**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

***Факультет інформатики та обчислювальної техніки***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(повне найменування інституту, факультету)

***Інформатики та програмної інженерії***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(повна назва кафедри)

«**До захисту допущено**»



\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (ініціали, прізвище)



**Курсова робота**

**з дисципліни Компоненти програмної інженерії**

**за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»**

**спеціальності «121 Інженерія програмного забезпечення»**

| **на тему** | Веб-застосунок для збору коштів(серверна частина) | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |
|  | | | | | | |
| **Виконав: студент III курсу, групи** | | | | *ІП-11 Веремчук Ігор Ігорович* |  |  |
|  | | | *(прізвище, ім’я, по батькові )* | |  | (підпис) |
| **Керівник** | | *Доцент к.ф.м.н., доц., Поперешняк С.В.* | | |  |  |
|  | | *посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові* | | |  | (підпис) |
| **Члени комісії** | | *Доцент к.ф.м.н., доц., Поперешняк С.В.* | | |  |  |
|  | | *посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові* | | |  | (підпис) |
|  | | *ст. вик. Головченко М.М.* | | |  |  |
|  | | *посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові* | | |  | (підпис) |

Засвідчую, що у цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Київ – 2023 року

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**



Факультет (інститут)

(повна назва)

Кафедра

(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність *– 121 Інженерія програмного забезпечення*

Освітньо-професійна програма *– Програмне забезпечення інформаційних управляючих систем та технологій*



**ЗАВДАННЯ**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

*Веремчука Ігоря Ігоровича*

(прізвище, ім’я, по батькові)

| **1. Тема роботи** «*Веб-застосунок для збору коштів(серверна частина)*» |
| --- |
|  |

керівник роботи *Поперешняк Світлана Володимирівна, к.ф.м.н., доцент я*

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

**2. Термін подання студентом роботи**  «14» *грудня 2022 року*

**3.** **Вихідні дані до роботи**

| *Технічне завдання* |
| --- |
|  |

**4.** **Зміст пояснювальної записки**

| *1) Аналіз вимог до програмного забезпечення: основні визначення та терміни,* |
| --- |
| *опис предметного середовища, огляд існуючих технічних рішень та відомих* |
| *програмних продуктів, розробка функціональних та нефункціональних вимог* |
| *2) Моделювання та конструювання програмного забезпечення: моделювання та* |
| *аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура* |
| *програмного забезпечення* |
| *3) Розгортання та впровадження програмного забезпечення* |
| *4) Керівництво користувача, методика випробувань програмного продукту* |

**5. Перелік графічного матеріалу**

| *1) Схема структурна варіантів використань* |
| --- |
| *2) Схема структурна послідовностей* |
| *3) Схема бази даних* |
| *4) Схема структурна компонент* |

**6.** **Дата видачі завдання** «12» *жовтня 2022 року а*

**КАЛЕНДАРНИЙ ПЛАН**

| **№ з/п** | **Назва етапів виконання курсової роботи** | **Термін виконання етапів роботи** | **Примітка** |
| --- | --- | --- | --- |
|  | *Вивчення рекомендованої літератури* | *20.10.2023* |  |
|  | *Аналіз існуючих методів розв’язання задачі* | *30.10.2023* |  |
|  | *Постановка та формалізація задачі* | *04.11.2023* |  |
|  | *Аналіз вимог до програмного забезпечення* | *08.11.2023* |  |
|  | *Алгоритмізація задачі* | *14.11.2023* |  |
|  | *Моделювання програмного забезпечення* | *20.11.2023* |  |
|  | *Обґрунтування використовуваних технічних засобів* | *24.11.2023* |  |
|  | *Розробка архітектури програмного забезпечення* | *30.11.2023* |  |
|  | *Розробка програмного забезпечення* | *05.12.2023* |  |
|  | *Налагодження програми* | *10.12.2023* |  |
|  | *Виконання графічних документів* | *16.12.2023* |  |
|  | *Оформлення пояснювальної записки* | *24.12.2023* |  |
|  | *Подання КР на перевірку* | *21.01.2024* |  |
|  | *Захист КР* | *23.01.2024* |  |

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ігор ВЕРЕМЧУК

(підпис)

**Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Світлана ПОПЕРЕШНЯК

(підпис)

**АНОТАЦІЯ**

**Структура та обсяг роботи.** Пояснювальна записка курсової роботи складається з 4 розділів, містить 5 рисунків, 39 таблиць, 3 додатки, 17 джерел.

**Мета.** Розробка ставить перед собою досягнення наступних цілей:

* Створення зборів з можливістю прив’язки банки з monobank;
* Пошук зборів за назвою та тегами;
* Звіти по зборах з можливістю завантаження файлів;
* Адміністрування додатку.

У розділі аналізу вимог були поставлені вимоги для програмного забезпечення.

У розділі моделювання програмного забезпечення було описано архітектуру програмного забезпечення та алгоритми вирішення прикладних задач.

У розділі аналіз якості були описані основні тест кейси, та стани системи після проведення тестування.

У розділі впровадження та супровід було описано процеси автоматизованого розгортання програмного забезпечення на виділеному сервері.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУВАННЯ, ЗБОРИ, ЗБІР КОШТІВ, ФАНДРЕЙЗИНГ

**ANNOTATION.**

**Structure and scope of the work.** The explanatory note of the course work consists of 4 chapters, contains 5 figures, 39 tables, 3 appendices, 17 sources.

**Objective.** The development aims to achieve the following goals:

* Creating fundraisings with the ability to link banks from monobank;
* Search for fundraisings by name and tags;
* Reports on fundraisings with the ability to download files;
* Administration of the application.

In the requirements analysis section, the software requirements were

established.

The software modeling section described the architecture of the software and the structural diagrams of the software.

The quality analysis section described the main test cases and states of the system after testing.

The deployment and maintenance section described the processes of deploying the software on a local server.

KEYWORDS: WEB APPLICATION, FUNDRAISING

**Зміст**

[**Перелік умовних позначень** 8](#_1fob9te)

[**Вступ** 9](#_3znysh7)

[**1**](#_2et92p0) **АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 10

[1.1](#_tyjcwt) Загальні положення 10

[1.2](#_3dy6vkm) Аналіз існуючих технологій та успішних IT-проєктів 11

[1.2.1](#_1t3h5sf) Аналіз відомих алгоритмічних та технічних рішень 13

[1.2.2](#_4d34og8) Аналіз допоміжних програмних засобів та засобів розробки 14

[1.2.3](#_2s8eyo1) Аналіз відомих програмних продуктів 14

[1.3](#_3rdcrjn) Аналіз вимог до програмного забезпечення 15

[1.3.1](#_lnxbz9) Розроблення функціональних вимог 22

[1.3.2](#_1ksv4uv) Розроблення нефункціональних вимог 24

[1.4](#_2jxsxqh) Постановка задачі 24

[Висновки до розділу 25](#_z337ya)

[**2**](#_3j2qqm3) **МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 26

[2.1](#_1y810tw) Моделювання та аналіз програмного забезпечення 26

[2.2](#_2xcytpi) Архітектура програмного забезпечення 27

[2.3](#_3whwml4) Конструювання програмного забезпечення 27

[2.4](#_2bn6wsx) Аналіз безпеки даних 35

[Висновки до розділу 35](#_qsh70q)

[**3**](#_3as4poj) **АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 37

[3.1](#_1pxezwc) Опис процесів тестування 37

[Висновки до розділу 42](#_49x2ik5)

[**4**](#_2p2csry) **ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 43

[4.1](#_147n2zr) Розгортання програмного забезпечення 43

[Висновки до розділу 43](#_3o7alnk)

[**ВИСНОВКИ** 44](#_23ckvvd)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 46](#_ihv636)

[**НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ** 51](#_1hmsyys)

[**ПІДСТАВА ДЛЯ РОЗРОБКИ** 52](#_41mghml)

[**ПРИЗНАЧЕННЯ РОЗРОБКИ** 53](#_2grqrue)

[**ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 54](#_vx1227)

[Вимоги до функціональних характеристик 54](#_3fwokq0)

[Для користувача: 54](#_1v1yuxt)

[Для адміністратора системи: 54](#_4f1mdlm)

[Вимоги до надійності 55](#_2u6wntf)

[Умови експлуатації 55](#_19c6y18)

[Вид обслуговування 55](#_3tbugp1)

[Обслуговуючий персонал 55](#_28h4qwu)

[Вимоги до складу і параметрів технічних засобів 55](#_nmf14n)

[Вимоги до інформаційної та програмної сумісності 55](#_37m2jsg)

[Вимоги до вхідних даних 55](#_1mrcu09)

[Вимоги до вихідних даних 55](#_46r0co2)

[Вимоги до мови розробки 55](#_2lwamvv)

[Вимоги до середовища розробки 56](#_111kx3o)

[Вимоги до маркування та пакування 56](#_3l18frh)

[Вимоги до транспортування та зберігання 56](#_206ipza)

[**ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ** 57](#_4k668n3)

[Попередній склад програмної документації 57](#_2zbgiuw)

[Спеціальні вимоги до програмної документації 57](#_1egqt2p)

[**СТАДІЇ І ЕТАПИ РОЗРОБКИ** 58](#_2dlolyb)

[**ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ** 59](#_sqyw64)

**Перелік умовних позначень**

| IDE | – | Integrated Development Environment – інтегроване середовище розробки. |
| --- | --- | --- |
| API | – | Application programming interface, прикладний програмний Інтерфейс |
| SDK | – | Software development kit |
| IT | – | Інформаційні технології |
| ER | – | Entity-Relation diagram |
| OC | – | Операційна система. |
| БД | – | База даних. |

**Вступ**

У сучасному світі зростає потреба в ефективних та зручних інструментах для збору коштів на різні потреби. Технологічний прогрес та поширення Інтернету відкрили нові, значно ширші можливості благодійності та збору коштів на різноманітні волонтерські, соціальні та громадські ініціативи.

Починаючи будь-який проект або громадську ініціативу, людина або організація неодмінно зіткнеться з потребою в залученні коштів на розвиток своїх починань, тому існує потреба в фандрейзинговій платформі, що задовільняє вимоги в зручності, простоті та швидкості взаємодії з нею.

Світові тенденції показують, що фандрейзингові платформи набирають популярності в різних сферах, включаючи військову сферу, благодійність, медицину, мистецтва та освіту. Ці інструменти стають необхідними для ефективного залучення фінансової підтримки від громадськості.

У рамках курсової роботи буде проведено дослідження потреб ринку фандрезйингу, аналіз існуючих веб-застосунків для збору коштів, а також розроблено концепцію нового веб-застосунку, який відповідатиме сучасним вимогам та трендам у даній сфері.

1. **АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**
   1. Загальні положення

Історія зборів коштів у мережі Інтернет розпочалася коли у 1997 році шанувальники британської рок-групи Marillion[1] без будь-якого залучення самої групи організували і провели інтернет-кампанію зі збору коштів для фінансування музичного туру групи усією територією США. Їм вдалося зібрати $60 000. Пізніше група використовувала такий спосіб для запису і просування кількох своїх альбомів.

Першим задокументованим фандрейзинговим сайтом є сайт для збору коштів на музичні ініціативи заснований компанією ArtistShare[2] у 2000-2001 році. У подальшому з розвитком та популяризації мережі Інтернет додатки для збору охоплювали все більше різних областей і досі не втратили свою актуальність.

У даному розділі будуть розглянуті проблеми, які існують у сфері створення веб-додатків для збору коштів, а також існуючі методи їх вирішення. Веб-додаток фандрейзингової платформи - це комплексна програмна система, яка надає інтерфейс для управління зборами, взаємодії з клієнтами та адміністрування.

Веб-додатки можуть надавати додаткові функції, такі як різні способи пожертвувань та поширень зборів, інтеграції зі сторонніми сервісами, звітність, категоризація зборів, що дозволяє легко знаходити збори на необхідні для користувачів теми та забезпечує використання їх коштів на бажані цілі.

Основні переваги фандрейзингових додатків:

* Уніфікація та централізація процесів: Додатки надають одну платформу для управління всіма аспектами збору коштів, що спрощує контроль над лістингами, пожертвами, звітністю та взаємодією з користувачами.
* Простота в надсиланні коштів: Більшість додатків дозволяють надсилати кошти в кілька кліків, часто навіть не вимагаючи реєстрації, що забезпечує швидку передачу коштів між відправником та отримувачем
* Безпека та конфіденційність: Використання сучасних технологій шифрування даних забезпечує високий рівень захисту і конфіденційності інформації.
* Доступність з будь-якого місця: Підтримка віддаленого доступу дозволяє користувачам збирати кошти та взаємодіяти з платформою з будь-якого пристрою та місця.
* Гнучкість та адаптивність: Можливість швидко адаптувати функціонал додатку під мінливі вимоги ринку та потреби користувачів.

Однак існують певні проблеми, які необхідно враховувати при розробці веб-додатків:

* Продуктивність: Веб-застосунки можуть потребувати значних ресурсів сервера, особливо при великій кількості користувачів, що може вплинути на продуктивність та швидкодію платформи.
* Безпека: Необхідність надійного захисту конфіденційної інформації та особистих даних користувачів.
* Сумісність: Важливість сумісності з різними пристроями та браузерами для комфортного користування різними групами користувачів.
* Відновлення та відміна змін: Розробка надійних механізмів для швидкого відновлення системи після помилок чи збоїв.
* Масштабованість: Здатність додатків ефективно масштабуватися для обслуговування зростаючої кількості користувачів та операцій.
  1. Аналіз існуючих технологій та успішних IT-проєктів

Проаналізуємо поточне програмне забезпечення в галузі та технічні рішення, які полегшать впровадження фандрейзингового додатку. Далі мова піде про додаткові програмні інструменти, засоби розробки та готові програмні рішення.

Ефективність веб-додатків оцінюється за різними критеріями, такими як продуктивність, що вимірюється швидкістю обробки запитів і здатністю витримувати великі навантаження. Надійність відображає стабільність системи та її здатність відновлювати роботу після збоїв. Масштабованість відображає здатність системи адаптуватися до збільшення кількості користувачів та даних. Безпека включає захист від несанкціонованого доступу та забезпечення конфіденційності даних. Іншими важливими аспектами є сумісність з різними пристроями та браузерами, здатність до інновацій та зручність для користувачів. Вартість власності також важлива, оскільки вона визначає витрати на підтримку та розвиток веб-додатку.

