ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**РОЗРОБКА ВЕБ ДОДАТКУ НА ПЛАТФОРМІ .NET CORE ДЛЯ ДЕТАЛЬНОГО КОНТРОЛЮ ФІНАНСОВИХ ВИТРАТ ДЛЯ БАНКІНГУ MONOBANK. »**

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ _Шевченко Дмитро___
(підпис)

Виконав: здобувач вищої освіти гр. ІСДМ-62
Дмитро ШЕВЧЕНКО

Керівник: Каміла СТОРЧАК
науковий ступінь, вчене звання: Доктор технічних наук, професор

Рецензент: Ольга ЗІНЧЕНКО
науковий ступінь, вчене звання: Доктор технічних наук, доцент

КИЇВ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

Ступінь вищої освіти - магістр

Спеціальність 126 Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС
_____ Каміла СТОРЧАК
«____» _____ 20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Шевченко Дмитро Геннадійович
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи: « Розробка веб додатку на платформі .NET Core для детального контролю фінансових витрат для банкінгу Monobank » ___
Керівник роботи: К.П. Сторчак, завідуючий кафедри ІПЗ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом вищого навчального закладу від «12» жовтня 2023 року №122.

2. Строк подання студентом роботи «30 » грудня 2023 року

3. Вхідні дані до роботи
Науково-технічна література з питань, пов'язаних з програмним забезпеченням щодо розробки та проведення математичних досліджень

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
4.1 Теоретичні аспекти в розробці веб-додатків на платформі .Net CORE.
4.2 Вибір складових при підгтовці до створення додатку.

4.3 Проектування та розробка додатку.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Титульний слайд
2. Outlay App
3. .NET 8
4. DDD/MediatR
5. Queries
6. Commands
7. EF Core
8. Automapper
9. Quartz
10. Azure KeyVault
11. Історія транзакцій
12. Сторінка з затратами і прибутками згрупованими по імені (15 днів)
13. Вибір діапазону часу
14. Сторінка з затратами і прибутками згрупованими по імені (1 місяць)
15. Статистика останніх витрат/прибутків по конкретному імені транзакції
16. Сторінка свагеру

6. Дата видачі завдання «14» жовтня 2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Вивчення теми магістерської роботи	14.10.2023	Виконано
2	Вивчення літературних джерел	17.10.2023	Виконано
3	Складання плану роботи	22.10.2023	Виконано
4	Узгодження плану роботи та списку використаних джерел з науковим керівником	01.11.2023	Виконано
5	Аналіз існуючих математичних рішень	03.11.2023	Виконано
6	Дослідження інформаційних технологій	07.11.2023	Виконано
7	Розробка модифікованої математичної моделі та проведення дослідження	13.11.2023	Виконано
8	Вступ, висновки, реферат	03.12.2023	Виконано

9	Розробка обов‘язкових демонстраційних матеріалів	12.12.2023	Виконано
10	Попередній захист роботи	22.12.2023	Виконано
11	Здача роботи	18.01.2024	

Студент Шевченко Д. Г.

(підпис) (прізвище та ініціали)

Керівник роботи К.П. Сторчак

(підпис)

(прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 54 с., 1 рис.., 8 табл. , 32 джерел.

Об'єкт дослідження – оптимальні шляхи розробки веб-додатку.

Предмет дослідження – забезпечення безпеки та ефективності фінансового контролю

Мета роботи – розробка веб-додатку та реалізація на платформі .NET Core, призначена для детального контролю фінансових витрат клієнтів Monobank.

Результат проведеного наукового дослідження дає розуміння, того що завдяки різним платформама та складовим можливо розробити ефективний додаток

Практичні значення отриманих результатів полягає в тому, що на основі проведених досліджень було розроблено веб-додаток на платформі .NET Core, що буде використаний для детального контролю фінансових витрат для банкінгу Monobank

ABSTRACT

The text part of the bachelor thesis is 54 pp., 1 fig.., 8 tables. , 32 sources.

The object of research is optimal ways of developing a web application.

The subject of the study is ensuring the safety and effectiveness of financial control

The purpose of the work is the development and implementation of a web application on the .NET Core platform, designed for detailed control of financial expenses of Monobank clients.

The result of the conducted scientific research gives an understanding that thanks to various platforms and components it is possible to develop an effective application

The practical significance of the obtained results is that, based on the conducted research, a web application was developed on the .NET Core platform, which will be used for detailed control of financial expenses for Monobank banking

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШІ	Штучний інтелект
VB	Visual Basic
DDD	Domain-Driven Design
ORM	Object-Relational Mapper
FIPS	Федеральним стандартам обробки інформації
EF Core	Entity Framework Core

ВСТУП

З постійним розвитком фінансової індустрії та швидким розвитком технологій стало дуже важливо мати можливість ефективно та зручно контролювати власні фінанси. Одним із ключових інструментів цього контролю може стати веб-додаток, що забезпечуватиме зручний та надійний інтерфейс для аналізу та управління фінансовими витратами.

Ця робота присвячена розробці веб-додатку на платформі .NET Core спеціально для детального контролю фінансових витрат за допомогою банківських послуг Monobank. Як інноваційний банк Monobank надає широкий спектр фінансових послуг, але ефективне управління витратами залишається важливим завданням для користувачів. Вибираючи платформу .NET Core для розробки додатків, важливо враховувати її переваги у високопродуктивних і масштабованих веб-додатках, а також чудову сумісність з багатьма іншими технологіями.

Основні завдання роботи включають розробку архітектури програми, створення ефективної бази даних для зберігання даних фінансової інформації, інтеграцію з Monobank API для автоматичного отримання фінансових транзакцій, розробку інтуїтивно зрозумілого та зручного інтерфейсу користувача. Процес розробки включатиме вибір і використання сучасних методів машинного навчання для підвищення точності аналізу фінансових даних і забезпечення високоякісних прогнозів.

Розроблений веб-додаток має стати не лише потужним інструментом для аналізу та контролю фінансових витрат, а й надати нові можливості для розуміння та оптимізації фінансового становища користувача з точки зору банківського досвіду.

Мета роботи: розробка веб-додатку та реалізація на платформі .NET Core, призначена для детального контролю фінансових витрат клієнтів Monobank.

Об’єкт дослідження: оптимальні шляхи розробки веб-додатку.

Предмет дослідження: забезпечення безпеки та ефективності фінансового контролю.

РОЗДІЛ 1: ТЕОРЕТИЧНІ АСПЕКТИ В РОЗРОБЦІ ВЕБ-ДОДАТКІВ НА ПЛАТФОРМІ .NET CORE

1.1 Огляд платформи .NET Core

1.1.1 Історія та значення платформи

.NET Core - це кросплатформна платформа для розробників. Корпорація Майкрософт тривалий час експериментувала з платформою .NET, що, у кунцевому рахунку, призвело до появи різноманітних фреймворків розробки програмного забезпечення. На даний момент існує багато фреймворків у сімействі .NET, включаючи .NET Framework, ASP.NET, .NET Core і .NET Standard.

Коротка історія ядра Dot Net:

Microsoft спочатку випустила .NET Framework, фреймворк, який був спеціально розроблений для роботи з операційною системою Windows. Пізніше компанія усвідомила цінність підтримки на різних платформах і створила .NET Core, який базувався на .NET Framework і мав підтримку на різних платформах. Однак .NET Core також зазнала значних змін після випуску .NET Core 3.1. Компанія відмовилася від бренду «Core» і випустила версію 4.x. Після цього випуску .NET Core було перейменовано в .NET, що вказує на випуск .NET 5. і вище. Microsoft внесла цю зміну, а потім хотіла, щоб .NET була основною платформою для її продуктів .NET.

Крім того, Dot Net Core (.NET Core) є відкритою, універсальною та безкоштовною реалізацією .NET Core. .NET Core доступний для використання на Windows, macOS і Linux.

Це універсальна платформа, яку можна використовувати для створення різноманітних програм, у тому числі великих мобільних програм, настільних програм, ігор, хмарних програм, програм Інтернету речей і веб-програм. На наступному малюнку показано всю платформу Dot NET.

Основними ж характеристиками платформи є:

1)Open-source Framework: .NET Core — це загальнодоступна платформа на основі вихідного коду. Код є доступним на GitHub і постійно активно вдосконалюється та просувається спеціальною спільнотою. Фреймворк захищений авторським правом згідно з ліцензіями Apache 2 та Массачусетського технологічного інституту.

2) Інструменти інтерфейсу командного рядка: .NET Core містить кілька інструментів CLI, призначених для допомоги розробникам у процесі розробки та постійної інтеграції.

3)Кілька мов програмування: за допомогою .NET розробники можуть створювати програми кількома мовами. Розробники можуть використовувати F#, C# або VB (Visual Basic) для створення різних програм. Модульна архітектура. .NET Core має модульну конструкцію, якою можна керувати за допомогою менеджера пакетів NuGet. Пакети NuGet полегшують додавання додаткових функцій до базового проекту Dot Net. Крім того, бібліотека .NET Core вважається пакетом NuGet.

4)Гнучкість розгортання. Для розгортання програми Dot Net доступні різні методи залежно від способу доступу до неї та її використання. Це дозволить вам поширити вашу програму по всій системі та базі користувачів. Чи це також стосується програми .NET

5)Кілька мов програмування: за допомогою .NET розробники можуть створювати програми кількома мовами. Розробники можуть використовувати F#, C# або VB (Visual Basic) . Модульна архітектура. .NET Core має модульну конструкцію, якою можна керувати за допомогою менеджера пакетів NuGet. Пакети NuGet полегшують додавання додаткових функцій до базового проекту Dot Net. Крім того, бібліотека .NET Core вважається пакетом NuGet.

6) Підтримується кілька архітектур: Dot NET Core може працювати на кількох процесорах, включаючи Arm64, x86 і x64.

7)Загальні підтримка: існує виняткова підтримка .NET Core від Microsoft і спільноти, яка просуває та вдосконалює структуру.

1.1.2 Основні переваги та недоліки, можливості та особливості

Особливості .NET Core:

.NET Core пропонує багато функцій lkz створювати різні типи програм.. З можливого вибрати розробку ядра dotnet для створення програм із функціями нового покоління

- LINQ (мовний інтегрований запит)

Це набір технологій, які залежать від можливості надсилати запити до бази даних у C#. Загалом це технологія, , яка дозволяє запитувати дані в C#. Таким чином, LINQ у C# — це мовна конструкція, яка нагадує клас і дозволяє розробникам підключатися до різних баз даних і отримувати доступ до різномантіної інформації. API запитує дані в C#, які дозволяють отримати доступ до даних із багатьох різних джерел, включаючи сервери SQL, об’єкти, XML тощо.

- Фільтри

Однією з основних функцій Dotnet Core є фільтри. Функція фільтрів використовується з ASP.NET Core для виконання коду до та після певного етапу обробки запиту. Ця функція використовується для виконання таких завдань:

1) Автентичність: переконання, що користувачі справжні та мають дозвіл на доступ до ресурсів

2) Кешування відповіді: запит до сервера виконується ранішк строків, щоб отримати кешовану відповідь. Кешування відповідей — це процес збереження інформації, отриманої після запиту, і її використання для наступних запитів.

Також можливо створювати власні фільтри за допомогою вбудованих. Ці спеціальні процесори можна використовувати для кешування, налаштування, обробки помилок тощо. Фільтри допомагають в унікненні дублювання коду.

- Глобалізація та локалізація

Через з глобалізацією та локалізацією основною метою dotnet є створення

культурно-відповідних програм і веб-сайтів. Тобто можливо створювати програми та веб-сайти з певними локалізованими даними, інформацією та вмістом. Ваш додаток або веб-сайт може представляти місцеві традиції та тим самим забезпечувати індивідуальний досвід.

.NET Core надає розробникам платформу для введення своїх програм або веб-сайтів на основі кількох мов. За допомогою ASP.NET ви можете створювати багатомовні веб-сайти та використовувати .NET Core для створення програм, які, в свою чергу, будуть охоплювати більшу аудиторію за допомогою вмісту, який спеціалізується на певному регіоні та має місцевий мовний колорит.

- Пропозиція автономії

Це одна з відмінних рис ядра dotnet і фреймворку ASP.NET Core. Впровадження залежностей (DI) — це модель програмування, яка дотримується принципу інверсії залежностей програмування на основі класів. По суті, DI передбачає створення класів, які залежать від інтерфейсів або абстрактних класів, а не їх пряму реалізацію.

Оскільки ASP.NET Core підтримує DI, можливо створювати незалежні один від одного класи, які спрощують код. За допомогою ін’єкції залежностей існує можливість створювати слабко пов’язані програми, які все ще мають високу гнучкість і легкість у розробці.

.NET є однією з найпопулярніших технологій для розробки програмного забезпечення на сьогодні, вона є гнучкою та має велику спільноту, яка її підтримує. Використання технології Dot Net для спеціальних програм і розробки програмного забезпечення має численні переваги. У цьому полягає цінність використання .NET Core для створення проектів програмного забезпечення широкого застосування.

Переваги .NET Core:

- Легке встановлення з будь-якого джерела

Через кросплатформну природу Dot Net існує можливість створювати програмне забезпечення, яке не залежить від використовуваної платформи. Незалежно від

того, чи використовуєте ви Linux , Windows чи macOS, ви можете легко створювати потужне програмне забезпечення, використовуючи фреймворк .NET Core.

- Єдина універсальна кодова база для кросплатформних програм.

Найвигіднішим аспектом використання .NET Core для розробки мобільних додатків є чистий базовий код, який можливо використовувати на різних платформах. Це дозволить вам створювати власні програми як для Android, так і для iOS, а написаний вами код буде ідентичним для обох.

Крім того, існує спільна мова програмування, яка використовується для створення програм для різних платформ. C# в основному використовується для розробки .NET, однак також використовуються інші мови, такі як F# і Visual Basic.

- Простий спосіб доступу до бази даних.

Підключення до бази даних і, в наслідок цього, отримання інформації є поширеним на різних веб-сайтах і в додатках. За допомогою складного коду можна спостерігати за використанням різних запитів до бази даних з метою отримання даних для вашого веб-сайту чи програми.

.NET Entity Framework Core розробив спосіб виконання запитів до бази даних. Entity Framework Core — це ORM, який полегшує зв’язок між програмуванням і базами даних. Це полегшує використання баз даних через об’єкти .NET. Entity Framework Core підтримує кілька баз даних, включаючи PostgreSQL, MySQL, SQLite тощо. Це більш вигідно для продуктивності запитів, відображення та міграції.

- Взаємодія інших технологій

Інтероперабельність .NET є вигідною, вона дозволяє додавати додатковий код до середовища Dot Net, що виходить з його меж. Це означає, що ви можете комбінувати код, який сумісний з іншою технологією чи платформою. У Dot Net код, який виконується за допомогою CLR, вважається керованим, а код, який не виконується за допомогою CLR, - некерованим.

Некерований код складається з компонентів ActiveX, компонентів C++, Microsoft Windows API, Com і Com+. У мережі, яка має точки, які не обслуговуються, взаємодія з некерованим кодом здійснюється за допомогою найбільшої кількості технологій. Наприклад, вам потрібна взаємодія COM (взаємодія COM), щоб регулювати сумісність коду на основі COM.

Служби викликів платформи та простори імен System.Runtime.InteropServices призначені для використання функцій, зворотних викликів і структурних компонентів із некерованих бібліотек коду, якими керують. Крім того, взаємодія з C++ дає змогу використовувати некерований код на основі мови C++.

Недоліки .NET Core:

Як можна помітити, ядро мережі зросло в області програмного забезпечення завдяки численним неймовірним перевагам, які вона надає. Однак створення програмного забезпечення на основі платформи .NET Core також має кілька недоліків. Наприклад:

- Старі бібліотеки не повністю сумісні.

Старі бібліотеки не повністю підтримують .NET Core. Якщо вам потрібно використати будь-яку зі старих бібліотек, які доступні з .NET Framework, це, ймовірно, буде складно з Dot NET Core.

- Неможливість співіснувати з попередніми платформами.

.NET Core не може забезпечити повний ступінь сумісності зі старими платформами, який можна досягти за допомогою .NET Framework.

1.1.3 Використання в розробці веб-додатків та майбутнє платформи

Найкращі веб-сайти та програми, створені на платформі .NET Core:

Оскільки Dot NET Core є кросплатформним, гнучким і багатофункціональним, багато компаній використовують його для розробки різноманітних програм. Є багато брендів, які обирають розробку додатків за точковою мережею для створення своїх програм та корпоративних веб-сайтів . Ось деякі з найкращих

компаній, які використовують цю платформу:

- Stack Overflow

Stack Overflow — це популярна платформа для запитань і відповідей, де програмісти можуть шукати та надавати допомогу. Він був створений з метою допомогти програмістам у всьом світі вирішувати свої проблеми за допомогою інших розробників. Ця платформа підтримується великою спільнотою розробників, включаючи як початківців, так і експертів.

