МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ГОРДОВСЬКИЙ Олег Анатолійович

Веб-базована система управління особистим бюджетом / Web-based Personal Budget Management System

спеціальність: 122 – Комп'ютерні науки

освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи
КН-42
О.А. Гордовський

_____

Науковий керівник:
к.т.н., доцент П.Є. Биковий

_____

Кваліфікаційну роботу допущено
дозахисту

«___»_____20___р.

Завідувач кафедри
_____М.П. Комар

Тернопіль – 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ГОРДОВСЬКИЙ Олег Анатолійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Веб-базована система управління особистим бюджетом / Web-based Personal Budget Management System
керівник роботи Биковий Павло Євгенович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 12 грудня 2023 р. №753.

2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- провести опис предметної області;
- провести аналіз існуючих рішень;
- розробити архітектуру веб-базованої системи;
- реалізувати функціонал для створення особистих профілів;
- реалізувати можливість керування необмеженою кількістю гаманців та бюджетів;
- створити інструменти для детальної категоризації витрат;
- провести тестування веб-базованої системи.

5. Перелік графічного матеріалу в роботі:
- Use case діаграма;
- діаграма класів.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Н. контроль	к.т.н., доцент П.Є. Биковий		

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент _____ О.А. Гордовський

(підпис)

Керівник проекту _____ П.Є. Биковий

(підпис)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Веб-базована система управління особистим бюджетом» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 59 сторінки і містить 27 ілюстрацій, одну таблицю, чотири додатки та 26 використаних джерел.

Метою роботи є розробка веб-базованої системи управління особистим бюджетом, яка дозволяє користувачам ефективно планувати, аналізувати та контролювати свої фінансові ресурси.

Методами розроблення обрано аналіз існуючих рішень у сфері управління особистими фінансами, визначено їхні переваги та недоліки, і на основі цього створено інноваційне програмне забезпечення, яке поєднує найкращі функції аналогів та уникає їхніх недоліків. Особлива увага приділяється забезпеченню інтуїтивно зрозумілого інтерфейсу для покращення користувацького досвіду та доступності системи для широкого кола користувачів.

Внаслідок виконання роботи обґрунтовано раціональний підхід до розроблення моделей функції створення особистих профілів, керування необмеженою кількістю гаманців та бюджетів, а також інструменти для детальної категоризації витрат та розроблено програмний засіб, який дозволяє створювати і досліджувати моделі та включає функції створення особистих профілів, керування необмеженою кількістю гаманців та бюджетів, а також інструменти для детальної категоризації витрат та має інтуїтивно зрозумілий інтерфейс для забезпечення легкості використання та доступності для широкого кола користувачів.

Результати дослідження підтвердили ефективність та актуальність розробленого рішення, яке забезпечує зручне відстеження фінансових операцій та активне управління ними в реальному часі. Робота демонструє високий рівень дослідницької та інженерної майстерності, відповідає сучасним вимогам та потребам користувачів у сфері управління особистими фінансами.

Ключові слова: ВЕБ-БАЗОВАНА СИСТЕМА, УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ, ОСОБИСТИЙ БЮДЖЕТ, ВЕБ-ДОДАТОК, ГАМАНЕЦЬ.

ANNOTATION

The qualification work titled " Web-based Personal Budget Management System" for the Bachelor's degree in specialty 122 "Computer Science" of the educational program "Computer Science" consists of 59 pages and includes 27 illustrations, one table, four appendices, and 26 references.

The goal of this qualification work is to develop a web-based personal budget management system that allows users to efficiently plan, analyze, and control their financial resources.

The development methodology was based on analyzing existing solutions in the field of personal finance management, identifying their advantages and disadvantages, and on this basis creating innovative software that combines the best features of analogues and avoids their disadvantages. Particular attention is paid to providing an intuitive interface to improve user experience and make the system accessible to a wide range of users

As a result of the work, a rational approach to the development of models of the function of creating personal profiles, managing an unlimited number of wallets and budgets, as well as tools for detailed categorization of expenses is substantiated and a software tool is developed that allows creating and researching models and includes the functions of creating personal profiles, managing an unlimited number of wallets and budgets, as well as tools for detailed categorization of expenses and has an intuitive interface to ensure ease of use.

The results of the study confirmed the effectiveness and relevance of the developed solution, which provides convenient tracking of financial transactions and their active management in real time. The work demonstrates a high level of research and engineering skills, meets modern requirements and user needs in the field of personal finance management.

Keywords: WEB-BASED SYSTEM, PERSONAL FINANCE MANAGEMENT, PERSONAL BUDGET, WEB APPLICATION, WALLET.

ЗМІСТ

ВСТУП

У сучасному економічному кліматі, який характеризується зростанням фінансової нестабільності та збільшенням економічних ризиків, планування особистого бюджету набуває вирішального значення для забезпечення фінансової стійкості та благополуччя індивідуальних осіб. Система управління особистими фінансами стає критично важливою для зменшення фінансових стресів і підвищення фінансової свідомості серед населення. Відсутність ефективних інструментів для контролю та аналізу фінансових ресурсів може призвести до фінансових труднощів і неспроможності досягти поставлених фінансових цілей.

Метою даної роботи є розробка веб-базованої системи для управління особистим бюджетом, яка дозволить користувачам ефективно планувати, аналізувати та контролювати свої фінансові ресурси. Система має поєднувати в собі найкращі риси існуючих аналогів і уникати їх недоліків, забезпечуючи користувачам інтуїтивно зрозумілий інтерфейс і широкі можливості для керування фінансами.

Основними завданнями роботи є: проведення опису предметної області; аналіз існуючих рішень; розробка архітектуру веб-базованої системи; реалізація функціоналу для створення особистих профілів; реалізація можливості керування необмеженою кількістю гаманців та бюджетів; створення інструментів для детальної категоризації витрат; проведення тестування розробленої системи з метою підтвердження її ефективності та актуальності.

Об'єктом дослідження є веб-базована система управління особистим бюджетом, яка дозволяє користувачам планувати, аналізувати та контролювати свої фінансові ресурси. Вона включає різноманітні функції для детального управління фінансами і забезпечує зручність використання для широкого кола користувачів.

Предметом дослідження є методи та засоби розробки веб-базованих систем управління особистими фінансами, включаючи архітектурні підходи, технології реалізації, безпеку даних та користувацький досвід. Особлива увага приділяється

забезпеченню інтуїтивно зрозумілого інтерфейсу та високого рівня зручності використання системи.

Апробацію дослідження проведено на міжнародній науково-практичній концеренції м. Ізмаїл, 18 травня 2024 р. “Перспективні напрямки розвитку економіки, обліку, управління та права.” В секції “Математичні методи моделі та інформаційні технології в економіці: теорія і практика” де була представлена доповідь на тему “Огляд Веб-базованих системи управління особистим бюджетом” і опубліковані тези по даній доповіді.

1.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз та актуальність теми

У сучасному економічному кліматі, що характеризується зростанням фінансової нестабільності та збільшенням економічних ризиків, планування особистого бюджету набуває вирішального значення для забезпечення фінансової стійкості та благополуччя індивідуальних осіб. Цей розділ ставить за мету дослідити ключові аспекти планування особистого бюджету [1], визначити його значення та вплив на фінансове управління, а також розглянути стратегії та інструменти, які можуть сприяти ефективному управлінню фінансами.

Основний акцент розділу спрямований на аналіз та розкриття важливості планування особистого бюджету у контексті забезпечення фінансової стабільності та досягнення фінансових цілей. Через призму наукових досліджень та практичного досвіду, буде розглянуто роль планування бюджету у формуванні фінансових навичок та умінь, а також вплив його використання на зменшення фінансових стресів та підвищення фінансової свідомості серед населення.

Планування особистого бюджету є ключовим етапом у фінансовому управлінні, спрямованим на ефективне розподілення фінансових ресурсів для досягнення конкретних фінансових цілей та забезпечення фінансової стабільності. Важливість планування бюджету [2] полягає в його здатності визначити реальні фінансові потреби та обмеження, а також встановити стратегії для їхнього вирішення. Цей процес передбачає аналіз поточних фінансових обставин, визначення пріоритетних цілей та розробку плану дій для їх досягнення.

Одним із ключових аспектів планування бюджету є створення бюджету на основі реальних фінансових даних [3]. Це означає врахування всіх джерел доходу та розподілу коштів на різні види витрат, включаючи основні потреби, такі як житло, харчування, одяг, а також екстрені витрати та розваги. Забезпечення адекватного бюджету вимагає уважного аналізу фінансової ситуації, зокрема оцінки рівня доходів, регулярних витрат та можливості для збереження.

Далі, планування бюджету включає в себе визначення конкретних фінансових цілей [4] та розробку стратегій для їх досягнення. Ці цілі можуть включати погашення боргів, накопичення екстреного фонду, вкладення в освіту чи розвиток бізнесу. Важливо встановити реалістичні та конкретні цілі, які можна виміряти та досягти за певний період часу. Після встановлення цілей, необхідно розробити план дій, який включає в себе конкретні кроки та стратегії, які допоможуть досягти цих цілей, а також визначити відповідні ресурси та часові рамки для їх виконання.

Завершальним етапом планування бюджету є відстеження та оцінка його ефективності[6]. Під час цього етапу важливо регулярно переглядати бюджет, аналізувати реальні витрати та доходи, порівнювати їх з запланованими показниками та коригувати стратегії, якщо необхідно. Постійне відстеження бюджету допомагає підтримувати фінансову дисципліну, [8] виявляти потенційні проблеми та розробляти ефективніші стратегії управління фінансами.

Важливість планування бюджету полягає в його здатності доцільно розподіляти фінансові ресурси для досягнення фінансових цілей та забезпечення фінансової стабільності в довгостроковій перспективі, види цих доходів зображено на рис 1.1.

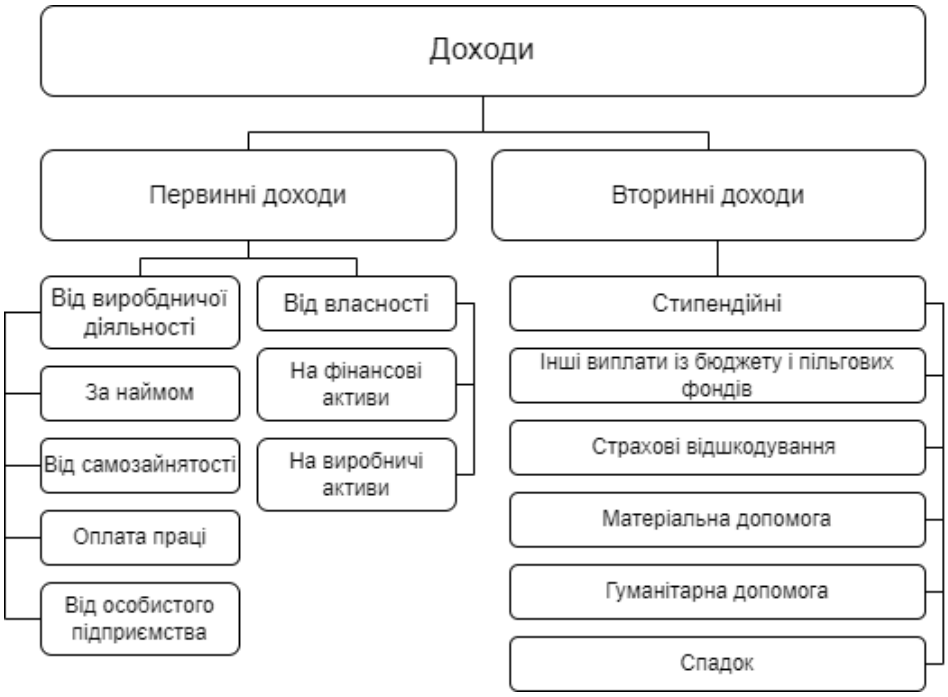


Рисунок 1.1 – Види доходів

Планування особистого бюджету є невід'ємною складовою фінансової грамотності та стабільності [9]. В даному розділі було розглянуто ключові аспекти цього процесу, починаючи від аналізу фінансової ситуації та закінчуючи відстеженням його ефективності.

Завдяки вивченню цього процесу, ми розуміємо, що планування бюджету вимагає не лише математичних обчислень, але й стратегічного мислення та управлінських навичок [10]. Встановлення реалістичних фінансових цілей, розробка конкретних планів дій для їх досягнення та постійне відстеження результатів допомагають керувати фінансами ефективно та забезпечити фінансову стабільність.

Отже, планування особистого бюджету є важливим інструментом, який допомагає кожній особі здійснити контроль над своїми фінансами та досягти фінансових цілей. Впровадження цього процесу в повсякденне життя сприяє розумінню та управлінню фінансовими ресурсами з вищим рівнем ефективності та відповідальності.

1.2 Аналіз існуючих аналогів

Незважаючи на те, що всесвітній ринок переповнений мобільними додатками для управління фінансами та бюджетуванням, не кожне рішення є досконалим або доступним для кожного користувача. Такі рішення, здебільшого, є непрактичними. [11-16] Інші рішення, з іншого боку, пропонують кілька основних функцій; однак основні з них заблоковані для користувачів платним планом підписки. У будь-якому випадку це може перешкодити користувачам у використанні таких програм.

Тому в наступному розділі аналізуються поточні рішення аналогів, наприклад, Spendee, Wallet, Pocket Expense 6. Кожну програму було обрано на основі її релевантності, рейтингів або популярності. Не дивно, що вони мають кілька спільних рис. Таким чином, мета полягає в тому, щоб зосередитися на наведених нижче фундаментальних характеристиках, які є спільними для поточних

рішень, і визначити їхні переваги та недоліки. Висновок і порівняння характеристик усіх проаналізованих програм таблиця 1.1 доступний у кінці розділу. Крім того, у цьому розділі гаманці та облікові записи або транзакції та записи використовуються більш-менш взаємозамінно, оскільки в проаналізованих програмах ці терміни називаються по-різному.

Spendee це персональний мультиплатформовий бюджетний додаток, який доступний в Інтернеті, на iOS і операційній системі (ОС) Android і з більш ніж трьома мільйонами завантажень, це також один із найпопулярніших бюджетних додатків у країні. Крім місцевого мобільного ринку, він також доступний у більш ніж 150 країнах. Spendee [17] зображено на рисунок.1.2

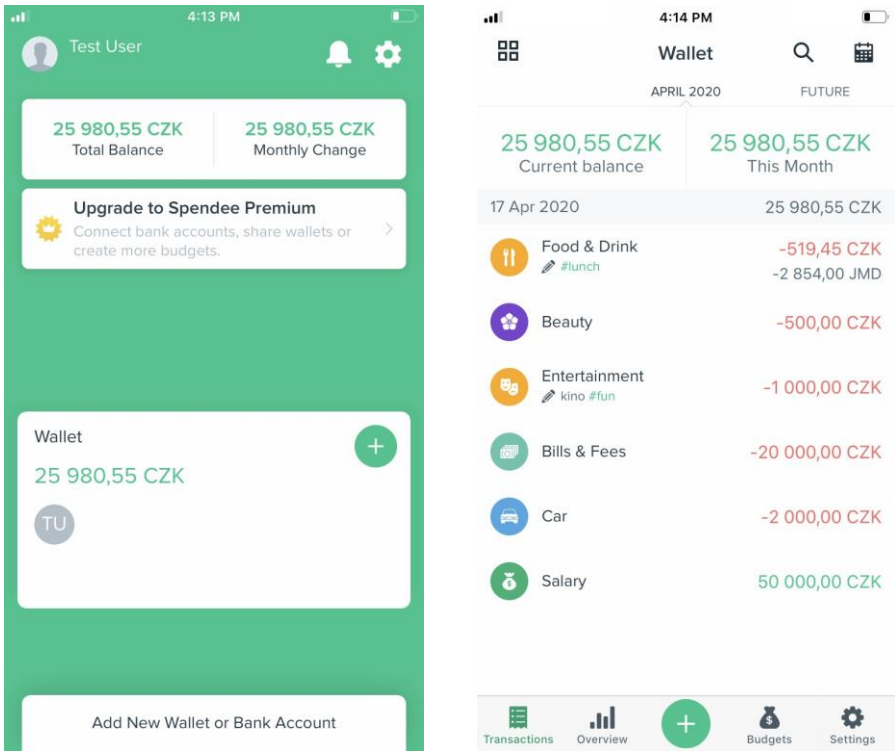


Рисунок 1.2 – Інтерфейс додатку Spendee

Сама програма безкоштовна; однак її основні функції (як зазначено в описі програми) помітно обмежені планом передплати, який мають користувачі безкоштовний або преміальний. Деякі з них обговорюються в параграфах нижче.