Розглянемо додатки аналоги у сфері фандрейзингу та відзначимо переваги та недоліки кожного з них.

GoFundMe[3]:

Переваги:

* Коментарі від користувачів
* Наявність відео туторіалів для користувачів

Недоліки:

* Недостатні можливості звітності
* Невелика кількість доступних категорій зборів
* Недостатньо широкі можливості пошуку зборів

WhyDonate[4]:

Переваги:

* Велика кількість доступних категорій зборів
* Широкі можливості пошуку зборів
* Можливість додавання звітів та фото

Недоліки:

* Неможливість додавання нових категорій
* Складна система додавання збору

Easyfunraising[5]:

Переваги:

* Можливість налаштування щомісячних списань на збори

Недоліки:

* Довга та складна процедура реєстрації
* Складна система створення збору
* Незручна система пошуку зборів

Fundly[6]:

Переваги:

* Коментарі від користувачів
* Можливість додавати відео до зборів

Недоліки:

* Застарілий інтерфейс
* Низька продуктивність роботи сайту
* Відсутність категоризації
* Відсутність звітності

Monobank[7]:

Переваги:

* Можливість налаштування щомісячних списань на збори
* Простота у використанні

Недоліки:

* Доступні тільки популярні фонди, про інші збори можна дізнатися лише зі сторонніх ресурсів
* Відсутність категоризації
* Відсутність звітності
  + 1. Аналіз відомих алгоритмічних та технічних рішень

Важливою частиною розробки даного додатку є інтеграція monobank. Ціллю цієї інтеграції є можливість прив’язати до збору банку з monobank. Реалізується це за допомогою HTTP запитів на API наданий monobank[8] для отримання даних про рахунки та банки користувача. Користувачу необхідно надати токен для отримання даних з особистого кабінету API клієнта monobank. Після надання токена користувач отримує можливість зв’язувати свої збори з банками.

* + 1. Аналіз допоміжних програмних засобів та засобів розробки

У цьому розділі оглянемо та проаналізуємо допоміжні програмні

засоби, та засоби розробки, що використовуються у даному проекті. Для розробки проекту було використано фреймворк ASP.NET Core[9] на мові програмування C#[10], написання програми було виконано в IDE Rider[11].

C# - це високорівнева об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона славиться своєю елегантністю синтаксису та високою продуктивністю. Величезний плюс C# - це його інтеграція з екосистемою .NET[12], яка надає доступ до широкого спектру бібліотек і інструментів для розробки. Однією з суттєвих переваг C# є його вбудована безпека, включаючи систему управління пам'яттю, що допомагає уникнути багатьох типових помилок, які можуть виникнути в процесі розробки програм.

ASP.NET Core - це потужний фреймворк, який використовується для розробки веб-додатків та веб-API на мові програмування C#. Основними його перевагами є висока продуктивність та масштабованість, платформонезалежність, безпека, модульність та гнучкість.

Rider від JetBrains - це одне з найпопулярніших IDE для C#, що пропонує широкий спектр функцій для поліпшення продуктивності розробки, включаючи розширену підтримку коду, інтеграцію з системами контролю версій, дебагінг, та велику кількість плагінів.

* + 1. Аналіз відомих програмних продуктів

Для порівняння курсової роботи з існуючими аналогами можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогом

|  | GoFundMe | WhyDonate | easyfundraising | Fundly | monobank | EFund |
| --- | --- | --- | --- | --- | --- | --- |
| Можливість надіслати кошти без реєстрації | + | + | - | + | + | + |
| Можливість додавання звітів | + | + | - | - | - | + |
| Можливість завантаження файлів звітів | - | + | - | - | - | + |
| Можливість додати банку monobank | - | - | - | - | + | + |

Після проведення детального аналізу кожного з додатків можна сказати що кожен з них відкриває свої можливості та має унікальні особливості, та кожен з них вирізняється наявністю та відсутністю певного функціоналу.

Узагальнюючи отримані дані, можна сказати, що додаток “EFund” має всі найважливіші для даних додатків функції та забезпечує наявність функціоналу якого не вистачає деяким додаткам.

* 1. Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є створення зборів для накопичення коштів на потреби користувачів, більше функцій можна побачити на рисунку 1.1

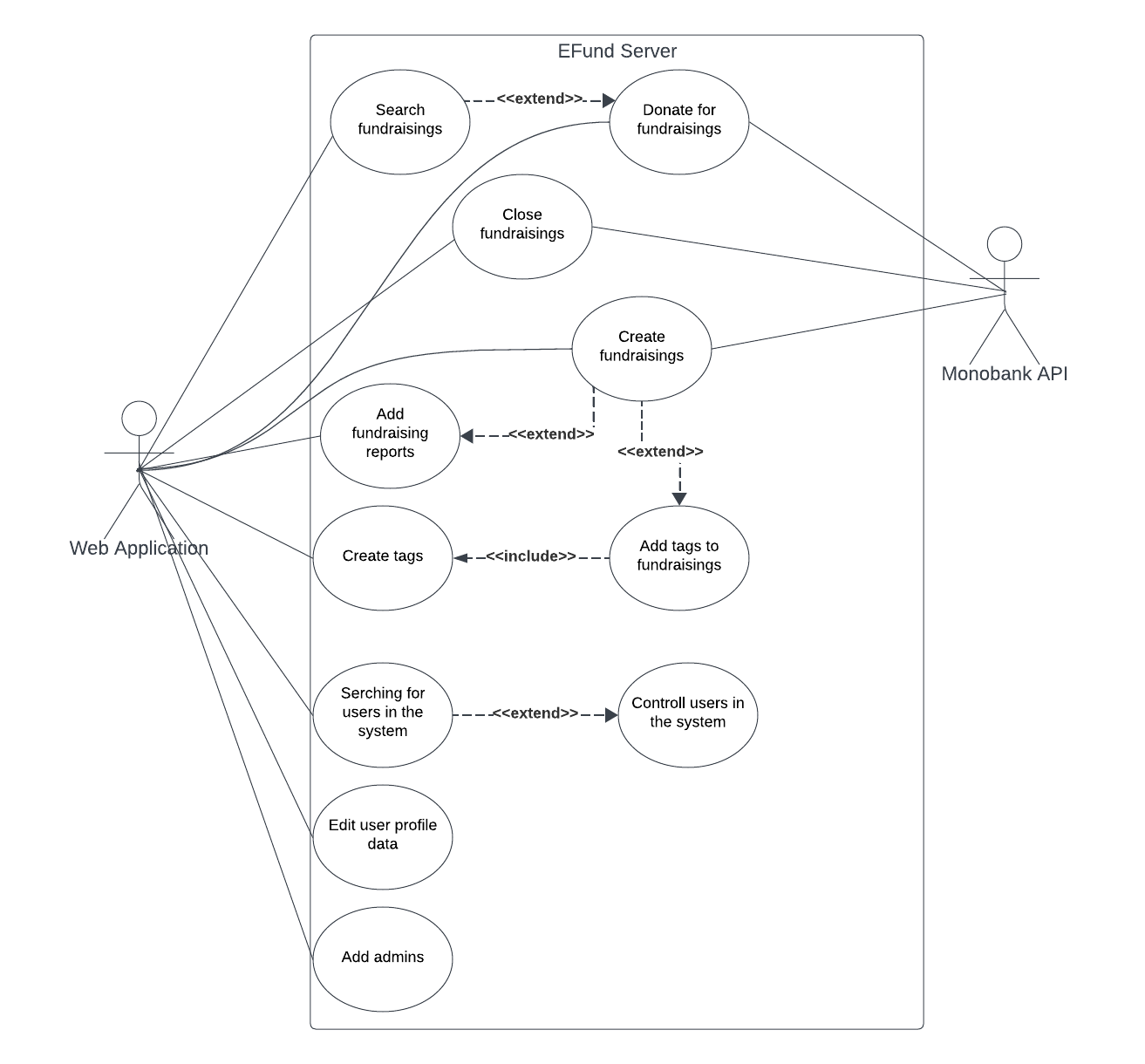


Рисунок 1.1 – Діаграма варіантів використання

В таблицях 1.2 - 1.13 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

| Use case name | Search fundraisings |
| --- | --- |
| Use case ID | UC-01 |
| Goals | Отримати список зборів |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер приймає параметри пошуку з запиту * У випадку некоректності параметрів сервер повертає помилку зі статус кодом 400 * Сервер повертає список зі статус кодом 200 |
| Extension | - |
| Post-Condition | - |

Таблиця 1.3 - Варіант використання UC-2

| Use case name | Donate for fundraisings |
| --- | --- |
| Use case ID | UC-02 |
| Goals | Надіслати кошти на збір |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Як частина даних збору повертається посилання на банку monobank для надсилання коштів |
| Extension | - |
| Post-Condition | - |

Таблиця 1.4 - Варіант використання UC-3

| Use case name | Create fundraisings |
| --- | --- |
| Use case ID | UC-03 |
| Goals | Створення збору |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер перевіряє дані збору з запиту * У випадку некоректності даних сервер повертає помилку зі статус кодом 400 * Сервер створює збір в базі даних * Сервер повертає створений збір зі статус кодом 201 |
| Extension | - |
| Post-Condition | - |

Таблиця 1.5 - Варіант використання UC-4

| Use case name | Add tags to fundraisings |
| --- | --- |
| Use case ID | UC-04 |
| Goals | Додавання тегів до збору |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер перевіряє дані з запиту * У випадку некоректності даних сервер повертає помилку зі статус кодом 400 * Сервер додає теги до збору * Сервер повертає збір з кодом 200 |
| Extension | - |
| Post-Condition | - |

Таблиця 1.6 - Варіант використання UC-5

| Use case name | Add fundraising reports |
| --- | --- |
| Use case ID | UC-05 |
| Goals | Додавання звітів до збору |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер перевіряє дані з запиту * У випадку некоректності даних сервер повертає помилку зі статус кодом 400 * Сервер зберігає звіт в бд * Сервер повертає створений звіт з кодом 201 |
| Extension | - |
| Post-Condition | - |

Таблиця 1.7 - Варіант використання UC-6

| Use case name | Create tag |
| --- | --- |
| Use case ID | UC-06 |
| Goals | Створення тегу |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер перевіряє дані з запиту * У випадку некоректності даних сервер повертає помилку зі статус кодом 400 * Сервер зберігає тег в бд * Сервер повертає створений звіт з кодом 201 |
| Extension | - |
| Post-Condition | - |

Таблиця 1.8 - Варіант використання UC-7

| Use case name | Close fundraisings |
| --- | --- |
| Use case ID | UC-07 |
| Goals | Закриття збору |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер перевіряє дані з запиту * У випадку некоректності даних сервер повертає помилку зі статус кодом 400 * Сервер закриває збір * Сервер повертає статус код 204 |
| Extension | - |
| Post-Condition | - |

Таблиця 1.9 - Варіант використання UC-8

| Use case name | Searching for users in the system |
| --- | --- |
| Use case ID | UC-08 |
| Goals | Пошук користувачів у системі |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер перевіряє параметри з запиту * У випадку некоректності параметрів сервер повертає помилку зі статус кодом 400 * Сервер повертає список користувачів зі статус кодом 200 |
| Extension | - |
| Post-Condition | - |

Таблиця 1.10 - Варіант використання UC-9

| Use case name | Control users in the system |
| --- | --- |
| Use case ID | UC-09 |
| Goals | Контроль користувачів у системі |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер перевіряє параметри з запиту * У випадку некоректності параметрів сервер повертає помилку зі статус кодом 400 * Сервер блокує або розблоковує користувача * Сервер повертає статус код 204 |
| Extension | - |
| Post-Condition | - |

Таблиця 1.11 - Варіант використання UC-10

| Use case name | Edit user profile data |
| --- | --- |
| Use case ID | UC-10 |
| Goals | Зміна даних користувача |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер перевіряє параметри з запиту * У випадку некоректності параметрів сервер повертає помилку зі статус кодом 400 * Сервер змінює дані користувача * Сервер повертає зміненого користувача з статус кодом 200 |
| Extension | - |
| Post-Condition | - |

Таблиця 1.12 - Варіант використання UC-11

| Use case name | Authorization |
| --- | --- |
| Use case ID | UC-11 |
| Goals | Авторизація користувачів та адміністраторів |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер перевіряє параметри з запиту * У випадку некоректності параметрів сервер повертає помилку зі статус кодом 400 * Сервер авторизує користувача * Сервер повертає токен авторизації з статус кодом 200 |
| Extension | - |
| Post-Condition | - |

Таблиця 1.13 - Варіант використання UC-12

| Use case name | Add admins |
| --- | --- |
| Use case ID | UC-12 |
| Goals | Додавання адміністраторів |
| Actors | Веб-додаток |
| Trigger | Веб-додаток надсилає запит до сервера |
| Pre-conditions | Встановлене з’єднання |
| Flow of Events | * Сервер перевіряє параметри з запиту * У випадку некоректності параметрів сервер повертає помилку зі статус кодом 400 * Сервер додає адміністартора * Сервер повертає статус код 204 |
| Extension | - |
| Post-Condition | - |

* + 1. Розроблення функціональних вимог

В таблиці 1.14 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити в таблиці 1.15

Таблиця 1.14 Функціональні вимоги до додатку

| Номер | Назва | Опис |
| --- | --- | --- |
| FR-1 | Авторизація | Користувач або адміністратор повинні мати змогу зареєструватися та ввійти у свій профіль використовуючи імейл та пароль. |
| FR-2 | Авторизація через Google | Користувач або адміністратор повинні мати змогу зареєструватися та ввійти у свій профіль через Google. |
| FR-3 | Зміна даних профілю користувача | Зміна імені, імейлу, паролю та фото користувача |
| FR-4 | Створення збору | Створення збору з назвою, описом та фото |
| FR-5 | Прив’язка банки monobank | Можливість прив’язати до збору банку з monobank |
| FR-6 | Додавання тегів до збору | Можливість додавати до збору категоризаційні теги для пошуку |
| FR-7 | Пошук зборів | Пошук зборів по назві та тегах |
| FR-8 | Додавання звітів по збору | Можливість додавати до збору звіти з назвою, описом, фото та файлами |
| FR-9 | Отримання списку користувачів у системі | Можливість отримання адміністратором повного списку користувачів системи |
| FR-10 | Контроль користувачів у системі | Можливість блокувати та розблокувати користувачів адміністратором |
| FR-11 | Додавання адміністраторів | Можливість робити існуючих користувачів адміністраторами та запрошувати адміністраторів через імейл |
| FR-12 | Закриття зборів | Можливість закриття збору користувачем по накопиченню необхідної суми або адміністратором |
| FR-13 | Перехід на банку monobank | Можливість перейти на банку monobank що прив’язана до збору |