Stack Overflow також працює на .NET Core. Він був розроблений за допомогою ASP.NET MVC і MySQL.

- W3schools.com

W3schools.com — це освітня онлайн-платформа, яка пропонує навчальні матеріали для різних навичок програмування. Це одна з найпопулярніших платформ для програмування освіти з багатьма корисними статтями та доволі зручним інтерфейсом. W3schools також використовує платформу .NET, яка створена шляхом поєднання ASP.NET framework з іншими бібліотеками та інструментами.

- Геокешинг

Геокешинг — є найкращою у світі грою в хованки, що забезпечує неперевершений ігровий досвід. У цій грі учасники повинні ховати та шукати «тайники», які є контейнерами для об’єктів, використовуючи мобільні телефони, GPS або інші методи навігації,

Сервер їхньої ігрової інфраструктури побудовано з використанням .NET, що дозволяє їм легко масштабуватись, додаючи мільйони користувачів.

- Bing

Bing вважається однією з найкращих пошукових систем, розробленою та належить корпорації Microsoft. Його основна структура базується на платформах .NET і .NET Core. Bing було визнано однією з найскладніших, ефективних і великих програм, створених за допомогою технології .NET.

Будучи веб-додатком, Bing використовує модель ASP.NET платформи .NET і пропонує різноманітні моделі пошуку, як-от зображення, картки, продукти тощо. Він є наступником MSN і Live Search, та наразі його вдосконалено ШІ Bing.

Майбутнє .NET Core:

Наразі .NET широко використовується як основа для створення різноманітних рішень. З моменту свого відкриття ця платформа з кожним роком мала все більше шанувальників і шанувальниць. Було випущено кілька адаптацій цього програмного забезпечення, наприклад .NET Framework, .NET Core, ASP.NET, ASP.NET Core та ASP.NET MVC, щоб назвати лише деякі.

Важливо також знати відмінності між парними та непарними ітераціями .NET framework. Ті що з парним числом підтримуються LTS (довгостроковою підтримкою), тоді як ті, що містять непарні цифри, проходять STS (стандартний термін підтримки).

Усі випуски мають однакову якість, за винятком довгострокової підтримки. Версія LTS пропонує безкоштовну підтримку та виправлення протягом більш тривалого періоду(3 роки), тоді як STS надає таку допомогу лише до 18 місяців.

Платформи	STS (стандартна термінова підтримка)	LTS (довгострокова підтримка)
.Net 5	Листопад 2020-Травень2022	
.Net 6	-	Листопад 2021- Листопад 2024
.Net 7	Листопад 2022-Травень2024	
.Net 8	-	Листопад 2023-...
.Net 9	Листопад 2024-….	

Таблиця 1.1 Відмінності STS і LTS та майбутніх випуски.

1.1.4 Порівняння з іншими фреймворками, платформами та мовами

Порівняння на основі популярних веб-фреймворків:

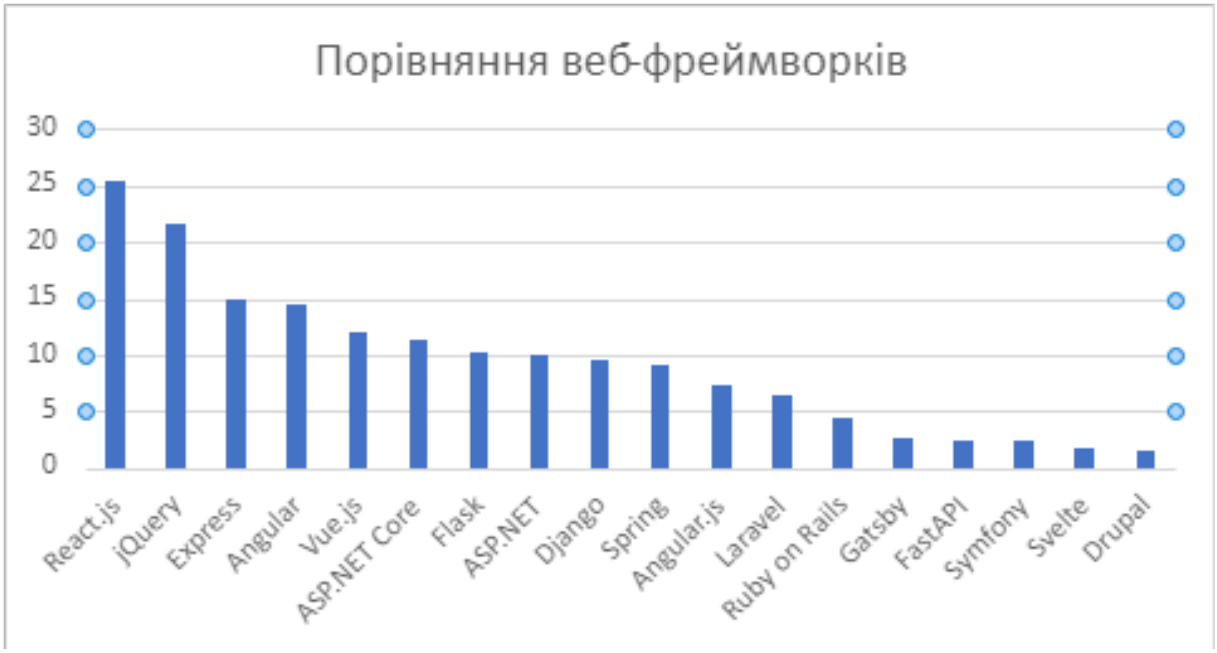


Рисунок 1.1 - Порівняння веб-фреймворків за 2021 рік

Згідно з опитуваннями Stack Overflow, гістограма демонструє, що веб-фреймворки ASP.NET і ASP.NET Core. У 2021 році займали 8 та 6 місце відповідно. [2]

Далі буде здійснено порівняння .NET Core з іншими платформами та мовами

1).NET Core проти Java:

Параметри	**Java**	**.NET Core**
Початковий випуск	1996 рік	2016 рік
Мови	Це сама по собі мова програмування.	C#, F# і VB.NET — це мови для розробки .NET Core.
Тип	Це мова програмування з відкритим кодом.	Це платформа розробника з відкритим кодом.
Час виконання	JRE (середовище виконання Java)	CLR (загальномовне середовище виконання)
Веб-фреймворки	Spring, Grails, Strut	ASP.NET, ASP.NET Core, ASP NET Core MVC
Технологія серверних сценаріїв	Сторінки JavaServer	ASP.NET

Таблиця 1.2 Порівняння Java та .NET Core

2) .NET Core проти PHP

Параметри	**PHP**	**.NET Core**
Початковий випуск	1995 рік	2016 рік
Тип	Серверна сценарна мова з відкритим кодом.	Платформа розробника з відкритим кодом.
Мови	Laravel, CakePHP, Zend, Symfony, CodeIgniter	C#, F#, VB.NET
Веб-фреймворки	Django, Flask, Web2py	ASP.NET, ASP.NET Core, ASP.NET Core MVC
виконання	Метод компільованого коду	Виконання інтерпретованого коду
Налагодження	Легке налагодження	Налагодження складне
Фокус	Краща продуктивність на стороні клієнта та користувальницький інтерфейс.	Налагодження складне

Таблиця 1.3 Порівняння PHP та .NET Core

3) .NET Core проти Node.JS

Параметри	**.NET Core**	**Node.JS**
Початковий випуск	2016 рік	1995 рік
Тип	Платформа розробника з відкритим кодом.	Серверна сценарна мова з відкритим кодом.
Мови	C#, F#, VB.NET	Laravel, CakePHP, Zend, Symfony, CodeIgniter
Налагодження	Налагодження складне	Легке налагодження
виконання	Виконання інтерпретованого коду	Метод компільованого коду
Веб-фреймворки	ASP.NET, ASP.NET Core, ASP.NET Core MVC	Django, Flask, Web2py

Фокус	Налагодження складне	Краща продуктивність на стороні клієнта та користувальницький інтерфейс.

Таблиця 1.4 Порівняння Node.JS та .NET Core

4) .NET Core проти Python

Параметри	**.NET Core**	**Python**
Початковий випуск	2016 рік	1991 рік
Тип	Платформа розробника з відкритим кодом.	Мова програмування з відкритим кодом.
Веб-фреймворки	ASP.NET, ASP.NET Core, ASP.NET Core MVC	Django, Flask, Web2py
Мови	C#, F#, VB.NET	Python — мова програмування високого рівня.
Час виконання	CLR	Сценарії Python виконуються на основі специфічних для платформи бібліотек і інструментів.

Таблиця 1.5 Порівняння Python та .NET Core

1.2 Основні принципи фінансового контролю

Загалом існують певні фундаментальні принципи, яких слід дотримуватись при здійсненні фінансового контролю. Це законність, незалежність, об'єктивність, відповідальність і гласність, а також системність.

1) Принцип законності передбачає неухильне дотримання всіма суб'єктами фінансового контролю та їх посадовими особами встановлених законів і нормативних актів.
2) Принцип незалежності же надає автономну роль, завдяки якій органи фінансового контролю можуть ефективно здійснювати перевірку/ моніторинг без втручання з боку інших державних або місцевих урядових установ (або політичних організацій). Контролери суворо дотримуються

фірмових стандартів, зберігаючи професіоналізм.

3) Принцип відповідальності означає вжиття всіх заходів проти будь-яких потенційних порушень матеріальних кодексів, які ведуть до злочинної діяльності.

4) Принцип гласності означає відкритість та доступність нашого суспільства й засобів масової інформації до відомостей про результати контрольно-ревізійних та експертно-аналітичних заходів, проведених в рамках фінансового контролю за умов збереження комерційної, державної або іншої захищеної законом таємниці.

З цих базових принципів можуть також випливати й інші, що носять більш прикладний характер, тобто результативність, непідкупність суб'єктів контролю, чіткість та логічність вимог, що пред'являються суб'єктами контролю, мається на увазі професіоналізм, принцип дотримання професійної етики, узгодженість дій різних контролюючих органів, та ін

1.3 Огляд банкінгу Monobank, історія та основні послуги, порівняння з Приват24

З розвитком нових технологій оформити кредит, відкрити депозитний чи валютний рахунок, оплатити послуги та товари стає можливим всього в один клік. В Україні вже багато людей користуються послугами мобільного банку monobank, який пропонує клієнтам сучасний сервіс і вигідні фінансові продукти.

Що таке монобанк?

В Україні новаторська ініціатива була запущена наприкінці 2017 року, що стало початком запровадження розрахунків Monobank. Ці віртуальні банківські послуги зробили революцію в традиційних фінансових операціях.

Колишні топ-менеджери ПриватБанку заснували Fintech Band, який запустив Monobank як перший мобільний банк в Україні. Ця кредитна структура працює повністю онлайн без будь-яких фізичних філій чи відділів.

Банківський сервіс monobank надає доступ до фінансових послуг Universal Bank. Його клієнти можуть скористатися різноманітними валютними вкладами, покупками в розстрочку та універсальною карткою. Потенційним користувачам достатньо лише встановити програму на свої гаджети Android або iOS, оскільки всі операції проводяться онлайн через віртуальні рахунки та картки. Для вирішення будь-яких проблем, які виникають під час роботи, цілодобова підтримка є доступною для консультацій зі спеціалістами в реальному часі.

Ідея створити такий банк належить Олегу Горохівському, Михайлу Рогальському та Дмитру Дубілету.

Особливості:

Раніше відкриття банківського рахунку вимагало відвідування відділення, де потрібно було підписати численні паперові документи та дочекатися, поки співробітники зроблять копії. Послуги мобільного банкінгу Monobank пропонують альтернативний процес, за допомогою якого клієнти можуть завантажити свою програму та слідувати підказкам, а саме - сфотографувати всю сторінку свого паспорта (включно з відмітками), а також ідентифікаційний код. Система також приймає нові ідентифікаційні паспорти, які можуть замінити зовнішню фотографію на обох сторінках лише плюс-кодом паспорта. Додаткова документація, як-от заяви про місце чи реєстрацію, не є обов'язковою для цього спрощеного підходу, вирішення усього іншого відбувається автоматично після реєстрації через додаток.

Замість того, щоб йти в банк, ви можете запросити виписку по рахунку через додаток і отримати її поштою за лічені секунди.

Закриття облікового запису також можна здійснити зі смартфона, увійшовши в меню «Налаштування», підтвердивши це в службі підтримки та видаливши додаток.

Мобільний банк має колекцію всіх поширених платіжних сервісів. Він дозволяє розраховуватися за комунальні послуги, поповнювати рахунки собі та іншим, а також переказувати кошти за реквізитами одержувачів або номерами

телефонів. Якщо користувач застосовує власні гроші, комісія не стягується з жодної транзакції. Але при використанні кредитних коштів діють процентні ставки за тарифами самого Monobank.

Для всіх транзакцій існує опція «Скасувати платіж» – протягом 10 секунд будь-який клієнт може натиснути на кнопку і заблокувати переведення коштів.

Картку Monabank можна безкоштовно поповнити різними способами: через термінали iBox і ТАСКОМБАНКУ, переказавши кошти з картки іншого банку в додатку Mono або через касу Universal Bank , це якщо ви є клієнтом. Переказ грошей з картки іншого банку в monobank має мінімальний ліміт - 100 грн (у додатку Mono). Однак поповнення з кіосків банків-партнерів або інших банківських додатків є необмеженим, але може стягувати плату за трансакцію (відбувається стягнення комісії у відсотках).

Додаток також дозволяє встановити PIN-код для додаткової безпеки та знімати кошти без пароля до певного ліміту. Наприклад, якщо встановлено поріг понад 500 грн, користувачеві не потрібно буде вводити PIN-код при покупці на суму до 450 грн. Однак важливо зазначити, що це правило стосується виключно платежів за фіпом, тобто у разі використання безконтактних способів оплати на суму понад 100 грн буде вимагатися введення PIN-коду.

Картка Monobank пропонує своєму власнику кешбек від 1% до 20% залежно від обраних категорій покупок клієнта. На початку кожного місяця клієнти можуть вибрати дві окремі сфери, такі як продукти чи подорожі. Додаткові варіанти включають «Автомобілі та АЗС», «Кафе та ресторани», «Краса та медицина», «Розваги, спорт» або фільми тощо.

Кешбек Monobank зараховується одразу після покупки, але на основний рахунок його можна перерахувати лише після накопичення 100 грн. Ця сума включає комісію 19,5% і може нараховуватися протягом року. Крім того, максимальний місячний ліміт кешбеку становить - 500 грн.

Клієнту не обов'язково повідомляти свій банк перед поїздкою за кордон. Комісія за зняття особистих коштів з банківського рахунку становить 2%, а

кредитних – 4%. Використання картки Monobank для здійснення платежів за кордоном не стягує комісії за конвертацію валюти. Крім того, всі витрати клієнта захищені від подвійної конвертації, оскільки будь-яка невідповідність понад 10 грн буде виявлена та усунена банком автоматично.

Monobank підхопив платіжну перевагу NFC, інтегрувавши свою картку з поширеними мобільними платформами, такими як Apple Pay і Google Pay, полегшуючи користувачам здійснювати покупки в магазинах за допомогою свого телефону.

У той час як інші банки дозволяють багаторазові видачі для своїх рахунків, Моно-рахунок дозволяє лише одну видачу. Однак це не означає, що користувач не може користуватися карткою, якщо залишиться вдома. Завдяки Apple Pay і Google Pay, які тепер доступні в Україні, користувачі можуть підключити свої кредитні картки до будь-якої з цих платформ і використовувати їх віртуально через смартфони для здійснення платежів онлайн або офлайн.

Крім того, доступ до картки monobank доступний через додаток у будь-який час; користувачі можуть зручно переглядати на своєму телефоні таку інформацію, як номер, термін дії та тризначний код CVV, якщо це необхідно.

Картка Monobank не зареєстрована за замовчуванням, але це не обмежує її використання в Україні. Тим не менш, під час міжнародних подорожей можуть бути випадки, коли картка з іменем може бути обов’язковою (наприклад, для оренди автомобіля).

Іменна картка Monobank

Monobank запровадив іменну кредитну картку для зручності мандрівників. Це дозволяє користувачам не потрапити в незручну ситуацію, коли їхня картка Monobank здається нікчемною під час оплати в готелях або на прокаті автомобілів за кордоном. .

У додатку Monobank можна замовити іменні картки з ім’ям та прізвищем на замовлення за 150 гривень. На оформлення та доставку нового пластику йде приблизно два тижні.

Валютна картка Monobank.

Використовуючи цю картку, ви можете робити покупки в міжнародних магазинах без необхідності подвійної конвертації валюти.