Профіль користувача У той час, коли для деяких програм не потрібен обліковий запис, для Spendee це необхідно. Це завдяки функції спільного доступу, яка дозволяє користувачам ділитися своїми гаманцями з іншими. Для цього користувачі змушені створити профіль за допомогою електронної пошти, Facebook або Google. Потім користувачі можуть змінити своє ім'я, прізвище або додати дату народження.

Тим не менш, зручно, що реєстрація проходить швидко і не вимагає перевірки адреси електронної пошти. З іншого боку, програма відразу після завершення реєстрації показує спливаюче діалогове вікно, пропонуючи преміум-план підписки, що є жахливим способом сказати, що функції програми заздалегідь обмежені.

Гаманець представляє місце, тобто банк, заощадження або готівку, де користувачі відстежують свої транзакції. Ця функція імітує реальний сценарій, у якому кожен користувач має різну кількість облікових записів. У результаті користувачі можуть відстежувати різні види фінансів окремо.

У той час як преміум-користувачі мають необмежену кількість гаманців, безкоштовні користувачі мають лише один гаманець. Незважаючи на те, що це єдиний недолік, він водночас знецінив свою єдину перевагу. Без кількох гаманців це просто відро купюр, скупчених в одному місці.

Бюджети ця функція встановлює певне обмеження на гаманець користувача для однієї або кількох категорій. Потім вона обробляє та показує всі транзакції в певних категоріях і відображає відсоток витрачених грошей, що дає користувачам певне розуміння їхніх фінансових звичок і може допомогти їм витрачати менше, таким чином, заощаджуючи більше. Подібно до попередньої функції, можливість створювати кілька бюджетів доступна лише преміум-користувачам.

Функція категоризації дозволяє користувачам підтримувати свої фінанси в порядку. Користувачі можуть класифікувати свої транзакції за категоріями, примітками або хештегами. Програма постачається з попередньо визначеним списком категорій, які користувачі можуть легко змінити або додати. Більше того, їх можна створити скільки завгодно. Пізніше в додатку кожну транзакцію можна класифікувати.

Додаток підтримує лише пошук, який виводить транзакції, які відповідають запитуваній фразі, але лише ті, які категоризовані. Результат розгортається залежно від того, що шукали.

Інтерфейс користувача програми сучасний, зрозумілий і водночас дуже простий. Кольори його інтерфейсу користувача залежать від колірної теми мобільного пристрою користувача, тобто світлого чи темного. Після запуску програми користувачам відкривається основний екран гаманця, який містить список транзакцій, де вони можуть швидко перевірити поточний баланс і витрати. Навігація в додатку інтуїтивно зрозуміла.

Для переміщення між такими екранами, як список транзакцій, огляд гаманця або налаштування, можна використовувати панель вкладок, розташовану внизу екрана мобільного пристрою. Незважаючи на те, що під час аналізу було виявлено кілька помилок, загальне враження відповідає оцінці програми.

Програма WalletApp зображено на (рис 1.3), розроблена чеським стартапом Budgetbakers, є ще одним улюбленим інструментом для керування особистими фінансами. Окрім трьох мільйонів завантажень та високого рейтингу 4,7 з 5 зірок в App Store, варто зазначити, що Budgetbakers може похвалитися найновішою функціональністю, яка на даний час тестується в бета-версії програми.

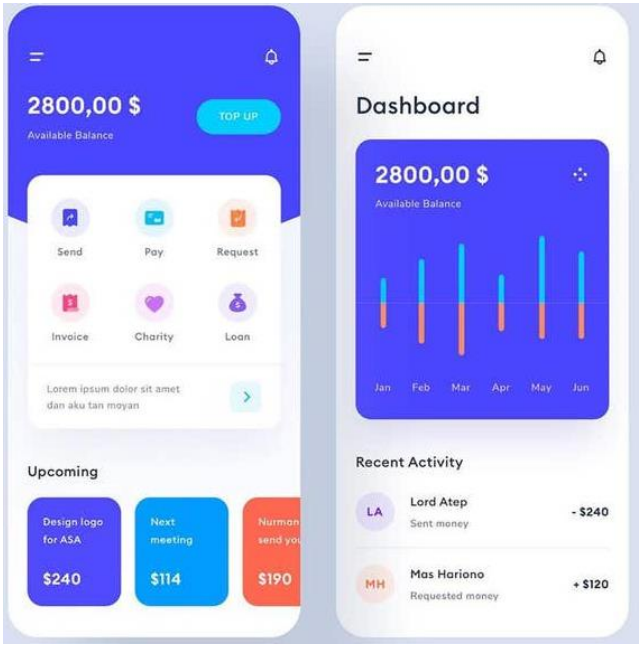


Рисунок 1.3 – Інтерфейс додатку Wallet

Ця функція дозволяє користувачам здійснювати фактичні фінансові транзакції прямо в додатку, що робить Wallet [18] першим в історії бюджетним додатком, який має таку можливість.

Програма WalletApp, на відміну від Spendee, оснащена широким спектром безкоштовних функцій та статистичних інструментів, що ідеально підійдуть для вимогливих користувачів, хоча можуть виявитися складними для початківців у сфері бюджетного планування. Проте, ймовірно, не всі функції є безкоштовними. Крім того, варто відзначити грувальну функцію програми, яка відстежує задоволеність користувача своїми записами.

Профіль користувача так само, як і в попередній програмі, користувачі можуть зареєструватися за допомогою тих самих методів, проте не потрібно підтверджувати адресу електронної пошти. Після реєстрації програма пропонує початкову валюту облікового запису та налаштування початкового балансу. Користувачі також можуть персоналізувати свій профіль, додавши додаткову інформацію та коригуючи свої дані та установки приватності (наприклад, відстеження геолокації, рекламні електронні листи або сегментація клієнтів) і навіть видаливши профіль.

Гаманці додаток надає користувачам можливість створити до трьох облікових записів; будь-яка інша кількість доступна лише для платних користувачів. Незважаючи на це, створення облікового запису є простим і дозволяє користувачам налаштувати обліковий запис з відповідними параметрами, такими як поточний баланс, тип рахунку, валюта, колір або навіть виключення облікового запису із загальної статистики. Такий процес є функцією, яку Spendee не пропонує навіть зі своїм єдиним гаманцем.

Функція бюджетів не має обмежень, що означає, що користувачі можуть створювати та відстежувати бюджети без обмежень. Функція чітко показує, скільки грошей витрачено та залишилося в конкретному бюджеті. Кожен бюджет зберігає пов’язані записи, які доступні лише платним користувачам, із детальною статистичною інформацією.

Додаток має два методи категоризації — категорії та мітки. Категорії мають більш-менш ту саму мету, що і в попередній програмі, проте вони відрізняються налаштуваннями. Заздалегідь підготовлено одинадцять категорій, які можна лише редагувати (користувачі не можуть створювати чи видаляти їх). Крім того, кожна категорія містить кілька підкатегорій, які також можна лише редагувати. Тим не менш, можна додати інші підкатегорії.

Мітка - це характеризуюче слово, яке надається об'єкту. У додатку мітки мають точне призначення і користувачі можуть розрізняти їх, визначаючи кольори. Проте, кольори — ще одна платна функція.

Фільтр і пошук за допомогою якого додаток ретельно відстежує записи користувача та детально відображає їх у статистичних графіках. Користувачі можуть змінювати свій вміст за допомогою фільтрації за рахунками та періодом, а також встановлювати власні фільтри за категоріями, мітками, валютами та іншими.

Весь додаток виконаний в базовому інтерфейсі користувача, який відповідає стандартам дизайну розробки iOS, забезпечуючи природну навігацію та використання програми. Однак, деякі екрани можуть бути перевантаженими інформацією, що може здатися хаотичною. Ще одним недоліком є розміщення екранів один поруч з одним, яке може стати проблемою для деяких користувачів.

Pocket Expense [19] менш популярний порівняно з попередніми програмами, але все ж пропонує такий же конкурентоспроможний дизайн і послуги, як показано на рисунку 1.4. Програма переважно підходить для користувачів, які не очікують складної інформації чи графіків і обходяться основними інструментами бюджетування.

На відміну від попередніх програм, Pocket Expense має потроху всі проаналізовані функції. Незважаючи на те, що на перший погляд він здається ідеальним, його характеристики мають як позитивні, так і негативні аспекти.

По відміну від попереднього додатка, Pocket Expense передбачає лише реєстрацію за допомогою електронної пошти. Однак приємно, що він також дає користувачам можливість реєструватися без пароля, для чого потрібна перевірка електронної пошти. На жаль, через помилку на стороні програми цю функцію не

вдалося протестувати. Що стосується самого профілю, то він слугує способом експорту або відновлення даних користувача (у той час як функція експорту є платною), оскільки немає додаткової персоналізації профілю, крім зміни фотографії профілю.

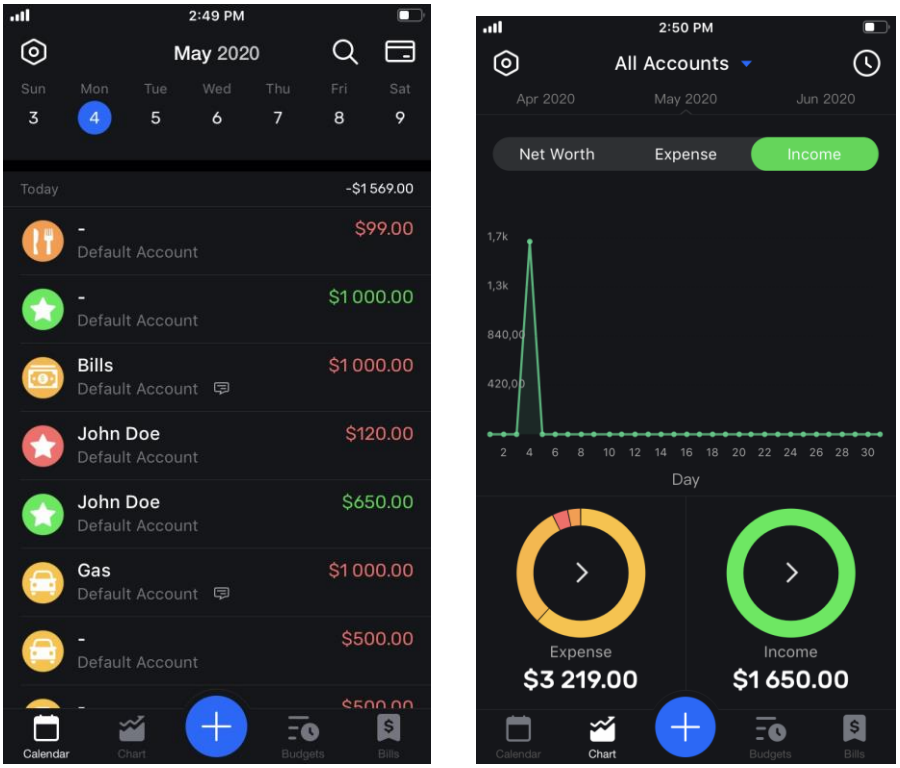


Рисунок 1.4 – Інтерфейс додатку Wallet

Як безкоштовний користувач, доступний лише один обліковий запис без початкового налаштування — преміум-профіль потрібен для створення кількох облікових записів із налаштуванням користувача. За замовчуванням для облікового запису встановлено валюту США, яка, здається, прив’язана до профілю, а не до облікового запису.

Після створення бюджету користувачам програми призначено заощаджувати гроші, використовуючи єдиний бюджет з однією незмінною категорією. Потім програма відображає суму, що залишилася в бюджеті, коли користувачі додають свої витрати у відстежувану категорію, пов’язану з фактичними витратами.

Користувачі можуть організувати свої витрати за категоріями, яких у них може бути необмежена кількість. Подібно до наведених вище програм, кожна

трансакція, що відстежується, пов’язана з однією категорією або, у цьому випадку, розділена на кілька категорій, наприклад, платіж щодо двох окремих елементів.

Функція фільтра змінює вміст статистичних даних усіх облікових записів або одного окремого облікового запису. За допомогою фільтра користувачі можуть відображати статистику транзакцій за вибраний відфільтрований період, наприклад тиждень, місяць, рік або спеціальний час. Пошук — це складна та швидка функція, яка знаходить транзакції, які відповідають пошуковій фразі, якою може бути категорія, примітка, одержувач платежу або сума. Крім того, можна шукати пов’язані транзакції тієї самої категорії в списку транзакцій.

Незважаючи на те, що програма непопулярна, вона має модний інтерфейс користувача, однозначні елементи та піктограми, що робить програму доступною навіть для користувачів без попереднього досвіду роботи з програмами такого типу.

Додаток представляє список транзакцій під час запуску, який, здається, показує все, що потрібно користувачеві; однак елементи відсутні в терміні UX. Неоднозначно, на яку категорію витрачалися видатки. Додаток також надає можливість створити транзакцію з розділеними категоріями, що означає розподіл суми. Результати порівняння цих платформ зображено у таблиці 1.1

Таблиця 1.1 Результати порівняння існуючих аналогів

Функціонал	Spendee	Wallet	Pocket Expense
Профіль користувача	+	+	+
Необмежені гаманці	-	-	-
Необмежені бюджети	-	+	-
Категоризація	+	+	+
Фільтри	-	+	+
Пошук	+	-	+

1.3 Постановка задачі дослідження

Проаналізувавши дану предметну область та існуючі аналоги, зрозуміло що у даній системі управління особистим бюджетом потрібно реалізувати наступні функції.

Створення особистих профілів, ця функція дозволяє користувачам створювати індивідуальні облікові записи, вводячи особисті дані, такі як ім'я, адреса електронної пошти, номер телефону та іншу контактну інформацію, а також зберігати та керувати своєю фінансовою інформацією в безпечному середовищі.

Персональні профілі є важливою функцією в системах особистого бюджету, оскільки дозволяє користувачам зберігати та керувати своєю фінансовою інформацією в безпечному та конфіденційному середовищі. При створенні профілю користувачі можуть вводити особисту інформацію, таку як ім'я, адреса електронної пошти, номер телефону та інші контактні дані, що дозволяє їм створити окремий обліковий запис для зберігання своїх фінансових даних.