Таблиця 1.15 Матриця трасування вимог

|  | UC-01 | UC-02 | UC-03 | UC-04 | UC-05 | UC-06 | UC-07 | UC-08 | UC-09 | UC-10 | UC-11 | UC-12 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FR-1 |  |  |  |  |  |  |  |  |  |  | + |  |
| FR-2 |  |  |  |  |  |  |  |  |  |  | + |  |
| FR-3 |  |  |  |  |  |  |  |  |  | + |  |  |
| FR-4 |  |  | + |  |  |  |  |  |  |  |  |  |
| FR-5 |  |  |  |  |  |  |  |  |  |  |  |  |
| FR-6 |  |  |  | + |  | + |  |  |  |  |  |  |
| FR-7 | + |  |  |  |  |  |  |  |  |  |  |  |
| FR-8 |  |  |  |  | + |  |  |  |  |  |  |  |
| FR-9 |  |  |  |  |  |  |  | + |  |  |  |  |
| FR-10 |  |  |  |  |  |  |  |  | + |  |  |  |
| FR-11 |  |  |  |  |  |  |  |  |  |  |  | + |
| FR-12 |  |  |  |  |  |  | + |  |  |  |  |  |
| FR-13 |  | + |  |  |  |  |  |  |  |  |  |  |

* + 1. Розроблення нефункціональних вимог
* Сервер має підтримувати асинхронну обробку запитів.
* Сервер повинен бути адаптивний до внесення змін або нового функціоналу, не змінюючи при цьому спілкування з клієнтом
* Сервер повинен гарантувати конфіденційність інформації та захищати від несанкціонованого доступу під час операцій
  1. Постановка задачі

На основі ретельного аналізу предметної області дослідження було визначено перелік вимог, які є важливою складовою розробки програмної системи. Аналіз цих вимог є критично важливим кроком у формуванні обсягу та характеристик майбутнього продукту. Ці вимоги охоплюють як функціональні, так і нефункціональні аспекти, необхідні для успішної реалізації бізнес-цілей та задоволення потреб користувачів. Цей аналіз створює міцний фундамент для наступних етапів проектування та розробки програмної системи.

* Авторизація та профіль користувача

Реєстрація та логін

Реєстрація та логін через Google

Профіль користувача та зміна його даних

* Інтеграція Monobank

Можливість діставати банки користувача

* Управління тегами

Створення та пошук тегів

* Управління зборам

Створення, зміна, пошук та видалення зборів

* Управління звітами по зборах

Створення, зміна та видалення звітів

* Адміністрування програми

Запрошення адміністраторів

Закриття зборів

Блокування та розблокування користувачів

Цей перелік вимог є основою для розробки програмної системи, яка відповідає потребам та очікуванням користувачів у даній предметній області.

Висновки до розділу

У цьому розділі було уточнено постановку задачі, розглянуто існуючі подібні додатки фандрейзингових платформ. Крім того, було проведено аналіз вимог, побудовано таблицю для відповідного use-case та кожної з функціональних вимог, що дало змогу більш точно оцінити вимоги до мови програмування та технологій, що мають бути застосовані. В результаті було обрано C# як мову програмування та ASP.NET Core як фреймворк для розробки серверної частини додатку, тому що ці інструменти дозволяють побудувати надійні високопродуктивні та безпечні додатки які легко підтримувати та масштабувати.

1. **МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**
   1. Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу програмного забезпечення використовується UML Sequence Diagram (рисунок 2.1).

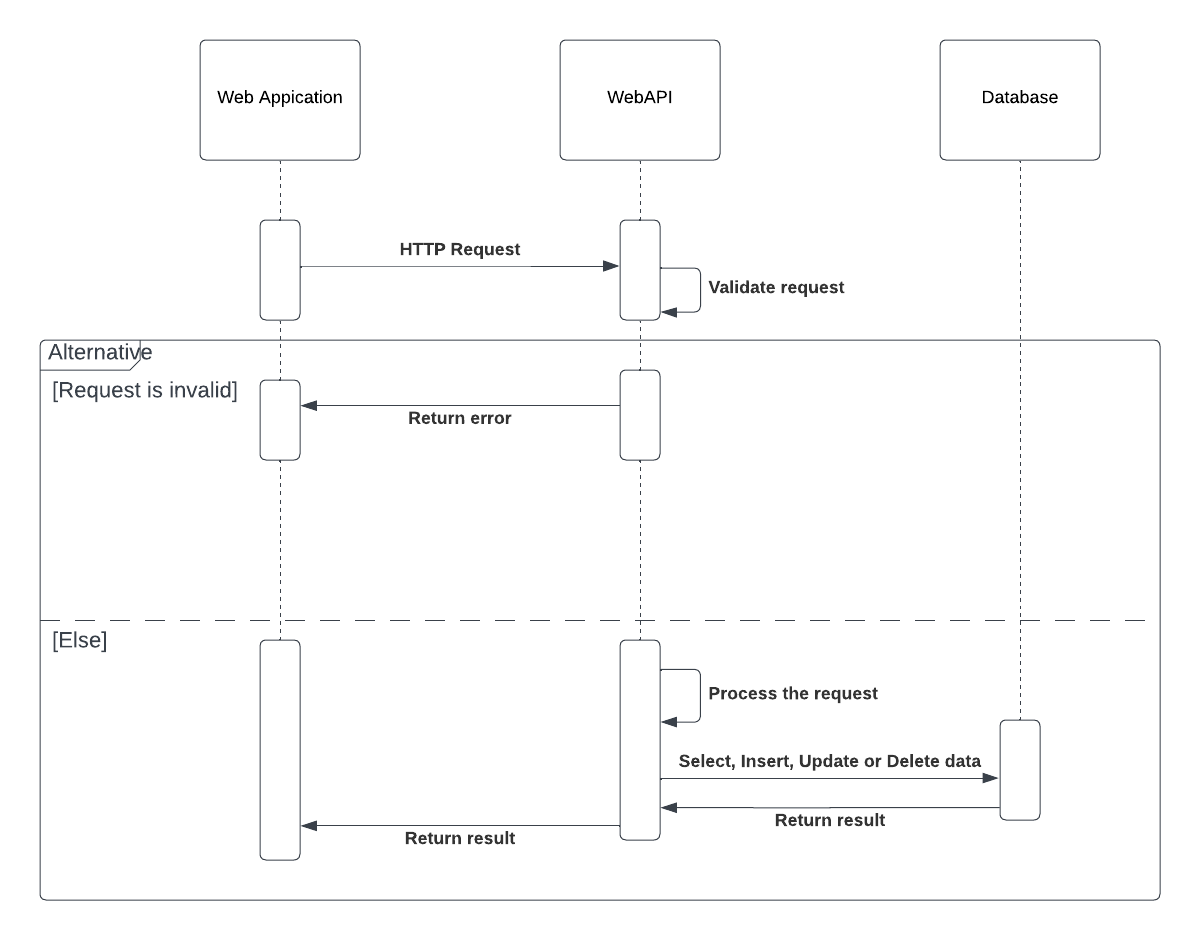


Рисунок 2.1 – Діаграма варіантів використання

Опис послідовності базового потоку комунікації сервера та веб-додатку:

* Веб-додаток надсилає запит
* Сервер перевіряє коректність вхідних параметрів та даних
* Якщо дані не коректні повертає помилку з відповідним статус кодом
* Якщо дані коректні обробляє запит
* Сервер повертає результат з відповідним статус кодом
  1. Архітектура програмного забезпечення

Сервер фандрейзингової платформи базується на n-рівневій архітектурі. Основною перевагою цієї архітектури є розділення різних рівнів функціональності, таких як презентація, бізнес-логіка, доступ до даних і зберігання даних, що сприяє гнучкості та масштабованості додатку.

N-рівнева архітектура[13] дозволяє розробникам самостійно оновлювати або покращувати різні частини системи без необхідності переписувати весь додаток, що особливо корисно для додатків з великою кількістю користувачів і великим обсягом даних.

Серверний рівень веб-додатку включає наступні рівні:

* API-рівень, який приймає вхідні запити та надсилає відповіді;
* Рівень Бізнес-логіки, яка обробляє бізнес-правила та операції;
* Рівень доступу до даних, який взаємодіє з базою даних.
* Спільний рівень, що містить моделі, які є спільними для всіх рівнів

Серверна частина реалізована з використанням REST-архітектури. авторизації використовується токен JWT[14].

Схема архітектури:

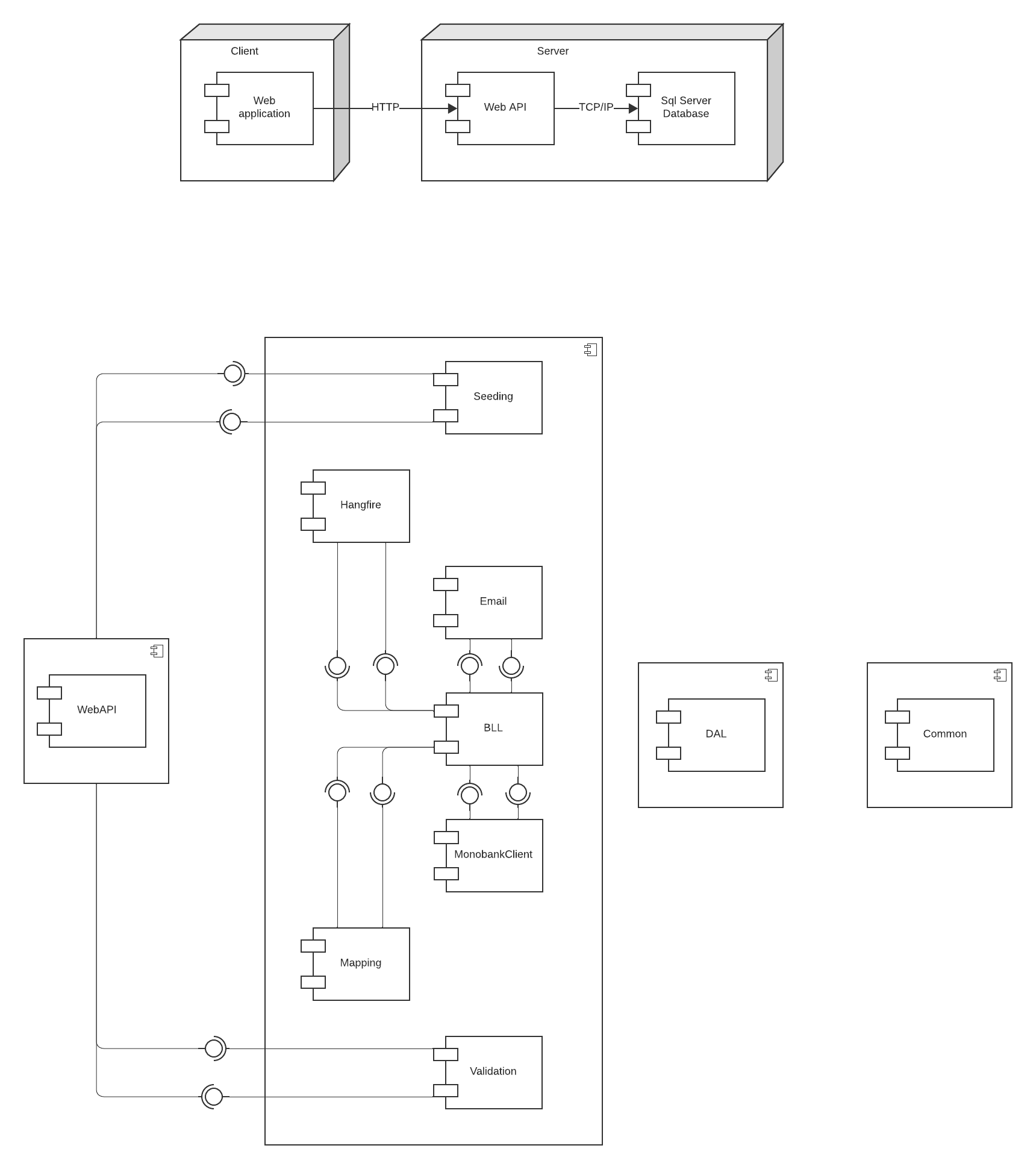


Рисунок 2.2 – Діаграма архітектури

* 1. Конструювання програмного забезпечення

При розробці програмного забезпечення було використано MS Sql Server[15] у якості системи управління бази даних. В базі зберігаються дані користувачів, зборів, дані звітів та всі інші використані сутності. В якості ORM було використано Entity Framework Core[16], який є найпопулярнішим ORM у платформі .NET та вирізняється найкращим балансом зручності, продуктивності та швидкості розробки серед аналогів. Також було інтегровано monobank API за допомогою HTTP запитів на надані ендпоінти для отримання даних користувача.

