БАКАЛАВРСЬКА РОБОТА

БР.КІ-36.00.00.000 ПЗ

Група КІ-21-2

Михайлишин Юлія

2025

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Факультет інформаційних технологій
Кафедра комп'ютерних систем і мереж

Михайлишин Юлія Петрівна

УДК 004.66

БАКАЛАВРСЬКА РОБОТА

Розробка програми для організації обліку роботи пункту обміну валюти засобами C# з ASP .NET та Vue.js

Комп'ютерна інженерія
(назва освітньої програми)

123 - Комп'ютерна інженерія
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____ *Михайлишин Ю. П.*
(підпис, ініціали та прізвище здобувача)

Науковий керівник _____ *Мойсеєнко О. В., к.т.н , доц.*
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри КСМ

д.т.н.,професор _____ *С.І. Мельничук*
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025 рік

Івано-Франківський національний технічний університет нафти і газу

(повне найменування вищого навчального закладу)

Факультет *інформаційних технологій*

Кафедра *комп'ютерних систем і мереж*

Освітній ступінь *бакалавр*

Спеціальність *123 – Комп'ютерна інженерія*

(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КСМ

(С.І. Мельничук)

«05» *травня* *2025* року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Михайлишин Юлії Петрівні

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи*) " Розробка програми для організації обліку роботи пункту обміну валюти засобами C# з ASP .NET та Vue.js"*

керівник проекту (роботи) *Мойсеєнко О. В., к.т.н , доц.*

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

затверджені наказом вищого навчального закладу від *05.05.2025 № 275/7*

2. Строк подання студентом роботи *12 червня 2025 р*

3. Вихідні дані до роботи *Матеріали і результати отримані під час проходження переддипломної практики, методичні вказівки, технічна література.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *1. Аналіз предметної області та постановка завдання. 2. Розробка структури програмного забезпечення. 3. Програмна реалізація застосунку.*

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

6. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата

7. Дата видачі завдання *29 січня 2025 р.*

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	*Аналіз предметної області та постановка завдання*	*Лютий, 2025*	
2	*Розробка структури застосунку*	*Березень, 2025*	
3	*Програмна реалізація програмного забезпечення*	*Травень, 2025*	
4	*Оформлення пояснювальної записки*	*Червень, 2025*	
5			

Студент _____ (підпис) *Михайлишин Ю. П.* (прізвище та ініціали)

Керівник роботи _____ (підпис) *Мойсеєнко О. В.* (прізвище та ініціали)

АНОТАЦІЯ

В дипломній роботі було здійснено розробку програми для організації обліку роботи пункту обміну валюти засобами C# з ASP .NET та Vue.js.

Метою роботи є розробка сучасного програмного забезпечення для пункту обміну валюти, яке забезпечить облік операцій, управління валютними курсами, формування звітів і ведення статистики з можливістю віддаленого доступу через вебінтерфейс.

У роботі проведено комплексний аналіз предметної області та визначено вимоги до програмного забезпечення для організації обліку роботи пункту обміну валюти. Розглянуто основні бізнес-процеси, специфіку обліку валютно-обмінних операцій та потребу інтеграції сучасних інформаційних технологій. На основі аналізу сформульовано завдання проєкту й запропоновано створення двох взаємопов’язаних платформ: адміністративного вебінтерфейсу та Telegram-бота для клієнтів. Розроблено архітектуру системи на основі C#, ASP.NET, PostgreSQL, Redis і Vue.js, спроєктовано базу даних, прототипи інтерфейсів і клієнт-серверну взаємодію. Проведено тестування функціональності, якісний аналіз коду та усунено виявлені дефекти.

В результаті створено ефективну, безпечну та масштабовану систему для автоматизації обліку валютно-обмінних операцій із сучасним інтерфейсом та зручною взаємодією для персоналу та клієнтів.

Ключові слова: бот, програмне забезпечення, Telegram, архітектура, база даних, НБУ.

SUMMARY

In the thesis, a program was developed to organize accounting of the work of a currency exchange point using C# with ASP .NET and Vue.js.

The purpose of the work is to develop modern software for a currency exchange point, which will provide accounting of transactions, exchange rate management, report generation and statistics with the possibility of remote access via a web interface.

The work conducted a comprehensive analysis of the subject area and determined the requirements for software for organizing accounting of the work of a currency exchange point. The main business processes, the specifics of accounting for currency exchange transactions and the need to integrate modern information technologies were considered. Based on the analysis, the project tasks were formulated and the creation of two interconnected platforms was proposed: an administrative web interface and a Telegram bot for clients. The system architecture was developed based on C#, ASP.NET, PostgreSQL, Redis and Vue.js, a database, interface prototypes and client-server interaction were designed. Functionality testing, qualitative code analysis and identified defects were eliminated.

As a result, an effective, secure and scalable system for automating the accounting of currency exchange transactions was created with a modern interface and convenient interaction for staff and clients.

Keywords: bot, software, Telegram, architecture, database, NBU.

ЗМІСТ

					БР.КІ-36.00.00.000 ПЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Михайлишин Ю. П.			*Розробка програми для організації обліку роботи пункту обміну валюти засобами C# з ASP .NET та Vue.js*	Літ.	Арк. 3 / Акрушів 73
Перевір.		Мойсеєнко О. В.					
Реценз.		Гарасимів Т. Г.				ІФНТУНГ КІ-21-2	
Н. контр.		Лазорів А.М.					
Затверд.		Мельничук С. І.					

ВСТУП

Сучасні фінансові установи, зокрема пункти обміну валют, потребують ефективних інструментів для автоматизації щоденних операцій, контролю обліку та забезпечення прозорості грошових операцій. У зв'язку зі зростанням обсягів валютно-обмінних транзакцій та вимог до безпеки й швидкості обробки даних, актуальним є створення спеціалізованих програмних рішень, здатних оптимізувати діяльність таких пунктів. Технологічний прогрес дає змогу поєднувати можливості сучасних вебтехнологій, що дозволяє розробляти гнучкі, масштабовані та зручні у використанні системи обліку та керування фінансовими операціями.

Актуальність теми полягає у необхідності впровадження надійного програмного забезпечення для пунктів обміну валюти, яке дозволяє вести точний облік фінансових операцій, контролювати курси валют, формувати звітність та забезпечувати захищене зберігання даних. В умовах жорсткої конкуренції та посилення контролю з боку регуляторних органів ефективне управління процесами валютно-обмінних операцій є важливою складовою безперервної роботи фінансових установ.

Об'єктом дослідження виступає діяльність пункту обміну валюти в частині організації обліку грошових операцій, моніторингу курсів валют і ведення звітності.

Предметом дослідження є методи та засоби створення програмного забезпечення для автоматизації обліку та контролю валютно-обмінних операцій із використанням C# на платформі ASP.NET для серверної частини та Vue.js для клієнтського інтерфейсу.

Метою роботи є розробка сучасного програмного забезпечення для пункту обміну валюти, яке забезпечить облік операцій, управління валютними курсами, формування звітів і ведення статистики з можливістю віддаленого доступу через вебінтерфейс.

Завданням дослідження є аналіз існуючих програмних рішень для валютно-обмінних операцій, розробка архітектури вебдодатку, проектування структури бази даних, реалізація серверної частини на базі ASP.NET, створення інтерактивного клієнтського інтерфейсу за допомогою Vue.js та тестування функціоналу розробленої системи.

Практичне значення роботи полягає у створенні прикладного програмного продукту, який можна впровадити в реальну діяльність пунктів обміну валюти для автоматизації обліку операцій, покращення точності розрахунків, підвищення швидкості обслуговування клієнтів і зменшення ймовірності помилок під час здійснення фінансових операцій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Опис предметної області

Предметна область, пов’язана з організацією роботи пункту обміну валюти, охоплює комплекс процесів, що забезпечують облік, контроль і аналіз операцій з купівлі та продажу іноземних валют за визначеними курсами. У рамках функціонування такого пункту щоденно здійснюється велика кількість транзакцій, які потребують точного фіксування, оперативної обробки та збереження інформації відповідно до чинного законодавства й внутрішніх стандартів фінансового контролю.

Основою предметної області є валютно-обмінні операції, що передбачають купівлю, продаж або обмін однієї валюти на іншу за встановленим курсом із можливістю нарахування комісій. Для забезпечення їх проведення пункт обміну повинен мати актуальні дані про валютні курси, які можуть змінюватися впродовж дня залежно від ринкових умов. Особливе значення в цій сфері має правильне відображення обсягів валют, контроль залишків у касі та формування звітів за результатами операційної діяльності за певний період.

Предметна область передбачає взаємодію касирів, адміністраторів та контролюючих органів, для яких важливо отримувати точну інформацію про всі здійснені операції, наявність готівкових коштів, зафіксовані курси валют та сформовані звіти. Організація цієї роботи вимагає використання надійного програмного забезпечення, здатного забезпечити швидку обробку даних, збереження історії транзакцій, захист від несанкціонованого доступу та можливість віддаленого адміністрування.

Важливим аспектом предметної області є ведення електронного обліку всіх валютно-обмінних операцій із відображенням даних про час, суму, курс, вид валюти та особу, яка здійснила операцію. Це дає змогу не лише забезпечити фінансову прозорість, а й оперативно формувати звітність для внутрішнього

контролю та подання до регуляторних органів. Предметна область також охоплює контроль за залишками валютних коштів у касі та управління курсами валют, які можуть змінюватися за рішенням керівництва відповідно до ринкових тенденцій.

Таким чином, предметна область передбачає автоматизацію процесів фіксації, зберігання, обробки та аналізу даних про валютні операції, курсів валют та касових залишків із можливістю формування звітної документації, що забезпечує ефективну, безпечну та прозору діяльність пункту обміну валюти.

Для забезпечення коректної роботи інформаційної системи обліку операцій пункту обміну валюти та її відповідності чинним нормам і стандартам, необхідно опрацювати низку нормативно-правових, технічних та методичних документів, що регламентують правила здійснення валютно-обмінних операцій, ведення обліку, зберігання й обробки фінансових даних, а також використання інформаційних технологій у цій сфері.

Перш за все, слід врахувати законодавчі документи України, зокрема Закон України «Про валюту і валютні операції», який визначає порядок проведення операцій з іноземною валютою, вимоги до фінансового моніторингу, ведення обліку та контролю за валютними транзакціями. Важливим є також Закон України «Про захист персональних даних», оскільки система обліку може містити дані про клієнтів та співробітників, що потребує дотримання правил захисту конфіденційної інформації.

Особливу увагу необхідно приділити нормативним актам Національного банку України (НБУ), який регулює діяльність пунктів обміну валют, затверджуючи правила їхньої роботи, порядок встановлення та оголошення валютних курсів, правила формування касових звітів і порядок проведення касових операцій. До таких документів належать постанови НБУ про порядок ведення касових операцій, обліку та контролю готівкових коштів, перелік документів для проведення валютно-обмінних операцій, а також вимоги до систем обліку.

Крім того, необхідно враховувати технічну документацію, зокрема міжнародні та національні стандарти в галузі розробки та експлуатації інформаційних систем, такі як ISO/IEC 27001:2013 для забезпечення інформаційної безпеки та ISO/IEC 25010:2011, який встановлює модель якості програмних продуктів. До технічної документації також належать описання архітектури вебдодатків, протоколи захисту даних, методики резервного копіювання та регламенти доступу до фінансових даних.

Наукові праці та методичні матеріали з інформаційних технологій, програмування на C# та розробки вебдодатків із використанням ASP.NET та Vue.js також є важливими, оскільки дозволяють врахувати сучасні підходи до побудови багаторівневих клієнт-серверних систем, оптимізації баз даних та забезпечення швидкодії й безпеки систем обліку.

У межах цієї нормативно-документальної бази можна виділити такі інформативні ознаки (параметри):

- Дата та час проведення операції — необхідні для обліку й звітності.
- Тип операції — купівля, продаж або обмін валюти.
- Вид валюти — визначає валютну пару, з якою проводиться операція.
- Курс валюти на момент операції — фіксується для точності розрахунків і контролю.
- Сума в національній та іноземній валюті — необхідна для грошових звітів і контролю касових залишків.
- Дані касира — дозволяють здійснювати персоніфікований облік операцій.
- Залишки валюти в касі до та після операції — для контролю готівкових коштів.
- Номер та дата формування звітного документа — необхідні для юридичного оформлення операцій.
- Логування подій — забезпечує контроль за діями користувачів у системі.
- Доступ до системи за ролями — передбачає обмеження прав користувачів згідно з їхніми посадовими обов’язками.

Отже, нормативно-правове, технічне та наукове забезпечення даного інформаційного процесу створює комплекс вимог і правил, на яких ґрунтується структура та функціональність розроблюваної системи. Це забезпечує відповідність програмного забезпечення чинному законодавству, галузевим стандартам і технічним регламентам, а також дозволяє впроваджувати її в практичну діяльність пунктів обміну валюти.

У межах даної роботи передбачено реалізацію двох ключових напрямів розробки програмного забезпечення: створення Telegram-бота, який виступає клієнтським інтерфейсом, та вебресурсу, призначеного для адміністративного управління. Telegram-бот забезпечує користувачам максимально простий та інтуїтивно зрозумілий засіб для взаємодії з системою, що дозволяє швидко здійснювати бронювання і відстежувати актуальні курси валют. Водночас вебсайт слугує робочим інструментом для персоналу пункту обміну та власників бізнесу, надаючи їм можливість виконувати широкий спектр адміністративних функцій — від додавання нових валют і регулювання курсів до контролю резервів і обліку фінансових операцій. Поєднання цих двох платформ дозволяє комплексно охопити потреби різних груп користувачів, створюючи для них комфортні умови для виконання щоденних завдань і забезпечуючи ефективну організацію бізнес-процесів.

1.2 Огляд аналогів

Щоб сформувати концепцію програмного продукту, було виконано огляд сучасного програмного забезпечення та технічних рішень, які можуть бути використані для створення вебзастосунку, що автоматизує облік та супровід валютно-обмінних операцій. Розглянуто як спеціалізовані продукти, так і допоміжні засоби розробки, а також готові програмні платформи, здатні частково закривати потреби такого бізнесу.

На сучасному ринку програмного забезпечення для автоматизації бізнес-процесів представлено низку потужних сервісів і платформ, які користуються популярністю у різних категорій користувачів. Серед них можна виділити такі системи, як Creatio, Salesforce та Zoho. Вони пропонують універсальні інструменти для бізнес-аналітики, автоматизації процесів та оптимізації внутрішніх операцій. Подібні платформи дозволяють ефективно організовувати звітність, автоматизувати бізнес-процеси та підтримувати сталу продуктивність у повсякденній роботі компаній.

Окремо варто виділити такі сервіси, як BankGovUa, Minfin, PrivatBank, Binance та OKX, які надають актуальні дані про фінансові ринки, офіційні та комерційні валютні курси, а також аналітику цифрових активів.

Creatio, завдяки широкому функціоналу для low-code розробки, є однією з найпопулярніших платформ для автоматизації бізнесу, пропонуючи готові рішення для управління маркетингом, продажами, клієнтським обслуговуванням та CRM. Вона дозволяє підприємствам оптимізувати свої операційні процеси за допомогою гнучких інструментів для автоматизації та підвищення продуктивності. На рисунку 1.1 продемонстровано головну сторінку налаштувань інтерфейсів у системі Creatio [2].

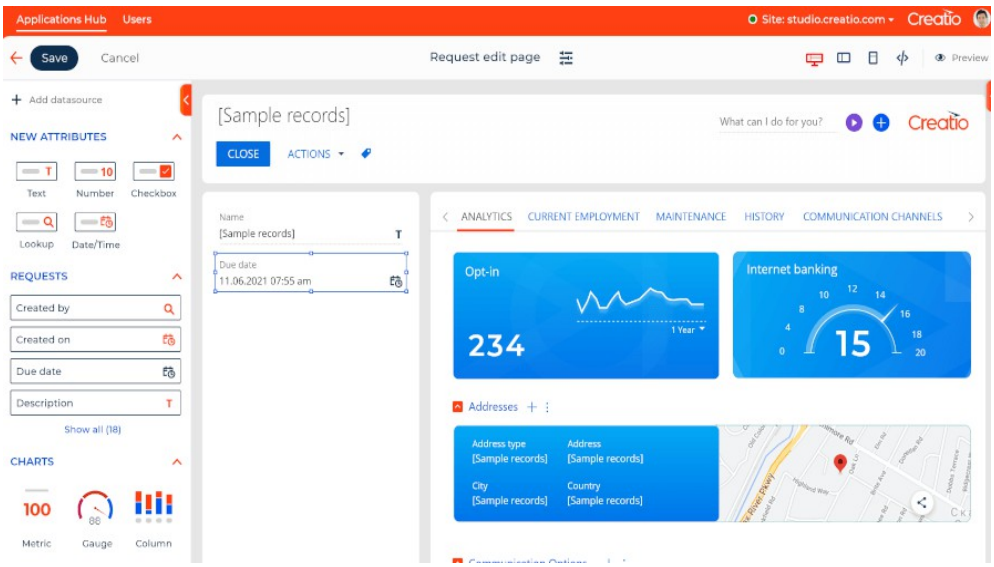

Рисунок 1.1 – Система Creatio

Salesforce по праву вважається однією з найпотужніших CRM-платформ, яка надає комплексні рішення для автоматизації маркетингової діяльності, процесів продажів, підтримки клієнтів та аналітики даних. Платформа допомагає компаніям ефективно організовувати роботу з клієнтами та забезпечувати злагоджене управління бізнес-процесами. Головну сторінку для реєстрації звернень у Salesforce наведено на рисунку 1.2 [3].

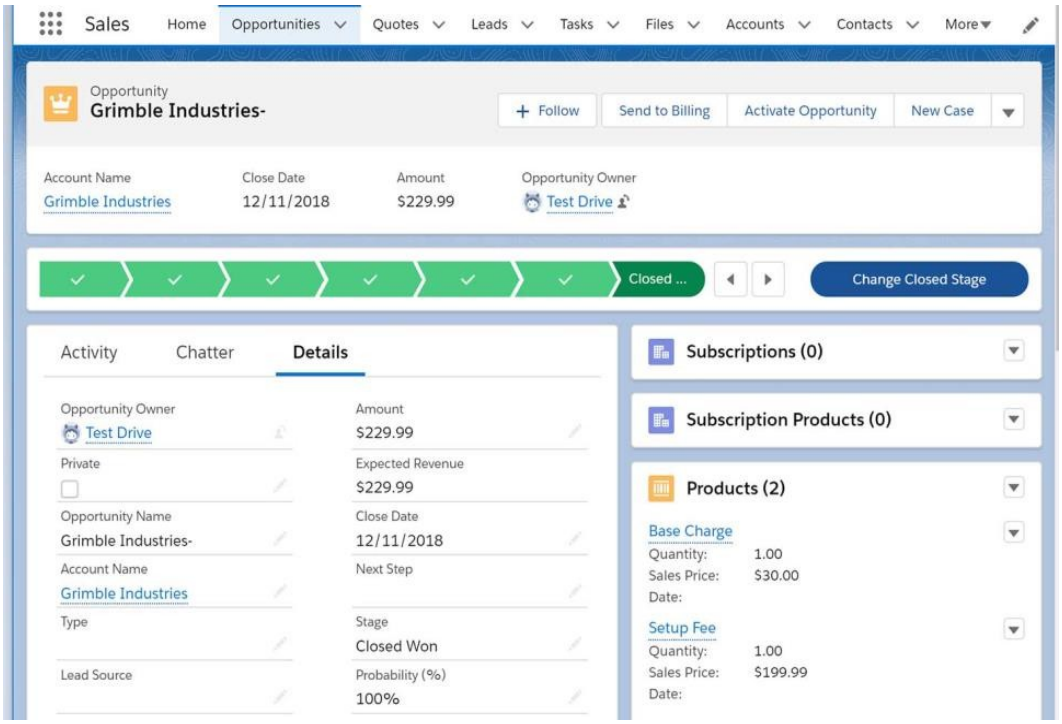

Рисунок 1.2 –Система Salesforce

Платформа Zoho CRM також є інтегрованим рішенням для управління клієнтськими відносинами, надаючи набір інструментів для організації маркетингових кампаній, обліку продажів та підтримки клієнтів. Її можливості охоплюють управління контактами, заявками, угодами та формування аналітичної звітності, що дозволяє підприємствам підвищувати ефективність комерційної діяльності та залучати нових клієнтів. На рисунку 1.3 представлено приклад дашбордів для відслідковування обсягів і кількості продажів у системі Zoho [4].

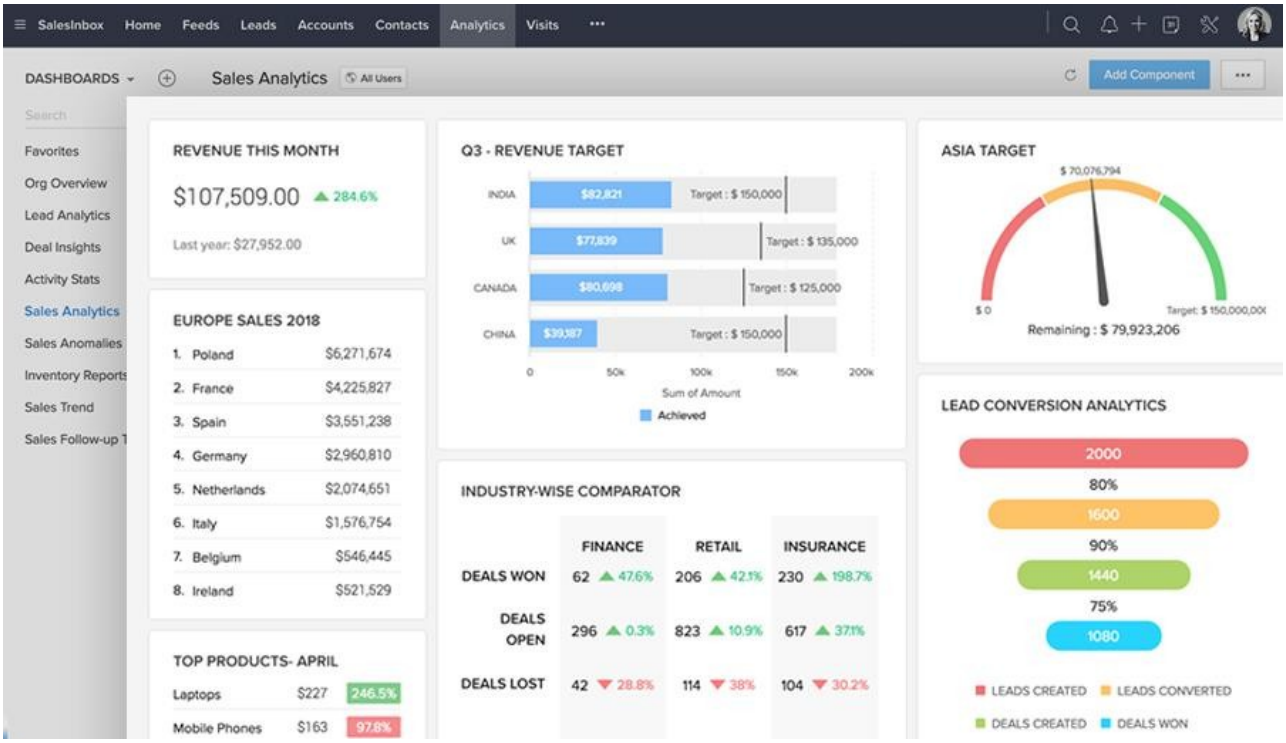

Рисунок 1.3 – Система Zoho

Кожна з цих платформ має свої характерні переваги та певні недоліки. Наприклад, Creatio вирізняється зручним функціоналом для low-code розробки, хоча потребує значних зусиль для повного налаштування системи. Salesforce і Zoho пропонують широкий спектр гнучких можливостей, високий рівень захисту даних та розвинену систему інтеграцій зі сторонніми сервісами. Проте для всіх трьох рішень характерною особливістю є висока вартість впровадження та обслуговування, що часто перевищує фінансові можливості малих і середніх бізнесів, яким багато з представлених функцій можуть бути зайвими.

