

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій  
Кафедра комп'ютерних систем та мереж

**Пояснювальна записка**

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ДЛЯ ДОМАШНЬОЇ  
БУХГАЛТЕРІЇ

Виконав: студент 4 курсу, групи КНТ-510  
спеціальності \_\_\_\_\_  
123 Комп'ютерна інженерія

Освітня програма (спеціалізація)  
Комп'ютерна інженерія

(код і назва спеціальності)

ЛУПОНОС І.А.

(прізвище та ініціали)

Керівник ХОХЛОВ М.М.

(прізвище та ініціали)

Рецензент ЄФИМЕНКО М.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
**Національний університет «Запорізька політехніка»**  
( повне найменування вищого навчального закладу )

Факультет Комп'ютерних наук і технологій  
Кафедра «Комп'ютерні системи та мережі»  
Ступінь вищої освіти (освітній ступінь) бакалаврський  
Спеціальність 123 Комп'ютерна інженерія  
(код і назва)  
Освітня програма (спеціалізація) Комп'ютерна інженерія  
(назва)

**ЗАТВЕРДЖУЮ**

Зав. кафедри Кудерметов Р.К.

“12” листопада 2024 року

**З А В Д А Н Н Я**  
**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ**

ЛУПОНОС Івана Андрійовича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка комп'ютерної системи для домашньої бухгалтерії

керівник проекту (роботи) ХОХЛОВ М.М.

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

затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105

2. Строк подання студентом проекту (роботи) 01.06.2024 року

3. Вихідні дані до проекту (роботи) мікросервісна архітектура, JavaScript, HTML, фреймворк Ruby on Rails, Visual Studio, Microsoft SQL Server

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Огляд вебдодатків;

2) Обґрунтування вибору технології розробки системи;

3) Розробка серверної частини системи;

4) Тестування роботи системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

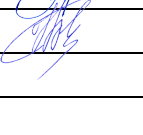
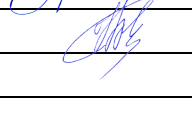
1) Інтерфейси додатків контролю витрат;

2) Характеристика фреймворків;

3) Характеристика середовища розробки та бази даних;

4) Інтерфейс додатку.

## 6. Консультанти розділів проекту (роботи)

| Розділ        | Прізвище, ініціали та посада консультанта | Підпис, дата                                                                        |                                                                                     |
|---------------|-------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|               |                                           | завдання видав                                                                      | прийняв виконане завдання                                                           |
| 1-4           | ХОХЛОВ М.М., ст. викладач                 |  |  |
| Нормоконтроль | ЩЕРБАК Н.В., ст. викладач                 |  |  |
|               |                                           |                                                                                     |                                                                                     |
|               |                                           |                                                                                     |                                                                                     |
|               |                                           |                                                                                     |                                                                                     |
|               |                                           |                                                                                     |                                                                                     |
|               |                                           |                                                                                     |                                                                                     |

7. Дата видачі завдання 01.04.2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проекту (роботи) | Строк виконання етапів проекту (роботи) | Примітка |
|-------|------------------------------------------|-----------------------------------------|----------|
| 1     | Огляд існуючих вебдодатків для контролю  | 19.04.2024                              |          |
| 2     | Вибір архітектури вебдодатку             | 23.04.2024                              |          |
| 3     | Вибір мови програмування                 | 27.04.2024                              |          |
| 4     | Вибір середовища розробки                | 03.05.2024                              |          |
| 5     | Розробка серверної частини системи       | 11.05.2024                              |          |
| 6     | Налаштування Ruby on Rails               | 16.05.2024                              |          |
| 7     | Розробка бази даних                      | 19.05.2024                              |          |
| 8     | Створення користувацької панелі          | 22.05.2024                              |          |
| 9     | Тестування роботи системи                | 25.05.2024                              |          |
| 10    | Оформлення пояснювальної записки         | 29.05.2024                              |          |
| 11    | Захист роботи                            | 06.06.2024                              |          |
|       |                                          |                                         |          |
|       |                                          |                                         |          |

Студент



( підпис )

**Іван ЛУПОНОС**

(ініціали та прізвище)

Керівник проекту (роботи)



( підпис )

**Микола ХОХЛОВ**

(ініціали та прізвище)

## РЕФЕРАТ

ПЗ: 53 с., 34 рис., 2 табл., 17 джерел.

ВЕБДОДАТОК, КОРИСТУВАЧ, ФІНАНСИ, БАЗА ДАНИХ,  
JAVASCRIPT, HTML, RUBY ON RAILS, MICROSOFT SQL SERVER

Метою дипломного проєкту є розробка комп'ютерної системи для домашньої бухгалтерії.

В процесі роботи над проєктом було проведено аналіз існуючих вебдодатків, сформульовано функціональні вимоги щодо ведення витрат коштів у додатку, обґрунтування вибір технологій необхідних для розробки системи, таких як: архітектура вебдодатків, мова програмування, середовища розробки, база даних.

Було проведена робота над розробкою серверної частини, інтерфейсу користувача, розроблена структура бази даних, включаючи необхідні таблиці та зв'язки між ними, що дозволяє ефективно зберігати та організовувати дані.

Заключним етапом даної розробки було тестування додатку для ведення домашньої бухгалтерії, що допомагає забезпечити високу якість та надійність програмного забезпечення для створення нових категорій та додавання нових транзакцій.

## ЗМІСТ

|                                                         |    |
|---------------------------------------------------------|----|
| Вступ.....                                              | 7  |
| 1 Огляд вебдодатків.....                                | 9  |
| 1.1 Поняття «вебдодаток».....                           | 9  |
| 1.2 Огляд існуючих вебдодатків для контролю.....        | 11 |
| 1.2.1 Додаток «Money Manager Expense & Budget» .....    | 12 |
| 1.2.2 Додаток «Monefy» .....                            | 13 |
| 1.2.3 Додаток «Money Lover» .....                       | 14 |
| 1.2.4 Додаток «Budget» .....                            | 16 |
| 1.3 Функціональні вимоги до додатків.....               | 17 |
| 1.4 Принцип роботи вебдодатку .....                     | 18 |
| 2 Обґрунтування вибору технології розробки системи..... | 20 |
| 2.1 Вибір архітектури вебдодатку.....                   | 20 |
| 2.2 Вибір мови програмування.....                       | 25 |
| 2.2.1 Вибір HTML .....                                  | 26 |
| 2.2.2 Вибір фреймворку .....                            | 27 |
| 2.3 Вибір середовища розробки .....                     | 29 |
| 2.4 Вибір бази даних.....                               | 30 |
| 3 Розробка серверної частини системи.....               | 33 |
| 3.1 Встановлення Ruby on Rails .....                    | 33 |
| 3.1.1 Налаштування Rails .....                          | 35 |
| 3.1.2 Генерація команди Rails.....                      | 37 |
| 3.1.3 Маршрутизації Rails .....                         | 38 |
| 3.2 Розробка бази даних .....                           | 41 |
| 3.2.1 Налаштування SQL Server .....                     | 41 |
| 3.2.2 Структура бази даних.....                         | 42 |
| 3.2.3 Побудова зв'язків між таблицями.....              | 44 |
| 3.3 Інтерфейс користувача.....                          | 45 |

|                                                             |    |
|-------------------------------------------------------------|----|
| 4 Тестування роботи системи .....                           | 48 |
| Висновки.....                                               | 51 |
| Перелік джерел посилання .....                              | 52 |
| Додаток А .....                                             | 54 |
| Додаток Б.....                                              | 59 |
| Графічна частина:                                           |    |
| Плакати 1 - Інтерфейси додатків контролю витрат             |    |
| Плакати 2 - Характеристика фреймворків                      |    |
| Плакати 3 -Характеристика середовища розробки та бази даних |    |
| Плакати 4 - Інтерфейс додатку                               |    |

## ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

|      |                                |
|------|--------------------------------|
| CSS  | каскадні таблиці стилів        |
| DB   | база даних                     |
| HTML | мова розмітки                  |
| IDE  | інтегроване середовищеразробки |
| URL  | унікальний ідентифікатор       |
| БД   | база даних                     |

## ВСТУП

Розвиток комп'ютерних технологій революціонував наше повсякденне життя та вплинув на світ навколо нас. Впровадження комп'ютерів супроводжувалося значними змінами, які порівнюються з попереднім прогресом у сфері інженерії. В сучасному світі комп'ютери стали необхідною складовою у всіх аспектах нашого життя. Вони забезпечують миттєвий доступ до інформації з будь-якої точки світу, керують майже усіма механічними системами, включаючи автомобілі, літаки та інші прилади, а також управляють великими мережами, наприклад, електроенергією та розподільними системами. Комп'ютери є основою для роботи мобільних телефонів та Інтернету. Вони сприяють розвитку творчості серед художників, надаючи їм нові можливості, а також змінюють підхід до відпочинку та розваг. Крім того, комп'ютери кардинально змінили спосіб, яким вчені та інженери виконують свою роботу. У наступному розділі ми оглянемо відповідні історії комп'ютерної та електронної промисловості.

Стрімкий прогрес у технологіях відкриває перед нами неймовірні можливості, дозволяючи отримувати доступ до інформації у будь-якому місці та у будь-який час. Сучасні технології привели до виникнення різноманітних мультимедійних пристроїв, таких як смартфони та розумні годинники. Комп'ютери стають все швидшими, потужнішими та переноснішими, ніж коли-небудь раніше. Зміни, які вони приносять, відчутні у всіх аспектах нашого життя: від розваг та спілкування до споживання медіа. Тепер не потрібно відвідувати банк для здійснення фінансових операцій, таких як зняття грошей або переказ коштів. Багато банків вже надають можливість здійснювати транзакції через Інтернет, доступні для користувачів з усього світу.

За допомогою будь-якого пристрою можна здійснювати миттєві грошові перекази та здійснювати покупки, починаючи від одягу і доставки їжі до придбання продуктів, меблів та інших товарів, незалежно від місця знаходження. Технології також спрощують оплату рахунків: можна автоматично планувати платежі на момент їх настання, уникнувши забутих чеків. Завдяки мобільному телефону та банківським додаткам ви можете керувати всіма необхідними

оплатами та рахунками онлайн. Це відкриває нові можливості, забезпечуючи безпеку, мобільність і постійний доступ до фінансових послуг.

Зростаюча важливість мобільного управління фінансами та фінансової грамотності стає необхідною складовою нашого повсякденного життя, оскільки контроль над витратами набуває особливого значення. Розуміння грошових потоків допомагає кожному усвідомити, як саме вони використовуються, що є дуже корисним для ефективного управління фінансами. Проте, в умовах великої кількості банків та фінансових додатків це стає все складніше.

Використання програм для домашньої бухгалтерії може значно полегшити життя. Вони допомагають користувачам краще розуміти свої фінансові потоки, виявляти непотрібні витрати, контролювати досягнення фінансових цілей та планувати майбутні витрати.

## 1 ОГЛЯД ВЕБДОДАТКІВ

### 1.1 Поняття «вебдодаток»

Вебдодатки є важливою складовою сучасного інтернет-простору, оскільки вони дозволяють реалізувати різноманітні функції та послуги онлайн. Їх характеристики, особливості та взаємодія можуть значно відрізнятися в залежності від конкретного вебдодатку і його цілей [1].

Вебдодатки можуть виконувати різноманітні функції, від обробки даних до інтерактивних інтерфейсів для користувачів. Часто вебдодатки розробляються для різних платформ, щоб забезпечити доступність для користувачів з різних пристроїв (комп'ютери, смартфони, планшети тощо).

Вебдодатки можуть бути статичними (з фіксованим контентом) або динамічними (з можливістю взаємодії користувача з сервером для отримання актуальної інформації).

До особливостей можна зазначити наступне:

- можливість доступу через будь-який пристрій з веббраузером;
- інтерактивність і можливість збереження даних на сервері;
- автоматичне оновлення без необхідності встановлення на пристрій користувача;
- масштабованість: можливість працювати з великою кількістю користувачів одночасно без втрати продуктивності.

Вебдодатки характеризуються вищим рівнем взаємодії та функціональності. Вони здатні обробляти введення користувача, виконувати складні операції та оновлювати інтерфейс без необхідності перезавантаження сторінки. Це позитивно впливає на користувацький досвід, роблячи його більш плавним та динамічним.

До особливостей вебдодатків можна віднести наступні [2]:

- обробка запитів від клієнтів, виконання необхідних обчислень, формування вебсторінки та її передача клієнту через мережу за допомогою

протоколу HTTP;

- можливість взаємодії з іншими сервісами, такими як бази даних або інші вебдодатки, що знаходяться на інших серверах;

- незалежність від операційної системи клієнта: додаток розробляється один раз для будь-якої платформи і розгортається на ній, але різні реалізації HTML, CSS, DOM та інших специфікацій у браузері може ускладнювати розробку та підтримку вебдодатків;

- простота використання: користувачі повинні легко розуміти, як користуватися сайтом.

Перевагами, за яких обирають вебдодатки для створення різноманітних сервісів є:

- доступність на всіх пристроях і браузерах, що дозволяє користувачам відкривати свої облікові записи на будь-якому пристрої без прив'язки до конкретного гаджета;

- економія пам'яті, особливо для мобільних телефонів;

- легкість масштабування порівняно з мобільною версією;

- автоматичне оновлення інформації без необхідності вручного встановлення нових версій;

- економія витрат на обслуговування вебпродукту порівняно з мобільною версією;

- вищий рівень безпеки завдяки розміщенню даних на віддаленому сервері, що зменшує вразливість до зловмисного втручання порівняно з мобільними додатками.

У вебдодатку користувач не лише переглядає інформацію, але й активно взаємодіє з нею, здійснюючи такі дії, як бронювання квитків, покупка товарів, проведення фінансових транзакцій, а також листування тощо.

Однією з особливостей вебдодатків є розширена система аутентифікації. Після входу в свій обліковий запис користувач отримує можливість створювати, завантажувати і виконувати різні маніпуляції з контентом.

## 1.2 Огляд існуючих вебдодатків для контролю

Планування особистих або сімейних фінансів завжди залишається важливим. Важливо пам'ятати про своєчасну оплату рахунків, кредитів та страхувань. Програми для управління фінансами значно спрощують і можуть навіть оптимізувати цей процес [3]. Вони, будь то безкоштовні або платні, пропонують різноманітні функції для ефективного фінансового управління. Однак необхідно регулярно вносити актуальні дані в ці програми. Вебдодатки для контролю за витратами можуть надати значні переваги для домашньої бухгалтерії, ось деякі з можливостей:

- зручний доступ - завдяки вебдодаткам користувачі можуть мати доступ до своєї бухгалтерії з будь-якого пристрою, що підключений до Інтернету, це дозволяє вносити та переглядати дані в будь-який час і з будь-якого місця, що вельми зручно, особливо для тих, хто часто подорожує або просто хоче мати постійний доступ до своїх фінансів;

- автоматизація - вебдодатки можуть автоматизувати багато процесів у веденні бухгалтерії, таких як регулярні платежі, повторювані витрати тощо, це допомагає зекономити час та зменшити ймовірність помилок в обліку;

- аналіз та звітність - багато вебдодатків надають інструменти для аналізу витрат та генерування звітів, користувачі можуть аналізувати свої витрати за різними категоріями, переглядати тренди, виявляти надмірні витрати та знаходити способи для їх оптимізації;

- безпека - вебдодатки можуть надавати різні засоби для захисту конфіденційності та безпеки фінансових даних користувачів, це може включати шифрування даних, двофакторну аутентифікацію та інші заходи безпеки;

- спільний доступ - деякі вебдодатки дозволяють користувачам спільно керувати своєю бухгалтерією з іншими членами сім'ї або колегами, це дозволяє забезпечити більш ефективний контроль витрат та спільне планування фінансів.

Для того щоб зрозуміти принцип дії та обрання свого вебзастосунку

розглянемо декілька найпоширеніших вебдодатків серед користувачів.

### 1.2.1 Додаток «Money Manager Expense & Budget»

Це мобільний додаток, який допомагає користувачам вести облік своїх фінансів, витрат та бюджетів [4]. Розроблений для мобільних пристроїв Android та iOS. Додаток дозволяє вести облік доходів і витрат, створювати бюджети, аналізувати витрати за різними категоріями, генерувати звіти і діаграми для кращого розуміння фінансової ситуації (рис.1.1).

До характеристик Money Manager Expense & Budget можна віднести наступні:

- ведення обліку витрат та доходів: користувачі можуть додавати витрати та доходи, вказуючи суму, дату та категорію;
- бюджетування: можливість створювати бюджети на різні категорії витрат і контролювати їх виконання;
- аналітика та звіти: генерування звітів і діаграм для аналізу фінансових потоків, витрат за певний період, порівняння доходів та витрат тощо;
- синхронізація: можливість синхронізації даних між різними пристроями, щоб мати доступ до них з будь-якого мобільного пристрою.