Деталізована діаграма архітектури сервера зображена на рисунку 2.3

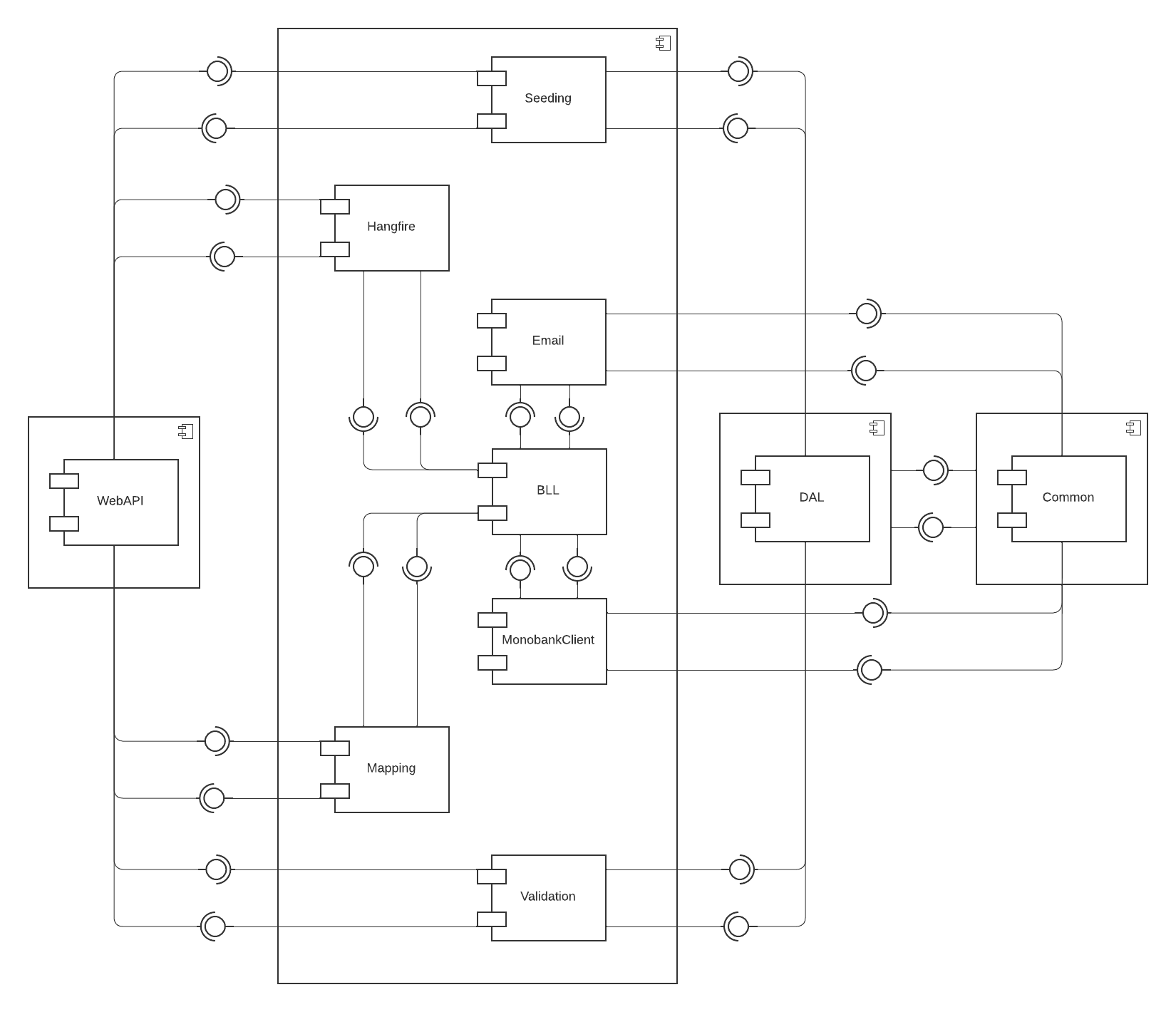


Рисунок 2.3 – Діаграма архітектури сервера

Діаграма бази даних зображена на рисунку 2.4

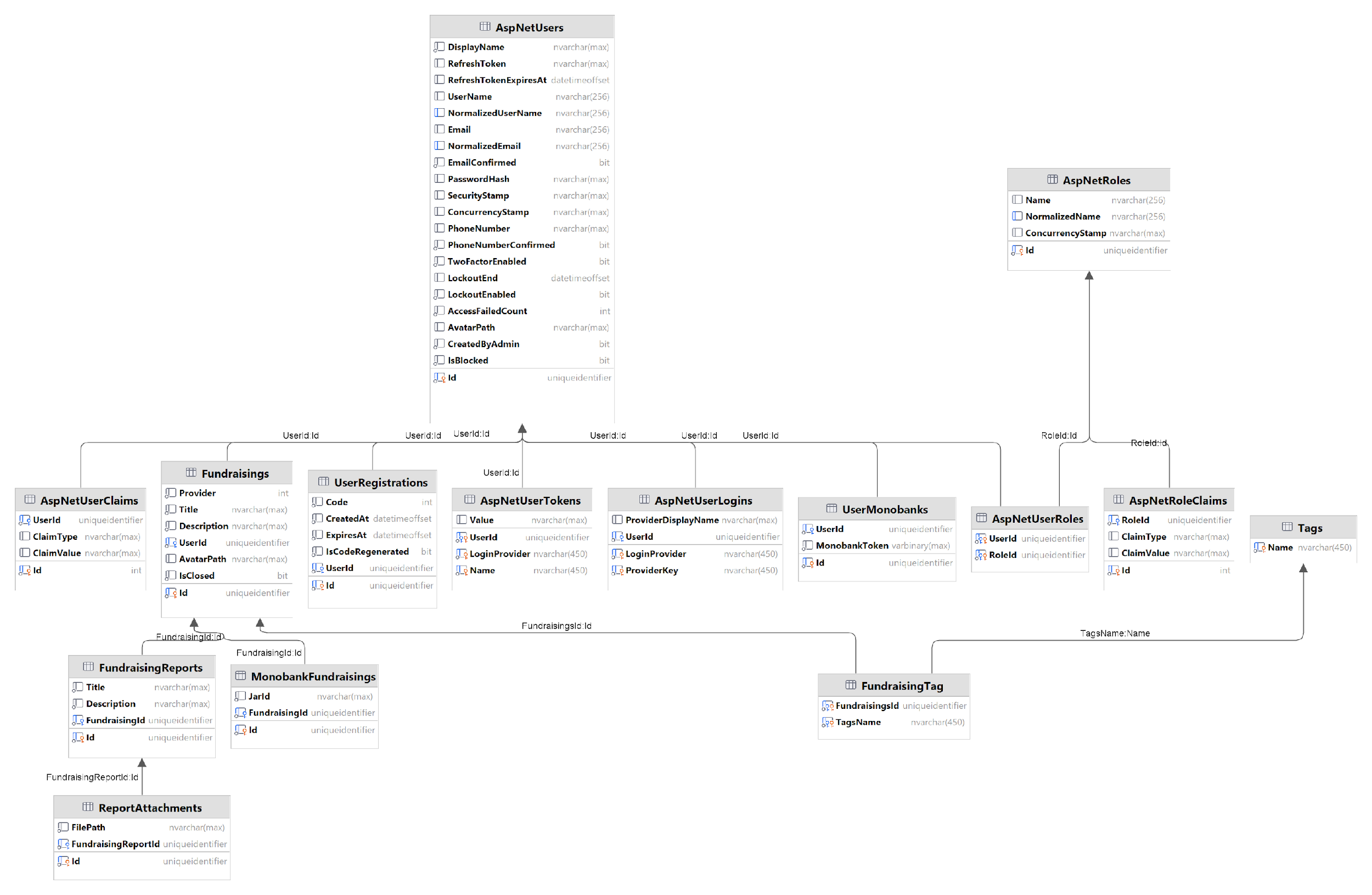


Рисунок 2.4 – Діаграма бази даних

Таблиця 2.1 – Опис таблиці AspNetUsers

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| AspNetUsers | Id | uniqueidentifier | Ідентифікатор користувача |
| DisplayName | nvarchar | Повне ім’я користувача |
| RefreshToken | nvarchar | Рефреш токен користувача |
| RefreshTokenExpiresAt | datetime | Час коли рефреш токен стане недійсним |
| Email | nvarchar | Імейл користувача |
| NormalizedEmail | nvarchar | Нормазіований імейл користувача |
| EmailConfirmed | bit | Чи підтверджений імейл |
| PasswordHash | nvarchar | Захешований пароль |
| SecurityStamp | nvarchar | Штамп безпеки користувача |
| ConcurrencyStamp | nvarchar | Штамп безпеки конкурентних задач користувача |
| AvatarPath | nvarchar | Шлях до аватару користувача |
| CreatedByAdmin | bit | Чи був користувач створений адміністратором |
| IsBlocked | bit | Чи заблокований користувач |

Таблиця 2.2 – Опис таблиці AspNetUserLogins

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| AspNetUserLogins | ProviderDisplayName | nvarchar | Ім’я стороннього логіну |
| UserId | uniqueidentifier | Ідентифікатор користувача |
| LoginProvider | nvarchar | Провайдер стороннього логіну |
| ProviderKey | nvarchar | Ключ стороннього логіну |

Таблиця 2.3 – Опис таблиці AspNetUserClaims

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| AspNetUserClaims | Id | int | Ідентифікатор клеімс |
| UserId | uniqueidentifier | Ідентифікатор користувача |
| ClaimType | nvarchar | Тип клеімс |
| ClaimValue | nvarchar | Значення клеімс |

Таблиця 2.4 – Опис таблиці AspNetRoles

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| AspNetRoles | Id | uniqueidentifier | Ідентифікатор ролі |
| Name | nvarchar | Ім’я ролі |
| NormalazideName | nvarchar | Нормалізоване ім’я ролі |
| ConcurrencyStamp | nvarchar | Штамп безпеки конкурентних задач ролі |

Таблиця 2.5 – Опис таблиці AspNetUserRoles

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| AspNetUserRoles | UserId | uniqueidentifier | Ідентифікатор користувача |
| RoleId | uniqueidentifier | Ідентифікатор ролі |

Таблиця 2.6 – Опис таблиці AspNetUserRoleClaims

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| AspNetRoleClaims | Id | int | Ідентифікатор |
| RoleId | uniqueidentifier | Ідентифікатор ролі |
| ClaimType | nvarchar | Тип клеімс |
| ClaimValue | nvarchar | Значення клеімс |

Таблиця 2.7 – Опис таблиці UserRegistrations

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| UserRegisrations | Id | uniqueidentifier | Ідентифікатор |
| Code | int | Код підтвердження імейлу |
| CreatedAt | datetimeoffset | Час створення |
| ExpiresAt | datetimeoffset | Час коли код стане недійсним |
| IsCodeRegenerated | bit | Чи був код перегенерований |
| UserId | uniqueidentifier | Ідентифікатор користувача |

Таблиця 2.8 – Опис таблиці UserMonobanks

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| UserMonobanks | Id | uniqueidentifier | Ідентифікатор |
| UserId | uniqueidentifier | Ідентифікатор користувача |
| MonobankToken | varbinary | Захешований токен monobank |

Таблиця 2.9 – Опис таблиці Fundraisings

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| Funndraisings | Id | uniqueidentifier | Ідентифікатор збору |
| Provider | int | Провайдер збору |
| Title | nvarchar | Назва збору |
| Description | nvarchar | Опис |
| UserId | uniqueidentifier | Ідентифікатор користувача |
| AvatarPath | nvarchar | Шлях до аватару збору |
| IsClosed | bit | Чи збір закритий |

Таблиця 2.10 – Опис таблиці MonobankFundraisings

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| MonobankFundraisings | Id | uniqueidentifier | Ідентифікатор |
| Jarld | nvarchar | Ідентифікатор банки з monobank |
| FundraisingId | uniqueidentifier | Ідентифікатор збору |

Таблиця 2.11 – Опис таблиці FundraisingReports

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| FundraisingReports | Id | uniqueidentifier | Ідентифікатор звіту |
| Title | nvarchar | Назва звіту |
| Description | nvarchar | Опис |
| FundraisingId | uniqueidentifier | Ідентифікатор збору |

Таблиця 2.12 – Опис таблиці Tags

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| Tags | Name | nvarchar | Назва тегу |

Таблиця 2.13 – Опис таблиці FundraisingTag

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| FundraisingTag | FundraisingId | uniqueidentifier | Ідентифікатор збору |
| TagsName | nvarchar | Назва тегу |

Таблиця 2.14 – Опис таблиці ReportAttachments

| Таблиця | Назва поля | Тип даних | Опис |
| --- | --- | --- | --- |
| ReportAttachments | Id | uniqueidentifier | Ідентифікатор файлу |
| FilePath | nvarchar | Шлях до файлу звіту |
| FundraisingReportId | uniqueidentifier | Ідентифікатор звіту |

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.15.

Таблиця 2.15 – Опис утиліт

| №  п/п | Назва утиліти | Опис застосування |
| --- | --- | --- |
| 1 | Rider | Головне середовище розробки програмного забезпечення серверної частини курсової роботи. |
| 2 | Swagger | Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів. |
| 3 | DataGrip | Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних. |

* 1. Аналіз безпеки даних

Оскільки користувач надає особисті дані такі як пароль та токен з monbank, необхідно забезпечити безпеку зберігання цих даних. Ці дані зберігаються захешованими у базі даних, для роботи з паролями використовується одностороннє хешування, для роботи з токенами monobank – двостороннє. Завдяки хешуванню, ми забезпечуємо що навіть якщо хтось отримає доступ до даних, то він нічого не зможе зробити з ними, оскільки вразливі дані не зберігаються в необхідному форматі.

Висновки до розділу

Цей розділ містить детальний аналіз та опис ключових аспектів

розвитку системи. Розділ охоплює широкий спектр елементів, від архітектурного планування до використання конкретних інструментів і технологій. Тут описано діаграму послідовності, яка демонструє взаємодію між різними компонентами системи, зокрема процеси запитів і відповідей між клієнтом і сервером. Ця діаграма дає чітке розуміння потоку даних і взаємодії всередині системи. Значну увагу було приділено архітектурі сервера. Було також створено та описано діаграму компонентів сервера, яка візуально представляє структуру та взаємозв'язки між основними компонентами сервера. Ця діаграма є важливим інструментом для розуміння архітектури системи та її компонентів. Також було розроблено діаграму бази даних та опис кожної з таблиць. Загалом, розділ дає ґрунтовне розуміння того, як побудована система, підкреслюючи важливість кожного компонента та інструмента у розробці додатку фандрейзингової платформи.

1. **АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**
   1. Опис процесів тестування

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.9.

