

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему РОЗРОБКА КРИПТОВАЛЮТНОЇ БІРЖІ
З ДЕМО-РАХУНКОМ
Development of a Crypto Exchange with
a Demo Account

Виконав: студент(ка) 4 курсу, групи КНТ-139

Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)
забезпечення

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ЗУЄВ Б.В.

(ПРИЗВИЩЕ та ініціали)

Керівник СЕРДЮК С.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЗУСВА Богдана Віталійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка криптовалютної біржі з демо рахунком.
Development of a Crypto Exchange with a Demo Account

керівник проєкту (роботи) к.т.н., доцент Університету штату Пенсільванія
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

СЕРДЮК Сергій Микитович,
затверджені наказом закладу вищої освіти від “04” квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 07 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка структури програмного
забезпечення. 3. Основні рішення щодо реалізації компонентів системи.

4. Експлуатація та тестування програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СЕРДЮК С.М., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю. ст. викладач		

7. Дата видачі завдання «04» лютого 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Розробка архітектури програми	2 тиждень	Розділ 2
4	Розробка програми.	3-4 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	5 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
7	Нормоконтроль та рецензування.	7 тиждень	
8	Захист роботи.	8 тиждень	

Студент(ка)

_____ Богдан ЗУСВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Сергій СЕРДЮК
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
110 с., 40 рис., 6 табл., 3 дод., 30 джерел.

ТРЕЙДИНГ, КРИПТОВАЛЮТА, ДІАГРАМИ, QT, СЕРВІС ОБМІНУ
КРИПТОВАЛЮТ, КРИПТОГРАФІЧНА БІБЛІОТЕКА, ОПЕРАЦІЙНА
СИСТЕМА, WINDOWS, БАЗА ДАНИХ.

Об'єктом дослідження є процес розробки та впровадження
криптовалютної біржі з демо рахунком.

Предметом дослідження є функціональні можливості криптовалютною
біржі з віртуальним рахунком та методи взаємодії з користувачами.

Метою роботи є розробка комп'ютерного застосунку, що надає
можливості відпрацьовування торгових стратегій, перегляду успішності
власних рішень та аналізу ситуації на криптовалютному ринку в режимі
«онлайн».

Матеріали, методи та технічні засоби: об'єктно-орієнтоване
програмування, мова програмування C++, середовище розробки Qt Creator,
фреймворк Graphics View Framework для інтеграції діаграм із інтерфейсами
користувача, криптографічна бібліотека OpenSSL, аналізатор об'єктного коду
Dependency Walker, персональний комп'ютер з процесором з частотою 1 ГГц та
вище під управлінням операційної системи Microsoft Windows 7.

Висновки. Розроблено трейдингову платформу з віртуальним рахунком,
що дозволяє формувати базові навички взаємодії з криптовалютою,
спостереженні за рухом та аналізу зміни ціни активів.

Галузь використання – інформаційні технології.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 110 pages, 40 figures, 6 tables, 3 appendixes, 30 sources.

TRADING, CRYPTOCURRENCY, CHARTS, QT, CRYPTO EXCHANGES, CRYPTOGRAPHIC LIBRARY, OPERATING SYSTEM, WINDOWS, DATABASE.

The object of the study is the process of development and implementation of a cryptocurrency exchange with a demo account.

The subject of the study is the functionality of a cryptocurrency exchange with a virtual account and methods of interaction with users.

The purpose of the work is to develop of a computer application that provides opportunities to work out trading strategies, review the success of one's own decisions and analyze the situation on the cryptocurrency market in "online" mode.

Materials, methods and technical means: object-oriented programming, programming language C++, Qt Creator development environment, Graphics View Framework for integrating diagrams with user interfaces, OpenSSL cryptography library, object code analyzer Dependency Walker, personal computer with a processor 1 GHz or higher, running the Microsoft Windows 7 operating system.

Results. A trading platform with a virtual account has been developed, which allows you to form basic skills of interacting with cryptocurrency, monitoring movement and analyzing asset price changes.

The field of use is Information Technology.

ЗМІСТ

Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз предметної області	10
1.1 Аналіз проблеми та постановка завдання дослідження	10
1.2 Аналіз програмних аналогів	12
1.3 Висновки за розділом 1	18
2 Розробка структури програмного забезпечення	19
2.1 Визначення функціональних вимог до системи	19
2.2 Вибір API для отримування даних в реальному часі	19
2.2.1 Вибір сервісу, що надаватиме дані в реальному часі	20
2.2.2 Огляд ключових запитів до API	21
2.3 Обґрунтування вибору мови програмування	25
2.4 Вибір середовища розробки	27
2.5 Проектування архітектури програмного забезпечення	27
2.6 Концептуальна модель бази даних	30
2.7 Висновки за розділом 2	31
3 Основні рішення щодо реалізації компонентів системи	32
3.1 Розробка алгоритму програми	32
3.2 Розробка класів програмного забезпечення	35
3.3 Розробка графічного інтерфейсу користувача	42
3.4 Особливості реалізації застосунку	50
3.5 Висновки за розділом 3	52
4 Експлуатація та тестування програми	54
4.1 Призначення та умови застосування програми	54
4.2 Інструкція по експлуатації програмного забезпечення	54
4.2.1 Звернення до програми	55
4.2.2 Вхідні й вихідні дані	55

4.2.3	Повідомлення системи.....	56
4.3	Тестування програмного забезпечення	57
4.3.1	Функціональне тестування програми.....	57
4.3.2	Нефункціональне тестування програми	60
4.4	Висновки за розділом 4	63
	Висновки	64
	Перелік джерел посилання	65
	Додаток А	68
	Додаток Б.....	72
	Додаток В	103

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – інтерфейс прикладного програмування, набір чітко визначених методів для взаємодії різних компонентів;

DLL – динамічно приєднувана бібліотека;

HTTP — протокол передачі даних, що використовується в комп'ютерних мережах. Назва скорочена від «HyperText Transfer Protocol», протокол передачі гіпертекстових документів;

SSL – криптографічний протокол, що забезпечує встановлення безпечного з'єднання між клієнтом та сервером;

БД – база даних;

Індикатор – функція, побудована на значеннях статистики торгових показників, такі як ціна, об'ємні торги;

Криптовалюта - штучна платіжна система, що прирівнюється до реальних грошей і має офіційний курс;

Мбіт – швидкість передачі даних в комп'ютерній мережі;

Месенджер – система обміну миттєвими повідомленнями;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

СКБД – система керування базою даних;

Ринкова тенденція або тренд – тенденція фінансових ринків рухатися в певному напрямку з часом;

ТЗ – технічне завдання;

Трейдинг – діяльність, спрямована на одержання прибутку від різниці в курсі, шляхом купівлі та продажу валют, акцій і інших цінних паперів;

Цифровий актив – цифрове вираження вартості інформаційного ресурсу, який може використовуватися для платіжних або інвестиційних цілей.

ВСТУП

На сьогоднішній день криптовалюта є досить розповсюдженим способом зберігання, накопичування та передавання коштів. За даними досліджень компанії «Tripple A», відсоток українців, що володіють цифровими активами у 2022 році, складає 15.72%, приблизно 6.5 мільйонів людей [1].

Популярний сайт «Cointelegraph» на початку лютого 2022 року, опублікував вартість найдорожчих тематик, рекламу яких постійно бачить більшість користувачів месенджеру Telegram [2]. Найдорожчою тематикою для продажу реклами визнано «Криптовалюта», далі «Трейдинг» та «Інвестиції».

Наведені дані відображають популярність тематики криптовалютної торгівлі, а це означає, що попит на різноманітні продукти та інструменти взаємодії з крипто монетами лише зростає.

З 17 лютого 2022 року, Президентом України Володимиром Зеленським було підписано закон «Про віртуальні активи» [3]. Таким чином, криптовалюта стала узаконеним способом зберігання коштів для всіх фізичних осіб.

Станом на лютий, 2021 року, голова відомого сервісу обміну криптовалют «Binance», Гліб Костарєв повідомив, що більша частина відвідувачів платформи лише купує цифрові активи, ніяк не взаємодіючи з ними, середній вклад на кожен акаунт має на рахунку не більше однієї тисячі доларів [4]. Так як кількість бажаючих розібратися в новому напрямку інвестицій постійно зростає, проблема опанування нових інструментів зберігання та накопичування грошових активів є актуальною.

Мета даної кваліфікаційної роботи бакалавра полягає в розробці програмного забезпечення, що допоможе сформувати базові навички взаємодії з криптовалютою звичайного користувача, спостереженні за рухом та аналізу зміни ціни активів.

Результатом використання розроблюваного застосунку є практичні навички та стратегії торгування тими чи іншими активами кінцевого користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз проблеми та постановка завдання дослідження

Криптовалюта є одним з найбільших трендів в сучасному світі фінансів. Все більше людей зацікавлюються в можливостях заробітку на криптовалюті, що призводить до зростання попиту на криптовалютні біржі. Криптовалютні біржі є основними майданчиками для купівлі та продажу цифрових монет. Усі учасники криптовалютної біржі вважаються трейдерами.

Трейдер – учасник біржової торгівлі, що здійснює торгово-інвестиційні операції з цінними паперами (облігаціями, акціями або іншими фінансовими інструментами) з метою отримання прибутку. Вся робота полягає в покупці деяких віртуальних активів за однією ціною, та перепродаж їх за іншою, більш високою [5].

Як правило, всіх трейдерів поділяють за їх стратегіями:

- денні трейдери – проводять щонайменше однієї транзакції щодня;
- позиційні трейдери – декілька транзакцій протягом місяця;
- середньостроковий трейдер – декілька транзакцій щороку;
- інвестор – 1 транзакція в півроку, а то й декілька років.

Безліч нових клієнтів сучасних бірж прагнуть спробувати себе в ролі денного трейдера, створити власну прибуткову стратегію, навчитися розпізнавати сигнали ринку та робити переважно прибуткові угоди.

Вид стратегії торгівлі трейдер вибирає сам індивідуально, залежно від складу власного характеру, вільного часу тощо, може використовувати будь-який метод торгівлі, механічну торгову систему, технічний аналіз, фундаментальний, інтуїтивний. Однак стратегії, які не орієнтовані на довгострокову перспективу, вважаються негативними як особиста поведінка, так і тому, що вони створюють потенційну шкоду для реальної економіки [5].

Ринок криптовалют є досить волатильним, це означає, що за дуже короткий проміжок часу будь-яка віртуальна монета може показати значні відхилення в ціні як вгору так і вниз. Весь ринок криптовалют і всі цифрові

активи позбавлені будь-якого зв'язного та універсального регулювання і класифікації. При цьому, кожна монета піддається спекулятивним діям, що стають причиною необґрунтованих рухів того чи іншого цифрового активу. Головні гравці ринку, що володіють значними сумами, можуть сприяти штучному росту або падінню вартості монети в найменші періоди часу. Стрімка зміна вартості цифрової монети впливає на рішення як початківців так і досвідчених гравців бірж, тому існують певні правила та рекомендації при аналізі графіків, індикаторів та трендів [6]. Різні дослідження відображають однаково невтішні дані – більше 90% професійних гравців на біржах втрачають свої гроші повністю або частково впродовж декількох років [7]-[9]. Потенційні користувачі, які не мають достатнього досвіду у торгівлі цифровими активами, скоріш за все, поступово втрачатимуть всі свої інвестиції через недостатню аналітику ринку, неправильні торгові стратегії та недостатнє розуміння ризиків. В такому разі необхідність створення криптобіржі з демо рахунком виникає з декількох причин.

Перша причина полягає в тому, що ринок криптовалют є досить ризикованим та волатильним. Клієнти сервісів обміну криптовалют можуть відчувати реальні умови торгівлі без ризику втрати реальних коштів, використовуючи демо рахунок з невеликим стартовим капіталом.

Друга причина полягає у відсутності демо рахунків на більшості крипто бірж. Найчастіше, користувач, що вирішує спробувати себе в торгівлі криптовалютою, бачить історії лише успішних людей, яким пощастило розбагатіти на спекуляції цифровими монетами [10]. При цьому, в мережі «Інтернет» не так багато прикладів невдач інвесторів, тому кожен новий потенційний клієнт сучасної біржі може мати хибні очікування та сподівання щодо подальшого розвитку подій в інвестуванні в криптомонети.

Завданням даної роботи є:

- аналіз стану предмету дослідження;
- виявлення способів демонстрації реального стану навичок та розуміння принципу функціонування криптовалютного ринку;

- обґрунтування системи заходів, необхідних для вирішення прикладних завдань;
- розробка настільного застосунку, що враховує результати досліджень на основі існуючих рішень.

1.2 Аналіз програмних аналогів

Перш ніж розпочинати проектування програмного забезпечення, необхідно провести аналіз існуючих аналогів та їх методи вирішення проблем, виділити їх переваги та недоліки, унікальні особливості тощо.

Більшість програмних продуктів, що надають демо рахунки для торгівлі криптовалютою мають спільну проблему – складна форма реєстрації. Це означає, що надані послуги можуть бути використані лише в браузері, користувачу необхідно створювати справжній обліковий запис, зберігати паролі, дбати про безпеку і лише тоді переходити до можливості тренування на віртуальному рахунку. Розроблюваний застосунок має вирішити цю проблему і врахувати її при проектуванні і розробці.

Всі розглянуті аналоги є безкоштовними та пропонують власні рішення, що дозволяють перевіряти власні навички та стратегії управління цифровими активами.