Серед ресурсів, що забезпечують доступ до актуальної інформації про фінансові ринки, виділяються сервіси BankGovUa, Minfin, PrivatBank, Binance та OKX. Наприклад, Minfin є великим агрегатором фінансових новин і курсових показників обмінних пунктів, що дозволяє аналізувати ринкову ситуацію та стежити за змінами валютних курсів [5]. Офіційний ресурс Національного банку України — BankGovUa — надає офіційні курси валют, які встановлюються регулятором і є базою для здійснення валютних операцій у державі [6].

Найбільший український комерційний банк — PrivatBank — публікує власні курси валют, які зазвичай є найнижчими серед комерційних установ [7]. Binance і OKX — це міжнародні криптовалютні біржі, що дозволяють здійснювати аналіз цифрових активів, що відкриває можливість прогнозувати курсові коливання на основі поведінки криптовалютного ринку [8][9].

Таблиця 1.1 – Порівняльна таблиця аналогів

Функціонал	Creatio	Salesforce	Zoho
Створення транзакції	+	+	+
Додавання валют	+	+	+
Створення резервів	+	+	+
Перегляд резервів і транзакцій	+	+	+
Створення курсів	+	+	+
Створення користувача	+	+	+
Телеграм бот для взаємодії з клієнтом	-	-	-
Зручність використання	+	-	-
Легкість занурення у розробку	-	-	-
Low-code підтримка	+	-	-
Первинне налаштування	-	-	-
Безкоштовне використання	-	-	-

Усі згадані сервіси та платформи відзначаються високим рівнем надійності, оперативністю оновлення даних і широким охопленням фінансових ринків. Їхні функції дозволяють як підприємцям, так і приватним користувачам своєчасно отримувати актуальну інформацію про фінансову ситуацію та формувати обґрунтовані рішення щодо проведення валютних операцій чи інвестування. Крім

того, інтерфейси більшості цих сервісів є зрозумілими та простими у використанні, що сприяє комфортній роботі з фінансовими даними.

1.3 Постановка завдання

У межах цієї роботи передбачено створення програми для організації обліку роботи пункту обміну валюти. Система забезпечує комплексний облік валютно-обмінних операцій, контроль актуальних курсів валют, бронювання заявок клієнтів та формування аналітичної звітності. Основна мета полягає в підвищенні ефективності роботи пункту обміну валют шляхом упровадження сучасного програмного забезпечення, здатного забезпечити зручну та надійну взаємодію між клієнтами, касирами, менеджерами та власником бізнесу.

У рамках роботи визначено такі основні задачі:

− розробити програмний комплекс, який дозволяє організувати облік щоденних операцій пункту обміну валют;

− реалізувати функціонал для реєстрації валютно-обмінних транзакцій із можливістю збереження їх в історії та формування аналітики;

− забезпечити можливість бронювання заявок клієнтами та відстеження їх статусів у реальному часі;

− впровадити засіб для контролю резервів валюти в обмінному пункті з можливістю віддаленого доступу до даних про залишки та актуальні курси валют;

− реалізувати інтеграцію з офіційним джерелом курсів валют — Національним банком України, з подальшим розрахунком комерційного курсу для пункту обміну;

− забезпечити облік доходів пункту обміну та формування звітності про проведені операції та бронювання за задані часові проміжки;

− створити зручний, зрозумілий та багатофункціональний інтерфейс для відображення фінансових та бізнесових даних, доступний з різних пристроїв.

Для реалізації зазначених задач передбачено виконання таких завдань:

− розробка Telegram-бота на основі C# та ASP.NET, який виступатиме основним інтерфейсом для клієнтів, надаючи можливість здійснювати бронювання та переглядати актуальні курси валют;

− впровадження механізму кешування та збереження тимчасових даних за допомогою Redis для підвищення швидкодії системи та забезпечення стабільної роботи бота;

− створення адміністративного вебінтерфейсу на основі шаблону Materio з використанням технологій Vue.js, Vuetify, HTML, SCSS та JavaScript, адаптованого під потреби пункту обміну валют;

− реалізація функціоналу для касирів щодо проведення валютно-обмінних транзакцій, їх обліку та реєстрації в системі;

− створення інтерфейсу для менеджерів із можливістю додавання нових валют, налаштування курсів і параметрів обміну;

− забезпечення можливості бронювання валют клієнтами через адміністративний модуль;

− розробка модулю для формування фінансових звітів про доходи та кількість проведених операцій за визначені проміжки часу;

− організація централізованого збереження даних у реляційній базі PostgreSQL для забезпечення швидкого доступу, надійного зберігання інформації та можливості масштабування системи в подальшому;

− оптимізація системи для роботи з великим обсягом одночасних запитів та забезпечення її надійної роботи в умовах постійного зростання навантаження.

Таким чином, робота передбачає створення багатофункціональної програмної системи, що дозволить організувати ефективний облік та контроль фінансових операцій у пункті обміну валют, забезпечити доступність даних для клієнтів та персоналу і надати керівництву можливість оперативного управління всіма бізнес-процесами.

У даному розділі було проведено детальний аналіз предметної області та визначено основні вимоги до розробки та впровадження програми для організації обліку роботи пункту обміну валюти засобами C# з ASP .NET та Vue.js.

Проведений аналіз предметної області дозволив окреслити ключові особливості організації роботи пунктів обміну валют, їх основні бізнес-процеси, вимоги до обліку та контролю валютно-обмінних операцій, а також сформувати уявлення про необхідність інтеграції сучасних інформаційних технологій у цю сферу. Установлено, що діяльність таких пунктів пов’язана з обробкою великого обсягу фінансових даних, контролем залишків валюти, веденням історії транзакцій, бронюванням коштів клієнтів та формуванням звітів відповідно до чинного законодавства і нормативних актів.

Аналіз наявного програмного забезпечення та технічних рішень показав, що на ринку представлено низку потужних платформ для автоматизації бізнес-процесів, таких як Creatio, Salesforce та Zoho, які мають широкий функціонал і високу надійність, однак потребують значних фінансових ресурсів для впровадження та обслуговування. Також було розглянуто спеціалізовані сервіси, які надають актуальну інформацію про курси валют і цифрових активів, що дозволяє ефективно організовувати аналітику фінансових ринків і приймати обґрунтовані рішення.

На основі проведеного аналізу сформульовано завдання проєкту зі створення власної програмної системи для обліку діяльності пункту обміну валюти. Запропоновано реалізувати дві взаємопов’язані платформи: Telegram-бота для клієнтського сервісу та адміністративний вебінтерфейс для персоналу та керівництва. Визначено основні функціональні можливості системи, перелік користувацьких ролей і технічні засоби, що будуть використані для розробки, зокрема мови програмування C#, ASP.NET, технології Vue.js та реляційну СУБД PostgreSQL. У межах поставлених завдань окреслено функціональні модулі та операції, які повинна підтримувати система для забезпечення ефективної, прозорої та безпечної роботи пункту обміну валют.

2 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз бізнес-процесів

Процес створення програмного забезпечення для організації обліку роботи пункту обміну валюти супроводжується моделюванням відповідних бізнес-процесів, що дозволяє структурувати послідовність дій та забезпечити точне їх виконання. Для цього доцільно застосовувати нотацію BPMN, яка дає можливість чітко візуалізувати кожен етап роботи системи, що в свою чергу сприяє кращому розумінню специфіки виконання операцій як для користувачів, так і для розробників.

Коли користувач здійснює вхід до системи обліку валютних операцій, він має можливість перейти на сторінку для створення нової транзакції. Після заповнення всіх передбачених формою обов’язкових полів, користувач підтверджує свої дії натисканням кнопки "Створити транзакцію". Система формує запит, який надсилається на сервер. Далі здійснюється перевірка наявного резерву відповідної валюти. У разі, якщо резерв є достатнім для виконання операції, система проводить транзакцію, оновлюючи залишки резерву відповідно до типу операції, застосованого курсу та зазначеної суми. Якщо ж доступного обсягу валюти недостатньо для обслуговування цієї операції, користувачу виводиться повідомлення про неможливість виконання транзакції. Графічне представлення цього бізнес-процесу відображено на рисунку 2.1.

Щоб створити бронювання, адміністратор авторизується у системі, після чого переходить до розділу створення бронювання. Тут необхідно заповнити всі передбачені для цього поля, вказавши контактні дані та обрану валюту. Після натискання кнопки "Створити бронювання" система надсилає запит на сервер, де перевіряється коректність введених даних та наявність вказаних валют у системі. У випадку позитивного результату бронювання фіксується в базі даних, і

користувачу відображається повідомлення про успішне створення заявки. Послідовність дій у межах даного бізнес-процесу представлено на рисунку 2.2.

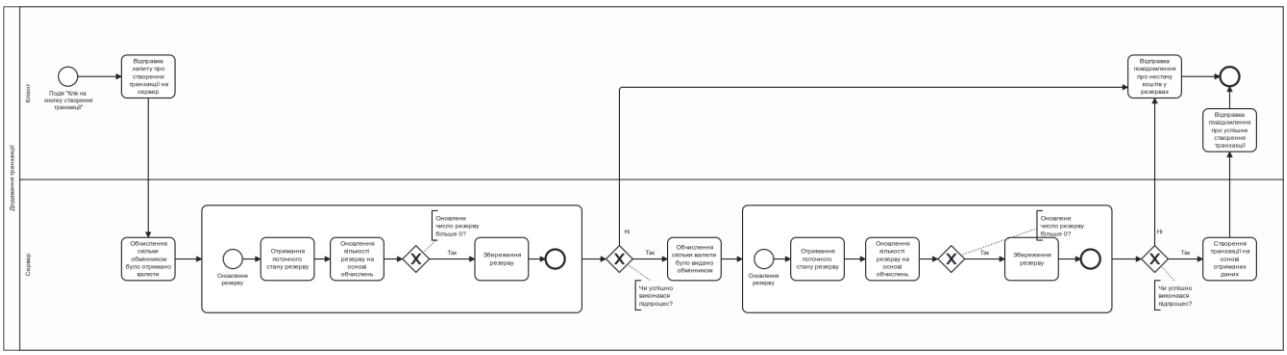

Рисунок 2.1 – Бізнес-процес створення транзакції

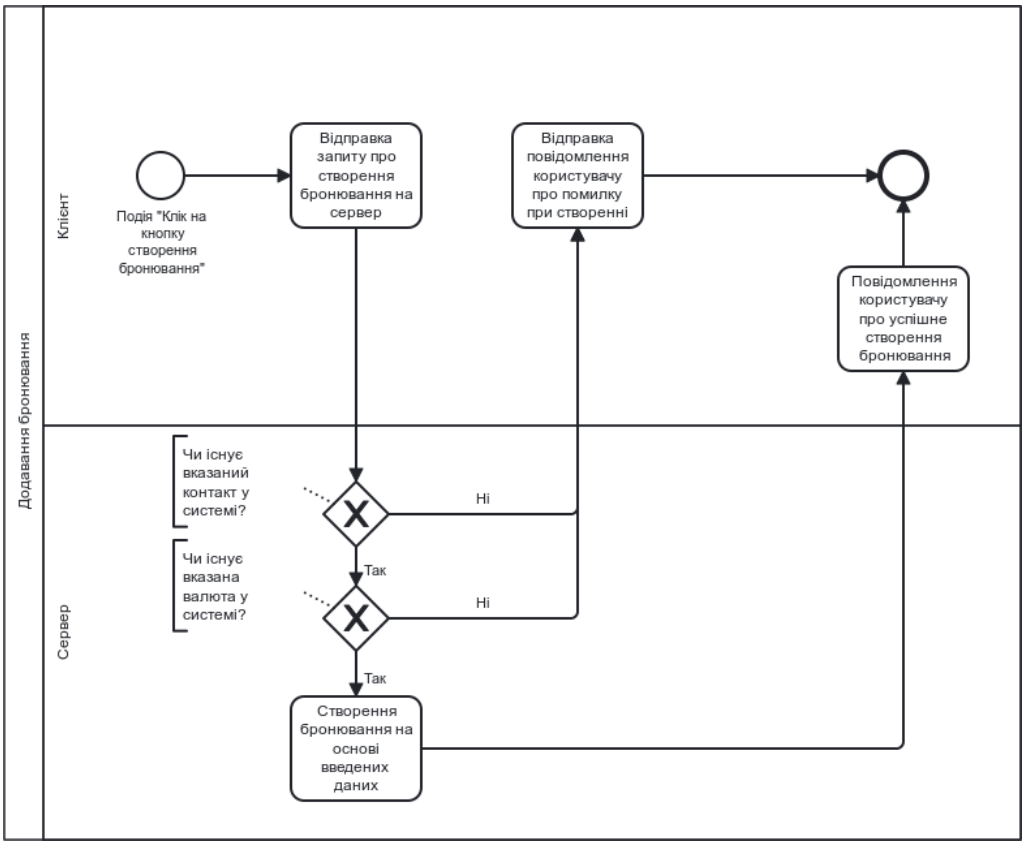

Рисунок 2.2– Бізнес-процес створення бронювання

Створення нової валюти в системі передбачає перехід користувача до відповідного розділу валют. Там він активує функцію додавання нової одиниці валюти шляхом натискання кнопки "Створити валюту". Далі користувач

заповнює всі необхідні поля і підтверджує введення. На сервер надсилається запит, який викликає звернення до API Національного банку України для отримання актуального офіційного курсу зазначеної валюти. Після цього система створює нову валюту, розраховує комерційний курс з урахуванням встановленого відсотка прибутковості для обмінного пункту та формує резерв цієї валюти. Весь бізнес-процес створення нової валюти представлений на рисунку 2.3.

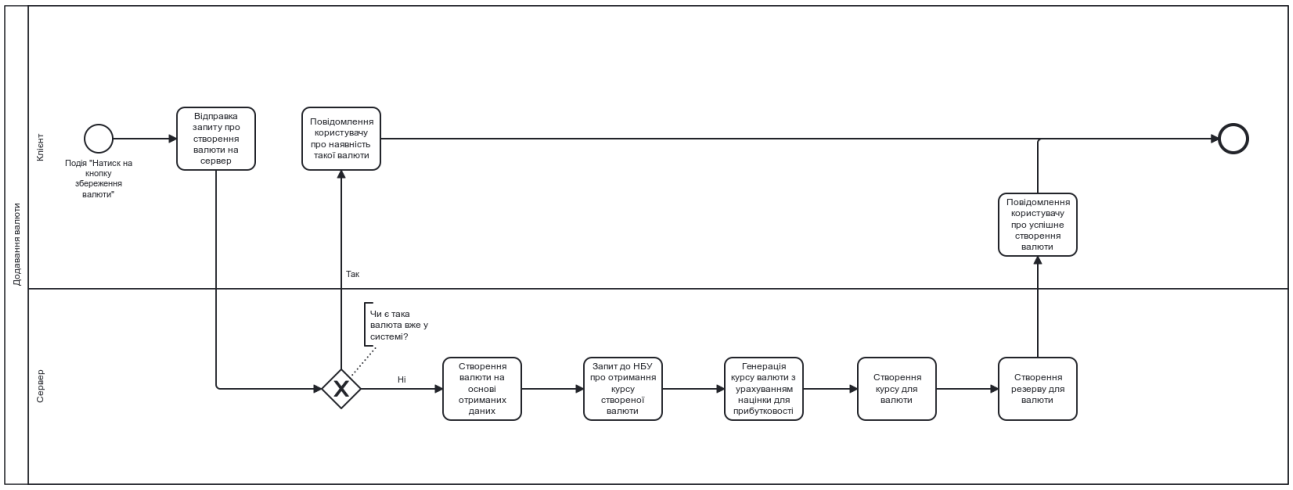

Рисунок 2.3 – Бізнес-процес створення валюти

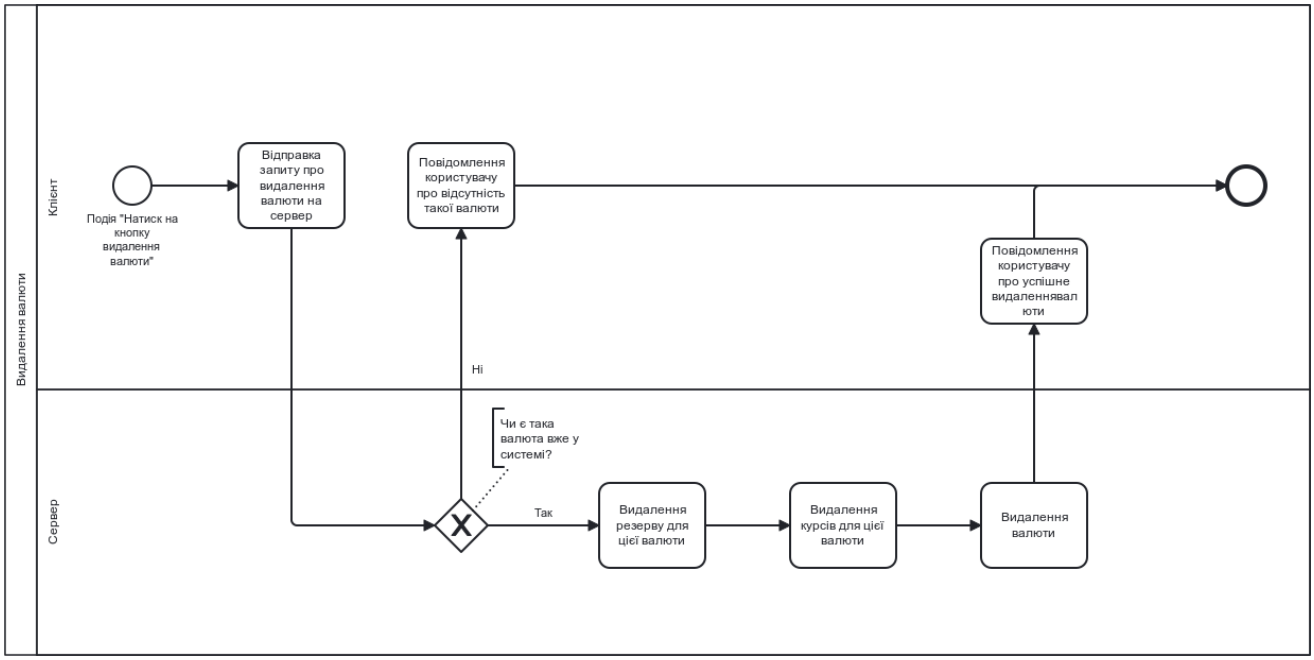

Рисунок 2.4 – Бізнес-процес видалення валюти з системи

Для видалення валюти користувач також повинен увійти до розділу валют і натиснути на кнопку "Видалити валюту". Після цього формується запит на сервер, який ініціює послідовне видалення всіх пов’язаних з обраною валютою записів — спочатку видаляється комерційний курс, потім резерви, а вже після цього сама валюта виключається з системи. Візуалізацію послідовності дій у рамках цього бізнес-процесу наведено на рисунку 2.4.

2.2 UML-діаграми варіантів використання

У розробленій системі для організації обліку роботи пункту обміну валют передбачено кілька категорій користувачів (акторів), кожна з яких виконує власні функції, взаємодіючи із системою відповідно до наданих прав доступу. Актори є безпосередніми учасниками бізнес-процесів і взаємодіють із програмним забезпеченням для виконання своїх завдань.

Клієнт — це користувач, який здійснює бронювання валюти, переглядає курси валют та створює контактні дані через Telegram-бот. Основні дії клієнта включають створення власного контакту, перегляд доступних валют і їх курсів, оформлення бронювань та отримання інформації про стан своїх замовлень. Взаємодія клієнта із системою побудована таким чином, щоб максимально спростити доступ до сервісів обмінного пункту та надати можливість віддаленого обслуговування.

Касир — співробітник пункту обміну, який безпосередньо здійснює валютно-обмінні операції, реєструє транзакції, переглядає та створює бронювання, працює з контактами клієнтів, а також має доступ до перегляду курсу валют. Касир виконує важливу роль у процесі обслуговування клієнтів, забезпечуючи швидке проведення фінансових операцій та їх облік у системі.

Менеджер — користувач із розширеними правами доступу, який здійснює управління довідниками валют, створює, видаляє та редагує валютні курси, проводить адміністрування резервів, формує бронювання, контролює дані

клієнтів і контакти, а також має можливість переглядати всі транзакції. Менеджер виконує організаційно-контрольну функцію у пункті обміну валюти, забезпечуючи підтримку актуальності даних і стабільність роботи системи.

Власник — актор із максимальними адміністративними повноваженнями в системі. До його компетенції належить повний контроль за курсами валют, резервами, транзакціями, бронюваннями та контактами. Власник має можливість створювати, редагувати та видаляти всі об’єкти в системі, управляти курсами, валютами та проводити аналіз фінансової діяльності пункту обміну за допомогою відповідної звітності. Цей актор відповідає за стратегічне управління всіма бізнес-процесами та підтримання актуальної інформації в системі.

Кожна діаграма містить набір варіантів використання, які відповідають конкретним бізнес-процесам, що реалізовані в системі. За їх допомогою відображено повну структуру взаємодії користувачів із програмним забезпеченням, а також уточнено доступні для кожного актору функції.

Для відображення функціональних можливостей програмного забезпечення пункту обміну валюти було розроблено чотири діаграми варіантів використання (Use Case Diagram) — окремо для кожного типу користувача. Ці діаграми дозволяють наочно продемонструвати взаємодію акторів із системою та перелік доступних їм функцій.

Рисунок 2.5 – Діаграма варіантів використання клієнта

Діаграма варіантів використання клієнта (рис. 2.5) – на цій діаграмі відображено взаємодію клієнта з Telegram-ботом — основним інтерфейсом для віддаленого користувача.

Основні варіанти використання:

– Створити контакт — клієнт має можливість зареєструвати свої персональні дані у системі (ім'я, телефон, пошта) через бот.

– Перегляд курсів валют — надає змогу отримати список доступних валют і їх актуальні курси.

– Створити бронювання — дозволяє клієнту зарезервувати валюту для майбутнього обміну за фіксованим курсом.

– Переглянути бронювання — надає клієнту інформацію про всі створені ним бронювання, їх статуси та деталі.

Ця діаграма демонструє обмежений, але зручний набір функцій, орієнтований на дистанційне обслуговування клієнтів без відвідування пункту обміну.