Таблиця 3.1 – Тест 1.1

| Тест | Sign up |
| --- | --- |
| Модуль | POST request |
| Номер тесту | 1.1 |
| Початковий стан системи | Сервер очікує на запит |
| Вхідні данні | Електронна пошта, пароль, ім’я користувача |
| Опис проведення тесту | В тілі запиту отримуємо необхідні поля, а саме імейл, який не був зареєстрований у системі, пароль, що містить одну малу та велику літери, цифри та спеціальний символ, ім’я користувача |
| Очікуваний результат | Повертається ідентифікатор створеного користувача зі статус кодом 200, на імейл приходить код підтвердження |
| Фактичний результат | Повертається ідентифікатор створеного користувача зі статус кодом 200, на імейл приходить код підтвердження |

Таблиця 3.2 – Тест 1.2

| Тест | Search fundraisings |
| --- | --- |
| Модуль | POST request |
| Номер тесту | 1.2 |
| Початковий стан системи | Сервер очікує на запит |
| Вхідні данні | Назва збору, теги збору |
| Опис проведення тесту | В тілі запиту отримуємо необхідні параметри, а саме назву збору та масив тегів |
| Очікуваний результат | Повертаються збори що співпадають по імені та тегам зі статус кодом 200 |
| Фактичний результат | Повертаються збори що співпадають по імені та тегам зі статус кодом 200 |

Таблиця 3.3 – Тест 1.3

| Тест | Create fundraising |
| --- | --- |
| Модуль | POST request |
| Номер тесту | 1.3 |
| Початковий стан системи | Сервер очікує на запит |
| Вхідні данні | Назва збору, опис збору, теги збору, ідентифікатор банки з monobank |
| Опис проведення тесту | В тілі запиту отримуємо необхідні параметри, а саме назву збору, опис збору, масив тегів та ідентифікатор існуючої банки з monobank |
| Очікуваний результат | Повертається створений збір зі статус кодом 201 |
| Фактичний результат | Повертається створений збір зі статус кодом 201 |

Таблиця 3.4 – Тест 1.4

| Тест | Add valid avatar |
| --- | --- |
| Модуль | POST request |
| Номер тесту | 1.4 |
| Початковий стан системи | Сервер очікує на запит |
| Вхідні данні | Ідентифікатор збору, файл із розширенням .png |
| Опис проведення тесту | З посилання отримуємо ідентифікатор збору та з тіла запиту файл з аватаром, що повинен мати розширення .png, .jpg, .jpeg або .gif |
| Очікуваний результат | Повертається статус код 204. При наступному отриманні збору отримуємо посилання на доданий аватар в тілі відповіді |
| Фактичний результат | Повертається статус код 204. При наступному отриманні збору отримуємо посилання на доданий аватар в тілі відповіді. |

Таблиця 3.5 – Тест 1.5

| Тест | Add invalid avatar |
| --- | --- |
| Модуль | POST request |
| Номер тесту | 1.5 |
| Початковий стан системи | Сервер очікує на запит |
| Вхідні данні | Ідентифікатор збору, файл із розширенням .pdf |
| Опис проведення тесту | З посилання отримуємо ідентифікатор збору та з тіла запиту файл з аватаром, що повинен мати розширення .png, .jpg, .jpeg або .gif |
| Очікуваний результат | Повертається статус код 400 |
| Фактичний результат | Повертається статус код 400 |

Таблиця 3.6 – Тест 1.6

| Тест | Delete avatar |
| --- | --- |
| Модуль | DELETE request |
| Номер тесту | 1.6 |
| Початковий стан системи | Сервер очікує на запит |
| Вхідні данні | Ідентифікатор збору |
| Опис проведення тесту | З посилання отримуємо ідентифікатор збору |
| Очікуваний результат | Повертається статус код 204. При наступному отриманні збору отримуємо посилання на стандартний аватар в тілі відповіді |
| Фактичний результат | Повертається статус код 204. При наступному отриманні збору отримуємо посилання на стандартний аватар в тілі відповіді |

Таблиця 3.7 – Тест 1.7

| Тест | Invite admin |
| --- | --- |
| Модуль | POST request |
| Номер тесту | 1.7 |
| Початковий стан системи | Сервер очікує на запит |
| Вхідні данні | Імейл запрошуваного адміністратора |
| Опис проведення тесту | З тіла запиту отримуємо імейл запрошуваного адміністратора який не був зареєстрований у системі |
| Очікуваний результат | Повертається статус код 204. На вказаний імейл надсилається посилання на реєстрацію |
| Фактичний результат | Повертається статус код 204. На вказаний імейл надсилається посилання на реєстрацію |

Таблиця 3.8 – Тест 1.8

| Тест | Block user |
| --- | --- |
| Модуль | POST request |
| Номер тесту | 1.8 |
| Початковий стан системи | Сервер очікує на запит |
| Вхідні данні | Ідентифікатор користувача |
| Опис проведення тесту | З тіла запиту отримуємо ідентифікатор користувача |
| Очікуваний результат | Повертається статус код 204. Користувач не може створювати та змінювати збори, додавати звіти |
| Фактичний результат | Повертається статус код 204. Користувач не може створювати та змінювати збори, додавати звіти |

Таблиця 3.9 – Тест 1.9

| Тест | Add report attachments |
| --- | --- |
| Модуль | POST request |
| Номер тесту | 1.9 |
| Початковий стан системи | Сервер очікує на запит |
| Вхідні данні | Ідентифікатор звіту, файли звітів з розширенням .pdf, .jpg, .xslx |
| Опис проведення тесту | З посилання отримуємо ідентифікатор користувача. З тіла запиту отримуємо файл, що повинен мати розширення .png, .jpg, .jpeg, .gif, .pdf, .doc, .docx, .xsl або xslx |
| .Очікуваний результат | Повертається статус код 204. При наступному отриманні звіту повертаються посилання на всі додані файли |
| Фактичний результат | Повертається статус код 204. При наступному отриманні звіту повертаються посилання на всі додані файли |

Висновки до розділу

Отже, у даному розділі було проведене мануальне тестування основного функціоналу додатку через запити до серверу, завдяки чому ми впевнилися в надійності системи та переконалися що запити обробляються вірно та при введенні некоректних даних помилки правильно обробляються та повертають відповідні статус коди помилок. Завдяки тестуванню було покращено якість та надійність системи, перевірено її стійкість до хибних даних та правильність виконання запитів та забезпечення повноцінного очікуваного алгоритму дій та відповідей програми на певні запити.

1. **ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**
   1. Розгортання програмного забезпечення

Для розгортання серверу необхідно виконати наступні кроки

* Підготовка середовища

Встановлення .NET 7 та MS Sql Server

* Склонувати репозиторій застосунку

git clone

* Запустити додаток

dotnet run

Висновки до розділу

У даному розділі було висвітлено процес локального розгортання серверу для додатку зі збору коштів на localhost. При локальному розгортанні серверу ми отримуємо повний контроль над середовищем та конфігурацією, а також відсутність затримок, пов'язаних із зовнішнім хостингом. Цей процес підтвердив, що сервер можна ефективно встановити і зробити придатним до використання ззовні без великих витрат або складнощів, що забезпечує доступність, зручність та гнучкість для користувачів.

**ВИСНОВКИ**

В результаті виконання курсової роботи було розроблено веб-додаток для збору коштів, частиною якого є сервер на базі ASP.NET Core що включає в себе інтеграцію зовнішнього API monobank для отримання банок користувача, які можна прив’язувати до зборів. В якості середовища розробки обрано Rider, через його зручність, та розширену підтримку коду. У якості БД використано MS Sql Server.

Оцінка одержаних результатів:

Одержані результати демонструють що система відповідає сучасному рівню технічних і наукових завдань у галузі розробки програмного забезпечення. У системі успішно інтегрована БД, велика кількість бібліотек та сторонній API, завдяки чому забезпечується успішність управління зборами.

Ступінь впровадження та сфери використання

Даний додаток має значний потенціал для впровадження у сферах, де часто необхідні збори коштів, а саме волонтерські проєкти та платформи, громадські організації, збори для військових та стартапи.

Доцільність проведення досліджень:

Наявний великий потенціал для подальшого розвитку проекту, а саме інтеграцї інших сторонніх сервісів, платіжних систем, розширення функціоналу взаємодії між користувачами.

Стан вирішення поставлених задач

Всі поставлені у курсові роботі задачі були вирішені, система показала себе як продуктивну та надійну. Курсова робота надала надійний фундамент для подальших досліджень та розвитку даної галузі.

Наукова новизна роботи полягає в наступному

Вперше:

* інтегровано monobank API, що дозволило прив’язувати банки до зборів

Модифіковано:

Набуло подальший розвиток:

* покращена система звітності зборів з можливості додавання великої кількості файлів різних форматів

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Marillion [Електронний ресурс] - <https://en.wikipedia.org/wiki/Marillion>
2. ArtistShare [Електронний ресурс] - <https://en.wikipedia.org/wiki/ArtistShare>
3. GoFundMe [Електронний ресурс] - <https://www.gofundme.com/>
4. WhyDonate [Електронний ресурс] - [https://whydonate.com/en/](https://www.gofundme.com/)
5. easyfunraising [Електронний ресурс] - [https://www.easyfundraising.org.uk/](https://www.gofundme.com/)
6. Fundly [Електронний ресурс] - <https://fundly.com/>
7. monobank [Електронний ресурс] - <https://www.monobank.ua/>
8. monobank API [Електронний ресурс] - <https://api.monobank.ua/docs/>
9. ASP.NET Core [Електронний ресурс] –

<https://dotnet.microsoft.com/en-us/apps/aspnet> .

1. C# [Електронний ресурс] - <https://learn.microsoft.com/en-us/dotnet/csharp/>
2. Rider [Електронний ресурс] - <https://dotnet.microsoft.com/en-us/apps/aspnet>
3. .NET [Електронний ресурс] - <https://dotnet.microsoft.com/en-us/>
4. N-tier architecture [Електронний ресурс] -<https://en.wikipedia.org/wiki/Multitier_architecture>
5. JWT [Електронний ресурс] - [https://jwt.io/](https://jwt.io/%20)
6. MS SQL Server [Електронний ресурс] –

<https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16>

1. Entity Framework [Електронний ресурс] –

<https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16>

DataGrip [Електронний ресурс] - <https://www.jetbrains.com/datagrip/>

**ДОДАТОК А**

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_\_\_\_\_ Світлана ПОПЕРЕШНЯК

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ЗБОРУ КОШТІВ (СЕРВЕРНА ЧАСТИНА)

**Технічне завдання**

КПІ.ІП-1103.045440.01.91

“ПОГОДЖЕНО”

Керівник роботи:

Світлана ПОПЕРЕШНЯК

| Консультант: | Виконавець: |
| --- | --- |
| Максим ГОЛОВЧЕНКО | Ігор ВЕРЕМЧУК |

Київ – 2023

**ЗМІСТ**

[**НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ** 49](#_1hmsyys)

[**ПІДСТАВА ДЛЯ РОЗРОБКИ** 50](#_41mghml)

[**ПРИЗНАЧЕННЯ РОЗРОБКИ** 51](#_2grqrue)

[**ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 52](#_vx1227)

[Вимоги до функціональних характеристик 52](#_3fwokq0)

[Для користувача: 52](#_1v1yuxt)

[Для адміністратора системи: 52](#_4f1mdlm)

[Вимоги до надійності 53](#_2u6wntf)

[Умови експлуатації 53](#_19c6y18)

[Вид обслуговування 53](#_3tbugp1)

[Обслуговуючий персонал 53](#_28h4qwu)

[Вимоги до складу і параметрів технічних засобів 53](#_nmf14n)

[Вимоги до інформаційної та програмної сумісності 53](#_37m2jsg)

[Вимоги до вхідних даних 53](#_1mrcu09)

[Вимоги до вихідних даних 53](#_46r0co2)

[Вимоги до мови розробки 53](#_2lwamvv)

[Вимоги до середовища розробки 54](#_111kx3o)

[Вимоги до маркування та пакування 54](#_3l18frh)

[Вимоги до транспортування та зберігання 54](#_206ipza)

[**ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ** 55](#_4k668n3)

[Попередній склад програмної документації 55](#_2zbgiuw)

[Спеціальні вимоги до програмної документації 55](#_1egqt2p)

[**СТАДІЇ І ЕТАПИ РОЗРОБКИ** 56](#_2dlolyb)

[**ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ** 57](#_sqyw64)

**1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ**

Назва розробки: EFund: Веб-застосунок для збору коштів.

Галузь застосування:

Наведене технічне завдання поширюється на розробку серверної частини додатку “EFund: Веб-застосунок для збору коштів”, котре використовується для збору коштів на потреби користувачів та призначена для спрощення залучення коштів користувачами. Додаток буде корисний волонтерським організаціям, стартаперам та всім хто хоче реалізовувати нові проекти та потребує допомоги.

**2 ПІДСТАВА ДЛЯ РОЗРОБКИ**

Підставою для розробки “EFund: Веб-застосунок для збору коштів” є завдання на курсову роботу, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

**3 ПРИЗНАЧЕННЯ РОЗРОБКИ**

Розробка призначена для створення програмного забезпечення, яке забезпечить зручний та швидкий спосіб збору коштів через мережу Інтернет.

Метою розробки є поліпшення роботи зі зборами, спрощення та пришвидшення процесу взаємодії з ними шляхом інтеграції monobank, що робить можливим прив’язку банок до зборів та додаванням інформативних звітів по зборах, що дасть впевненість у коректному використанні коштів залучених користувачами.

Цей сервер для веб-застосунку розробляється для підвищення доступності, популяризації та структуризації зборів. Це може допомогти людям що працюють над проектами та стартапами що потребують залучення коштів, волонтерам та військовим.

**4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Вимоги до функціональних характеристик

Для користувача:

* Реєстрація через імейл та пароль
* Реєстрація через Google OAuth2
* Логін через імейл та пароль
* Логін через Google OAuth2
* Відновлення паролю через імейл
* Зміна фото профілю, імені користувача, імейлу
* Зміна паролю
* Прив’язка акаунту Monobank до профілю користувача
* Створення та зміна збору з назвою, описом, та фото
* Прив’язка банки з Monobank до збору
* Додавання тегів при створенні збору
* Додавання звітів по збору
* Закриття збору при накопиченні достатньої кількості коштів
* Пошук зборів по назві, тегах
* Можливість перейти на банку для донату

Для адміністратора системи:

* Реєстрація через імейл та пароль по запрошенню
* Реєстрація через Google OAuth2 по запрошенню
* Логін через імейл та пароль
* Логін через Google OAuth2
* Можливість запросити адміністратора через імейл
* Відновлення паролю через імейл
* Зміна фото профілю, імені користувача, імейлу
* Зміна паролю
* Можливість видаляти збори за певних причин
* Можливість блокувати користувачів за певних причин

Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

Вид обслуговування

Вимоги до виду обслуговування не висуваються.

Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

* тип процесору: Intel Core i3;
* об‘єм ОЗП: 4 Гб;
* підключення до мережі Інтернет зі швидкістю від 10 мегабіт;

Рекомендована конфігурація технічних засобів:

* тип процесору: Intel Core i5;
* об‘єм ОЗП: 6 Гб;
* підключення до мережі Інтернет зі швидкістю від 50 мегабіт;

Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 або Unix. На комп'ютері має бути встановлений .NET 7.

Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: HTTP запити на ендпоінти API у форматі JSON

Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: відповіді у форматі JSON

Вимоги до мови розробки

Розробку виконати на мові програмування C#

Вимоги до середовища розробки

Розробку виконати на платформі .NET 7

Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

**5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ**

Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

* пояснювальна записка;
* технічне завдання;
* керівництво програміста;
* програма та методика тестування;
* текст програми.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

* схема структурна послідовностей;
* схема структурна варіантів використання;
* схема структурна компонент.

Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

**6 СТАДІЇ І ЕТАПИ РОЗРОБКИ**

| № | Назва етапу | Строк | Звітність |
| --- | --- | --- | --- |
| 1. | Вивчення літератури за тематикою роботи | 06.10 |  |
| 2. | Розробка технічного завдання | 18.10 | Технічне завдання |
| 3. | Аналіз вимог та уточнення специфікацій | 25.10 | Специфікації програмного забезпечення |
| 4. | Проектування структури програмного забезпечення, проектування компонентів | 05.10 | Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму) |
| 5. | Програмна реалізація програмного забезпечення | 12.11 | Тексти програмного забезпечення |
| 6. | Тестування програмного забезпечення | 20.11 | Тести, результати тестування |
| 7. | Розробка матеріалів текстової частини роботи | 30.11 | Пояснювальна записка |
| 8. | Розробка матеріалів графічної частини роботи | 08.12 | Графічний матеріал проекту |
| 9. | Оформлення технічної документації роботи | 17.12 | Технічна документація |

**7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ**

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”

**ДОДАТОК Б**

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_\_\_\_\_ Світлана ПОПЕРЕШНЯК

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ЗБОРУ КОШТІВ (СЕРВЕРНА ЧАСТИНА)

**Текст програми**

КПІ.ІП-1103.045440.01.91

“ПОГОДЖЕНО”

Керівник роботи:

Світлана ПОПЕРЕШНЯК

| Консультант: | Виконавець: |
| --- | --- |
| Максим ГОЛОВЧЕНКО | Ігор ВЕРЕМЧУК |

Київ – 2023

**Program.cs**

using System.Text;

using EFund.BLL.Extensions;

using EFund.BLL.Services;

using EFund.BLL.Services.Auth.Auth;

using EFund.BLL.Services.Auth.Interfaces;

using EFund.BLL.Services.Cache;

using EFund.BLL.Services.Cache.Interfaces;

using EFund.BLL.Services.Interfaces;

using EFund.Client.Monobank;

using EFund.Common.Constants;

using EFund.Common.Models.Configs;

using EFund.DAL.Contexts;

using EFund.DAL.Entities;

using EFund.DAL.Repositories;

using EFund.DAL.Repositories.Interfaces;

using EFund.Email.Services;

using EFund.Email.Services.Interfaces;

using EFund.Hangfire;

using EFund.Hangfire.Abstractions;

using EFund.Mapping.Profiles;

using EFund.Seeding;

using EFund.Validation.Auth;

using EFund.Validation.Extensions;

using EFund.WebAPI.Extensions;

using FluentEmail.MailKitSmtp;

using Hangfire;

using HangfireBasicAuthenticationFilter;

using Microsoft.AspNetCore.Authentication.Google;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using Serilog;

var builder = WebApplication.CreateBuilder(args);

//Configs

var appDataConfig = new AppDataConfig();

var jwtConfig = new JwtConfig();

var emailConfig = new EmailConfig();

var googleConfig = new GoogleConfig();

var hangfireConfig = new HangfireConfig();

var monobankConfig = new MonobankConfig();

builder.Services.AddConfigs(builder.Configuration, opt =>

opt.AddConfig<AppDataConfig>(out appDataConfig, configureOptions: config =>

{

config.AppDataPath = config.AppDataPath.ToAbsolutePath();

config.SeedingDataPath = config.SeedingDataPath.ToAbsolutePath();

config.WebRootPath = builder.Environment.WebRootPath;

})

.AddConfig<JwtConfig>(out jwtConfig)

.AddConfig<EmailConfig>(out emailConfig,

configureOptions: config => config.TemplatesPath = config.TemplatesPath.ToAbsolutePath())

.AddConfig<AuthConfig>()

.AddConfig<GoogleConfig>(out googleConfig)

.AddConfig<HangfireConfig>(out hangfireConfig)

.AddConfig<EncryptionConfig>()

.AddConfig<MonobankConfig>(out monobankConfig, "ApiClients:Monobank")

.AddConfig<CallbackUrisConfig>()

.AddConfig<CacheConfig>());

//DbContext

var dbContextLoggerFactory = LoggerFactory.Create(cfg => cfg.AddConsole());

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"))

.UseLoggerFactory(dbContextLoggerFactory));

//Hangfire

builder.Services.AddHangfire(cfg =>

cfg.UseSqlServerStorage(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddHangfireServer();

builder.Services.AddScoped<IHangfireService, HangfireService>();

//Repositories

builder.Services.AddScoped(typeof(IRepository<>), typeof(Repository<>));

//Services

builder.Services.AddScoped<IAuthService, AuthService>();

builder.Services.AddScoped<IEmailConfirmationService, EmailConfirmationService>();

builder.Services.AddScoped<IGoogleAuthService, GoogleAuthService>();

builder.Services.AddScoped<IPasswordService, PasswordService>();

builder.Services.AddScoped<IRefreshTokenService, RefreshTokenService>();

builder.Services.AddScoped<ICacheService, MemoryCacheService>();

builder.Services.AddScoped<IEncryptionService, EncryptionService>();

builder.Services.AddScoped<IFundraisingReportService, FundraisingReportService>();

builder.Services.AddScoped<IFundraisingService, FundraisingService>();

builder.Services.AddScoped<IMonobankService, MonobankService>();

builder.Services.AddScoped<ITagService, TagService>();

builder.Services.AddScoped<IUserCleanerService, UserCleanerService>();

builder.Services.AddScoped<IUserRegistrationService, UserRegistrationService>();

builder.Services.AddScoped<IUserService, UserService>();

//ApiClients

builder.Services.AddHttpClient(monobankConfig.HttpClientName,

client => client.BaseAddress = new Uri(monobankConfig.BaseAddress));

builder.Services.AddScoped<IMonobankClient, MonobankClient>();

//Email

builder.Services.AddFluentEmail(emailConfig.DefaultEmail)

.AddRazorRenderer(emailConfig.TemplatesPath)

.AddMailKitSender(new SmtpClientOptions

{

Server = emailConfig.SmtpServer,

Port = emailConfig.SmtpPort,

User = emailConfig.DefaultEmail,

Password = emailConfig.Password,

UseSsl = false,

RequiresAuthentication = true

});

builder.Services.AddScoped<IEmailSender, EmailSender>();

//Utility

builder.Services.AddSeeding();

builder.Services.AddCors();

builder.Services.AddMemoryCache();

//Mapper

builder.Services.AddAutoMapper(typeof(UserProfile));

//Validators

builder.Services.AddValidatorServiceFromAssemblyContaining<SignInDTOValidator>();

//Logger

var logger = new LoggerConfiguration()

.MinimumLevel.Information()

.WriteTo.Map(

evt => evt.Level,

(level, wt) =>

wt.File(

$@"{appDataConfig.AppDataPath}\{appDataConfig.LogDirectory}\{level}-{DateTime.Today:yyyy-MM-dd}.log"))

.CreateLogger();

builder.Logging.AddSerilog(logger, dispose: true);

//Auth

builder.Services.AddIdentity<User, IdentityRole<Guid>>()

.AddEntityFrameworkStores<ApplicationDbContext>()

.AddTokenProvider<DataProtectorTokenProvider<User>>(TokenOptions.DefaultProvider);

var tokenValidationParameters = new TokenValidationParameters

{

ValidateIssuerSigningKey = true,

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtConfig.Secret)),

ValidIssuer = jwtConfig.Issuer,

ValidAudience = jwtConfig.Audience,

ClockSkew = jwtConfig.ClockSkew

};

builder.Services.AddSingleton(tokenValidationParameters);

builder.Services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

options.SaveToken = true;

options.TokenValidationParameters = tokenValidationParameters;

})

.AddGoogle(GoogleDefaults.AuthenticationScheme, opt =>

{

opt.ClientId = googleConfig.ClientId;

opt.ClientSecret = googleConfig.ClientSecret;

});

builder.Services.AddAuthorization(options =>

{

options.AddPolicy(Policies.Admin, policy => policy.RequireRole(Roles.Admin));

options.AddPolicy(Policies.User,

policy => policy.RequireRole(Roles.User).RequireClaim(Claims.Blocked, "False"));

options.AddPolicy(Policies.Shared,

policy => policy.RequireRole(Roles.Admin, Roles.User).RequireClaim(Claims.Blocked, "False"));

});

//Swagger

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "EFund API", Version = "v1" });

c.AddSecurityDefinition("Bearer",

new OpenApiSecurityScheme

{

Description = "Standard Authorization header using the Bearer scheme. Example: \"bearer {token}\"",

In = ParameterLocation.Header,

Name = "Authorization",

Type = SecuritySchemeType.ApiKey

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference { Type = ReferenceType.SecurityScheme, Id = "Bearer" }

},

Array.Empty<string>()

}

});

});

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

var app = builder.Build();

app.MigrateDatabase();

await app.SeedDataAsync();

await app.ValidateConfigsAsync();

app.UseHangfireDashboard("/hangfire", new DashboardOptions

{

DashboardTitle = "EFund Dashboard",

Authorization = new[]

{

new HangfireCustomBasicAuthenticationFilter

{

User = hangfireConfig.User, Pass = hangfireConfig.Password

}

}

});

app.MapHangfireDashboard();

app.SetupHangfire(hangfireConfig);

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseCors(x => x.AllowAnyHeader()

.AllowAnyOrigin()

.AllowAnyMethod());

app.UseAuthorization();

app.UseAuthentication();

app.MapControllers();

app.Run();

**AuthController.cs**

using EFund.BLL.Services.Auth.Interfaces;

using EFund.Common.Models.DTO.Auth;

using EFund.Validation;

using EFund.Validation.Extensions;

using EFund.WebAPI.Extensions;

using Microsoft.AspNetCore.Mvc;

namespace EFund.WebAPI.Controllers;

[ApiController]

[Route("api/auth")]

public class AuthController : ControllerBase

{

private readonly IAuthService \_authService;

private readonly IEmailConfirmationService \_emailConfirmationService;

private readonly IRefreshTokenService \_refreshTokenService;

private readonly IPasswordService \_passwordService;

private readonly IValidatorService \_validator;

public AuthController(IAuthService authService,

IEmailConfirmationService emailConfirmationService,

IRefreshTokenService refreshTokenService,

IPasswordService passwordService,

IValidatorService validator)

{

\_authService = authService;

\_emailConfirmationService = emailConfirmationService;

\_refreshTokenService = refreshTokenService;

\_passwordService = passwordService;

\_validator = validator;

}

[HttpPost("sign-up")]

public async Task<IActionResult> SignUp(SignUpDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_authService.SignUpAsync(dto);

return result.ToActionResult();

}

[HttpPost("confirm-email")]

public async Task<IActionResult> ConfirmEmail(ConfirmEmailDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_emailConfirmationService.ConfirmEmailAsync(dto);

return result.ToActionResult();

}

[HttpPost("sign-in")]

public async Task<IActionResult> SignIn(SignInDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_authService.SignInAsync(dto);

return result.ToActionResult();

}

[HttpPost("refresh-token")]

public async Task<IActionResult> RefreshToken(RefreshTokenDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_refreshTokenService.RefreshTokenAsync(dto);

return result.ToActionResult();

}

[HttpPost("resend-confirmation-code")]

public async Task<IActionResult> ResendConfirmationCode(ResendConfirmationCodeDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_emailConfirmationService.ResendConfirmationCodeAsync(dto);

return result.ToActionResult();

}

[HttpPost("forgot-password")]

public async Task<IActionResult> ForgotPassword(ForgotPasswordDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_passwordService.ForgotPasswordAsync(dto);

return result.ToActionResult();

}

[HttpPost("reset-password")]

public async Task<IActionResult> ResetPassword(ResetPasswordDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_passwordService.ResetPasswordAsync(dto);

return result.ToActionResult();

}

}

**FundraisingController.cs**

using EFund.BLL.Services.Interfaces;

using EFund.Common.Constants;

using EFund.Common.Models.DTO.Common;

using EFund.Common.Models.DTO.Fundraising;

using EFund.Validation;

using EFund.Validation.Extensions;

using EFund.WebAPI.Extensions;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace EFund.WebAPI.Controllers;

[ApiController]

[Route("api/fundraisings")]

public class FundraisingController : ControllerBase

{

private readonly IFundraisingService \_fundraisingService;

private readonly IValidatorService \_validator;

public FundraisingController(IFundraisingService fundraisingService, IValidatorService validator)

{

\_fundraisingService = fundraisingService;

\_validator = validator;

}

[HttpPost("search")]

public async Task<IActionResult> Search([FromBody] SearchFundraisingDTO dto, [FromQuery] PaginationDTO pagination)

{

var validationResult = await \_validator.ValidateAsync(pagination);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_fundraisingService.Search(dto, pagination, HttpContext.GetApiUrl());

return result.ToActionResult();

}

[HttpGet("{id}")]

public async Task<IActionResult> GetById(Guid id)

{

var result = await \_fundraisingService.GetByIdAsync(id, HttpContext.GetApiUrl());

return result.ToActionResult();

}

[HttpPost]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.User)]

public async Task<IActionResult> Add([FromBody] CreateFundraisingDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_fundraisingService.AddAsync(HttpContext.GetUserId(), dto, HttpContext.GetApiUrl());

return result.Match<IActionResult>(

Right: fundraising => CreatedAtAction(nameof(GetById), new { id = fundraising.Id }, fundraising),

Left: BadRequest

);

