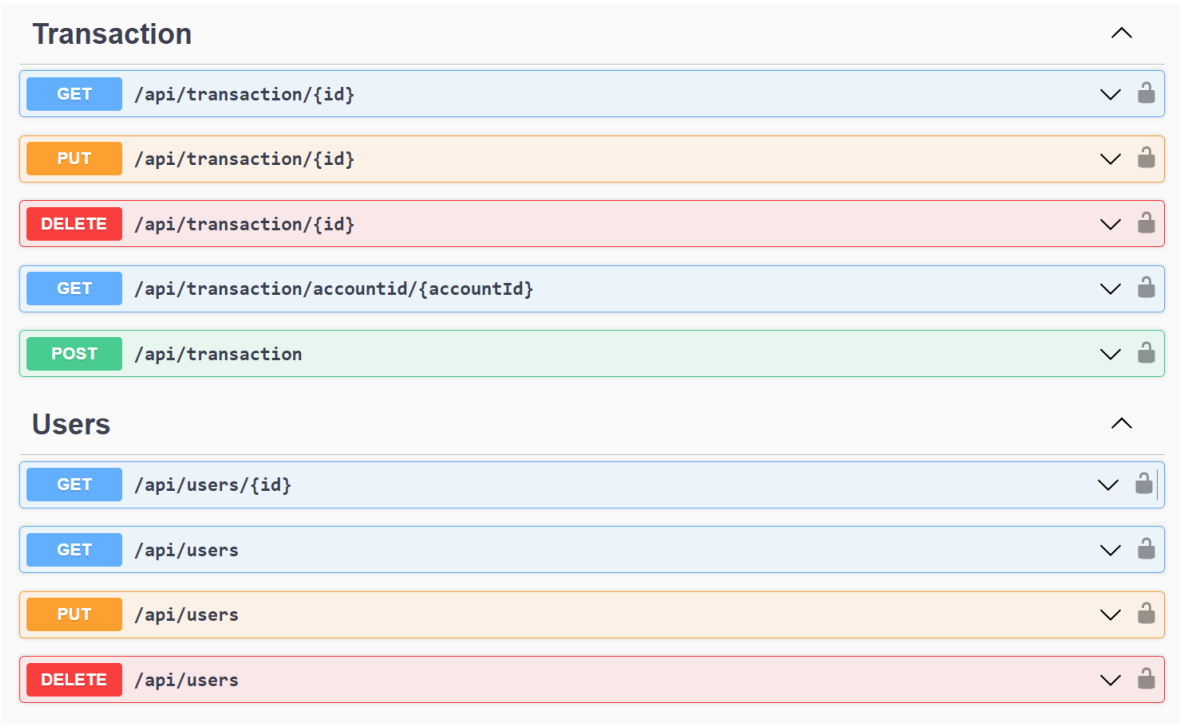
```
}

public class SummaryByCategoryType
{
    public decimal Total { get; set; }
    public List<CategorySummary> CategoriesSummary { get; set; }
}

public class CategorySummary
{
    public int CategoryId { get; set; }
    public string CategoryName { get; set; }
    public decimal Total { get; set; }
}
```

Приклад звіту по рахунку.


```
{
  "accountId": 1,
  "accountName": "Monobank account",
  "total": 980,
  "balance": 1430,
  "income": {
    "total": 2700,
    "categoriesSummary": [
      {
        "categoryId": 4,
        "categoryName": "Salary",
        "total": 2500
      },
      {
        "categoryId": 5,
        "categoryName": "Interest on deposits",
        "total": 200
      }
    ]
  },
  "expenses": {
    "total": 1720,
    "categoriesSummary": [
      {
        "categoryId": 6,
        "categoryName": "Transfer to another account",
        "total": 1250
      },
      {
        "categoryId": 1,
        "categoryName": "Food",
        "total": 170
      },
      {
        "categoryId": 2,
        "categoryName": "Home",
        "total": 300
      }
    ]
  }
}
```

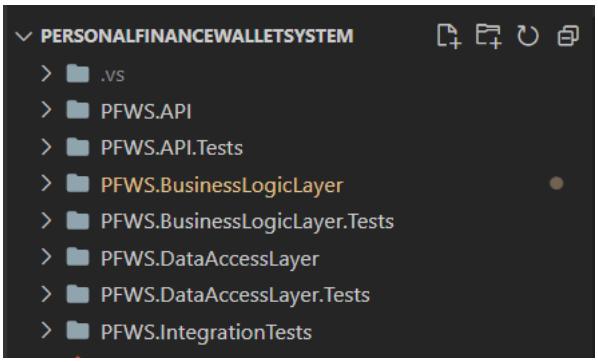
## Контролери і кінцеві точки.



**Тести.**

Для написання модульних тестів було створено 3 окремі проекти – для кожного шару. А для інтеграційний ще один. Всі вони використовують моки з бібліотеки Moq.




Результати тестів.

```
Passed!  - Failed:     0, Passed:     5, Skipped:     0, Total:     5, Duration: 708 ms - PFWS.DataAccessLayer.Tests.dll (net7.0)
Passed!  - Failed:     0, Passed:    26, Skipped:     0, Total:    26, Duration: 507 ms - PFWS.BusinessLogicLayer.Tests.dll (net7.0)
Passed!  - Failed:     0, Passed:    10, Skipped:     0, Total:    10, Duration: 128 ms - PFWS.API.Tests.dll (net7.0)
Passed!  - Failed:     0, Passed:     5, Skipped:     0, Total:     5, Duration: 151 ms - PFWS.IntegrationTests.dll (net7.0)
```