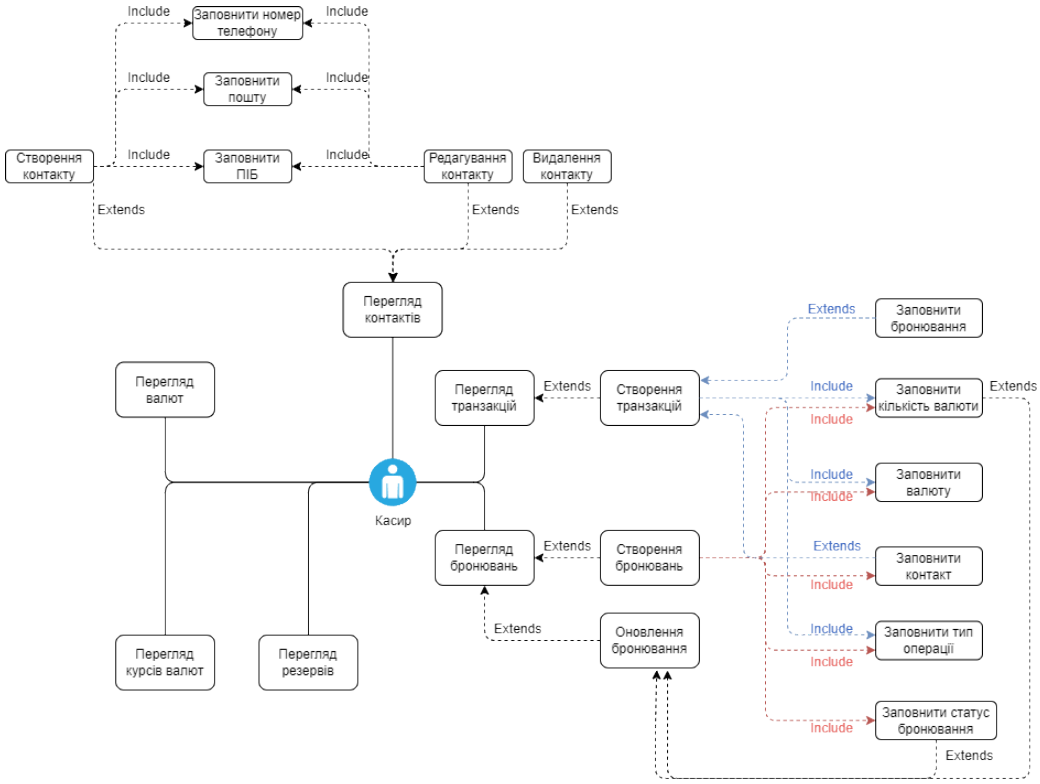

Рисунок 2.6 – Діаграма варіантів використання касира

Діаграма варіантів використання касира (рис. 2.6):

Касир — це співробітник обмінного пункту, що виконує операції безпосередньо в адміністративній частині системи.

Основні варіанти використання:

− Переглянути валюти — дозволяє ознайомитися зі списком доступних валют у системі.

− Переглянути активні курси валют — відображає наявні курси валютних пар для виконання операцій.

− Створити транзакцію — реєстрація валютно-обмінної операції, проведеної клієнту.

− Переглянути транзакції — доступ до списку всіх проведених транзакцій.

− Створити бронювання — фіксація бронювання валюти за запитом клієнта безпосередньо в касі.

− Оновити та видалити бронювання — зміна деталей або скасування бронювання.

− Перегляд контактів клієнтів — отримання списку контактних даних користувачів.

Діаграма показує, що касир володіє правами на операції з транзакціями, бронюваннями та переглядом інформації, але не має можливості керувати валютами та курсами.

Діаграма варіантів використання менеджера (рис. 2.7) – менеджер має розширені повноваження, що дозволяють адмініструвати налаштування пункту обміну.

Основні варіанти використання:

− Усі функції касира.

− Створення, перегляд, оновлення та видалення валют — управління довідником валют.

− Створення, оновлення та перегляд валютних курсів — додавання та коригування курсів для валютних пар.

– Управління контактами клієнтів — створення, оновлення та перегляд контактної інформації користувачів системи.

Менеджер контролює всі бізнес-процеси, пов'язані з валютами, курсами, бронюваннями та контактами.

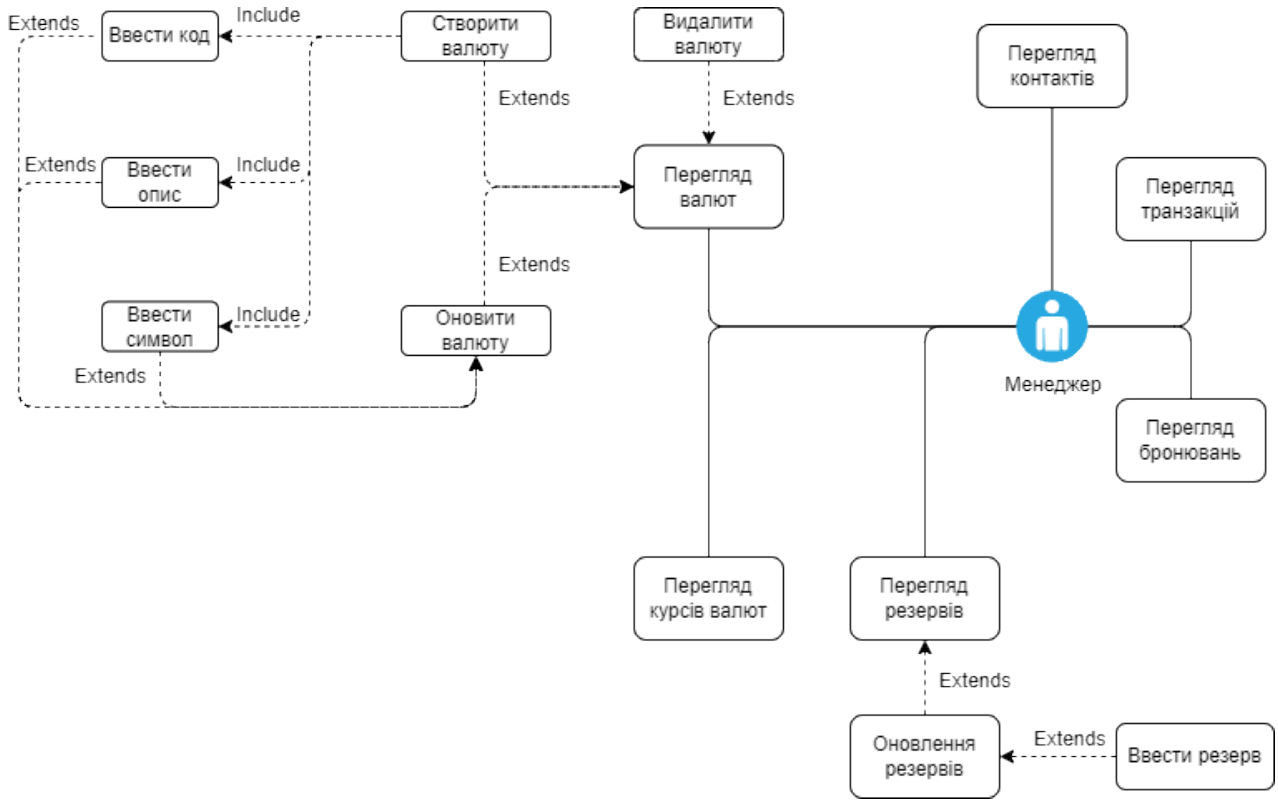

Рисунок 2.7 – Діаграма варіантів використання менеджера

Діаграма варіантів використання власника (рис. 2.8) – власник є користувачем із максимальними правами у системі. Він виконує функції менеджера, а також стратегічне управління та контроль бізнес-процесів.

Основні варіанти використання:

– Усі функції менеджера.

– Повний контроль за всіма транзакціями, валютами, курсами та бронюваннями.

– Можливість оновлювати будь-які налаштування та параметри системи.

На цій діаграмі власник виступає універсальним адміністратором, що має доступ до всіх даних і функцій системи.

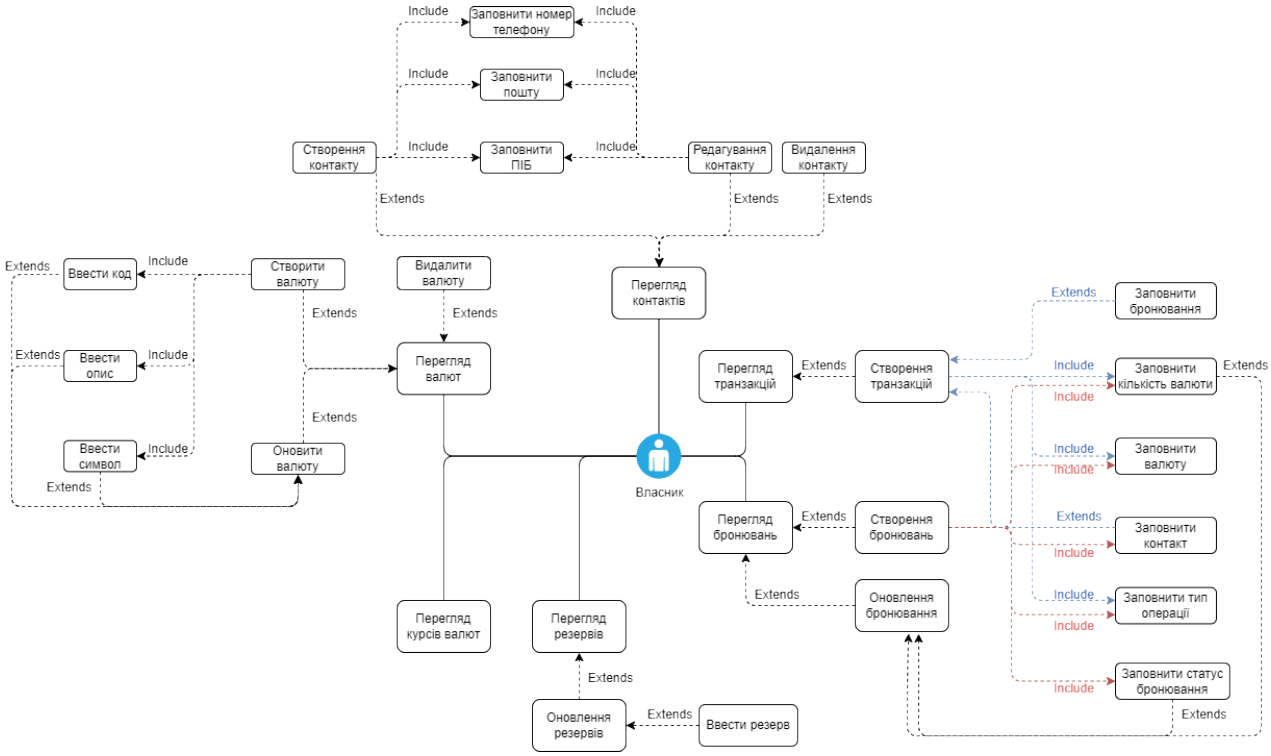

Рисунок 2.8 – Діаграма варіантів використання власника

2.3 UML-діаграми послідовності

Процес починається з того, що клієнт, користуючись Telegram-ботом, обирає опцію створення бронювання. Бот звертається до API сервера із запитом про доступні валюти. Сервер, у свою чергу, надсилає запит до бази даних і отримує перелік доступних валют. Отриманий список передається назад до бота, який відображає його клієнту. Після цього клієнт обирає потрібну валюту, а бот надсилає серверу запит на отримання актуального курсу. Сервер звертається до бази даних, отримує курс і передає цю інформацію боту, який показує її клієнту. Після ознайомлення клієнт підтверджує створення бронювання. Бот надсилає цю інформацію на сервер, де вона записується до бази даних. Після успішного створення бронювання сервер надсилає боту підтвердження, а той — клієнту повідомлення про завершення процедури.

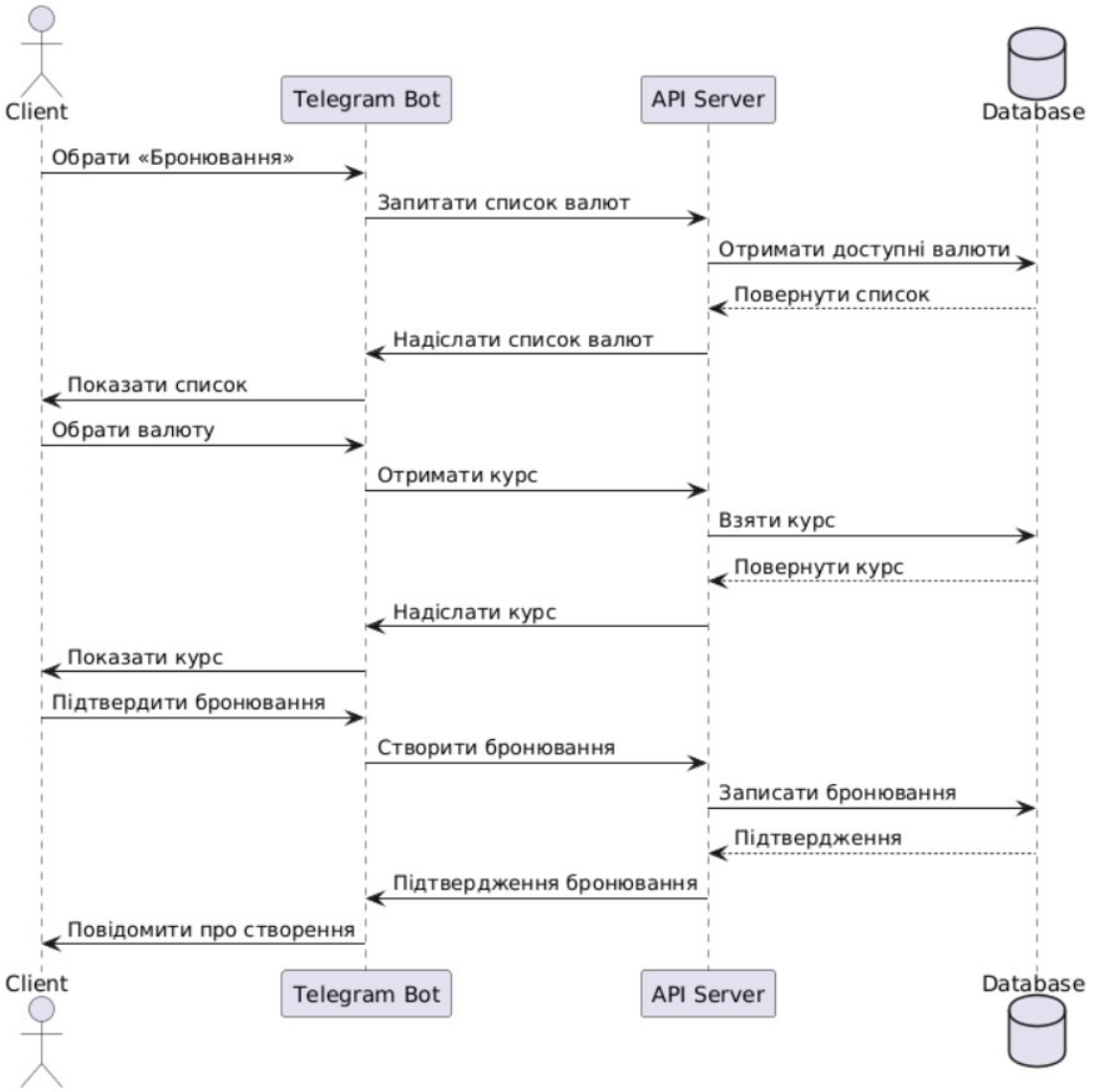

Рисунок 2.9 – Діаграма послідовності: створення бронювання клієнтом

Після авторизації в системі касир у вебінтерфейсі обирає функцію створення нової транзакції. Інтерфейс надсилає дані транзакції на сервер. Сервер перевіряє актуальний резерв валюти у базі даних, відправляючи відповідний запит. Якщо резервів достатньо, сервер реєструє транзакцію в базі, оновлює залишки валют і надсилає підтвердження про успішне проведення операції до інтерфейсу. У разі нестачі резерву сервер формує повідомлення про помилку та передає його вебінтерфейсу. Інтерфейс інформує касира про результат виконання операції — успішне завершення чи відмову через відсутність необхідного обсягу валюти.

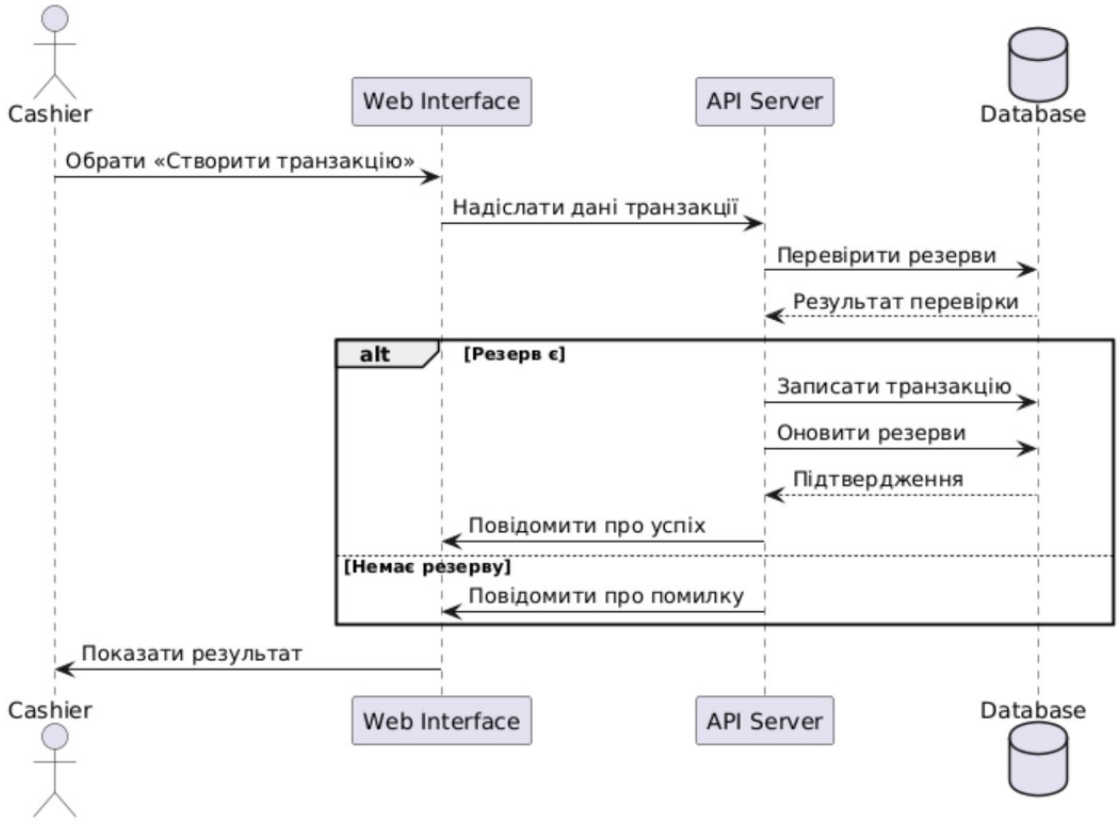

Рисунок 2.10 – Діаграма послідовності: створення транзакції касиром

Менеджер авторизується у вебінтерфейсі та обирає функцію додавання нової валюти. Інтерфейс передає введені параметри валюти на сервер. Сервер надсилає запит до API Національного банку України для отримання офіційного курсу цієї валюти. Після отримання відповіді сервер розраховує комерційний курс, враховуючи встановлений відсоток прибутковості обмінного пункту. Далі сервер формує запис у базі даних, додає нову валюту разом із курсом і резервом, а після успішного завершення операції надсилає до інтерфейсу підтвердження. Інтерфейс повідомляє менеджера про успішне створення валюти в системі.

Власник входить у вебінтерфейс і обирає валюту, курс якої потрібно оновити. Інтерфейс надсилає запит серверу для отримання поточного курсу валюти. Сервер звертається до бази даних, отримує необхідні дані та повертає їх інтерфейсу. Інтерфейс відображає поточний курс власнику. Після внесення нового значення курсу власник підтверджує оновлення. Інтерфейс передає змінені дані на сервер. Сервер оновлює запис у базі даних і надсилає повідомлення про

успішне завершення операції до інтерфейсу. Інтерфейс показує власнику повідомлення про успішне оновлення курсу.

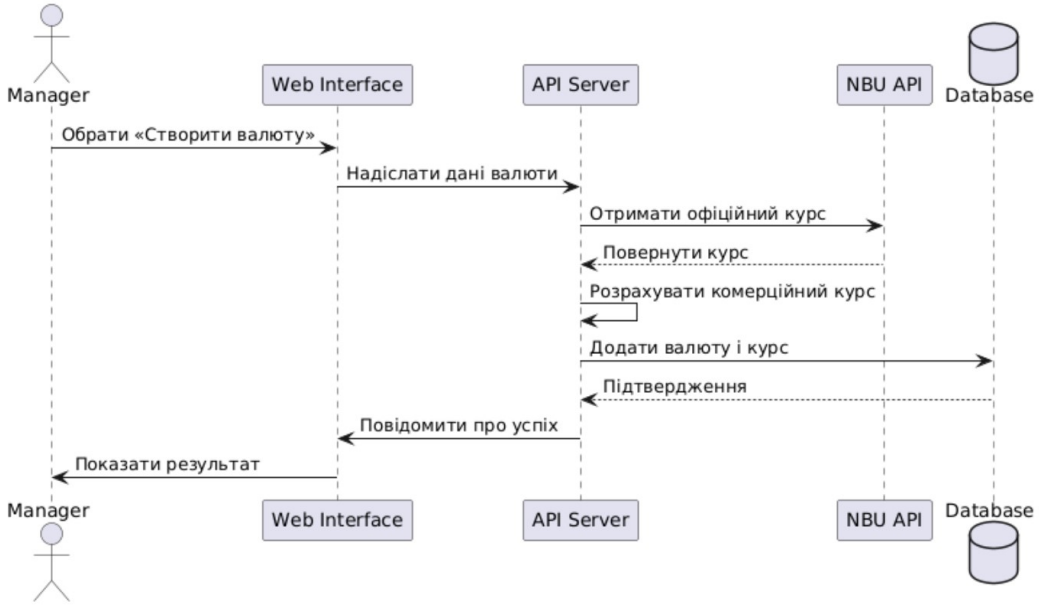

Рисунок 2.11 – Діаграма послідовності: створення валюти менеджером

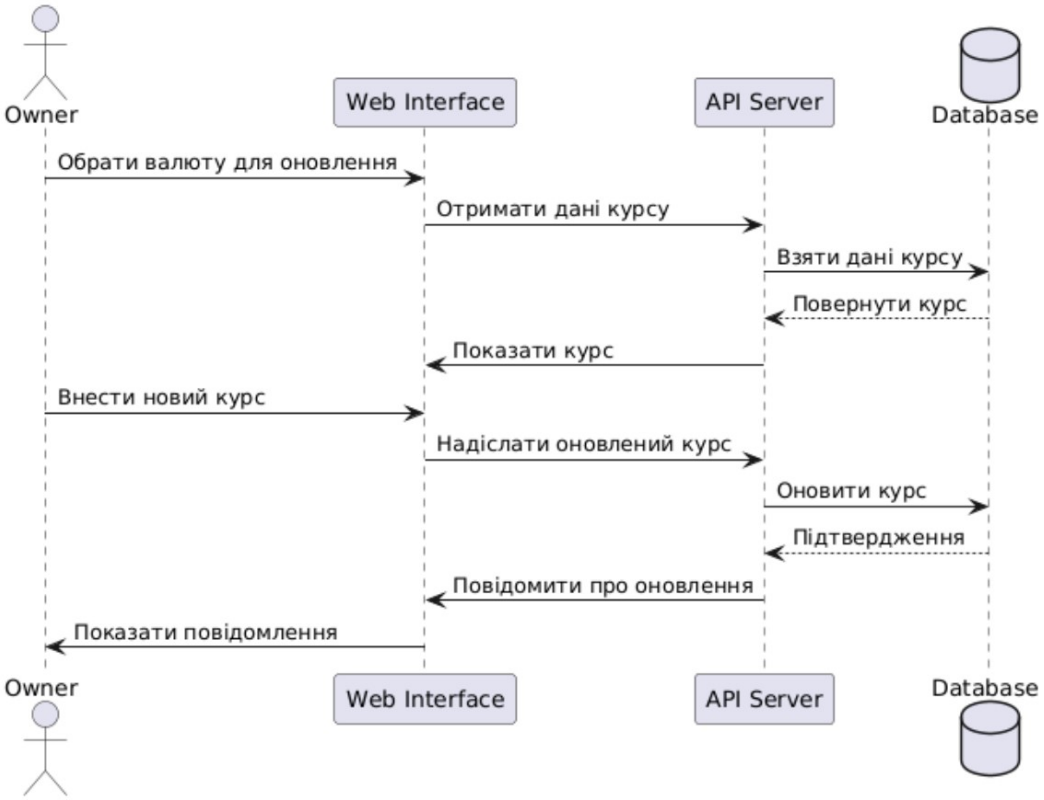

Рисунок 2.12 – Діаграма послідовності: оновлення курсу валюти власником

2.3 Розробка бази даних

Під час проектування бази даних було спроектовано структуру (рис. 2.13), що складається з кількох основних таблиць, які разом формують систему для обліку валютних операцій, бронювань, транзакцій та користувачів. Нижче подано детальний опис таблиць і логічну структуру бази даних.