Крім того, створення особистого профілю надає користувачам доступ до додаткових можливостей і функцій, які можуть допомогти їм ефективно управляти своїми фінансами. Приклади включають зберігання історії транзакцій, встановлення фінансових цілей, нагадування про важливі події та інші персоналізовані функції. Такі персональні профілі є необхідним компонентом побудови персоналізованого підходу до управління фінансами та досягнення фінансових цілей.

Управління бюджетом дозволяє користувачам аналізувати доходи та витрати, визначати ключові фінансові цілі та розподіляти фінансові ресурси для їх досягнення.

Функція бюджетування, це важливий компонент системи особистого фінансового планування. Ця функція надає користувачам корисний інструмент для ефективного управління доходами та витратами. Мета бюджетного управління - забезпечити раціональне використання фінансових ресурсів, визначити пріоритетні напрями видатків і підтримувати фінансову стабільність.

Метою даної роботи є реалізація трьох ключових функцій бюджетного контролю: встановлення бюджетних лімітів для різних видів видатків, що дає змогу

користувачам уникати перевищення видатків, дотримуватися фінансових лімітів; функція підтримки фінансової дисципліни дозволяє досягати фінансових цілей; функція управління бюджетом дає змогу користувачам аналізувати видатки та виявляти статті, на яких можна заощадити, оптимізуючи таким чином фінансові ресурси та підвищуючи ефективність управління бюджетом.

Відстеження витрат. Функція дозволяє користувачам реєструвати та аналізувати свої витрати, надаючи повний огляд їхньої фінансової поведінки та допомагаючи виявити непотрібні або надмірні витрати.

Відстеження витрат є одною із найважливіших функцій системи планування особистого бюджету. Вона дозволяє користувачам повністю контролювати свої особисті фінанси та уникати непередбачуваних витрат. Ця функція дозволяє користувачам реєструвати всі транзакції в різних категоріях, надаючи їм детальну фінансову картину їхнього життя.

Дана функція відстеження також допомагає користувачам аналізувати свої фінансові звички та виявляти тенденції у витрачанні коштів. Це дозволяє їм визначити сфери, де вони можуть заощадити гроші, і розробити стратегії для оптимізації своїх витрат. Крім того, відстеження витрат допомагає користувачам планувати свої майбутні витрати і створювати реалістичні бюджетні прогнози. Це ключовий елемент забезпечення фінансової стабільності та ефективного управління особистим бюджетом.

Планування фінансових цілей, вона дозволяє користувачам визначати конкретні цілі, розробляти стратегії для їх досягнення та встановлювати відповідні ресурси і часові рамки для їх досягнення.

Можна ставити конкретні цілі та розробляти стратегії для їх досягнення. Ця функція допомагає користувачам визначати пріоритети своїх витрат і раціонально розподіляти фінансові ресурси для досягнення поставлених цілей. Без планування фінансових цілей користувачам може бути важко зосередитися на конкретних фінансових завданнях і відповідно оцінити свої фінансові можливості.

Більше того, функція планування фінансових цілей допомагає користувачам створити мотивацію для ефективного управління своїми фінансами. Постановка

конкретних цілей допомагає користувачам зосередитися на досягненні певних результатів і сконцентрувати свою увагу на досягненні фінансової стабільності. Крім того, планування фінансових цілей допомагає користувачам зосередитися на своїх фінансових цілях і приймати обґрунтовані фінансові рішення, забезпечуючи фінансове благополуччя і максимізуючи результати.

Аналіз та звітність ця функція надає користувачам можливість аналізувати фінансову звітність, включаючи зведені таблиці, діаграми та графіки, щоб допомогти їм зрозуміти своє фінансове становище та приймати обґрунтовані фінансові рішення.

Функція аналізу та звітності є важливою частиною системи особистого бюджету і надає користувачам можливість систематично аналізувати свої фінанси та отримувати звіти про свої доходи та витрати.

Ця функція допомагає користувачам зрозуміти, як вони витрачають свої фінансові ресурси, і визначити потенційні можливості для економії та оптимізації витрат. Без аналізу та звітності користувачам важко мати об'єктивне уявлення про свій фінансовий стан і приймати обґрунтовані фінансові рішення.

Крім того, функція аналізу та звітності дозволяє користувачам створювати детальні звіти про свої фінанси, включаючи зведені таблиці, діаграми та графіки. Це надає користувачам зручний спосіб візуалізувати фінансову інформацію та робити об'єктивні висновки про свій фінансовий стан. Крім того, фінансові звіти допомагають користувачам відстежувати прогрес у досягненні фінансових цілей і своєчасно реагувати на негативні тенденції або проблеми, що виникають.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Архітектури системи

Система контролю особистого бюджету буде побудована на основі модульної архітектури [20], що дозволить логічно розділити функціональність на окремі компоненти. Кожен модуль буде відповідати за конкретний аспект системи, такий як управління бюджетом, аналіз витрат, аутентифікація користувачів тощо. Це спростить розробку, тестування та підтримку системи, а також дозволить змінювати або розширювати окремі компоненти незалежно один від одного без впливу на решту системи.

Для забезпечення масштабованості та гнучкості системи буде використана мікросервісна архітектура [21-23] рисунок 2.1, в якій кожен функціональний блок буде реалізований як окремий сервіс. Кожен мікросервіс буде відповідати за власну область бізнес-логіки та матиме власну базу даних. Це дозволить легко масштабувати та розвивати систему, додавати нові функціональність без змін усієї системи та підвищувати надійність та стабільність роботи системи в цілому.

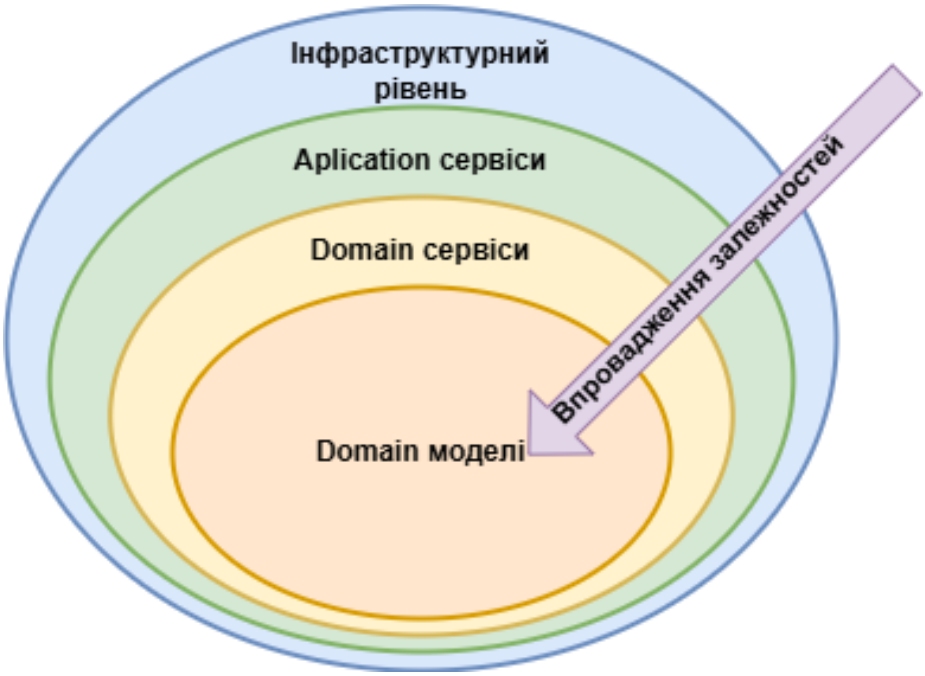


Рисунок 2.1 – Структура архітектурних рівнів

В системі буде приділена особлива увага захисту даних користувачів, що є критично важливим аспектом у контексті фінансової інформації. Застосування сучасних методів шифрування та безпеки, таких як SSL/TLS для захищеного з'єднання, а також регулярне оновлення системи та моніторинг безпеки допоможуть запобігти витоку конфіденційної інформації та злому даних. Крім того, будуть вжиті заходи для захисту від відомих типових атак, таких як SQL-ін'єкції або перехоплення сесій, що забезпечить високий рівень надійності та безпеки фінансових даних користувачів.

Архітектурний патерн Model-View-Controller (MVC) зображений на рисунку 2.2 забезпечує структуру для розробки веб-додатків, розділяючи його на три основні компоненти модель (Model), представлення (View) та контролер (Controller).

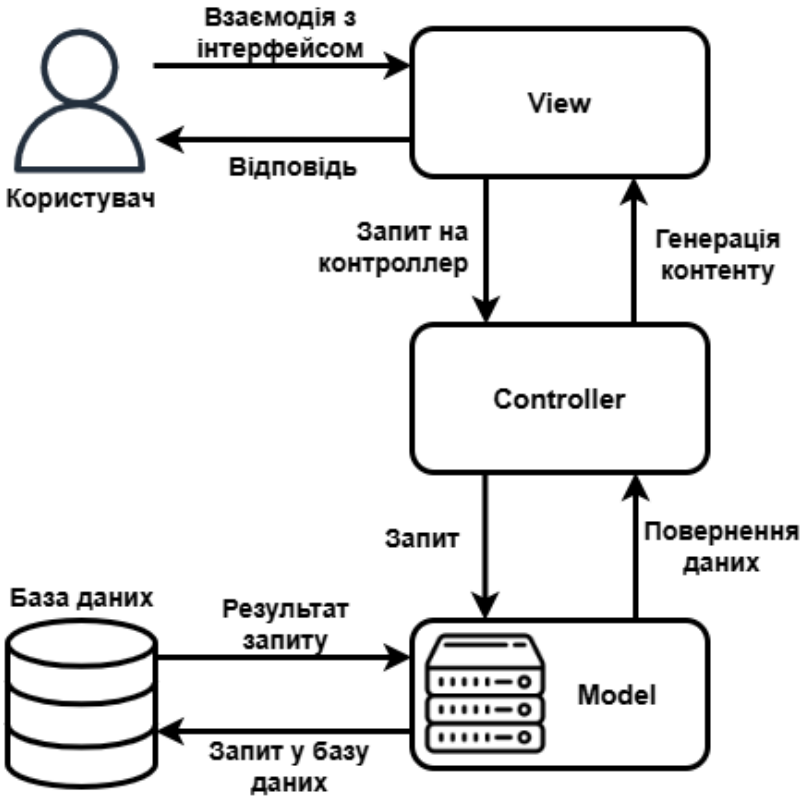


Рисунок 2.2 – Архітектурний патерн MVC

Модель відповідає за обробку даних та бізнес-логіку програми, вона управляє доступом до даних та виконує всі операції, необхідні для їх обробки. Представлення

відповідає за відображення даних користувачу та відображення результатів обробки даних моделлю, воно відокремлене від моделі та контролера і може бути змінене без впливу на решту системи.

Контролер відповідає за обробку вхідних запитів від користувача, він отримує запити від користувача через представлення, обробляє їх та ініціює відповідні дії в моделі. Контролер реагує на дії користувача, такі як натискання кнопок або введення даних в форму, та ініціює відповідні дії в моделі, яка змінює свій стан або виконує необхідні операції. Таким чином, архітектурний патерн MVC забезпечує гнучку та розділену структуру веб-базованої системи, що полегшує розробку, тестування та підтримку веб-додатків.

XML (Extensible Markup Language) обирається для зберігання конфігураційних даних системи, таких як параметри підключення до бази даних, налаштування безпеки, права доступу користувачів та інші налаштування. XML є популярним форматом для обміну структурованими даними через інтернет, і він забезпечує чітку та зручну структуру для збереження налаштувань у вигляді текстових файлів.

Використання XML для конфігурації дозволяє легко змінювати налаштування системи без необхідності змінювати сам код програми. Це забезпечує гнучкість та масштабованість системи, оскільки адміністратори можуть змінювати налаштування відповідно до потреб без перекомпіляції або перезапуску веб-додатку. Крім того, XML є стандартним форматом для обміну даними між різними системами, що спрощує інтеграцію та взаємодію з іншими програмами та сервісами. Таким чином, використання XML для конфігурації додає гнучкості та зручності в управління системою контролю особистого бюджету.

Entity Framework обирається як ORM (Object-Relational Mapping) для роботи з базою даних у веб-додатку. ORM дозволяє використовувати об'єктно-орієнтований підхід до роботи з даними, що спрощує розробку та підтримку системи [24-25]. Entity Framework автоматизує процеси створення запитів до бази даних, взаємодії з об'єктами даних та відображення даних в об'єктному вигляді, що робить розробку додатку більш ефективною та продуктивною.

Використання Entity Framework дозволяє позбутися від рутинної роботи з написанням SQL-запитів, оскільки весь доступ до бази даних відбувається через об'єктно-орієнтований інтерфейс. Це спрощує розробку та підтримку коду, а також робить його більш зрозумілим та читабельним. Крім того, Entity Framework забезпечує автоматичне відстеження змін в об'єктах даних та їх автоматичну синхронізацію з базою даних, що робить роботу з даними більш безпечною та надійною. Таким чином, використання Entity Framework у системі контролю особистого бюджету дозволить зосередитися на бізнес-логіці додатку, не звертаючи уваги на деталі роботи з базою даних.

Забезпечення безпеки є надзвичайно важливим аспектом для системи контролю особистого бюджету. З урахуванням чутливості фінансової інформації користувачів, система повинна мати ефективні механізми захисту, щоб забезпечити конфіденційність, цілісність та доступність даних. Для досягнення цієї мети буде використано різні методи та технології, такі як шифрування даних, аутентифікація та авторизація користувачів, контроль доступу та моніторинг активності користувачів.

Шифрування даних в базі даних та під час їх передачі по мережі забезпечить конфіденційність фінансової інформації, запобігаючи несанкціонованому доступу до неї.

Інтеграція з веб-сервером є ключовою складовою архітектури системи контролю особистого бюджету, оскільки вона забезпечує доступ користувачів до функціональності системи через інтернет. Для цього головний веб-додаток буде розгорнутий на веб-сервері, такому як Internet Information Services (IIS) або Apache, що забезпечить надійний та безпечний доступ до системи для користувачів через веб-браузери.

Інтеграція з веб-сервером передбачає налаштування середовища для виконання веб-додатків, включаючи установку та конфігурацію веб-сервера, налаштування доступу до бази даних, налаштування безпеки та моніторингу системи. Це забезпечить стабільну та ефективну роботу системи у веб-середовищі

та забезпечить користувачам зручний та безпечний доступ до своїх фінансових даних через інтернет.

Використання веб-сокетів є важливим аспектом для забезпечення реального часу оновлення та взаємодії між клієнтом та сервером у системі контролю особистого бюджету. Сокети забезпечують двостороннє з'єднання між клієнтом та сервером, що дозволяє обмінюватися даними в реальному часі без необхідності постійного оновлення сторінок веб-сайту. Це дозволяє користувачам отримувати миттєві оновлення про їхні фінансові транзакції, статус бюджету та іншу важливу інформацію без необхідності перезавантаження сторінки.

Це використання дозволяє знизити навантаження на сервер та мережу, оскільки вони дозволяють передавати дані без зайвих запитів з боку клієнта (рис. 2.3). Це забезпечує зменшення затримок та покращення продуктивності системи, особливо при великому обсязі даних або великій кількості активних користувачів. Крім того, веб-сокети дозволяють реалізувати різноманітні функціональні можливості, такі як чати, сповіщення та живе оновлення, що покращує користувацький досвід та забезпечує більшу залученість користувачів до системи контролю особистого бюджету.