1.2.1. Binance

Один з найвідоміших сервісів обміну криптовалют, що був заснований в 2017 році, і вже зарекомендував себе як надійний, потужний сервіс для керування криптовалютних портфелем, активної торгівлі тощо. Даний програмний продукт за версією сайту «minfin.com» визнано найпопулярнішим та найефективнішим серед інших відомих сервісів обміну криптовалюти в Україні. Об'єм торгів за добу перевищує 10 млрд \$, біржа має в списку 357 монет, які утворюють 1414 валютних пар [11].

Даний сервіс має сучасний інтерфейс, який дозволяє швидко орієнтуватися та реагувати на всі події криптовалютного ринку (рис. 1.1).

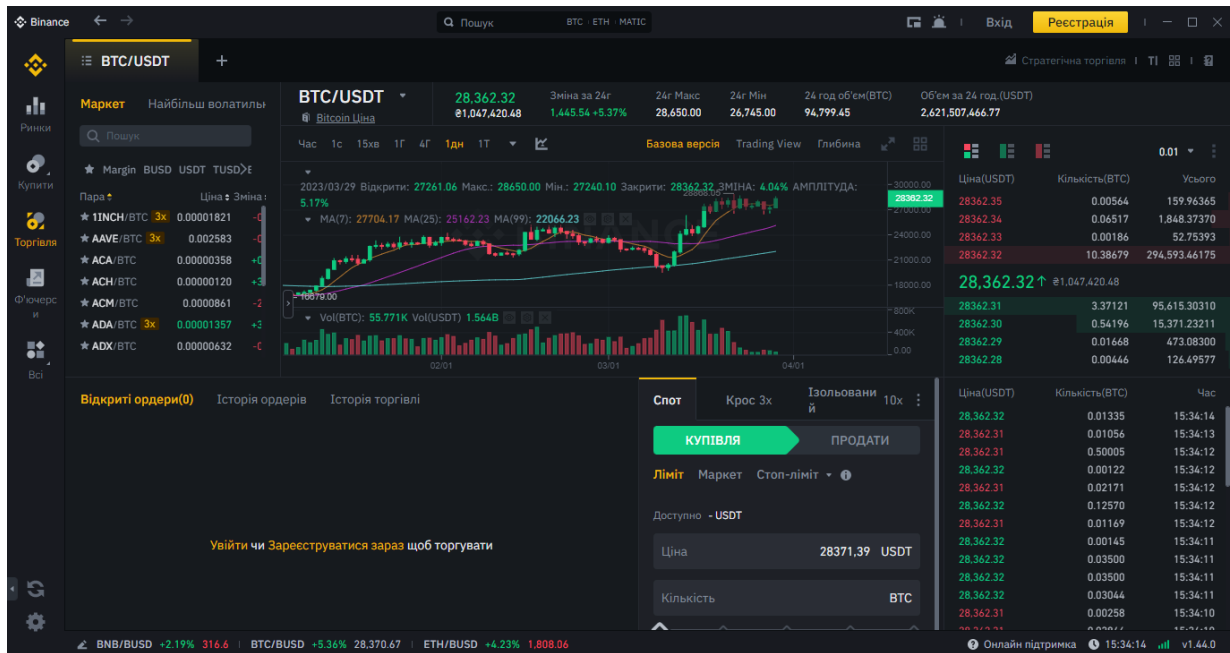


Рисунок 1.1 – Інтерфейс сервісу Binance

Головні переваги даної платформи:

- наявність україномовного інтерфейсу;
- повна кросплатформність (існує версія для ОС Windows, Linux, MacOS, мобільні версії для Android та IOS), а також веб-версія;

- наявність демо рахунку як для початківців, так і професіоналів;

Недоліками платформи, що розглядається можна вважати:

- складна процедура верифікації;
- технічні проблеми з обслуговування сайту можуть перешкоджати роботі з біржею;
- більшість з перекладів користувацького інтерфейсу виконані неякісно, що суттєво знижує рівень зручності використання платформи;

1.2.2 MetaTrader 4

Одним з найсильніших конкурентів у трейдинговому сегменті є програмне забезпечення від онлайн брокеру «ThinkMarkets» (рис. 1.2). MetaTrader 4 повністю вдосконалюється, і оновлюється, що свідчить про потужну підтримку продукту та серйозність компанії щодо його позиції на ринку. При встановленні торгової платформи додатково встановлюється редактор MetaEditor, яким користуються професійні користувачі, розробляючи і доповнюючи власними програмними рішеннями вже готову систему спеціальною мовою MQL4. Торгова система надає можливість створення віртуального рахунку, при цьому, такі функції, як «торгові сигнали», «копіювання угод» та інші залишаються доступними.

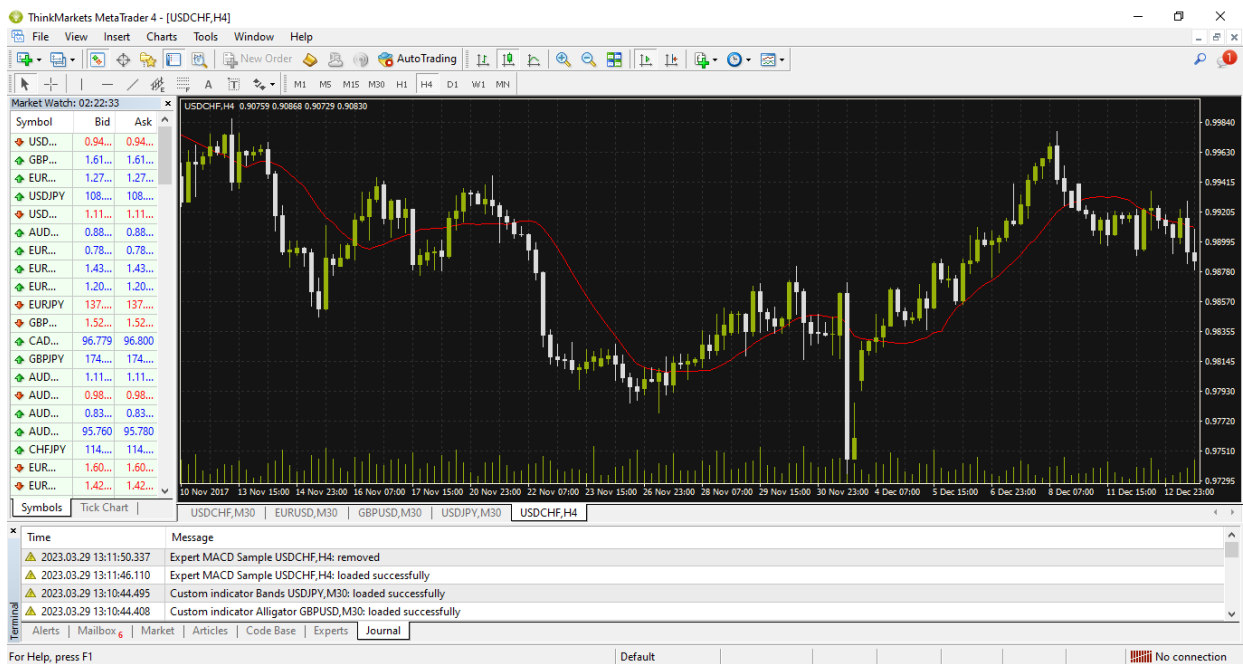


Рисунок 1.2 – Інтерфейс сервісу MetaTrader 4

Розглянемо головні переваги даного програмного продукту:

- більше 30 торгових інструментів;
- 9 варіантів часового інтервалу будування діаграми;
- можливість створювати власні індикатори;

До мінусів можна віднести:

- проблеми з котируванням, ціни на поточні торгові пари можуть не співпадати з ринковими, а інколи суттєво відрізнятись;
- автоматичне закриття позицій торгів по завершенню доби;
- відсутність україномовного інтерфейсу.

NinjaTrader

Досить відома платформа для трейдингу, що дозволяє керувати не лише криптовалютичним портфелем, а й акціями, валютою та іншими цифровими активами. Дана трейдингова біржа має більше 100 безкоштовних технічних індикаторів, що легко налаштовуються та допомагають орієнтуватися в поточній ринковій ситуації. Також, система надає власну мову програмування «NinjaScript», засновану на C# [12]. За допомогою неї можливо створювати власні індикатори та стратегії (алгоритми торгівлі). Даний сервіс має безліч професійних інструментів керування фінансами та взаємодії з ними. Головну форму платформи зображено нижче, на рис. 1.3.

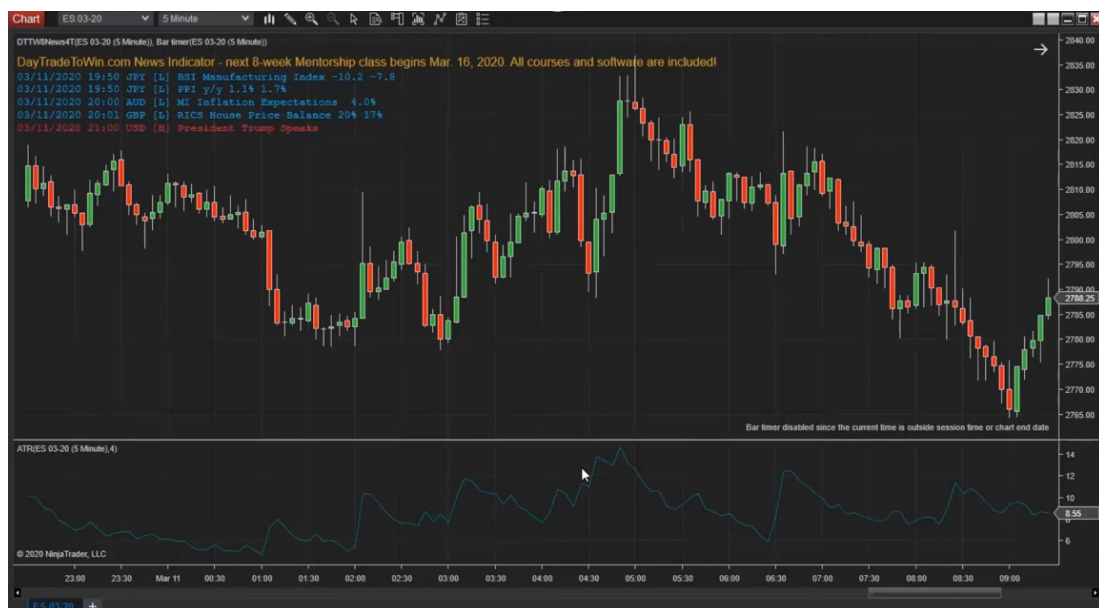


Рисунок 1.3 – Інтерфейс платформи NinjaTrader 8

Серед найважливіших переваг даної платформи:

- підключення до різних брокерів для торгівлі;
- безкоштовне використання більшості функціоналу, в тому числі на демо рахунку достатнього для опанування предметної області;
- можливість редагувати графіки за часовим інтервалом, відображати лінії по закриттю, відкриттю, змінювати свічки за об'ємом, діапазоном.

До мінусів даного програмного продукту відносять:

- постійні технічні проблеми з терміналом у процесі роботи;
- доисть незручний інтерфейс, який важко налаштувати;
- компанія орієнтована на професійних трейдерів, що можуть вносити великі суми та активно використовувати їх (мін. депозит 400\$) .

Після визначення аналогів, необхідно дібрати критерії, які є найголовнішими при порівнянні існуючих рішень.

Перш за все, застосунок має передбачати використання інтерфейсу українською мовою, адже це ключовий фактор забезпечення комфортної та ефективної роботи з програмою.

Важливою умовою використання є простота налаштування, зокрема відсутність обов'язкової реєстрації реального облікового запису, перш ніж стане доступним використання віртуального рахунку. Кількість таймфреймів (торгових періодів) впливає на можливість спостереження за тенденціями криптовалютного ринку, тому даний параметр також буде врахованим при порівнянні аналогів.

Врешті, візуальне представлення результатів торгівлі у вигляді діаграм та таблиць є найголовнішим елементом функціональності демонстраційного облікового запису біржового застосунку.

Порівняльна характеристика розглянутих платформ наведена у табл. 1.1.

Таблиця 1. 1 – Порівняння торгових платформ

Назва	Binance	MetaTrader 4	NinjaTrader 8
1	2	3	4
Інтерфейс (українська)	Так	Ні	Ні
Демо рахунок	Так	Так	Так
Обов’язкова реєстрація	Так	Так	Так
Історія операцій	Так	Так	Так
Безкоштовні котирування	Так	Так	Ні
Кількість торгових періодів	16	9	22
Складні налаштування	Ні	Так	Так
Динамічний графік	Так	Так	Так
Перегляд портфелю	Так	Ні	Ні
Діаграма розподілу криптовалюти	Ні	Ні	Ні
Редактор платформи	Ні	Так	Так

Розглянуті аналоги пропонують складні професійні платформи, кінцеві користувачі яких – досвідчені інвестори, що постійно використовують найновіші інструменти та засоби отримання вигоди.

Хоча застосунки й надають демо рахунки для здобуття навичок торгівля цифровими активами, більшість з них є нелегкі в використанні, включають в себе службові термінали та редактори.

Для використання будь-якої з існуючих програм необхідно зареєструватись та налаштувати складне робоче середовище, що є суттєвим недоліком кожної з них. Найбільш доцільним аналогом є платформа Binance, адже вона має найпростіші налаштування, сучасний інтерфейс, не містить суттєвих недоліків порівняно з існуючими аналогами, а також надає зручний інтерфейс перегляду криптовалютного портфелю.