Таблиця Rates зберігає курси валют. Кожен запис містить унікальний ідентифікатор, дату створення і модифікації, посилання на валюту, курси купівлі та продажу, а також ознаку активності запису.

Таблиця Reservations фіксує бронювання валютних операцій. Вона пов’язана з контактами, курсами, операціями, типами операцій і статусами бронювань. Містить поля для суми, дати створення/зміни та ознаки активності.

Таблиця Transactions відповідає за фактичні валютні транзакції. Вона має зв’язки з контактами, логами курсів, бронюваннями, типами і видами операцій. Також фіксує суму валюти, дату і статус.

Таблиця Users містить інформацію про користувачів системи, кожен з яких має контакт та певну роль. Зберігає дати створення/зміни та активність.

Таблиця UserRoles описує ролі користувачів системи, наприклад, адміністратор чи касир. Складається з ідентифікатора, дати створення та назви ролі.

Таблиця Contacts веде облік контактної інформації осіб, включаючи ПІБ, email, телефон, чорний список, дати створення/зміни та статус.

Таблиця Funds відображає резерв валюти у системі. Містить прив'язку до валюти, кількість, дати створення/зміни та активність.

Таблиця Currencies описує доступні валюти: код, опис, символ, прив’язку до ідентифікатора Нацбанку, а також дату створення/зміни та активність.

Таблиця IssuanceLogs зберігає історію курсів валют, що були використані в операціях. Запис містить курс купівлі/продажу, дату та валюту.

Таблиця ExchangeSettings встановлює правила обміну: відсоток доходу від транзакції, порогові значення для мінімального резерву та максимальної транзакції, а також валюту, до якої це відноситься.

Таблиця OperationTypes містить типи операцій (наприклад, обмін, покупка), включаючи назву та дату створення.

Таблиця ReservationStatuses містить перелік можливих статусів бронювання, таких як "очікує", "підтверджено", "скасовано".

Таблиця TransactionTypes визначає типи транзакцій, подібно до операцій, наприклад, "готівка", "перерахування".

Загалом структура бази даних добре нормалізована, містить чіткі зв'язки між таблицями через UUID-поля, передбачає контроль версій даних через поля CreatedOn/ModifiedOn та має продуману логіку щодо активності записів і ролей користувачів.

У базі даних зв’язки між таблицями реалізуються через поля з унікальними ідентифікаторами, що встановлюють логічну залежність між сутностями. Таблиця бронювань є центральною точкою для збереження даних про валютні операції. Вона пов’язана з контактами, курсами, типами операцій, типами транзакцій та статусами бронювань через зовнішні ключі. Таким чином, кожне бронювання має прив’язку до особи, курсу, виду операції, способу оплати і поточного статусу.

Таблиця транзакцій також прив’язується до контактів і може бути пов’язана з бронюванням, з якого вона виникла. Для вказівки курсу, який був використаний, транзакція посилається на лог запису курсу. Вона також включає тип операції та тип транзакції, подібно до бронювання, але фіксує вже фактичний рух коштів.

Користувачі системи пов’язані з контактами, які зберігають загальну особисту інформацію, і з ролями, що визначають рівень доступу в системі. Таким чином, користувач є конкретною контактною особою з певними повноваженнями.

Таблиця курсів валют пов’язується з таблицею валют через ідентифікатор валюти, яка є базовою для розрахунків. Аналогічно, таблиця резерву валюти містить зв’язок з конкретною валютою, для якої обліковується доступна сума.

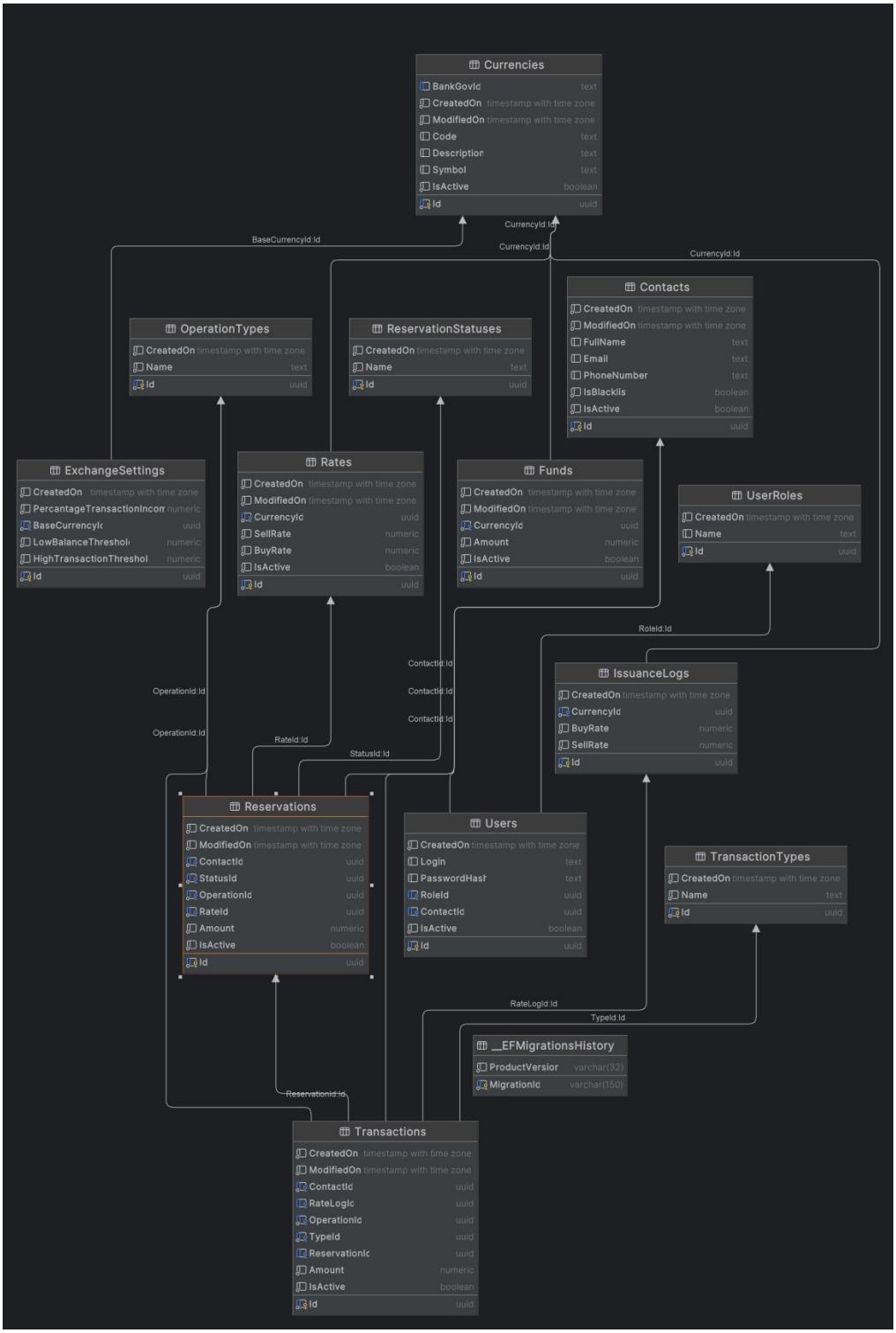

Рисунок 2.13 – Діаграма бази даних

Журнали видачі курсів також прив’язані до валют і зберігають історію зміни курсів.

Налаштування обміну містять посилання на валюту, до якої застосовуються ці параметри, і визначають правила транзакцій, такі як допустимі ліміти і відсоток доходу.

Зв’язки у базі забезпечують цілісність даних і дозволяють побудувати узгоджену модель фінансових операцій, де кожна транзакція, бронювання або користувач мають чітке місце у загальній структурі.

2.4 Проектування прототипів екранів і форм системи

В бакалаврській роботі передбачено створення низки прототипів для ключових екранів і форм системи, що автоматизує діяльність пункту обміну валют. Дизайн кожного з цих прототипів відповідає загальним вимогам до зручності користування, інтуїтивності навігації, чіткості подання інформації та адаптивності для різних пристроїв.

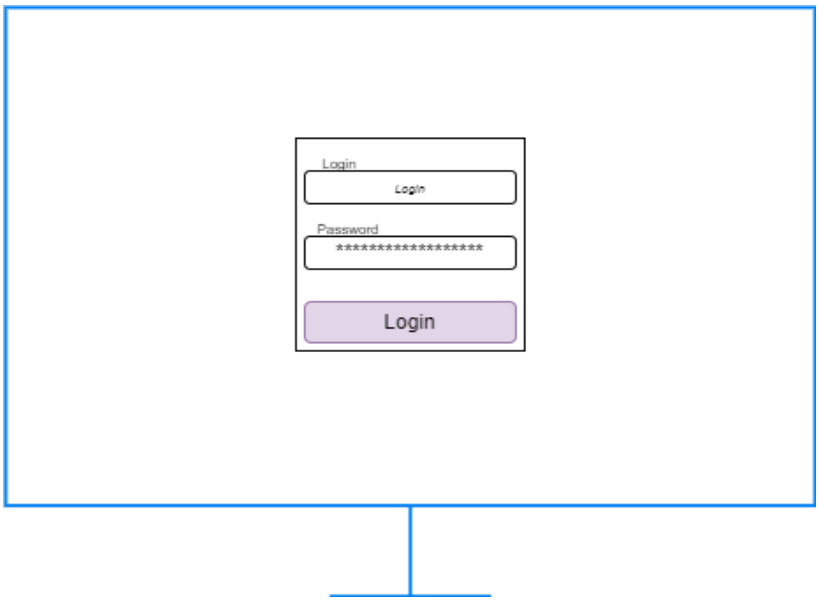

Рисунок 2.14 – Прототип форми авторизації

Першим елементом є форма авторизації (рис. 2.14). Дизайн цього вікна передбачає розміщення полів для введення логіна та пароля з помітним заголовком і кнопкою входу. Розташування елементів упорядковане для швидкого

та безпомилкового введення даних. Після успішної авторизації користувач потрапляє на головне меню. Це вікно містить зрозумілу навігаційну панель з доступом до основних функцій: перегляду валют, бронювань, транзакцій, курсів та резервів.

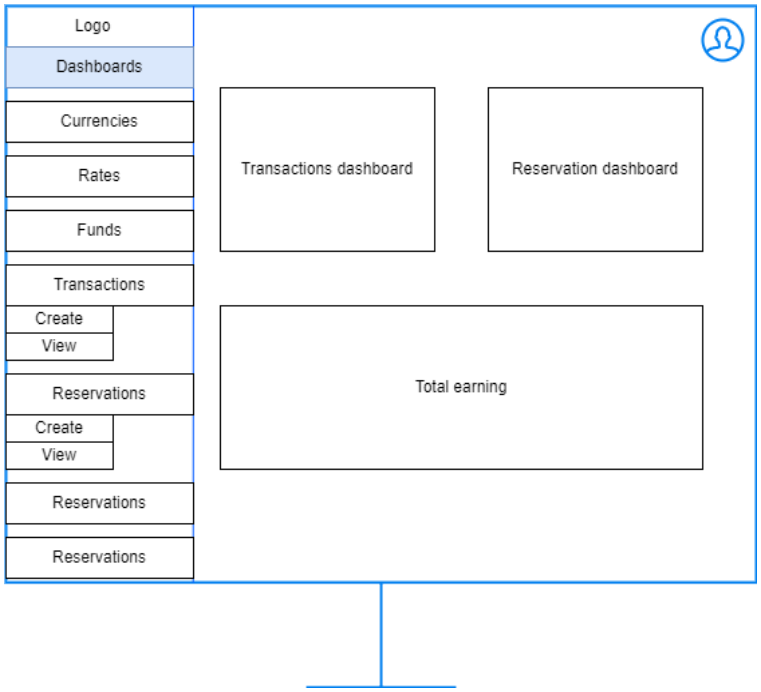

Рисунок 2.15 – Прототип головного меню

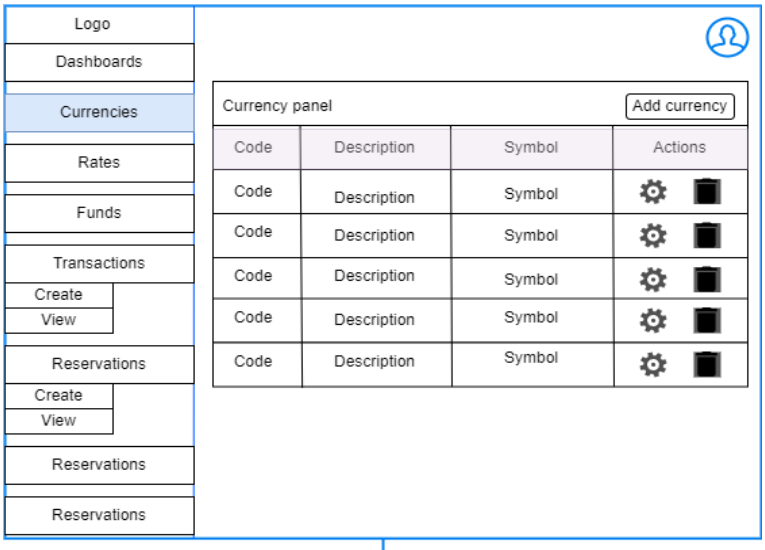

Рисунок 2.16 – Прототип списку валют

У системі реалізовано форму перегляду списку валют (рис. 2.16). На цій сторінці всі наявні валюти відображаються у вигляді таблиці із зазначенням назви, умовного коду та ідентифікатора. Додатково передбачено окремі кнопки для оновлення та створення нових валют. Для створення нової валюти форму було спроєктовано так, щоб користувач вводив назву валюти, вибирав її код і міг відразу задати початкові параметри курсу. Оновлення валют реалізується за аналогічним принципом, із можливістю редагування вже збережених даних.

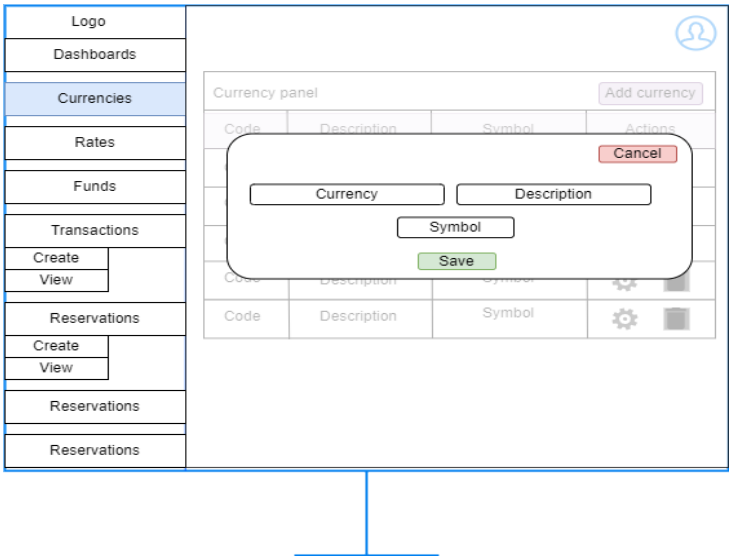

Рисунок 2.17 – Прототип створення валют

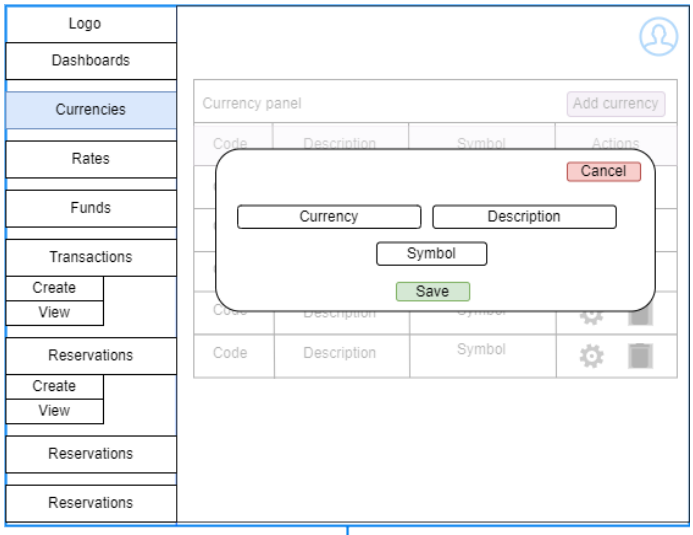

Рисунок 2.18 – Прототип зміни або оновлення валют

Перегляд і керування курсами реалізовано у вигляді списку, де для кожної валюти вказано курс купівлі та продажу, дату встановлення курсу і адміністратора, який його додав. Для створення нового курсу користувач заповнює форму, в якій обирає валюту, вводить курс купівлі та продажу, після чого дані зберігаються в системі.

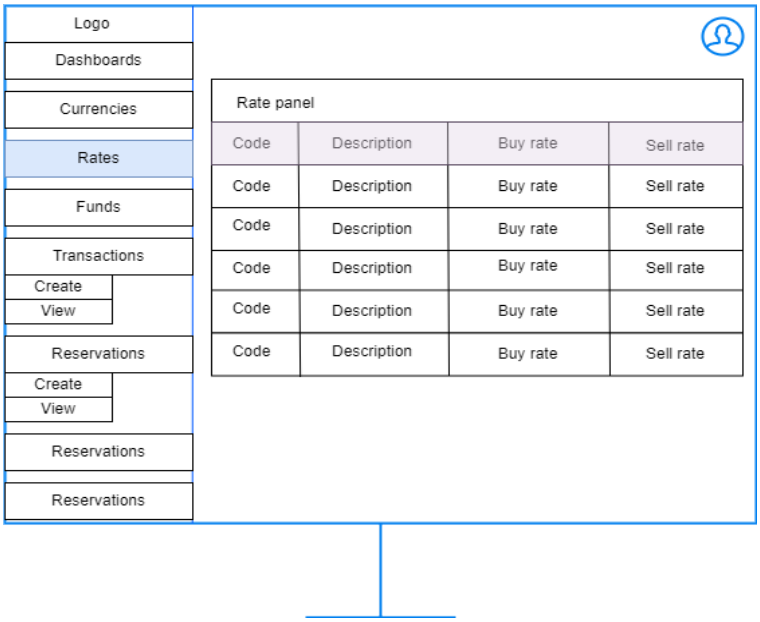

Рисунок 2.19 – Прототип списку курсів валют

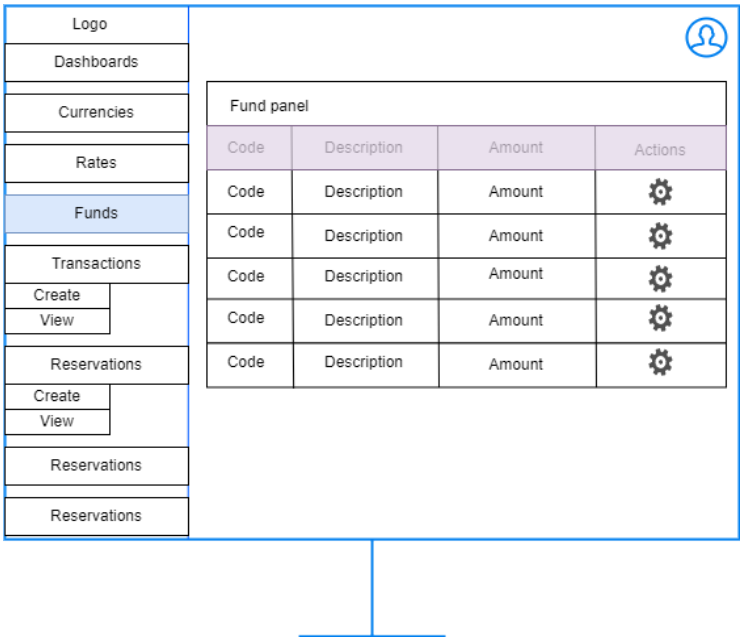

Рисунок 2.20 – Прототип списку резервів

У системі також передбачено перегляд списку резервів, де відображено назви валют і кількість готівки у касі. Окремий функціонал дозволяє редагувати ці дані — форма редагування резерву містить поле для введення нової кількості валюти, що дозволяє підтримувати актуальну інформацію про наявні залишки.

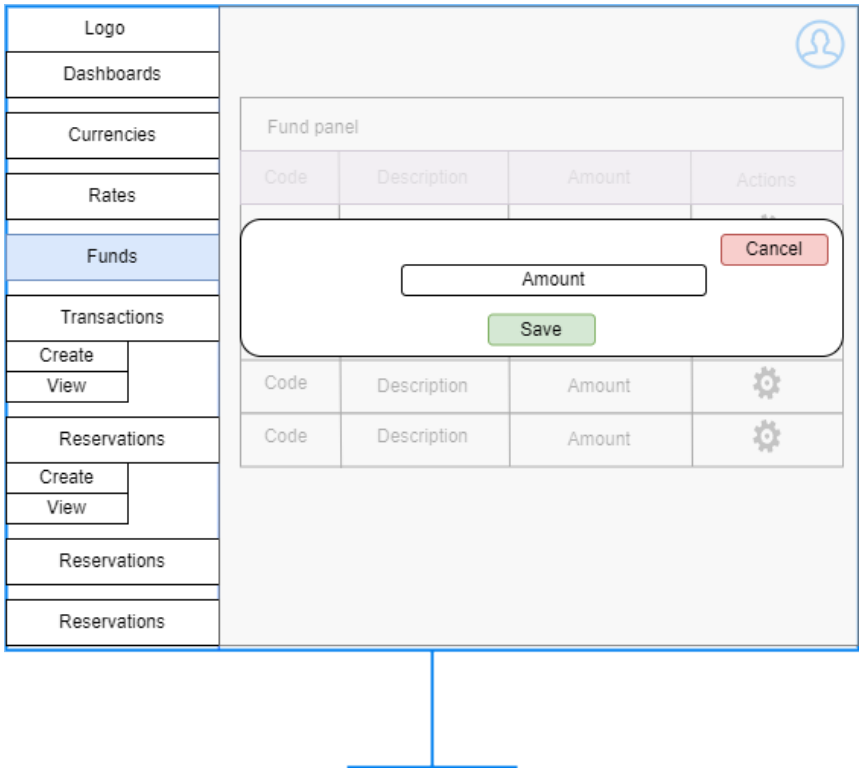

Рисунок 2.21 – Прототип зміни або оновлення резерву

Бронювання відображаються у вигляді списку з зазначенням контактних даних клієнта, назви валюти, суми, статусу і дати. Процес створення бронювання передбачає вибір валюти, введення кількості, зазначення контактних даних клієнта та збереження інформації в системі. Для редагування бронювання передбачено форму зі зміною основних параметрів заявки.

Перегляд транзакцій реалізовано у вигляді журналу операцій, де зазначено тип транзакції, валюту, суму, курс, касира та дату. Створення нової транзакції виконується через відповідну форму, яка передбачає вибір валюти, типу операції (купівля чи продаж), введення суми та фіксацію курсу.