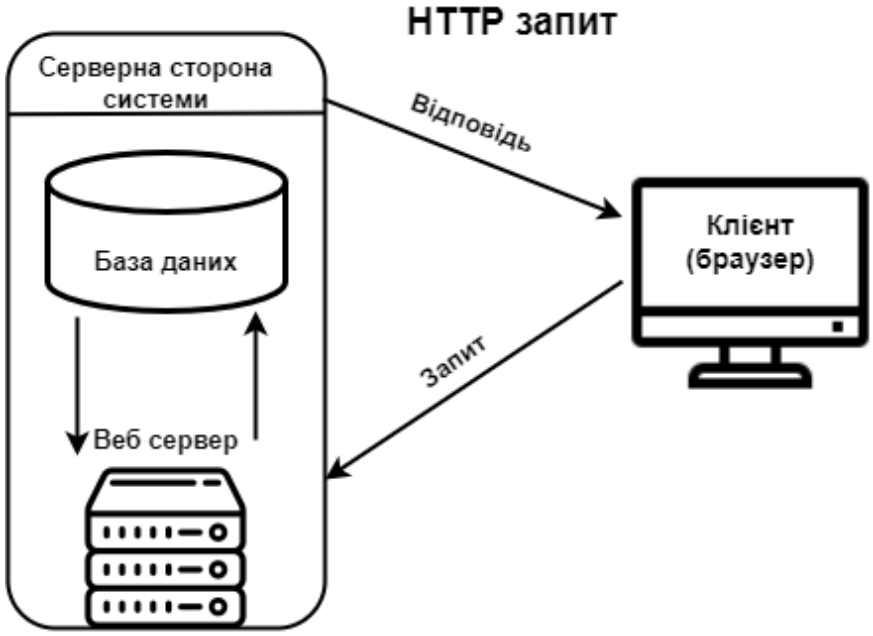


Рисунок 2.3 – Схема роботи запитів

Хоча окремі запити будуть відбуватися за допомогою HTTP запитів, за для разового використання та отримання даних від сервера.

2.3 Алгоритм роботи

Взаємодію між користувачем та системою управління особистими фінансами зображено у додатку A.Use case діаграма. Дана діаграма ілюструє ключові функції системи, які користувач може виконувати, а також відповідні підфункції, які підтримуються системою. Користувач взаємодіє з основними випадками використання, такими як створення та управління особистим профілем, управління бюджетом, відстеження витрат, планування фінансових цілей, та аналіз і звітність.

Кожен з цих основних випадків використання має свої підвипадки, що деталізують конкретні дії користувача та системи.

Система особистого бюджету передбачає створення особистого профілю рисунок 2.4, що дозволяє користувачеві вводити персональні дані, такі як ім'я, адреса електронної пошти, номер телефону та фінансова інформація. Ця функція забезпечує безпечне збереження інформації у системі, що є основою для подальшого управління фінансами.

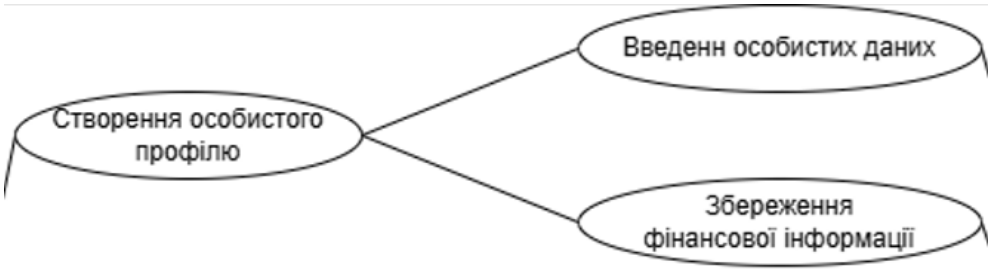


Рисунок 2.4 – Фрагмент зі створення особистого профілю

Управління особистим профілем включає можливість оновлення та редагування користувачем своїх особистих даних і фінансової інформації. Ця функція (рисунок 2.5) забезпечує підтримання актуальності даних, дозволяючи

користувачеві ефективно керувати своїм профілем у системі, що сприяє більш точному відображенню фінансової ситуації .

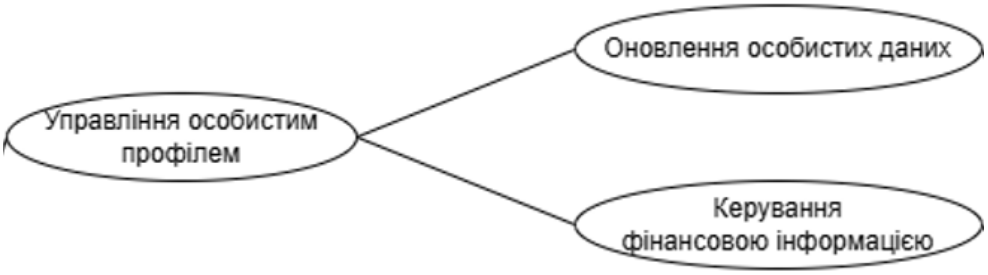


Рисунок 2.5 – Управління особистим профілем

Функція управління бюджетом (рисунок 2.6) дозволяє користувачеві аналізувати доходи та витрати, встановлювати бюджетні ліміти для різних категорій витрат і визначати ключові фінансові цілі. Це забезпечує раціональне використання фінансових ресурсів, підтримання фінансової стабільності та ефективне планування витрат відповідно до пріоритетів користувача.

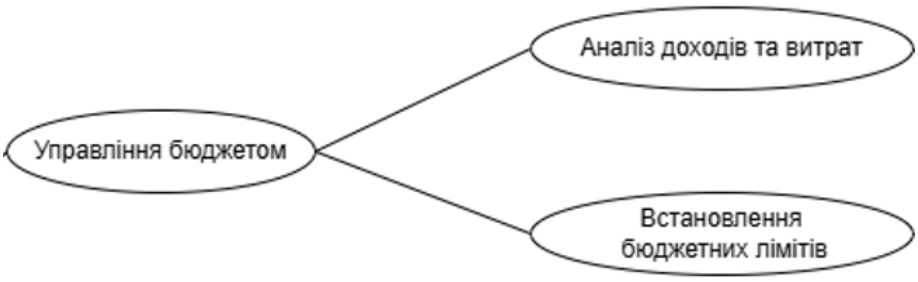


Рисунок 2.6 – Фрагмент управління бюджетом

Відстеження витрат дає змогу користувачеві реєструвати всі свої витрати (рисунок 2.6), що дозволяє отримати детальний огляд фінансової поведінки. Ця функція сприяє аналізу фінансових звичок, виявленню тенденцій у витрачанні коштів і оптимізації витрат, що є ключовим для контролю над особистими фінансами.

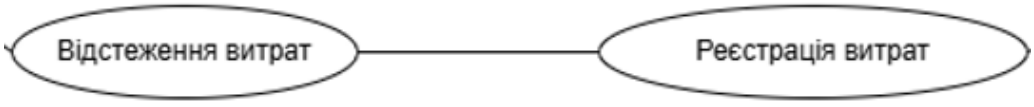


Рисунок 2.7 – Відстеження витрат

Планування фінансових цілей надає користувачеві можливість визначати конкретні фінансові цілі, розробляти стратегії для їх досягнення та встановлювати відповідні ресурси і часові рамки. Це сприяє концентрації зусиль на досягненні фінансових результатів, ефективному управлінню фінансами та досягненню фінансової стабільності (рисунок 2.8).

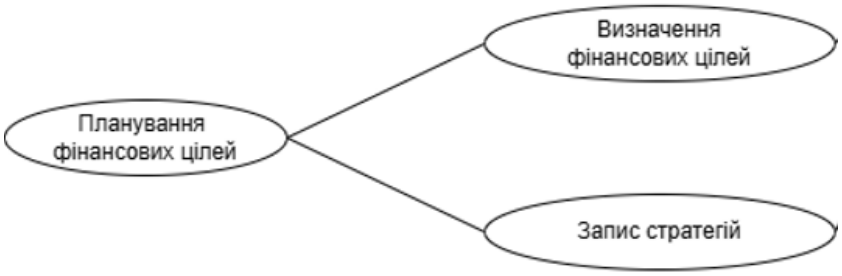


Рисунок 2.8 – Планування фінансових цілей

Функція аналізу та звітності рисунок 2.9 дозволяє користувачеві систематично аналізувати фінансову звітність, використовуючи зведені таблиці, діаграми та графіки. Це забезпечує об'єктивне уявлення про фінансовий стан, допомагаючи користувачеві приймати обґрунтовані фінансові рішення та оптимізувати використання фінансових ресурсів.

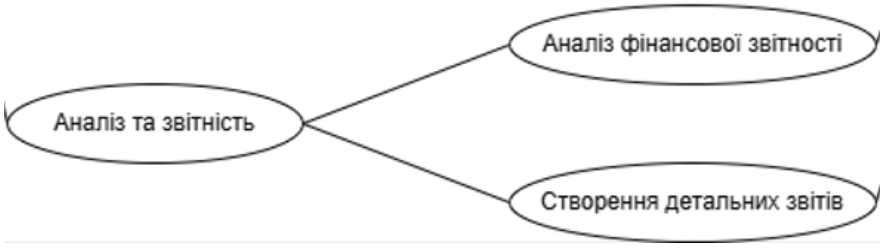


Рисунок 2.9 – Фрагмент управління бюджетом

Таким чином, система особистого бюджету, яка включає створення і управління особистим профілем, управління бюджетом, відстеження витрат, планування фінансових цілей та аналіз і звітність, забезпечує користувачеві комплексний інструмент для ефективного управління своїми фінансами, підвищення фінансової дисципліни та досягнення фінансової стабільності.

2.3- Проектування бази даних

Microsoft SQL Server обирається як система управління базами даних для забезпечення надійного та ефективного зберігання фінансової інформації користувачів. SQL Server володіє великим функціоналом та надійністю, що робить його ідеальним вибором для веб-базованих систем з великим обсягом даних. Використання цієї технології дозволяє створювати складні структури даних, виконувати операції транзакцій, забезпечувати цілісність даних та здійснювати ефективний доступ до інформації.

Крім того, SQL Server має широкий набір інструментів для моніторингу та оптимізації продуктивності бази даних, що дозволяє підтримувати високу швидкодію та доступність системи. Його інтеграція з іншими продуктами та технологіями Microsoft, такими як .NET Framework, забезпечує легку інтеграцію з різноманітними веб-додатками та забезпечує стабільну роботу системи у вимогливих середовищах. Таким чином, використання Microsoft SQL Server у системі контролю особистого бюджету забезпечить надійне та ефективне зберігання фінансових даних користувачів.

Entity Framework обирається як засіб об'єктно-реляційного відображення (ORM) для роботи з базою даних у веб-додатку. Використання ORM дозволяє застосовувати об'єктно-орієнтований підхід до управління даними, що сприяє спрощенню процесів розробки та підтримки системи. Entity Framework автоматизує процеси генерації запитів до бази даних, маніпуляції об'єктами даних та їхнього перетворення до об'єктно-орієнтованої моделі, що підвищує ефективність та продуктивність розробки додатків.

Наступним кроком є створення таблиць бази даних та її проектування.
Таблиця "Користувачі" призначена для зберігання користувачів системи, містить наступні поля

- ID Унікальний ідентифікатор користувача.
- Ім'я користувача.
- Електронна адреса користувача.

- Хешований пароль користувача.

Зв'язок один до багатьох між таблицями "Користувачі" і "Доходи". Кожен користувач може мати багато записів про доходи, але кожен запис про дохід належить тільки одному користувачеві. Це відображається через зовнішній ключ "ID користувача" у таблиці "Доходи", який посилається на поле "ID" у таблиці "Користувачі".

Наступною таблицею є таблиця критерій доходів для зберігання різних категорій доходів, що дозволяє класифікувати джерела доходів.

- ID Унікальний ідентифікатор категорії.
- Назва категорії, наприклад, "Зарплата", "Інвестиції", "Подарунки".

Зв'язок багато до одного між таблицями "Категорії доходів" і "Доходи" де кожен дохід може належати лише одній категорії, але в кожній категорії може бути багато записів про доходи. Це відображається через зовнішній ключ "Категорія ID" у таблиці "Доходи", який посилається на поле "ID" у таблиці "Категорії доходів".
Таблиця доходи призначатиметься для записів про доходи користувачів, та міститиме наступні поля

- ID Унікальний ідентифікатор доходу.
- ID користувача як зовнішній ключ, пов'язаний з користувачем.
- Сума отриманого доходу.
- Дата отримання доходу.
- Зовнішній ключ, пов'язаний з категорією доходів

Міститиме зв'язок один до багатьох між таблицями "Користувачі" і "Витрати" Кожен користувач може мати багато записів про витрати, але кожен запис про витрату належить тільки одному користувачеві. Це відображається через зовнішній ключ "ID користувача" у таблиці "Витрати", який посилається на поле "ID" у таблиці "Користувачі".

Наступною є таблиця способи отримання доходу яка міститиме

- ID як унікальний ідентифікатор способу.
- Назва способу отримання доходу, наприклад, "Банківський переказ", "Готівка", "Електронний платіж" і т.д.

У ній кожен дохід може бути отриманий за допомогою лише одного способу, але кожен спосіб може мати багато записів про доходи. Це відображається через зовнішній ключ "Спосіб ID" у таблиці "Доходи", який посилається на поле "ID" у таблиці "Способи отримання доходу".

Таблиця "Категорії витрат" відповідатиме за зберігання різних категорій витрат, що дозволяє класифікувати витрати.

- ID Унікальний ідентифікатор категорії.
- Назва категорії витрат, наприклад, "Продукти", "Транспорт", "Розваги"

У цій таблиці кожна витрата може належати лише одній категорії, але в кожній категорії може бути багато записів про витрати. Це відображається через зовнішній ключ "Категорія ID" у таблиці "Витрати", який посилається на поле "ID" у таблиці "Категорії витрат".

Таблиця "Витрати" буде створена для зберігання записів про різні витрати користувачів

- ID Унікальний ідентифікатор витрати.
- ID користувача як зовнішній ключ, пов'язаний з користувачем
- Сума витрати.
- Дата витрати.
- Категорія ID зовнішній ключ, пов'язаний з категорією витрат з таблицею "Категорії витрат".
- Спосіб ID зовнішній ключ, пов'язаний зі способом витрати із таблицею "Способи отримання доходу".

У дані таблиці кожен дохід може бути отриманий за допомогою лише одного способу, але кожен спосіб може мати багато записів про доходи. Це відображається через зовнішній ключ "Спосіб ID" у таблиці "Доходи", який посилається на поле "ID" у таблиці "Способи отримання доходу".