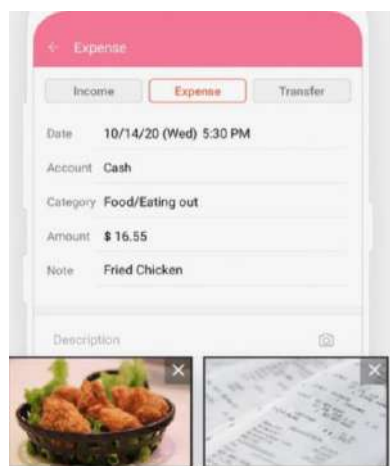


Рисунок 1.1 – Інтерфейс мобільного додатку Money Manager Expense & Budget

Особливості:

- простий та зручний інтерфейс: інтуїтивно зрозуміле розташування

функцій і легкість використання;

- підтримка різних валют: можливість вести облік фінансів в різних валютах для користувачів з різних країн;

- можливість встановлення нагадувань: додаток дозволяє встановлювати нагадування про заплановані платежі або події.

Користувачі взаємодіють з "Money Manager Expense & Budget" через мобільний додаток на своєму смартфоні або планшеті. Дані зберігаються на пристрої користувача або можуть синхронізуватися з хмарним сховищем для доступу до них з різних пристроїв.

Отже, "Money Manager Expense & Budget" - це потужний інструмент для керування фінансами, який дозволяє користувачам ефективно вести облік своїх витрат, створювати бюджети та аналізувати фінансові потоки.

### **1.2.2 Додаток «Monefy»**

Цей додаток є одним із найпопулярніших серед мобільних додатків. Він приваблює користувачів зручним та легким інтерфейсом, який дозволяє миттєво додавати нові записи та контролювати усі витрати. Monefy надає можливість планувати витрати відповідно до встановленого бюджету, а не всієї суми доходів. Крім того, додаток має функцію синхронізації через Dropbox, що дозволяє додавати записи з різних пристроїв або відстежувати загальний сімейний бюджет.

Інтерфейс додатку відзначається зручністю та інтуїтивністю. Усі категорії витрат представлені на головному екрані у вигляді зображень. Діаграма розподілу витрат також відображається на головному екрані, що дає змогу швидко переглянути, на яку категорію витрачено найбільше коштів [4].

Користувач може встановити звітний період на день, тиждень або місяць, а також розділити витрати на карткові та готівкові.



Рисунок 1.2 – Вікно додатку Monefy

За додаткову плату доступна преміум версія застосунку, яка включає в себе можливість додавання власних категорій, синхронізацію з декількома пристроями та підтримку різних валют.

В ході тривалого використання Monefy було виявлено, що в ньому відсутня функція запису витрат після їх здійснення. Наприклад, якщо в останній день місяця витрату не вдалося внести в систему, а вона була зроблена наступного дня, вона автоматично зараховується до поточного місяця.

### 1.2.3 Додаток «Money Lover»

Додаток приваблює цікавим дизайном та розмаїттям можливостей. Ви розподіляєте свої кошти між різними "гаманцями" та відстежуєте їх стан. Головною особливістю Money Lover є можливість фіксувати ваші борги та регулярні платежі, нагадуючи вам про них у відповідний момент, наприклад, про оплату комунальних послуг або внесення платежу за кредитом (рис.1.3).

Додаток також дозволяє створювати плани для великих покупок або

накопичень - кошти, які ви хочете відкласти щомісяця, автоматично виключаються з вашого бюджету, щоб ви не могли їх витратити. Крім того, ви можете створювати окремі "гаманці" та відстежувати, скільки ви витратили, наприклад, на відпочинок [5].



Рисунок 1.3 – Інтерфейс додатка Money Lover

На головному екрані програми Money Lover відображаються загальні витрати у вигляді графіка, який порівнюється з попередніми місяцями. Нижче можна побачити діаграму, яка представляє список найбільших витрат.

Проте, варто відзначити, що програма має кілька недоліків, таких як не дуже зручний інтерфейс для додавання витрат або доходів, а також обмеження на використання кількох рахунків або "гаманців", за які потрібно буде заплатити. Вартість включає в себе відсутність реклами та можливість встановлення лімітів для витрат у різних категоріях, проте в безкоштовній версії це можливо лише для

однієї категорії.

#### 1.2.4 Додаток «Budget»

Цей застосунок дозволяє відстежувати історію вашого доходу та витрат, будувати графіки за днями і категоріями. Крім того, можна використовувати його для спільного управління бюджетом і встановлення цілей для планування витрат або накопичень (рис.1.4).

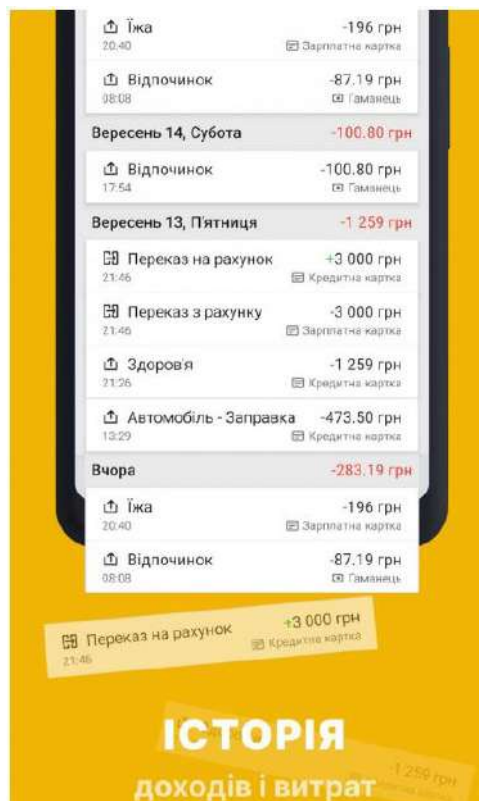


Рисунок 1.4 – Інтерфейс додатку Budget

В даний момент програма доступна лише для користувачів Android. Надається можливість скористатися безкоштовним пробним періодом тривалістю 45 днів. Після цього доступні різні варіанти підписки: щомісячна за 50 грн з автоматичним продовженням, щотримісячна за 150 грн без автоматичного продовження або щорічна за 500 грн з автоматичним продовженням [5].

Таким чином, вебдодатки для контролю витрат можуть значно спростити та покращити процес управління фінансами, забезпечуючи зручний доступ, автоматизацію, аналітику, безпеку та можливість спільного використання даних.

### 1.3 Функціональні вимоги до додатків

Функціональні вимоги для вебдодатків для контролю витрат у домашній бухгалтерії можуть включати різні функції, які забезпечують користувачам зручний та ефективний спосіб керування своїми фінансами [6].

Деякі з них:

- реєстрація користувача - додаток повинен надати можливість користувачам створити обліковий запис і увійти в систему;
- додавання витрат і доходів - користувачі повинні мати можливість додавати витрати і доходи, вказуючи суму, дату, категорію тощо;
- створення бюджетів - можливість створювати бюджети для різних категорій витрат, встановлювати ліміти та контролювати їх виконання;
- аналітика витрат - генерація звітів і діаграм для аналізу витрат за різними категоріями, перегляд трендів, виявлення надмірних витрат тощо;
- синхронізація даних - можливість синхронізувати дані між різними пристроями для забезпечення доступу до них з будь-якого місця;
- нагадування про платежі - додаток може надавати можливість встановлювати нагадування про заплановані платежі або інші фінансові події;
- безпека даних - забезпечення безпеки фінансових даних користувачів шляхом шифрування та інших заходів безпеки;
- експорт та імпорт даних - можливість експортувати імпортувати дані для резервного копіювання або переміщення між різними системами;
- підтримка різних платіжних систем - якщо додаток має функціональність для оплати рахунків або переказу грошей, важливо мати підтримку різних платіжних систем.

Конкретний набір функцій може варіюватися в залежності від потреб користувачів та стратегії розробки.

## 1.4 Принцип роботи вебдодатку

Вебдодаток - це програма, яка розміщується на вебсервері та надає функціональні можливості користувачам через їх веббраузери. Робота вебдодатка ґрунтується на постійному обміні інформацією між браузером та сервером, що дозволяє взаємодіяти з додатком та використовувати його функціонал через Інтернет.

Коли користувач відкриває вебдодаток у своєму веббраузері, він вводить адресу додатку, і браузер відправляє запит до сервера, який містить цю адресу. Сервер отримує цей запит і знаходить вебдодаток, який повинен обробити запит [7].

Вебдодаток обробляє запит, виконує різні дії та створює відповідь, яку сервер відправляє назад до браузера. Браузер отримує цю відповідь і відображає її на екрані користувача. Користувач може взаємодіяти з вебдодатком, заповнюючи форми, натискаючи кнопки та виконуючи інші дії.

Кожна дія користувача породжує новий запит до сервера, який оброблює його та повертає відповідь. Такий обмін запитами та відповідями відбувається протягом усього процесу використання додатком.

Створення вебдодатків включає в себе застосування різноманітних технологій. Зазвичай це комбінація мов програмування для серверної частини (наприклад, PHP, Python, Ruby, Java), систем управління базами даних (наприклад, MySQL, PostgreSQL), та мов програмування для клієнтської частини (наприклад, HTML, CSS, JavaScript).

Однією з ключових переваг вебдодатків є їх робота у вебсередовищі, що не потребує встановлення додаткового програмного забезпечення на пристроях користувачів. Крім того, вони забезпечують централізоване управління, оскільки оновлення та зміни можуть бути внесені на сервері і автоматично відображатися для користувачів, не потребуючи оновлення самого додатка на кожному пристрої..

Отже, вебдодатки є зручним та широко використовуваним методом отримання доступу до функціональності через веббраузери, оскільки вони взаємодіють з користувачем шляхом постійного обміну інформацією між браузером та сервером.

Вебдодатки за контролем витрат працюють зазвичай за такою схемою:

- реєстрація користувача в системі - користувач створює обліковий запис, вказуючи електронну пошту та пароль;

- після входу в систему користувач може почати вводити свої витрати і доходи (він вказує суму, дату, категорію витрати або доходу, можливо додає короткий коментар);

- користувач може створювати бюджети для різних категорій витрат, для цього він встановлює ліміт витрат для кожної категорії і може встановлювати період бюджету;

- аналіз витрат - вебдодаток збирає введені дані про витрати і дає змогу користувачеві аналізувати їх (це може бути візуалізовано через діаграми, графіки, та таблиці, що дозволяє зрозуміти, на що йде найбільше грошей, які категорії потребують уваги тощо);

- при необхідності вебдодаток може надсилати нагадування користувачеві про підходящі терміни платежів, а також повідомлення, коли витрати перевищують встановлені ліміти бюджету;

- збереження даних користувачів -дані повинні бути безпечно збережені на серверах додатка і синхронізовані між різними пристроями користувача, які використовують додаток;

- спільний доступ до бюджетів і витрат для кількох користувачів, що дозволяє родинам або командам керувати фінансами разом.

Кожен додаток може мати свої особливості та розширені функції в залежності від потреб користувачів та стратегії розробки.

## 2 ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЇ РОЗРОБКИ СИСТЕМИ

При розробці даної комп'ютерної системи, яка включає в себе створення вебдодатку домашньої бухгалтерії будуть використані різні технології для реалізації різноманітних функцій та забезпечення ефективної роботи додатку. В даному розділі буде обгрунтовано вибір стосовно кожного компоненту системи, такі як: вебфреймворки, база даних, архітектура, мова програмування.

### 2.1 Вибір архітектури вебдодатку

Архітектура вебдодатків визначає взаємодію компонентів, їх структуру та спосіб співпраці для досягнення ефективності та можливості розширення. Один з найпоширеніших підходів - це модель клієнт-сервер, де вебдодаток складається з клієнтської та серверної частин. Клієнт (зазвичай веббраузер) відображає інтерфейс користувача та збирає дані, в той час як сервер обробляє запити, виконує бізнес-логіку та надає необхідні дані. У вебдодатках існують різні компоненти, які співпрацюють для забезпечення функціональності [8].

Основні складові включають клієнську та серверну частини.

Клієнтська частина (Front-end) - це інтерфейс, який відображається у веббраузері. Вона використовує HTML, CSS та JavaScript для створення інтерактивності та користувацького досвіду.

Серверна частина (Back-end) - це сервер, який обробляє запити, зберігає дані, виконує бізнес-логіку та повертає відповіді. Ця частина може включати вебсервери, бази даних, мови програмування та інші технології.

У сфері архітектури вебдодатків використовуються різноманітні шаблони, які допомагають організувати та структурувати компоненти додатку. Деякі з найпоширеніших архітектурних шаблонів включають монолітну архітектуру,

клієнт-серверну архітектуру, розподілену архітектуру та мікросервісну архітектуру. Загалом існують три типи верхньорівневої архітектури вебдодатків: моноліт, мікросервіси та серверлес.

Монолітна архітектура - це підхід до розробки програмного забезпечення, в якому всі компоненти додатку (такі як інтерфейс, база даних, бізнес-логіка) розглядаються як єдина, нероздільна система. У такій архітектурі весь код зазвичай розміщується в одному монолітному додатку та розгортається на одному сервері. Цей підхід дозволяє простіше управляти та розгортати додаток, але може призводити до проблем з масштабованістю та підтримкою великих проєктів через залежності між компонентами і складність у внесенні змін (рис2.1).

Взагалі прийнято думку, що ця архітектурна концепція застаріла, але вона виявляється дуже ефективною для побудови певних проєктів.

Монолітна архітектура має свої переваги і недоліки:

Переваги монолітної архітектури:

- простота розробки - весь код зазвичай знаходиться в одному монолітному додатку, що полегшує розробку та розгортання;
- простота вдосконалення -несення змін у монолітний додаток зазвичай менше складне, оскільки весь код знаходиться в одному місці;
- легше тестування - тести можуть бути здійснені централізовано, що спрощує виявлення помилок та управління тестами;
- витрати на інфраструктуру - одне додаткове середовище зазвичай потребує менше ресурсів порівняно з декількома мікросервісами або розподіленими системами.

Недоліки монолітної архітектури:

- складність масштабування - підвищення навантаження вимагає масштабування всього додатку, що може бути складним;
- проблеми з підтримкою - з часом монолітний додаток може стати важким для розуміння та підтримки, особливо у великих проєктах;

- залежність від технологій - часто монолітні додатки побудовані на одній технологічній стеку, що може призвести до затримок у впровадженні нових технологій та розвитку;

- однопоточність - зазвичай використовується один потік виконання, що може стати проблемою в разі великої кількості запитів або великого обсягу даних.

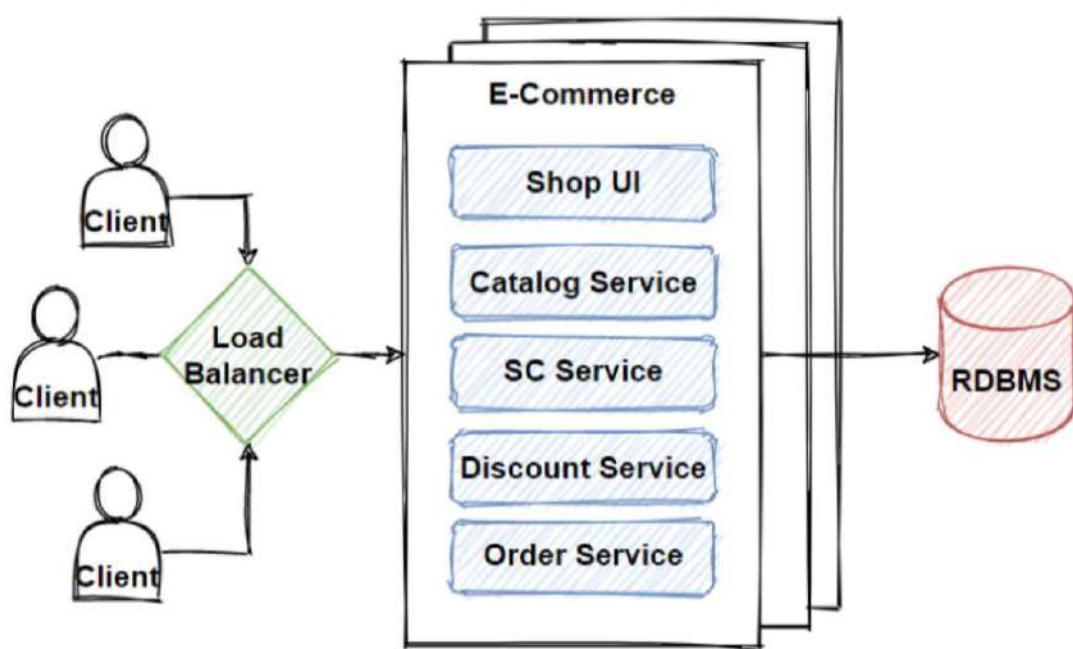


Рисунок 2.1 – Приклад схеми монолітної архітектури

Мікросервісна архітектура - це підхід до розробки програмного забезпечення, в якому додаток розбивається на невеликі, незалежні сервіси, що функціонують як окремі процеси і взаємодіють між собою за допомогою мережі. Кожен мікросервіс відповідає за виконання конкретної функції або послуги, що дозволяє забезпечити більшу гнучкість, масштабованість та швидкість розробки та впровадження програмного забезпечення (рис. 2.2).