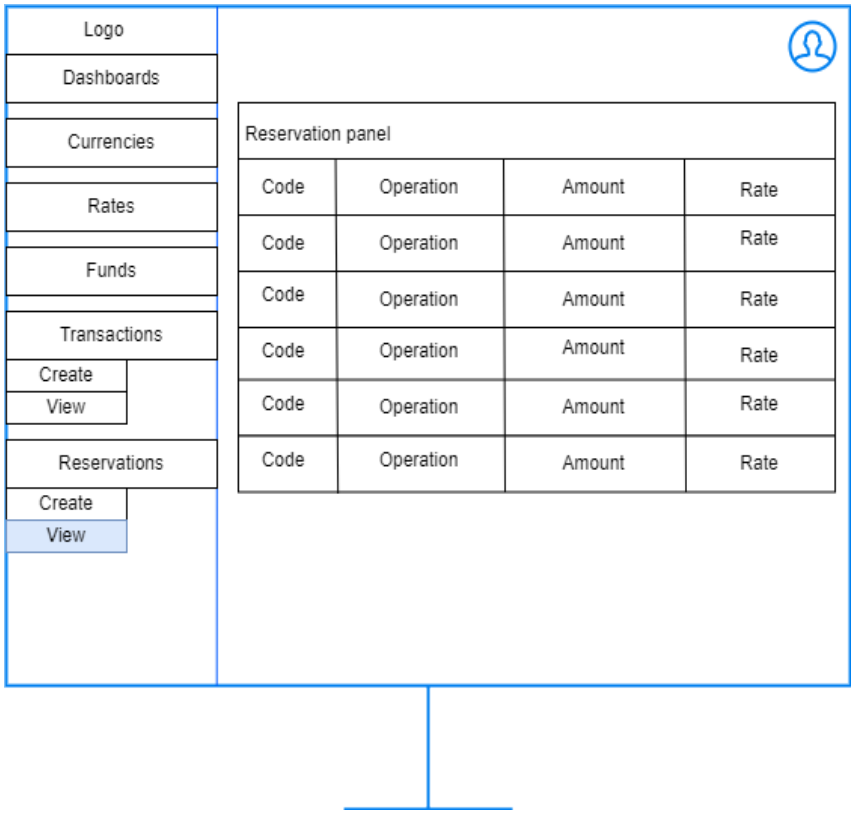

Рисунок 2.22 – Прототип бронювань валюти

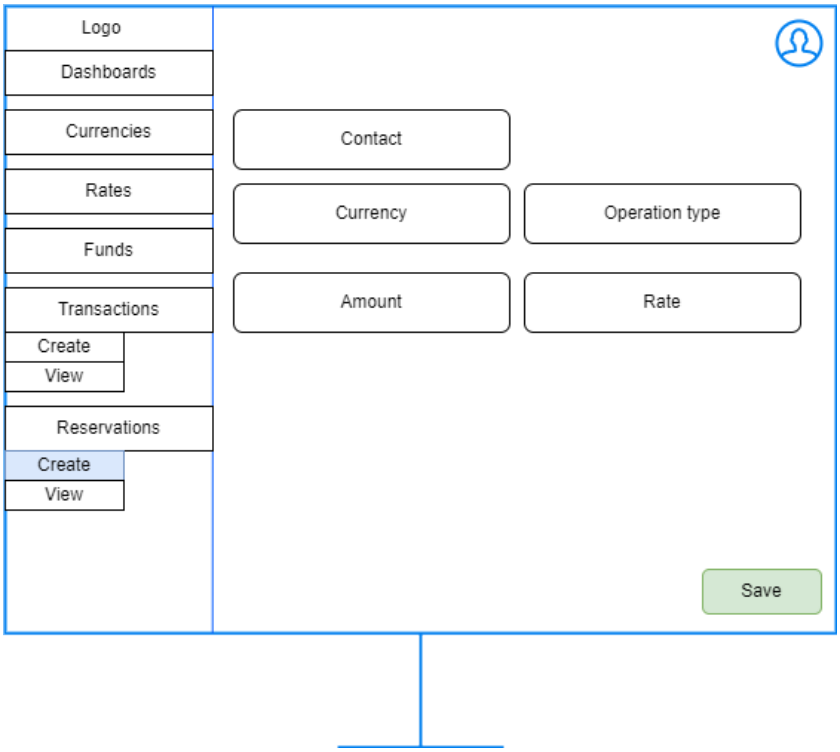

Рисунок 2.23 – Прототип створення бронювання

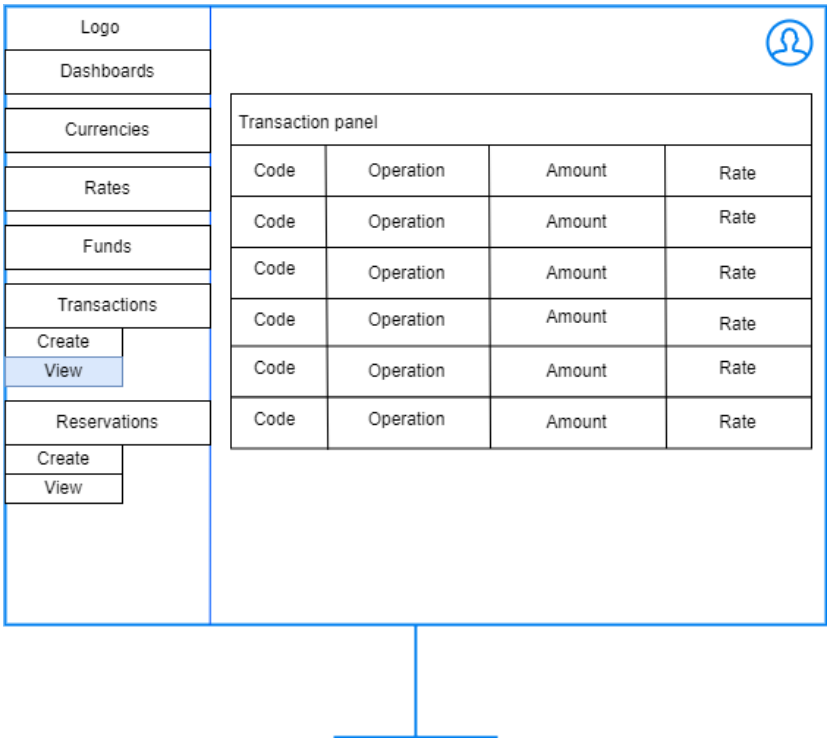

Рисунок 2.24 – Прототип перегляду транзакцій

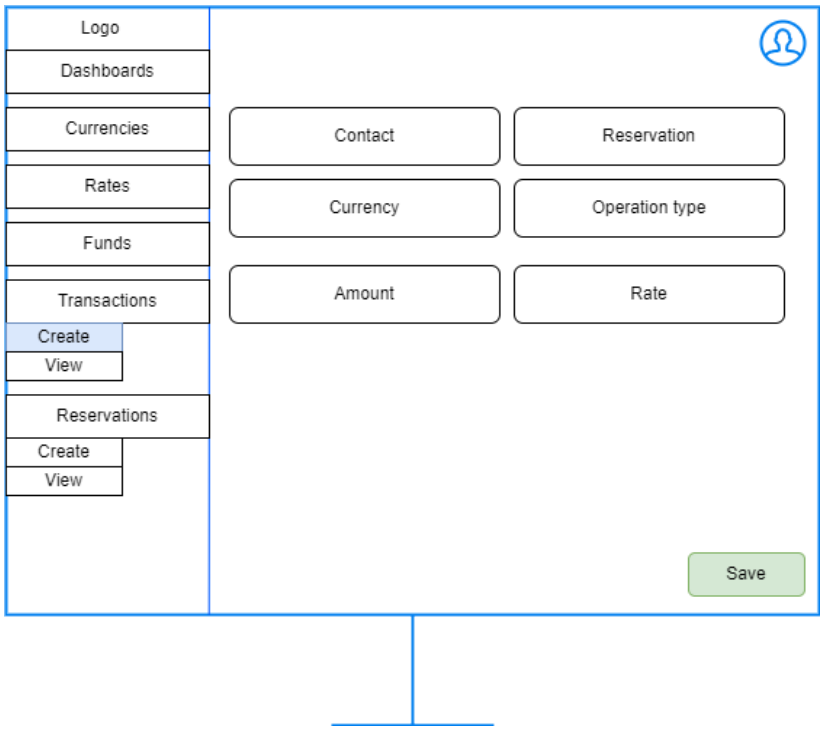

Рисунок 2.25 – Прототип створення транзакцій

Окремий блок прототипів стосується Telegram-бота, що виступає в ролі клієнтського інтерфейсу. Головне меню бота містить варіанти перегляду курсів

валют, створення бронювання, перегляду особистих бронювань і даних контакту. Прототип форми створення контакту в боті реалізований у вигляді послідовного заповнення полів для імені, телефону та електронної пошти. Перегляд курсу валют у боті виконується через виведення списку доступних валют із їхнім актуальним курсом. Аналогічно реалізовано перегляд бронювань, де виводиться інформація про дату, валюту, суму та статус заявки.

Рисунок 2.26 – Прототип меню

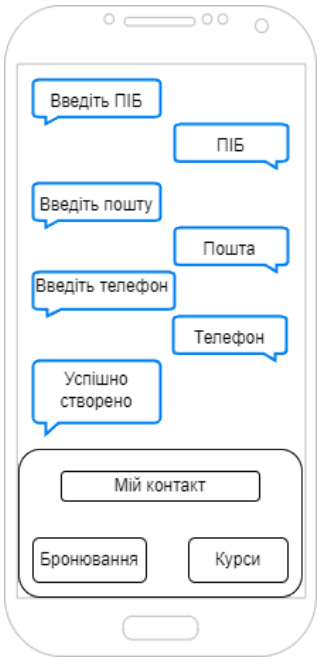

Рисунок 2.27 – Прототип створення контакту

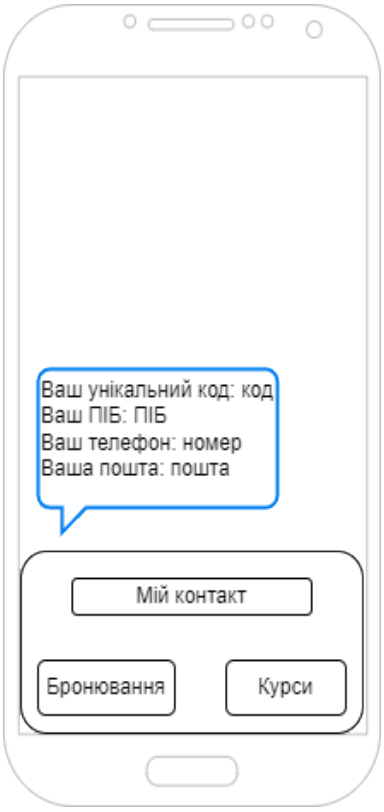

Рисунок 2.28 – Прототип форми з інформацією про контакт

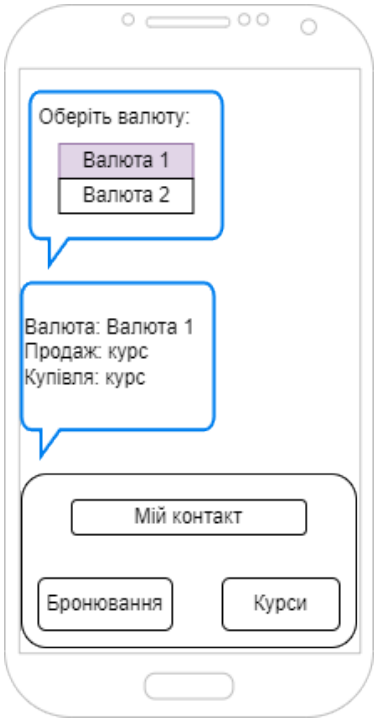

Рисунок 2.29 – Прототип форми роботи з валютою

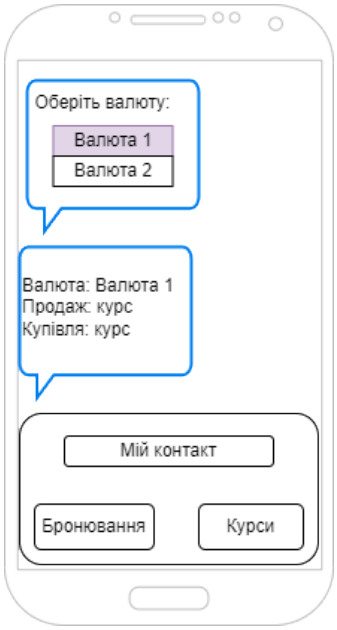

Рисунок 2.30 – Прототип форми створення бронювання

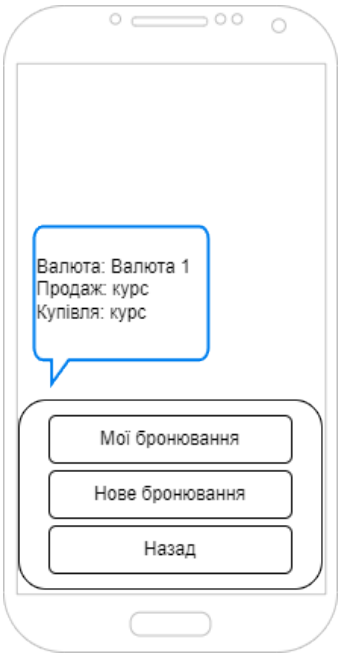

Рисунок 2.31 – Прототип форми перегляду бронювань

Особливістю проєктування інтерфейсів є орієнтація на максимально спрощену та зрозумілу взаємодію користувачів різного рівня підготовки, тому всі форми мають логічну послідовність заповнення, а кнопки виконання дій мають зрозумілі назви та розміщені так, щоб їх було легко знайти.

Загальна концепція дизайну витримана в єдиному стилі з сучасним візуальним оформленням, яке відповідає вимогам до професійного програмного забезпечення для фінансових установ. Інтерфейси адаптовані для роботи як на десктопах, так і на мобільних пристроях, що дозволяє забезпечити доступ до системи з різних пристроїв і в різних умовах. Усі форми мають обов'язкову валідацію введених даних та сповіщення про помилки або успішне виконання дій, що підвищує надійність роботи системи.

Також дизайн передбачає використання контрастних кольорів для основних кнопок дій, що дозволяє користувачам миттєво ідентифікувати функціональні елементи та забезпечує швидку взаємодію з системою в умовах високого навантаження чи обмеженого часу на виконання операцій.

2.5 Архітектура проекту

Для реалізації програми для організації обліку роботи пункту обміну валюти було обрано клієнт-серверну архітектуру. Такий підхід передбачає розподіл функціональності між клієнтською частиною, яка надсилає запити, та серверною, що обробляє ці запити, управляє ресурсами й повертає клієнту відповідь. У межах цієї концепції взаємодія між компонентами відбувається через мережу, завдяки чому дана модель також відома як мережеві обчислення або архітектура клієнт-сервер.

Архітектура програмного продукту побудована на основі патерну MVC (Model-View-Controller). Ця структура дозволяє розділити програмну логіку на три окремі складові: модель, яка відповідає за роботу з даними та бізнес-логікою, представлення, що відображає інформацію для користувача, та контролер, який обробляє запити та координує взаємодію між моделлю і представленням. Застосування цього патерну забезпечує зрозумілу структуру коду, що значно спрощує його модифікацію, підтримку та подальший розвиток. Додатковою

перевагою є можливість тестування окремих компонентів без впливу на роботу всієї системи.

Під час розробки вебзастосунку було впроваджено стиль REST (Representational State Transfer). Його основний принцип полягає в тому, що всі ресурси в системі мають власні унікальні ідентифікатори у вигляді URL-адрес, а робота з ними здійснюється за допомогою стандартних HTTP-методів: «GET» використовується для отримання даних, «POST» — для додавання нових записів, «PUT» — для внесення змін до вже існуючих, а «DELETE» — для видалення. Кожен ресурс системи доступний через свою адресу, що дозволяє зручно здійснювати операції з інформацією та контролювати її стан.

Передача даних у такій системі реалізується за допомогою REST-запитів, в яких параметри передаються у складі HTTP-запитів. Для вебслужб, які працюють відповідно до принципів REST, використовується визначення «RESTful», що свідчить про відповідність встановленим стандартам і методам цього стилю.

Завдяки впровадженню REST-архітектури система отримала гнучку та оптимізовану структуру, яка спрощує обслуговування, розширення функціоналу та забезпечує стабільність роботи вебзастосунку навіть за зростання навантаження. Такий підхід дозволив організувати обмін даними між клієнтською та серверною частинами у максимально ефективний спосіб, що особливо важливо для системи, яка працює з фінансовими операціями в режимі реального часу.

В цьому розділі була здійснена розробка структури програмного забезпечення для організації обліку роботи пункту обміну валюти, яка дала змогу створити логічно цілісну архітектуру, що охоплює всі основні бізнес-процеси й механізми взаємодії користувачів із системою. Змодельовані діаграми варіантів використання та послідовності дозволили детально описати порядок виконання ключових операцій, забезпечивши прозорість та узгодженість процесів.

База даних спроєктована з урахуванням нормалізації, контролю зв’язків і актуальності даних, що гарантує її стабільну роботу та надійне збереження інформації.

Особливу увагу приділено проектуванню прототипів інтерфейсів, які відповідають сучасним стандартам зручності та доступності, забезпечуючи швидкий доступ до функцій для всіх категорій користувачів. Впроваджена клієнт-серверна архітектура на основі MVC-патерну та REST-стилю дозволяє ефективно організувати обробку запитів, підтримку резервів, роботу з транзакціями й бронюваннями в реальному часі.

В результаті створено універсальну, надійну та масштабовану систему, адаптовану до потреб пунктів обміну валют.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

3.1 Вибір технологій

Розроблене програмне забезпечення для ведення обліку діяльності пункту обміну валют забезпечує зручний функціонал для оперативного контролю всіх валютно-обмінних операцій, що здійснюються в обмінному пункті щодня. Система дозволяє виконувати транзакції та зберігати їх в історії операцій, створювати бронювання, контролювати та оновлювати їхній статус, а також забезпечує віддалений доступ до інформації про кількість доступної валюти в резервах і актуальні курси валют. Курси надходять із офіційного джерела — Національного банку України — і автоматично коригуються відповідно до встановленого комерційного відсотка, що забезпечує фінансову дохідність обмінного пункту. Крім того, система дозволяє аналізувати доходи за певні періоди, відстежувати обсяги транзакцій та кількість бронювань, що оброблені адміністративним персоналом. Інтерфейс програмного забезпечення є зручним та функціонально насиченим, створеним для ефективного управління фінансовими і бізнес-даними.

З метою підвищення ефективності роботи обмінного пункту у розробці використано сучасні технологічні рішення, що гарантують стабільність, продуктивність і простоту використання програмного комплексу. Архітектура проєкту передбачає чотири основні категорії користувачів: власника, менеджера, касира та клієнта. Кожен із них має свій набір функціональних можливостей і адаптований інтерфейс для взаємодії із системою відповідно до своїх повноважень.

Важливою складовою є Telegram-бот, який виступає основним засобом комунікації для клієнтів. Реалізований із застосуванням мови програмування C# та ASP.NET, бот базується на Telegram Bot API, що забезпечує гнучкі можливості для створення інтерактивних сервісів, обробки клієнтських запитів і виконання

необхідних бізнес-операцій. Функціонал бота дозволяє користувачам створювати бронювання та переглядати актуальні курси валют у будь-який час. Для тимчасового збереження даних і кешування запитів застосовується Redis, що забезпечує швидке реагування системи та стабільну роботу навіть при великій кількості запитів.

Для адміністративного управління в системі використано сучасний шаблон Materio, адаптований для потреб проєкту. Шаблон побудований на базі технологій Vue.js та Vuetify із застосуванням HTML, SCSS і JavaScript. Це дозволяє створити сучасний, інтуїтивно зрозумілий та візуально привабливий інтерфейс для адміністраторів, касирів, менеджерів та власників. Касири отримують можливість виконувати валютно-обмінні операції та реєструвати їх у системі, менеджери — додавати нові валюти, встановлювати курси та керувати параметрами, а також створювати бронювання за запитами клієнтів.

Основою для збереження даних у системі виступає реляційна база даних PostgreSQL. Ця система управління базами даних забезпечує стабільне та безпечне збереження інформації про користувачів, фінансові операції, курси валют, резерви й контактні дані. Її переваги — висока швидкодія, можливість ефективного масштабування та стабільна робота в умовах великої кількості одночасних запитів.

До складу технічного забезпечення увійшли такі технології та методи. Мова C# разом із фреймворком ASP.NET використовується для побудови серверної логіки системи та Telegram-бота, забезпечуючи високу продуктивність і гнучкість розробки. Redis служить для кешування й оперативного збереження тимчасових даних, зменшуючи навантаження на базу даних та підвищуючи швидкість обробки запитів. Інтерфейсна частина адміністрування реалізована за допомогою Vue.js і Vuetify — сучасних фреймворків, що забезпечують комфортну розробку та якісну візуалізацію. PostgreSQL виступає основною СУБД, яка забезпечує надійне довгострокове зберігання даних, реалізацію складних запитів і підтримку цілісності даних.

Комплексне поєднання цих технологічних рішень дозволяє створити надійну, функціональну та масштабовану систему обліку валютно-обмінних операцій, яка забезпечує стабільну роботу, зручність в обслуговуванні та високу продуктивність. Вичерпний перелік використаних бібліотек, утиліт та додаткових програмних компонентів наведено в таблиці 3.1, що містить повний опис програмного середовища розробки.

Таблиця 3.1 – Перелік стеку технологій

№ п/п	Назва утиліти	Опис застосування
1	Visual Studio	Головне IDE для серверної частини проєкту.
2	Postman	Засіб для перевірки API та клієнтських запитів до сервера.
3	Drawio	Онлайн-платформа для створення діаграм і схем.
4	Redis Insight	Утиліта для побудови схем без встановлення програм.
5	WebStorm	Вебпанель для перегляду даних і адміністрування Redis-бази.
6	DataGrip	Інтегроване середовище для розробки на JavaScript, HTML, CSS.
7	SourceTree	IDE для адміністрування та роботи з базами даних.
8	Github	Графічний клієнт для взаємодії з системами Git та Mercurial.
9	ngrok	Платформа для хостингу проєктів із системою контролю версій Git.
10	Materio	Сервіс для створення захищених тунелів між локальним ПК і мережею.

3.2 Аналіз безпеки інформації

Під час створення системи з Telegram-ботом, спрямованої на оптимізацію та автоматизацію обліку у пункті обміну валют, особлива увага була приділена питанням захисту даних.

Для збереження тимчасових даних використовується Redis. Ця технологія застосовується для кешування і зберігання тимчасової інформації, зокрема сесійних даних користувачів і оперативних повідомлень. Redis характеризується високою швидкістю доступу до інформації та забезпечує продуктивність системи. Безпека Redis забезпечується через впровадження шифрування та обмеження доступу, який надається лише авторизованим сервісам і додаткам.

Постійне зберігання даних здійснюється за допомогою PostgreSQL. Ця система управління базами даних використовується для надійного збереження відомостей про клієнтів, їхні операції та інші важливі записи. PostgreSQL відрізняється стабільністю, гнучкістю і підтримкою різних методів автентифікації та шифрування, що гарантують захист конфіденційної інформації та її цілісність. Для недопущення несанкціонованого доступу реалізується суворий контроль прав доступу до бази.

Адміністративний інтерфейс, створений на базі Materio з використанням Vue.js, застосовується для керування та моніторингу системи обміну валют. Для підвищення рівня безпеки впроваджені додаткові механізми: доступ до адміністративних функцій регламентується ролями (власник, менеджер, касир), а також реалізована автентифікація через токени та сесійні ключі, що забезпечує захищений вхід та контроль активності користувачів.

3.3 Реалізація основних модулів системи

При реалізації програмного забезпечення можна виділити 3 ключові модулі, які формують ядро бізнес-логіки проєкту обміну валют:

1. CurrencyModule – управління валютами та курсами.

Основні файли: CurrencyRepository.cs, RateRepository.cs, ExchangeSettingsRepository.cs, CurrencyManager.cs, RateManager.cs, ExchangeSettingManager.cs.