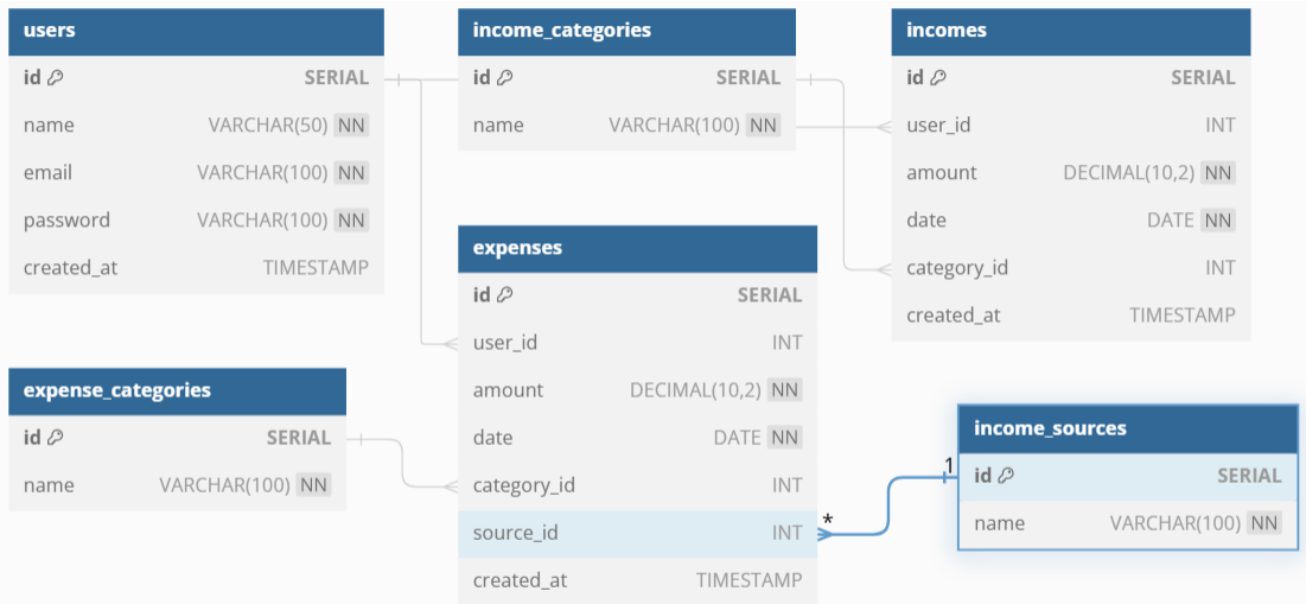


Рисунок 2.10 – Діаграма бази даних

На основі опису структурних компонентів бази даних, було створено діаграму на рисунку (2.10), яка наочно ілюструє взаємозв'язки між таблицями. Діаграма відображає зв'язки типу "один до багатьох" між таблицею користувачів та відповідними таблицями доходів і витрат, а також між категоріями доходів та витрат і їх відповідними записами. Зокрема, таблиці "Доходи" та "Витрати" містять зовнішні ключі, які посилаються на таблицю "Користувачі", що дозволяє однозначно ідентифікувати власника кожного запису. Таблиці "Категорії доходів" та "Категорії витрат" забезпечують класифікацію доходів та витрат відповідно, в той час як таблиця "Способи отримання доходу" містить інформацію про методи, за допомогою яких доходи були отримані.

Дана діаграма структуровано представляє логічну модель бази даних та сприяє легшому розумінню її архітектури та взаємозв'язків між її елементами, дана діаграма зображена

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Реалізація бази даних

Сутності бази даних було реалізовано за допомогою об'єктів передачі даних (Data Transfer Objects, DTO) є важливим аспектом архітектури програмного забезпечення. DTO використовуються для перенесення даних між підсистемами веб-базованої системи, зокрема між рівнями доступу до даних та презентаційним рівнем. Це сприяє зменшенню кількості викликів методів, що дозволяє оптимізувати продуктивність системи. В контексті Entity Framework, DTO допомагають ізолювати доменні моделі від представлення даних, забезпечуючи таким чином чітке розмежування відповідальностей та підвищуючи гнучкість і масштабованість системи.

Також дозволяє знизити обсяг переданих даних, оскільки DTO можуть містити лише необхідні для конкретної операції поля, що зменшує навантаження на мережу та підвищує ефективність обробки даних. Далі представлено ці сутності у наші міграції даних Entity Framework.

```
migrationBuilder.CreateTable(
    name: "Users",
    columns: table => new
        Id = table.Column<int>(type: "INTEGER", nullable: false)
            .Annotation("Sqlite:Autoincrement", true),
        FirstName = table.Column<string>(type: "TEXT", nullable: true),
        LastName = table.Column<string>(type: "TEXT", nullable: true),
        FullName = table.Column<string>(type: "TEXT", nullable: true),
        Email = table.Column<string>(type: "TEXT", nullable: true),
        CurrencyRegionName = table.Column<string>(type: "TEXT", nullable: true),
        UseDarkMode = table.Column<bool>(type: "INTEGER", nullable: false),
        CreatedAt = table.Column<DateTime>(type: "TEXT", nullable: false),
        UpdatedAt = table.Column<DateTime>(type: "TEXT", nullable: false),
        Archived = table.Column<bool>(type: "INTEGER", nullable: false)
    constraints: table =>
        table.PrimaryKey("PK_Users", x => x.Id);
```

Рисунок 3.1 – Фрагмент коду таблиці

Цей фрагмент коду визначає створення таблиці Users (рисун, яка містить такі атрибути: Id, FirstName, LastName, FullName, Email, SystemName, CurrencyRegionName, CreatedAt, UpdatedAt та Archived. Id є первинним ключем і автоматично збільшується для кожного нового запису.

Поля FirstName, LastName, FullName, Email, SystemName, CurrencyRegionName є текстовими і можуть мати порожні значення. Поле UseDarkMode є булевим і не допускає порожніх значень, що дозволяє визначати, чи користувач використовує темний режим. Поля Hash та Salt зберігають хеш і сіль паролів відповідно у вигляді масиву байтів. CreatedAt та UpdatedAt містять дати створення та останнього оновлення запису і також не допускають порожніх значень.

```
migrationBuilder.CreateTable(
    name: "Expenses",
    columns: table => new{
        Id = table.Column<int>(type: "INTEGER", nullable: false)
            .Annotation("Sqlite:Autoincrement", true),
        Date = table.Column<DateTime>(type: "TEXT", nullable: false),
        CategoryId = table.Column<int>(type: "INTEGER", nullable: true),
        TypeId = table.Column<int>(type: "INTEGER", nullable: true),
        Value = table.Column<decimal>(type: "TEXT", nullable: false),
        Comments = table.Column<string>(type: "TEXT", nullable: true),
        UserId = table.Column<int>(type: "INTEGER", nullable: true),},
    constraints: table =>{
        table.PrimaryKey("PK_Expenses", x => x.Id);
        table.ForeignKey(
            name: "FK_Expenses_ExpenseCategories_CategoryId",
            column: x => x.CategoryId,
            principalTable: "ExpenseCategories",
            principalColumn: "Id",
            onDelete: ReferentialAction.Restrict);
        table.ForeignKey(
            name: "FK_Expenses_ExpenseTypes_TypeId",
            column: x => x.TypeId,
            principalTable: "ExpenseTypes",
            principalColumn: "Id",
            onDelete: ReferentialAction.Restrict);
        table.ForeignKey(
            name: "FK_Expenses_Users_UserId",
            column: x => x.UserId,
            principalTable: "Users",
            principalColumn: "Id",
            onDelete: ReferentialAction.Restrict);});
```

Рисунок 3.2 – Фрагмент коду таблиці витрат

Ця частина міграції створює таблицю Expenses, яка включає наступні атрибути: Id, Date, CategoryId, TypeId, Value, Comments, UserId, CreatedAt, UpdatedAt та Archived. Id є первинним ключем і автоматично збільшується для кожного нового запису.

Поле Date зберігає дату витрати і не допускає порожніх значень. Поля CategoryId та TypeId є зовнішніми ключами, що встановлюють зв’язок з таблицями ExpenseCategories та ExpenseTypes відповідно, дозволяючи кожній витраті бути пов’язаною з певною категорією та типом витрат

Поле UserId є зовнішнім ключем, що встановлює зв’язок з таблицею Users, дозволяючи кожній витраті бути пов’язаною з певним користувачем. Поля CreatedAt та UpdatedAt зберігають дати створення та останнього оновлення запису відповідно і не допускають порожніх значень. Поле Archived є булевим, вказуючи на те, чи витрата є архівованою.

Зовнішні ключі CategoryId, TypeId та UserId встановлюють обмеження onDelete, ReferentialAction.Restrict, запобігаючи видаленню пов'язаних записів у відповідних таблицях.

```
migrationBuilder.CreateTable(
    name: "ExpenseCategories",
    columns: table => new
        Id = table.Column<int>(type: "INTEGER", nullable: false)
            .Annotation("Sqlite:Autoincrement", true),
        Name = table.Column<string>(type: "TEXT", nullable: true),
        Description = table.Column<string>(type: "TEXT", nullable: true),
        Budget = table.Column<decimal>(type: "TEXT", nullable: false),
        ColourHex = table.Column<string>(type: "TEXT", nullable: true),
        UserId = table.Column<int>(type: "INTEGER", nullable: true),
        CreatedAt = table.Column<DateTime>(type: "TEXT", nullable: false),
        UpdatedAt = table.Column<DateTime>(type: "TEXT", nullable: false),
    constraints: table =>
        table.PrimaryKey("PK_ExpenseCategories", x => x.Id);
        table.ForeignKey(
            name: "FK_ExpenseCategories_Users_UserId",
            column: x => x.UserId,
            principalTable: "Users",
            principalColumn: "Id",
            onDelete: ReferentialAction.Restrict);
    });
```

Рисунок 3.3 – Фрагмент коду таблиці витрат

Код вище створює коду створює таблицю ExpenseCategories, яка включає наступні атрибути: Id, Name, Description, Budget, ColourHex, UserId, CreatedAt, UpdatedAt та Archived. Id є первинним ключем, який автоматично збільшується для кожного нового запису. Name, Description та ColourHex є текстовими полями, які можуть залишатися порожніми. Поле Budget є десятковим і не допускає порожніх

значень, що дозволяє визначити бюджет для кожної категорії витрат. Поле UserId є зовнішнім ключем, що встановлює зв’язок з таблицею Users, дозволяючи кожній категорії витрат бути пов’язаною з певним користувачем.

Users визначається через зовнішній ключ UserId, із використанням обмеження onDelete, ReferentialAction.Restrict, що обмежує видалення пов'язаних записів у таблиці Users.

Код нижче описує створє індекси які визначають створення таблиць, їх стовпців, первинних і зовнішніх ключів, а також індексів, які покращують продуктивність запитів до бази даних.

```
migrationBuilder.CreateIndex(¶
····name:·"IX_ExpenseCategories_UserId",¶
····table:·"ExpenseCategories",¶
····column:·"UserId");¶
migrationBuilder.CreateIndex(¶
····name:·"IX_Expenses_CategoryId",¶
····table:·"Expenses",¶
····column:·"CategoryId");¶
migrationBuilder.CreateIndex(¶
····name:·"IX_Expenses_TypeId",¶
····table:·"Expenses",¶
····column:·"TypeId");¶
migrationBuilder.CreateIndex(¶
····name:·"IX_Expenses_UserId",¶
····table:·"Expenses",¶
····column:·"UserId");¶
migrationBuilder.CreateIndex(¶
····name:·"IX_ExpenseTypes_UserId",¶
····table:·"ExpenseTypes",¶
····column:·"UserId");¶
migrationBuilder.CreateIndex(¶
····name:·"IX_RefreshTokens_UserId",¶
····table:·"RefreshTokens",¶
····column:·"UserId");¶
```

Рисунок 3.2 – Фрагмент коду зв’язків

У даному проекті використання індексів є ключовим для підвищення продуктивності запитів до бази даних. Створення індексів на певних полях дозволяє суттєво прискорити виконання запитів, особливо при великих обсягах даних. В даному випадку, індекси створюються для полів, які часто використовуються в умовах запитів або при об'єднанні таблиць.

Створення індексу на полі UserId у таблиці ExpenseCategories (міграція створює індекс IX_ExpenseCategories_UserId) значно покращує швидкість виконання запитів, які здійснюють пошук або фільтрацію категорій витрат на основі ідентифікатора користувача.

Загалом, створення цих індексів значно підвищує швидкодію системи, дозволяючи ефективніше здійснювати операції вибірки, фільтрації та об’єднання даних, що є критично важливим

3.2 Реалізація основного функціоналу

Процес успішного відновлення та побудови проєкту ASP.NET Core, який реалізує API, використовуючи командний рядок PowerShell рисунок 3.3.

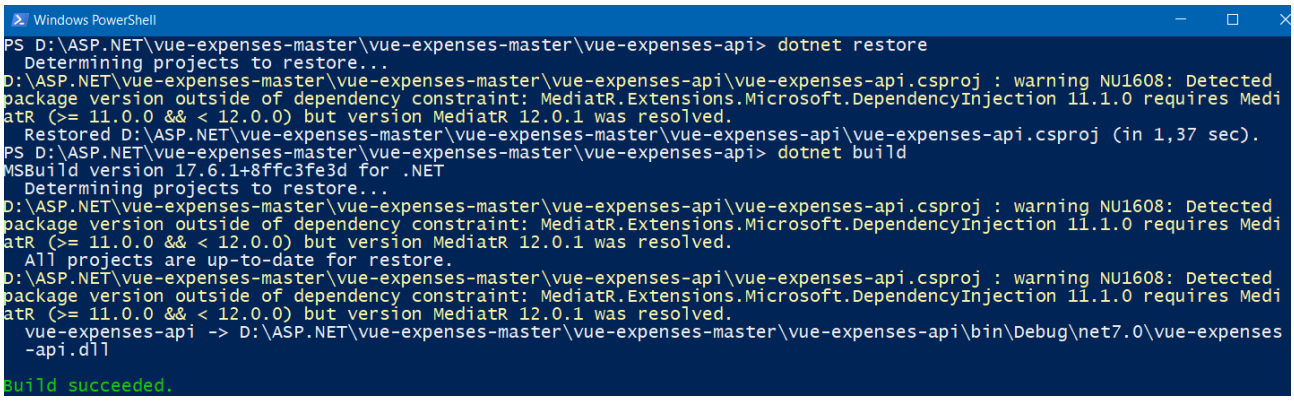


Рисунок 3.3 – Процес побудови серверу

Команда dotnet build ініціювала процес компіляції проєкту, під час якого були знову виявлені ті ж самі попередження, проте ці конфлікти були автоматично вирішені, і процес завершився успішною побудовою. Згідно з результатами, жодних критичних помилок не було виявлено, що підтверджує правильність конфігурації проєкту та його залежностей. Час виконання команд та відсутність помилок вказують на те, що система знаходиться в стабільному стані, готовому до подальших етапів розробки або розгортання.

Запуск сервера ASP.NET Core для проекту, що реалізує API. Першочергово виконуються команди Entity Framework Core для перевірки та міграції бази даних, зокрема запити на наявність та перевірку таблиць і міграцій.

В результаті, система підтверджує, що база даних є актуальною і не потребує додаткових міграцій. Далі, сервер ініціює систему захисту даних, використовуючи

локальний сховище ключів для шифрування конфіденційної інформації. Сервер починає прослуховувати HTTP-запити на портах 5000 і 5001 для HTTP і HTTPS відповідно, рисунок 3.4

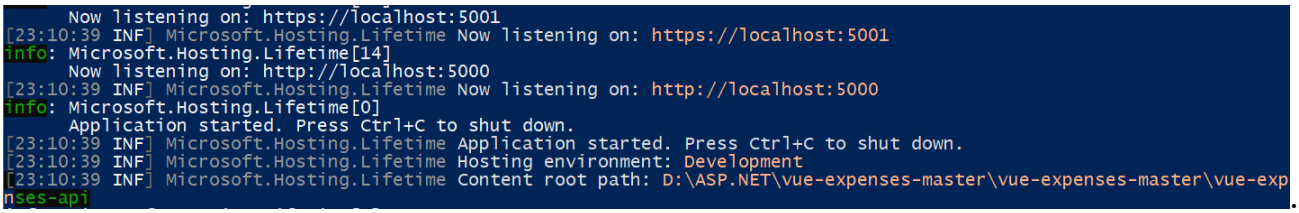


Рисунок 3.4 – Процес запуску серверу

На скріншоті видно успішний запуск головного модуля системи, який скомпільовано без помилок за 115581 мілісекунд. Система працює на локальному сервері за адресою http://localhost:8081 та доступна у мережі за адресою http://192.168.1.101:8081 ,це зображено на рисунку 3.5.