Отже, програмне рішення має містити базову функціональність, наявну у рішень-аналогів, а також містити власні, унікальні варіанти взаємодії кінцевого користувача з програмою. До програмного забезпечення буде розроблено функціональні вимоги, що й визначатимуть повний набір функціоналу та можливостей.

1.3 Висновки за розділом 1

Проаналізовані причини проблеми відсутності систематизації навичок та вмінь в віртуальних рахунках найпопулярніших трейдингових платформах, таких як «Binance», «MetaTrader 4» та «NinjaTrader». Визначено основні можливості існуючих рішень, політику їх використання, переваги та недоліки.

Виявлено, що більшість популярних трейдингових платформ мають базовий, безкоштовний план використання продукту.

Визначено, що більшість програмних продуктів не мають жодних форм представлення розподілення криптовалютного портфелю, внаслідок цього у цільового користувача можуть виникати проблеми з оцінкою власних результатів, ініціатив та планів. Отже, доцільною є розробка програмного забезпечення трейдингової платформи з демо рахунком.

2 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Визначення функціональних вимог до системи

Огляд і аналіз аналогів програмного забезпечення, описані в розділі 1 сформуvalи наступні функціональні вимоги:

- створення облікового запису;
- вхід в існуючий обліковий запис;
- відображення кругової діаграми розподілення монет у відсотках;
- відображення таблиці з портфоліо криптовалют, їх наявності, зміна вартості за останню добу, загальна вартість кожної з куплених монет;
- відображення графіка зміни вартості кожної монети (за поточною, обраною монетою);
- відображення графіка зміни вартості за годинами, хвилинами;
- можливість прокручувати графік для перегляду попередніх позицій криптовалюти за визначеним інтервалом
- відображення торгових операцій користувача.
- відображення балансу в доларах;
- купівля та продаж обраної монети;

2.2 Вибір API для отримання даних в реальному часі

Так як розроблюване програмне забезпечення має постійно оновлювати й демонструвати реальні дані криптовалютного ринку, необхідно обрати API, що зможе надавати усю необхідну інформацію.

Таке API надається існуючими на ринці сервісами обміну криптовалюти для забезпечення доступу до даних торговельних пар, цін, обсягів торгів та інших показників. Надана інформація має бути розбірливою, чіткою та добре структурованою. Далі буде розглянуто найвідоміші програмні інтерфейси, що можуть бути використані при розробці даної програми.

2.2.1 Вибір сервісу, що надаватиме дані в реальному часі

Обраний сервіс надання даних про стан криптовалютного ринку має легко інтегруватися та використовуватися. Важливою умовою є можливість використовувати API сервісу без унікального ключа, так як проєктована система потребує лише ринкову вартість монет та історію зміни її ціни. Кількісні показники обмеження запитів та наявність найпопулярніших цифрових монет серед даних сервісу є також необхідним при порівнянні. Також, сервіс має надавати зручну документацію, що полегшить процес взаємодії з API.

Порівняємо найвідоміші сервіси обміну криптовалюти, що надають безкоштовне та необмежене у використанні API, можливе для застосування у розробці власних програмних засобів (табл. 2.1).

Таблиця 2.1 – Порівняння API відомих криптобірж

Назва	Kucoin API	Poloniex API
Зручна документація	Так	Так
Безкоштовне використання	Так	Так
Обмеження запитів	500 за 10 сек.	200 за сек.
Графіки зміни ціни	Так	Так
Кількість монет	понад 300	понад 600
Обов'язкове використання API ключа	Ні	Ні

Розглянуті крипто-обмінники надають досить схожі можливості для використання, мають зрозумілу документацію для використання API, а також не потребують використання унікального ключа для доступу до загальнодоступної інформації. При цьому, усі дані, необхідні для відображення є доступними.

Зважаючи на параметри обох бірж, було обрано API від сервісу обміну криптовалютами Poloniex, так як API саме цієї біржі дозволяє надсилати більше запитів за секунду, доступні історичні дані для набагато більшої кількості монет, а також простіший формат JSON відповідей, що надходять у відповідь на запити до біржі.

2.2.2 Огляд ключових запитів до API

Програмне забезпечення потребує дані, що дозволять:

- Оновлювати ціну поточної криптовалюти не рідше 1 разу в секунду;
- Будувати графіки зміни ціни монети в різних періодах ;
- Отримувати дані про зміну вартості у відсотках.

Усім цим вимогам відповідає API обраної біржі, відповідно, буде представлено 3 запити, що використовуватимуться при розробці. Загальна сигнатура запиту: - «<https://api.poloniex.com>».

Розглянемо функції отримання необхідних даних та структуру відповіді:

- «`/markets/{symbol}/Price`»;

Ключ «symbol» - валютна пара, інформація про яку надійшла.

Ключ «price» - ціна криптовалюти відносно іншої криптовалюти.

Ключ «dailyChange» - десятковий дріб, що позначає зміну вартості поточної валютної пари за добу. Щоб перетворити десятковий або звичайний дріб у відсотки, потрібно це число помножити на 100, відповідно, денна зміна відображає зростання вартості монети «Ethereum», як показано на рисунку 2.1, на 1.85% відносно валюти USDT. Даний сервіс надає ціни однієї криптовалюти

відносно іншої, при цьому, сервіс не відображає вартість монет відносно реальних грошових одиниць. При цьому, кожна монета має зазначену вартість, обраховану відносно криптовалюти «Tether» (далі USDT). Кожен USDT на криптовалютному ринці дорівнює одному долару США [13], а отже, вартість тієї чи іншої монети можна обрахувати абсолютно точно.

Ключ «time» зазначає момент часу, у який сервіс обміну оновив дані.

Ключ «ts» вказує на точний час, коли запис був створений на сервері.

Ключі з зазначеним часом висвітлюється за UNIX часом. UNIX час – кількість секунд, які минули від початку епохи до поточного моменту (з першого січня, 1970 року) [14].

Дана функція може приймати один необов'язковий параметр – «symbol», вказується валютна пара у форматі «{SYMBOL}_{SYMBOL2}».

```
{
  "symbol" : "ETH_USDT",
  "price" : "1910.21",
  "time" : 1680701938237,
  "dailyChange" : "0.0185",
  "ts" : 1680701938242
}
```

Рисунок 2.1 - Структура відповіді функції /markets/ETH_USDT/price

- «/markets/{symbol}/markPrice»;

Приклад відповіді на запит з параметром «BTC_USDT» відображено на рисунку 2.2. Структура JSON файлу відповіді:

Ключ «symbol» - валютна пара, інформація про яку надійшла.

Ключ «markPrice» - ціна криптовалюти відносно іншої криптовалюти. При цьому, показник вартості поточної функції оновлюється значно частіше за попередню розглянуту функцію, швидкість оновлення вартості змінюється частіше ніж 1 раз в секунду, що є достатнім для відслідковування зміни ціни.

Ключ «time» зазначає момент часу, у який сервіс обміну оновив дані.

Функція приймає лише один необов'язковий параметр - «symbol».

```
{
  "symbol" : "BTC_USDT",
  "markPrice" : "28341.43",
  "time" : "1680606696169"
}
```

Рисунок 2.2 – Структура відповіді функції /markets/BTC_USDT/markPrice

- «/markets/{symbol}/candles?».

Функція, що повертає дані для побудови графіка руху ціни валютної пари. Відповідь на GET запит, що надсилається, не містить ключів, тому для правильного зчитування структури необхідно стежити за послідовністю розміщених чисел та текстових змінних. Структуру відповіді на даний запит відображено на рисунку 2.3.

```
], [ "1893.81", "1897.06", "1897.01", "1894.76", "1278.45", "0.674267", "741",
"0.390737", 20, 1680713049970, "1896.06", "MINUTE_5", 1680712800000, 1680713099999
], [ "1893.77", "1894.76", "1894.76", "1894.7", "1604.02", "0.846785", "855.04",
"0.451359", 14, 1680713391969, "1894.24", "MINUTE_5", 1680713100000, 1680713399999
], [ "1892.5", "1894.71", "1894.71", "1894.36", "996.22", "0.526118", "551.8",
"0.291405", 21, 1680713638970, "1893.53", "MINUTE_5", 1680713400000, 1680713699999
], [ "1892.04", "1894.25", "1894.25", "1893.82", "3355.08", "1.77253", "1028.51",
"0.543223", 34, 1680713997970, "1892.82", "MINUTE_5", 1680713700000, 1680713999999
], [ "1890.96", "1896.22", "1893", "1895.12", "1396.44", "0.73794", "587.65",
"0.310442", 31, 1680714297969, "1892.35", "MINUTE_5", 1680714000000, 1680714299999
], [ "1895.12", "1896.54", "1895.13", "1896.46", "1470.72", "0.775714", "995.17",
"0.524896", 30, 1680714585970, "1895.96", "MINUTE_5", 1680714300000, 1680714599999
], [ "1894.55", "1896.54", "1896.54", "1895.16", "1059.1", "0.558645", "194.7",
"0.102733", 18, 1680714748970, "1895.83", "MINUTE_5", 1680714600000, 1680714899999
] ]
```

Рисунок 2.3 – Структура відповіді функції /markets/{SYMBOL}/candles

Перші 4 числа містять дані, що формують свічку. Відповідно, це параметри «low», «high», «open», «close». Структура свічки зображена на рисунку 2.4. Сформована свічка, як правило, суттєво відрізняється від інших, сусідніх свічок, має інші показники, тому візуально може бути зовсім не схожа на інші елементи графіку. При цьому кожна свічка відображає рух ціни протягом певного таймфрейму (торгового періоду).

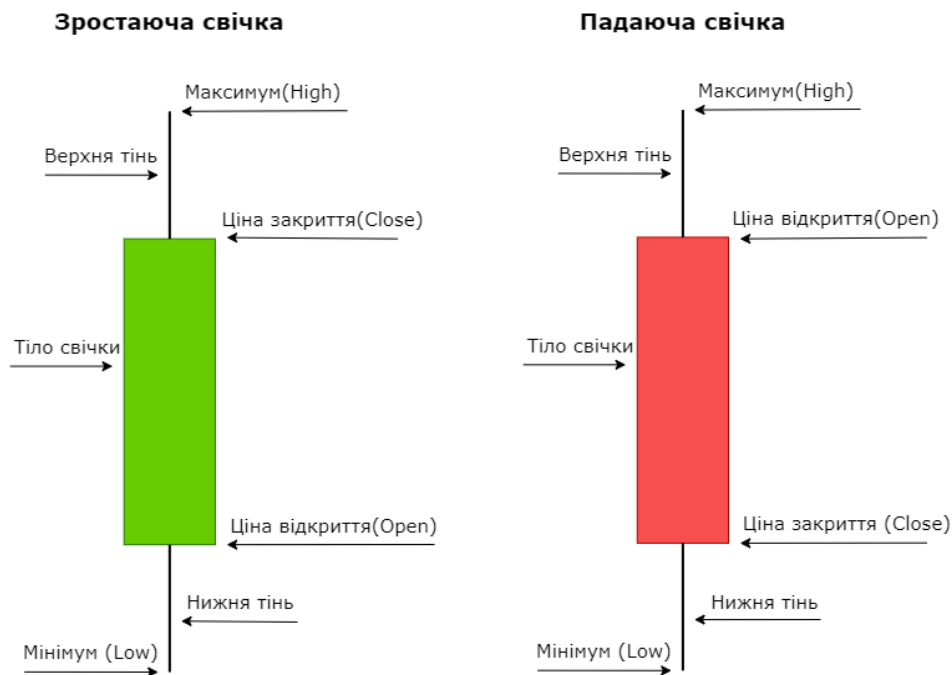


Рисунок 2.4 – Будова японської свічки

Далі йдуть допоміжні характеристики «amount», «buyTakerAmount», «buyTakerQuantity», «tradeCount», що визначають одиниці котирування, що торгуються за різними інтервалами.

Наступний параметр – «ts» - час, коли запис був створений на сервері.

Параметр «weightedAverage» - деяке середньозважене за інтервал.

Наступні дані, типу «рядок» - інтервал свічок, вказаний користувачем.

Останні дані «startTime» та «endTime» - відповідають за фільтр за часом.

Дана функція приймає такі параметри:

- «symbol» - валютна пара формату «{SYMBOL}_{SYMBOL2}»;
- «interval» - інтервал торгових свічок формату «MINUTE_1», «HOURL_2», «DAY_3», «MONTH_1» та інші;
- «limit» - кількість свічок, що буде отримано, за замовчування встановлено 100, максимальна кількість 500;
- «startTime» - початковий час в UNIX системі, за яким визначається початкова точка отримання даних; за замовчуванням 0;
- «endTime» - кінцева точка, за замовчуванням, поточний час.

2.3 Обґрунтування вибору мови програмування

Серед мов програмування, що найчастіше використовуються при розробці Desktop застосунків, виділяють наступні мови: C++, C# та Java. Кожна з мов має власні особливості, що досить сильно впливають на вибір програмних модулів, можливостей при розробці даного програмного забезпечення.

Розглянемо основні параметри, на які необхідно звернути увагу.

Так як проєктований застосунок має забезпечувати швидку обробку даних, динамічне оновлення форм, програма, розроблена за допомогою обраної мови програмування має працювати швидко, без суттєвих затримок в роботі.

Мова реалізації застосунку повинна мати графічні оболонки для швидкого створення користувацького інтерфейсу, що однаково відображатиметься на різних платформах. Такі параметри як типізація та наявність автоматичного збирача сміття є ваговими показниками, так як підхід та практики написання програмного коду дещо відрізнятимуться.