Зараз це дуже популярний та актуальний підхід у сфері архітектури, проте варто відзначити, що він має свої обмеження, і, на мою думку, він не є найкращим вибором для невеликих та середніх проєктів [9].

Переваги мікросервісної архітектури включають:

- гнучкість - кожен мікросервіс є автономною системою, що дозволяє внесення змін без впливу на інші компоненти;

- масштабованість - можливість масштабувати окремі частини системи, що дозволяє оптимізувати витрати та ресурси;
- гнучкість технологій - кожен мікросервіс може бути реалізований з використанням будь-якої мови програмування та технологій, що полегшує розробку та підтримку;
- надійність - випадкові відмови в одному мікросервісі не впливають на роботу інших, що підвищує стійкість системи в цілому.

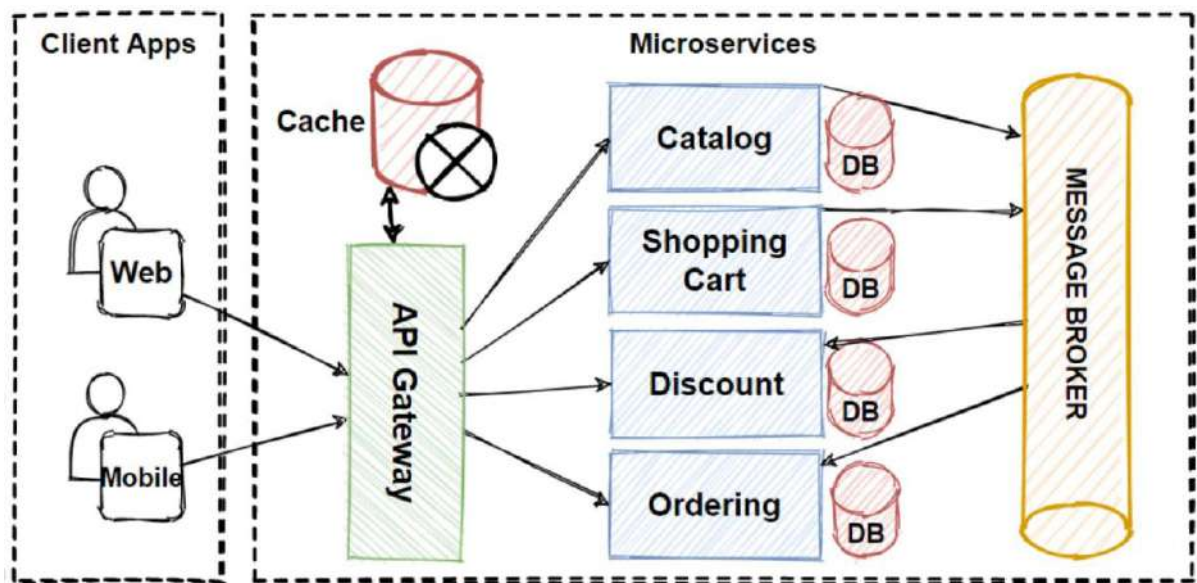


Рисунок 2.2 – Приклад мікросервісної архітектури

Недоліки:

- процес розробки - створення мікросервісів є складним процесом через необхідність розробки кількох підсистем та забезпечення їх взаємодії;
- комунікація між командами - часто виникає ситуація, коли окремі команди працюють над різними підсистемами, тому потрібно забезпечити ефективну комунікацію між ними для координації робіт та забезпечення взаємодії підсистем;
- відлагодження - може бути складним завданням, оскільки необхідно визначити, який саме сервіс викликав проблему та виявити причини цього збою.

Серверлес - архітектурний підхід, при якому розроблені програмні компоненти виконуються в середовищі хмарної платформи, де відповідальність за

управління інфраструктурою, такою як сервери та обчислювальні ресурси, покладається на хмарного провайдера. У цьому підході розробники можуть зосередитися на написанні функціонального коду, не турбуючись про управління інфраструктурою, що може значно полегшити процес розробки та виконання програмного забезпечення (рис.2.3).

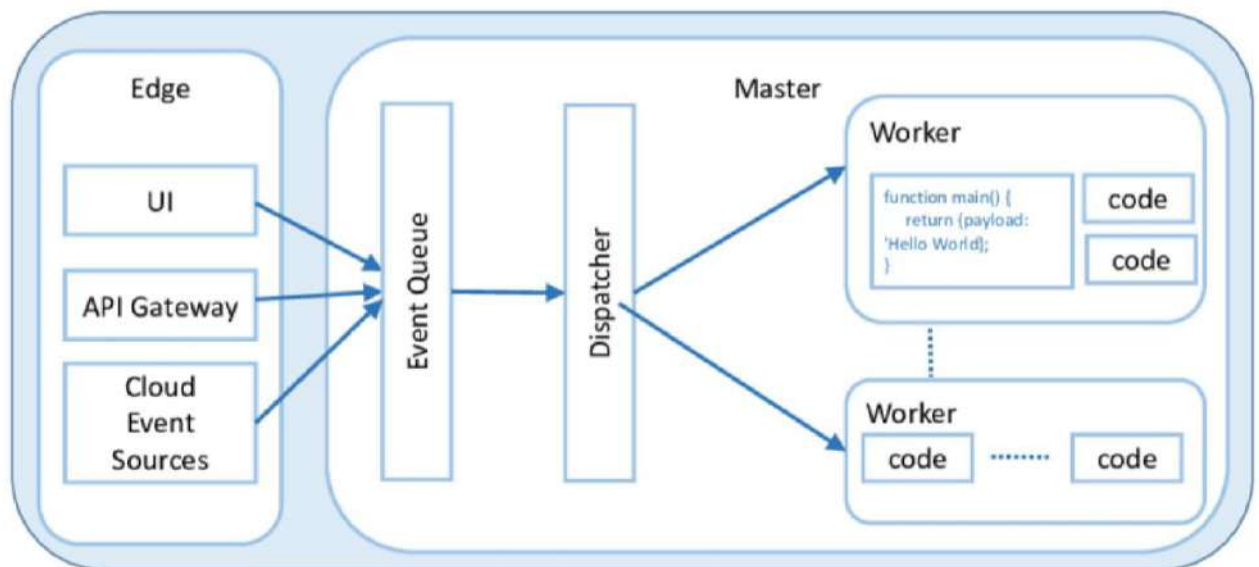


Рисунок 2.3 – Приклад архітектури серверлесс

Звичайні переваги та недоліки серверлесс архітектури включають:

Переваги даної архітектури:

- менші витрати - вартість використання хмарних послуг у серверлесс архітектурі залежить від кількості виконаних функцій, що може призвести до економії коштів порівняно з традиційними серверами;
- більша масштабованість - дозволяє автоматично масштабувати ресурси в залежності від потреб, що дозволяє легко впоратися зі збільшенням навантаження;
- швидкий запуск - розгортання та виконання функцій може відбуватися майже миттєво, що дозволяє швидко впроваджувати зміни та оновлення.

Недоліки:

- складніше налаштування - розробка та налаштування серверлесс додатків складніша через необхідність розбиття програмного коду на функції та

управління їх виконанням;

- проблеми з відлагодженням - може бути важким через розподілену природу виконання функцій та складність відстеження помилок;

- залежність від провайдера - використання хмарних послуг означає залежність від конкретного хмарного провайдера, що може виникнути проблеми при перенесенні додатків на інші платформи.

Проаналізувавши вищанаведні архітектури було вирішено використати в даній роботі мікросервісну архітектуру, яка найкраще підходить для реалізації системи через можливості масштабування окремих сервісів та декомпозиції по функціоналу.

## **2.2 Вибір мови програмування**

Вибір мови програмування визначається завданням або проектом, і це ключове рішення, яке може вплинути на продуктивність, якість та ефективність розробки програмного забезпечення. При виборі мови програмування слід враховувати такі фактори, як вимоги проекту, досвід розробника, екосистема та підтримка, швидкість та продуктивність, масштабованість та підтримка [10].

Кожна мова програмування має свої унікальні особливості та застосування, тому важливо провести аналіз відповідно до потреб вашого проекту та знань вашої команди, щоб зробити найкращий вибір. Частота використання мов програмування залежить від багатьох факторів, включаючи область застосування, популярність, розвиток екосистеми та інші чинники. Однак, деякі мови програмування використовуються частіше через їх властивості та можливості, які відповідають потребам багатьох розробників.

Наприклад, «JavaScript» є основною мовою для розробки вебдодатків та фронтенду, а «Python» застосовується для широкого спектру завдань, включаючи веброзробку, наукові обчислення, штучний інтелект та інше. «Java»

використовується для розробки великих підприємницьких систем та мобільних додатків, а «C#» - для програмного забезпечення на платформі Microsoft .NET. «C++» застосовується для створення швидких та ефективних систем, а «PHP» - для розробки вебдодатків та вебсайтів.

Це лише кілька прикладів з найпопулярніших мов програмування, і вибір мови може значно відрізнятись залежно від конкретного проєкту та його потреб. Для реалізації даного завдання буде використано мову програмування JavaScript.

### **2.2.1 Вибір HTML**

Для створення та представлення вебдодатків у веббраузері використовують спеціальні інструменти - розмітки. Існує певна кількість стандартних інструментів, а саме:

- HTML (HyperText Markup Language) використовується для створення структури вебсторінок, визначає скелет сторінки, включаючи розміщення тексту, зображень, відео, посилань і форм;

- XHTML (eXtensible HyperText Markup Language) - використовує синтаксис XML, має більш строгі правила щодо синтаксису та валідності документів;

- HTML5 - включає багато нових функцій і можливостей, таких як підтримка відео та аудіо, відтворення з мультимедіа, покращений API для роботи з вебдодатками, підтримка мобільних пристроїв та багато іншого;

- Markdown - це легковагий мова розмітки, яка концентрується на простоті та зручності написання тексту, не призначений для створення вебсторінок як таких, він широко використовується для написання контенту для вебсайтів, в блогах, на форумах тощо.

Кожен з цих інструментів має свої власні переваги і використовується в різних ситуаціях в залежності від потреб проєкту та вподобань розробника.

Проаналувавши вищенаведені інструменти було вирішено зупинитися на більш перевіреному варіанті як звичайний HTML.

HTML є ключовою складовою веброботи і використовується в широкому спектрі інтернет-проєктів.

HTML використовує набір тегів (tags), які оточують різні елементи вебсторінки та визначають їхню роль і вигляд. Кожен тег має певне значення і може мати атрибути, що додатково визначають його характеристики. Наприклад, тег <p> використовується для визначення абзацу тексту, тег <img> для вставки зображень, а тег <a> для створення посилань [11].

HTML також використовується для створення базової структури вебсторінки, такої як заголовки, списки, таблиці і форми. Крім того, він дозволяє вбудовувати відео, аудіо та інші мультимедійні елементи у вебсторінки.

HTML використовується у всьому Інтернеті, включаючи вебсайти, електронну пошту, вебдодатки, мобільні додатки, вбудовані системи.

Щоб вебсторінка була повністю функціональною і привабливою, HTML зазвичай комбінується з CSS (Cascading Style Sheets) для стилізації та зовнішнього оформлення, а також з JavaScript для додавання інтерактивності та динаміки. Це дозволить визначати зовнішній вигляд і форматування HTML-документів.

### **2.2.2 Вибір фреймворку**

Фреймворки, або рамкові системи, є готовими середовищами для розробки, що надають набори інструментів та структури. Розробникам достатньо налаштувати їх під свої бізнес-потреби.

Фреймворк – це комплекс інструментів, бібліотек та правил, призначених для створення програмного забезпечення. Він визначає структуру, забезпечуючи взаємодію компонентів програми, шаблони для інтерфейсів та методи роботи з базами даних та іншими ресурсами [12].

Ці засоби спрощують процес розробки, забезпечуючи готові рішення для типових задач. Це дозволяє розробникам зосередитися на бізнес-логіці програми, не витрачаючи час на рутинні завдання.

Фреймворки мають високу продуктивність і гнучкість, дозволяючи створювати якісний каркас продукту без втрати його функціоналу.

Загалом, фреймворки:

- спрощують та прискорюють розробку;
- реалізують базовий функціонал вебпродукту;

- прискорюють процес роботи;
- знижують ймовірність помилок.

Існує багато фреймворків для різних мов програмування та завдань, які можна поділити на багатофункціональні та мікрофреймворки. Основна мета фреймворку – забезпечити ефективну роботу продукту. Наприклад, бекенд-фреймворки допомагають організувати бізнес-логіку та роботу з даними, забезпечуючи безпечну передачу даних та кібербезпеку.

Найпопулярніші фреймворки представлені на рисунку 2.4:

|               |                                                                                                                                |
|---------------|--------------------------------------------------------------------------------------------------------------------------------|
| Laravel       | процює з PHP, підходить для написання простих проєктів, простий у роботі                                                       |
| Django        | популярна рамкова система для Python, має стандартну структуру, спрямована на масштабування проєктів у процесі роботи          |
| Symfony       | старий фреймворк, який підходить для великих порталів та складних продуктів, гнучкий та зручний                                |
| Ruby on Rails | призначений для роботи з Ruby, підходить для створення проєктів, які мають витримувати великі навантаження та працювати швидко |

Ак

Рисунок 2.4 - Найпопулярніші фреймворки та їх опис

Після аналізу було визначено, що найкращим вибором для створення серверної частини додатку, реалізації логіки, обробки запитів та доступу до бази даних буде використання фреймворку Ruby on Rails.

Переваги цього вибору стали його простота, ефективність та швидкість у розробки вебдодатків.

Також слід зазначити наступні основні риси та принципи, які визначають Rails:

- вбудовує багато заздалегідь визначених конвенцій для розробки, що дозволяє розробникам уникнути багатьох ручних конфігурацій, що прискорює процес розробки, оскільки розробники можуть зосередитися на логіці додатку, а

не на встановленні і налаштуванні фреймворку;

- модель-вид-контролер (Model-View-Controller, MVC) - Rails базується на архітектурному шаблоні MVC, що допомагає розділити логіку додатку на три основні компоненти: моделі для роботи з базою даних, види для відображення інтерфейсу користувача та контролери для обробки запитів;

- генератори коду - Rails надає ряд генераторів коду, які автоматизують створення стандартних елементів додатку, таких як моделі, контролери, види тощо, це сприяє швидкій розробці, зменшуючи кількість написаного вручну коду;

- RESTful маршрутизація - пропонує підтримку RESTful маршрутизації, що дозволяє організувати логіку додатку навколо стандартних HTTP-запитів, таких як GET, POST, PUT та DELETE;

- активний запис (Active Record) - дозволяє розробникам взаємодіяти з базою даних за допомогою простих Ruby-об'єктів.

Rails знаменитий своєю простотою та продуктивністю, що робить його відмінним вибором для швидкої розробки вебдодатків будь-якої складності.

Rails - це найпопулярніший фреймворк у світі Ruby, який має великий потенціал для створення надійного фундаменту в архітектурі продукту, що дозволяє подальше його масштабування.

## 2.3 Вибір середовища розробки

Правильно обране інтегроване середовище розробки (IDE) може значно полегшити вашу роботу. IDE для розробки програмних мов - це програмне забезпечення, яке дозволяє програмістам створювати, тестувати та налагоджувати код [13].

Існує велика кількість IDE для написання, управління та налагодження програмного коду. У таблиці 2.1 наведено кілька популярних IDE з їх коротким описом.