Кредитні послуги Monobank

Mono пропонує стартові позики на різноманітні цілі, такі як придбання товарів, житла чи транспортних засобів. Кошти також можуть використовувати ФОПи для покриття витрат, пов’язаних із охороною здоров’я, освітою та різноманітною підприємницькою діяльністю.

Умови змінюються залежно від вибраної вами проблеми - Black, Platinum або Iron. Вибір Black означає безкоштовну послугу без додаткових витрат на обслуговування, що робить її найбільш вигідною. Спочатку розглянемо першу згадану картку [5]

- Чорна картка

Випуск	Безкоштовно
Обслуговування	Безкоштовно
Максимальна сума кредиту	100 000 ₴
Комісія за зняття готівки	Кредитні кошти — 4% Власні кошти в Україні — 0.9% Власні кошти за кордоном — 2%
Пільгова відсоткова ставка	0,00001% річних
Пільговий період	62 дні
Збільшена відсоткова ставка	6.2% на місяць (нараховується, якщо є прострочена заборгованість)
Базова відсоткова ставка	3.1% на місяць (нараховується, якщо не погасити заборгованість в пільговий період)
Річна відсоткова ставка	37.2%
Розмір обов’язкового щомісячного платежу	4% від суми (не менше 100 ₴ та не більше залишку заборгованості)

Максимальний термін кредиту	25 років

Таблиця 1.6 Характеристика Чорної картки

- Платинова картка

Випуск	500 ₴
Обслуговування (фізичної основної та віртуальної додаткової разом)	3600 ₴ в рік
Обслуговування (фізичної основної та фізичної додаткової разом)	4800 ₴ в рік
Максимальний кредит	100 000 ₴
Комісія за зняття готівки	Кредитні кошти — 3.5% Власні кошти в Україні — 0.9% Власні кошти за кордоном — безкоштовно до 5000 ₴ в місяць і 2% більше 5000 ₴
Безвідсотковий період	62 дні
Базова відсоткова ставка	2.9% на місяць (нараховується, якщо не погасити заборгованість в пільговий період)
Пільгова відсоткова ставка	0,00001% річних
Збільшена відсоткова ставка	5.8% на місяць (нараховується, якщо є прострочена заборгованість)
Розмір обов’язкового щомісячного платежу	4% від суми (не менше 100 ₴ та не більше залишку заборгованості)
Річна відсоткова ставка	34.8%
Максимальний термін кредиту	25 років

Таблиця 1.7 Характеристика Платинової картки

- Залізна картка

Випуск	2000 ₴
Обслуговування	9000 ₴ в рік
Максимальний кредитний ліміт	100 000 ₴

Комісія за зняття готівки	Кредитні кошти — 3% Власні кошти в Україні — 0.9% Власні кошти за кордоном — безкоштовно до 10 000 ₴ і 2% від суми більше 10 000 ₴ в місяць
Комісія за зняття готівки	Кредитні кошти — 3% Власні кошти в Україні — 0.9% Власні кошти за кордоном — безкоштовно до 10 000 ₴ і 2% від суми більше 10 000 ₴ в місяць
Пільгова відсоткова ставка	0,00001% річних
Базова відсоткова ставка	2.9% на місяць (нараховується, якщо не погасити заборгованість в пільговий період)
Збільшена відсоткова ставка	5.8% на місяць (нараховується, якщо є прострочена заборгованість)
Річна відсоткова ставка	34.8%
Розмір обов’язкового щомісячного платежу	4% від суми (не менше 100 ₴ та не більше залишку заборгованості)
Максимальний термін кредиту	25 років

Таблиця 1.8 Характеристика Залізної картки

- Історія Monobank

Бета-тестування програми Fintech Band почалося в жовтні 2017 року, у ньому взяли участь близько 18 000 користувачів. Monobank був офіційно запущений на ринок 22 листопада.

По суті, Monobank – це пакет послуг, які Universal Bank пропонує за ліцензією, але під власним брендом. Він надає різні програми та послуги підтримки для клієнтів.

Віддалене обслуговування було першою зміною, яка зачепила користувачів. Отримання картки в банку стало непотрібним, що здавалося незвичним для

галузі, де більшість банків продовжують використовувати традиційні методи, такі як відділення та менеджери для взаємодії з клієнтами.

Банк зробив новаторський крок, запропонувавши кешбек у реальній валюті, а не просто в бонусах. Це був доволі інноваційний й крок, оскільки жодна інша компанія не робила цього до Mono, таким чином перетворивши банківську картку з просто платіжного інструменту на засіб заробітку грошей.

Сам додаток – третій аспект, який викликав дискусії. На відміну від інших програм мобільного банкінгу з обмеженими версіями настільних інтернет-банків (включно з Приват24, який довго очікували, але не виконали), monobank пропонує комплексні та повні банківські функції з самого початку. Крім того, його зручний і добре розроблений інтерфейс додає йому привабливості.

Емблема monobank у вигляді кота не залишилася непоміченою. Поряд із монологотипом, із зображенням кота, на кожній картці були включені наклейки з жартівливими підписами, як-от «Ізі, Ізі! ріел ток! сінк ебаут іт!» та «всьо нормально, дєньгі єсть». Це був перший випадок, коли тварина викликала асоціації з фінансовими послугами.

На момент відкриття монобанку штат Fintech Band налічував близько 150 осіб (без урахування монослужби підтримки), а штаб-квартира була розташована в Дніпрі. Надання карток клієнтам передбачало доставку працівниками банку або отримання в будь-якому з 30 центрів розподілу.

Початок служби перевершив очікування. Інавгураційна картка, за винятком учасників тестової групи, була опублікована в жовтні 2017 року та до 13 лютого 2018 року; загалом на послугу зареєструвалося сто тисяч клієнтів. До липня того ж року ця кількість зросла до трьохста тисяч.

Навіть після запуску робота над проектом тривала. Додаток monobank швидко розширювався своїми відмінними функціями.

Технологія Shake to pay була початковою інтригуючою новинкою, яка працювала в нетрадиційний спосіб. Струснувши смартфон із відкритою програмою, він запускає пошук пристроїв поблизу.

Monobank також представив безкоштовний план «Купівля в розстрочку» через рік після запуску.

На момент запуску «Закупівлі запчастин» у monobank вже було понад 500 тис. активних користувачів.

monobank представив свою першу пропозицію для VIP-персон, випуск Iron Bank у лютому 2019 року.

Після першого враження заможні клієнти були вражені продуктивністю та зовнішнім виглядом продукту. Унікальною особливістю, яка привернула їх увагу, став її темний відтінок, що нагадує мокрий асфальт, це досяглося за рахунок використання металу на відміну від традиційних матеріалів для виробництва банківських карток в Україні.

Листівка до цієї карти була створена за мотивами «Гри престолів» із зображенням кота, що сидить на вершині мечів у королівському стилі.

Крім того, власники Iron Bank отримали безкоштовний вхід до корпоративних залів у понад 820 аеропортах світу. Крім того, вони мали привілей користуватися засобами Fast Line, що прискорюють усі процедури польоту під час Прибуття та вильоту

Весь Залізний банк, як правило, покликаний підкреслити скрутне становище його власника.

За перший рік було виготовлено понад 3000 залізних карток, і їх кількість швидко зростає.

У 2019 році розробники Monobank оновили центр безпеки свого додатку, що призвело до несподіваної появи п’яти функцій.

1. Відхилення транзакцій, якщо місцезнаходження не відповідає вказаній країні: Після ввімкнення ця функція перехресно перевіряє мову оплати з даними GPS зі смартфона користувача і припиняє обробку будь-яких платежів, що надходять з іншого регіону.
2. Запит на пін-код карти: Залежно від налаштувань, його або запитували

завжди, або можна було вказати суму, до якої при оплаті карткою він не буде потрібен.

3. Оплата завдяки магнітній смузі: Оплата за допомогою магнітної смуги є вразливою для шахрайства, оскільки шахраї можуть дублювати банківські картки з магнітною смугою та знімати кошти з рахунку, попередньо отримавши номер картки та PIN-код. Щоб запобігти цьому, mono надає можливість вимкнути функцію магнітної смуги.
4. Зняття готівки у банкоматі: Для зняття готівки з картки в банкоматах заборонена опція готовності до зняття з картки, яка є в інших банках..
5. Безконтактна оплата: Розробники включили можливість вимкнути безконтактну оплату для обережних клієнтів, які можуть бути стурбовані шахрайською діяльністю осіб, які використовують портативні POS-термінали.

Шахраї, які намагаються обдурити клієнтів, зіткнулися зі значними перешкодами через п’ять нових функцій. Однак, у червні, робники моно ще більше розчарували їх, представивши шосту опцію, яка є безпрецедентною не лише на українському банківському ринку, а й у всьому світі.

6. Налаштування CVC2: Значна увага приділяється коду безпеки картки, відомому як код CVV або CVC2, який розташований на задній частині картки. Цей код є одним із основних ідентифікаторів, пов’язаних із карткою. У зв’язку з номером рахунку та терміном дії, введеними в Інтернеті, введення цієї комбінації унеможливлює оскарження платежу, здійсненого через цифрові системи – тому шахраїв особливо приваблюють ці 3 цифри. Щоб запобігти потенційній шкоді від таких дій, Monobank пропонує два варіанти захисту для вашої цінної конфігурації цифр - можливо або активувати динамічну зміну цього коду раз на годину, або налаштувати спеціальний пароль, який можна самостійно змінювати його за потреби.

У березні 2020 року з Mono відбулося багато важливих подій.

Лише за 2,5 роки компанія досягла важливої віхи: цього місяця її клієнтська

база перевищила позначку в 2 мільйони.

У березні з’явилося нове доповнення – картка, розроблена спеціально для підлітків до 16 років, тобто «Дитяча». Оскільки батьки постійно забезпечують їх грошима, діти вважаються важливою групою користувачів. Mono запропонував практичне рішення, представивши цю спеціальну картку, яка дозволяє дітям відчувати себе більш незалежними, дозволяючи їхнім родичам краще контролювати витрати та встаовлювати ліміти на конкретні транзакції, що полегшує життя обом залученим сторонам

Наступні введена інновація – піонерська мобільна гра в банківському додатку, яка може стати першою в Україні та світі. Monobank визнав, що одних тільки стимулів недостатньо, щоб захопити користувачів, тому вони включили «Space Invader» у свою систему. Щоб її запустити потрібно було потягнути виписку вниз, після чого вилітатиме ракета, і якщо встигнути натиснути, поки вона не покине межі екрану, починеться гра. Далі треба ловити котів, ухилятися від астероїдів і коронавірусу. Мета – набрати 100 очок - по одному за кожного падаючого кота.

Протягом вересня 2020 відбувся унікальний обмін між клієнтом Monobank та одним із співробітників служби підтримки. Клієнт грайливо написав працівнику: «Привіт! Можна 2 гривні?» На диво, представник перерахувавши йому необхідну суму. Ця безтурботна взаємодія була записана на TikTok і відразу стала інтернет-сенсацією. Після цього вірусного відеодопису багато людей звернулися до інших представників monobank з аналогічними закликами, просячи також отримати 2 гривні.

Результати флешмобу застали всіх зненацька. Співробітники Mono, як зазначив співзасновник Олег Гороховський, видали користувачам понад 30 тис. грн зі своїх особистих карток. Однак ще більш дивовижним є те, що ці самі користувачі відшкодували їм майже подвійну початкову суму.

Крім того, працівник, який ініціював цей флешмоб, отримав винагороду у розмірі 10 тисяч гривень.

РОЗДІЛ 2: ВИБІР СКЛАДОВИХ ПРИ ПІДГТОВЦІ ДО СТВОРЕННЯ ДОДАТКУ

2.1 Використання .NET 8

Конференція .NET Conf 2023 відзначила презентацію остаточної версії .NET 8 від Microsoft 14 листопада. Кожна ітерація цієї платформи є важливою подією, що означає важливу контрольну точку для роздумів над минулими розробками, а також окреслюює майбутні технологічні досягнення.

Причини вибору саме цієї платформи:

- Легкість налагодження.

Процес налагодження коду став значно зручнішим завдяки внесенню хоч і незначних але численних змін інженерами Microsoft. Примітно те, що з NET 8 тепер простіше переглядати властивості для різних типів, включаючи HttpResponse, HttpRequest і HttpContext. Крім того, надлишкова інформація є прихованою, яка відображалася для багатьох типів. [15]

- Посилення заходів безпеки

Підхід до контейнеризації, реалізований у версії 8 платформи, значно покращив її безпеку. Усі образи контейнерів на базі Linux тепер матимуть некореневе користувача, що дозволить контейнерам .NET працювати без привілеїв суперкористувача. Використання концепції найменших привілеїв призведе до обмеження можливостей потенційних зловмисників, покращить безпеку додатків і всієї екосистеми .NET . [11]

- Використання ШІ

За рахунок штучного інтелекту та генеративного штучного інтелекту, можливості інтеграції .NET 8 було розширено, що забезпечує покращену підтримку обробки масивних мовних моделей і підключення до різноманітних служб ШІ.

- Покращення продуктивності та ефективності.

Зусилля щодо оптимізації платформи команди .NET були значними. Самі

інженери підтверджують це, посилаючись на те, що ".NET 8" перевершив попередню версію за швидкістю, і наразі цю версію називають найшвидшою [12]

- .NET Aspire

Обговорюючи інноваційний напрямок, до якого прямує .NET 8, важливо визнати два основні аспекти: ШІ та .NET Aspire. Другий створений спеціально для розробки хмарних додатків, цей стек включає різні компоненти, розроблені для оптимальної продуктивності в хмарному середовищі за допомогою таких функцій, як: 1) виявлення служб, 2) відстеження телеметрії та 3) тестування справності. Хоча наразі доступна лише попередня версія Aspire. [13]

- Розробка додатків для клієнтів.

Корпорація Майкрософт постійно працює над удосконаленням своїх технологій для розробки додатків, десктопних чи веб-орієнтованих. MAUI та Blazor — дві такі розробки, які привернули увагу розробників у всьому світі. Завдяки повній пропозиції, Blazor дозволяє користувачам доволі легко та швидко створювати власні веб-програми. Ця технологія дозволяє використовувати як сервер Blazor, так і WebAssembly в одній програмі, що забезпечує високу продуктивність.

Що стосується MAUI, то інженери в першочергово зосередилися на вирішенні проблем із помилками. Загалом було надіслано понад 1500 запитів на вилучення, щоб вирішити близько 600 окремих проблем. Варто також зазначити, що як команда .NET MAUI, так і ширша спільнота зробили свій внесок у ці зусилля.

- Нові версії мов

8 версія платформи включає оновлені варіанти мов C# і F#, причому перша асимілює широкий спектр додаткових характеристик, таких як: 1) первинні конструктори, 2) вирази колекції, 3) вбудовані масиви, 4) додаткові параметри в лямбда-виразах, 5) параметри лише для читання ref або перехоплювачі будь-яких типів. Подібним чином останнє видання F# може похвалитися декількома функціями, спрямованими на те, щоб зробити програми простішими та

ефективнішими, одночасно забезпечуючи однаковість коду. [14,16]

2.2 Моделювання програмного забезпечення завдяки DDD

2.2.1 Зміна напрямку розробок програмного забезпечення

Підхід до розробки програмного забезпечення на основі доменно-керованого проектування (DDD) зосереджується на розумінні та моделюванні проблемної області. Цей метод надає перевагу використанню єдиної мови та стратегічного дизайну для того щоб створити програмне забезпечення, яке відповідатиме інтелектуальним структурам, що будуть використовуватися експертами галузі. Виконання DDD передбачає а) ідентифікацію фундаментального домену, б) створення його відповідної моделі та в) використання доменів структурованих шаблонів проектування. Цей метод підвищує як якість програмного забезпечення, так і ефективність розробника, в той самий час зменшуючи непорозуміння під час процесів розробки.

Бути в курсі останніх методологій та підходів є надзвичано важливим для розробників у сучасному швидкоплинному світі розробки програмного забезпечення. Одним із підходів, який нещодавно отримав значну увагу та визнання, є Domain-Driven Design (DDD), що ставить домен у центр дизайну, наголошуючи на доменно-орієнтованому підході.

DDD — це підхід до розробки програмного забезпечення, який надає пріоритет розумінню та конструюванню рішення проблем домену за допомогою моделювання. У минулому розробка програмного забезпечення надавала пріоритет технічним аспектам, таким як фреймворки та бази даних. Однак зосередження лише на цих елементах може перешкодити зрозуміти проблемну область, що, у свою чергу, може призвести до неповного або некерованого програмного забезпечення, яке не відповідає належним чином вимогам бізнесу. DDD ж використовує методологію, орієнтовану на домен, яка спонукає розробників до комплексного вирішення проблеми. Отримавши широке розуміння бізнес-концепцій, правил і процесів у проблемній області, вони можуть розробити

програмне забезпечення, яке правдиво відображатиме її складність.