Головним параметром мови програмування є її застосування, типові завдання та проєкти, що розробляються за допомогою того чи іншого інструменту. Результати аналізу мов програмування наведено у табл.2.2.

Таблиця 2.2 – Порівняння мов програмування

Критерії оцінки	Мова програмування		
	Java	C#	C++
1	2	3	4
Графічна оболонка	Swing SWT	WinForms WPF	wxWidgets WinAPI Qt
Типізація	сувора та статична	сувора та статична	слабка та статична

Продовження табл.2.2

1	2	3	4
Кросплатформність	Всі популярні платформи	Можлива, використовуючи платформу .Net5	Можлива завдяки фреймворку Qt, перекомпіляції
Автоматичний збирач сміття	Присутній	Присутній	Ручне керування пам'ятю
Швидкість програми	Повільніша за C++ Швидша за C#	Найповільніша	Найшвидша
Застосування	Веб-сервіси веб та мобільні застосунки	Ігри, веб та настільні застосунки	Системні утиліти настільні ігри драйвери, ОС

Після структурування усіх важливих критеріїв можна зробити висновок, що для розробки програмного забезпечення найкращим вибором буде мова C++. Дана мова застосовується усюди, де необхідно мінімум витрачання пам'яті та максимум продуктивності. Швидкодія розробленого застосунку може відображати найвищі результати при правильному використанні можливостей даної мови програмування та її потенціалу.

Окрім цього, обрана мова програмування має безліч бібліотек та програмних модулів, графічних оболонок, що дозволяють розробити гарний та зручний користувацький інтерфейс. Наступним етапом є вибір середовища розробки, яке надаватиме весь необхідний інструментарій для створення швидких та якісних рішень.

2.4 Вибір середовища розробки

Так як обраною мовою програмування є мова C++, вибір середовища розробки суттєво спрощується. Так як програмний застосунок повинен мати гарний користувацький інтерфейс, можливість додавати готові модулі для роботи, було обрано інтегроване середовище розробки Qt Creator 6.4.2. Дане середовище підтримує як класичні програми мовами C++ та Python, так і програми, написані з використанням мови QML, JavaScript, а структура і параметри елементів інтерфейсу задаються CSS-блоками [15].

Обране середовище розробки має вбудований редактор форм Qt Designer, довідкову систему Qt Assistant, що пришвидшують розробку програми. Окрім цього, компілятори, що мають використовуватися, легко налаштувати під конкретний проєкт, вказавши шляхи до них, та визначивши, мову, яку необхідно скомпілювати (C/C++). Також, при розробці можливо виконувати відлагодження програми, відстежувати дії системи на різних етапах, розділяти екран для працювання з декількома файлами, налаштовувати збірку під свої потреби.

Дане середовище легко інтегрує нові модулі та класи, для будь-яких потреб, наприклад, модуль «Qt Charts» (для побудови найрізноманітніших графів), або клас «QSoundEffect», призначений для найпростіших маніпуляцій аудіо файлами (як правило, формату «wav»).

Отже, обране середовище розробки надає усі необхідні можливості для створення кросплатформових застосунків, їх підтримання та відлагодження.

2.5 Проєктування архітектури програмного забезпечення

Перш ніж перейти до безпосередньої розробки програмного забезпечення, необхідно описати його архітектуру (рис. 2.5).

Для можливості отримання даних від обраного сервісу, необхідно використовувати розширений протокол передачі даних HTTPS, при цьому важливо встановити SSL сертифікат.

OpenSSL – повноцінна криптографічна бібліотека, що підтримує майже всі алгоритми хешування, шифрування та електронного підпису. Функціонал даної бібліотеки буде встановлено та використано у якості динамічних бібліотек для коректної взаємодії з обраним сервісом. Таким чином, відправляючи GET запити, користувач зможе отримувати дані та наповнювати програму інформацією.

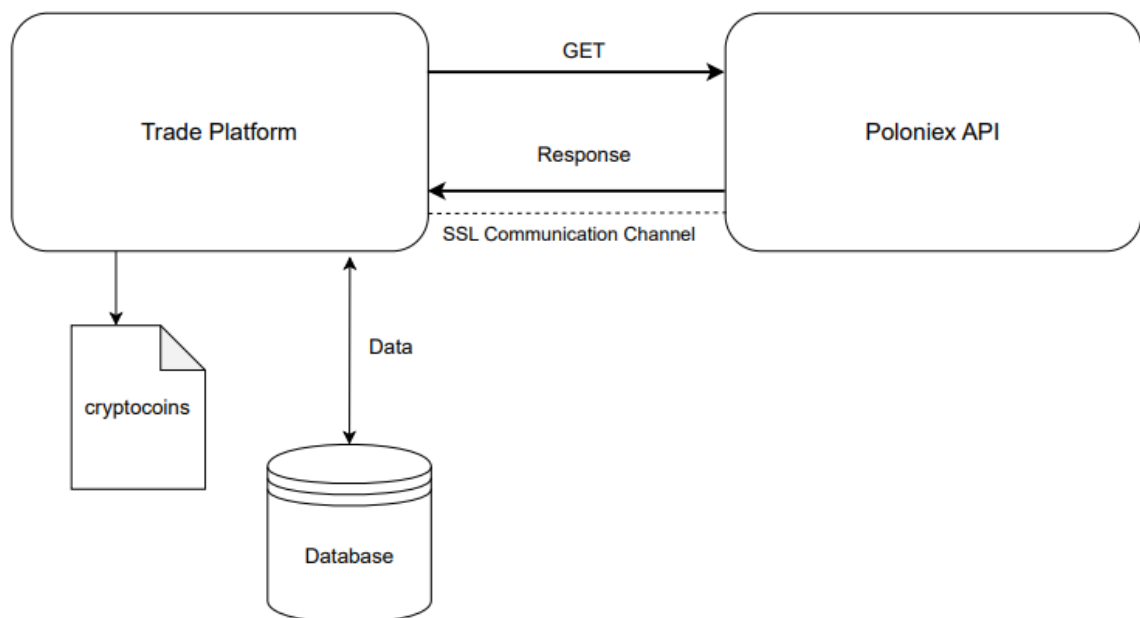


Рисунок 2.5 – Схема архітектури застосунку

Даний застосунок пов’язаний з локальною базою даних та деякими налаштуваннями конкретного користувача. Для доступу до бази даних буде використовуватися вбудована СУБД SQLite, яка підтримується обраним середовищем розробки Qt Creator. Дана база даних зберігається в одному файлі на диску, тому вона є компактною та зручною для локального зберігання на комп’ютері користувача.

Наступним етапом проєктування є вибір патернів(шаблонів) проєктування. Розроблюваний застосунок є настільним, тому загальні правила його побудови не схожі на шаблони, що застосовуються при побудові веб-застосунків. Згідно архітектури роботи, зображеної на рисунку 2.5, найкращими патернами будуть Singleton(Одинак), а також Adapter(Адаптер).

Перший патерн, Singleton, використовуватиметься для побудови класів, що слугуватимуть, як службові, наприклад, клас, що працюватиме з базою даних, або відправлятиме запити до обраного сервісу.

Патерн Adapter якнайкраще підходить для розроблюваного застосунку, так як дані, що надаватимуться, приходитимуть у вигляд JSON файлів, а це означає, що класи, що працюватимуть над відображенням даних, мають отримувати такі об'єкти, інформацію яких зможуть правильно інтерпретувати та застосовувати. В такому випадку, клас-адаптер відповідатиме за перетворення даних з одного об'єкту в такий, оперувати яким зможе інший клас, відповідальний за відображення схем та графіків. Приклад застосування патерну Adapter наведено на рисунку 2.6.

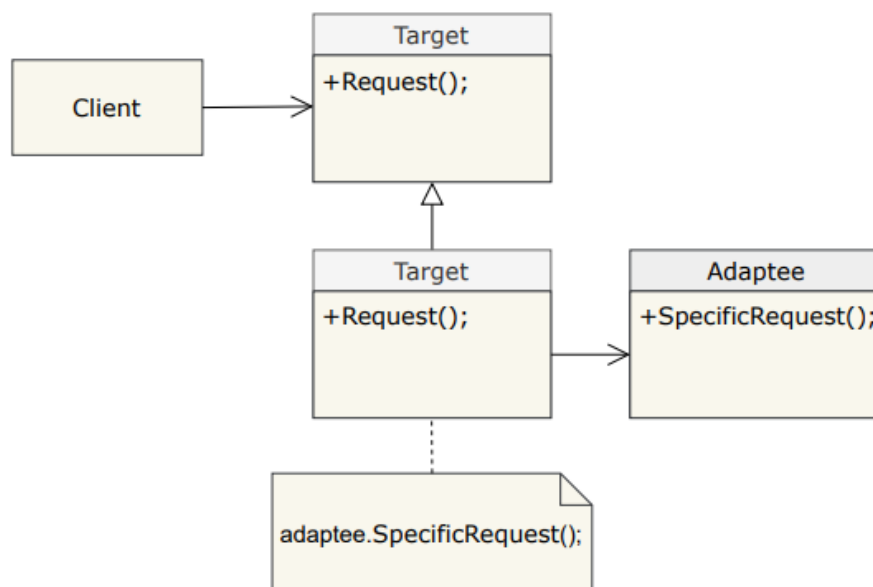


Рисунок 2.6 – Приклад схеми застосування патерну Adapter

2.6 Концептуальна модель бази даних

Функціонал розроблюваного застосунку передбачає збереження даних користувача. Такими даними є імена клієнтів в системі, баланс на рахунку, кількість придбаних монет, сукупність яких утворює портфоліо користувача, перелік здійснених торгових операцій, тощо.

Схему бази даних, необхідну для розроблюваного програмного забезпечення, зображено нижче, на рис. 2.7. Представлена схема бази даних складається з 3 сутностей, пов'язаних двома зв'язками.

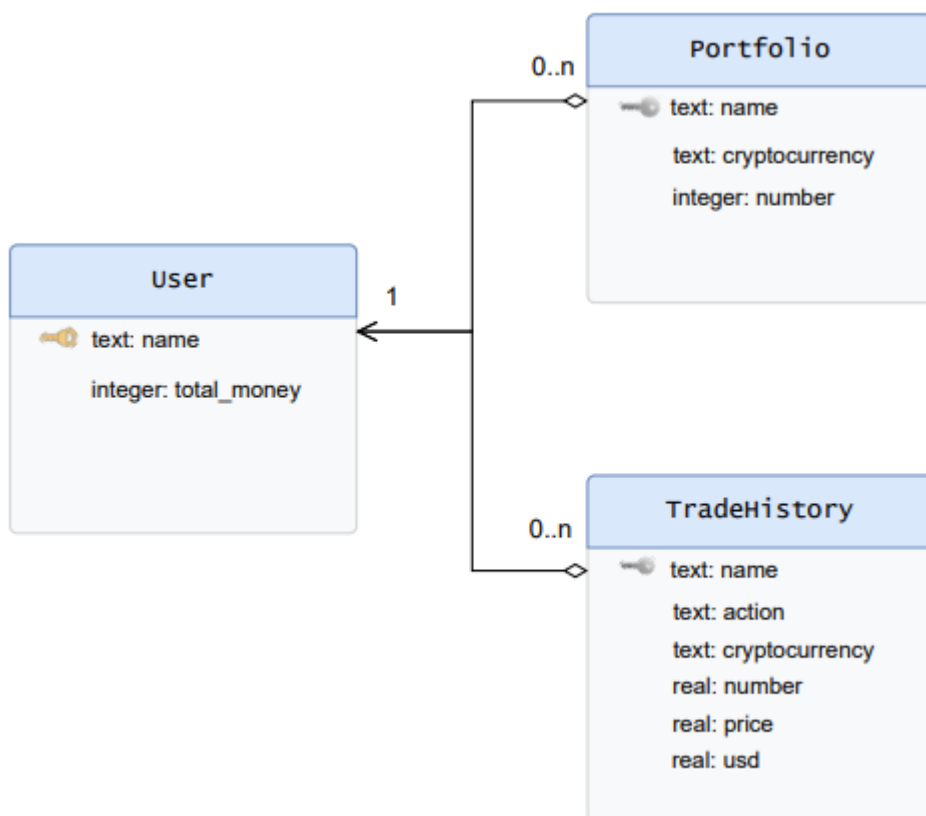


Рисунок 2.7 – Схема бази даних

User – перелік усіх створених користувачів, в якому зберігаються ім'я та кількість грошей в доларах;

Portfolio – таблиця, кожен запис якої містить кількість тієї чи іншої монети конкретного користувача;

TradeHistory – таблиця з усіма торговими операціями користувача, такими як «купівля» та «продаж», кількістю валюти що торгувалася, її ціною та витратами або прибутку в доларах.

Схема бази даних відображає декілька відношень між створеними таблицями. Сутності «User» та «Portfolio» мають відношення «один до багатьох». Це означає, що кожен користувач може мати в портфоліо декілька монет, котрі він придбає. Сутність «User» та «TradeHistory» також мають відношення «один до багатьох», а значить, одному користувачу може відповідати безліч здійснених торгових операцій. Таблиця з історією операцій «купівлі» та «продажу» заповнюється під час торгівлі зі збереженням ім'я активного клієнта в системі.

2.7 Висновки за розділом 2

В даному розділі було визначено функціональні вимоги до проєктованого програмного забезпечення. В якості мови програмування було обрано мову C++, так як саме вона найбільше підходить для створення високонавантажених систем, програм з постійним, динамічним оновленням даних та їх обробкою.