}

[HttpPut("{id}")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.User)]

public async Task<IActionResult> Update(Guid id, [FromBody] UpdateFundraisingDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_fundraisingService.UpdateAsync(id, HttpContext.GetUserId(), dto, HttpContext.GetApiUrl());

return result.ToActionResult();

}

[HttpDelete("{id}")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.Shared)]

public async Task<IActionResult> Delete(Guid id)

{

var result = HttpContext.IsAdmin()

? await \_fundraisingService.DeleteAsync(id)

: await \_fundraisingService.DeleteAsync(id, HttpContext.GetUserId());

return result.ToActionResult();

}

[HttpPost("{id}/upload-avatar")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.User)]

public async Task<IActionResult> UploadAvatar(Guid id, IFormFile file)

{

var result = await \_fundraisingService.UploadAvatarAsync(id, HttpContext.GetUserId(), file);

return result.ToActionResult();

}

[HttpPost("{id}/delete-avatar")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.User)]

public async Task<IActionResult> DeleteAvatar(Guid id)

{

var result = await \_fundraisingService.DeleteAvatarAsync(id, HttpContext.GetUserId());

return result.ToActionResult();

}

}

**FundraisingReportController.cs**

using EFund.BLL.Services.Interfaces;

using EFund.Common.Constants;

using EFund.Common.Models.DTO.FundraisingReport;

using EFund.Validation;

using EFund.Validation.Extensions;

using EFund.WebAPI.Extensions;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace EFund.WebAPI.Controllers;

[ApiController]

[Route("api/fundraising-reports")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.User)]

public class FundraisingReportController : ControllerBase

{

private readonly IFundraisingReportService \_fundraisingReportService;

private readonly IValidatorService \_validator;

public FundraisingReportController(IFundraisingReportService fundraisingReportService, IValidatorService validator)

{

\_fundraisingReportService = fundraisingReportService;

\_validator = validator;

}

[HttpPost]

public async Task<IActionResult> AddReport([FromBody] CreateFundraisingReportDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_fundraisingReportService.AddAsync(HttpContext.GetUserId(), dto);

return result.ToActionResult();

}

[HttpPut("{id}")]

public async Task<IActionResult> UpdateReport(Guid id, [FromBody] UpdateFundraisingReportDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_fundraisingReportService.UpdateAsync(id, HttpContext.GetUserId(), dto);

return result.ToActionResult();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteReport(Guid id)

{

var result = await \_fundraisingReportService.DeleteAsync(id, HttpContext.GetUserId());

return result.ToActionResult();

}

[HttpPost("{id}/upload-attachments")]

public async Task<IActionResult> AddAttachment(Guid id, IFormFileCollection files)

{

var result = await \_fundraisingReportService.AddAttachmentsAsync(id, HttpContext.GetUserId(), files);

return result.ToActionResult();

}

[HttpPost("{id}/delete-attachments")]

public async Task<IActionResult> DeleteAttachments(Guid id, [FromBody] DeleteAttachmentsDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_fundraisingReportService.DeleteAttachmentsAsync(id, HttpContext.GetUserId(), dto);

return result.ToActionResult();

}

}

**GoogleAuthController.cs**

using EFund.BLL.Services.Auth.Interfaces;

using EFund.WebAPI.Extensions;

using Microsoft.AspNetCore.Mvc;

namespace EFund.WebAPI.Controllers;

[ApiController]

[Route("api/google-auth")]

public class GoogleAuthController : ControllerBase

{

private readonly IGoogleAuthService \_googleAuthService;

public GoogleAuthController(IGoogleAuthService googleAuthService)

{

\_googleAuthService = googleAuthService;

}

[HttpPost("sign-up")]

public async Task<IActionResult> SignUp([FromHeader(Name = "Authorization-Code")] string authorizationCode)

{

var result = await \_googleAuthService.SignUpAsync(authorizationCode);

return result.ToActionResult();

}

[HttpPost("sign-in")]

public async Task<IActionResult> SignIn([FromHeader(Name = "Authorization-Code")] string authorizationCode)

{

var result = await \_googleAuthService.SignInAsync(authorizationCode);

return result.ToActionResult();

}

}

**MonobankController.cs**

using EFund.BLL.Services.Interfaces;

using EFund.Common.Constants;

using EFund.WebAPI.Extensions;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace EFund.WebAPI.Controllers;

[ApiController]

[Route("api/monobank")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.Shared)]

public class MonobankController : ControllerBase

{

private readonly IMonobankService \_monobankService;

public MonobankController(IMonobankService service)

{

\_monobankService = service;

}

[HttpPost("link-token")]

public async Task<IActionResult> AddToken([FromHeader] string token)

{

var result = await \_monobankService.AddOrUpdateMonobankTokenAsync(HttpContext.GetUserId(), token);

return result.ToActionResult();

}

[HttpGet("jars")]

public async Task<IActionResult> GetJars()

{

var result = await \_monobankService.GetJarsAsync(HttpContext.GetUserId());

return result.ToActionResult();

}

}

**TagController.cs**

using EFund.BLL.Services.Interfaces;

using EFund.Common.Constants;

using EFund.Common.Models.DTO.Common;

using EFund.Common.Models.DTO.Tag;

using EFund.Validation;

using EFund.Validation.Extensions;

using EFund.WebAPI.Extensions;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace EFund.WebAPI.Controllers;

[ApiController]

[Route("api/tags")]

public class TagController : ControllerBase

{

private readonly ITagService \_tagService;

private readonly IValidatorService \_validator;

public TagController(ITagService tagService, IValidatorService validator)

{

\_tagService = tagService;

\_validator = validator;

}

[HttpGet]

public async Task<IActionResult> GetAll([FromQuery] PaginationDTO pagination)

{

var validationResult = await \_validator.ValidateAsync(pagination);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_tagService.GetAllAsync(pagination);

return Ok(result);

}

[HttpPost]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.Shared)]

public async Task<IActionResult> Add([FromBody] CreateTagDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_tagService.AddAsync(dto);

return result.Match<IActionResult>(

Right: tag => CreatedAtAction(nameof(GetByName), new { name = tag.Name }, tag),

Left: BadRequest

);

}

[HttpGet("{name}")]

public async Task<IActionResult> GetByName(string name, [FromQuery] PaginationDTO pagination)

{

var validationResult = await \_validator.ValidateAsync(pagination);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_tagService.GetByNameAsync(name, pagination);

return Ok(result);

}

[HttpDelete("{name}")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.Admin)]

public async Task<IActionResult> Delete(string name)

{

var result = await \_tagService.DeleteAsync(name);

return result.ToActionResult();

}

}

**UserController.cs**

using EFund.BLL.Services.Auth.Interfaces;

using EFund.BLL.Services.Interfaces;

using EFund.Common.Constants;

using EFund.Common.Models.DTO.Common;

using EFund.Common.Models.DTO.User;

using EFund.Validation;

using EFund.Validation.Extensions;

using EFund.WebAPI.Extensions;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace EFund.WebAPI.Controllers;

[ApiController]

[Route("api/users")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.Shared)]

public class UserController : ControllerBase

{

private readonly IUserService \_userService;

private readonly IPasswordService \_passwordService;

private readonly IValidatorService \_validator;

public UserController(IUserService userService, IPasswordService passwordService, IValidatorService validator)

{

\_userService = userService;

\_passwordService = passwordService;

\_validator = validator;

}

[HttpGet("me")]

public async Task<IActionResult> GetById()

{

var result = await \_userService.GetByIdAsync(HttpContext.GetUserId(), HttpContext.GetApiUrl());

return result.ToActionResult();

}

[HttpPost("search")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.Admin)]

public async Task<IActionResult> Search([FromBody] SearchUserDTO dto, [FromQuery] PaginationDTO pagination)

{

var validationResult = await \_validator.ValidateAsync(pagination);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_userService.SearchAsync(dto, pagination, HttpContext.GetApiUrl());

return result.ToActionResult();

}

[HttpPut("me")]

public async Task<IActionResult> UpdateUser(UpdateUserDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_userService.UpdateUserAsync(HttpContext.GetUserId(), dto, HttpContext.GetApiUrl());

return result.ToActionResult();

}

[HttpPost("change-password")]

public async Task<IActionResult> ChangePassword(ChangePasswordDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_passwordService.ChangePasswordAsync(HttpContext.GetUserId(), dto);

return result.ToActionResult();

}

[HttpPost("add-password")]

public async Task<IActionResult> AddPassword(AddPasswordDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_passwordService.AddPasswordAsync(HttpContext.GetUserId(), dto);

return result.ToActionResult();

}

[HttpPost("change-email")]

public async Task<IActionResult> ChangeEmail(ChangeEmailDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_userService.SendChangeEmailCodeAsync(HttpContext.GetUserId(), dto);

return result.ToActionResult();

}

[HttpPost("confirm-change-email")]

public async Task<IActionResult> ConfirmChangeEmail(ConfirmChangeEmailDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_userService.ChangeEmailAsync(HttpContext.GetUserId(), dto);

return result.ToActionResult();

}

[HttpPost("upload-avatar")]

public async Task<IActionResult> UploadAvatar(IFormFile file)

{

var result = await \_userService.UploadAvatarAsync(HttpContext.GetUserId(), file);

return result.ToActionResult();

}

[HttpPost("delete-avatar")]

public async Task<IActionResult> DeleteAvatar()

{

var result = await \_userService.DeleteAvatarAsync(HttpContext.GetUserId());

return result.ToActionResult();

}

[HttpPost("make-admin")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.Admin)]

public async Task<IActionResult> MakeAdmin(MakeAdminDTO dto)

{

var result = await \_userService.MakeAdminAsync(dto);

return result.ToActionResult();

}

[HttpPost("invite-admin")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.Admin)]

public async Task<IActionResult> Invite(InviteAdminDTO dto)

{

var validationResult = await \_validator.ValidateAsync(dto);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_userService.InviteAdminAsync(dto);

return result.ToActionResult();

}

[HttpPost("action")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = Policies.Admin)]

public async Task<IActionResult> BlockUser(UserActionDTO actionDTO)

{

var validationResult = await \_validator.ValidateAsync(actionDTO);

if (!validationResult.IsValid)

return BadRequest(validationResult.ToErrorDTO());

var result = await \_userService.PerformUserActionAsync(actionDTO);

return result.ToActionResult();

}

}

**HttpContextExtensions.cs**

using EFund.Common.Constants;

namespace EFund.WebAPI.Extensions;

public static class HttpContextExtensions

{

public static Guid GetUserId(this HttpContext httpContext)

{

return Guid.Parse(httpContext.User.Claims.Single(c => c.Type == Claims.Id).Value);

}

public static bool IsAdmin(this HttpContext httpContext)

{

return httpContext.User.Claims.Any(c => c is { Type: "role", Value: Roles.Admin });

}

public static string GetApiUrl(this HttpContext httpContext)

{

return $"{httpContext.Request.Scheme}://{httpContext.Request.Host}";

}

}

**LanguageExtExtensions.cs**

using EFund.Common.Models.DTO.Error;

using LanguageExt;

using Microsoft.AspNetCore.Mvc;

namespace EFund.WebAPI.Extensions;

public static class LanguageExtExtensions

{

public static IActionResult ToActionResult<T>(this Either<ErrorDTO, T> either)

{

return either.Match<IActionResult>(

Left: error => new BadRequestObjectResult(error),

Right: x => new OkObjectResult(x)

);

}

public static IActionResult ToActionResult(this Option<ErrorDTO> either)

{

return either.Match<IActionResult>(

Some: error => new BadRequestObjectResult(error),

None: () => new NoContentResult()

);

}

}

**ServicesExtensions.cs**

using EFund.DAL.Contexts;

using EFund.WebAPI.Utility;

using Microsoft.EntityFrameworkCore;

namespace EFund.WebAPI.Extensions;

public static class ServicesExtensions

{

public static void MigrateDatabase(this IHost app)

{

using var scope = app.Services.CreateScope();

var services = scope.ServiceProvider;

var context = services.GetRequiredService<ApplicationDbContext>();

context.Database.Migrate();

}

public static IServiceCollection AddConfigs(this IServiceCollection services, IConfiguration configuration,

Action<ConfigurationServiceBuilder> configurationBuilder)

{

var builder = new ConfigurationServiceBuilder(services, configuration);

configurationBuilder(builder);

return services;

}

}

ConfigurationServiceBuilder.cs

using EFund.Common.Models.Configs.Abstract;

namespace EFund.WebAPI.Utility;

public class ConfigurationServiceBuilder

{

private readonly IServiceCollection \_services;

private readonly IConfiguration \_configuration;

public ConfigurationServiceBuilder(IServiceCollection services, IConfiguration configuration)

{

\_services = services;

\_configuration = configuration;

}

public ConfigurationServiceBuilder AddConfig<T>(string? sectionName = null,

Action<T>? configureOptions = null)

where T : ConfigBase, new()

{

CreateAndAddConfig(sectionName, configureOptions);

return this;

}

public ConfigurationServiceBuilder AddConfig<T>(out T config, string? sectionName = null,

Action<T>? configureOptions = null)

where T : ConfigBase, new()

{

config = CreateAndAddConfig(sectionName, configureOptions);

return this;

}

private T CreateAndAddConfig<T>(string? sectionName, Action<T>? configureOptions)

where T : ConfigBase, new()

{

var config = new T();

var configSection = \_configuration.GetSection(sectionName ?? typeof(T).Name);

if (!configSection.Exists())

configSection = \_configuration.GetSection(typeof(T).Name.Replace("Config", string.Empty));

if (!configSection.Exists())

throw new ArgumentException("Failed to find specified config section");

configSection.Bind(config);

configureOptions?.Invoke(config);

\_services.AddSingleton(config);

return config;

}

**ДОДАТОК В**

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_\_\_\_\_ Світлана ПОПЕРЕШНЯК

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ЗБОРУ КОШТІВ (СЕРВЕРНА ЧАСТИНА)

**Програма та методика тестування**

КПІ.ІП-1103.045440.01.91

“ПОГОДЖЕНО”

Керівник роботи:

Світлана ПОПЕРЕШНЯК

| Консультант: | Виконавець: |
| --- | --- |
| Максим ГОЛОВЧЕНКО | Ігор ВЕРЕМЧУК |

Київ – 2023

**зміст**

[**1**](#_3cqmetx) **ОБ’ЄКТ ВИПРОБУВАНЬ** 104

[**2**](#_1rvwp1q) **МЕТА ТЕСТУВАННЯ** 105

[**3**](#_4bvk7pj) **МЕТОДИ ТЕСТУВАННЯ** 106

[**4**](#_2r0uhxc) **ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ** 107

1. **ОБ’ЄКТ ВИПРОБУВАНЬ**

Об’єктом випробування є сервер для веб-застосунку для збору коштів.