2.2.2 Головні принципи DDD

1) Принцип розробки та використання універсальної мови є невід'ємною частиною DDD. Ця практика слугує важливим зв'язком між нетехнічними та технічними зацікавленими сторонами, сприяючи ефективній комунікації та взаєморозумінню щодо предметної сфери. Завдяки узгодженості у використанні мови, розробники можуть ефективніше співпрацювати з експертами з різних областей та створювати потірбне програмне забезпечення

2) DDD сприяє стратегічному плануванню, яке передбачає ідентифікацію та визначення фундаментальних понять, обмежень і зв'язків у певній галузі. Для досягнення цієї мети використовуються моделі доменів, оскільки вони представляють ключові об'єкти домену, їх взаємодію та поведінку.

2.2.3 Переваги DDD

Доменно-орієнтований підхід дає повне розуміння проблемних областей. Завдяки тісній взаємодії з фахівцями галузі розробники програмного забезпечення отримують суттєве розуміння складнощів і тонкощів поставленої задачі. Ці знання дозволяють їм розробляти спеціалізоване програмне забезпечення, яке відповідає конкретним вимогам і потребам певної галузі, що в кінцевому підсумку призводить до більш ефективних рішень. Таким чином, таке програмне забезпечення, створене на замовлення, спрямоване на вирішення фактичних проблем користувача, одночасно ефективно забезпечуючи бажані функції.

2.2.4 Приклади успішного впровадження DDD

- Ерік Еванс та система доставки вантажів.

Приклад системи доставки завантажень Еріка Еванса є відомою ілюстрацією спільного проектування, керованого доменом. Завдяки комплексному застосуванню методів DDD систему доставки вантажів було успішно розроблено для обробки складних бізнес-правил і взаємодії між різними об'єктами в домені.

- Система управління полісами Aviva Insurance.

Впровадження принципів DDD для повного оновлення запровадила

впливова страхова компанія Aviva.

Загалом, Domain-Driven Design (DDD) сприяє новому погляду на розробку програмного забезпечення, який визначає пріоритетність домену в основі. Коли розробники розуміють і використовують фундаментальні принципи DDD, вони можуть розробляти програмне забезпечення, яке оптимально відповідатиме вимогам. Переваги використання DDD включають вдосконалену архітектуру програмного забезпечення, підвищену гнучкість і меншу складність, а отже це все робить його незамінним інструментарієм для будь-якої команди розробників, яка прагне досягти успіху. Крім того, приклади реального життя змогли успішно використати цей сприятливий підхід, підкреслюючи його ефективність для досягнення бажаних результатів щодо доменів.

2.3 MediatR та CQRC

Шаблон проєктування MediatR відноситься до категорії поведінкових шаблонів і служить для зменшення залежності в ситуаціях, пов’язаних із численними класовими відносинами. Завдяки використанню єдиного каналу зв’язку цей шаблон дозволяє класам знати, що потрібно повідомити, не турбуючись про те, як це має відбуватися. Отже, взаємозалежність об’єктів зменшується, що призводить до кращої організації наявної системи. По суті, об’єкти полегшуються через проміжне програмне забезпечення, яке керує їхніми функціями спільної роботи, щоб вони могли функціонувати незалежно один від одного без прямих каналів зв’язку між ними.

Для ілюстрації концепції зазвичай використовується літак. Під час зльоту та посадки літаки повинні знати про поточний стан один одного. Коли є лише два або три літаки, прямий зв’язок між пілотами може надати цю інформацію без проблем, однак все стає складніше, коли задіяно кілька літаків. У результаті пілоти запитують дозвіл у вежі, щоб отримати доступ до деталей про статуси своїх колег. У цьому контексті наш клас «Колега» складається з літаків, тоді як наш клас «Посередник» втілює роль диспетчерських веж.

Материнська плата може служити відповідним прикладом для ілюстрації взаємозалежного комп’ютерного обладнання, поряд із широко відомим літаком. Всі ми знаємо, що різні компоненти комп’ютера залежать один від одного. Наприклад, для відтворення аудіо- чи відеофайлу з компакт-диска, окрім встановленого приводу компакт-дисків, потрібне обладнання звукової та відеокарти; тоді як зв’язок між драйверами DVD і динаміками відбувається окремо через материнську плату, яка виконує роль нашого класу посередників, залишаючись при цьому відмінною від інших внутрішніх підсистем, таких як - графічні карти.

Розподіл відповідальності за командний запит (CQRC):

Концепція програмування CQRC передбачає використання різних об’єктів для керування функціями запису та читання. Його розроблено з метою розподілу відповідальності між командами та відповідного керування запитами.

Команда: стосується виконання дій із запису бази даних, таких як зміна або видалення наявних даних і додавання нової інформації. Приклади включають оператори INSERT, DELETE та UPDATE.

Варіант: виберає із наявних даних і відновлює їх. Його функція полягає в отриманні інформації, що зберігається в базі даних.

CQRC принципово розділяє операції читання та запису для підвищення загальної продуктивності. Використовуючи різні об’єкти для керування читанням і записом, проект стає більш модульним із незалежними операціями. Подальший прогрес дозволяє керувати цими завданнями в окремих базах даних, де збій однієї операції не буде перешкоджати іншій. Це означає, що користувачі можуть отримати доступ до даних, навіть якщо виникає помилка запису. Крім того, складну логіку домену можна створити окремо для цих двох операцій. Це також спрощує складне відображення об’єтів, тому для кожної операції ми можемо легко надати користувачеві лише ту інформацію, яка буде йому потрібна.

MediatR

MediatR — це бібліотека .NET framework, яка використовує шаблон

проектування Mediator для регулювання та обробки повідомлень у програмних програмах.

За допомогою MediatR можливо розробляти код, який буде легше підтримувати та змінювати, оскільки він дозволяє створювати незалежні рівні додатків і компоненти. MediatR є особливо потужним і корисним при використанні з шаблоном CQRC (Command Query Responsibility Segregation). Таким чином, окрема обробка команд і запитів у програмі робить програму більш керованою та масштабованою.

Queries

```
[HttpGet("by-period")]
Dmytro Shevchenko
public async Task<IActionResult> GetTransactionsByPeriod(Guid clientCardId, DateTime? dateFrom, DateTime? dateTo,
    CancellationToken cancellationToken)
{
    var command = new GetClientTransactionsQuery(clientCardId, dateFrom, dateTo);
    var result = await _mediator.Send(command, cancellationToken);
    return result.IsSuccess ? Ok(result.Value) : BadRequest(result.Error);
}
```

```
public class GetClientTransactionsQueryHandler : IQueryHandler<GetClientTransactionsQuery, List<ClientTransactionDto>>
{
    private readonly IClientTransactionRepository _clientTransactionRepository;
    private readonly IMapper _mapper;

    Dmytro Shevchenko
    public GetClientTransactionsQueryHandler(IClientTransactionRepository clientTransactionRepository, IMapper mapper)
    {
        _clientTransactionRepository = clientTransactionRepository;
        _mapper = mapper;
    }

    Dmytro Shevchenko
    public async Task<Result<List<ClientTransactionDto>>> Handle(GetClientTransactionsQuery request,
        CancellationToken cancellationToken)
    {
        var (dateFrom, dateTo) = TransactionsPeriodHelper
            .GetMonobankTransactionsPeriod(request.DateFrom, request.DateTo);

        var transactions = await _clientTransactionRepository.GetByPeriod(request.ClientCardId, dateFrom, dateTo, cancellationToken);
        return _mapper.Map<List<ClientTransactionDto>>(transactions);
    }
}
```

```
[HttpGet("latest")]
Dmytro Shevchenko
public async Task<IActionResult> FetchLatestTransactions(string externalCardId, CancellationToken cancellationToken)
{
    var command = new FetchLatestTransactionsCommand(externalCardId);
    var result = await _mediator.Send(command, cancellationToken);
    return result.IsSuccess ? Ok() : BadRequest(result.Error);
}
```

Commands

```
public FetchLatestTransactionsCommandHandler(IHttpClientFactory factory, IClientCardsRepository cardsRepository,
    IClientTransactionRepository transactionRepository, IClientRepository clientRepository, IUnitOfWork unitOfWork,
    ISender sender)
{
    _unitOfWork = unitOfWork;
    _sender = sender;
    _cardsRepository = cardsRepository;
    _transactionRepository = transactionRepository;
    _clientRepository = clientRepository;
    _httpClient = factory.CreateClient(MonobankConstants.HttpClient);
}

Dmytro Shevchenko
public async Task<Result> Handle(FetchLatestTransactionsCommand request, CancellationToken cancellationToken)
{
    var clientCard = await _cardsRepository.GetByExternalId(request.ExternalCardId, cancellationToken);
    if (clientCard is null)
        return Result.Failure(new Error("ClientCard.NotFound",
            $"No client card with External Id {request.ExternalCardId}"));

    long unixTimeFrom;
    var latest = await _transactionRepository.GetLatest(clientCard.Id, cancellationToken);
    if (latest is null)
        unixTimeFrom = DateTimeOffset.Now.AddDays(-MaxDaysPeriod).ToUnixTimeSeconds();
    else
        unixTimeFrom = DateTimeOffset.Now - latest.DateOccured
                       < TimeSpan.FromDays(MaxDaysPeriod)
            ? ((DateTimeOffset)latest!.DateOccured).ToUnixTimeSeconds() + 1
            // because we dont want to take the existing record from monobank api
            : DateTimeOffset.Now.AddDays(-MaxDaysPeriod).ToUnixTimeSeconds();

    var unixTimeTo = DateTimeOffset.Now.ToUnixTimeSeconds();
    var url = BuildUrl(clientCard.ExternalCardId, unixTimeFrom, unixTimeTo);
    var client = await _clientRepository.GetById(clientCard.ClientId, cancellationToken);
    _httpClient.DefaultRequestHeaders.Add(MonobankConstants.TokenHeader, client.PersonalToken);

    var result = await _httpClient.GetAsync(url, cancellationToken);
    var monobankTransactions = (await result.Content.ReadFromJsonAsync<IEnumerable<MonobankTransaction>>(
        cancellationToken:
        cancellationToken) ?? Array.Empty<MonobankTransaction>()).ToList();
```

РОЗДІЛ 3: ПРОЕКТУВАННЯ ТА РОЗРОБКА ДОДАТКУ

3.1 Важливість використання Entity Framework

Entity Framework Core (EF Core) — це рекомендований об’єктно-реляційний картограф (ORM) для .NET , раніше відомого як .NET Core. Як наступник EF6, EF Core був повністю перероблений і став відкритим кодом на GitHub .

Причини використовувати ORM:

ORM (Object-Relational Mapper) використовується для взаємодії з базою даних за допомогою об’єктно-орієнтованої мови програмування. ORM дозволяють розробникам працювати з базами даних, використовуючи знайомі об’єктно-орієнтовані концепції, а не писати необроблені оператори SQL.

1) Це може зробити процес розробки ефективнішим і менш схильним до помилок, а також спростить обслуговування кодової бази.
2) Крім того, ORM часто надають такі функції, як кешування, відкладене завантаження та об’єднання з’єднань, що може покращити продуктивність програми.
3) Вони також забезпечують рівень абстракції між додатком і базою даних, щоб код програми міг бути ізольованим від змін базової схеми бази даних.
4) Це може полегшити перехід до іншої бази даних у майбутньому або покращити масштабованість шляхом розподілу даних між кількома серверами.
5) Загалом, ORM можуть допомогти підвищити продуктивність розробників, зручність обслуговування коду та продуктивність програми.

Більшість фреймворків розробки включають бібліотеки, які забезпечують доступ до даних із реляційних баз даних через структури даних, подібні до набору записів.

Особливості Entity Framework Core:

Entity Framework Core (EF Core) має кілька функцій, які роблять його потужним і ефективним інструментом для доступу до даних і керування ними:

1. Кросплатформенність : EF Core можна використовувати на різних платформах, включаючи Windows, Linux і Mac.
2. Легкість : EF Core займає меншу площу та менше залежностей, ніж повна версія Entity Framework.
3. Перш за все код : EF Core дозволяє розробникам створювати базу даних із коду, що забезпечує гнучкіший робочий процес розробки, керований тестами.
4. Підтримка LINQ : EF Core підтримує LINQ, потужну та експресивну мову запитів, яка дозволяє розробникам писати ефективні та читабельні запити за допомогою C# або Visual Basic.
5. Підтримка кількох баз даних : EF Core підтримує широкий спектр реляційних баз даних, включаючи SQL Server, MySQL, SQLite та PostgreSQL.
6. Міграції : EF Core має вбудовану підтримку для створення та керування міграціями баз даних, що дозволяє легко керувати змінами бази даних з часом.
7. Покращення продуктивності : EF Core оптимізовано для продуктивності та може ефективно обробляти великі набори даних.
8. Моделювання : підтримка складного типу та типу власності.
9. Відносини : підтримка один-до-одного , один-до-багатьох і багато-до-багатьох
10. Успадкування : підтримка TPC , TPH і TPT
11. Підтримка клієнтської оцінки та відкладеного завантаження.
12. Підтримка явного завантаження, відстеження змін і кешування.

Entity Framework Core (EF Core) — це платформа ORM (Object-Relational Mapping) для платформи .NET. Однією з ключових особливостей EF Core є підтримка жорсткої типізації.

Працюючи з EF Core, можливо визначити схему бази даних за допомогою класів C#, відомих як сутності, потім ці класи зіставляються з відповідними таблицями в базі даних.

Оскільки сутності строго типізовані, компілятор може виявити будь-які помилки, пов’язані зі схемою, під час компіляції, а не під час виконання.

Крім того, EF Core підтримує використання LINQ (Language-Integrated Query) для надсилання запитів до даних у базі даних, що надає чітко типізований, експресивний і простий у використанні API для запиту даних.

Це може полегшити розробникам написання зручного та ефективного коду.

ORM — це попередньо написані бібліотеки коду, які виконують цю роботу за розробника. Повнофункціональні ORM також роблять набагато більше. Вони можуть:

1) зіставити модель предметної області з об’єктами бази даних
2) керувати транзакціями
3) генерувати SQL і виконувати його з базою даних
4) відстежувати об'єкти, які вже були вилучені
5) створювати бази даних і підтримувати схему відповідно до змін моделі

```
public class ClientRepository : IClientRepository
{
    private readonly OutlayContext _context;

    Dmytro Shevchenko
    public ClientRepository(OutlayContext context)
    {
        _context = context;
    }

    0+1 usages  Dmytro Shevchenko
    public Task AddAsync(Client client, CancellationToken cancellationToken = default)
    {
        return _context.AddAsync(client, cancellationToken).AsTask();
    }

    Dmytro Shevchenko
    public void Update(Client client, CancellationToken cancellationToken = default)
    {
        _context.Update(client);
    }

    0+1 usages  Dmytro Shevchenko
    public Task<Client> GetByPersonalToken(string token, CancellationToken cancellationToken = default)
    {
        return _context.Clients.FirstOrDefaultAsync(x => x.PersonalToken == token, cancellationToken)!;
    }

    0+1 usages  Dmytro Shevchenko
    public Task<Client> GetById(Guid clientId, CancellationToken cancellationToken = default)
    {
        return _context.Clients.FirstOrDefaultAsync(x => x.Id == clientId, cancellationToken)!;
    }

    0+1 usages  Dmytro Shevchenko
    public Task<Client> GetByIdWithCards(Guid clientId, CancellationToken cancellationToken = default)
    {
        return _context.Clients.Include(x => x.Cards).FirstOrDefaultAsync(x => x.Id == clientId, cancellationToken)!;
    }
}
```

3.2 Automapper

AutoMapper використовує API вільної конфігурації для визначення стратегії відображення об’єкт-об’єкт. AutoMapper використовує алгоритм зіставлення на основі конвенцій, щоб зіставити вихідні та цільові значення. AutoMapper орієнтований на сценарії проектування моделей, щоб зводити моделі складних об’єктів до DTO та інших простих об’єктів, чий дизайн краще підходить для серіалізації, зв’язку, обміну повідомленнями або просто антикорупційного рівня між доменом і прикладним рівнем.

```
public class ClientTransactionConverter : ITypeConverter<GroupedTransaction, ClientTransactionsGroupedResponse>
{
    private readonly OutlayInMemoryContext _inMemoryContext;
    private readonly ILogoReferenceRepository _logoReferenceRepository;

    Dmytro Shevchenko
    public ClientTransactionConverter(OutlayInMemoryContext inMemoryContext, ILogoReferenceRepository logoReferenceRepository)
    {
        _inMemoryContext = inMemoryContext;
        _logoReferenceRepository = logoReferenceRepository;
    }

    Dmytro Shevchenko
    public ClientTransactionsGroupedResponse Convert(GroupedTransaction source,
        ClientTransactionsGroupedResponse destination,
        ResolutionContext context)
    {
        var cat = _inMemoryContext.MccInfos.FirstOrDefault(x => x.Mcc == source.Mcc)!.ShortDescription;
        var name = source.Name.Replace("Скасування. ", string.Empty);
        var icon = _logoReferenceRepository.GetByName(name, CancellationToken.None).Result?.Url ?? string.Empty;
        return new ClientTransactionsGroupedResponse
        {
            Name = source.Name,
            Amount = source.Amount,
            Icon = icon,
            Category = cat
        };
    }
}
```

```
Dmytro Shevchenko
public class ClientTransactionsProfile : Profile
{
    Dmytro Shevchenko
    public ClientTransactionsProfile()
    {
        CreateMap<GroupedTransaction, ClientTransactionsGroupedResponse>()
            .ConvertUsing<ClientTransactionConverter>();

        CreateMap<ClientTransaction, ClientTransactionByDescriptionResponse>()
            .ForMember(x => x.DateOccured, opt => opt.MapFrom(x => $"{x.DateOccured:g}"))
            .ForMember(x => x.Name, opt => opt.MapFrom(x => x.Description));

        CreateMap<ClientTransaction, ClientTransactionDto>()
            .ConvertUsing<ClientTransactionsRawConverter>();
    }
}
```

3.3 Quartz.NET

Фонові завдання мають вирішальне значення для багатьох програмних програм, і планування їх виконання періодично або в певний час є загальною вимогою. В екосистемі .NET Quartz.NET забезпечує надійну та гнучку структуру для планування таких завдань.