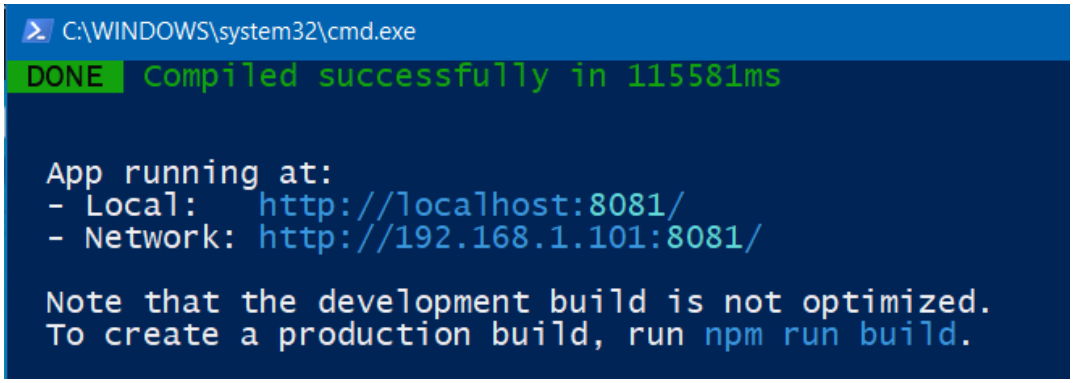


Рисунок 3.5 – Запуск клієнтської частини

На рисунку 3.6 зображено інтерфейс системи для управління особистим бюджетом, де користувач переглядає поточні витрати, створює нову категорію витрат

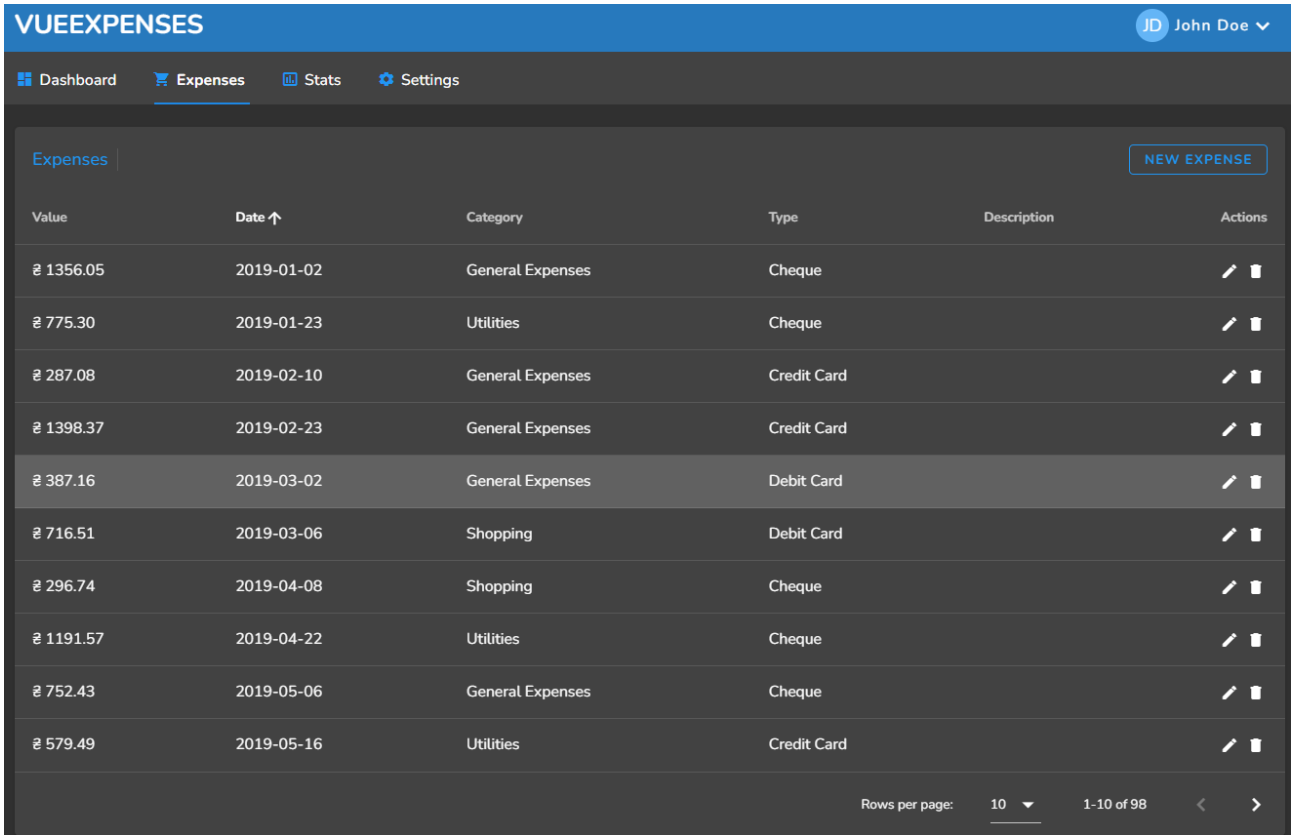
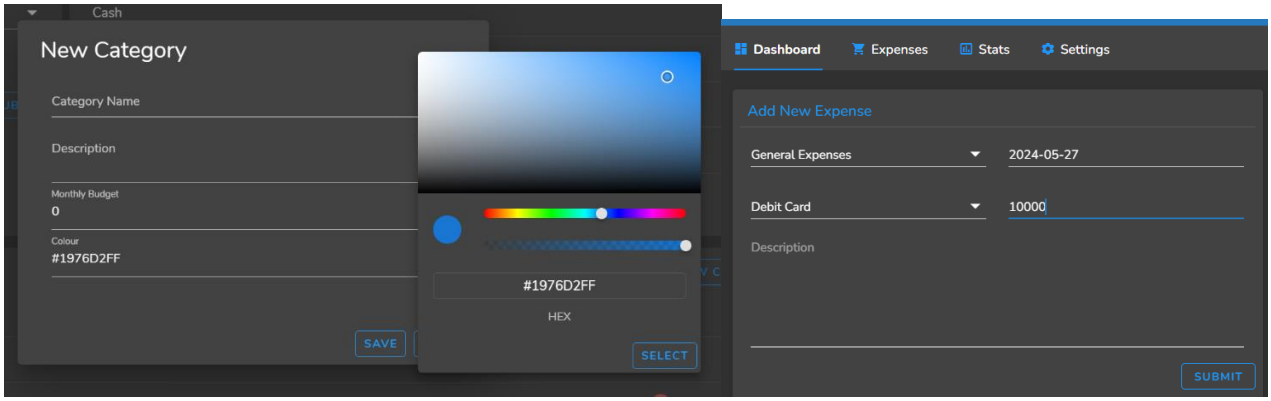


Рисунок 3.6 – Перегляд наявних витрат

Рисунок 3.7 а) показує створення користувачем нового запису про витрату. У формі рисунок 3.7 б) для введення витрат користувач обирає категорію витрат зі списку, задає дату витрати за допомогою календаря або відповідного поля, вводить суму витрати у валюті, яку підтримує система, та вказує рахунок, з якого було здійснено платіж. Інтерфейс є інтуїтивно зрозумілим і забезпечує зручність для користувача, дозволяючи легко додати нову витрату з точними деталями.



а) б)

Рисунок 3.7 – а) Створення нової категорії витрат; б) Додавання нових витрат

Поля для введення мають чіткі підписи, а випадаючі списки допомагають уникнути помилок при виборі категорій і рахунків. Загалом, цей екран забезпечує користувачу ефективний спосіб ведення особистих фінансових записів, підвищуючи точність і зручність управління бюджетом.

Наступними є аналітичні сторінки системи для управління особистим бюджетом, яка містить декілька графіків для візуалізації витрат рисунок 3.8.

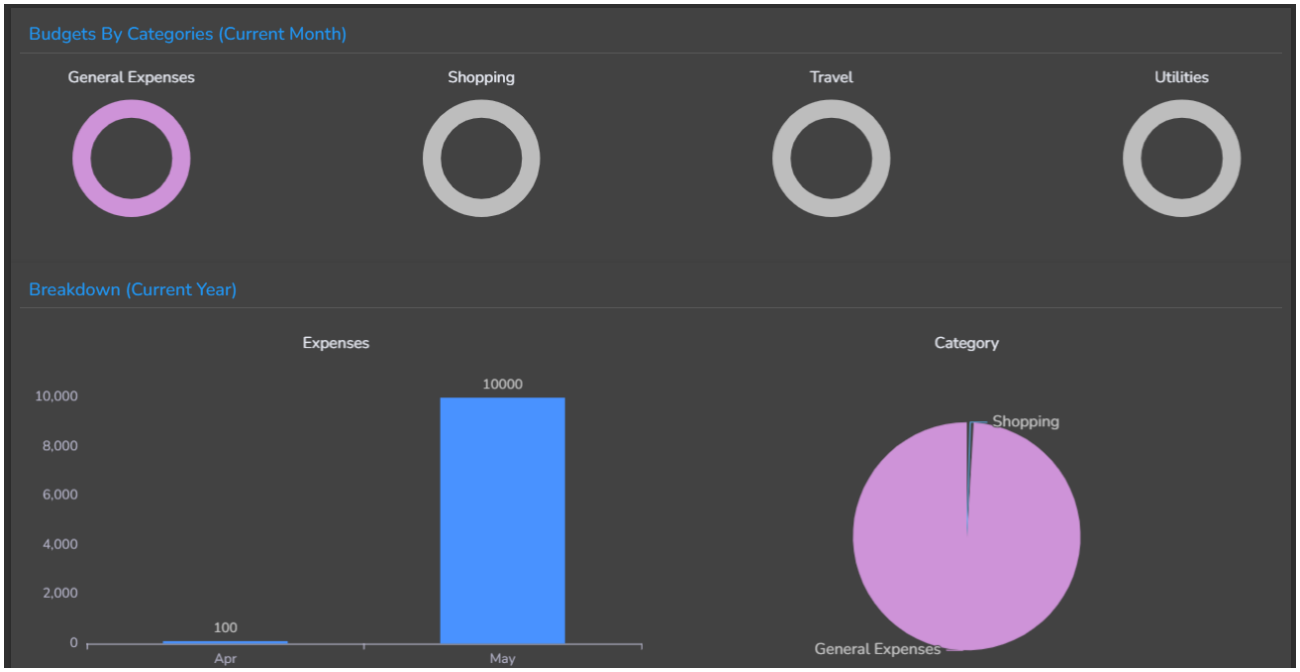


Рисунок 3.8 – Діаграми поточних витрат

Перший графік відображає загальну кількість витрат за певний період, дозволяючи користувачу швидко оцінити обсяги витрат, другий графік є круговою діаграмою, яка ілюструє співвідношення витрат за різними категоріями, допомагаючи візуалізувати, яка частка витрат припадає на кожну категорію.

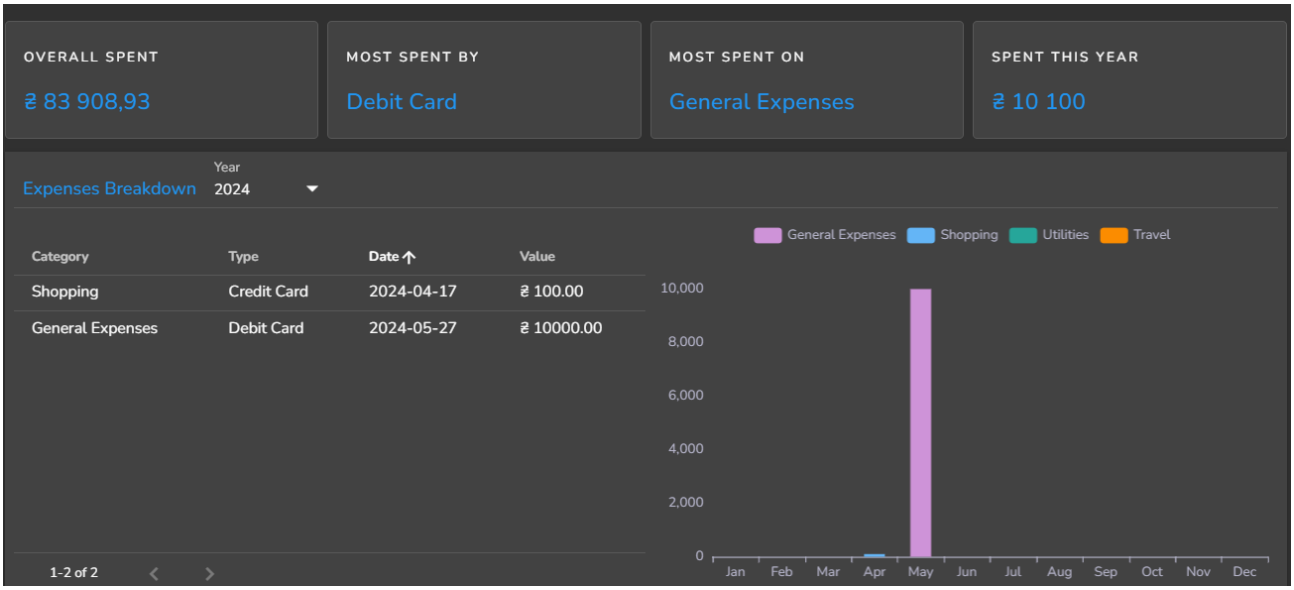


Рисунок 3.9 – Діаграми звітності усіх витрат

Останньою сторінкою є звітність яка демонструє динаміку витрат рисунок 3.9 у часі, відображаючи як кількість, так і суму витрат за різні періоди, що дозволяє користувачу аналізувати тенденції і зміни у фінансовій активності. Інтерфейс графіків забезпечує зручну навігацію та інтерпретацію даних, сприяючи ефективному управлінню фінансами і прийняттю обґрунтованих рішень.

3.3 Тестування системи

Тестування системи зроблене за допомогою інструменту Swagger для тестування API, що значно спрощує процес перевірки і налагодження ендпоїнтів.

У контексті цього проекту Swagger забезпечує інтерактивну документацію, яка дозволяє розробникам легко виконувати різноманітні запити до API, відстежувати відповіді сервера та аналізувати їх. Зручний інтерфейс Swagger підтримує методи HTTP, такі як GET, POST, PUT, DELETE, що дозволяє здійснювати повний спектр операцій з API, включаючи створення, читання, оновлення та видалення даних. Крім того, можливість вводити параметри запиту і

переглядати структуру відповідей у режимі реального часу забезпечує точне тестування та швидке виявлення помилок.

На рисунку 3.10 тестування зображена сторінка Swagger інтерфейсу для вашого API, що забезпечує зручний засіб тестування різних ендпоїнтів.