Таблиця 2.1 - Середовища розробки (IDE)

|                              |                                                                                                                                                                                                                                              |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Visual Studio Code (VS Code) | - безкоштовне та легке у використанні середовище розробки, розроблене компанією Microsoft. VS Code має велику кількість розширень, що робить його дуже гнучким для програмістів будь-яких мов програмування;                                 |
| IntelliJ IDEA                | - потужне комерційне IDE, що розроблене компанією JetBrains. Воно підтримує багато мов програмування, включаючи Java, Kotlin, Python, JavaScript та багато інших. IntelliJ IDEA має багато корисних функцій для роботи з великими проектами; |
| Eclipse                      | - відкрите безкоштовне середовище розробки, що підтримує багато мов програмування, таких як Java, C++, Python тощо. Eclipse має широкий вибір плагінів і розширень, що робить його гнучким та налаштовуваним;                                |
| Sublime Text                 | - легке, швидке та потужне редактор коду, який підтримує багато мов програмування. Sublime Text має багато корисних функцій, таких як маркування синтаксису, відкладення, широкий вибір плагінів і тем;                                      |

Для виконання нашого завдання ми вибрали Visual Studio - легке у використанні та зручне середовище для написання, відлагодження та управління програмним кодом. Воно володіє розгалуженою набірм інструментів та функціональних можливостей, що спрощують процес розробки.

## 2.4 Вибір бази даних

Вибір бази даних залежить від конкретних вимог вашого проєкту, включаючи масштаб проєкту, типи даних, які потрібно зберігати та обробляти, швидкість доступу до даних, потреби в резервному копіюванні та відновленні, а також ваші знання та досвід з конкретними системами баз даних.

Бази даних (БД) вирішують різноманітні завдання у сферах бізнесу, науки, технологій та інших галузях [14]. Нижче перераховано деякі з найбільш

поширених задач, які вони вирішують:

- збереження даних: дозволяють зберігати великі обсяги даних у структурованому форматі, що спрощує їхнє управління;
- організація даних: допомагають організувати дані у логічні групи та зв'язки, що спрощує доступ та операції з ними;
- швидкий доступ до даних: надають ефективний доступ до даних за допомогою механізмів індексації та оптимізації запитів;
- множинність користувачів та одночасний доступ: дозволяють багатьом користувачам одночасно отримувати доступ до даних;
- забезпечення цілісності даних: запобігають помилкам та дублюванню даних за допомогою механізмів контролю цілісності;
- забезпечення безпеки даних: встановлюють права доступу для різних користувачів та ролей для захисту від несанкціонованого доступу;
- аналітика та звітність: виконують аналітичні запити та генерують звіти для прийняття стратегічних рішень;
- інтеграція з іншими системами: обмінюються даними та спільно виконують бізнес-процеси з іншими програмними засобами та системами;
- забезпечення резервного копіювання та відновлення: надають можливість резервного копіювання та відновлення даних в разі втрати або пошкодження;
- підтримка розширюваності та масштабованості: легко розширюються та масштабуються для відповіді на зростаючі потреби та обсяги даних.

Найчастіше при створенні бази даних використовують MySQL, PostgreSQL, SQLite, Microsoft SQL Server. Їх характеристика наведена у таблиці 2.2.

Таблиця 2.2 - Характеристика баз даних

| MySQL                                                                                                                                                                                                                                          |                                                                                                                                                                                                                       |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Переваги:</p> <ul style="list-style-type: none"> <li>- легка встановлюваність та налаштування;</li> <li>- велика швидкодія;</li> <li>- широко підтримується вебзастосунками;</li> </ul>                                                     | <p>Недоліки:</p> <ul style="list-style-type: none"> <li>- обмежені можливості щодо аналітики та розширених функцій;</li> <li>- менша підтримка для складних транзакцій порівняно з іншими СУБД;</li> </ul>            |
| PostgreSQL                                                                                                                                                                                                                                     |                                                                                                                                                                                                                       |
| <p>Переваги:</p> <ul style="list-style-type: none"> <li>- повна підтримка стандартів SQL;</li> <li>- підтримка географічних об'єктів та розширення для аналітики даних;</li> <li>- висока надійність та безпека</li> </ul>                     | <p>Недоліки:</p> <ul style="list-style-type: none"> <li>- більш високі вимоги до ресурсів порівняно з деякими іншими СУБД;</li> <li>- якщо не налаштована правильно, може бути складною у використанні;</li> </ul>    |
| SQLite                                                                                                                                                                                                                                         |                                                                                                                                                                                                                       |
| <p>Переваги:</p> <ul style="list-style-type: none"> <li>- проста використання інтеграції;</li> <li>- не вимагає серверу баз даних, або налаштування;</li> <li>- підтримується в більшості мов програмування;</li> </ul>                        | <p>Недоліки**:</p> <ul style="list-style-type: none"> <li>- не підходить для великих проєктів з високими навантаженнями;</li> <li>- має обмежену підтримку функцій та типів даних порівняно з іншими СУБД;</li> </ul> |
| Microsoft SQL Server                                                                                                                                                                                                                           |                                                                                                                                                                                                                       |
| <p>Переваги:</p> <ul style="list-style-type: none"> <li>- широкі можливості для аналітики та обробки даних;</li> <li>- велика надійність та підтримка корпоративних середовищ;</li> <li>- інтеграція з іншими продуктами Microsoft;</li> </ul> | <p>Недоліки:</p> <ul style="list-style-type: none"> <li>- ліцензійна вартість може бути високою для малих та середніх підприємств;</li> <li>- може бути складним для налаштування та адміністрування;</li> </ul>      |

Аналізуючи вимоги до системи та зважаючи на обрану мікросервісну архітектуру буде використана Microsoft SQL Server.

Обрані технології взаємодіють між собою таким чином:

- розробка додатку здійснюється у середовищі Visual Studio, де використовуються мови програмування Ruby on Rails, HTML, CSS та JavaScript;

- Ruby on Rails використовується для створення серверної частини додатку, реалізації логіки, обробки запитів та доступу до бази даних;
- HTML використовується для створення структури вебсторінок, включаючи розмітку та розташування елементів;
- CSS використовується для оформлення сторінок, задавання зовнішнього вигляду, кольорів, шрифтів та розмірів елементів;
- JavaScript використовується для створення динамічних елементів, валідації форм, взаємодії з користувачем та сервером.

За допомогою цих технологій розроблений вебдодаток може ефективно взаємодіяти з користувачем, обробляти його запити, зберігати та відображати дані, а також забезпечувати зручний та естетичний інтерфейс користувача.

## **3 РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ СИСТЕМИ**

### **3.1 Встановлення Ruby on Rails**

Один з факторів, що сприяє зручності роботи з Ruby, полягає у його синтаксисі. Код у цій мові є інтуїтивним, тому для написання, розуміння та підтримки ним потрібно значно менше зусиль.

Моделі у Ruby on Rails є основним компонентом, який використовується для організації та роботи з даними у вебдодатку. Моделі виконують кілька важливих функцій у вебдодатку. По-перше, вони забезпечують доступ до бази даних та дозволяють зберігати, зчитувати, редагувати та видаляти дані.

Перш за все, необхідно налаштувати оточення [15]. Найпростіший спосіб інсталювати Ruby на комп'ютер з операційною системою Windows - це завантажити Rubyinstaller. Після завантаження інсталятора потрібно активувати його, натиснувши на rubyinstaller-2.4.1-2-x64.exe. Оберіть перемикач "Я приймаю ліцензію" і натисніть кнопку "Далі", що приведе вас до наступних кроків (рис. 3.1).

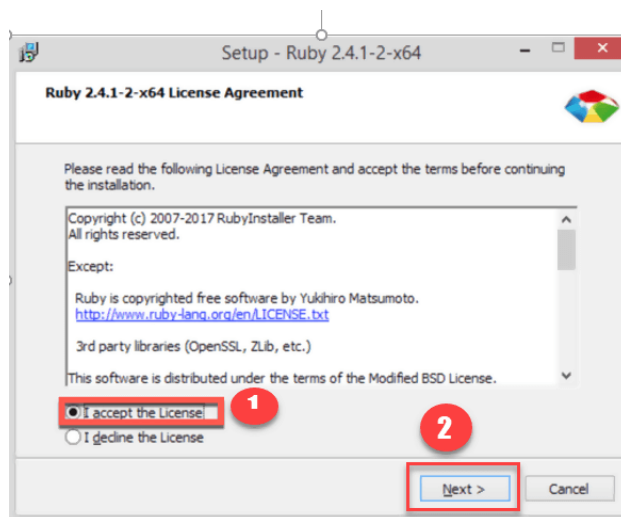


Рисунок 3.1 – Вікно інсталяції Ruby

Після завершення процесу інсталяції виконаємо наступні дії для переходження на новий етап установки (рис.3.2).

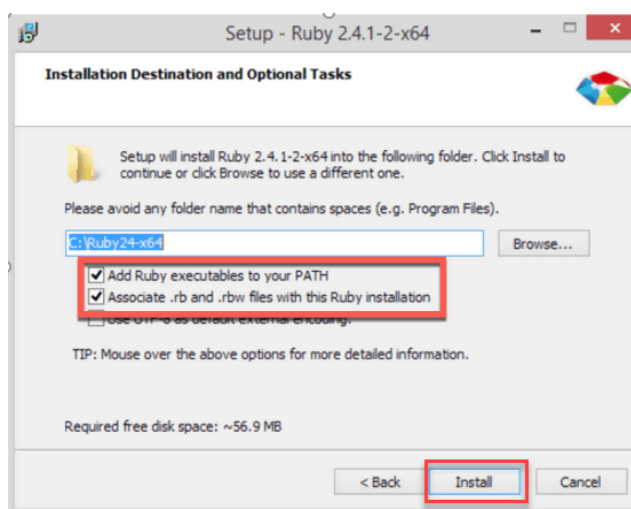


Рисунок 3.2 - Вікно командного рядка інсталяції Ruby

Після чого з'явиться запит на встановлення MSYS2, платформи для створення, яка містить менеджер пакетів для легкого встановлення пакетів Windows (рис.3.3).

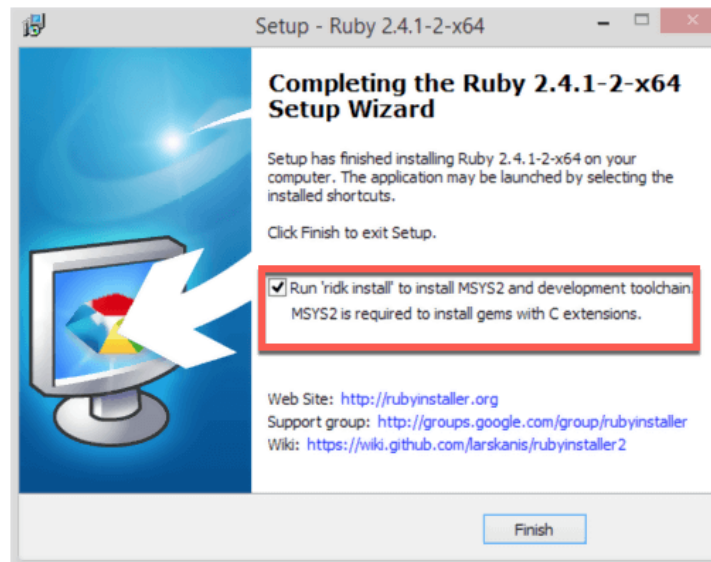


Рисунок 3.3 - Запрос на встановлення MSYS2

Інсталиємо всі компоненти, оскільки всі вони потрібні для роботи в середовищі розробки Ruby on Rails Windows , в ході успішного встановлення отримаємо наступний вигляд на екрані (рис.3.4).

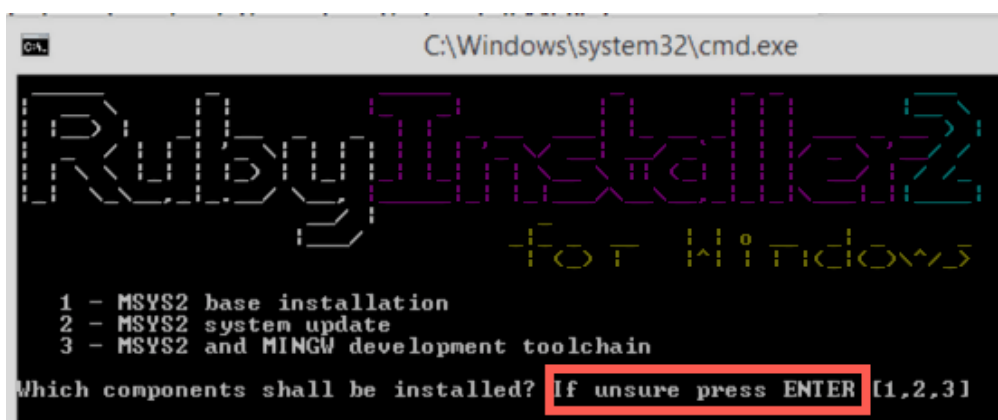


Рисунок 3.4 – Встановлення компонента MSYS2

### 3.1.1 Налаштування Rails

Наступним кроком буде встановлення Rails. Для цього потрібно встановити Rails на вже встановлений Ruby за допомогою пакета RailsInstaller. Щоб отримати Rails у вашій системі, запустіть наступну команду у командному рядку: 'gem install rails.' Для правильної роботи бібліотек, від яких залежить Rails, ми використовуємо обрану мову програмування JavaScript.

Для створення додатка зі списком завдань необхідно виконати наступну команду у терміналі: 'rails new todo\_app', щоб створити програму [16]. Ця команда створює каталог під назвою 'todo\_app' у поточному каталозі з базовою структурою папок (рис.3.5).

Після заміни каталогу 'todo\_app' на 'todo\_app' та запуску сервера Rails за допомогою команди 'rails server', для запуску програми необхідно відкрити "localhost:3000" у браузері. Якщо все пройшло успішно, ми побачимо наступне зображення (рис. 3.6).

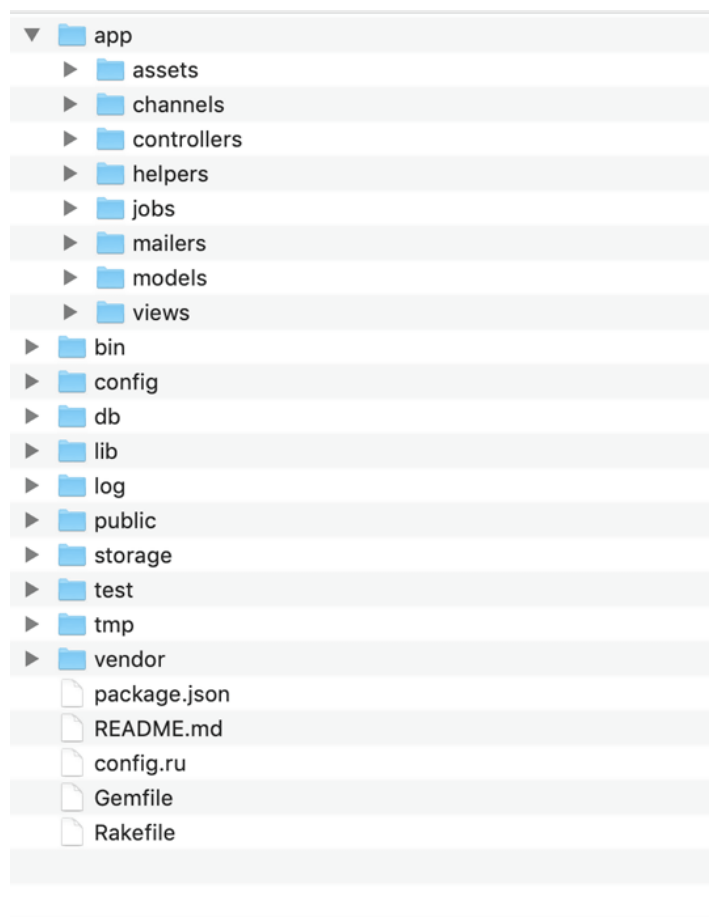


Рисунок 3.5 - Структура папок вебпрограми Rails

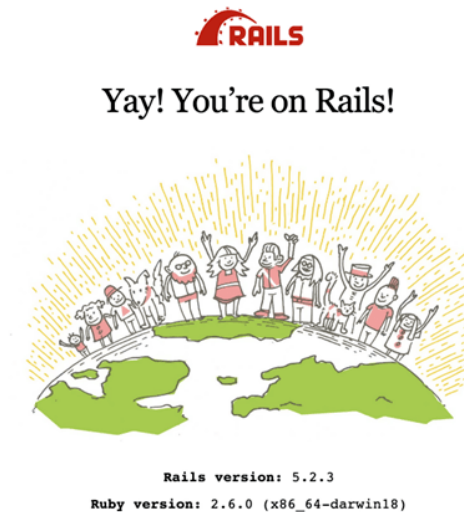


Рисунок 3.6 - Домашня сторінка програми за замовчуванням

### 3.1.2 Генерація команди Rails

Команда Rails generate використовує шаблони для створення різноманітних корисних компонентів у програмі, що спрощує написання коду, необхідного для функціонування вебпрограми.