Цей модуль відповідає за логіку створення, оновлення, видалення та активації валют, зберігання і встановлення курсів купівлі/продажу, а також конфігурацій обміну (відсотки доходу, пороги тощо).

```
    public async Task<Currency> GetCurrencyAsync(Guid id)
{
        var entity = await _context.Currencies
            .Where(x => x.Id == id && x.IsActive == true)
            .FirstOrDefaultAsync();
        if (entity == null) {
            throw new RecordNotFoundException(404,
"DataAccess", "Currency not found");
        }
        return entity;
    }
```

Репозиторії працюють із DbContext, менеджери – з DTO та координацією інших сервісів (наприклад, автоматичне створення Fund та Rate для нової валюти).

2. ReservationModule – управління бронюваннями валют.

Основні файли: ReservationRepository.cs, ReservationManager.cs.

Призначений для обробки запитів на бронювання валютних операцій. Включає зв’язки з курсами, операціями, контактами та статусами бронювання.

```
    public async Task<Reservation>
GetReservationAsync(Guid id) {
        var entity = await _context.Reservations
            .Include(x => x.OperationType)
            .Include(x => x.Status)
            .Include(x => x.Contact)
```

```
            .Include(x => x.Rate).ThenInclude(r =>
r.Currency)
            .Where(x => x.Id == id && x.IsActive == true)
            .AsNoTracking()
            .FirstOrDefaultAsync();

        if (entity == null) {
            throw new RecordNotFoundException(404,
"DataAccess", "Reservation not found");
        }
        return entity;
    }
```

Репозиторій надає CRUD-операції над бронюваннями. Менеджер відповідає за оркестрацію – передає DTO і забезпечує валідацію логіки на вищому рівні.

3. TransactionModule – управління транзакціями та фондами

Основні файли: TransactionRepository.cs, FundRepository.cs, TransactionManager.cs, FundManager.cs

Призначення - центральна логіка виконання обміну валюти. Вона оновлює резерви (Funds) відповідно до налаштувань і типу операції, а також змінює статус пов’язаного бронювання.

```
    private async Task UpdateFundForSellTypeAsync(FundDto
fund, InsertTransactionDto transaction) {
        var setting = await
_settingManager.GetSettingAsync();
        var baseFund = await
_fundManager.GetFundByCurrencyIdAsync(setting.BaseCurrencyI
d);
        var acceptedAmount = Math.Round(transaction.Amount
* transaction.RateLog.SellRate);
```

```
        baseFund.Amount += acceptedAmount;
        await _fundManager.UpdateFundAsync(baseFund.Id,
_mapper.Map<InsertFundDto>(baseFund));
        fund.Amount -= transaction.Amount;
        await _fundManager.UpdateFundAsync(fund.Id,
_mapper.Map<InsertFundDto>(fund));
    }
```

Окрім створення транзакцій, модуль також керує резервами та адаптує фінансові баланси залежно від типу транзакції (купівля/продаж).

4. UserModule – користувачі та автентифікація

Основні файли: UserRepository.cs, UserManager.cs, UserRoleRepository.cs, UserRoleManager.cs.

Призначення – реєстрація, логін, управління ролями користувачів. Логін обробляється із генерацією JWT-токенів.

```
    public async Task<AuthUserDto> LoginAsync(string
login, string password) {
        var user = await _service.GetUserAsync(login);
        if (user == null ||
!JwtTokenProvider.VerifyPassword(password,
user.PasswordHash)) {
            throw new Exception("Invalid login or
password");    }
        var authUser = new AuthUserDto {
            User = user,
            Token = JwtTokenProvider.GenerateToken(login,
user.Role.Name)
        };
        return authUser;}
```

Модуль забезпечує керування правами доступу та захищеним зберіганням паролів.

Ці чотири модулі утворюють повноцінну логічну архітектуру системи – від управління валютами до обробки користувачів і транзакцій.

3.4 Інтерфейс програми для організації обліку роботи пункту обміну валюти

У рамках роботи платформи передбачено три рівні доступу: власник, менеджер і касир. Хоча усі ролі мають спільний функціонал, деякі дії доступні лише користувачам із підвищеним рівнем доступу. Далі проаналізуємо можливості власника, які водночас охоплюють доступний функціонал і для інших ролей, за винятком обмежених прав.

Коли власник вводить URL-адресу, вказану під час встановлення системи, відкривається інтерфейс входу до системи (рис. 3.1).

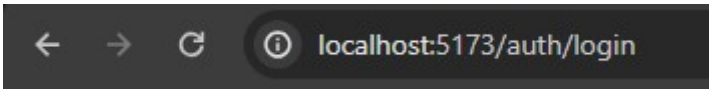

Рисунок 3.1 – Перехід до системи через браузер

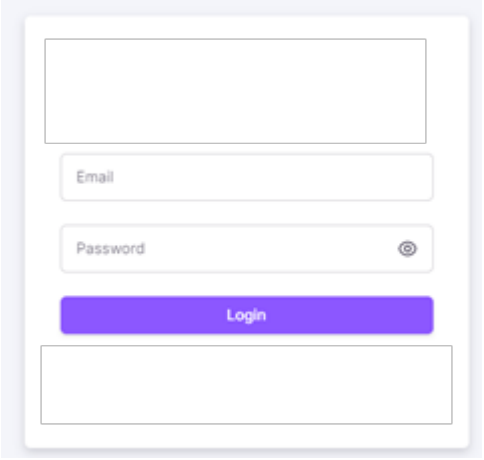

Рисунок 3.2 – Форма входу до облікового запису

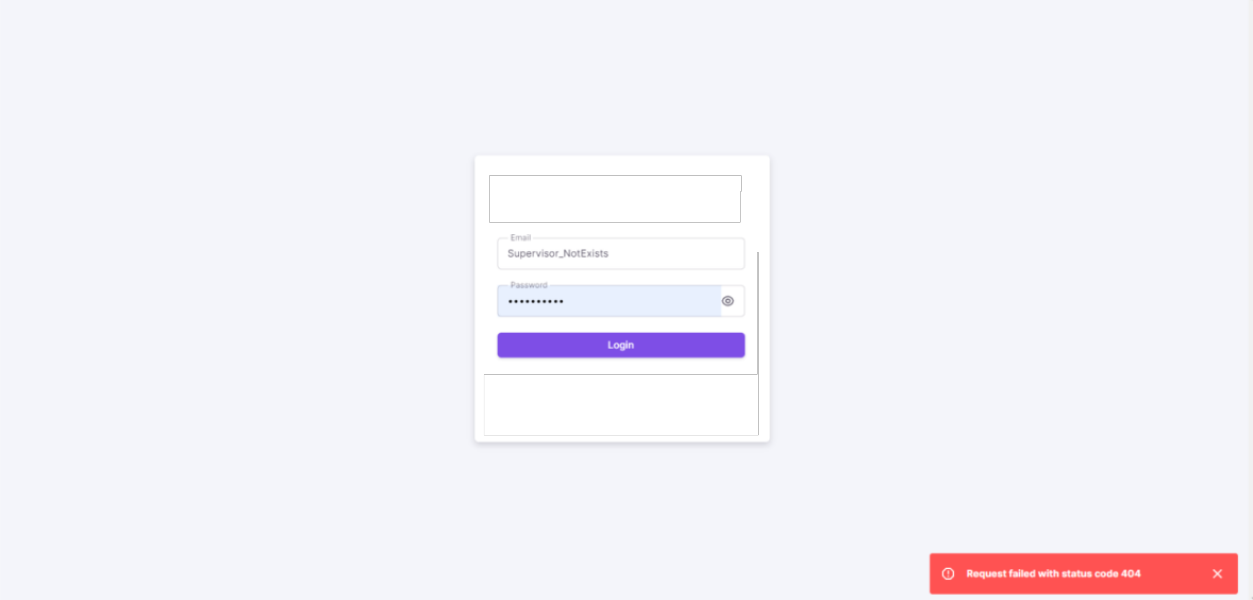

Рисунок 3.3 – Повідомлення про невірні облікові дані

Після успішного входу власник бачить головне вікно з інформаційними дашбордами, які відображають ключові фінансові показники (рис. 3.4).

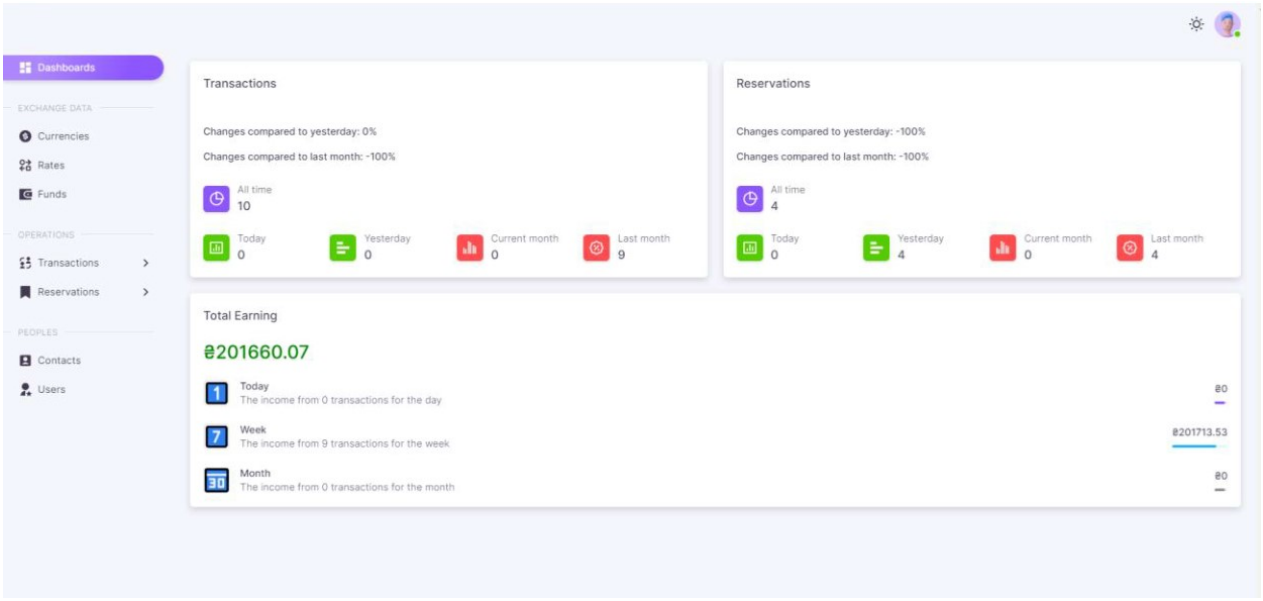

Рисунок 3.4 – Панель моніторингу обмінного пункту

Іконка користувача дозволяє переглянути особисті дані облікового запису. У розкритому вікні видно логін користувача та його роль у системі (рис. 3.5).

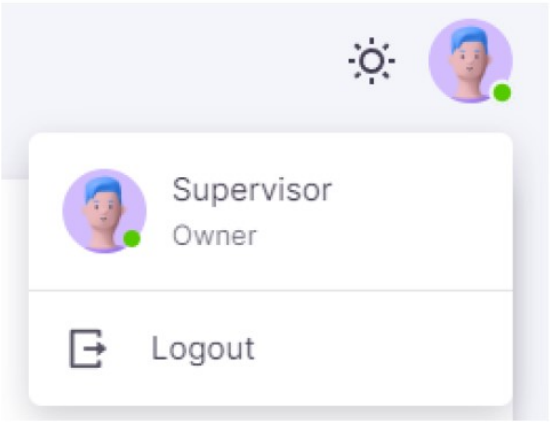

Рисунок 3.5 – Відомості про авторизованого користувача

Подальша робота у системі здійснюється через навігаційне меню, розташоване зліва (рис. 3.6).

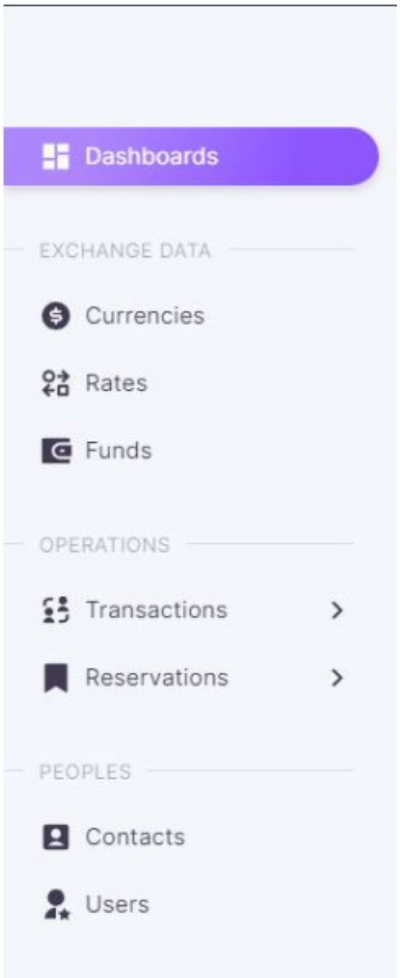

Рисунок 3.6 – Головне меню функцій системи

На вкладці "Currencies" користувач переглядає наявні валюти (рис. 3.7).

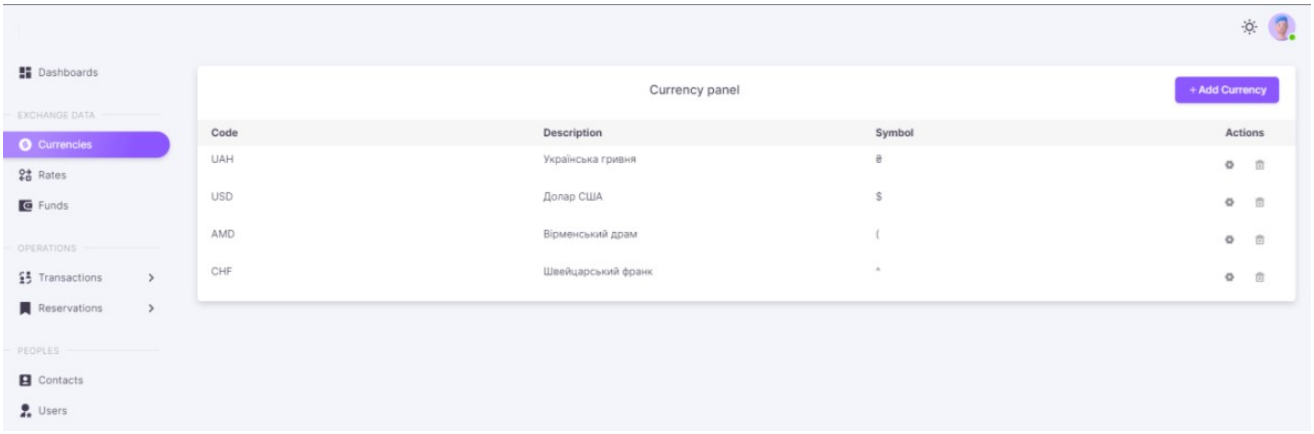

Рисунок 3.7 – Перелік активних валют

Нова валюта створюється натисканням кнопки "Add currency" (рис. 3.8).

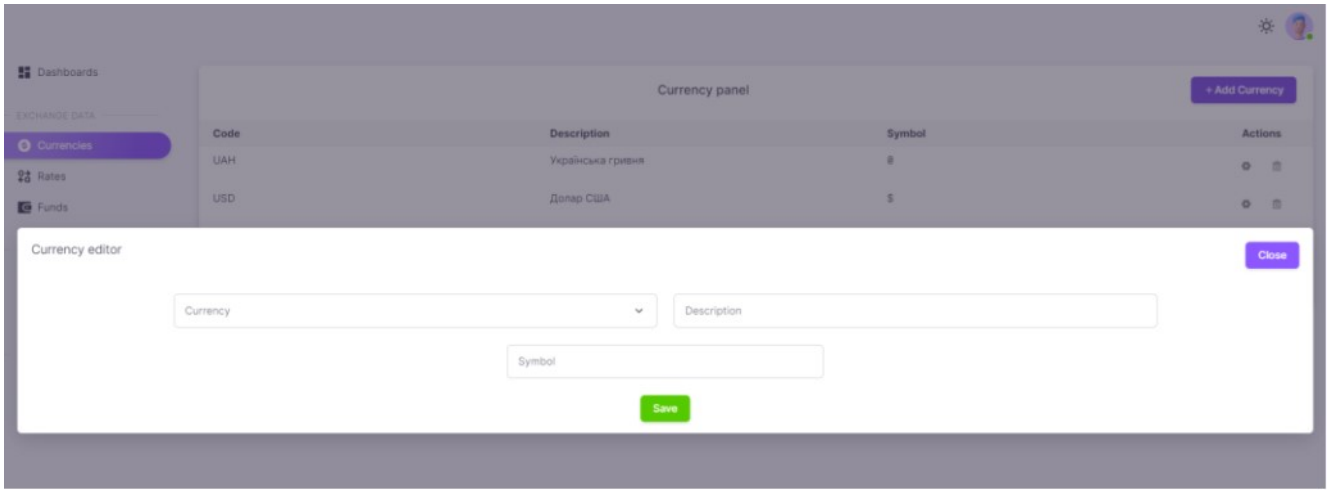

Рисунок 3.8 – Форма додавання нової валюти

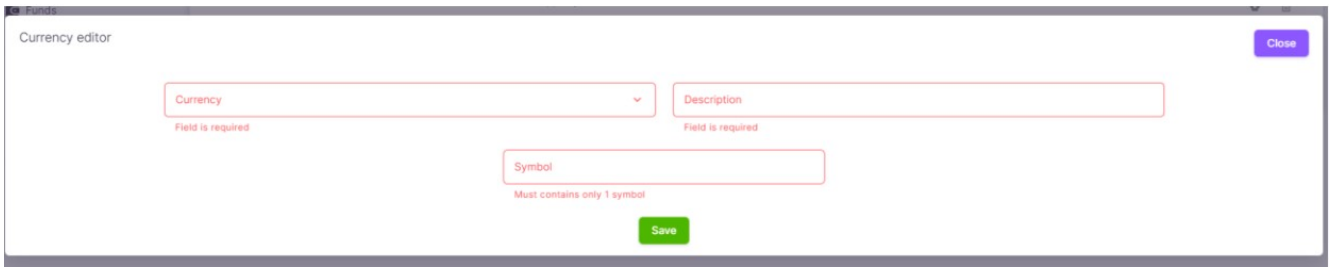

Рисунок 3.9 – Попередження про порожні обов’язкові поля

Якщо обов’язкові поля не заповнено, система виводить підказку щодо необхідності їх введення (рис. 3.9).

Поле "Currency" заповнюється на основі переліку валют НБУ (рис. 3.10).

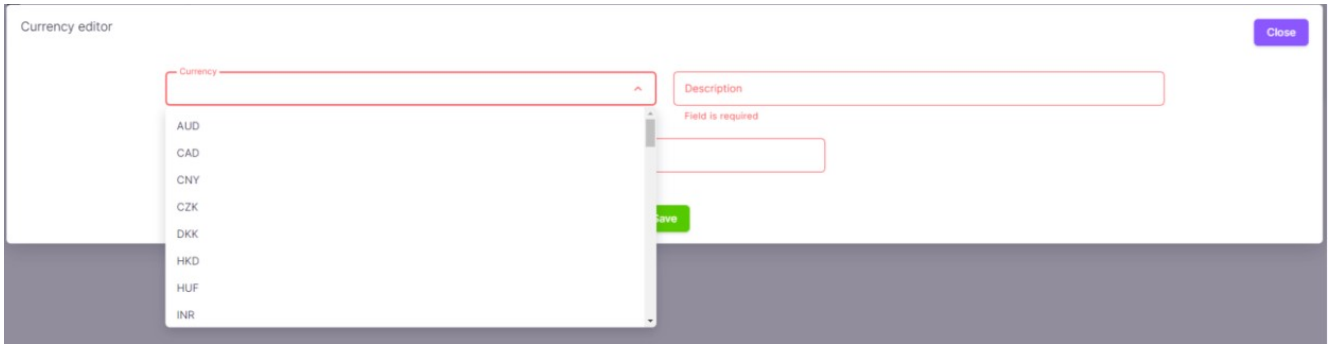

Рисунок 3.10 – Перелік валют, офіційно затверджених НБУ

Опис валюти автозаповнюється, але його можна змінити.

Після збереження нової валюти з’являється підтвердження успішного додавання (рис. 3.11).

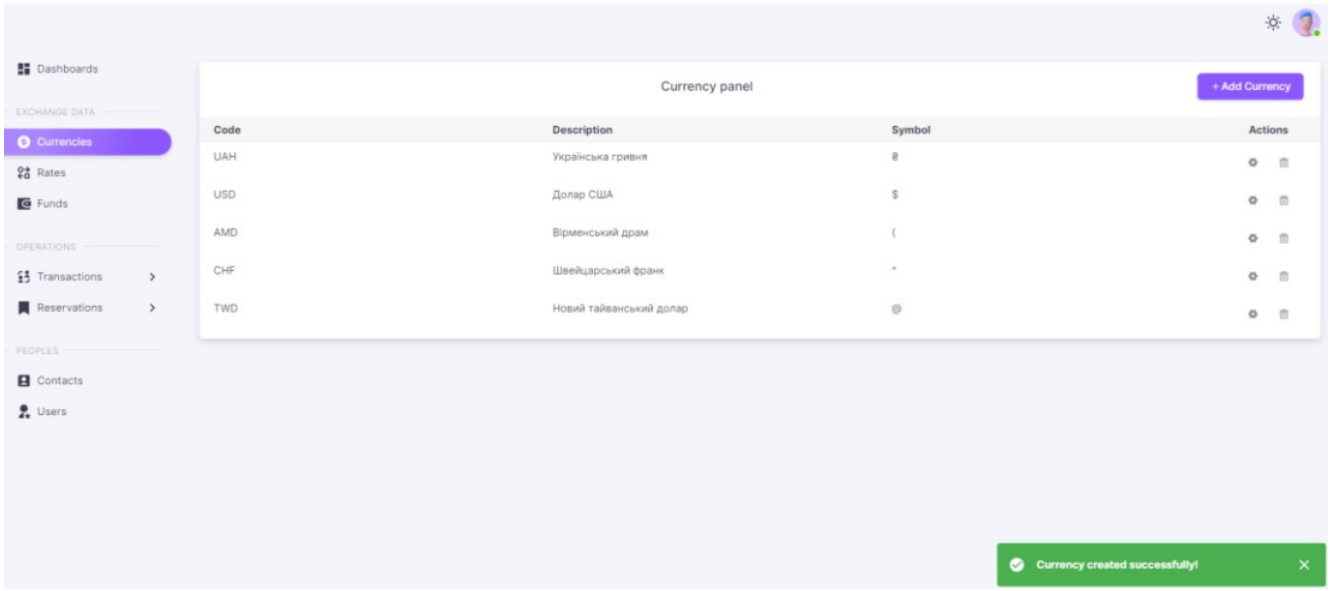

Рисунок 3.11 – Повідомлення про успішне створення валюти

Для редагування наявної валюти натискається іконка редагування. У новому вікні редагуються поля, після чого натискається кнопка збереження.

Для деактивації валюти використовується кнопка з іконкою смітника. Система сповіщає про успішне видалення

Далі користувач переходить до розділу "Rates", де відображаються наявні курси валют (рис. 3.12).