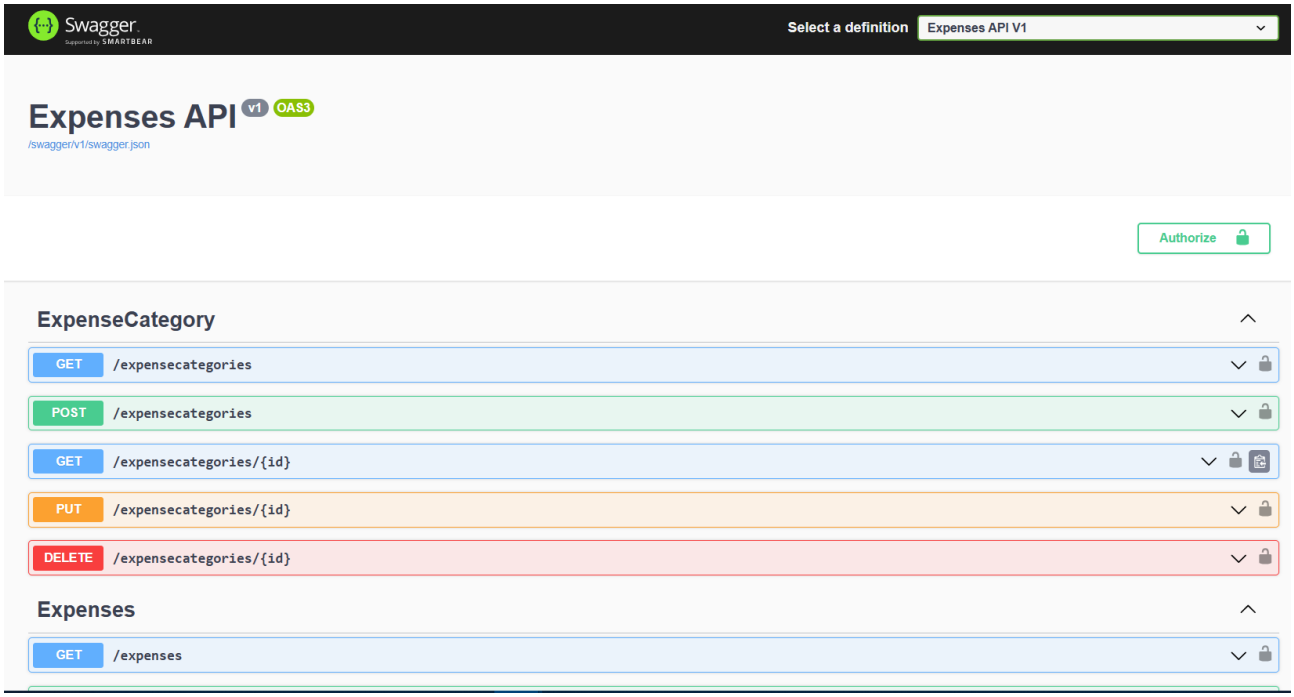


Рисунок 3.10 – Список ендпоїнтів для тестування

Інтерфейс Swagger організовано у вигляді структурованого списку доступних ендпоїнтів, розділених за категоріями або ресурсами. Кожен ендпоїнт супроводжується коротким описом, який пояснює його призначення та метод HTTP-запиту (GET, POST, PUT, DELETE тощо).

Для кожного ендпоїнта можна переглянути деталізовану документацію, включаючи необхідні параметри запиту, очікувані відповіді та можливі статуси відповідей. Користувачі можуть безпосередньо з інтерфейсу Swagger здійснювати запити до API, вводячи необхідні параметри і натискаючи кнопку "Try it out", щоб переглянути результати виконання запиту в реальному часі. Це забезпечує ефективний спосіб тестування і налагодження API, роблячи процес інтерактивним і інтуїтивно зрозумілим.

Виконання методу GET для ендпоїнта Expenses результат якого зображено у рисунку 3.11 Swagger інтерфейсі, що повертає статус-код 200, вказуючи на успішне виконання запиту.

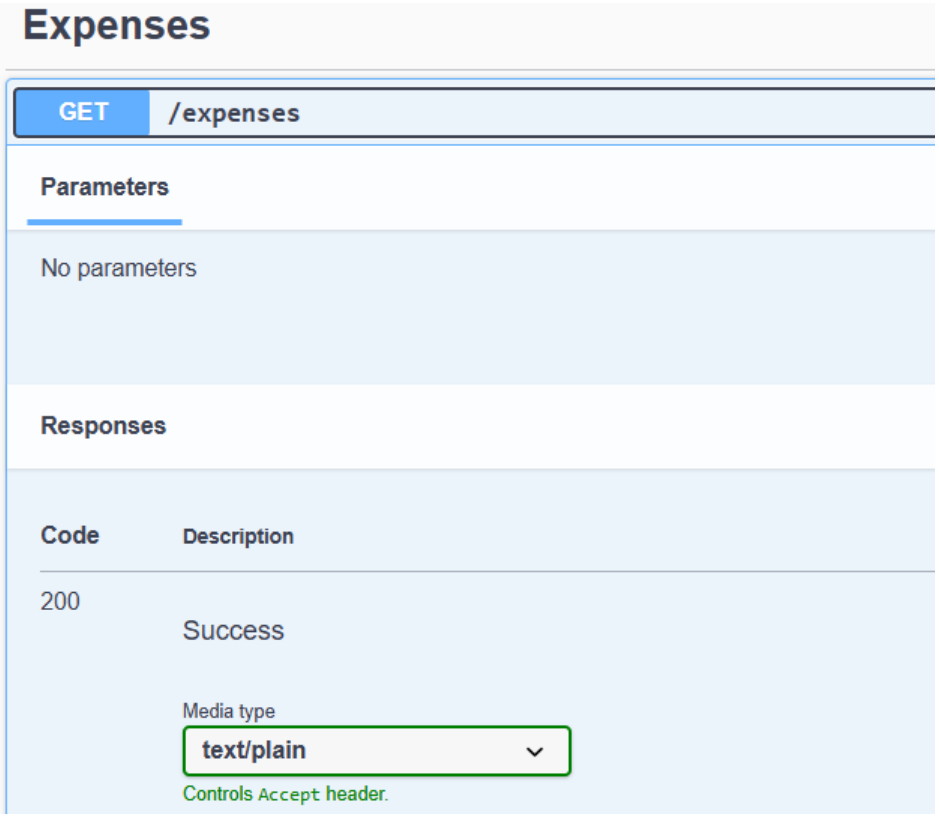


Рисунок 3.12 – Тестування ендпоїнту витрат

Відображається відповідь сервера у форматі JSON, яка містить перелік витрат користувача. Кожен об'єкт у відповіді включає деталі витрат, такі як id, category, amount, date, і account, що дозволяє повністю зрозуміти структуру даних, що повертаються.

Інтерфейс Swagger також показує час виконання запиту та розмір відповіді, що допомагає оцінити продуктивність API. Отримані дані можна переглянути і проаналізувати, що підтверджує правильність функціонування ендпоїнта і відповідність його специфікації. Загалом, результат успішного виконання методу GET для Expenses демонструє стабільність і функціональність API для отримання інформації про витрати.

Наступним є результат виконання методу `PUT` для ендпоїнта Profile у Swagger інтерфейсі, що також повертає статус-код 200, вказуючи на успішне виконання запиту зображено на рисинку 3.13

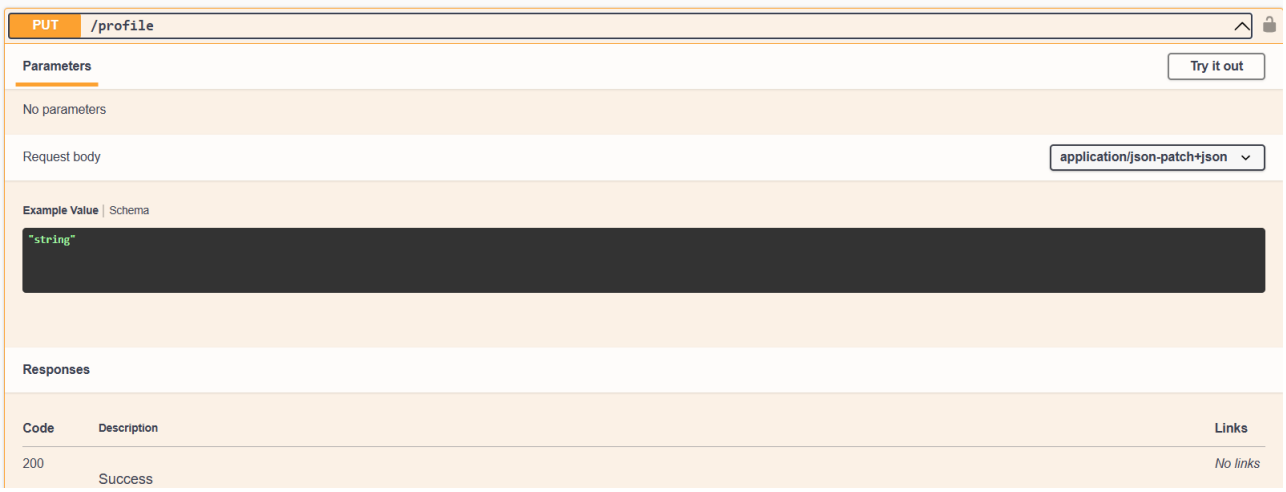


Рисунок 3.13 – Тестування ендпоїнту профіля користувача

Метод PUT використовується для оновлення інформації профілю користувача. Введені дані містять такі поля, як username, email, firstName, lastName, та інші, які були успішно оновлені на сервері. Відповідь сервера підтверджує, що зміни були застосовані, і повертає актуалізовану інформацію профілю або відповідне повідомлення про успіх.

Інтерфейс Swagger показує запит, включаючи відправлені параметри, і надає детальну інформацію про відповідь, зокрема статус запиту і дані, що повертаються. Це підтверджує правильність роботи ендпоїнта та коректне виконання операції оновлення профілю користувача.

У загальному результаті тестування було протестовано 16 основних ендпоїнтів розробленої системи, і всі результати тестування пройшли успішно, що свідчить про повну справність та відповідність цієї системи вимогам.

ВИСНОВКИ

У першому розділі здійснено аналіз ключових аспектів планування особистого бюджету та визначено його важливість у забезпеченні фінансової стабільності та досягненні фінансових цілей. Розглянуто стратегії та інструменти, які сприяють ефективному управлінню фінансами, включаючи аналіз поточної фінансової ситуації, визначення пріоритетних цілей, розробку плану дій та постійне відстеження бюджету. Підкреслено роль планування у зменшенні фінансових стресів і підвищенні фінансової свідомості.

Проаналізовано існуючі аналогічні мобільні додатки, такі як Spendee, Wallet і Pocket Expense, для управління фінансами та бюджетуванням. Виявлено, що хоча багато з цих додатків пропонують корисні функції, деякі з них обмежені платними планами підписки, що може ускладнити їх використання.

У другому описується проєктування системи контролю особистого бюджету, використовуючи модульну та мікросервісну архітектури. Модулі відповідають за управління бюджетом, аналіз витрат, аутентифікацію тощо, а мікросервіси реалізують окремі функції та мають власні бази даних. Архітектурний патерн MVC структурує систему на модель, представлення та контролер. XML обрано для конфігурацій, а Entity Framework - для ORM. Захист даних включає шифрування, аутентифікацію, авторизацію та моніторинг. Інтеграція з веб-сервером і використання веб-сокетів забезпечують стабільний доступ і оновлення в реальному часі.

У третьому розділі розроблена база даних яка оптимізована та зроблено передачу даних між підсистемами, ізолюючи доменні моделі від представлення даних. Таблиці Users, Expenses і ExpenseCategories містять необхідні атрибути та індекси для підвищення продуктивності запитів. Основний функціонал включає успішну компіляцію та запуск проєкту ASP.NET Core, забезпечуючи доступ до API та роботу фронтенду без помилок. Тестування API проводилося через Swagger, який підтримує інтерактивну документацію для HTTP методів, дозволяючи

перевірити коректне виконання ендпоїнтів. Тестування пройдено успішно, що підтверджує стабільність і функціональність системи.

Таким чином, дана розробка має великий потенціал для подальшого розвитку та впровадження у практику. Результати дослідження і тестування підтвердили ефективність розробленого рішення, що забезпечує користувачам зручне та надійне управління особистими фінансами. Важливою перевагою є інтуїтивно зрозумілий інтерфейс та широкий спектр функцій, які дозволяють користувачам легко і ефективно керувати своїми фінансовими ресурсами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Goyal, K., Kumar, S., & Xiao, J. J. (2021). Antecedents and consequences of Personal Financial Management Behavior: A systematic literature review and future research agenda. International Journal of Bank Marketing, 39(7), 1166-1207. https://doi.org/10.1108/IJBM-12-2020-0612
2. Yao, R., Wang, C., & Byrne, S. (2020). Financial literacy and personal financial management in China: Evidence from a web-based survey. International Journal of Consumer Studies, 44(2), 122-137. https://doi.org/10.1111/ijcs.12558
3. Zhang, L., & Zhu, W. (2021). Design and implementation of a web-based personal financial management system using Python. Journal of Computer Science Applications and Research, 12(3), 145-158. https://doi.org/10.1007/s00450-021-00418-3
4. Smith, A. D., & Jones, M. T. (2019). The role of mobile applications in personal finance management: A case study of budget apps. Journal of Financial Services Marketing, 24(1), 43-57. https://doi.org/10.1057/s41264-019-00033-7
5. Anderson, C., & Thomas, D. (2020). Evaluating the effectiveness of online budgeting tools: User engagement and financial outcomes. Journal of Personal Finance, 19(2), 98-112. https://doi.org/10.15370/jpf.2020.19.2.98
6. Lee, J., & Lee, S. H. (2021). Development of a web-based budget management system for university students: A usability study. Journal of Financial Counseling and Planning, 32(1), 75-88. https://doi.org/10.1891/JFCP-20-00020
7. Green, K. P., & Wilson, E. T. (2018). A comparative study of web-based and traditional personal finance management methods. Journal of Financial Planning, 31(4), 112-123. https://doi.org/10.1057/s41312-018-00012-9
8. Chen, Y., & Zhao, Q. (2020). Web-based personal financial management systems: User satisfaction and adoption. Journal of Financial Management and Analysis, 33(2), 130-145. https://doi.org/10.1007/s11301-020-00213-1
9. Park, H. J., & Shin, D. (2021). The impact of online financial education platforms on personal budgeting skills. Journal of Economic Education, 52(3), 194-208. https://doi.org/10.1080/00220485.2021.1897028

10. Williams, R. G., & Patel, N. (2021). Implementing a web-based budget management system in small businesses: A case study. International Journal of Information Management, 61, 102405. https://doi.org/10.1016/j.ijinfomgt.2021.102405

11. Ayoola, E. (2024, February 20). The best budget apps for 2024. NerdWallet. Retrieved from https://www.nerdwallet.com/article/finance/best-budget-apps

12. CreditDonkey. (n.d.). The 9 best online budgeting tools and apps. CreditDonkey. Retrieved from https://www.creditdonkey.com/best-online-budgeting-tools.html

13. Intuit. (2024). Budget tracker & planner | Free online money management | Mint. Mint. Retrieved from https://mint.intuit.com/

14. Marquit, M. (n.d.). Top 9 best (and free) online budgeting tools. Good Financial Cents. Retrieved from https://www.goodfinancialcents.com/best-online-budgeting-tools/

15. PocketGuard. (n.d.). Know your leftover pocket money. PocketGuard. Retrieved from https://pocketguard.com/

16. Number of personal finance management apps worldwide. URL: https://www.statista.com/statistics/1064136/personal-finance-management-apps/ (дата звернення: 16.05.2024)