На рисунках 3.7 та 3.8 наведено приклад контролерів для реєстрації і входу користувача в систему.

```
class AccountsController < ApplicationController
  before_action :set_account, only: [:show, :edit, :update, :destroy] def new
    @account = Account.new end

  def create
    @account = Account.new(account_params) @account.user = current_user

    if @account.save
      redirect_to accounts_path, notice: "Account was succesfully created."
    else
      render :new, status: :unprocessable_entity end
    end
  end
end
```

Рисунок 3.7 – Функції для реєстрації користувача

```
def account_params
  params.require(:account).permit(:name, :account_type,
    :account_number, :bank_name, :owner_name, :current_balance) end

def set_account
  @account = Account.find(params[:id]) End
```

Рисунок 3.8 – Функції системи для входу в систему

Для оновлення даних в системі створюємо наступні коди: для оновлення даних акаунту (рисунок 3.9) для видалення акаунту (рисунок 3.10).

```
class AccountsController < ApplicationController
  before_action :set_account, only: [:show, :edit, :update,
    :destroy]

  def index
    @accounts = current_user.accounts End

  def update
    if @account.update(account_params)
      redirect_to accounts_path, notice: "Account was successfully updated."
    else
      render :edit, status: :unprocessable_entity end
    end
  end
```

Рисунок 3.9 – Функція для оновлення даних акаунту

```
def destroy
  @account.destroy
  redirect_to accounts_path, notice: "Account was deleted successfully."
end
```

Рисунок 3.10 – Функція для видалення акаунта

### 3.1.3 Маршрутизації Rails

Система маршрутизації Rails, так званий "rails router", обробляє всі вхідні запити до вебпрограми, перевіряючи URL-адресу вхідних запитів та порівнюючи кожен запит із дією контролера, відповідальною за його обробку. Це здійснюється

за допомогою спеціального синтаксису, що вказаний у файлі маршрутів (config/routes.rb).

Файл маршрутів дозволяє контролювати кожен аспект URL-адреси програми. Rails за замовчуванням використовує дизайн RESTful на основі архітектурного стилю REST, який забезпечує відповідність між дієсловами HTTP та запитам (URL-адресами) до дій контролера [17].

Перш ніж запускати команду 'rails db:migrate', переконайтеся, що ви все ще перебуваєте в корені програми (каталог todo\_app). Для запуску сервера використовуйте команду 'rails server'. Після цього в командному рядку введіть `http://localhost:3000/todo_lists` (рис. 3.11).

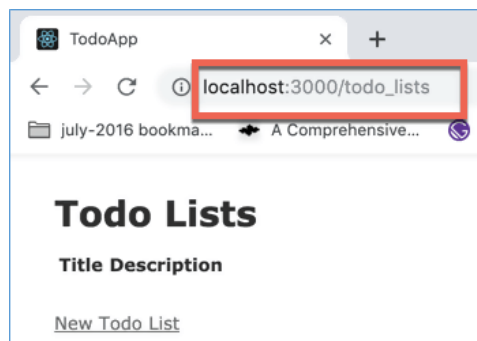


Рисунок 3.11 - Вебсторінка

Додаймо список завдань, кладнувши «Новий список справ» на сторінці (рис.3.12).

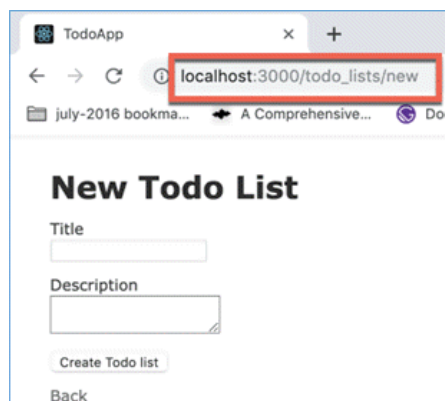


Рисунок 3.12— Додавання «Новий список справ»

В результаті цього можна побачити, що URL-адреса змінилася на `http://localhost:3000/todo_lists/new`, що є сторінкою для створення нового списку завдань, і вона керується новим методом або дією `TodoListsController`.

Після введення назви та опису списку завдань у форму та натискання кнопки "Створити список завдань", URL-адреса має змінитися на `http://localhost:3000/todo_lists/1` (рис. 3.13), як показано на прикладі нижче у коді Ruby on Rails.

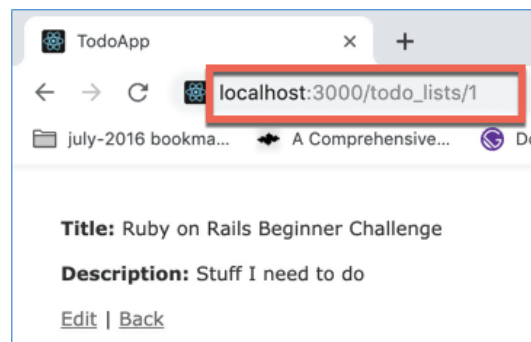


Рисунок 3.13 - Створення нового форми завдання

Ця сторінка відображає список завдань та керується методом або дією `TodoListsController`. Повертаючись до `http://localhost:3000/todo_lists/`, ви побачите вікно з новим доданим списком справ (рис. 3.14).

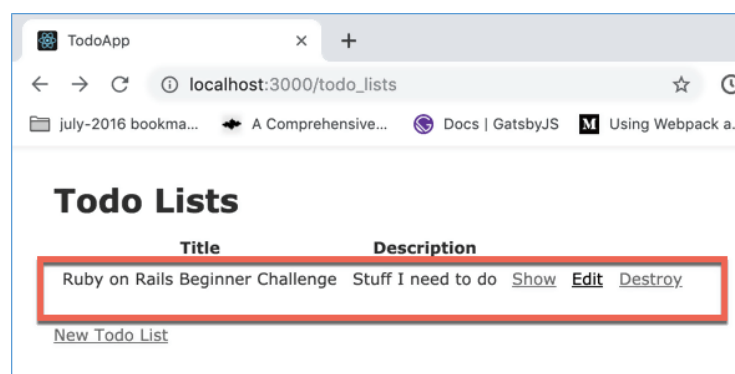


Рисунок 3.14 - Вікно з доданим новим списком справ

Rails вдалося відобразити різні запити (URL-адреси) на відповідну дію `TodoListsController`, використовуючи визначення маршруту в `config/routes.rb`.

## 3.2 Розробка бази даних

Бази даних представляють собою набір інформації, який зазвичай організований у вигляді таблиць. Дані у базах даних зберігаються таким чином, щоб їх можна було легко отримати та використовувати за допомогою комп'ютерної програми.

На початку процесу роботи над базою даних створюються таблиці та визначаються зв'язки між ними. Кожна таблиця має свої поля, які відображають відповідну інформацію. Наприклад, для даних про користувачів це може бути ім'я, прізвище, вік, електронна пошта та хеш пароля.

В результаті створення бази даних отримуємо структуровану систему для зберігання та управління даними. Це дозволяє ефективно працювати з інформацією, забезпечувати цілісність даних та здійснювати різноманітні операції, такі як додавання, оновлення та видалення даних. Для проєктування бази даних було використано MS SQL Server 2005.

Microsoft SQL Server - це система управління базами даних, яка розробляється корпорацією Microsoft. Як сервер даних, вона виконує основну функцію збереження та надання даних у відповідь на запити інших програм, які можуть працювати як на тому ж сервері, так і в мережі.

### 3.2.1 Налаштування SQL Server

Для встановлення Microsoft SQL Server на комп'ютер необхідно покроково виконати стандартну процедуру щодо встановлення програми. Для цього потрібно провести наступні дії:

- завантажити інсталяційний файл SQL Server;
- запустити файл інсталятора;
- обрати типу інсталяції - базова установка або налаштування розширених параметрів або розгортання самої конкретної служби;
- вибір складових для встановлення – треба обрати компоненти SQL

Server, це може включати движок баз даних, інструменти управління, засоби розробки та інші компоненти;

- налаштувати інстанс - вказати назву інстансу SQL Server, порти, налаштування безпеки та інші;

- налаштувати службу облікових записів - вказати облікові записи користувачів, які будуть використовуватися для запуску служб;

- після налаштування всіх параметрів запускаємо процес інсталяції при цьому інсталятор розпочне копіювання файлів, налаштування компонентів та встановлення SQL Server на комп'ютер;

Після завершення процесу інстації та перезапуску комп'ютера можна почати перевірку підключення до серверу та стану бази даних.

### **3.2.2 Структура бази даних**

Було розроблено структуру бази даних (рис.3.15), написано код до таблиць основних компонентів User, Wallet і Category, які представлені на рисунках 3.16 - 3.18. Повний код приведено у додатку А.

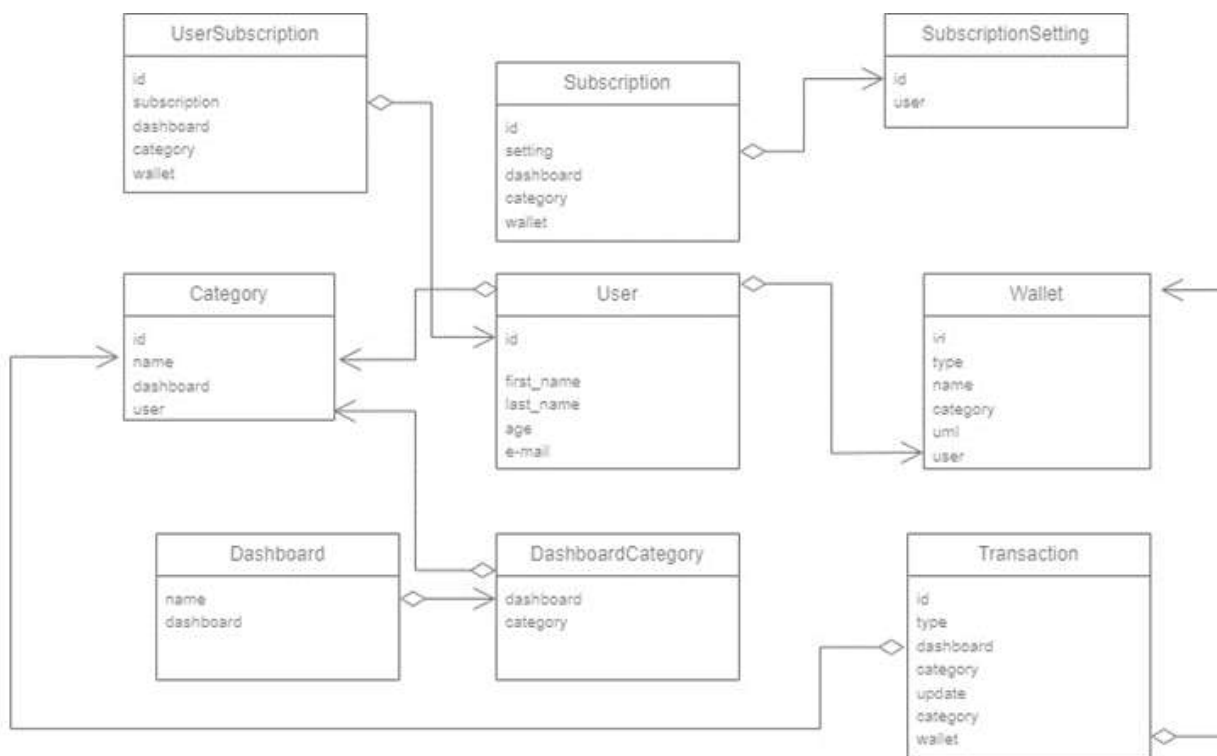


Рисунок 3.15– Структура бази даних

```

CREATE TABLE [User] (
    Id INT PRIMARY KEY,
    FirstName NVARCHAR(255), LastName NVARCHAR(255), Age INT,
    Email NVARCHAR(255),
    PasswordHash NVARCHAR(255)
);
  
```

Рисунок 3.16 – Створення таблиці User

```

CREATE TABLE Wallet (
    Id INT PRIMARY KEY, Type NVARCHAR(255), Name NVARCHAR(255),
    Amount DECIMAL(18, 2),
    Currency NVARCHAR(255), IconURL NVARCHAR(255), UserID INT,
    FOREIGN KEY (UserID) REFERENCES [User] (Id)
);
  
```

Рисунок 3.17 – Створення таблиці Wallet

```
CREATE TABLE Category (
    Id INT PRIMARY KEY, Name NVARCHAR(255),
    Description NVARCHAR(MAX), UserID INT,
    FOREIGN KEY (UserID) REFERENCES [User] (Id)
);
```

Рисунок 3.18 – Створення таблиці Category

Ці таблиці формують базу даних, яка призначена для збереження та керування інформацією про користувачів, гаманці та категорії. Використовуючи ці таблиці, можна створювати взаємозв'язки, виконувати операції з даними та забезпечувати функціональність вебзастосунку.

### 3.2.3 Побудова зв'язків між таблицями

При проектуванні бази даних важливо належним чином налаштувати зв'язки між її складовими для забезпечення оптимальної структури та взаємодії даних. Зв'язки дозволяють встановлювати взаємозалежності та взаємодію між різними таблицями, що сприяє уникненню повторення інформації та забезпечує цілісність та стабільність даних. Більш детально опишемо зв'язки між таблицями.

У таблиці "User" поле "Id" є основним ключем, тоді як у таблиці "Wallet" поле "UserID" є зовнішнім ключем, що посиляється на "Id" таблиці "User". Це формує зв'язок "один до багатьох" між користувачем і його гаманцями, оскільки один користувач може мати кілька гаманців, згідно з встановленим відношенням "один до багатьох".

У таблиці "Category" поле "UserID" використовується як зовнішній ключ, що посиляється на "Id" у таблиці "User". Це формує зв'язок "багато до одного" між категорією і користувачем, оскільки один користувач може мати багато категорій.

У таблиці "User" поле "Id" є основним ключем, а у таблиці "Wallet" поле "UserID" використовується як зовнішній ключ, що посиляється на поле "Id" у таблиці "User". Це встановлює зв'язок "один до багатьох" між користувачем і його гаманцями, дозволяючи кожному користувачеві мати декілька гаманців.

У таблиці "DashboardCategory" поле "DashboardID" є зовнішнім ключем, який посиляється на "Id" у таблиці "Dashboard". Це утворює зв'язок "багато до

одного" між категорією та панеллю управління. Таким чином, одній панелі управління може належати багато категорій, оскільки встановлене відношення "багато до одного" між категорією та панеллю управління.

У таблиці "Transaction" поле "WalletID" використовується як зовнішній ключ, який посилається на "Id" у таблиці "Wallet". Це створює зв'язок "багато до одного" між транзакцією і гаманцем, оскільки один гаманець може мати багато транзакцій.

У таблиці "UserSubscriptions" поля "UserID" та "SubscriptionID" використовуються як зовнішні ключі, що посилаються відповідно на "Id" у таблицях "User" та "Subscription". Це створює зв'язок "багато до багатьох" між користувачами та їх підписками. Усі ці взаємозв'язки сприяють створенню міцної та продуктивної бази даних, що дозволяє зручно та ефективно зберігати, організовувати та використовувати інформацію.

У таблиці "Subscription" поле "SettingsID" використовується як зовнішній ключ, що посилається на "Id" у таблиці "SubscriptionSettings". Це формує зв'язок "один до одного" між підпискою та її налаштуваннями. Кожна підписка може мати лише одні налаштування, оскільки встановлено відношення "один до одного" між підпискою та налаштуваннями.

### **3.3 Інтерфейс користувача**

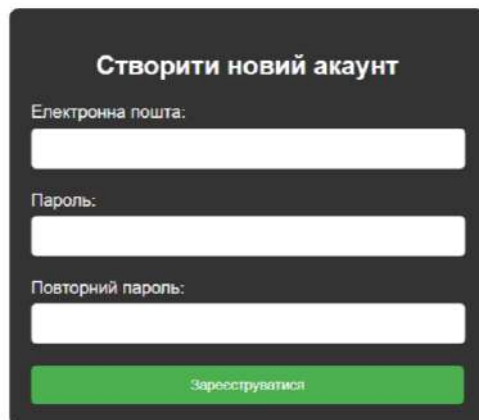
Для створення користувацької панелі були використані технології Ruby on Rails, HTML та CSS. Ця панель містить такі функціональні елементи:

- система авторизації для користувачів;
- панель для відображення статистичних даних;
- панель керування категоріями;
- панель гаманців;
- історія транзакцій;

- зміна теми інтерфейсу.

Код, який використовується для реалізації цих функціональних елементів, наведено в додатку А.