- Середовища виконання

1) Quartz.NET може працювати вбудовано в іншу автономну програму
2) Quartz.NET може працювати як окрема програма (у власному екземплярі віртуальної машини .NET) для використання через .NET Remoting
3) Quartz.NET можна створити як кластер автономних програм (з можливостями балансування навантаження та відновлення після збоїв)

- Планування

Виконання завдань заплановано, коли виникає певний тригер. Тригери можна створювати за допомогою майже будь-якої комбінації наступних директив:

1) в певний час доби (з точністю до мілісекунди)
2) в певні дні тижня
3) в певні дні місяця
4) в певні дні року
5) не в певні дні, зазначені в зареєстрованому календарі (наприклад, робочі свята)
6) повторюється певну кількість разів
7) повторюється до певного часу/дати
8) повторюється безкінечно
9) повторюється із затримкою

Роботи отримують назви від їх творців, а також можуть бути організовані в іменовані групи. Тригерам також можна давати назви та розміщувати їх у групи, щоб легко впорядкувати їх у планувальнику. Завдання можна додати до планувальника один раз, але зареєструвати їх у кількох тригерах.

- Виконання завдання

Екземпляри класів завдань можуть бути створені Quartz.NET або фреймворком програми.

Коли виникає тригер, планувальник сповіщає про нуль або більше об’єктів .NET, що реалізують інтерфейси JobListener і TriggerListener. Ці слухачі також отримують сповіщення після виконання завдання.

Коли завдання завершуються, вони повертають JobCompletionCode, який інформує планувальник про успіх чи невдачу. JobCompletionCode також може вказати планувальнику будь-які дії, які він має виконати на основі коду успіху чи невдачі, наприклад негайне повторне виконання завдання.

- Наполегливість у роботі

Дизайн Quartz.NET включає інтерфейс IJobStore, який можна реалізувати, щоб забезпечити різні механізми для зберігання завдань.

За допомогою включеного AdoJobStore усі завдання та тригери, налаштовані як «енергонезалежні», зберігаються в реляційній базі даних через ADO.NET.

Завдяки використанню включеного RAMJobStore усі завдання та тригери зберігаються в оперативній пам’яті й тому не зберігаються між виконаннями програми, але це має перевагу, оскільки не потрібна зовнішня база даних.

- Кластеризація

1) Відмова.
2) Балансування навантаження.

- Listeners & Plug-Ins

Програми можуть перехоплювати події планування, щоб відстежувати або контролювати поведінку завдання/тригера, реалізувавши один або кілька інтерфейсів слухачів. Механізм плаїіна можна використовувати для додавання функціональності Quartz, наприклад для збереження історії виконання завдань або завантаження визначень завдань і тригерів із файлу. Quartz поставляється з низкою «заводських» плагінів і прослуховувачів.

```
public static class DependencyInjection
{
    1 usage  Dmytro Shevchenko
    public static IServiceCollection AddBackgroundJobs(this IServiceCollection services)
    {
        services.AddQuartz(configure =>
        {
            var jobKey = new JobKey(nameof(ProcessOutboxMessagesJob));

            configure.AddJob<ProcessOutboxMessagesJob>(jobKey)
                .AddTrigger(trigger =>
                    trigger
                        .ForJob(jobKey)
                        .WithSimpleSchedule(schedule =>
                            schedule
                                .WithIntervalInSeconds(10)
                                .RepeatForever()));

            configure.UseMicrosoftDependencyInjectionJobFactory();
        });

        services.AddQuartzHostedService();

        return services;
    }
}
```

```
public class ProcessOutboxMessagesJob : IJob
{
    private readonly OutlayContext _context;
    private readonly IPublisher _publisher;

    Dmytro Shevchenko
    public ProcessOutboxMessagesJob(OutlayContext context, IPublisher publisher)
    {
        _context = context;
        _publisher = publisher;
    }

    Dmytro Shevchenko
    public async Task Execute(IJobExecutionContext context)
    {
        var messages = await _context
            .Set<OutboxMessage>()
            .Where(m => m.ProcessedOnUtc == null)
            .Take(20)
            .ToListAsync(context.CancellationToken);

        foreach (var message in messages)
        {
            var domainEvent = JsonConvert.DeserializeObject<IDomainEvent>(
                message.Content,
                new JsonSerializerSettings { TypeNameHandling = TypeNameHandling.All }
            );

            if (domainEvent is null)
                continue;

            await _publisher.Publish(domainEvent, context.CancellationToken);
            message.ProcessedOnUtc = DateTimeOffset.Now.ToUnixTimeSeconds();
            await _context.SaveChangesAsync();
        }
    }
```

3.4 Рішення керування ключами в Azure

Azure Key Vault — це одне з кількох рішень керування ключами в Azure , яке допомагає вирішити наступні проблеми:

1) Керування секретами – Azure Key Vault можна використовувати для безпечного зберігання та жорсткого контролю доступу до токенів, паролів, сертифікатів, ключів API та інших секретів
2) Керування ключами – Azure Key Vault можна використовувати як рішення

для керування ключами. Azure Key Vault спрощує створення та керування ключами шифрування, які використовуються для шифрування ваших даних.

3) Керування сертифікатами – Azure Key Vault дозволяє легко надавати, керувати та розгортати загальнодоступні та приватні сертифікати безпеки транспортного рівня/рівня захищених сокетів (TLS/SSL) для використання з Azure і вашими внутрішніми підключеними ресурсами.

Azure Key Vault має два рівні обслуговування: стандартний, який шифрує за допомогою програмного ключа, і преміальний рівень, який включає ключі, захищені апаратним модулем безпеки (HSM)

Причини використовувати Azure Key Vault?

- Надійно зберігаються секрети та ключі

Для доступу до сховища ключів необхідна належна автентифікація та авторизація, перш ніж абонент (користувач або програма) зможе отримати доступ. Автентифікація встановлює особу абонента, а авторизація визначає операції, які їм дозволено виконувати.

Автентифікація здійснюється через Microsoft Entra ID. Авторизацію можна здійснити за допомогою керування доступом на основі ролей Azure (Azure RBAC) або політики доступу Key Vault. Azure RBAC можна використовувати як для керування сховищами, так і для доступу до даних, що зберігаються в сховищі, тоді як політику доступу до сховища ключів можна використовувати лише під час спроби отримати доступ до даних, що зберігаються в сховищі.

Сховища ключів Azure можуть бути захищені програмним забезпеченням або, за допомогою рівня Azure Key Vault Premium, апаратно захищені апаратними модулями безпеки (HSM). Захищені програмним забезпеченням ключі, секрети та сертифікати захищаються Azure за допомогою галузевих стандартних алгоритмів і довжини ключів. У ситуаціях, коли вам потрібна додаткова гарантія, ви можете імпортувати або генерувати ключі в HSM, які ніколи не виходять за межі HSM. Azure Key Vault використовує nCipher HSM, які відповідають Федеральним стандартам обробки інформації (FIPS) 140-2 рівня 2. Ви можете використовувати

інструменти nCipher, щоб перемістити ключ із свого HSM до Azure Key Vault

Нарешті, Azure Key Vault розроблено таким чином, що Microsoft не бачить і не витягує дані користувачів.

- Централізовані секрети програми

Централізоване зберігання секретів програми в Azure Key Vault дозволяє контролювати їх розподіл. Key Vault значно зменшує ймовірність випадкового витоку секретів. Коли розробники додатків використовують Key Vault, їм більше не потрібно зберігати інформацію про безпеку у своїх додатках. Відсутність необхідності зберігати інформацію про безпеку в програмах усуває необхідність робити цю інформацію частиною коду. Наприклад, програмі може знадобитися підключення до бази даних. Замість того, щоб зберігати рядок підключення в коді програми, ви можете безпечно зберегти його в Key Vault.

Програми можуть безпечно отримувати доступ до необхідної інформації за допомогою URI. Ці URI дозволяють програмам отримувати певні версії секрету. Немає необхідності писати спеціальний код для захисту будь-якої секретної інформації, що зберігається в Key Vault..

- Спрощене адміністрування секретів програми

Зберігаючи цінні дані, потрібно зробити кілька кроків. Інформація про безпеку має бути захищеною, пройти життєвий цикл і бути високодоступною. Azure Key Vault спрощує процес виконання цих вимог за допомогою:

a) Усунення потреби у внутрішніх знаннях апаратних модулів безпеки.

b) Розширення в найкоротші терміни, щоб відповідати стрибкам використання вашою організацією.

c) Копіювання вмісту вашого сховища ключів у межах регіону та до додаткового регіону. Реплікація даних забезпечує високу доступність і усуває необхідність будь-яких дій з боку адміністратора для ініціювання відновлення після відмови.

d) Надання стандартних параметрів адміністрування Azure через портал, Azure

CLI та PowerShell.

е) Автоматизація певних завдань щодо сертифікатів, які ви купуєте в державних центрах сертифікації, як-от реєстрація та оновлення.

Крім того, сховища ключів Azure дозволяють відокремлювати секрети програм. Програми можуть отримувати доступ лише до сховища, доступ до якого їм дозволено, і їх можна обмежити виконанням лише певних операцій. Є можливість створити сховище ключів Azure для кожної програми та обмежити секрети, що зберігаються в сховищі ключів, певною програмою та групою розробників.

- Контроль доступу та використання

Створивши кілька сховищ ключів, користувач схочете контролювати, як і коли здійснюється доступ його ключів і секретів. Він можете відстежувати активність, увімкнувши журналювання для своїх сховищ. Існує можливість налаштувати Azure Key Vault на:

1) Архівування в обліковий запис зберігання.
2) Потік до центру подій.
3) Надішлення журналів до журналів Azure Monitor.

Також можливо видалити журнали, які більше не потрібні

- Інтеграція з іншими службами Azure

Як безпечне сховище в Azure, Key Vault використовувався для спрощення таких сценаріїв:

1) Шифрування диска Azure
2) Функція завжди зашифрованого та прозорого шифрування даних на SQL-сервері та базі даних SQL Azure
3) Служба додатків Azure .
4) Сам Key Vault може інтегруватися з обліковими записами зберігання, центрами подій і аналітикою журналів.

```
public static class DependencyInjection
{
    1 usage  Dmytro Shevchenko
    public static ConfigurationManager AddKeyVault(this ConfigurationManager configuration, bool isProduction)
    {
        if (!isProduction)
            return configuration;

        var vaultUrl = configuration[KeyVaultConstants.Url];
        var tenantId = configuration[KeyVaultConstants.TenantId];
        var clientId = configuration[KeyVaultConstants.ClientId];
        var secretId = configuration[KeyVaultConstants.ClientSecretId];

        var credential = new ClientSecretCredential(tenantId, clientId, secretId);
        var client = new SecretClient(new Uri(vaultUrl!), credential);
        configuration.AddAzureKeyVault(client, new AzureKeyVaultConfigurationOptions());

        return configuration;
    }
}
```

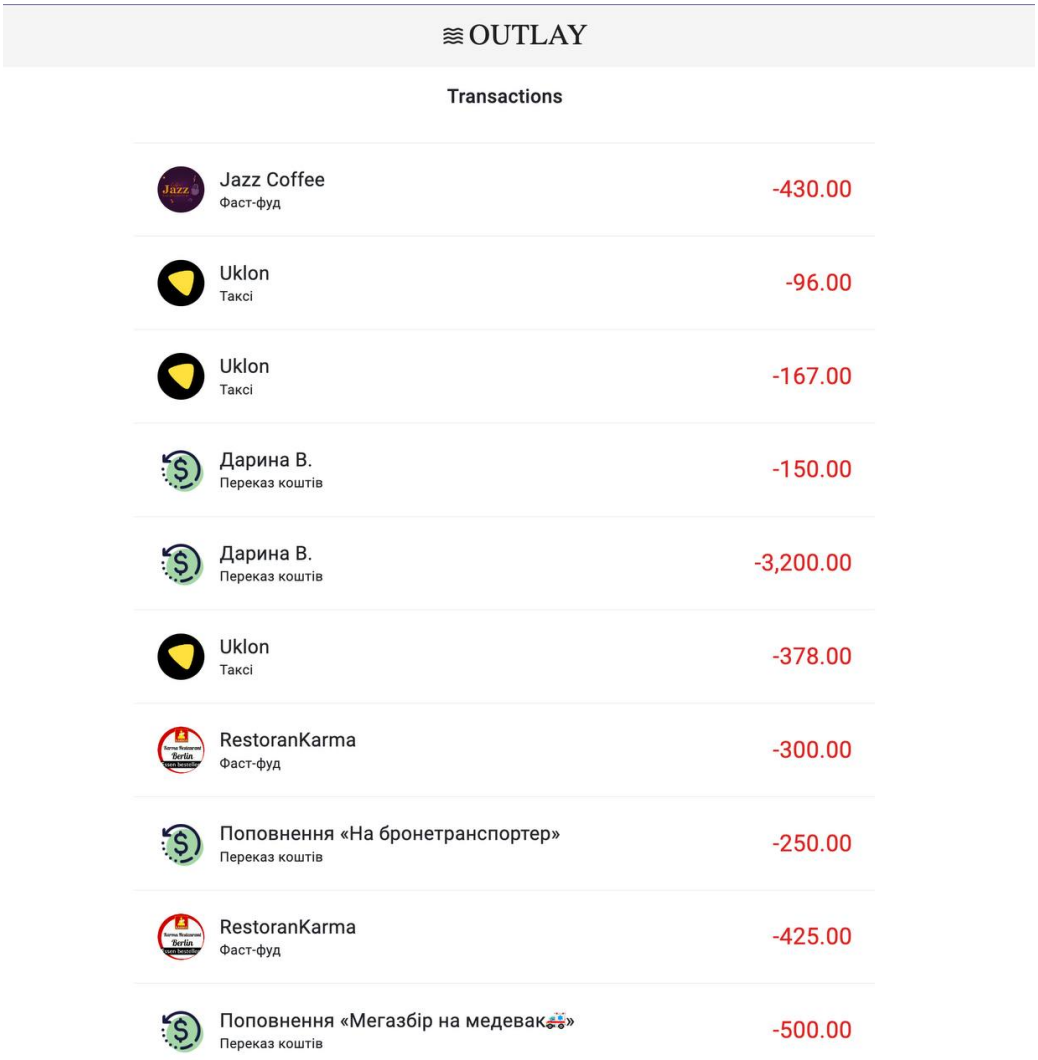
3.5 Проєкт Outlay App для для відстеження статистики та деталей транзакцій Monobank

3.5.1 Історія транзакцій, сторінка з затратами і прибутками за 15 днів та 1 місяць.

Outlay — це проєкт, створений для відстеження статистики та деталей транзакцій вашого Monobank, дає можливість побачити витрачені фінанси згрупованих різним чином Технічна інформація: .NET 7, DDD/MediatR, EF Core, Automapper, Quartz, Azure KeyVault.

Цей інструмент стане в нагоді тим, хто хоче мати зручний та наочний доступ до своїх фінансових даних, з можливістю аналізу та керування своїми фінансами ефективніше. Розроблений з використанням сучасних технологій, Outlay забезпечує надійність та безпеку ваших даних, пропонуючи інтуїтивно зрозумілий інтерфейс та функціональні можливості для слідкування за своїми особистими фінансами.