Було обрано сервіс обміну криптовалют Poloniex, що надає безкоштовний набір даних криптовалютного ринку в режимі реального часу. Серед всіх запитів, що можуть надсилати різноманітну інформацію, було розглянуто найголовніші, ключові запити, що дозволять постійно оновлювати дані графіків, будувати діаграми, використовуючи тенденції ринку, виражені в числових показниках.

Було визначено архітектуру системи, шаблони, згідно яких проводитиметься розробка, а також розроблено концептуальну модель бази даних.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Розробка алгоритму програми

Для кращого розуміння процесу взаємодії користувача з системою, було розроблено схему алгоритму програми (рис. 3.1).

При запуску програмного забезпечення, відвідувач може створити один або декілька облікових записів та здійснити вхід в будь-який з них.

При успішній авторизації, клієнт потрапляє на головне вікно програми, де зосереджено весь головний функціонал, що стосується, насамперед, торгівлі цифровими монетами.

При переході до вікна «Портфоліо», користувач має змогу ознайомитися з успішністю торгівлі. При цьому, з даного вікна доступний перехід до кругової діаграми та повернення до головного вікна програми. Даний крок в алгоритмі взаємодії з програмою є найважливішим, так як саме вікно «Портфоліо» може надати інформацію про поточну ситуацію на криптовалютному ринці, котра може бути використана перед безпосередньою торгівлею.

Наступним кроком є «Відкриття кругової діаграми». При переході на дане вікно, користувач оцінює, яку долю складає відсоток тієї чи іншої придбаної криптовалюти. Також, діаграма відображає відсоток доларів, що не був витрачений на жодну криптовалюту, та залишається на залишку користувача.

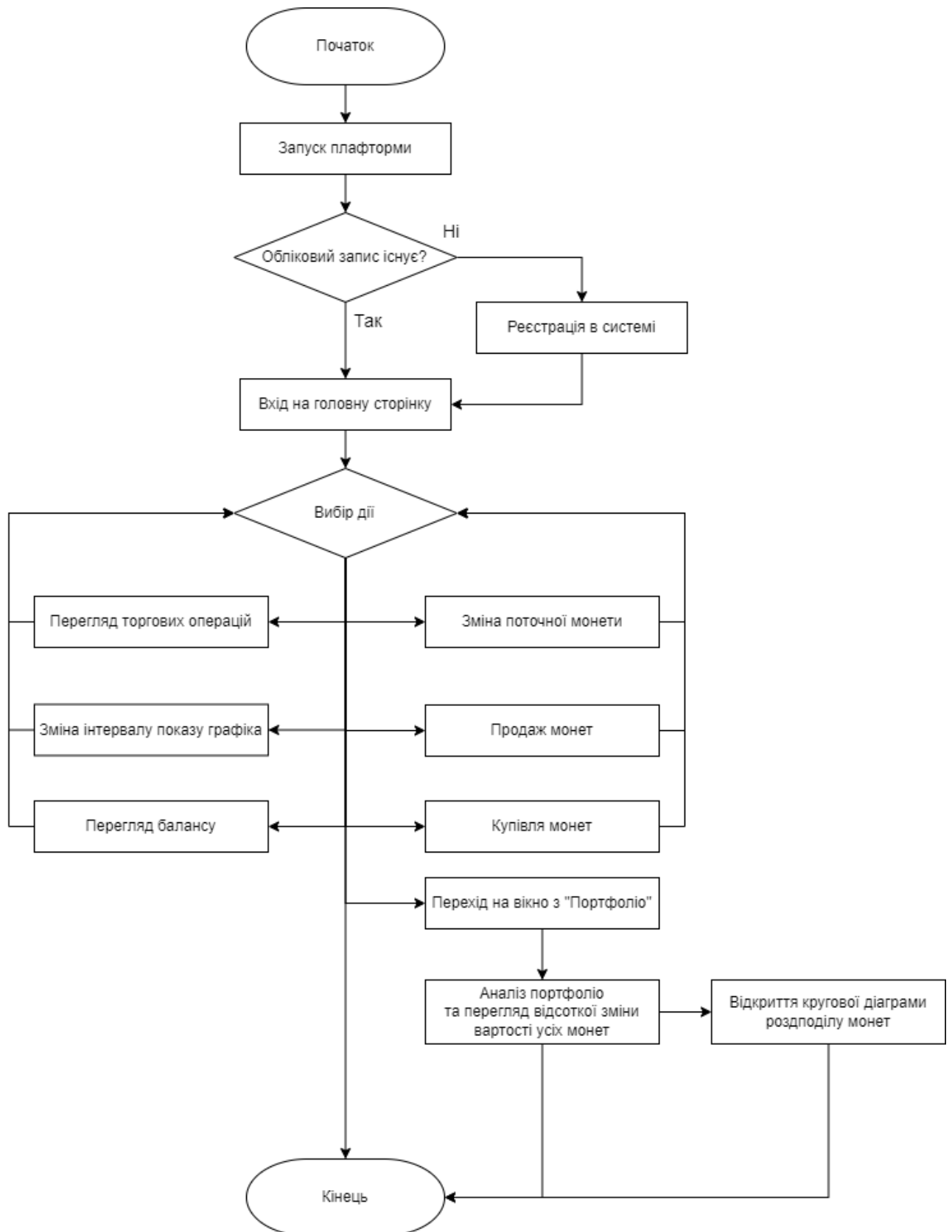


Рисунок 3.1 – Схема алгоритму програми

Також, було розроблено діаграму прецедентів, побудовану у вигляді графа (рис. 3.2). Дана діаграма, на відміну від схеми алгоритму, допомагає краще дослідити саме функціональність програмного забезпечення, вплив дій користувача на систему.

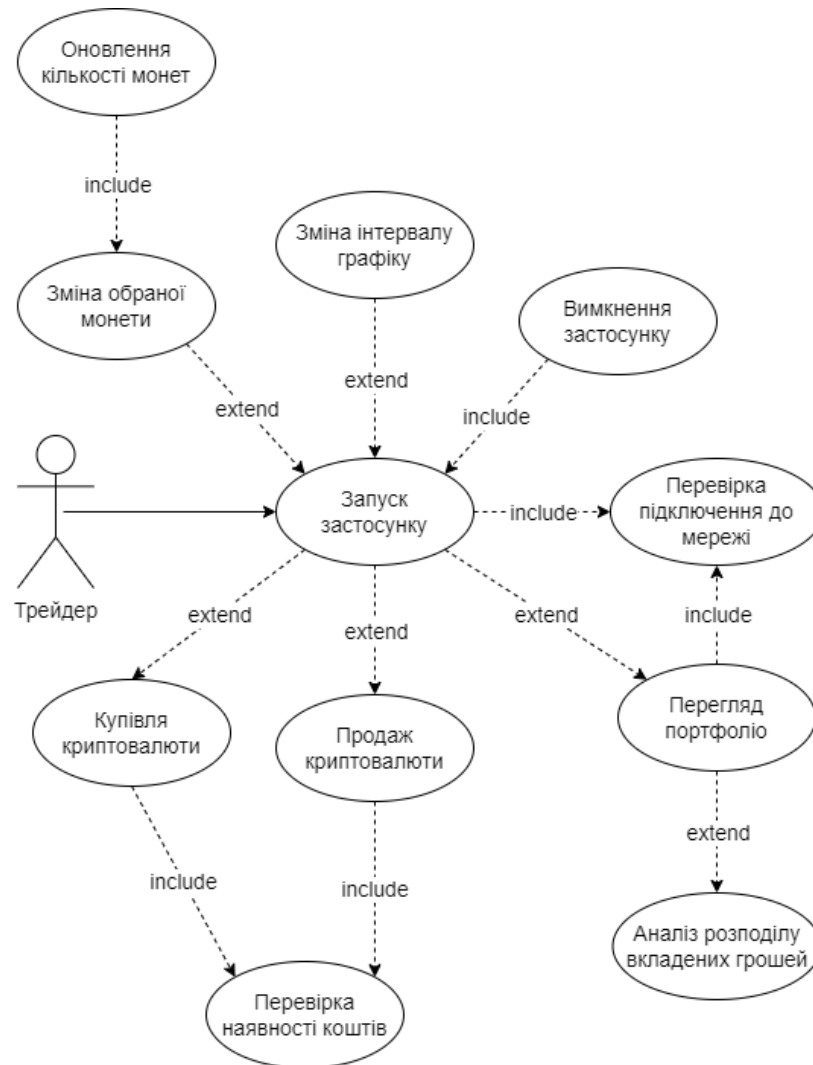


Рисунок 3.2 – Діаграма прецедентів

Ще однією особливістю використання саме діаграми прецедентів є відсутність строгої спрямованості, оскільки актор може взаємодіяти з прецедентами у будь-якому порядку, тоді як блок-схема алгоритму має чітку послідовність дій. Для організації зображених прецедентів були використані відношення включення та розширення.

Деякі з варіантів використання можуть не застосовуватися у процесі здобуття навичок торгівлі цифровими монетами, але залишатимуться доступними для користувача. Серед обов'язкових варіантів використання – «Перевірка підключення до мережі», що є головною умовою коректної роботи програмного забезпечення.

3.2 Розробка класів програмного забезпечення

В ході проектування програмного забезпечення було розроблено діаграму класів, визначено залежності та усі зв'язки між ними за допомогою уніфікованої мови моделювання UML (рис.3.3).

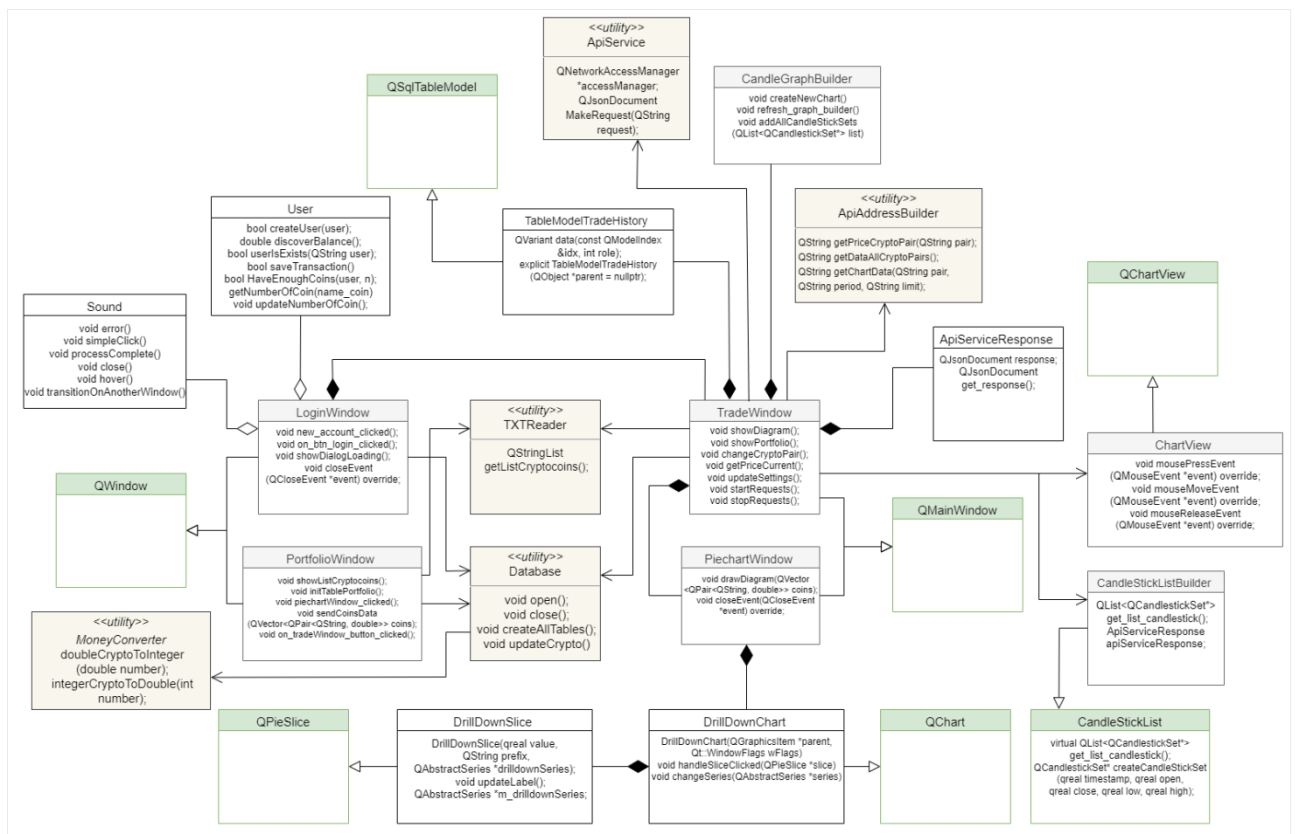


Рисунок 3.3 – Діаграма класів

Дана діаграма містить 6 класів, запропоновані фреймворком Qt, що використовуються у якості базових для нових, користувацьких класів з

розширеними можливостями, перевизначеними методами та важливими полями.

Далі буде розглянуто всі класи від фреймворку Qt, від яких було успадковано інші, користувацькі класи.

QWidget – клас, що є базовим для усіх об'єктів користувацького інтерфейсу. Містить методи та властивості для керування поведінкою віджета, є частиною Qt Core. Успадкований розробленими класами форм PortfolioWindow та LoginWindow;

QMainWindow – клас, що забезпечує головне вікно програми, дозволяє створювати меню, панелі інструментів, статус-бар тощо. Важливою особливістю даного класу є метод setCentralWidget(), що дозволяє розмістити на формі єдиний елемент керування, розміщений по центру, при цьому добре адаптований до розмірів екрану. Успадкований класами TradeWindow та PieChartWindow.