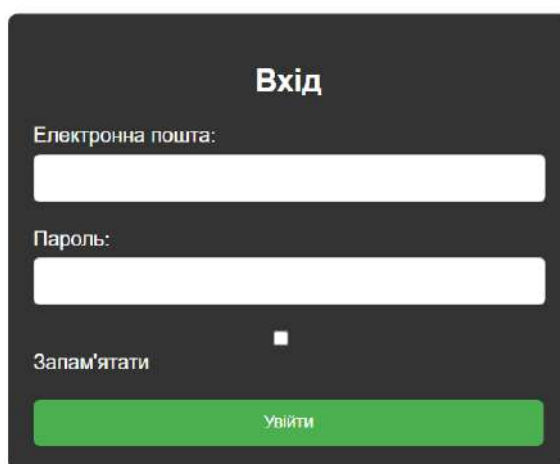
Для нових користувачів, які ще не зареєстровані в системі, сторінка авторизації може містити посилання на сторінку реєстрації. Це дозволить їм створити новий обліковий запис та отримати доступ до системи (рис.3.19).



The image shows a dark-themed registration form titled "Створити новий акаунт" (Create new account). It contains three input fields: "Електронна пошта:" (Email), "Пароль:" (Password), and "Повторний пароль:" (Repeat password). Below the fields is a green button labeled "Зареєструватися" (Register).

Рисунок 3.19 - Сторінка реєстрації користувача

Для користувачів, які вже мають обліковий запис, наявне меню авторизації при вході до системи (рис. 3.20, 3.21).



The image shows a dark-themed login form titled "Вхід" (Login). It contains two input fields: "Електронна пошта:" (Email) and "Пароль:" (Password). Below the password field is a small square checkbox labeled "Запам'ятати" (Remember me). At the bottom is a green button labeled "Увійти" (Login).

Рисунок 3.20 - Сторінка авторизації користувача

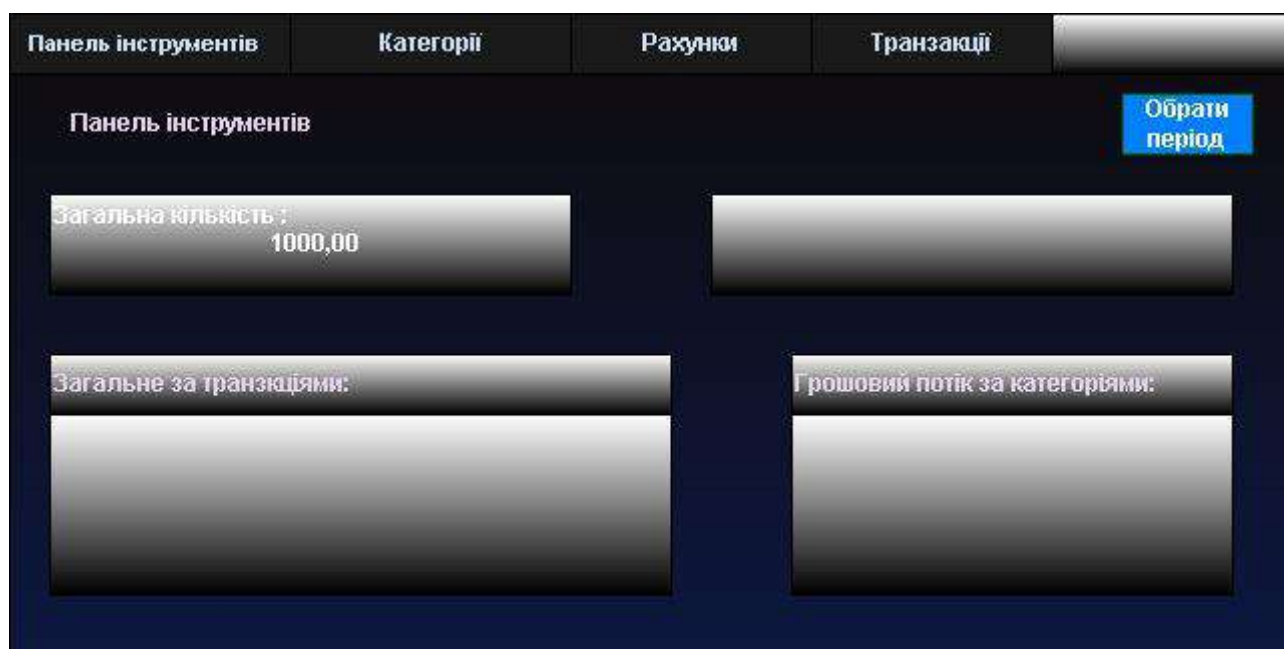


Рисунок 3.21 – Головна сторінка застосунку

Для зручності користувачів головне меню розташоване у верхній частині сторінки і залишається незмінним навіть при переході на інші сторінки, що робить його статичним. На сторінці категорій доступний список, де користувач може переглянути всі створені ним категорії, а також відредагувати чи видалити їх (рис.3.22).



Рисунок 3.22 – Сторінка керування категоріями

У вкладці "Рахунки" користувач може переглядати всі свої рахунки. Крім цього, він може внести додаткову інформацію, таку як назва банку, номер картки, власник та тип кошелька (рис.3.23).



Рисунок 3.23 – Перелік гаманців

На вкладці "Історія транзакцій" користувач може переглядати всі свої витрати та доходи, які також будуть відображені в діаграмах на головній сторінці застосунку.

Під час створення інтерфейсу були узяті до уваги потреби та очікування цільової аудиторії. Використані зрозумілі та логічні елементи керування, а також розроблений зручний дизайн з естетичним виглядом.

У процесі цієї роботи над комп'ютерною системою для домашньої бухгалтерії було зроблено значну роботу над розробкою серверної частини, бази даних та дизайну інтерфейсу користувача. Була розроблена структура бази даних, включаючи необхідні таблиці та зв'язки між ними, що дозволяє ефективно зберігати та організовувати дані.

## 4 ТЕСТУВАННЯ РОБОТИ СИСТЕМИ

Тестування є важливим етапом у процесі розробки програмного забезпечення, який гарантує його належне функціонування та відсутність помилок перед впровадженням у експлуатацію. У даному розділі перевірено коректність роботи функцій створення нових категорій та внесення нових транзакцій. Щоб додати нову категорію через функціональне меню, користувачу необхідно лише ввести її назву та підтвердити збереження (рис.4.1).

**Додати нову категорію**

Назва категорії:

Додати

Рисунок 4.1 – Приклад додавання нової категорії

Для перегляду списку, треба перейти до меню категорій. Якщо щось стало неактуальним, можна видалити або відредагувати цю інформацію.

Для зручного відстеження транзакцій було додано декілька пунктів, а саме: дату, опис, суму та категорію. Таким чином, у списку буде відображена вся необхідна інформація (рис.4.2) .

Dashboard Categories Accounts Transactions

Transactions

Import transactions Add new transaction

Expand table

| # | DATE  | DESCRIPTION | AMOUNT   | TYPE  | CATEGORY   | ACCOUNT | NOTES           | ACTIONS                                     |
|---|-------|-------------|----------|-------|------------|---------|-----------------|---------------------------------------------|
| 1 | 06-06 | coffee      | \$150.00 | Debit | Restaurant | Yulia   | coffee and coke | <a href="#">Edit</a> <a href="#">Delete</a> |
| 2 | 05-29 | vegetables  | \$100.00 | Debit | Food       | Yulia   | vegetables      | <a href="#">Edit</a> <a href="#">Delete</a> |

Рисунок 4.2 – Список транзакцій

На наступному рисунку можна побачити обрану нами тему та аналітичний обзор витрат та доходів (рис.4.3).

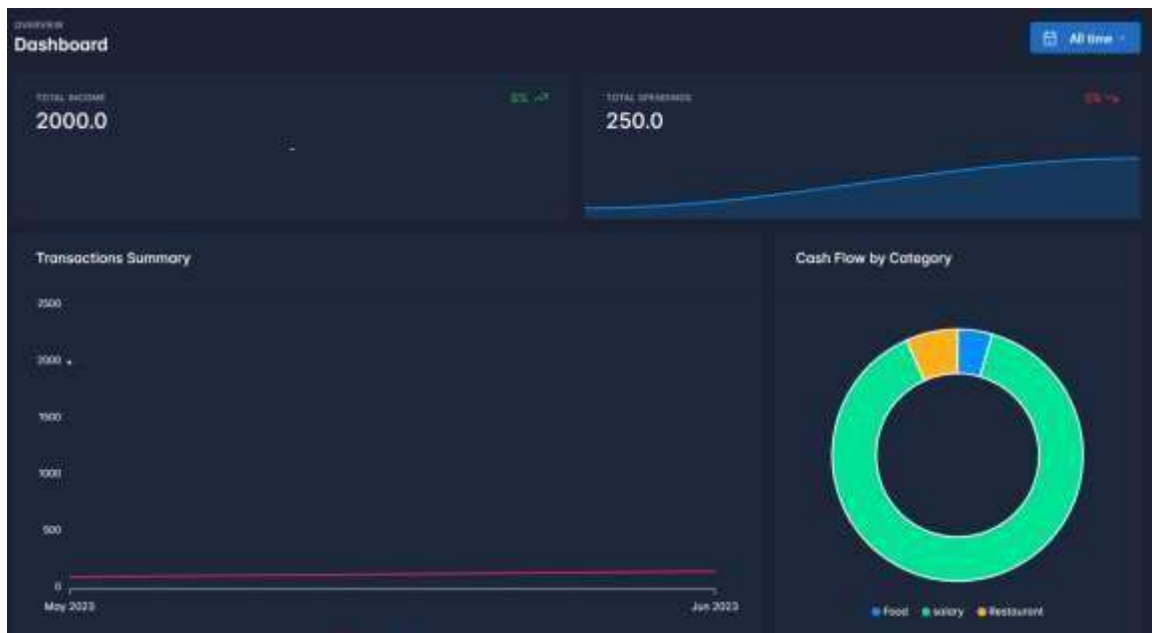


Рисунок 4.3 – Аналітика витрат та доходів

Під час тестування було проведено перевірку правильності функціональності додатку. Користувач зміг успішно взаємодіяти з інтерфейсом, додавати та редагувати дані, а також виконувати необхідні дії.

Узагальнено, тестування функціонування системи підтвердило, що вебдодаток працює відповідно до очікувань, надаючи користувачам зручний та функціональний інтерфейс для управління своїми обліковими записами, категоріями та транзакціями.

## ВИСНОВКИ

У ході створення комп'ютерної системи для ведення домашнього бюджету була успішно завершена розробка серверної частини, яка відповідає за комунікацію між клієнтськими додатками та сховищем даних, а також за перевірку автентичності користувачів.

Була спроектована архітектура бази даних, яка містить всі потрібні таблиці та зв'язки між ними, що забезпечує ефективне збереження та структурування інформації. Крім того, був розроблений простий та зрозумілий інтерфейс користувача, який полегшує управління профілями, категоріями та фінансовими операціями.

Після ретельного тестування системи, яке довело її стабільність та правильне функціонування, був отриманий готовий до використання вебдодаток, який допомагає користувачам відстежувати свої витрати та керувати грошовими потоками, надаючи зручний та надійний інструмент для обліку та аналізу фінансової інформації.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Вебдодаток: що це таке, типи, переваги, принцип роботи [Електронний ресурс] – Режим доступу: <https://megasite.ua/ua/veb-prilogenie-chto-eto-takoe-tipi-preimushchestva-printsip-raboti> (дата звернення 10.04.2024).
2. Особливості web-додатків [Електронний ресурс] – Режим доступу: <https://sites.znu.edu.ua/webprog/lect/1191.ukr.html> (дата звернення 23.03.2024).
3. Мобільні застосунки для планування бюджету та контролю витрат [Електронний ресурс] – Режим доступу: <https://www.epravda.com.ua/publications/2023/02/22/697326/> (дата звернення 31.04.2024).
4. 10 фінансових додатків для домашньої бухгалтерії [Електронний ресурс] – Режим доступу: <https://new.minfin.com.ua/ua/ideabank/10-prilozhenij-kotorye-pomogut-razbogatet> (дата звернення 17.04.2024).
5. Мобільні застосунки для планування бюджету та контролю витрат [Електронний ресурс] – Режим доступу: <https://www.epravda.com.ua/publications/2023/02/22/697326/> (дата звернення 21.03.2024).
6. Що таке функціональні вимоги: приклади, визначення, повний посібник [Електронний ресурс] – Режим доступу: <https://visuresolutions.com/uk> (дата звернення 24.04.2024).
7. Як створити вебдодаток: типи, переваги, принцип роботи [Електронний ресурс] – Режим доступу: <https://wezom.com.ua/ua/blog/kak-sozdat-veb-prilozhenie> (дата звернення 18.04.2024).
8. Архітектура веб додатків [Електронний ресурс] – Режим доступу: <https://medium.com/@IvanZmerzlyi/%D0%B0%D1%80%D1%85%D1%96%> (дата звернення 12.04.2024).
9. Як обрати архітектуру для веб додатку [Електронний ресурс] – Режим доступу: <https://blog.ithillel.ua/articles/web-application-architecture> (дата звернення

7.05.2024).

10. Популярні середовища розробки мов програмування [Електронний ресурс] – Режим доступу: <https://ua5.org/osnprog/1600-populyarni-seredovyshha-rozrobky-mov-programuvannya.html> (дата звернення 27.04.2024).

11. Що таке html? [Електронний ресурс] – Режим доступу: [https://css.in.ua/article/shcho-take-css\\_3](https://css.in.ua/article/shcho-take-css_3) (дата звернення 01.05.2024).

12. Що таке фреймворк: пояснюємо простими словами [Електронний ресурс] – Режим доступу: <https://brainlab.com.ua/uk/blog-uk/shho-take-frejmvork-royasnyuemo-prostymy-slovamy> (дата звернення 25.04.2024).

13. Інтегроване середовище розробки (IDE) [Електронний ресурс] – Режим доступу: <https://w3schoolsua.github.io/hyperskill/ide.html#gsc.tab=0> (дата звернення 18.04.2024).

14. Бази даних – основи, системи керування та мови їх обробки [Електронний ресурс] – Режим доступу: <https://ua5.org/database> (дата звернення 27.04.2024).

15. Підручник Ruby on Rails для початківців [Електронний ресурс] – Режим доступу: <https://www.guru99.com/uk/ruby-on-rails-tutorial.html> (дата звернення 29.04.2024).

16. Путівник галактикою Ruby [Електронний ресурс] – Режим доступу: <https://dou.ua/lenta/articles/ruby-guide-1/> (дата звернення 03.05.2024).

17. Установите Ruby On Rails в Windows 11 [Електронний ресурс] – Режим доступу: <https://gorails.com/setup/windows/11> (дата звернення 30.04.2024).