Історія транзакцій:

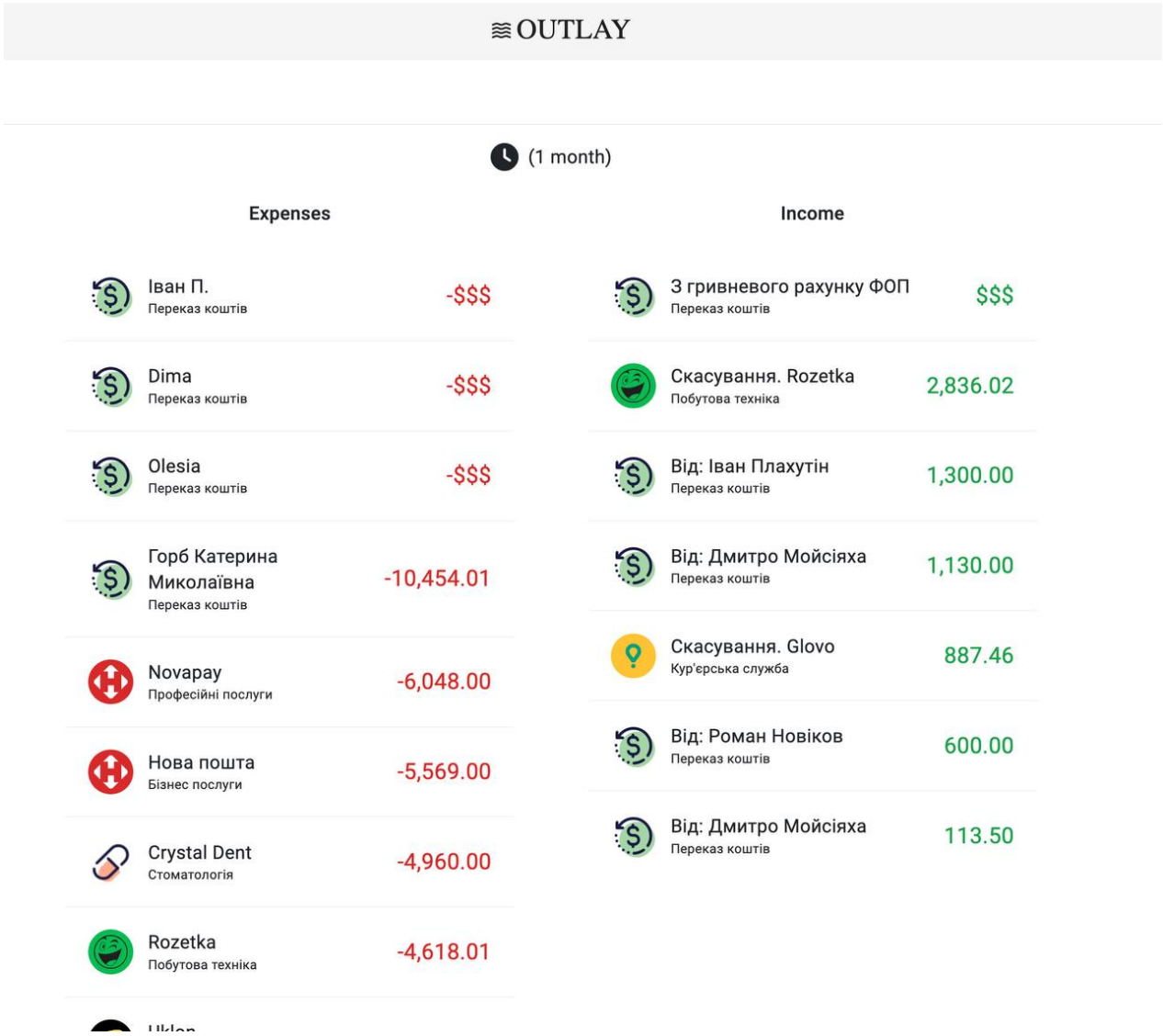


Сторінка з затратами і прибутками за 15 днів та 1 місяць.:

OUTLAY

(15 days)

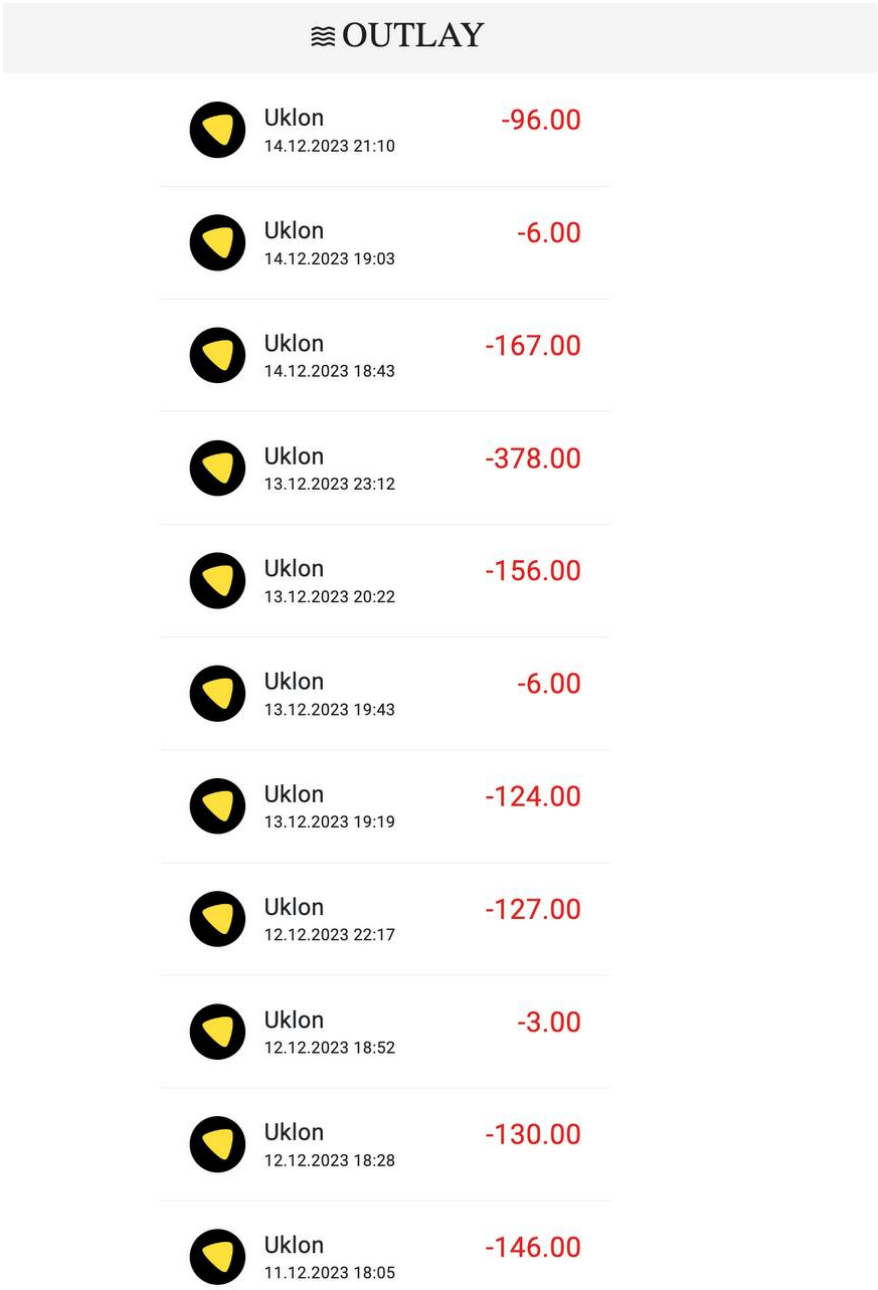
Expenses			Income		
Dima	Переказ коштів	-$$$	З гривневого рахунку ФОП	Переказ коштів	$$$
Olesia	Переказ коштів	-$$$	Скасування. Rozetka	Побутова техніка	1,782.01
Нова пошта	Бізнес послуги	-5,569.00	Від: Іван Плахутін	Переказ коштів	1,220.00
Rozetka	Побутова техніка	-3,564.01	Від: Дмитро Мойсіяха	Переказ коштів	1,130.00
Дарина В.	Переказ коштів	-3,400.00	Скасування. Glovo	Кур'єрська служба	887.46
Jazz Coffee	Фаст-фуд	-2,263.00	Від: Роман Новіков	Переказ коштів	600.00
Glovo	Кур'єрська служба	-2,237.49			
PRTMN COURIERSERV W	Кур'єрська служба	-2,200.26			
Uklon	Таксі	-2,121.00			



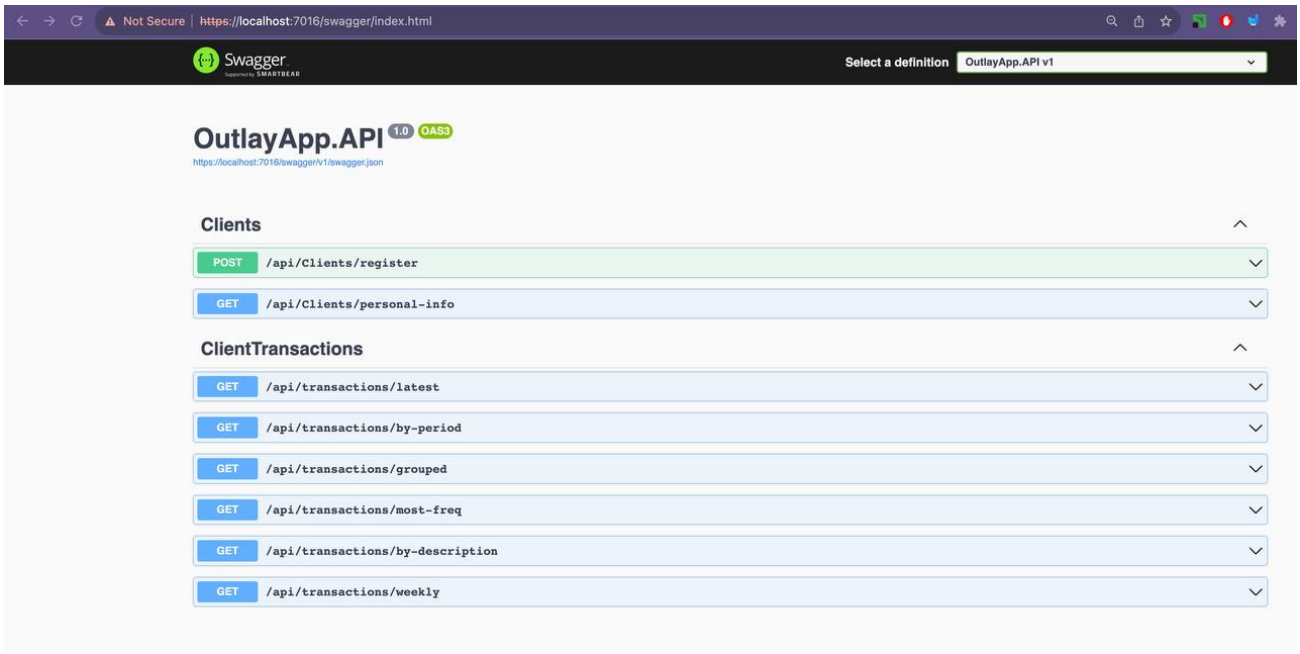
3.5.2 Статистика останніх витрат/прибутків по конкретному імені транзакції та сторінка свагеру

OUTLAY

З гривневого рахунку ФОП 14.12.2023 20:27	$,000.00
З гривневого рахунку ФОП 13.12.2023 19:24	$,000.00
З гривневого рахунку ФОП 10.12.2023 14:07	10,000.00
З гривневого рахунку ФОП 19.10.2023 19:55	20,000.00



Сторінка свагеру:



Висновки

У цій роботі був розроблений веб-додаток на платформі .NET Core для детального контролю фінансових витрат для банкінгу Monobank. Для цього був проведений огляд платформи .NET Core(історія, значення платформи та її характеристики, недоліки та переваги, особливості та можливості), також ми порівняли цю платформу з іншими лптформами, фремвормами та мовами. Для кращого розуміння ми розглянули основні принципи фінансового контролю та сам банкінг Monobank, історію та основні послуги, попри це все порівняли з Приват24. При виборі складових при підготовці до створення додатку було використано .NET 8, також ми змоделювали програмне забезпечення завдяки DDD (розглянули зміни напрямку розробок програмного забезпечення, основні принципи, переваги та приклади успішного впровадження DDD), після всього цього розглянули MediatR та CQRC. НА стадії роектування та розробки додатку було визначено важливість використання Entity Framework, Automapper та Quartz.NET. Далі було розглянуте рішення керування клювачи Azureю Настцпним кроком стало сворення проєкту Outlay App, що відстежуватме статистику та деталі транзакцій Monobank, і також було наочно показана робота програми, що висвітлювала історію транзакцій, сторінку з затратами та прибутками за 15 днів та

1 місяць, також була представлена статистика останніх витрат та прибутків по конкретному імені транзакцій та сторінка свагеру

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Deep Dive into .NET Core Development: Unleashing the Power of Modern Apps. *Radixweb*. URL: https://radixweb.com/introduction-to-dotnet-core (date of access: 28.12.2023).
2. Stack Overflow Developer Survey 2021. *Stack Overflow*. URL: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks (date of access: 31.12.2023).
3. ІПС ЛІГА:ЗАКОН - система пошуку, аналізу та моніторингу нормативно-правової бази. *ÐÐ½ÑÐ¾ÑÐ¼Ð°ÑÑÐ¹Ð½Ð¾-Ð¿ÑÐ°Ð²Ð¾Ð²Ð°Ñ Ð¸ÑÑÐµÐ¼Ð° LIGA:ZAKON*. URL: https://ips.ligazakon.net/document/JF1K400A?an=44 (дата звернення: 01.01.2024).
4. *Журнал Державне управління: удосконалення та розвиток - наукове фахове видання з питань державного управління*. URL: http://www.dy.nayka.com.ua/pdf/8_2018/111.pdf (дата звернення: 01.01.2024).
5. Як взяти кредит в Монобанку. Які проценти і пільговий період. *Monobank*. URL: https://mobanking.com.ua/uk/credit/ (дата звернення: 01.01.2024).
6. Про банк | monobank. *monobank – мобільний банк*. URL: https://www.monobank.ua/about (дата звернення: 01.01.2024).
7. Як працює Monobank: свіжий огляд мобільного банку без відділень. *PaySpace Magazine*. URL: https://psm7.com/uk/mobile-payments/monobank-vyxodit-iz-beta-kak-rabotaet-pervyj-mobilnyj-bank-v-ukraine.html (дата звернення: 01.01.2024).
8. Монобанк: що це за банк, принцип роботи, умови, тарифи, відгуки про Monobank Монобанк: що це за банк, принцип роботи, умови, тарифи, відгуки про Monobank. *Кредити 24/7*. URL: https://money-bank.top/monobank/.
9. Битва мобільних банків: monobank vs Приват24. *finance.ua*. URL: https://

finance.ua/ua/cards/bitva-mobilnyh-bankov-monobank-vs-privat24 (дата звернення: 01.01.2024).

10. Андрій Жуматій. MONOBANK ПЕРШИЙ БАНК БЕЗ ВІДДІЛЕНЬ. *finance.ua*. URL: https://new.finance.ua/ua/30-rokiv-nezalezhnosti/monobank.
11. Secure your .NET cloud apps with rootless Linux Containers - .NET Blog. *.NET Blog*. URL: https://devblogs.microsoft.com/dotnet/securing-containers-with-rootless/ (date of access: 03.01.2024).
12. Performance Improvements in .NET 8 - .NET Blog. *.NET Blog*. URL: https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-8/ (date of access: 03.01.2024).
13. Introducing .NET Aspire: Simplifying Cloud-Native Development with .NET 8 - .NET Blog. *.NET Blog*. URL: https://devblogs.microsoft.com/dotnet/introducing-dotnet-aspire-simplifying-cloud-native-development-with-dotnet-8/ (date of access: 03.01.2024).
14. Announcing F# 8 - .NET Blog. *.NET Blog*. URL: https://devblogs.microsoft.com/dotnet/announcing-fsharp-8/ (date of access: 03.01.2024).
15. Debugging Enhancements in .NET 8 - .NET Blog. *.NET Blog*. URL: https://devblogs.microsoft.com/dotnet/debugging-enhancements-in-dotnet-8/ (date of access: 03.01.2024).
16. What's new in C# 12 - C# Guide - C#. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-12 (date of access: 03.01.2024).
17. Вивільнення потужності DDD. Вичерпний посібник із підходу, орієнтованого на домен, у розробці програмного забезпечення - Sna pwix. *Sna pwix*. URL: https://snapwix.com/unleashing-the-power-of-ddd-a-comprehensive-guide-to-a-domain-focused-approach-in-software-development/ (дата звернення: 03.01.2024).
18. Омер Абай. MediatR. *Medium*. URL: https://omerabay1.medium.com/

mediatr-795155841b27.