QPieSlice. Даний клас не використовується у якості самостійного, він представляє дані у вигляді сегменту кругової діаграми. Клас є частиною модуля Qt Charts та використовується для створення кругових діаграм. Є базовим класом для класу DrillDownSlice.

QChart – клас, що дозволяє налаштовувати та відображати графіки, діаграми та інші типи візуальних даних. Серед особливостей - гнучка архітектура, що надає можливість розробити діаграму довільного типу, наприклад «лінійну», «кругову», «стовпчасту» та інші. Клас керує графічним представленням серії, легенд і осей, дозволяє створювати власні серії даних, зміни кольорів та стилів, налаштування вісей координат та масштабування графіків. Успадковується класом DrillDownChart.

QChartView представляє собою віджет, який відображає об'єкти класу QChart. Забезпечує відображення та інтерактивність діаграм, наприклад, збільшення масштабу, перетягування графіку для зміни його положення, тощо. Є успадкованим класом ChartView для реалізації методів «перетягування» меж графіка, створюючи анімацію його руху.

QSqlTableModel є моделлю для роботи з таблицями бази даних, яка дозволяє зчитувати та змінювати дані в таблицях, використовуючи SQL-запити. Успадкований класом TableModelTradeHistory для перевизначення налаштувань представлення записів, наданих базою даних, зокрема зміна кольору фону записів в залежності від виду торгової операції. Надає простий інтерфейс для отримання доступу до даних з бази даних, можливе застосування фільтрів, сортування даних, а також можливість редагування даних безпосередньо в таблиці. Підтримує автоматичне оновлення даних в представленні при оновленні інформації у базі даних. Механізм архітектури «Model-View», що реалізує клас QSqlTableModel представлено нижче (рис. 3.4).

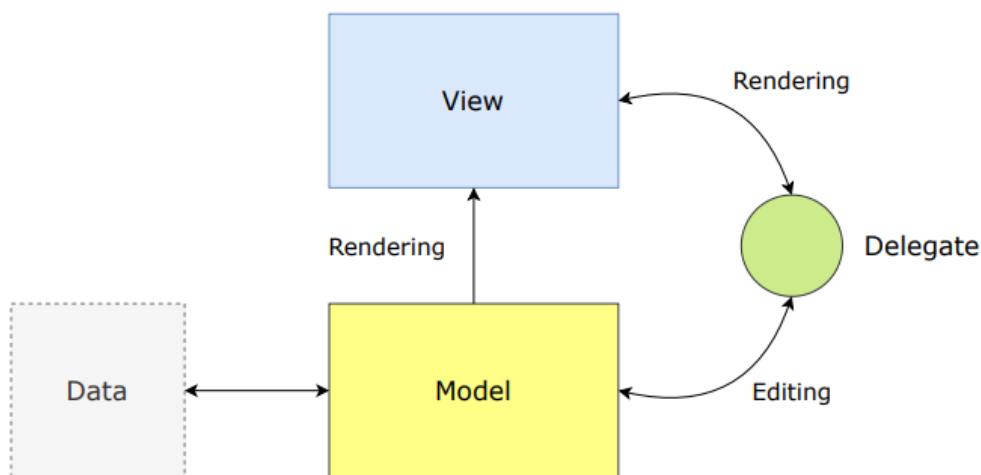


Рисунок 3.4 – Схема архітектури модель-представлення

Моделі відповідають за доступ до даних та надають стандартний інтерфейс для роботи з ними, представлення відображають ці дані у вигляді таблиць, списків та графіків. Делегати використовуються для налаштування відображення та редагування даних в представленні інтерфейсу.

Розглянемо класи, що мають лише статичні поля та методи, тобто не потребують створення екземпляру класу для використання.

Клас `TXTReader` слугує для зчитування усіх назв криптовалюти з файлу «cryptocoins.txt». Єдиний метод класу `getListCryptocoins()` повертає список усіх криптовалют, назви яких буде розміщено на головній формі.

Клас `Database` необхідний для налаштувань з'єднання з базою даних, створення таблиць, якщо такі не існують, оновлення усіх даних користувача, занесені в відповідні таблиці.

Клас `ApiAddressBuilder` відповідає за формування запитів до API обраного сервісу обміну криптовалют.

Клас `ApiService` створює та відправляє запити за допомогою об'єкта класу `QNetworkAccessManager`, при правильному проєктуванні архітектури системи, для всієї програми має бути достатньо лише одного об'єкту `QNetworkAccessManager`. [20]. Для уникнення накопичування неопрацьованих запитів у разі зникнення підключення до мережі Інтернет, було встановлено час очікування на відповідь 3 секунди. Даний клас містить лише один метод, що відправляє запит до API Poloniex. Розглянемо код даного методу:

```
QJsonDocument ApiService::MakeRequest(QString requestStr) {
    QNetworkRequest request((QUrl(requestStr)));
    request.setTransferTimeout(3000);
    request.setHeader(QNetworkRequest::ContentTypeHeader,
        "application/json");
    QNetworkReply *reply = accessManager->get(request);
    while (!reply->isFinished()) {
        QApplication->processEvents();
    }
    QByteArray response_data = reply->readAll();
    reply->deleteLater();
    return QJsonDocument::fromJson(response_data);
}
```

Даний метод отримує сформований класом `ApiAddressBuilder` рядок, що містить запит до API біржі Poloniex, наприклад:

```
https://api.poloniex.com/markets/BTC_USDT/markPrice
```

Відповіддю на даний запит є дані типу JSON, його структура описана в пункті 2.2. Наступним кроком є створення об'єкт класу `QNetworkRequest`, що є представленням запиту. Далі встановлюється максимальне очікування відповіді у мілісекундах, в даному випадку, це 3 секунди. Після цього, встановлюється заголовок та формат даних, який передається в запиті, відповідно, це JSON. Далі, об'єкт `accessManager`, що є статичною змінною класу, що розглядається, викликає метод `get()`, параметром якого є сформований раніше запит, відповідь надісланого запиту надсилається до змінної типу `QNetworkReply`. Після цього, викликається цикл, що продовжує обробляти всі зареєстровані в черзі події застосунка. Після отримання відповіді на запит, метод повертає дані типу `QJsonDocument`, а покажчик на відповідь викликає метод `deleteLater()` для відкладеного видалення об'єкту відповіді для уникнення проблем з доступом до пам'яті. Отримані дані готові до використання.

Клас `MoneyConverter` використовується для конвертування балансу користувача, та кількість кожної з криптовалют в ціле число та назад, в дробове число. Дані дії є одним з способів вирішення «проблемою округлення» [21]. Детальний опис вирішення вищезгаданої проблеми наведено у пункті 3.4.

Наступні реалізовані класи вирізняються способом їх використання, адже їх об'єкти створюються лише один раз та передаються між вікнами за допомогою механізму сигналів та слотів.

`Sound` це клас, що містить набір методів, що реалізують звуковий супровід під час виконання успішних або невдалих операцій в системі.

Клас `User` містить ім'я користувача, що увійшов в систему та методи взаємодії з БД. Має збережений перелік монет та кількість їх в портфоліо для передання у інші вікна, знижуючи при цьому залежність від бази даних.

Наступні описані класи є класами форм.

LoginWindow – Початкове вікно програми, що дозволяє зареєструватися в системі або здійснити вхід в існуючий акаунт, є нащадком класа QWindow.

Клас Portfolio призначене для відображення успішності торгівлі клієнту, показ вкладень в ту чи іншу криптовалюту, відсоткову зміну ціни цифрових монет за останню добу та загальну вартість придбаних активів.

Клас TradeWindow є головним вікном програми, де здійснюються торгові операції, розташовано графік зміни ціни монети, весь головний функціонал зосереджений саме тут.

Клас PieChartWindow є нащадком класу QMainWindow, призначений для перегляду та аналізу вкладень за допомогою кругової діаграми, важливою особливістю є те, що дана діаграма відображає також кількість валюти, що залишилася на рахунку клієнта.

Клас, що дозволяє будувати саму діаграму, містить всі налаштування осей, серій та легенд – CandleGraphBuilder. Дозволяє оновлювати створювати, оновлювати діаграму, додавати, оновлювати та видаляти свічки. Для того, щоб налаштувати звичайний графік, скористаємося кодом:

```
void CandleGraphBuilder::initAxes() {
    axisValue = new QValueAxis;
    axisDate = new QDateTimeAxis;

    chart->addAxis(axisValue, Qt::AlignLeft);
    chart->addAxis(axisDate, Qt::AlignBottom);
    acmeSeries->attachAxis(axisDate);
    acmeSeries->attachAxis(axisValue);

    axisDate->setFormat("dd.MM-hh:mm");
    axisDate->setTitleText("Дата");
    axisValue->setTitleText("Ціна");
}
```

В даному кодi відбувається iнiцiалiзацiя осей графiку, пiсля цього вiсь типу `QValueAxis` розташовується з лiвого боку i представляє вiсь `Y`, вiдповiдно, вiсь `X` є об'єктом класу `QDateTimeAxis`, розташована внизу графiку. Об'єкт `asmeSeries` це серiя даних, що представляє собою перелiк всiх свiчок на графiку.

Далi, для осi, що визначає дату та час, налаштовується формат показу, в даному випадку, кожна подiлка по осi часу вiдображає число, мiсяць, поточну годину та хвилини. Також, для створених осей вставляється надпис «дата» та «цiна» вiдповiдно.

Наступнi класи формують кругову дiаграму розподiлу криптовалюти.

`DrillDownSlice` успадкований вiд класу `QPieSlice`, мiстить всю необхідну реалiзацiю для представлення сегменту дiаграми, включаючи анiмацiю «вiдокремлювання» сегменту.

`DrillDownChart` є пiдкласом класу `QChart` у фреймворку `Qt`, вiдтворює набiр даних у виглядi кругової дiаграми, реалiзуючи при цьому анiмацiю «вiддiлення» сегменту при наведеннi на нього. Будує дiаграму, використовуючи вiдсотковi показники елементiв, є повнiстю адаптивним.

Для роботи з свiчками створено такi класи:

Клас `CandleStickList` є базовим класом, що мiстить вiртуальнi методи, перевизначення яких дозволить формувати та повертати перелiк свiчок, готових для використання в вiдповiдних серiях графiку.

`CandleStickListBuilder` є класом-нащадком вiд `CandleStickList`, дозволяє створювати список готових свiчок, що будуть вставленi в дiаграму з свiчками. Мiстить об'єкт класу `ApiResponse`, що iнiцiалiзується пiд час створення об'єкту поточного класу та використовується для отримання всiх необхідних даних для створення свiчок.

Клас `ChartView` необхідний для реалiзацiї взаємодiї користувача з графiком змiни цiни монети. Даний клас є вiджетом, перевизначення методiв якого дозволяють налаштувати додатковий функцiонал, такий як змiна масштаба, прокручування графiка влiво чи вправо та iншi.

3.3 Розробка графічного інтерфейсу користувача

Для розробки даного застосунку передбачається наявність чотирьох основних форм:

- форма входу та реєстрації, вікно, в якому користувач створює облікові записи та входить у вже існуючі;
- головне вікно програми, де зосереджено весь функціонал, що дозволяє здійснювати торгові операції, переглядати їх історію, спостерігати та аналізувати графік тієї чи іншої монети;
- форма, що містить портфоліо користувача, кількість всіх видів цифрових монет та їх загальна сума за останньою ринковою ціною;
- форма з круговою діаграмою, де відображається відсоткове відношення кількості монет, що є в наявності, а також залишків грошей на балансі відвідувача.

Для можливості взаємодії користувача на обчислювальних машинах з дисплеєм довільного розміру, розроблено адаптивний користувацький інтерфейс, гнучкість якого буде перевірено далі, в пункті 4.3. Розглянемо форму, з якої розпочинається робота програмного застосунку (рис. 3.5).

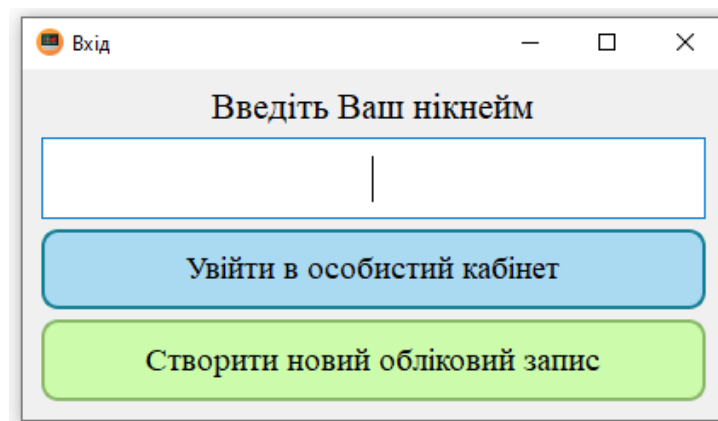


Рисунок 3.5 – Вікно входу в систему

При першому запуску застосунку, користувач має створити обліковий запис та увійти в нього. Користувач не зможе продовжити роботу з програмою, допоки не створить віртуальний рахунок.

Після успішного створення акаунту, користувач отримає повідомлення про реєстрацію в системі (рис.3.6). При цьому, кожна дія клієнта супроводжується звуковими ефектами, що допомагають швидше зрозуміти наслідки свої дій.