## ДОДАТОК А

### Код таблиць основних компонентів user, wallet і category

#### Лістинг А.1 – Код реалізації бази даних

```
class AccountsController < ApplicationController
  before_action :set_account, only: [:show, :edit, :update,
:destroy]

  def index
    @accounts =
current_user.accounts
end

  def
sho
w
end

  def new
    @account =
Account.new
end

  def create
    @account =
Account.new(account_params)
    @account.user = current_user

    if @account.save
      redirect_to accounts_path, notice: "Account was
successfully created."
    else
      render :new, status:
:unprocessable_entity
end
end

  def
edi
t
end

  def update
    if @account.update(account_params)
      redirect_to accounts_path, notice: "Account was
successfully updated."
    else
      render :edit, status:
:unprocessable_entity
end
end
```

```

    def destroy
      @account.destroy
      redirect_to accounts_path, notice: "Account was deleted
successfully."
    end

    private
    def account_params
      params.require(:account).permit(:name, :account_type,
:account_number, :bank_name, :owner_name, :current_balance)
    end

    def set_account
      @account =
      Account.find(params[:id]) end
    end

class CategoriesController < ApplicationController
  before_action :set_category, only: [:show, :edit, :update,
:destroy]

  def index
    @categories =
    current_user.categories.page(params[:page]) end

  def
  sho
  w
  end

  def new
    @category =
    Category.new end

  def create
    @category =
    Category.new(category_params)
    @category.user = current_user
    if @category.save
      redirect_to categories_path, notice: "Category was
succesfullycreated."
    else
      render :new, status:
      :unprocessable_entity end
    end

  def
  edi
  t
  end

  def update

```

```

        if @category.update(category_params)
          redirect_to categories_path, notice: "Category was
succesfullyupdated."
        else
          render :edit, status:
          :unprocessable_entityend
        end

        def destroy
          @category.destroy
          redirect_to categories_path, notice: "Category was
successfullydestroyed."
        end private

        def category_params
          params.require(:category).permit(:name)
        end

        def set_category
          @category =
          Category.find(params[:id])end
        end

class DashboardController <
  ApplicationControllerdef show
    @debits =
current_user.transactions.transaction_type_debit.group_by_month
(:date, format: "%b %Y").sum(:amount)
    @credits =
current_user.transactions.transaction_type_credit.group_by_mont
h(:date, format: "%b %Y").sum(:amount)

    @labels = @debits.keys
    @series = [@debits.values, @credits.values]

    category_chart_data =
current_user.transactions.group(:category).sum(:amount)
    @category_chart_labels =
category_chart_data.keys.map(&:name)
    @category_chart_series = category_chart_data.values

    @transactions =
current_user.transactions.first(10)end
  end

class TransactionsController < ApplicationController
  before_action :set_transaction, only: [:show, :edit,
:update,
:destroy]

  def index
    @transactions =

```

```

current_user.transactions.ordered.page(params[:page])end

def show
  @transaction = Transaction.new(params[:transaction_id])
end

def new
  @transaction = Transaction.new
end

def create
  @transaction = Transaction.new(transaction_params)
  @transaction.category = Category.find_or_create_by(name:
'Uncategorized', user: current_user) if
params[:transaction][:category].nil?
  if @transaction.save
    Transaction.update_account_balance(@transaction)
    redirect_to transactions_path, notice: "Transaction was
succesfully created."
  else
    render :new, status:
:unprocessable_entityend
end

def edit
  @transaction = Transaction.find(params[:transaction_id])
end

def update
  if @transaction.update(transaction_params)
    redirect_to transactions_path, notice: "Transaction was
succesfully updated."
  else
    render :edit, status:
:unprocessable_entityend
end

def destroy
  @transaction.destroy
  redirect_to transactions_path, status::see_other, alert:
"Transaction was deleted successfully."
end

def read
  @transaction = Transaction.new(params[:transaction_id])
end

def import
  file =

```

```

      params[:transaction][:file] if
      file.content_type ==
      'text/csv'
      ImportTransactionsService.new.call(file,
      params[:transaction][:account_id])
      redirect_to transactions_path, notice: 'Transactions
      imported successfully.'
    else
      redirect_to new_transaction_path, notice: 'Only CSV files
      accepted.'
    end
  end
end

  private

  def transaction_params
    params.require(:transaction).permit(:date, :description, :transaction_
    type, :amount, :notes, :account_id, :category_id, : false)
  end

  def set_transaction
    @transaction = Transaction.find(params[:id])
  end

```

## ДОДАТОК Б

### Реалізація функціональних елементів

#### Лістинг Б.1 – Код створення інтерфейсу

```

<div class="col-sm-6 col-lg-6">
  <div class="card d-flex flex-column">
    <div class="row row-0 flex-fill">
      <div class="col-md-3">
        <%= image_tag "westpac-logo.png", class: "w-100 h-100" %>
      </div>
      <div class="col border-start">
        <div class="card-body text-center">
          <h3><%= account.name %></h3>
          <div class="row">
            <div class="col">
              <samp class="text-muted">Bank: <%=
account.bank_name.capitalize %></samp>
            </div>
            <div class="col">
              <samp class="text-muted">Type: <%=
account.account_type.capitalize %></samp>
            </div>
          </div>
          <div class="row">
            <div class="col">
              <samp class="text-muted">Owner: <%=
account.owner_name%></samp>
            </div>
          </div>
        </div>
      <div class="col">
        <samp class="text-muted">Number: <%=account.account_number
%></samp>
      </div>
    </div>
    <div class="row">
      <div class="col mt-3">
        <strong>Balance: $<%=
number_with_precision(account.current_balance,precision: 2)
%></strong>
      </div>
    </div>
  </div>
</div>
<div class="page-header d-print-none">
  <div class="container-xl">
    <div class="row g-2 align-items-center">

```

```

    <div class="col">
      <h3 class="page-title">Accounts</h3>
    </div>
    <div class="col-auto ms-auto">
<div class="btn-list"
  <%= link_to new_account_path, data: { turbo_frame:
    'modal'
  }, class: 'btn btn-primary' do %>
    <svg xmlns="http://www.w3.org/2000/svg"
class="icon" width="24" height="24" viewBox="0 0 24 24" stroke-
width="2" stroke="currentColor" fill="none" stroke-
linecap="round" stroke- linejoin="round"><path stroke="none"
d="M0 0h24v24H0z" fill="none"></path><path d="M12 5l0
14"></path><path d="M5 12l14 0"></path></svg>Add new account
    <% end %>
  </div>
</div>
</div>
</div>
</div>
<div class="page-body">
  <div class="container-xl">
    <div class="row row-deck row-cards">
      <%= render @accounts %>
    </div>
  </div>
</div>
</div>

<div class="page-header d-print-none">
  <div class="container-xl">
    <div class="row g-2 align-items-center">
      <div class="col">
        <h3 class="page-title">Categories</h3>
      </div>
      <div class="col-auto ms-auto">
        <div class="btn-list">
          <%= link_to new_category_path, data: { turbo_frame:
            'modal' }, class: 'btn btn-primary' do %>
            <svg xmlns="http://www.w3.org/2000/svg"
class="icon" width="24" height="24" viewBox="0 0 24 24" stroke-
width="2" stroke="currentColor" fill="none" stroke-
linecap="round" stroke- linejoin="round"><path stroke="none"
d="M0 0h24v24H0z" fill="none"></path><path d="M12 5l0
14"></path><path d="M5 12l14 0"></path></svg>Add new category
          <% end %>
        </div>
      </div>
    </div>
  </div>
</div>
<div class="page-body">
  <div class="container-xl">

```

```

<div class="col-12">
  <div class="card">
    <div class="table-responsive">
      <table class="table card-table table-vcenter text-
nowrapdatatable">
        <thead>
          <tr>
            <th scope="col">Name</th>
            <th scope="col">Transactions</th>
            <th scope="col">Created at</th>
            <th scope="col">Updated at</th>
            <th scope="col">Actions</th>
          </tr>
        </thead>
        <tbody>
          <%= render @categories %>
        </tbody>
      </table>
    </div>
  </div>
</div>
<div class="container-xl my-3">
  <%= paginate @categories, theme: 'bootstrap-5',
pagination_class: "pagination-sm flex-wrap justify-content-
center"
%>
</div>
</div>

<tr>
  <td><%= category.name %></td>
  <td><%= category.transactions.count %></td>
  <td><%= category.created_at.to_date %></td>
  <td><%= category.updated_at.to_date %></td>
  <td>
    <%= link_to edit_category_path(category), class: 'text-
primary', data: { turbo_frame: 'modal' } do %><i class="bi bi-
pencil-square"></i><% end %>
    <%= link_to category_path(category), class: 'text-danger',
data:
{ turbo_method: :delete, turbo_confirm: 'Are you sure?' } do
%><i class="bi bi-trash"></i><% end %>
  </td>
</tr>

<!DOCTYPE html>
<html>
  <head>
    <title>My Cash Flow</title>
    <meta name="viewport" content="width=device-width,initial-
scale=1">
    <%= csrf_meta_tags %>

```

```

    <%= csp_meta_tag %>
    <%= stylesheet_link_tag "application", "data-turbo-
track":"reload" %>
    <link rel="preconnect" href="https://rsms.me/">
    <link rel="stylesheet" href="https://rsms.me/inter/inter.css">
    <script async
src="https://www.googletagmanager.com/gtag/js?id=G-
0SPWQM395B"></script>
    <%= javascript_include_tag 'analytics', async: true %>
    <%= javascript_include_tag "application", "data-turbo-track":
"reload", defer: true %>
  </head>
  <body class="theme-light" data-controller="theme">
    <div class="page">
      <%= render 'shared/navbar' %>
      <%= render 'shared/flashs' %>
      <div class="page-wrapper">
        <%= yield %>
        <div
          class="modal
          fade"
          tabindex="-1"
          data-controller="modal"
          data-action="turbo:frame-load->modal#open
turbo:submit-end->modal#close"
        >
          <div class="modal-dialog">
            <div class="modal-content">
              <%= turbo_frame_tag "modal" %>
            </div>
          </div>
        </div>
      </div>
      <%= render 'shared/footer' %>
    </div>
  </body>
</html>
<div class="page-header d-print-none">
  <div class="container-xl">
    <div class="row g-2 align-items-center">
      <div class="col">
        <h3 class="page-title">Transactions</h3>
      </div>
      <div class="col-auto ms-auto">
        <div class="btn-list">
          <%= link_to read_transactions_path, data: {
turbo_frame:'modal' }, class: 'btn btn-secondary' do %>
            <svg xmlns="http://www.w3.org/2000/svg" class="icon
icon-tabler icon-tabler-table-import" width="24" height="24"
viewBox="0 0 24 24" stroke-width="2" stroke="currentColor"
fill="none" stroke-linecap="round" stroke-linejoin="round">
              <path stroke="none" d="M0 0h24v24H0z"
fill="none"></path>

```

```

<path d="M4 13.5v-7.5a2 2 0 0 1 2 -2h12a2 2 0 0 1 2
2v12a2 2 0 0 1 -2 2h-6m-8 -10h16m-10 -6v11.5m-8 3.5h7m-3 -3l3
3l-3
3"></path>
    </svg>Import transactions
    <% end %>
    <%= link_to new_transaction_path, data: {
turbo_frame:'modal' }, class: 'btn btn-primary' do %>
        <svg xmlns="http://www.w3.org/2000/svg"
class="icon" width="24" height="24" viewBox="0 0 24 24" stroke-
width="2" stroke="currentColor" fill="none" stroke-
linecap="round" stroke-linejoin="round"><path stroke="none"
d="M0 0h24v24H0z" fill="none"></path><path d="M12 5l0
14"></path><path d="M5 12l14 0"></path></svg>Add new
transaction
    <% end %>
    </div>
  </div>
</div>
</div>
<div class="page-body">
  <div class="container-xl">
<div class="col-12">
  <div class="card">
    <div class="table-responsive" data-controller="cell-
expand">
      <button type="button" data-cell-expand-
target="button" data-action="click->cell-expand#toggle"
class="btn btn-ghost-primary btn-sm m-2">
        <svg xmlns="http://www.w3.org/2000/svg" class="icon
icon-tabler icon-tabler-plus" width="24" height="24" viewBox="0
0 24 24" stroke-width="2" stroke="currentColor" fill="none"
stroke-linecap="round" stroke-linejoin="round">
          <path stroke="none" d="M0 0h24v24H0z"
fill="none"></path>
          <path d="M12 5l0 14"></path>
          <path d="M5 12l14 0"></path>
        </svg>Expand table
      </button>
      <table class="table card-table table-vcenter table-
nowrap datatable table-collapse">
        <thead>
          <tr>
            <th scope="col">#</th>
            <th scope="col">Date</th>
            <th
              scope="co
              l">
              Descripti
              on
            </th>
            <th scope="col" class="text-end">Amount</th>

```

```

        <th scope="col">Type</th>
        <th scope="col">Category</th>
        <th scope="col">Account</th>
        <th scope="col">Notes</th>
        <th scope="col">Actions</th>
      </tr>
    </thead>
    <tbody>
      <%= render @transactions %>
    </tbody>
  </table>
</div>
</div>
</div>
</div>
<div class="container-xl my-3">
  <%= paginate @transactions, theme: 'bootstrap-5',
pagination_class: "pagination-sm flex-wrap justify-content-
center"
%>
</div>
</div>

<div class="modal-header">
  <%= turbo_frame_tag 'modal' do %>
    <h2 class="modal-header">New transaction</h2>
    <button type="button" class="btn-close" data-bs-
dismiss="modal"aria-label="Close"></button>
    <div class="modal-body">
      <%= render 'form', transaction: @transaction %>
    </div>
  <% end %>
</div>

```



Рисунок 1 - Додаток Money Manager Expense & Budget

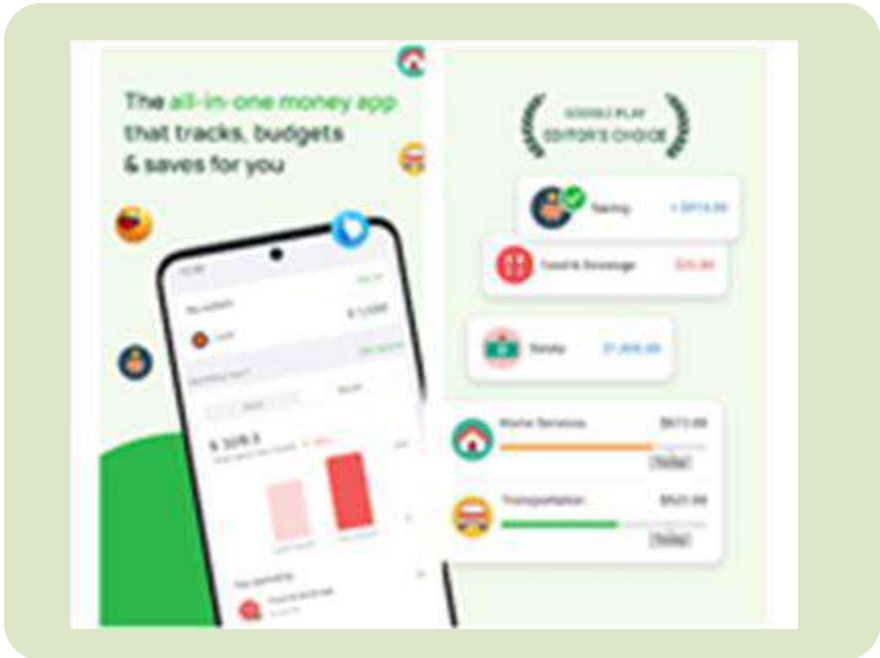


Рисунок 3 - Додаток Money Lover

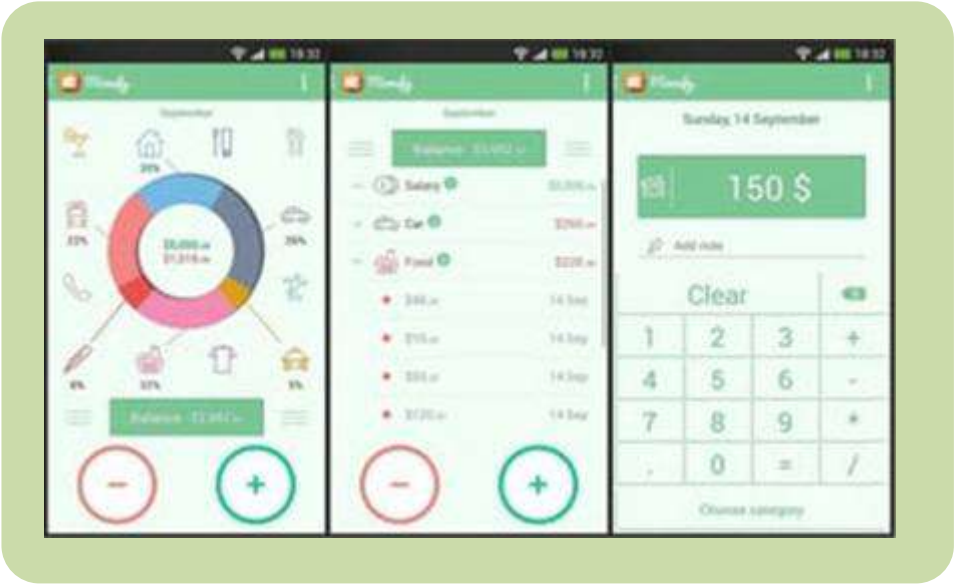


Рисунок 2 - Додаток Monefy

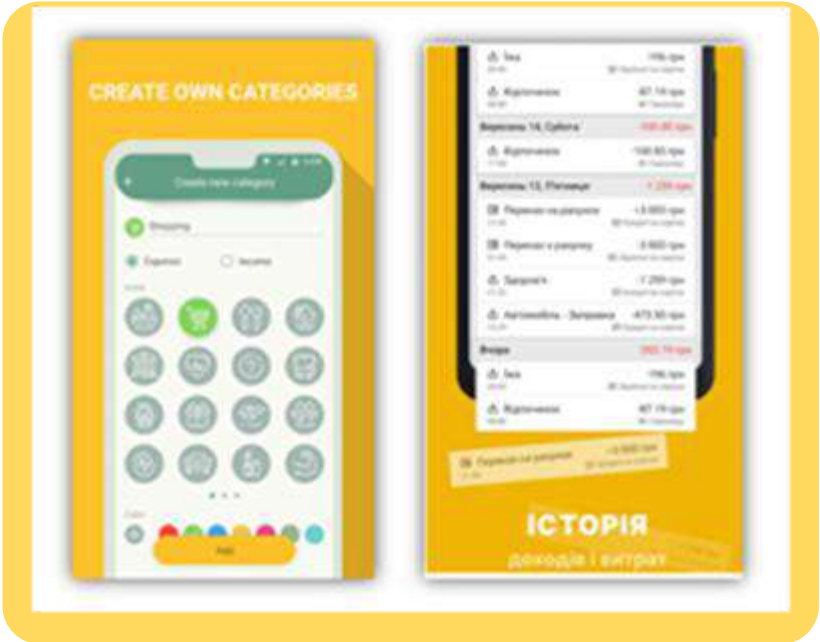


Рисунок 4 - Додаток Budget

|           |      |                 |                                                                                       |      |                                                         |                                          |  |  |           |  |         |  |
|-----------|------|-----------------|---------------------------------------------------------------------------------------|------|---------------------------------------------------------|------------------------------------------|--|--|-----------|--|---------|--|
|           |      |                 |                                                                                       |      | 13.02070849.00008 ПЛ1                                   |                                          |  |  |           |  |         |  |
|           |      |                 |                                                                                       |      | Розробка комп'ютерної системи для домашньої бухгалтерії | Літ.                                     |  |  | Маса      |  | Масштаб |  |
| Зм.       | Арк. | № документа     | Підпис                                                                                | Дата |                                                         |                                          |  |  |           |  |         |  |
| Розробив  |      | Лунонос І.А.    |  |      |                                                         |                                          |  |  |           |  |         |  |
| Перевішив |      | Хохлов М.М.     |                                                                                       |      |                                                         |                                          |  |  |           |  |         |  |
| Т.контр.  |      |                 |                                                                                       |      |                                                         | Аркуш 1                                  |  |  | Аркушів 4 |  |         |  |
| Н.контр.  |      | Щербак Н.В.     |  |      | Інтерфейси додатків контролю витрат                     | НУ «Запорізька політехніка», гр. КНТ-510 |  |  |           |  |         |  |
| Затвердив |      | Кудерметов Р.К. |                                                                                       |      |                                                         |                                          |  |  |           |  |         |  |

Характеристика фреймворків

Фреймворк Laravel



Використовують для розробки складних індивідуальних сайтів та додатків.