19. Yakimov I. Why Do We Need MediatR?. *CodeProject - For those who code*. URL: https://www.codeproject.com/Articles/5317666/Why-Do-We-Need-MediatR (date of access: 04.01.2024).
20. Why use MediatR? 3 reasons why and 1 reason not. *CodeOpinion*. URL: https://codeopinion.com/why-use-mediatr-3-reasons-why-and-1-reason-not/ (date of access: 04.01.2024).
21. Learn Entity Framework Core - Getting Started EF Core Tutorial. *Learn Entity Framework Core - Getting Started EF Core Tutorial*. URL: https://www.learnentityframeworkcore.com/#why-use-an-orm (date of access: 04.01.2024).
22. Overview of Entity Framework Core - EF Core. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/ef/core/ (date of access: 04.01.2024).
23. Karia R. Entity Framework Core. *Entity Framework Tutorial*. URL: https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx (date of access: 04.01.2024).
24. Miskovic J. What is AutoMapper? - The Definite Guide – Josip Miskovic. *Josip Miskovic*. URL: https://josipmisko.com/posts/automapper-guide (date of access: 05.01.2024).
25. Getting Started Guide – AutoMapper documentation. *AutoMapper – AutoMapper documentation*. URL: https://docs.automapper.org/en/stable/Getting-started.html (date of access: 05.01.2024).
26. How to work with AutoMapper in C#. *InfoWorld*. URL: https://www.infoworld.com/article/3192900/how-to-work-with-automapper-in-csharp.html (date of access: 05.01.2024).
27. What Is AutoMapper In ASP.NET Core. *C# Corner - Community of Software and Data Developers*. URL: https://www.c-sharpcorner.com/article/what-is-

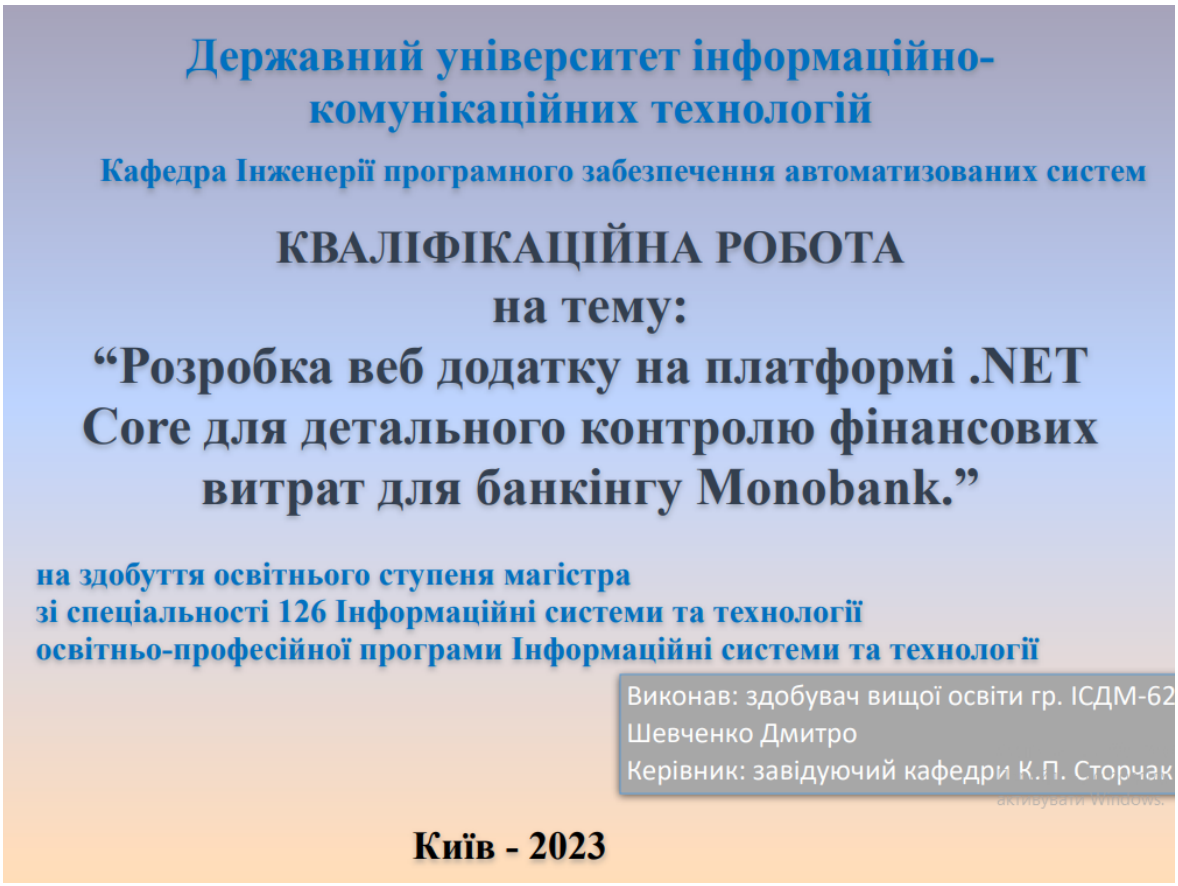
automapper-in-asp-net-core/ (date of access: 05.01.2024).

28. AutoMapper – AutoMapper documentation. *AutoMapper – AutoMapper documentation*. URL: https://docs.automapper.org/en/stable/#:~:text=A%20convention-based%20object-object,up%20source%20to%20destination%20values. (date of access: 05.01.2024).
29. Quartz.NET Features | Quartz.NET. *Home | Quartz.NET*. URL: https://www.quartz-scheduler.net/features.html (date of access: 05.01.2024).
30. Quartz 3 Quick Start | Quartz.NET. *Home | Quartz.NET*. URL: https://www.quartz-scheduler.net/documentation/quartz-3.x/quick-start.html (date of access: 05.01.2024).
31. Azure Key Vault Overview - Azure Key Vault. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/azure/key-vault/general/overview (date of access: 05.01.2024).
32. What is Azure Key Vault?. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/azure/key-vault/general/basic-concepts (date of access: 05.01.2024).

ДОДАТКИ

ДОДАТОК А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ


Державний університет інформаційно-комунікаційних технологій
Кафедра Інженерії програмного забезпечення автоматизованих систем
КВАЛІФІКАЦІЙНА РОБОТА
на тему:
“Розробка веб додатку на платформі .NET Core для детального контролю фінансових витрат для банкінгу Monobank.”
на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології
Виконав: здобувач вищої освіти гр. ІСДМ-62
Шевченко Дмитро
Керівник: завідуючий кафедри К.П. Сторчак
Київ - 2023


Outlay App
Outlay — це проєкт, створений для відстеження статистики та деталей транзакцій вашого Monobank, дає можливість побачити витрачені фінанси згрупованих різним чином
Технічна інформація: .NET 7, DDD/MediatR, EF Core, Automapper, Quartz, Azure KeyVault.
Додано від себе:
Цей інструмент стане в нагоді тим, хто хоче мати зручний та наочний доступ до своїх фінансових даних, з можливістю аналізу та керування своїми фінансами ефективніше.
Розроблений з використанням сучасних технологій, Outlay забезпечує надійність та безпеку ваших даних, пропонуючи інтуїтивно зрозумілий інтерфейс та функціональні можливості для слідкування за своїми особистими фінансами.

.NET 8

.NET — це безкоштовна кросплатформна платформа розробника з відкритим вихідним кодом для створення багатьох різних типів програм.

Завдяки .NET ви можете використовувати кілька мов, редакторів і бібліотек для створення веб-сайтів, мобільних пристроїв, комп'ютерів, ігор, Інтернету речей тощо.

.NET CLR

C# — проста, сучасна, об'єктно-орієнтована та безпечна для типів мова програмування.
F# — це мова програмування, яка дозволяє легко писати стислий, надійний і продуктивний код.
Visual Basic — це доступна мова з простим синтаксисом для створення безпечних типів об'єктно-орієнтованих програм.

DDD/MediatR

Доменно-кероване проектування (DDD) — це основний підхід до розробки програмного забезпечення, який зосереджується на моделюванні програмного забезпечення для відповідності домену відповідно до вхідних даних експертів із цього домену.

У рамках доменно-орієнтованого проектування структура та мова програмного коду (імена класів, методи класів, змінні класу) мають відповідати бізнес-домену. Наприклад: якщо програмне забезпечення обробляє заявки на позику, воно може мати такі класи, як «заявка на позику», «клієнти» та такі методи, як «прийняти пропозицію» та «зняти».

Що таке MediatR?
MediatR — це пакет NuGet для програм .NET, який допомагає реалізувати шаблон Mediator, який можна використовувати як корисний інструмент для реалізації CQRS у програмі .NET.

Патерн MediatR допомагає зменшити пряму залежність між кількома об'єктами та зробити їх спільною за допомогою MediatR.

Queries

```
[HttpGet("by-period")]
Dmytro Shevchenko
public async Task<IActionResult> GetTransactionsByPeriod(Guid clientCardId, DateTime? dateFrom, DateTime? dateTo,
    CancellationToken cancellationToken)
{
    var command = new GetClientTransactionsQuery(clientCardId, dateFrom, dateTo);
    var result = await _mediator.Send(command, cancellationToken);
    return result.IsSuccess ? Ok(result.Value) : BadRequest(result.Error);
}
```

```
Dmytro Shevchenko
public class GetClientTransactionsQueryHandler : IQueryHandler<GetClientTransactionsQuery, List<ClientTransactionDto>>
{
    private readonly IClientTransactionRepository _clientTransactionRepository;
    private readonly IMapper _mapper;

    Dmytro Shevchenko
    public GetClientTransactionsQueryHandler(IClientTransactionRepository clientTransactionRepository, IMapper mapper)
    {
        _clientTransactionRepository = clientTransactionRepository;
        _mapper = mapper;
    }

    Dmytro Shevchenko
    public async Task<Result<List<ClientTransactionDto>>> Handle(GetClientTransactionsQuery request,
        CancellationToken cancellationToken)
    {
        var (dateFrom, dateTo) = TransactionsPeriodHelper
            .GetMonobankTransactionsPeriod(request.DateFrom, request.DateTo);

        var transactions = await _clientTransactionRepository.GetByPeriod(request.ClientCardId, dateFrom, dateTo, cancellationToken);
        return _mapper.Map<List<ClientTransactionDto>>(transactions);
    }
}
```


Commands

```
[HttpGet("latest")]
Dmytro Shevchenko
public async Task<IActionResult> FetchLatestTransactions(string externalCardId, CancellationToken cancellationToken)
{
    var command = new FetchLatestTransactionsCommand(externalCardId);
    var result = await _mediator.Send(command, cancellationToken);
    return result.IsSuccess ? Ok() : BadRequest(result.Error);
}
```

```
public class FetchLatestTransactionsCommandHandler : ICommandHandler<FetchLatestTransactionsCommand>
{
    Properties

    Dmytro Shevchenko
    public FetchLatestTransactionsCommandHandler(IHttpClientFactory factory, IClientCardsRepository cardsRepository,
        IClientTransactionRepository transactionRepository, IClientRepository clientRepository, IUnitOfWork unitOfWork,
        ISender sender)
    {
        _unitOfWork = unitOfWork;
        _sender = sender;
        _cardsRepository = cardsRepository;
        _transactionRepository = transactionRepository;
        _clientRepository = clientRepository;
        _httpClient = factory.CreateClient(MonobankConstants.HttpClient);
    }

    Dmytro Shevchenko
    public async Task<Result> Handle(FetchLatestTransactionsCommand request, CancellationToken cancellationToken)
    {
        var clientCard = await _cardsRepository.GetByExternalId(request.ExternalCardId, cancellationToken);
        if (clientCard is null)
            return Result.Failure(new Error("ClientCard.NotFound",
                $"No client card with External Id {request.ExternalCardId}"));

        long unixTimeFrom;
        var latest = await _transactionRepository.GetLatest(clientCard.Id, cancellationToken);
        if (latest is null)
            unixTimeFrom = DateTimeOffset.Now.AddDays(-MaxDaysPeriod).ToUnixTimeSeconds();
        else
            unixTimeFrom = DateTimeOffset.Now - latest.DateOccured
                        < TimeSpan.FromDays(MaxDaysPeriod)
                ? ((DateTimeOffset)latest!.DateOccured).ToUnixTimeSeconds() + 1
                // because we dont want to take the existing record from monobank api
                : DateTimeOffset.Now.AddDays(-MaxDaysPeriod).ToUnixTimeSeconds();

        var unixTimeTo = DateTimeOffset.Now.ToUnixTimeSeconds();
        var url = BuildUrl(clientCard.ExternalCardId, unixTimeFrom, unixTimeTo);
        var client = await _clientRepository.GetById(clientCard.ClientId, cancellationToken);
        _httpClient.DefaultRequestHeaders.Add(MonobankConstants.TokenHeader, client.PersonalToken);

        var result = await _httpClient.GetAsync(url, cancellationToken);
        var monobankTransactions = (await result.Content.ReadFromJsonAsync<IEnumerable<MonobankTransaction>>(
```

EF Core

Entity Framework (EF) Core — це легка, розширювана кросплатформна версія популярної технології доступу до даних Entity Framework із відкритим кодом.

EF Core може служити об'єктно-реляційним картографом (O/RM), який:

Дозволяє розробникам .NET працювати з базою даних за допомогою об'єктів .NET. Усуває потребу в більшості коду доступу до даних, який зазвичай потрібно писати.

```
public class ClientRepository : IClientRepository
{
    private readonly OutlayContext _context;

    Dmytro Shevchenko
    public ClientRepository(OutlayContext context)
    {
        _context = context;
    }

    0+1 usages  Dmytro Shevchenko
    public Task AddAsync(Client client, CancellationToken cancellationToken = default)
    {
        return _context.AddAsync(client, cancellationToken).AsTask();
    }

    Dmytro Shevchenko
    public void Update(Client client, CancellationToken cancellationToken = default)
    {
        _context.Update(client);
    }

    0+1 usages  Dmytro Shevchenko
    public Task<Client> GetByPersonalToken(string token, CancellationToken cancellationToken = default)
    {
        return _context.Clients.FirstOrDefaultAsync(x => x.PersonalToken == token, cancellationToken)!;
    }

    0+1 usages  Dmytro Shevchenko
    public Task<Client> GetById(Guid clientId, CancellationToken cancellationToken = default)
    {
        return _context.Clients.FirstOrDefaultAsync(x => x.Id == clientId, cancellationToken)!;
    }

    0+1 usages  Dmytro Shevchenko
    public Task<Client> GetByIdWithCards(Guid clientId, CancellationToken cancellationToken = default)
    {
        return _context.Clients.Include(x => x.Cards).FirstOrDefaultAsync(x => x.Id == clientId, cancellationToken)
    }
}
```

Automapper

AutoMapper використовує API вільної конфігурації для визначення стратегії відображення об'єкт-об'єкт. AutoMapper використовує алгоритм зіставлення на основі конвенцій, щоб зіставити вихідні та цільові значення. AutoMapper орієнтований на сценарії проектування моделей, щоб зводити моделі складних об'єктів до DTO та інших простих об'єктів, чий дизайн краще підходить для серіалізації, зв'язку, обміну повідомленнями або просто антикорупційного рівня між доменом і прикладним рівнем.

```
public class ClientTransactionsProfile : Profile
{
    Dmytro Shevchenko
    public ClientTransactionsProfile()
    {
        CreateMap<GroupedTransaction, ClientTransactionsGroupedResponse>()
            .ConvertUsing<ClientTransactionConverter>();

        CreateMap<ClientTransaction, ClientTransactionByDescriptionResponse>()
            .ForMember(x => x.DateOccured, opt => opt.MapFrom(x => $"{x.DateOccured:g}"))
            .ForMember(x => x.Name, opt => opt.MapFrom(x => x.Description));

        CreateMap<ClientTransaction, ClientTransactionDto>()
            .ConvertUsing<ClientTransactionRawConverter>();
    }
}
```

```
public class ClientTransactionConverter : ITypeConverter<GroupedTransaction, ClientTransactionsGroupedResponse>
{
    private readonly OutlayInMemoryContext _inMemoryContext;
    private readonly ILogoReferenceRepository _logoReferenceRepository;

    Dmytro Shevchenko
    public ClientTransactionConverter(OutlayInMemoryContext inMemoryContext, ILogoReferenceRepository logoReferenceRepository)
    {
        _inMemoryContext = inMemoryContext;
        _logoReferenceRepository = logoReferenceRepository;
    }

    Dmytro Shevchenko
    public ClientTransactionsGroupedResponse Convert(GroupedTransaction source,
        ClientTransactionsGroupedResponse destination,
        ResolutionContext context)
    {
        var cat = _inMemoryContext.MccInfos.FirstOrDefault(x => x.Mcc == source.Mcc)!.ShortDescription;
        var name = source.Name.Replace("Оксузаков. ", string.Empty);
        var icon = _logoReferenceRepository.GetByName(name, CancellationToken.None).Result?.Url ?? string.Empty;
        return new ClientTransactionsGroupedResponse
        {
            Name = source.Name,
            Amount = source.Amount,
            Icon = icon,
            Category = cat
        };
    }
}
```