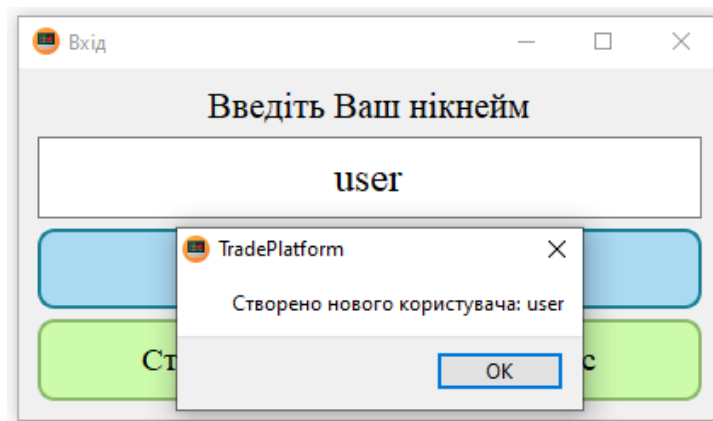


Рисунок 3.6 - Повідомлення про створення облікового запису

У разі неможливості створення облікового запису, якщо такий вже існує, програма сповіщає клієнта, що система вже містить користувача з таким ім'ям (рис.3.7).

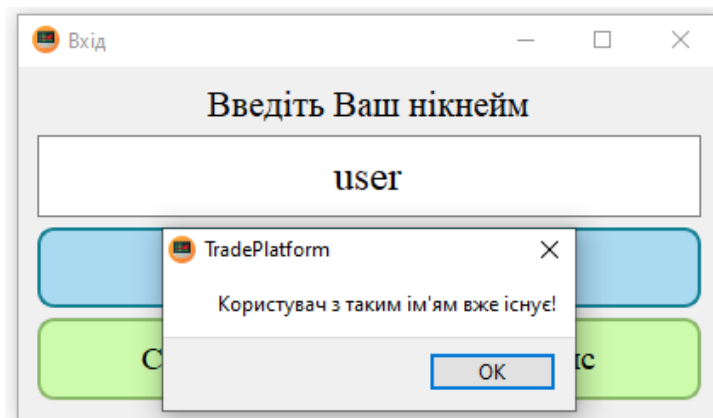


Рисунок 3.7 – Повідомлення про існування акаунту з введеним ім'ям

Після того, як користувач визначився з обліковим записом, при натисканні кнопки «Увійти в особистий кабінет», він потрапляє на головне вікно програми, де зосереджено весь функціонал, що стосується безпосередньої торгівлі цифровими монетами (рис. 3.8).

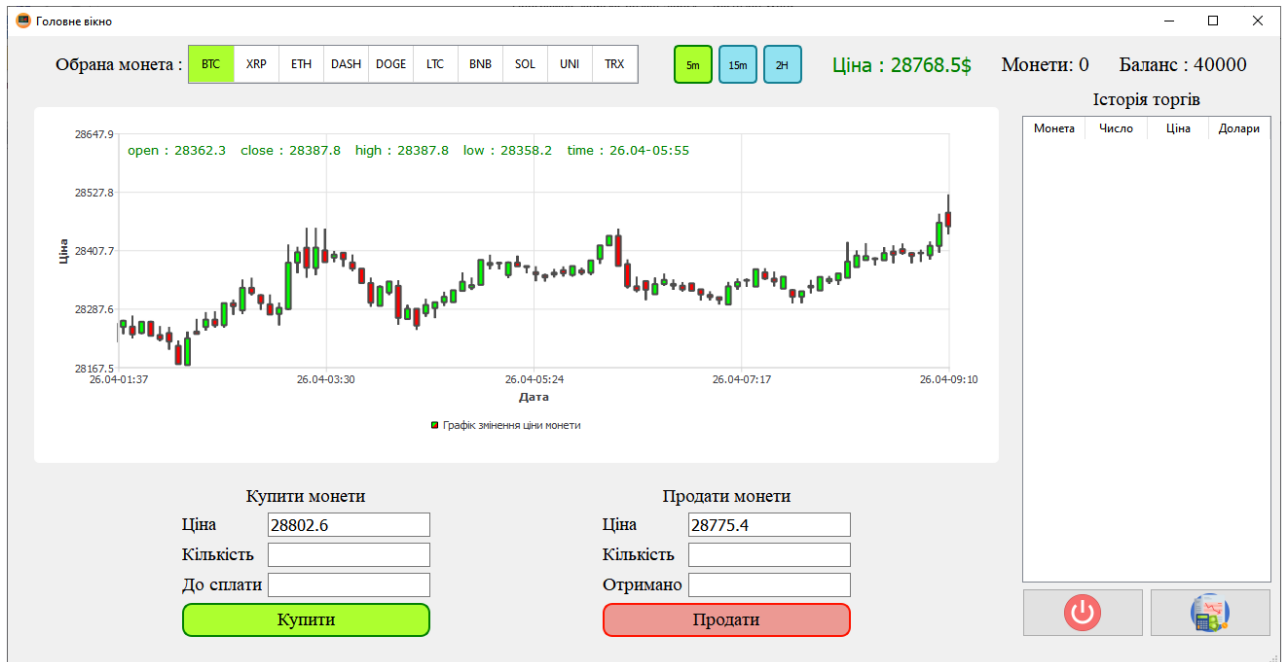


Рисунок 3.8 – Головне вікно програми

Головне вікно програми містить графік змінення ціни поточної обрані монети з певним інтервалом. Побудований графік є екземпляром класу `ChartView`, що просто вставляється у головний елемент типу «`QBoxLayout`», приховуючи при цьому елемент «`QLabel`», в якому розміщено анімацію «Очікування підключення». Користувач може перетягувати графік в будь яку сторону або змінювати його масштаб, переглядаючи більше свічок на інтервальному графіку. Дана можливість реалізована створенням класу, що є нащадком класу `QChartView`, з перевизначеними методами `mousePressEvent`, `wheelEvent` та іншими. Завдяки даним обробникам подій при відслідковуванні подій миші, графік зсувається на потрібну відстань та добре масштабується.

Розглянемо код обробника події `wheelEvent`, що дозволяє змінювати масштаб графіку монети:

```
void ChartView::wheelEvent(QWheelEvent *event) {
    qreal factor = 1;
    if(event->angleDelta().y() > 0) {
        factor = 1.3;
        chart()->zoom(factor);
    }
    else {
        factor = 0.9;
        chart()->zoom(factor);
    }
    event->accept();
    QtCharts::QChartView::wheelEvent(event);
}
```

При прокрутці коліщатка миші перевіряється, в яку сторону було виконано прокрутку, для цього викликається метод `angleDelta()`, що повертає кутові зміни між двома подіями коліщатка миші, при цьому властивість `y()` містить зміни по вертикалі. Тому, якщо метод `y()` повертає значення, що більше 0, миша була прокручена вгору, інакше, вниз. Після цього викликається метод `zoom` з параметром у вигляді коефіцієнту зміни масштабу графіку.

Також головне вікно містить текстові поля, серед яких доступні до редагування лише поля з текстовим надписом зліва «Кількість», тобто, користувач визначає лише кількість криптовалюти для купівлі та продажу, при цьому ціна та поле «До сплати» встановлюється автоматично. Для того, щоб користувач орієнтувався на особливості тієї чи іншої торгової платформи, кожна торгова операція користувача коштує один долар, що вважається комісією торгової платформи.

Також, на формі розміщено елементи QLabel, що містять кількість поточної обраної монети, баланс, а також ціну поточної монети, що оновлюється щосекунди. Дане вікно містить елемент QTableWidgetItem з єдиним рядком, який заповнюється назвами монет, що доступні для взаємодії. Справа знаходяться три віджета QPushButton, що дозволяють перемикаати інтервал побудови свічок на графіку. Також, форма відображає елемент QTableView, що є представленням таблиці TradeHistory з використанням деяких фільтрів, тобто, висвітлює усі торгові операції, здійснені відвідувачем. У правому нижньому кутку розміщено кнопки «Завершення роботи» та «Перехід до портфолію». При зникненні підключення до мережі, система очікує на отримання даних, при цьому відображає анімацію завантаження (рис. 3.9).

Під час очікування на з'єднання, унеможлиблюється купівля чи продаж монет, шляхом блокування відповідних кнопок та поля введення кількості монет для операції.

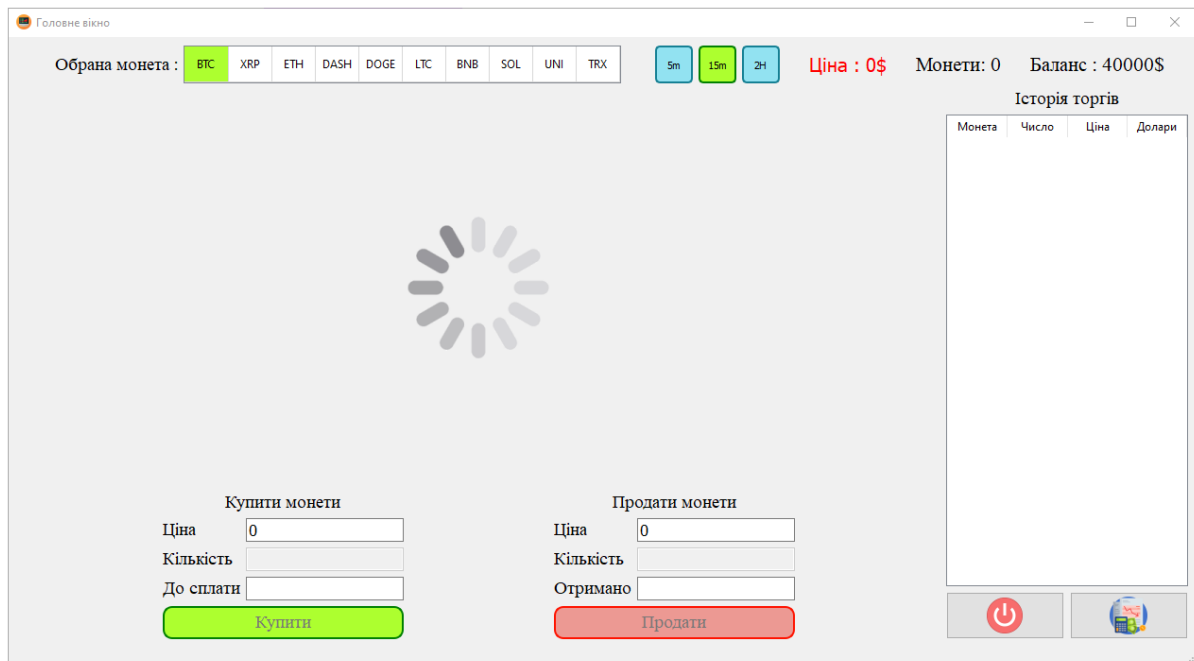
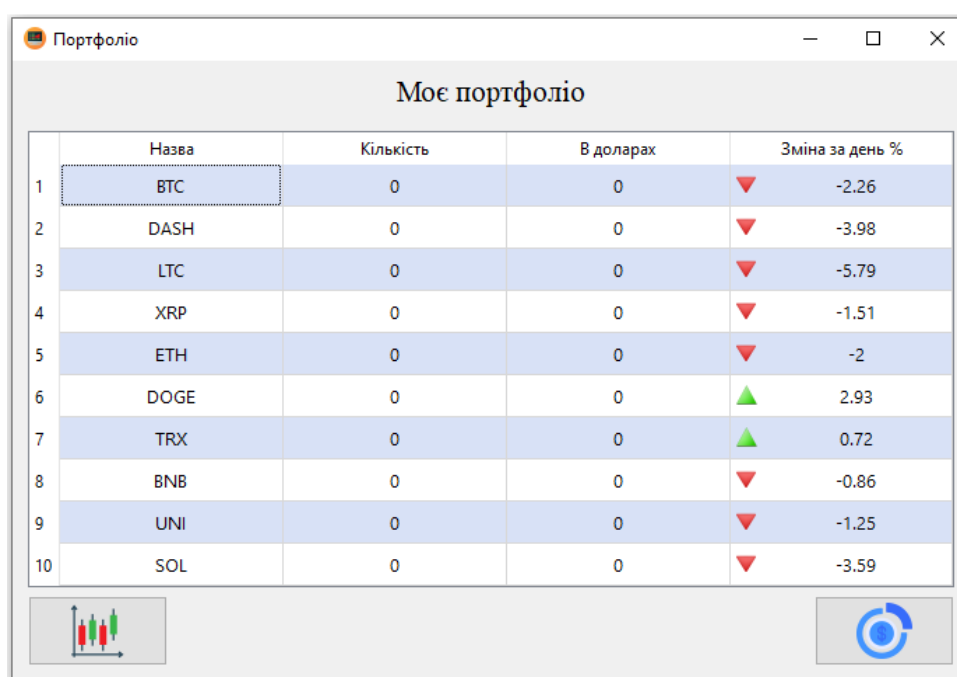


Рисунок 3.9 – Відсутність підключення до мережі «Інтернет»

При переході до вікна «Портфоліо», користувачу відображається форма, що містить елемент QTableWidget, де зосереджено інформацію про портфель користувача у вигляді таблиці (рис. 3.10). Таблиця відображає усю криптовалюту, яку можливо купити, її кількість в наявності, обрахунок загальної вартості придбаних монет та показник зміни за останню добу з відповідним індикатором у вигляді зеленої або червоної стрілки. Розроблена форма містить дві кнопки, в лівому та правому нижніх кутках для переходу на попереднє вікно та наступне – «Кругова діаграма».