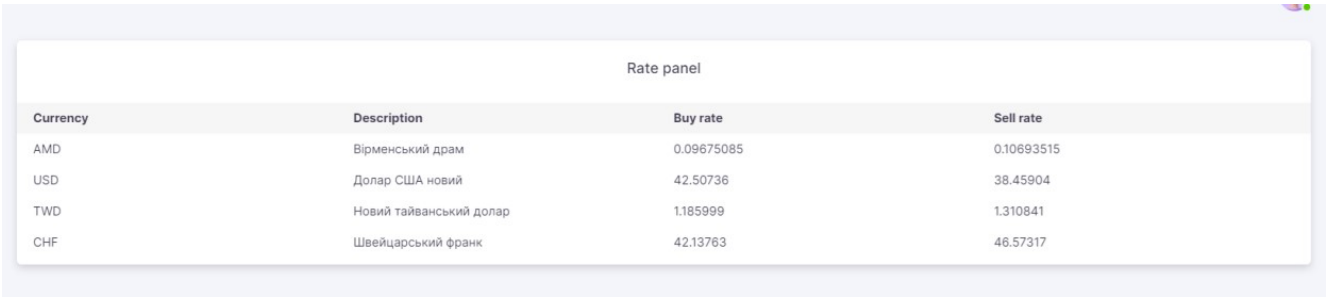

Рисунок 3.12 – Поточні курси валют у системі

Для управління резервами слід перейти до вкладки "Funds". Там користувач бачить обсяг доступних залишків для кожної валюти (рис. 3.13).

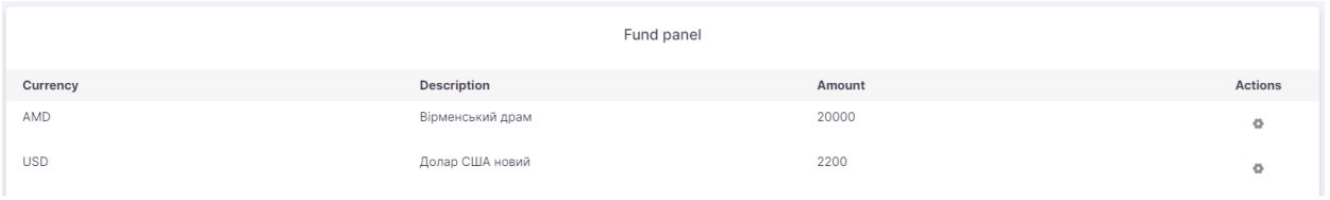

Рисунок 3.13 – Загальний вигляд таблиці з резервами

Контактні дані клієнтів зберігаються у секції "Contacts". Тут можна переглядати наявні контакти та створювати нові. Інтерфейс дозволяє змінити або видалити контактну інформацію.

Бронювання створюються у секції "Reservations". Користувач має змогу обрати опцію створення або перегляду (рис. 3.14).

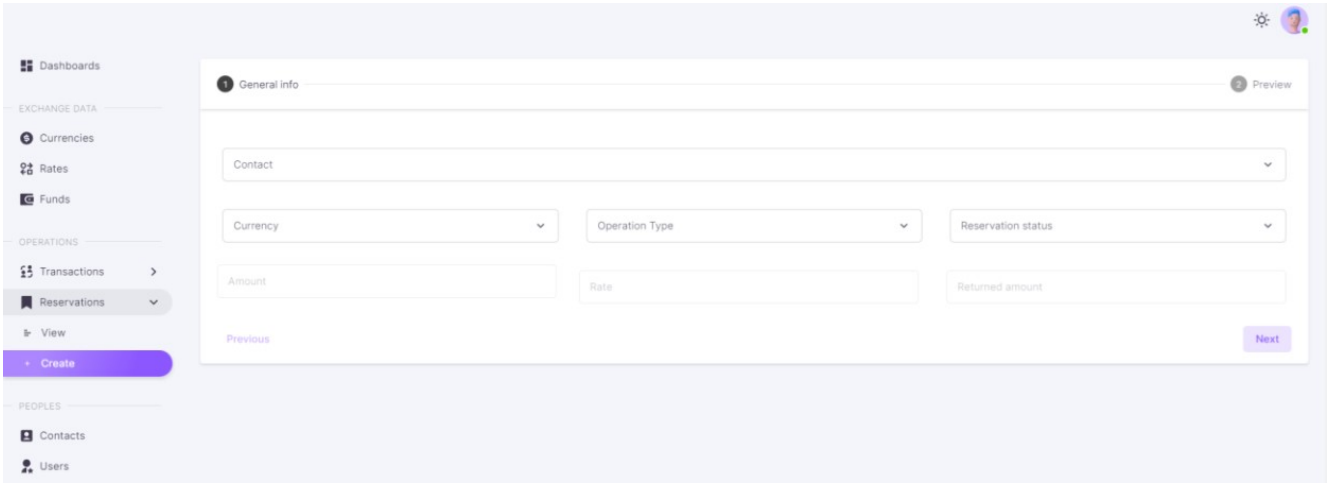

Рисунок 3.14 – Вікно створення бронювання

Для фіксації фактичних транзакцій використовується модуль "Transactions". Користувач створює нову операцію, вводить дані, підтверджує і переглядає результат.

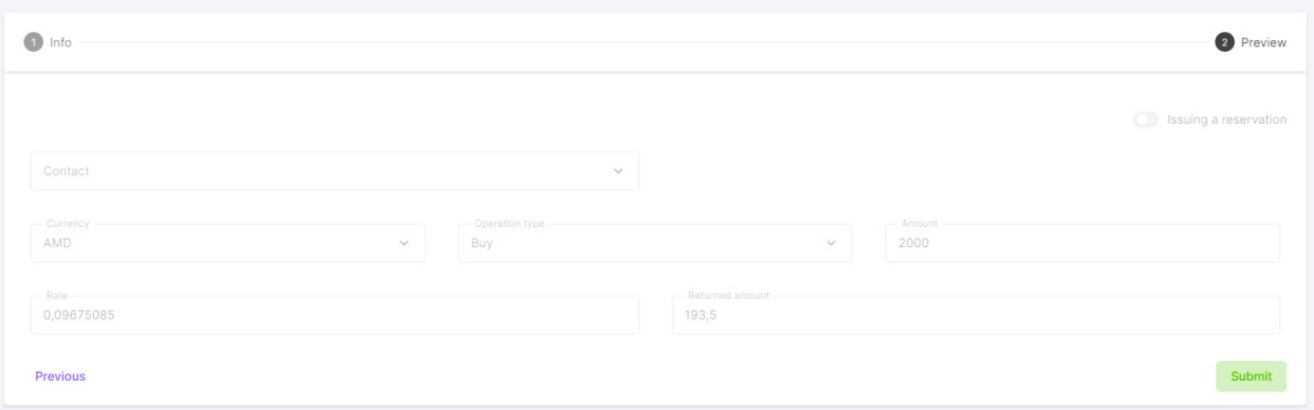

Рисунок 3.15 – Підсумкове збереження транзакції

Адміністративна частина платформи миттєво фіксує кожне нове бронювання або контакт, що дає змогу менеджерам оперативно реагувати на запити клієнтів через інтерфейс основної системи.

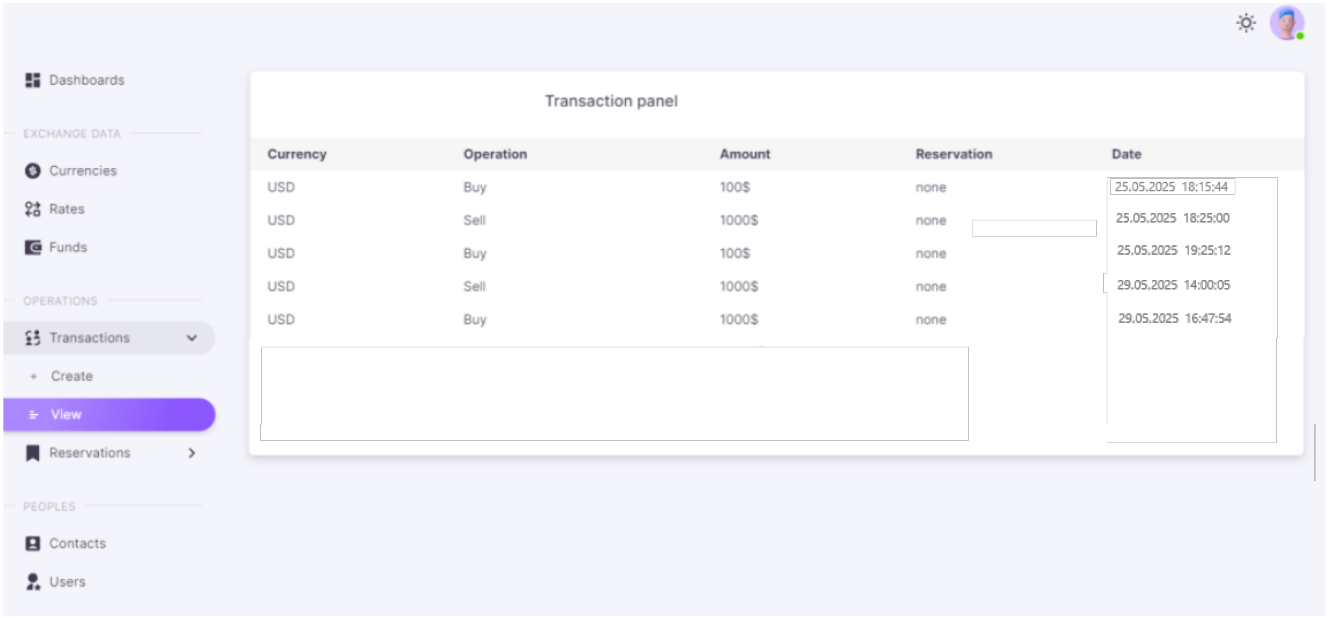

Рисунок 3.16 – Таблиця транзакцій

Після завершення роботи необхідно вийти з системи, натиснувши "Logout".

Для зручної взаємодії клієнта з пунктом обміну реалізовано Telegram-бота. Після знаходження бота за назвою, користувач виконує команду /start (рис. 3.17).

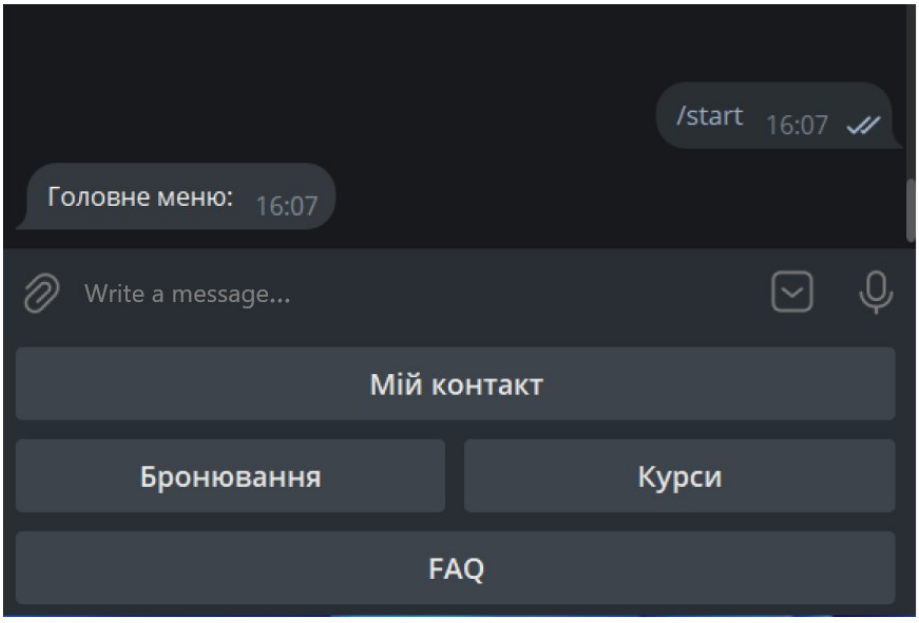

Рисунок 3.17 – Стартове меню бота

Щоб скористатися ботом, клієнт створює свій контакт, натискаючи на "Мій контакт" і заповнюючи відповіді. Після створення контактних даних можна перевірити результат. Кнопка "Курси" дозволяє ознайомитись з наявними курсами валют, вибираючи потрібну.

Для створення бронювання потрібно перейти до відповідного розділу через кнопку "Бронювання".

3.5 Результати тестування розробленої програми

Цілями проведення тестування є наступні завдання:

− перевірка відповідності функціональним вимогам розробленого програмного продукту;

− оцінка процесу створення бронювань клієнтом через Telegram-бот;

− перевірка додавання бронювань адміністратором через систему;

− тестування оновлення статусів бронювань адміністраторами;

− контроль правильності відображення бронювань у інтерфейсі адміністратора;

− перевірка коректного відображення користувачеві бронювань у Telegram-боті;

− тестування додавання транзакцій адміністраторами в системі;

− перевірка створення транзакцій, що відповідають окремим бронюванням, адміністраторами;

− тестування можливості створення нових валют у системі;

− перевірка функціоналу зміни даних про валюту адміністраторами;

− оцінка процесу видалення валют із системи;

− тестування відображення наявних валют у системному інтерфейсі адміністратора;

− перевірка відображення актуальних курсів валют в інтерфейсі адміністратора;

– тестування перегляду курсів валют клієнтами через Telegram-бот;

– оцінка відображення резервів валют у системі адміністратора;

– перевірка оновлення резервів для окремих валют;

– тестування створення контактної інформації адміністратором;

– перевірка можливості додавання контакту клієнтом через Telegram-бот;

– тестування створення нових облікових записів користувачів системи адміністраторами;

– перевірка зміни ролей користувачів у системі;

– оцінка точності обчислень, що відображаються в аналітичних панелях (дашбордах) адміністратора;

– виявлення та ідентифікація помилок, дефектів або недоліків з метою їх подальшого усунення.

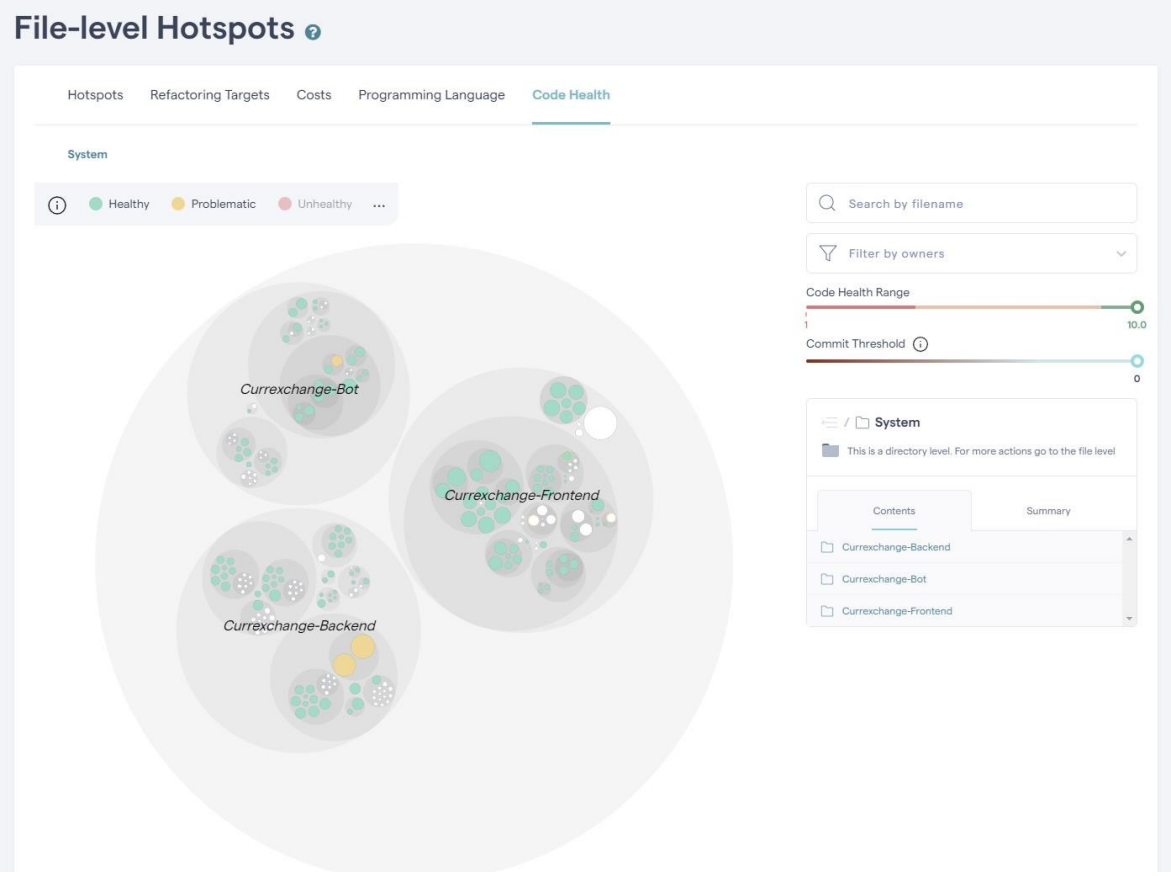

Рисунок 3.18 – Візуалізація результатів оцінювання коду за допомогою CodeScene

Процес контролю якості та визначення ефективності програмного забезпечення є одним із ключових етапів при його створенні. Для дотримання високих стандартів якості та своєчасного виявлення проблемних ділянок застосовуються засоби кількісного аналізу — метрики. Саме вони забезпечують об’єктивне вимірювання таких характеристик програмного продукту, як його складність, продуктивність, об’єм і стабільність, що дає змогу приймати виважені рішення щодо оптимізації системи.

Для цієї мети було застосовано аналітичну платформу CodeScene [12]. Візуальні результати оцінювання подано на рисунках 3.18–3.20.

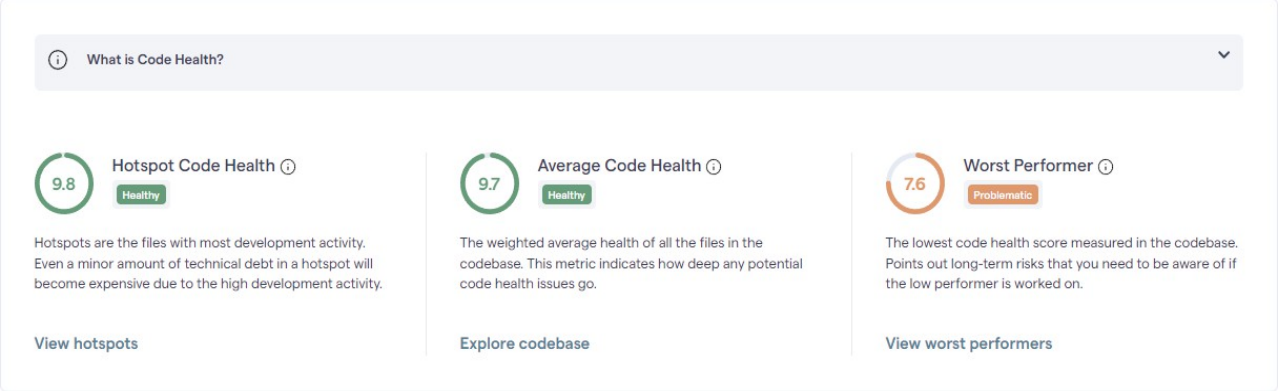

Рисунок 3.19 – Загальний звіт аналізу коду, згенерований платформою CodeScene

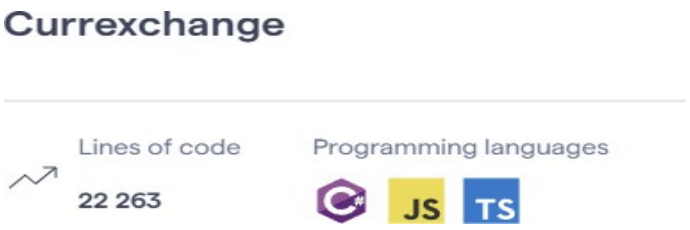

Рисунок 3.20 – Підрахунок рядків коду у проєкті

Було проведено ручне тестування всіх основних функцій програмного рішення. Опис і результати тестів наведено в таблицях 3.2– 3.11.

Таблиця 3.2 містить тест-кейс, що перевіряє процес входу користувача в систему. Вона описує початкові умови, дії користувача під час авторизації, а також очікуваний і фактичний результати.

Таблиця 3.2 - Перевірка 1.1

Назва тесту	Авторизація користувача в системі
Код тесту	1.1
Початковий стан	Користувач відкрив сторінку входу до облікового запису
Вхідна інформація	Відсутня
Опис дій	Користувач вводить свої облікові дані (логін і пароль), після чого натискає кнопку "Login"
Очікуваний результат	Користувач успішно авторизується, система переадресовує його на сторінку з аналітичними панелями
Реальний результат	Авторизація пройшла успішно, система відкрила дашборди

Таблиця 3.3 – Перевірка 1.2

Назва тесту	Додавання нової валюти в систему
Код тесту	1.2
Початковий стан	Користувач перебуває на сторінці перегляду валют
Вхідна інформація	Відсутня
Опис дій	Користувач натискає кнопку “Add currency”. Відкривається діалогове вікно, де користувач заповнює обов'язкові поля та натискає “Save”
Очікуваний результат	До переліку валют додається нова одиниця. Автоматично створюються відповідні курс і резерв
Реальний результат	Валюта була додана, у системі створено курс і резерв для неї

Таблиця 3.4 – Перевірка 1.3

Назва тесту	Додавання нової транзакції
Код тесту	1.3
Початковий стан	Відкрито сторінку додавання транзакції
Вхідна інформація	Відсутня
Опис дій	Користувач заповнює всі поля форми, натискає “Next”, переходить на сторінку перевірки введених даних, після чого натискає “Submit”
Очікуваний результат	Користувач потрапляє до списку транзакцій. У цьому списку з’являється нова транзакція, а резерв валюти оновлюється відповідно до суми
Реальний результат	Система успішно перенаправила до переліку транзакцій, нова транзакція з’явилась у списку. Резерв валюти змінено згідно даних

Таблиця 3.5 – Перевірка 1.4

Назва тесту	Формування нового бронювання через інтерфейс
Код тесту	1.4
Початковий стан	Користувач перебуває на сторінці створення бронювання
Вхідна інформація	Відсутня
Опис дій	Користувач заповнює поля форми, натискає “Next” для переходу до перевірки введених даних, після чого натискає кнопку “Submit”
Очікуваний результат	Користувача автоматично перенаправляє на загальну сторінку бронювань, нове бронювання відображається у списку
Реальний результат	Система виконує перенаправлення до переліку транзакцій, в якому видно новостворене бронювання

Таблиця 3.6 – Перевірка 1.5

Назва тесту	Додавання нового контакту через веб-інтерфейс
Код тесту	1.5
Початковий стан	Користувач знаходиться на сторінці створення контакту
Вхідна інформація	Відсутня
Опис дій	Користувач натискає кнопку “Add contact”, у діалоговому вікні вводить необхідну інформацію, натискає “Next” і підтверджує створення натисканням на “Submit”
Очікуваний результат	Діалогове вікно закривається, і в загальному списку з’являється новий контакт
Реальний результат	Список контактів оновлено, новий контакт збережено успішно

Таблиця 3.7 – Перевірка 1.6

Назва тесту	Активація Telegram-бота
Код тесту	1.6
Початковий стан	Користувач знаходиться в чаті з ботом
Вхідна інформація	Відсутня
Опис дій	Користувач надсилає команду “/start”
Очікуваний результат	Бот відповідає повідомленням і відкриває головне меню
Реальний результат	Telegram-бот стартує і надсилає меню головного навігаційного вікна

Таблиця 3.8 – Перевірка 1.7

Назва тесту	Додавання контакту користувачем через Telegram-бот
Код тесту	1.7
Початковий стан	Користувач знаходиться в головному меню Telegram-бота
Вхідна інформація	Відсутня
Опис дій	Користувач натискає кнопку “Створити контакт”, після чого бот надсилає запитання, на які користувач послідовно відповідає
Очікуваний результат	Після завершення введення контакт створюється, а користувач отримує відповідне повідомлення
Реальний результат	Контакт додано успішно, користувач отримав підтвердження в чаті з ботом

Таблиця 3.9 – Перевірка 1.8

Назва тесту	Створення бронювання клієнтом через Telegram
Код тесту	1.8
Початковий стан	Користувач у головному меню бота
Вхідна інформація	Відсутня
Опис дій	Користувач натискає “Бронювання” → “Нове бронювання”, бот надсилає список доступних валют, користувач вибирає одну, потім обирає тип транзакції, вводить суму, підтверджує введені дані
Очікуваний результат	У системі з’являється нове бронювання зі статусом “У процесі”, бот повідомляє користувача
Реальний результат	Бронювання успішно створено, бот надіслав повідомлення з підтвердженням і статусом “In progress”

Таблиця 3.10 – Перевірка 1.9

Назва тесту	Перегляд клієнтом своїх бронювань у боті
Код тесту	1.9
Початковий стан	Користувач знаходиться у головному меню Telegram-бота
Вхідна інформація	Відсутня
Опис дій	Користувач обирає “Бронювання” → “Мої бронювання”, бот формує відповідь
Очікуваний результат	Користувачу надсилається перелік усіх його бронювань, збережених у системі
Реальний результат	Список бронювань передано користувачу через бот відповідно до очікуваного сценарію

Таблиця 3.11 – Перевірка 1.10

Назва тесту	Перегляд користувачем актуальних курсів валют у боті
Код тесту	1.10
Початковий стан	Користувач відкрив головне меню Telegram-бота
Вхідна інформація	Відсутня
Опис дій	Користувач натискає на кнопку “Курси”, бот надсилає перелік валют, після чого користувач вибирає потрібну
Очікуваний результат	Користувач отримує актуальний курс обраної валюти
Реальний результат	Бот надсилає дані про курс згідно обраної валюти. Виведено коректну інформацію

За результатами усіх перевірок проведено необхідні доопрацювання, завдяки чому якість програмного забезпечення значно підвищено, а система стала більш стабільною і зручною для кінцевих користувачів.