- + Висока швидкодія та багатофункціональність ядра.
- + Простота маршрутизації.
- + Ефективне об'єктно-реляційне відображення БД.
- + Проста інтеграція зі сторонніми сервісами та бібліотеками.

- залишає для фахівців чимало ручної праці, що уповільнює роботу, але забезпечує гнучкість під час створення програмних рішень.

Фреймворк Django



Найпопулярніший і найвідоміший фреймворк Python.

- + Стабільний, основні вразливі місця в ньому закриті
- + багато бібліотек, які можна використовувати для найрізноманітніших потреб (бібліотеки для роботи з соцмережами, найпопулярнішими платіжними системами)
- доволі високий поріг входження. для початківців може бути складно розібратись з такою великою кількістю файлів і специфічною структурою роботи з базами даних

Фреймворк Ruby on Rails



Використовується для розроблення веб-додатків, з відкритим вихідним кодом, яка використовується для створення сайтів і додатків.

- + Надає стандартизований і організований спосіб створення веб-додатків
- + Спрощує розробникам написання чистого, підтримуваного і масштабованого коду.
- + Побудований на мові програмування Ruby, відомій своєю простотою і читабельністю.
- Ruby не так широко використовується, як деякі інші мови програмування

Фреймворк Symfony



- + Висока стабільність та надійність
- + Можливості для масштабування та розширення функціоналу
- + Має високу гнучкість налаштування та можливість інтеграції з різними популярними CMS та сервісами

- Вивчення та оновлення фреймворку та зайняти багато часу

- Процес розвитку йде повільно

- На продуктивність програми впливає великий трафік

|           |      |                 |                                                                                       |      |                                                         |  |  |  |  |  |         |  |           |  |                                          |  |  |  |
|-----------|------|-----------------|---------------------------------------------------------------------------------------|------|---------------------------------------------------------|--|--|--|--|--|---------|--|-----------|--|------------------------------------------|--|--|--|
|           |      |                 |                                                                                       |      | 13.02070849.00008 ПЛІ2                                  |  |  |  |  |  |         |  |           |  |                                          |  |  |  |
|           |      |                 |                                                                                       |      | Розробка комп'ютерної системи для домашньої бухгалтерії |  |  |  |  |  | Літ.    |  | Маса      |  | Масштаб                                  |  |  |  |
|           |      |                 |                                                                                       |      |                                                         |  |  |  |  |  |         |  |           |  |                                          |  |  |  |
|           |      |                 |                                                                                       |      |                                                         |  |  |  |  |  |         |  |           |  |                                          |  |  |  |
| Зм.       | Арк. | № документа     | Підпис                                                                                | Дата | Характеристика фреймворків                              |  |  |  |  |  | Аркуш 2 |  | Аркушів 4 |  | НУ «Запорізька політехніка», гр. КНТ-510 |  |  |  |
| Розробив  |      | Лунонос І.А.    |  |      |                                                         |  |  |  |  |  |         |  |           |  |                                          |  |  |  |
| Перевірів |      | Хохлов М.М.     |  |      |                                                         |  |  |  |  |  |         |  |           |  |                                          |  |  |  |
| Т.контр.  |      |                 |                                                                                       |      | Характеристика фреймворків                              |  |  |  |  |  | Аркуш 2 |  | Аркушів 4 |  | НУ «Запорізька політехніка», гр. КНТ-510 |  |  |  |
| Н.контр.  |      | Щербак Н.В.     |  |      |                                                         |  |  |  |  |  |         |  |           |  |                                          |  |  |  |
| Затвердив |      | Кудерметов Р.К. |  |      |                                                         |  |  |  |  |  |         |  |           |  |                                          |  |  |  |

Характеристика середовища розробки (IDE)

|                              |                                                                                                                                                                                                                                              |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Visual Studio Code (VS Code) | - безкоштовне та легке у використанні середовище розробки, розроблене компанією Microsoft. VS Code має велику кількість розширень, що робить його дуже гнучким для програмістів будь-яких мов програмування;                                 |
| IntelliJ IDEA                | - потужне комерційне IDE, що розроблене компанією JetBrains. Воно підтримує багато мов програмування, включаючи Java, Kotlin, Python, JavaScript та багато інших. IntelliJ IDEA має багато корисних функцій для роботи з великими проектами; |
| Eclipse                      | - відкрите безкоштовне середовище розробки, що підтримує багато мов програмування, таких як Java, C++, Python тощо. Eclipse має широкий вибір плагінів і розширень, що робить його гнучким та налаштовуваним;                                |
| Sublime Text                 | - легке, швидке та потужне редактор коду, який підтримує багато мов програмування. Sublime Text має багато корисних функцій, таких як маркування синтаксису, відкладення, широкий вибір плагінів і тем;                                      |

Характеристика баз даних

| MySQL                                                                                                                                                                      |                                                                                                                                                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Переваги:<br>- легка встановлюваність та налаштування;<br>- велика швидкодія;<br>- широко підтримується вебзастосуванням;                                                  | Недоліки:<br>- обмежені можливості щодо аналізу та розширень функцій;<br>- менша підтримка для складних транзакцій порівняно з іншими СУБД;             |
| PostgreSQL                                                                                                                                                                 |                                                                                                                                                         |
| Переваги:<br>- повна підтримка стандартів SQL;<br>- підтримка географічних об'єктів та розширень для аналізу даних;<br>- висока надійність та безпека                      | Недоліки:<br>- більш високі вимоги до ресурсів порівняно з деякими іншими СУБД;<br>- якщо не налаштована правильно, може бути повільною у використанні; |
| SQLite                                                                                                                                                                     |                                                                                                                                                         |
| Переваги:<br>- проста використання інтеграції;<br>- не потребує серверу баз даних, або налаштування;<br>- підтримується в більшості мов програмування;                     | Недоліки:<br>- не підходить для великих проєктів з високими навантаженнями;<br>- має обмежену підтримку функцій та типів даних порівняно з іншими СУБД; |
| Microsoft SQL Server                                                                                                                                                       |                                                                                                                                                         |
| Переваги:<br>- широкі можливості для аналізу та обробки даних;<br>- велика надійність та підтримка корпоративних середовищ;<br>- інтеграція з іншими продуктами Microsoft; | Недоліки:<br>- ліцензійна вартість може бути високою для малих та середніх підприємств;<br>- може бути складною для налаштування та адміністрування;    |

|           |      |                 |                                                                                       |      |  |                                                         |                                          |      |           |
|-----------|------|-----------------|---------------------------------------------------------------------------------------|------|--|---------------------------------------------------------|------------------------------------------|------|-----------|
|           |      |                 |                                                                                       |      |  | 13.02070849.00008 ПЛЗ                                   |                                          |      |           |
|           |      |                 |                                                                                       |      |  | Розробка комп'ютерної системи для домашньої бухгалтерії | Літ.                                     | Маса | Масштаб   |
| Зм.       | Арк. | № документа     | Підпис                                                                                | Дата |  |                                                         |                                          |      |           |
| Розробив  |      | Лупонос І.А.    |  |      |  |                                                         |                                          |      |           |
| Перевірів |      | Хохлов М.М.     |  |      |  |                                                         |                                          |      |           |
| Т.контр.  |      |                 |                                                                                       |      |  |                                                         | Аркуш 3                                  |      | Аркушів 4 |
| Н.контр.  |      | Щербак Н.В.     |  |      |  | Характеристика середовища розробки та бази даних        | НУ «Запорізька політехніка», гр. КНТ-510 |      |           |
| Затвердив |      | Кудерметов Р.К. |  |      |  |                                                         |                                          |      |           |

Інтерфейс додатку

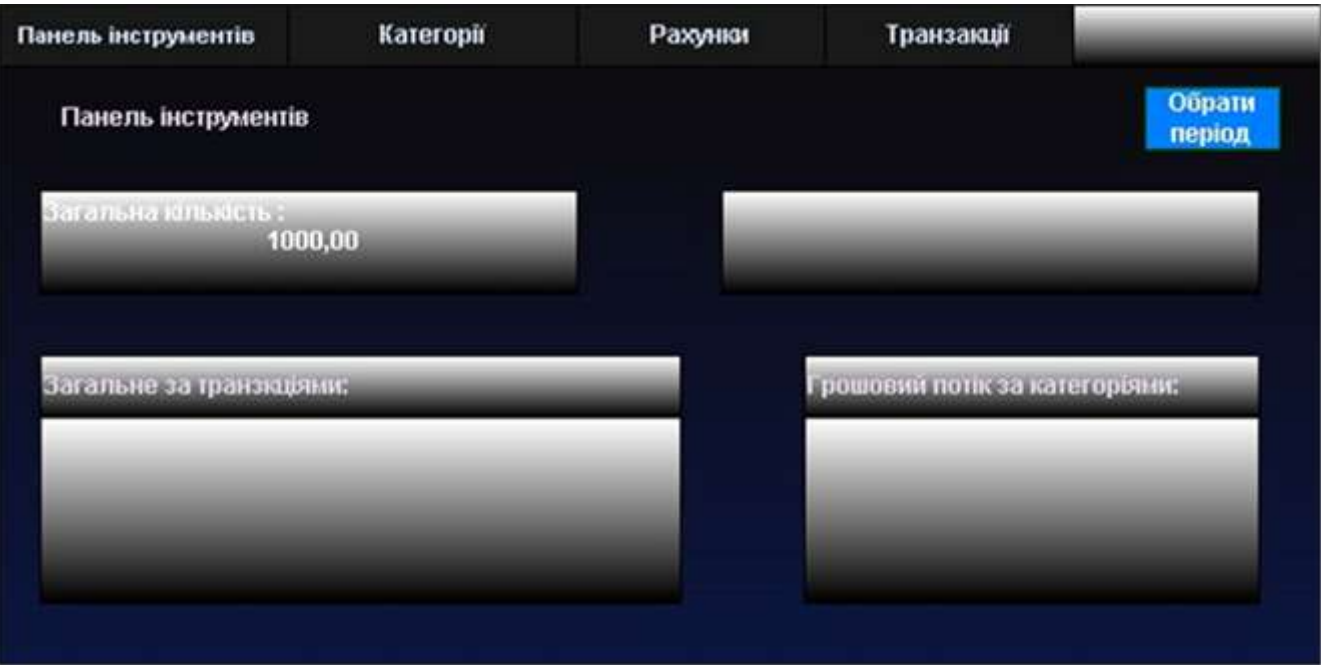


Рисунок 1 - Головна сторінка застосунку

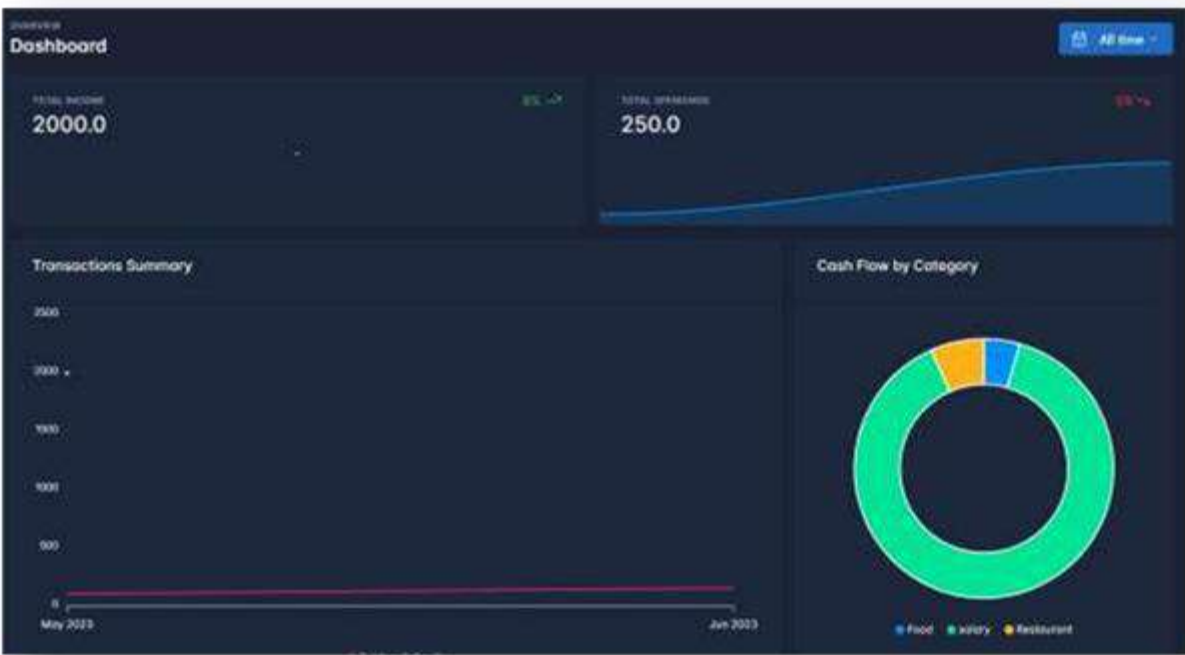


Рисунок 2 - Аналітика витрат та доходів



Рисунок 3 - Перелік гаманців

|           |      |              |        |      |                                                         |                 |  |  |           |  |         |                                          |
|-----------|------|--------------|--------|------|---------------------------------------------------------|-----------------|--|--|-----------|--|---------|------------------------------------------|
|           |      |              |        |      | 13.02070849.00003 ПЛ4                                   |                 |  |  |           |  |         |                                          |
|           |      |              |        |      | Розробка комп'ютерної системи для домашньої бухгалтерії | Літ.            |  |  | Маса      |  | Масштаб |                                          |
|           |      |              |        |      |                                                         |                 |  |  |           |  |         |                                          |
|           |      |              |        |      |                                                         |                 |  |  |           |  |         |                                          |
|           |      |              |        |      |                                                         |                 |  |  |           |  |         |                                          |
| Зм.       | Арк. | № документа  | Підпис | Дата | Інтерфейс додатку                                       | Аркуш 4         |  |  | Аркушів 4 |  |         | НУ «Запорізька політехніка», гр. КНТ-510 |
| Розробив  |      | Лупонос І.А. |        |      |                                                         |                 |  |  |           |  |         |                                          |
| Перевірів |      | Хохлов М.М.  |        |      |                                                         |                 |  |  |           |  |         |                                          |
| Т.контр.  |      |              |        |      |                                                         |                 |  |  |           |  |         |                                          |
| Н.контр.  |      | Щербак Н.В.  |        |      | Затвердив                                               | Кудерметов Р.К. |  |  |           |  |         |                                          |
|           |      |              |        |      |                                                         |                 |  |  |           |  |         |                                          |