1. **МЕТА ТЕСТУВАННЯ**

Метою тестування є наступне:

* перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
* перевірка збереження даних;
* знаходження проблем, помилок і недоліків з метою їх усунення;

1. **МЕТОДИ ТЕСТУВАННЯ**

Для тестування програмного забезпечення використовуються такі методи:

* функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
* системне тестування – перевіряється усе програмне забезпечення в цілому;
* мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;
* тестування «чорної скриньки» − об’єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;

1. **ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ**

Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

* тестування коректності обробки HTTP запитів
* тестування коректності та цілісності зберігання даних

**ДОДАТОК Г**

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_\_\_\_\_ Світлана ПОПЕРЕШНЯК

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ЗБОРУ КОШТІВ (СЕРВЕРНА ЧАСТИНА)

**Керівництво програміста**

КПІ.ІП-1103.045440.01.91

“ПОГОДЖЕНО”

Керівник роботи:

Світлана ПОПЕРЕШНЯК

| Консультант: | Виконавець: |
| --- | --- |
| Максим ГОЛОВЧЕНКО | Ігор ВЕРЕМЧУК |

Київ – 2023

**зміст**

[**1**](#_1664s55) **Призначення програми** 111

[**2**](#_3q5sasy) **API** 112

1. **Призначення програми**

Сервер веб-додатку для збору коштів створений для обробки запитів від клієнта по управлінню зборами, звітами, профілем користувача, зберігання їх даних та файлів.

1. **API**

Для користування сервером необхідно використати API наданий нижче

**Base URLs:**

* <https://localhost:7008>
* http://localhost:5282

**Endpoints:**

**POST api/auth/sign-up**

Request:

Request body example:

{

"Name": "",

"Email": "",

"Password": "",

"AdminToken": null

}

Response:

Status code: 200

Error status codes:

400

Response body example:

{

"UserId": "00000000-0000-0000-0000-000000000000"

}

**POST api/auth/confirm-email**

Request:

Request body example:

{

"UserId": "00000000-0000-0000-0000-000000000000",

"Code": 0

}

Response:

Status code: 200

Error status codes:

400

Response body example:

{

"AccessToken": "",

"RefreshToken": ""

}

**POST api/auth/sign-in**

Request:

Request body example:

{

"Email": "",

"Password": ""

}

Response:

Status code: 200

Error status codes:

400

Response body example:

{

"AccessToken": "",

"RefreshToken": ""

}

**POST api/auth/refresh-token**

Request:

Request body example:

{

"AccessToken": "",

"RefreshToken": ""

}

Response:

Status code: 200

Error status codes:

400

Response body example:

{

"AccessToken": "",

"RefreshToken": ""

}

**POST api/auth/resend-confirmation-code**

Request:

Request body example:

{

"UserId": "00000000-0000-0000-0000-000000000000"

}

Response:

Status code: 204

Error status codes:

400

**POST api/auth/forgot-password**

Request:

Request body example:

{

"Email": ""

}

Response:

Status code: 204

Error status codes:

400

**POST api/auth/reset-password**

Request:

Request body example:

{

"Email": "",

"Token": "",

"NewPassword": ""

}

Response:

Status code: 204

Error status codes:

400

**POST api/fundraisings/search**

Request:

URL parameters:

page: int

pageSize: int

Request body example:

{

"Title": null,

"Tags": [],

"IncludeClosed": false

}

Response:

Status code: 200

Error status codes:

400

Response body example:

{

"Items": [],

"Page": 0,

"TotalPages": 0,

"PageSize": 0,

"TotalCount": 0,

"HasPrevious": false,

"HasNext": false

}

**GET api/fundraisings/{id}**

Request:

URL parameters:

id: System.Guid

Response:

Status code: 200

Error status codes:

400

Response body example:

{

"Id": "00000000-0000-0000-0000-000000000000",

"Title": "",

"Description": "",

"AvatarUrl": "",

"UserId": "00000000-0000-0000-0000-000000000000",

"MonobankJarId": "",

"IsClosed": false,

"Tags": [],

"Reports": [],

"MonobankJar": null

}

**POST api/fundraisings**

Request:

Request body example:

{

"Title": "",

"Description": "",

"MonobankJarId": "",

"Tags": []

}

Response:

Status code: 201

Error status codes:

400

401

403

Response body example:

{

"Id": "00000000-0000-0000-0000-000000000000",

"Title": "",

"Description": "",

"AvatarUrl": "",

"UserId": "00000000-0000-0000-0000-000000000000",

"MonobankJarId": "",

"IsClosed": false,

"Tags": [],

"Reports": [],

"MonobankJar": null

}

**PUT api/fundraisings/{id}**

Request:

URL parameters:

id: System.Guid

Request body example:

{

"Title": "",

"Description": "",

"MonobankJarId": "",

"IsClosed": false,

"Tags": []

}

Response:

Status code: 200

Error status codes:

400

401

403

Response body example:

{

"Id": "00000000-0000-0000-0000-000000000000",

"Title": "",

"Description": "",

"AvatarUrl": "",

"UserId": "00000000-0000-0000-0000-000000000000",

"MonobankJarId": "",

"IsClosed": false,

"Tags": [],

"Reports": [],

"MonobankJar": null

}

**DELETE api/fundraisings/{id}**

Request:

URL parameters:

id: System.Guid

Response:

Status code: 200

Error status codes:

400

401

403

Response body example:

{

"Id": "00000000-0000-0000-0000-000000000000",

"Title": "",

"Description": "",

"AvatarUrl": "",

"UserId": "00000000-0000-0000-0000-000000000000",

"MonobankJarId": "",

"IsClosed": false,

"Tags": [],

"Reports": [],

"MonobankJar": null

}

**POST api/fundraisings/{id}/avatar**

Request:

URL parameters:

id: System.Guid

File parameters:

Allowed file extensions:

.png

.jpg

.jpeg

.gif

Response:

Status code: 204

Error status codes:

400

401

403

**DELETE api/fundraisings/{id}/avatar**

Request:

URL parameters:

id: System.Guid

Response:

Status code: 204

Error status codes:

400

401

403

**POST api/fundraising-reports**

Request:

Request body example:

{

"Title": "",

"Description": "",

"FundraisingId": "00000000-0000-0000-0000-000000000000"

}

Response:

Status code: 201

Error status codes:

400

401

403

Response body example:

{

"Id": "00000000-0000-0000-0000-000000000000",

"Title": "",

"Description": "",

"FundraisingId": "00000000-0000-0000-0000-000000000000",

"Attachments": []

}

**PUT api/fundraising-reports/{id}**

Request:

URL parameters:

id: System.Guid

Request body example:

{

"Title": "",

"Description": ""

}

Response:

Status code: 200

Error status codes:

400

401

403

Response body example:

{

"Id": "00000000-0000-0000-0000-000000000000",

"Title": "",

"Description": "",

"FundraisingId": "00000000-0000-0000-0000-000000000000",

"Attachments": []

}

**DELETE api/fundraising-reports/{id}**

Request:

URL parameters:

id: System.Guid

Response:

Status code: 204

Error status codes:

400

401

403

**POST api/fundraising-reports/{id}/attachments**

Request:

URL parameters:

id: System.Guid

File parameters:

Allowed file extensions:

.png

.jpg

.jpeg

.gif

.pdf

doc

docx

xls

xlsx

Response:

Status code: 204

Error status codes:

400

401

403

**DELETE api/fundraising-reports/{id}/attachments**

Request:

URL parameters:

id: System.Guid

Request body example:

{

"AttachmentIds": []

}

Response:

Status code: 204

Error status codes:

400

401

403

POST api/google-auth/sign-up

Request:

Header parameters:

authorizationCode: System.String

Request body example:

{

"AdminToken": null

}

Response:

Status code: 200

Error status codes:

400

Response body example:

{

"AccessToken": "",

"RefreshToken": ""

}

**POST api/google-auth/sign-in**

Request:

Header parameters:

authorizationCode: System.String

Response:

Status code: 200

Error status codes:

400

Response body example:

{

"AccessToken": "",

"RefreshToken": ""

}

**POST api/monobank/link-token**

Request:

Header parameters:

token: System.String

Response:

Status code: 204

Error status codes:

400

**GET api/monobank/jars**

Request:

URL parameters:

name: System.String

Response:

Status code: 200

Error status codes:

400

Response body example:

[]

**GET api/tags**

Request:

Response:

Status code: 200

Response body example:

[]

**POST api/tags**

Request:

Request body example:

{

"Name": ""

}

Response:

Status code: 201

Error status codes:

400

401

403

Response body example:

{

"Name": ""

}

**GET api/tags/{name}**

Request:

URL parameters:

name: System.String

Response:

Status code: 200

Response body example:

[]

**DELETE api/tags/{name}**

Request:

URL parameters:

name: System.String

Response:

Status code: 204

Error status codes:

400

401

403

**GET api/users/me**

Request:

Response:

Status code: 200

Error status codes:

400

401

403

Response body example:

{

"Id": "00000000-0000-0000-0000-000000000000",

"Email": "",

"Name": "",

"AvatarUrl": "",

"HasPassword": false,

"HasMonobankToken": false

}

**POST api/users/search**

Request:

URL parameters:

page: int

pageSize: int

Request body example:

{

"UserIds": null,

"Emails": null,

"UserNames": null,

"CreatedByAdmin": null,

"IsBlocked": null,

"EmailConfirmed": null

}

Response:

Status code: 200

Error status codes:

400

401

403

Response body example:

{

"Items": [],

"Page": 0,

"TotalPages": 0,

"PageSize": 0,

"TotalCount": 0,

"HasPrevious": false,

"HasNext": false

}

**PUT api/users/me**

Request:

Request body example:

{

"Name": ""

}

Response:

Status code: 200

Error status codes:

400

401

403

Response body example:

{

"Id": "00000000-0000-0000-0000-000000000000",

"Email": "",

"Name": "",

"AvatarUrl": "",

"HasPassword": false,

"HasMonobankToken": false

}

**POST api/users/change-password**

Request:

Request body example:

{

"OldPassword": "",

"NewPassword": ""

}

Response:

Status code: 204

Error status codes:

400

401

403

**POST api/users/add-password**

Request:

Request body example:

{

"Password": ""

}

Response:

Status code: 204

Error status codes:

400

401

403

**POST api/users/change-email**

Request:

Request body example:

{

"NewEmail": ""

}

Response:

Status code: 204

Error status codes:

400

401

403

**POST api/users/confirm-change-email**

Request:

Request body example:

{

"NewEmail": "",

"Code": 0

}

Response:

Status code: 204

Error status codes:

400

401

403

**POST api/users/avatar**

Request:

File parameters:

Allowed file extensions:

.png

.jpg

.jpeg

.gif

Response:

Status code: 204

Error status codes:

400

401

403

**DELETE api/users/avatar**

Request:

Response:

Status code: 204

Error status codes:

400

401

403

**POST api/users/make-admin**

Request:

Request body example:

{

"UserId": "00000000-0000-0000-0000-000000000000"

}

Response:

Status code: 204

Error status codes:

400

401

403

**POST api/users/invite-admin**

Request:

Request body example:

{

"Email": ""

}

Response:

Status code: 204

Error status codes:

400

401

403

**POST api/users/action**

Request:

Request body example:

{

"UserId": "00000000-0000-0000-0000-000000000000",

"Action": ""

}

Response:

Status code: 204

Error status codes:

400

401

403

| *№ з/п* | | *Формат* | *Позначення* | | | | | *Найменування* | | | | | *Кількість листів* | *Примітка* | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | | *А4* |  | | | | | *Завдання на курсову роботу* | | | | | *2* |  | |
| *2* | | *А4* | КПІ.ІП-1103.045440.01.91 | | | | | *Пояснювальна записка* | | | | | *42* |  | |
| *3* | | *А4* | КПІ.ІП-1103.045440.01.91 | | | | | *Технічне завдання* | | | | | *11* |  | |
| *4* | | *А4* | КПІ.ІП-1103.045440.01.91 | | | | | *Керівництво програміста* | | | | | *25* |  | |
| *5* | | *А4* | КПІ.ІП-1103.045440.01.91 | | | | | *Програма та методика тестування* | | | | | *6* |  | |
| *6* | | *А4* | КПІ.ІП-1103.045440.01.91 | | | | | *Опис програми* | | | | | *36* |  | |
| *7* | |  |  | | | | |  | | | | |  |  | |
| *8* | |  |  | | | | |  | | | | |  |  | |
| *9* | |  |  | | | | |  | | | | |  |  | |
| *10* | |  |  | | | | |  | | | | |  |  | |
|  | |  |  | | | | |  | | | | |  |  | |
|  | |  |  | | | | |  | | | | |  |  | |
|  | |  |  | | | | |  | | | | |  |  | |
|  | |  |  | | | | |  | | | | |  |  | |
|  | |  |  | | | | |  | | | | |  |  | |
|  | |  |  | | | | |  | | | | |  |  | |
|  | |  |  | | | | |  | | | | |  |  | |
|  | |  |  | | | | |  | | | | |  |  | |
|  | |  |  | | | | |  | | | | |  |  | |
|  | |  |  | | | | |  | | | | |  |  | |
|  |  | | |  |  |  | *КПІ.ІП-1103.045440.01.91* | | | | | | | | |
|  |  | | |  |  |  |
| ***Зм.*** | ***Арк.*** | | | ***ПІБ*** | ***Підп.*** | ***Дата*** |
| *Розробн.* | | | | Веремчук І.І. |  |  | *Відомість  курсової роботи* | | *Літ.* | | | *Лист* | | | *Листів* |
| *Керівн.* | | | | Поперешняк С.В. |  |  |  |  |  | *1* | | | *1* |
| *Консульт.* | | | |  |  |  | *КПІ ім.Ігоря Сікорського*  *кафедра ІПІ гр. ІП-11* | | | | | | |
| *Н/контр.* | | | |  |  |  |
| *В.о.зав.каф.* | | | |  |  |  |