В результаті розробки програмного забезпечення для обліку діяльності пункту обміну валют створено ефективну, масштабовану та безпечну систему, яка забезпечує повноцінну автоматизацію ключових бізнес-процесів. Для реалізації рішення було використано сучасний технологічний стек, що включає C#, ASP.NET, PostgreSQL, Redis, Telegram Bot API, Vue.js і Materio, що дозволило досягти високої продуктивності, стабільності та гнучкості.

Реалізована архітектура покриває повний життєвий цикл валютно-обмінних операцій — від створення та управління валютами й курсами до обробки транзакцій, бронювань і взаємодії з клієнтами через Telegram-бот. Всі модулі системи мають чітко визначені зони відповідальності, що сприяє зручному супроводу, масштабуванню та підтримці безпеки на всіх рівнях. Інтерфейс адміністративної частини розроблений з урахуванням зручності користувачів різних ролей: власників, менеджерів, касирів і клієнтів.

Проведене тестування функціональності та аналіз якості коду з використанням інструментів на кшталт CodeScene підтвердили відповідність системи функціональним і нефункціональним вимогам. На основі результатів були усунені виявлені дефекти, що дало змогу підвищити якість програмного

продукту. Розроблена система відповідає сучасним вимогам до фінансових вебдодатків, забезпечуючи надійність, інтерактивність і простоту використання як для кінцевих користувачів, так і для адміністративного персоналу.

Незначне зниження оцінки метрики Maintainability вказує на те, що підтримка деяких компонентів може бути утрудненою, однак за іншими параметрами порушень не виявлено.

Крім автоматизованого аналізу, було реалізовано серію ручних тестів, які охопили основні дії користувачів. Зокрема, перевірено обробку некоректних введень, роботу з валютами, бронюваннями, контактами та Telegram-ботом.

За результатами усіх перевірок проведено необхідні доопрацювання, завдяки чому якість програмного забезпечення значно підвищено, а система стала більш стабільною і зручною для кінцевих користувачів.

ВИСНОВКИ

В дипломній роботі було здійснено розробку програми для організації обліку роботи пункту обміну валюти засобами C# з ASP .NET та Vue.js.

В першому розділі було проведено детальний аналіз предметної області та визначено основні вимоги до розробки та впровадження програми для організації обліку роботи пункту обміну валюти засобами C# з ASP .NET та Vue.js.

Проведений аналіз предметної області дозволив окреслити ключові особливості організації роботи пунктів обміну валют, їх основні бізнес-процеси, вимоги до обліку та контролю валютно-обмінних операцій, а також сформувати уявлення про необхідність інтеграції сучасних інформаційних технологій у цю сферу. Установлено, що діяльність таких пунктів пов’язана з обробкою великого обсягу фінансових даних, контролем залишків валюти, веденням історії транзакцій, бронюванням коштів клієнтів та формуванням звітів відповідно до чинного законодавства і нормативних актів.

Аналіз наявного програмного забезпечення та технічних рішень показав, що на ринку представлено низку потужних платформ для автоматизації бізнес-процесів, таких як Creatio, Salesforce та Zoho, які мають широкий функціонал і високу надійність, однак потребують значних фінансових ресурсів для впровадження та обслуговування. Також було розглянуто спеціалізовані сервіси, які надають актуальну інформацію про курси валют і цифрових активів, що дозволяє ефективно організовувати аналітику фінансових ринків і приймати обґрунтовані рішення.

На основі проведеного аналізу сформульовано завдання проєкту зі створення власної програмної системи для обліку діяльності пункту обміну валюти. Запропоновано реалізувати дві взаємопов’язані платформи: Telegram-бота для клієнтського сервісу та адміністративний вебінтерфейс для персоналу та керівництва. Визначено основні функціональні можливості системи, перелік користувацьких ролей і технічні засоби, що будуть використані для розробки,

зокрема мови програмування C#, ASP.NET, технології Vue.js та реляційну СУБД PostgreSQL. У межах поставлених завдань окреслено функціональні модулі та операції, які повинна підтримувати система для забезпечення ефективної, прозорої та безпечної роботи пункту обміну валют.

В другому розділі була здійснена розробка структури програмного забезпечення для організації обліку роботи пункту обміну валюти, яка дала змогу створити логічно цілісну архітектуру, що охоплює всі основні бізнес-процеси й механізми взаємодії користувачів із системою. Змодельовані діаграми варіантів використання та послідовності дозволили детально описати порядок виконання ключових операцій, забезпечивши прозорість та узгодженість процесів.

База даних спроєктована з урахуванням нормалізації, контролю зв'язків і актуальності даних, що гарантує її стабільну роботу та надійне збереження інформації.

Особливу увагу приділено проектуванню прототипів інтерфейсів, які відповідають сучасним стандартам зручності та доступності, забезпечуючи швидкий доступ до функцій для всіх категорій користувачів. Впроваджена клієнт-серверна архітектура на основі MVC-патерну та REST-стилю дозволяє ефективно організувати обробку запитів, підтримку резервів, роботу з транзакціями й бронюваннями в реальному часі.

В результаті створено універсальну, надійну та масштабовану систему, адаптовану до потреб пунктів обміну валют.

В результаті розробки програмного забезпечення для обліку діяльності пункту обміну валют створено ефективну, масштабовану та безпечну систему, яка забезпечує повноцінну автоматизацію ключових бізнес-процесів. Для реалізації рішення було використано сучасний технологічний стек, що включає C#, ASP.NET, PostgreSQL, Redis, Telegram Bot API, Vue.js і Materio, що дозволило досягти високої продуктивності, стабільності та гнучкості.

Реалізована архітектура покриває повний життєвий цикл валютно-обмінних операцій — від створення та управління валютами й курсами до обробки

транзакцій, бронювань і взаємодії з клієнтами через Telegram-бот. Всі модулі системи мають чітко визначені зони відповідальності, що сприяє зручному супроводу, масштабуванню та підтримці безпеки на всіх рівнях. Інтерфейс адміністративної частини розроблений з урахуванням зручності користувачів різних ролей: власників, менеджерів, касирів і клієнтів.

Проведене тестування функціональності та аналіз якості коду з використанням інструментів на кшталт CodeScene підтвердили відповідність системи функціональним і нефункціональним вимогам. На основі результатів були усунені виявлені дефекти, що дало змогу підвищити якість програмного продукту. Розроблена система відповідає сучасним вимогам до фінансових вебдодатків, забезпечуючи надійність, інтерактивність і простоту використання як для кінцевих користувачів, так і для адміністративного персоналу.

Незначне зниження оцінки метрики Maintainability вказує на те, що підтримка деяких компонентів може бути утрудненою, однак за іншими параметрами порушень не виявлено.

Крім автоматизованого аналізу, було реалізовано серію ручних тестів, які охопили основні дії користувачів. Зокрема, перевірено обробку некоректних введень, роботу з валютами, бронюваннями, контактами та Telegram-ботом.

За результатами усіх перевірок проведено необхідні доопрацювання, завдяки чому якість програмного забезпечення значно підвищено, а система стала більш стабільною і зручною для кінцевих користувачів.

ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Violity shop. URL: https://violity.shop/ (дата звернення: 05.06.2025).
2. Creatio. URL: https://www.creatio.com (дата звернення: 05.06.2025).
3. Salesforce. URL: https://www.salesforce.com (дата звернення: 05.06.2025).
4. Zoho. URL: https://www.zoho.com (дата звернення: 05.06.2025).
5. Minfin. URL: https://minfin.com.ua (дата звернення: 05.06.2025).
6. Національний банк України. URL: https://bank.gov.ua (дата звернення: 06.06.2025).
7. PrivatBank. URL: https://privatbank.ua (дата звернення: 06.06.2025).
8. Криптовалютна біржа Binance. URL: https://www.binance.com (дата звернення: 06.06.2025).
9. Криптовалютна біржа OKX. URL: https://www.okx.com/ (дата звернення: 06.06.2025).
10. PostgreSQL. URL: https://www.postgresql.org (дата звернення: 06.06.2025).
11. Git. URL: https://www.git-scm.com (дата звернення: 07.06.2025).
12. CodeScene. URL: https://codescene.io (дата звернення: 07.06.2025).
13. Microsoft ASP.NET Core Documentation. URL: https://learn.microsoft.com/en-us/aspnet/core/ (дата звернення: 07.06.2025).
14. Vue.js Documentation. URL: https://vuejs.org/guide/introduction.html (дата звернення: 07.06.2025).
15. Entity Framework Core Documentation. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 07.06.2025).
16. SignalR for ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/signalr/ (дата звернення: 07.06.2025).
17. Dapper — a simple object mapper for .NET. URL: https://github.com/DapperLib/Dapper (дата звернення: 07.06.2025).

18. Currency Exchange API. URL: https://apilayer.com/marketplace/exchangerates_data-api (дата звернення: 07.06.2025).

19. ASP.NET Zero: Application Framework. URL: https://aspnetzero.com/ (дата звернення: 07.06.2025).

20. RESTful Web Services with ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/web-api/ (дата звернення: 07.06.2025).

21. Vuex State Management for Vue.js. URL: https://vuex.vuejs.org/ (дата звернення: 07.06.2025).

22. DevExtreme Components for Vue.js. URL: https://js.devexpress.com/Documentation/Guide/Vue_Components/ (дата звернення: 08.06.2025).

ДОДАТКИ

ДОДАТОК А

Фрагменти коду додатку

Файл UserRepository.cs

```
public class UserRepository : BaseRepository, IUserRepository {
private readonly DataAccessContext _context;
public UserRepository(DataAccessContext context) {
_context = context;
}
public async Task<IEnumerable<User>> GetUsersAsync() { return
await Task.FromResult(_context.Users
.Include(x => x.Contact)
.Include(x => x.Role));
}
public async Task<User> GetUserAsync(string login) { var entity
= await _context.Users
.Include(x=>x.Contact)
.Include(x=>x.Role)
.Where(x=>x.Login == login && x.IsActive == true)
.FirstOrDefaultAsync(); if (entity == null) {
throw new RecordNotFoundException(404, "DataAccess", "User with
such login not found");}
return entity;
}
public async Task<User> AddUserAsync(User entity) {
SetDefaultValues(entity);
await _context.Users.AddAsync(entity);
 await _context.SaveChangesAsync(); return entity;
}
public async Task<User> ChangeUserRoleAsync(string login, Guid
roleId)
```

```
{
var entity = await _context.Users
.Where(x => x.Login == login && x.IsActive == true)
.FirstOrDefaultAsync(); if (entity == null) {
throw new RecordNotFoundException(404, "DataAccess",
"User with such id not found");
}
var roleEntity = await _context.UserRoles.FindAsync(roleId); if
(roleEntity == null) {
throw new RecordNotFoundException(404, "DataAccess", "Role with
such id not found");
}
entity.RoleId = roleId; entity.Role = roleEntity;
await _context.SaveChangesAsync(); return entity;
}
public async Task<User> DeleteUserAsync(string login) {
var entity = await _context.Users.Where(x=>x.Login ==
login).FirstOrDefaultAsync();
if (entity == null) {
throw new RecordNotFoundException(404, "DataAccess", "User with
such id not found");
}
SetDeleteDefaultValues(entity);
await _context.SaveChangesAsync(); return entity;
}
}
```

Файл UserRoleRepository.cs

```
public class UserRoleRepository : IUserRoleRepository { private
readonly DataAccessContext _context;
public UserRoleRepository(DataAccessContext context) {
_context = context;
```

```
   }
   public async Task<IEnumerable<UserRole>> GetRolesAsync() {
return await
   Task.FromResult(_context.UserRoles.AsNoTracking().AsEnumerable()
);
   }
   }
```

Файл ReservationRepository.cs

```
   public class ReservationRepository : BaseRepository,
IReservationRepository { private readonly DataAccessContext
_context;
   private readonly IMapper _mapper;
   public ReservationRepository(DataAccessContext context, IMapper
mapper) {
   _context = context;
   _mapper = mapper;
   }
    public async Task<IEnumerable<Reservation>>
GetReservationsAsync() { return await
Task.FromResult(_context.Reservations
   .Include(x => x.OperationType)
   .Include(x => x.Status)
   .Include(x => x.Contact)
   .Include(x => x.Rate)
   .ThenInclude(r => r.Currency)
   .Where(x=>x.IsActive)
   .AsNoTracking());
   }
   public async Task<Reservation> GetReservationAsync(Guid id) {

   var entity = await _context.Reservations
```

```
   .Include(x=>x.OperationType)
   .Include(x=>x.Status)
   .Include(x => x.Contact)
   .Include(x => x.Rate)
   .ThenInclude(r => r.Currency)
   .Where(x => x.Id == id && x.IsActive == true)
   .AsNoTracking()
   .FirstOrDefaultAsync(); if (entity == null) {
   throw new RecordNotFoundException(404, "DataAccess",
"Reservation with such id not found");
   }
   return entity;
   }
   public async Task<Reservation> AddReservationAsync(Reservation
entity)
   {
    SetDefaultValues(entity);
   await _context.Reservations.AddAsync(entity); await
_context.SaveChangesAsync();
   return entity;
   }
   public async Task<Reservation>
UpdateReservationAsync(Reservation entity) {
   var oldEntity = await _context.Reservations
   .Include(x => x.OperationType)
   .Include(x => x.Status)
   .Include(x => x.Contact)
   .Include(x => x.Rate)
   .ThenInclude(r => r.Currency)
   .Where(x => x.Id == entity.Id && x.IsActive == true)
   .FirstOrDefaultAsync(); if (oldEntity == null) {
   throw new RecordNotFoundException(404, "DataLayer", "Reservation
```

```
   with such id not found");
   }
   var operationStatus = await _context.OperationTypes.Where(x =>
x.Id == entity.OperationId).FirstOrDefaultAsync();
   if (operationStatus == null) {
   throw new RecordNotFoundException(404, "DataLayer", "Reservation
with such id not found");
   }
   entity.OperationType = operationStatus;
   var status = await _context.ReservationStatuses.Where(x => x.Id
== entity.StatusId).FirstOrDefaultAsync();
   if (status == null) {
   throw new RecordNotFoundException(404, "DataLayer", "Reservation
with such id not found");
   }
   entity.Status = status;
   var contact = await _context.Contacts.Where(x => x.Id ==
entity.ContactId).FirstOrDefaultAsync();
   if (contact == null) {
   throw new RecordNotFoundException(404, "DataLayer", "Reservation
with such id not found");
   }
   entity.Contact = contact;
   var rate = await _context.Rates.Where(x => x.Id ==
entity.RateId).FirstOrDefaultAsync();
   if (rate == null) {
   throw new RecordNotFoundException(404, "DataLayer", "Reservation
with such id not found");
   }
   entity.Rate = rate;
   _mapper.Map(entity, oldEntity); await
_context.SaveChangesAsync(); return entity;
```

```
   }
   public async Task<Reservation> DeleteReservationAsync(Guid id) {
var entity = await _context.Reservations
   .Where(x=>x.Id==id && x.IsActive == true)
   .FirstOrDefaultAsync(); if (entity == null) {
   throw new RecordNotFoundException(404, "DataAccess",
"Reservation with such id not found");
   }


   SetDeleteDefaultValues(entity); await
_context.SaveChangesAsync(); return entity;
   }
   }
```

Файл TransactionRepository.cs

```
   public class TransactionRepository : BaseRepository,
ITransactionRepository { private DataAccessContext _context;
   private IMapper _mapper;
   public TransactionRepository(DataAccessContext context, IMapper
mapper)
   {
   _context = context;
   _mapper = mapper;
   }
   public async Task<IEnumerable<Transaction>>
GetTransactionsAsync() { return await
Task.FromResult(_context.Transactions
   .Where(x=>x.IsActive == true)
   .Include(x=>x.Contact)
   .Include(x=>x.RateLog)
   .Include(x=>x.RateLog.Currency)
```

```
   .Include(x=>x.TransactionType)
   .Include(x=>x.OperationType)
   .Include(x=>x.Reservation)
   .AsNoTracking());
   }
   public async Task<Transaction>
CreateTransactionAsync(Transaction entity) {
   SetDefaultValues(entity);
    SetDefaultValues(entity.RateLog); entity.RateLogId =
entity.RateLog.Id;
   await _context.Transactions.AddAsync(entity); await
_context.SaveChangesAsync();
   return entity;
   }
   public async Task<Transaction>
UpdateTransactionAsync(Transaction entity) {
   var oldEntity = await _context.Transactions
   .Where(x=>x.Id == entity.Id && x.IsActive == true)
   .FirstOrDefaultAsync(); if(oldEntity == null) {
   throw new RecordNotFoundException(404, "DataAccess",
"Transaction with such id didn't found");
   }
   _mapper.Map(entity, oldEntity); await
_context.SaveChangesAsync(); return oldEntity;
   }
   public async Task<Transaction> DeleteTransactionAsync(Guid id) {
var entity = await _context.Transactions
   .Where(x=>x.Id == id && x.IsActive == true)
   .FirstOrDefaultAsync(); if (entity == null) {
   throw new RecordNotFoundException(404, "DataAccess",
"Transaction with such id didn't found");
   }
```

```
   SetDeleteDefaultValues(entity); await
_context.SaveChangesAsync(); return entity;
   }
   }
```

Файл FundRepository.cs

```
   public class FundRepository : BaseRepository, IFundRepository {
#region Fields: Private
   private readonly DataAccessContext _context; private readonly
IMapper _mapper;
   #endregion
   #region Constructors: Public
   public FundRepository(DataAccessContext context, IMapper mapper)
{
   _context = context;
   _mapper = mapper;
   }
   #endregion
   #region Methods: Public
   public async Task<IEnumerable<Fund>> GetDeletedFundsAsync() {
    return await Task.FromResult(_context.Funds
   .Include(x => x.Currency)
   .Where(x => x.IsActive == false)
   .AsNoTracking());
   }
   public async Task<IEnumerable<Fund>> GetFundsAsync() { return
await Task.FromResult(_context.Funds
   .Include(x => x.Currency)
   .Where(x => x.IsActive == true)
   .AsNoTracking());
   }
   public async Task<Fund> GetFundAsync(Guid id) { var entity =
```

```
  await _context.Funds
  .Include(x => x.Currency)
  .Where(x=>x.Id == id && x.IsActive == true)
  .FirstOrDefaultAsync(); if (entity == null) {
  throw new RecordNotFoundException(404, "DataAccess", "Fund with
such id not found");
  }
  return entity;
  }
  public async Task<Fund> GetFundByCurrencyIdAsync(Guid
currencyId) { var entity = await _context.Funds
  .Include(x => x.Currency)
  .Where(x => x.CurrencyId == currencyId && x.IsActive ==
  .FirstOrDefaultAsync(); if (entity == null) {
   throw new RecordNotFoundException(404, "DataAccess", "Fund with
such currency id not found");
  }
  return entity;}
  public async Task<Fund> AddFundAsync(Fund entity) {
SetDefaultValues(entity);
  await _context.Funds.AddAsync(entity); await
_context.SaveChangesAsync(); return entity;
  }
  public async Task<Fund> UpdateFundByCurrencyIdAsync(Guid
currencyId, decimal amount) {
  var entity = await _context.Funds
  .Where(x => x.CurrencyId == currencyId && x.IsActive ==
   true)
   .FirstOrDefaultAsync(); if (entity == null) {
  throw new RecordNotFoundException(404, "DataAccess",
   "Fund with such currency id not found in database, so entity
didn't updated");
```

```
  }
  entity.ModifiedOn = DateTime.UtcNow; await
_context.SaveChangesAsync(); return entity;
  }
  public async Task<Fund> UpdateFundAsync(Fund entity) { var
oldEntity = await _context.Funds
  .Include(x=>x.Currency)
  .Where(x=>x.Id == entity.Id && x.IsActive == true)
  .FirstOrDefaultAsync();
   if (oldEntity == null) {
  throw new RecordNotFoundException(404, "DataAccess", "Fund with
such id not found");
  }
  _mapper.Map(entity, oldEntity); await
_context.SaveChangesAsync(); return oldEntity;
  }
  public async Task<Fund> ActivateDeletedFundAsync(Fund entity) {
var oldEntity = await _context.Funds
  .Where(x => x.Id == entity.Id && x.IsActive == false)
  .FirstOrDefaultAsync(); if (oldEntity == null) {
  throw new RecordNotFoundException(404, "DataAccess", "Fund with
such id not found");
  }
  _mapper.Map(entity, oldEntity); await
_context.SaveChangesAsync(); return entity;
  }
  public async Task<Fund> DeleteFundAsync(Guid id) { var entity =
await _context.Funds
  .Where(x=>x.Id == id && x.IsActive == true)
  .FirstOrDefaultAsync(); if (entity == null) {
  throw new RecordNotFoundException(404, "DataAccess", "Fund with
such id not found");
```

```
}
SetDeleteDefaultValues(entity);
await _context.SaveChangesAsync(); return entity;
}
#endregion
}
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: «Розробка програми для організації обліку роботи пункту обміну валюти засобами C# з ASP .NET та Vue.js»

Обсяг пояснювальної записки 73 аркуші.

12 таблиць;

54 рисунки;

1 додаток.

Дата завершення роботи: *12 червня 2025 р.*

Підпис студента-дипломника _____ *Михайлишин Ю. П.*

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Михайлишин Юлія Петрівна

Співавтор:

Назва: 2025_ МихайлишинЮ.П.ІІТ_КСМ_КІ-21-2

Науковий керівник: Кропивницька Віталія Богданівна

Підрозділ: Каф. КСМ

Коефіцієнт подібності 1:0.8%

Мікропробіли: 4

Заміна букв: 0

Інтервали: 0

Білі знаки: 37

Дата створення звіту: 2025-06-22 08:54:50.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-06-22 Надія Ширмовська

Дата експерт