17. Spendee – Smart budgeting app for managing your finances. URL: https://www.spendee.com/ (дата звернення: 16.05.2024)

18. Wallet: Budget, Expense Tracker. URL: https://play.google.com/store/apps/details?id=com.walletunion.wallet&hl=ru&pli=1 (дата звернення: 16.05.2024)

19. Pocket Expense 6. URL: https://pocket-expense-6.updatestar.com/ (дата звернення: 16.05.2024)

20. Application architecture guide. URL: https://docs.microsoft.com/en-us/azure/architecture/guide/ (дата звернення: 16.05.2024)

21. Fundamentals of algorithms. URL: https://www.geeksforgeeks.org/fundamentals-of-algorithms/ (дата звернення: 16.05.2024)

22. Design patterns. URL: https://www.nngroup.com/topic/design-patterns/ (дата звернення: 16.05.2024)

23. Clean Architecture: A Comprehensive Guide with C#. URL: https://medium.com/@pooja0403keshri/clean-architecture-a-comprehensive-guide-with-c-676beed5bdbb (дата звернення: 11.05.2024)

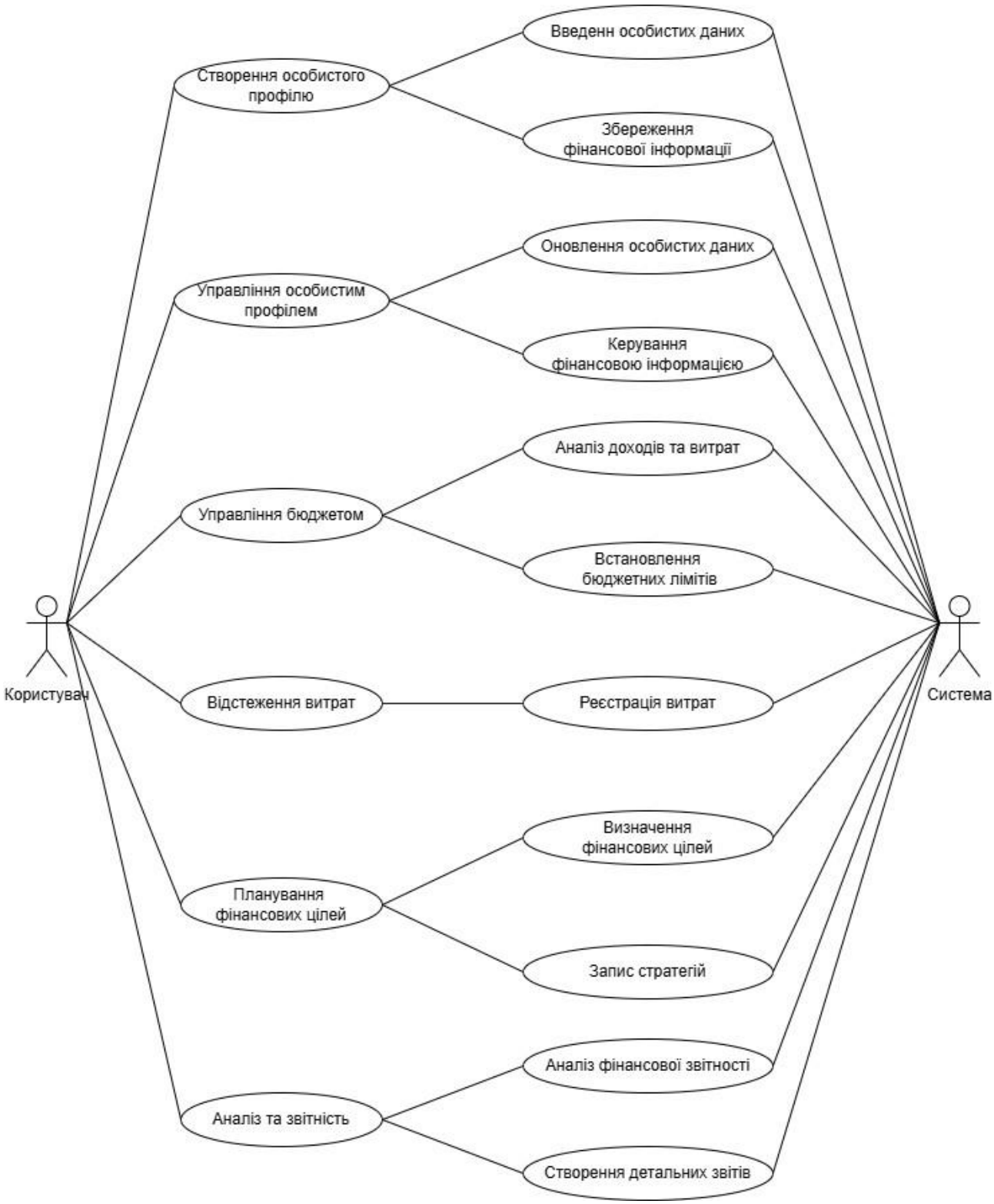
24. C# language documentation. URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/ (дата звернення: 13.04.2024)

25. Entity Framework documentation. URL: https://learn.microsoft.com/uk-ua/ef/ (дата звернення: 21.05.2024)

26. Методичні рекомендації до виконання кваліфікаційної роботи / [Комар М. П., Саченко А. О., Васильків Н. М. та ін.]. – Тернопіль: ЗУНУ, 2024. – 52 с.

ДОДАТОК А

Use case діаграма



ДОДАТОК Б

Діаграма класів

ДОДАТОК В

Апробація отриманих результатів

ЦФЕНД ЦЕНТР ФІНАНСОВО-ЕКОНОМІЧНИХ НАУКОВИХ ДОСЛІДЖЕНЬ

МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ
INTERNATIONAL SCIENTIFIC-PRACTICAL CONFERENCE

ПЕРСПЕКТИВНІ НАПРЯМКИ РОЗВИТКУ ЕКОНОМІКИ, ОБЛІКУ, УПРАВЛІННЯ ТА ПРАВА: ТЕОРІЯ І ПРАКТИКА

PERSPECTIVE DIRECTIONS FOR THE DEVELOPMENT OF ECONOMICS, ACCOUNTING, MANAGEMENT AND LAW: THEORY AND PRACTICE

Збірник тез доповідей
Book of abstracts

18 травня 2024 р.
May 18, 2024

м. Ізмаїл, Україна
Izmail, Ukraine

медіаграмотності серед широкої аудиторії. У корпораціях для моніторингу та захисту своєї репутації.

Розробка програмного модуль для виявлення фейкових новин є актуальним та перспективним напрямком для сфери інформаційної безпеки та машинного навчання. Його впровадження сприятиме покращенню якості інформаційного простору, забезпечуючи об'єктивність та достовірність інформації. Основними користувачами такого програмного модуля можуть бути редакції ЗМІ, медіаорганізації, дослідники, аналітики, а також широка громадськість. Виявлення фейкових новин стане ефективним інструментом для підвищення рівня довіри до інформації та попередження поширення маніпуляційних матеріалів.

Список літератури

1. Tandoc, E. C., Lim, Z. W. & Ling, R. Defining fake news. Digital J. 6, 137–153. https://doi.org/10.1080/21670811.2017.1360143 (2018).

2. Lazer, D. M. J. *et al.* The science of fake news. *Science* 359, 1094–1096. https://doi.org/10.1126/science.aao2998 (2018).

3. Ciampaglia, G. L. Fighting fake news: A role for computational social science in the fight against digital misinformation. *J. Comput. Soc. Sci.* 1, 147–153. https://doi.org/10.1007/s42001-017-0005-6 (2018).

УДК 004.738.5

Гордовський О. А.
студент 4 курсу,
Західноукраїнський національний університет

ОГЛЯД ВЕБ-БАЗОВАНИХ СИСТЕМ УПРАВЛІННЯ ОСОБИСТИМ БЮДЖЕТОМ

Веб-базована система управління особистим бюджетом — це сучасний інструмент, що дозволяє користувачам ефективно контролювати свої фінанси через інтернет. Такі системи включають функції обліку доходів і витрат, аналізу фінансових даних, планування бюджету та прогнозування майбутніх витрат. У сучасному світі, де фінансова грамотність є важливим елементом успішного життя, такі інструменти стають надзвичайно актуальними. Веб-базовані системи допомагають людям легко відстежувати свої витрати, аналізувати фінансові звички та приймати обґрунтовані рішення щодо витрат та заощаджень [1].

Основні функції веб-базованих систем управління особистим бюджетом включають облік доходів та витрат, автоматизацію фінансових операцій, створення звітів та графіків, налаштування фінансових цілей та інтеграцію з

банківськими рахунками. На ринку існує кілька популярних рішень, таких як Mint [2], YNAB (You Need A Budget) [3] та PocketGuard [4].

Mint пропонує зручний інтерфейс та широкий набір інструментів для управління фінансами. YNAB фокусується на методах бюджетування та наданні рекомендацій для поліпшення фінансових звичок. PocketGuard дозволяє користувачам бачити, скільки грошей у них залишилося після необхідних витрат.

Веб-базовані системи управління особистим бюджетом мають кілька значних переваг. По-перше, вони доступні з будь-якого пристрою, підключеного до інтернету, що робить їх зручними для користувачів. По-друге, ці системи використовують сучасні методи шифрування для захисту особистих даних, що забезпечує високий рівень безпеки. По-третє, інтуїтивно зрозумілий інтерфейс та можливість автоматизації багатьох фінансових процесів роблять ці системи дуже зручними. Крім того, веб-базовані системи дозволяють аналізувати фінансову інформацію в реальному часі, отримувати звіти та прогнози [5].

Таким чином, веб-базовані системи управління особистим бюджетом пропонують зручні та ефективні інструменти для контролю та планування фінансів. Вони мають значні переваги, такі як доступність, безпека, зручність та можливість аналізу даних, але також стикаються з певними викликами, включаючи залежність від інтернету та питання конфіденційності. Загалом, впровадження таких систем може суттєво покращити фінансову грамотність користувачів та сприяти більш раціональному використанню фінансових ресурсів. Для подальшого розвитку цих систем важливо зосередитися на підвищенні безпеки даних, розширенні функціональних можливостей та інтеграції з іншими фінансовими сервісами, а також на використанні новітніх технологій, таких як штучний інтелект.

Список літератури

1. Marquit Miranda. Top 9 Best (And Free) Online Budgeting Tools [Електронний ресурс] // Good Financial Cents. – 2021. – Режим доступу: https://www.goodfinancialcents.com/best-free-online-budgeting-tools/

2. Mint. Mint: Money Manager, Budget & Personal Finance [Електронний ресурс]. – 2024. – Режим доступу: https://www.mint.com

3. Johnson, D. The Importance of Personal Finance Management in the Digital Age // Journal of Financial Planning. – 2023. – Т. 34, №2. – С. 45-57.

4. PocketGuard. PocketGuard: Personal Finance, Money & Budgeting App [Електронний ресурс]. – 2024. – Режим доступу: https://www.pocketguard.com

5. Smith, A., Brown, L. Security Concerns in Web-Based Financial Applications // Cybersecurity Review. – 2022. – Т. 18, №4. – С. 89-104.

ДОДАТОК Г

Програмний код початкової конфігурації системи

```
using System.Collections.Generic;
using System.Data;
using System.Net.NetworkInformation;
using System.Text.Json.Serialization;
using FluentValidation;
using MediatR;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Data.Sqlite;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.OpenApi.Models;
using vue_expenses_api.Infrastructure;
using vue_expenses_api.Infrastructure.Security;
using vue_expenses_api.Infrastructure.Validation;

namespace vue_expenses_api;

public class Startup
{
    public Startup(
        IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    public void ConfigureServices(
        IServiceCollection services)
    {
        services.AddControllers()
            .AddNewtonsoftJson();

        services.AddMediatR(cfg => cfg.RegisterServicesFromAssembly(typeof(Startup).Assembly));

        //hook up validation into MediatR pipeline
        services.AddTransient(
            typeof(IPipelineBehavior<,>),
            typeof(ValidationPipelineBehavior<,>));
        services.AddTransient<IDbConnection>(
            db => new SqliteConnection(
                Configuration.GetConnectionString("DefaultConnection")));

        services
            .AddEntityFrameworkSqlite()
            .AddDbContext<ExpensesContext>();

        //Hook up swagger
        services.AddSwaggerGen(
            x =>
            {
                x.AddSecurityDefinition(
                    "Bearer",
                    new OpenApiSecurityScheme
                    {
                        Description =
```

```
                        "JWT Authorization header using the Bearer scheme. \r\n\r\n Enter
'Bearer' [space] and then your token in the text input below.\r\n\r\nExample: \"Bearer
12345abcdef\"",
                    Name = "Authorization",
                    In = ParameterLocation.Header,
                    Type = SecuritySchemeType.ApiKey,
                    Scheme = "bearer"
                });

            var requirement = new OpenApiSecurityRequirement
            {
                {
                    new OpenApiSecurityScheme
                    {
                        Reference = new OpenApiReference
                        {
                            Id = "Bearer", //The name of the previously defined security scheme.
                            Type = ReferenceType.SecurityScheme
                        }
                    },
                    new List<string>()
                }
            };
            x.AddSecurityRequirement(
                requirement);
            x.SwaggerDoc(
                "v1",
                new OpenApiInfo { Title = "Expenses API", Version = "v1" });
            x.CustomSchemaIds(y => y.FullName);
            x.DocInclusionPredicate(
                (
                    version,
                    apiDescription) => true);
        });

        //attach the the model validator and define the api grouping convention
        //setup fluent validation for the running assembly
        services.AddMvc(
                options =>
                {
                    options.Filters.Add<ValidateModelFilter>();
                    options.Conventions.Add(new GroupByApiRootConvention());
                })
            .AddJsonOptions(
                opt => { opt.JsonSerializerOptions.DefaultIgnoreCondition =
JsonIgnoreCondition.WhenWritingNull; });
        services.AddValidatorsFromAssemblyContaining<Startup>();
        services.Configure<JwtSettings>(Configuration.GetSection(nameof(JwtSettings)));

services.Configure<PasswordHasherSettings>(Configuration.GetSection(nameof(PasswordHasherSettings)))
;

        services.AddScoped<IPasswordHasher, PasswordHasher>();
        services.AddScoped<IJwtTokenGenerator, JwtTokenGenerator>();
        services.AddScoped<ICurrentUser, CurrentUser>();
        services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();

        services.AddCors();
        services.AddJwt();
    }

    public void Configure(
        IApplicationBuilder app,
        IWebHostEnvironment env,
        ILoggerFactory loggerFactory)
    {
        loggerFactory.AddSerilogLogging();
        app.UseMiddlewares();
```

```
        if (!env.IsDevelopment())
        {
            app.UseHsts();
            app.UseHttpsRedirection();
        }

        app.UseCors(
            builder =>
                builder
                    .AllowAnyOrigin()
                    .AllowAnyHeader()
                    .AllowAnyMethod()
                    .WithExposedHeaders("Token-Expired"));

        app.UseSwagger(c => { c.RouteTemplate = "swagger/{documentName}/swagger.json"; });

        app.UseSwaggerUI(
            x =>
            {
                x.SwaggerEndpoint(
                    "/swagger/v1/swagger.json",
                    "Expenses API V1");
            });

        app.UseRouting();
        app.UseAuthorization();
        app.UseEndpoints(
            endpoints =>
            {
                endpoints.MapControllers();
                endpoints.MapFallbackToController(
                    "Index",
                    "Home");
            });

        app.UseDefaultFiles();
        app.UseStaticFiles();
    }
}
```