Quartz

Фонові завдання мають вирішальне значення для багатьох програмних програм, і планування їх виконання періодично або в певний час є загальною вимогою. В екосистемі .NET Quartz.NET забезпечує надійну та гнучку структуру для планування таких завдань.

```
public static class DependencyInjection
{
    public static IServiceCollection AddBackgroundJobs(this IServiceCollection services)
    {
        services.AddQuartz(configure =>
        {
            var jobKey = new JobKey(nameof(ProcessOutboxMessagesJob));

            configure.AddJob<ProcessOutboxMessagesJob>(jobKey)
                .AddTrigger(trigger =>
                    trigger
                        .ForJob(jobKey)
                        .WithSimpleSchedule(schedule =>
                            schedule
                                .WithIntervalInSeconds(10)
                                .RepeatForever()));

            configure.UseMicrosoftDependencyInjectionJobFactory();
        });

        services.AddQuartzHostedService();

        return services;
    }
}
```

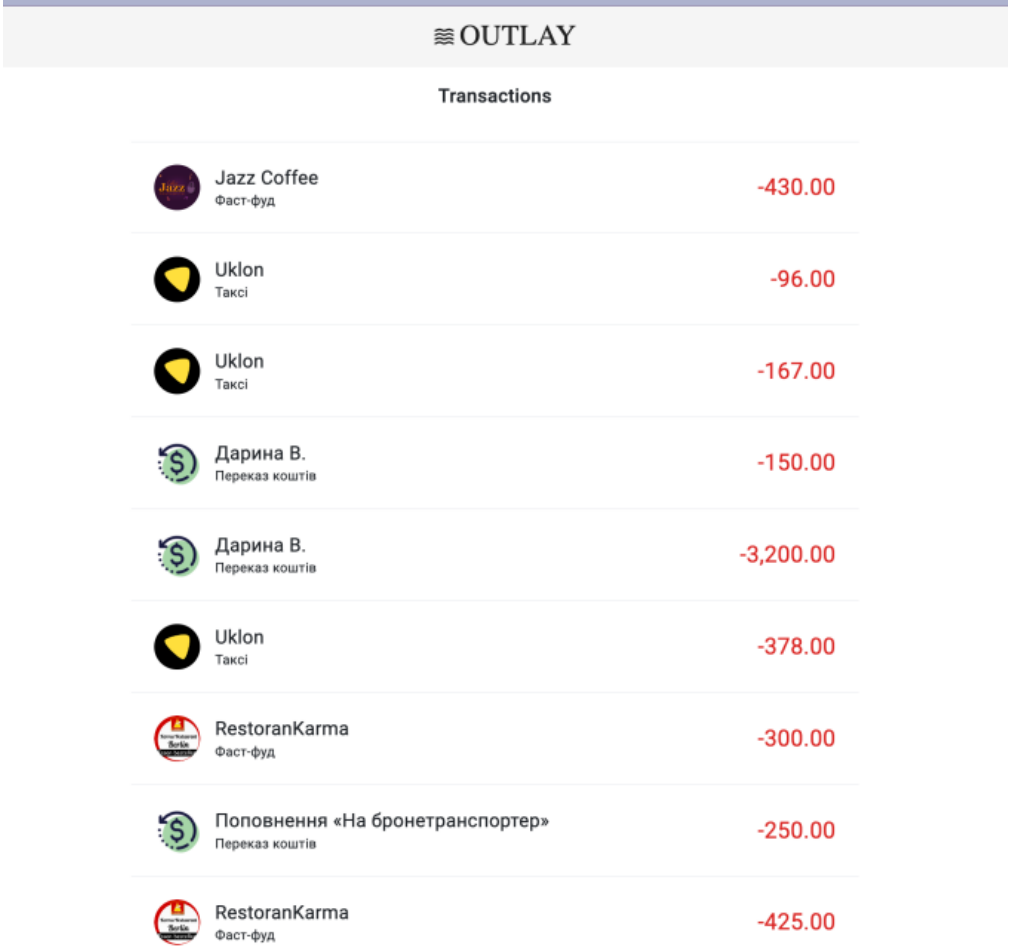
```
public class ProcessOutboxMessagesJob : IJob
{
    private readonly OutlayContext _context;
    private readonly IPublisher _publisher;

    public ProcessOutboxMessagesJob(OutlayContext context, IPublisher publisher)
    {
        _context = context;
        _publisher = publisher;
    }

    public async Task Execute(IJobExecutionContext context)
    {
        var messages = await _context
            .Set<OutboxMessage>()
            .Where(m => m.ProcessedOnUtc == null)
            .Take(20)
            .ToListAsync(context.CancellationToken);

        foreach (var message in messages)
        {
            var domainEvent = JsonConvert.DeserializeObject<IDomainEvent>(
                message.Content,
                new JsonSerializerSettings { TypeNameHandling = TypeNameHandling.All });

            if (domainEvent is null)
                continue;

            await _publisher.Publish(domainEvent, context.CancellationToken);
            message.ProcessedOnUtc = DateTimeOffset.Now.ToUnixTimeSeconds();
            await _context.SaveChangesAsync();
        }
    }
}
```

Azure KeyVault

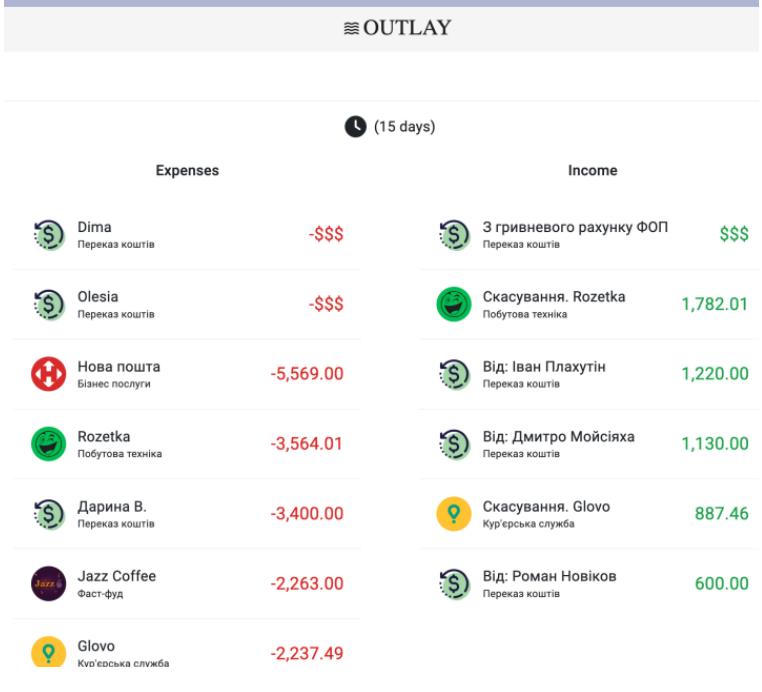
Azure Key Vault — це одне з кількох рішень керування ключами в Azure, яке допомагає вирішити такі проблеми: Керування секретами — Azure Key Vault можна використовувати для безпечного зберігання та суворого контролю доступу до маркерів, паролів, сертифікатів, ключів API та інших секретів.

```
public static class DependencyInjection
{
    public static ConfigurationManager AddKeyVault(this ConfigurationManager configuration, bool isProduction)
    {
        if (!isProduction)
            return configuration;

        var vaultUrl = configuration[KeyVaultConstants.Url];
        var tenantId = configuration[KeyVaultConstants.TenantId];
        var clientId = configuration[KeyVaultConstants.ClientId];
        var secretId = configuration[KeyVaultConstants.ClientSecretId];

        var credential = new ClientSecretCredential(tenantId, clientId, secretId);
        var client = new SecretClient(new Uri(vaultUrl!), credential);
        configuration.AddAzureKeyVault(client, new AzureKeyVaultConfigurationOptions());

        return configuration;
    }
}
```

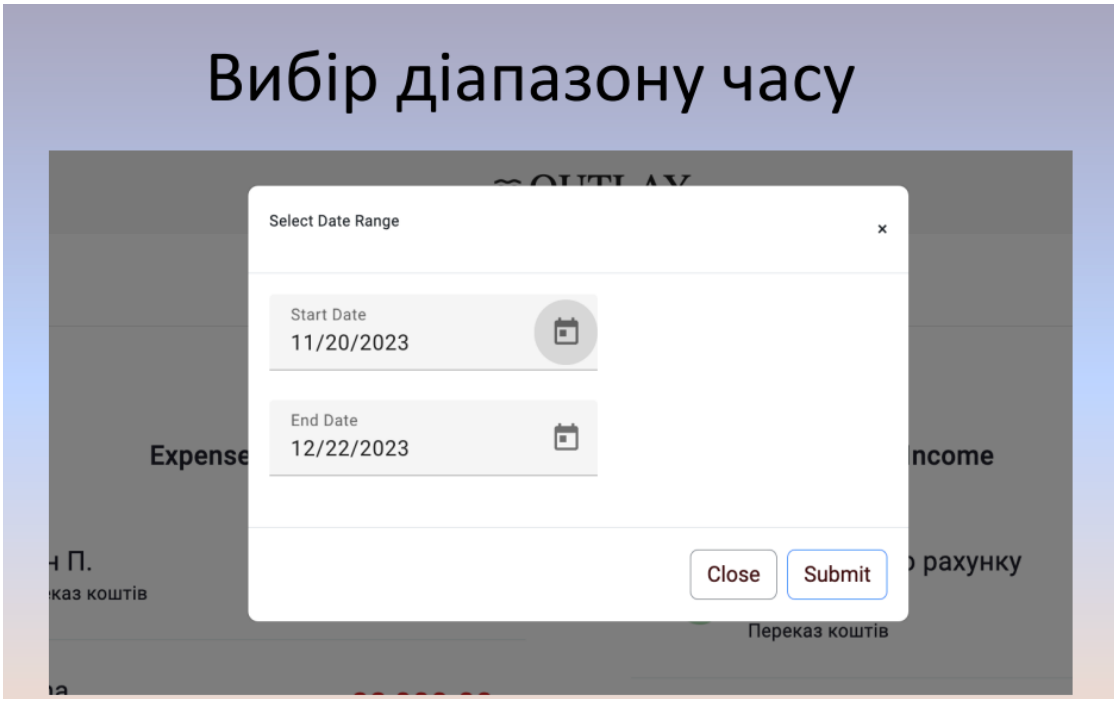
Історія транзакцій



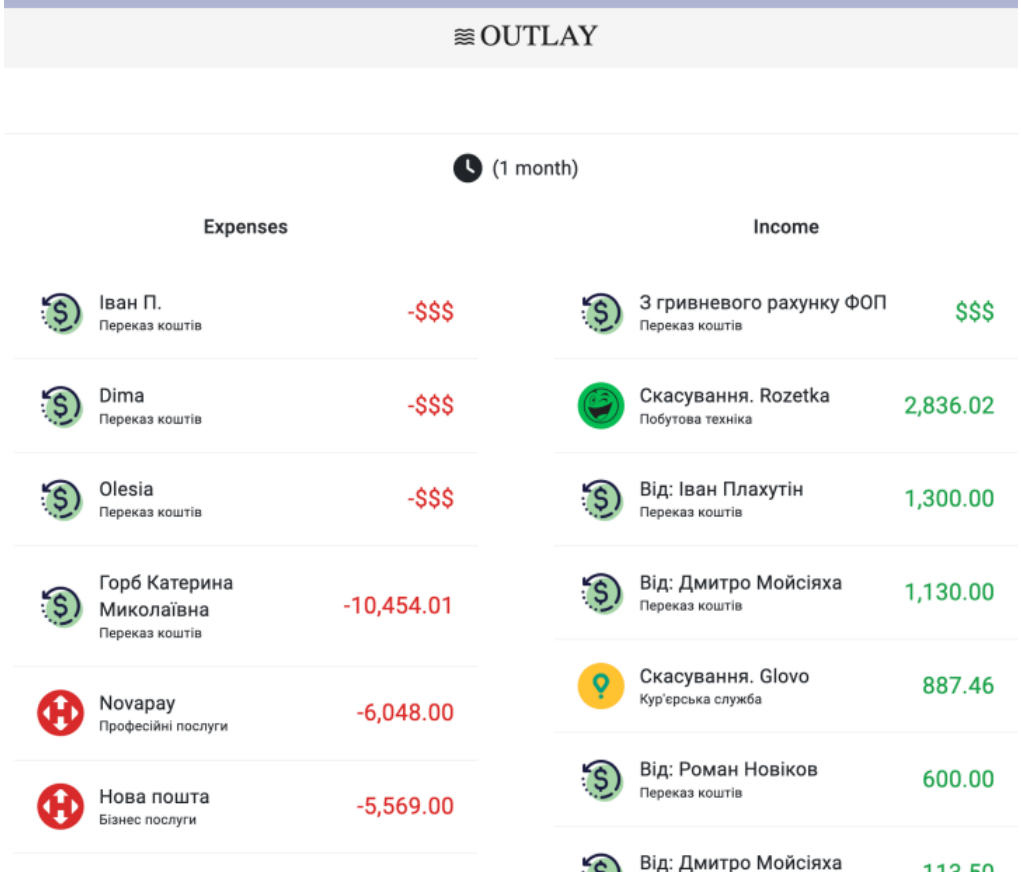
Сторінка з затратами і прибутками згрупованими по імені (15 днів)



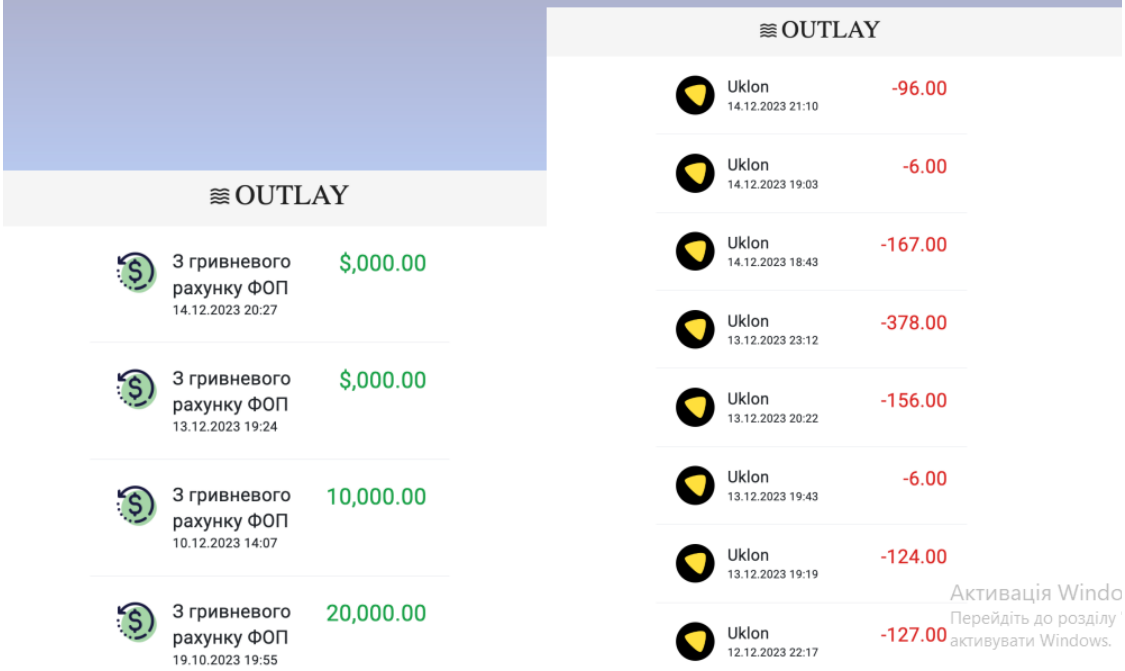
Вибір діапазону часу



Сторінка з затратами і прибутками згрупованими по імені (1 місяць)



Статистика останніх витрат/прибутків по конкретному імені транзакції



Сторінка свагеру