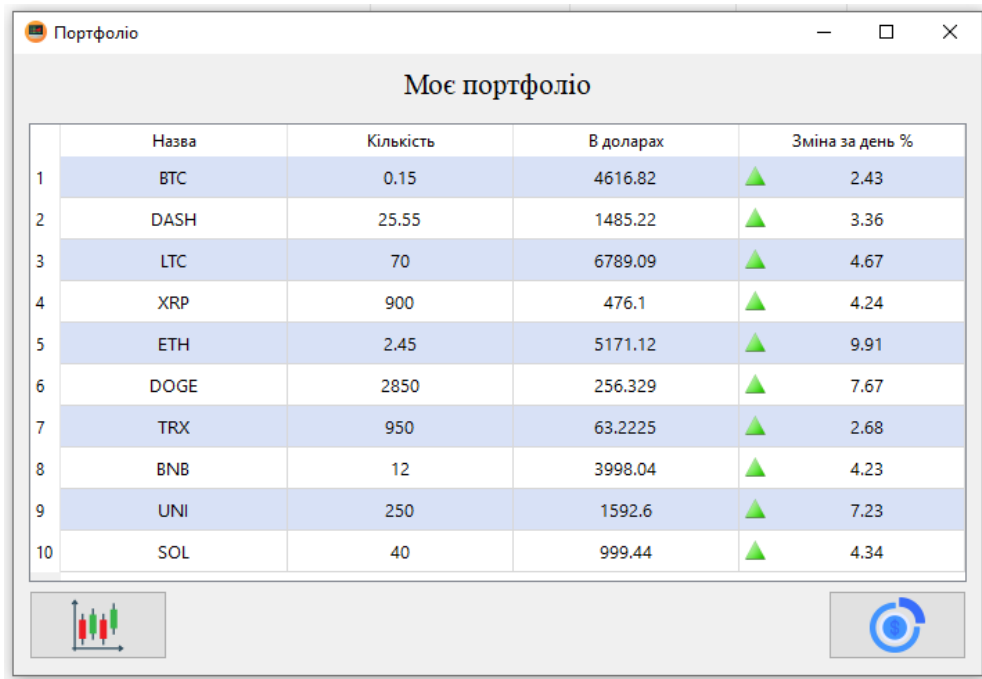


The screenshot shows a window titled 'Портфоліо' (Portfolio) with a subtitle 'Моє портфоліо' (My Portfolio). It contains a table with 5 columns: 'Назва' (Name), 'Кількість' (Quantity), 'В доларах' (In dollars), and 'Зміна за день %' (Change per day %). The table lists 10 cryptocurrencies: BTC, DASH, LTC, XRP, ETH, DOGE, TRX, BNB, UNI, and SOL. Each row shows a quantity of 0, a value of 0 in dollars, and a percentage change with a corresponding red or green arrow. At the bottom left, there is a small candlestick chart icon, and at the bottom right, there is a circular refresh icon.

	Назва	Кількість	В доларах	Зміна за день %
1	BTC	0	0	▼ -2.26
2	DASH	0	0	▼ -3.98
3	LTC	0	0	▼ -5.79
4	XRP	0	0	▼ -1.51
5	ETH	0	0	▼ -2
6	DOGE	0	0	▲ 2.93
7	TRX	0	0	▲ 0.72
8	BNB	0	0	▼ -0.86
9	UNI	0	0	▼ -1.25
10	SOL	0	0	▼ -3.59

Рисунок 3.10 – Інтерфейс вікна «Портфоліо»

Нижче, на рисунку 3.11 наведено приклад таблиці користувача, що має всі монети в своєму портфоліо, відповідно, в колонці «В доларах» вказано загальну вартість кожної з них. В даному прикладі усі монети показують ріст, оскільки навпроти кожної з них, в стовпчику «Зміна за день %» відсоток є додатнім, і колір стрілки визначено зеленим.



Мое портфоліо

	Назва	Кількість	В доларах	Зміна за день %
1	BTC	0.15	4616.82	▲ 2.43
2	DASH	25.55	1485.22	▲ 3.36
3	LTC	70	6789.09	▲ 4.67
4	XRP	900	476.1	▲ 4.24
5	ETH	2.45	5171.12	▲ 9.91
6	DOGE	2850	256.329	▲ 7.67
7	TRX	950	63.2225	▲ 2.68
8	BNB	12	3998.04	▲ 4.23
9	UNI	250	1592.6	▲ 7.23
10	SOL	40	999.44	▲ 4.34

Below the table, there is a small candlestick chart icon on the left and a circular refresh icon on the right.

Рисунок 3.11 – Портфолію користувача, зібране з усіх доступних монет

Після того, як користувач натискає кнопку переходу на наступне вікно, з'являється нова форма – «Кругова діаграма». Кругова діаграма побудована на основі класів модуля Qt Charts, а саме: QChart та QPieSlice. Дана діаграма має вбудовану анімацію відділення сегменту, який представляє собою дані про ту чи іншу цифрову монету або баланс на рахунку користувача. У разі відсутності деяких монет у портфелі користувача, на відміну від таблиці форми «Портфолію», дана діаграма не відображає їх відсотковий показник в загальній сумі вкладень. Приклад форми з розподіленням придбаної криптовалюти зображено на рис. 3.12.

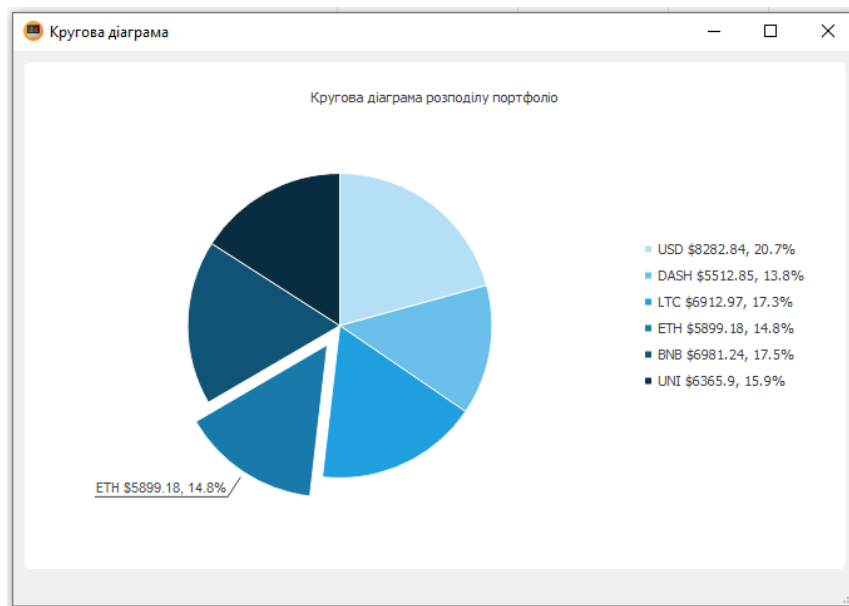


Рисунок 3.12 – Приклад форми «Кругова діаграма»

Нижче, на рисунку 3.13 наведено головну форму в момент взаємодії користувача, при цьому було здійснено декілька операцій купівлі та продажу, для перегляду діаграми обрано монету Dash в 15 хвилинному інтервалі побудови свічок.



Рисунок 3.13 – Приклад активного трейдингу на головній формі

Опис всіх форм показав, що розроблений користувацький інтерфейс є чітким, інтуїтивно зрозумілим, має кольорову гаму, основними кольорами якої є блакитний, червоний та зелений. Усі елементи розміщено згідно очікувань користувача, найбільшу схожість розроблений інтерфейс має з програмою-аналогом Binance, описану в пункті 1.1.

3.4 Особливості реалізації застосунку

Дана система має деякі особливості, притаманні лише настільним застосункам, що взаємодіють з веб-сервісами. Обрана криптобіржа Poloniex, що надає дані за допомогою API, використовує протокол HTTP з сертифікатом SSL/TLS від компанії CloudFlare, це означає, що для коректної взаємодії з даним ресурсом, до клієнту застосунку має бути підключена криптографічна бібліотека [22]. Було обрано криптографічну бібліотеку OpenSSL, що легко інтегрується для використання у веб-протоколі HTTPS. Для підключення усіх обов'язкових, динамічно приєднуваних бібліотек, у конфігураційному файлі проекту з розширенням «pro» необхідно вказати шляхи розміщення даних файлів. В даному випадку, в підкаталозі «dlls» окремо зберігаються бібліотеки, що забезпечують можливість надсилання запитів до API обраної криптобіржи, для 32 та 64 бітної версії Windows відповідно:

```
INCLUDEPATH += ../dlls/openssl_x32  
INCLUDEPATH += ../dlls/openssl_x64
```

Необхідно зазначити, що розроблюване програмне забезпечення використовує клас QSqlTableModel для легкого інтегрування даних з бази даних, але при цьому потребує деякого вдосконалення. Користувач, що здійснює торгові операції, має швидко орієнтуватися в своїх діях, тому елементи керування мають містити таку кольорову гаму, що забезпечить правильне сприйняття своїх вчинків.

В даному випадку при використанні QSqlTableModel, елемент QTableView відображатиме усі рядки, заповнені даними з бази даних з фоном білого кольору. Необхідно змінити спосіб відображення дій «купівля» та «продаж» у даній таблиці:

```
QVariant TableModelTradeHistory::data
(const QModelIndex &idx, int role) const {
    if(role == Qt::BackgroundRole) {
        if(QSqlTableModel::data
(this->index(idx.row(), 1)).toString() == "sell") {
            return QColor(255, 142, 136);
        }
        else if(QSqlTableModel::data
(this->index(idx.row(), 1)).toString() == "buy") {
            return QColor(173, 253, 135);
        }
    }
    else if(role == Qt::DisplayRole) {
        return QSqlTableModel::data(idx);
    }
    return QVariant();
}
```

Для цього було перевизначено метод «data», що використовується для повернення даних, які зберігаються в класі моделі, в певних ролях. В даному випадку, перевіряючи «BackgroundRole», встановлюється колір для поточного рядку, заповненого даними. Якщо текст другої комірки «sell», колір буде червоним, відповідно «buy» - зеленим.

Важливою особливістю даного застосунку є забезпечення коректності обрахунків та збереження балансу та кількості криптовалюти. Під час виконання математичних операцій з дробовими числами можуть виникати неточності, що можуть призвести до некоректного результату.

Розроблений клас MoneyConverter зберігає дані балансу не в доларах, а саме в центах:

$$y = 100 \times x, \quad (3.1)$$

де x – рахунок клієнта у вигляді числа з рухомою комою;

y – рахунок користувача, виражений в центах.

Відповідно, для занесення до бази даних кількості криптовалюти, використовується формула:

$$y = 1000 \times x, \quad (3.2)$$

де x – кількість криптовалюти у вигляді числа з рухомою комою, визначена з точністю до трьох знаків;

y – збережена кількість тієї чи іншої криптовалюти.

Для правильного відображення записаної інформації, необхідно провести зворотну дію, виразивши усі збережені дані у вигляді дробового числа. Таким чином, при виконанні торгових операцій «купівля» та «продаж» кількість криптовалюти та основного балансу користувача обраховується у вигляді цілого числа, відповідно, при відображенні на формі, балансу та криптовалютного портфелю, усі показники конвертуються в числа з рухомою комою.

3.5 Висновки за розділом 3

Було запропоновано діаграму класів, залежності та усі зв'язки між ними за допомогою уніфікованої мови моделювання UML. Створено функціональну схему та схему прецедентів для моделювання взаємодії користувача з розробленим програмним продуктом.

Було спроектовано графічний інтерфейс, визначено його кольорову гаму, розташування елементів керування згідно очікувань відвідувача, сформованих в результаті взаємодії з програмами-аналогами.

Описано особливості розробки настільного застосунку та наведено фрагменти програмного коду, що вирішують задачі побудови графічного інтерфейсу та динамічного оновлення даних на ньому.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення та умови застосування програми

Дана програма призначена для здобуття навичок торгівлі цифровими монетами, покращення розуміння принципу функціонування криптовалютного ринку та спостереження за зміною вартості цифрових активів. Для забезпечення користувача даними можливостями, застосунок містить наступні графічні представлення даних:

- таблиця торгових операцій, здійснених активним користувачем;
- динамічно оновлювані графіки японських свічок, що надають можливості змінювати масштаб та здійснювати його горизонтальну прокрутку;
- діаграма розподілу монет з врахуванням її ціни на момент перегляду портфолію, кожен сегмент якої є показником наявності цифрових монет у відсотковому відношенні.

Цільовою операційною системою є Windows, необхідна версія 7 та вище.

Важливою вимогою забезпечення працездатності системи є наявність компонентів «Microsoft Visual C++ Redistributable Packages». Даний набір компонентів включає деякі важливі DLL-бібліотеки, які використовуватимуться розробленою програмою.

4.2 Інструкція по експлуатації програмного забезпечення

Для використання даної програми було описано спосіб звернення до застосунку, дані, що будуть використані і оброблені під час експлуатації застосунку, а також принцип сповіщення користувача під час успішних дій або будь-яких помилок.

4.2.1 Звернення до програми

Для можливості звернення до програми через виконуваний файл, було використано аналізатор об'єктного коду Dependency Walker, який визначив обов'язкові динамічні бібліотеки, котрі необхідно помістити в каталог з виконуваним файлом (рис. 4.1).

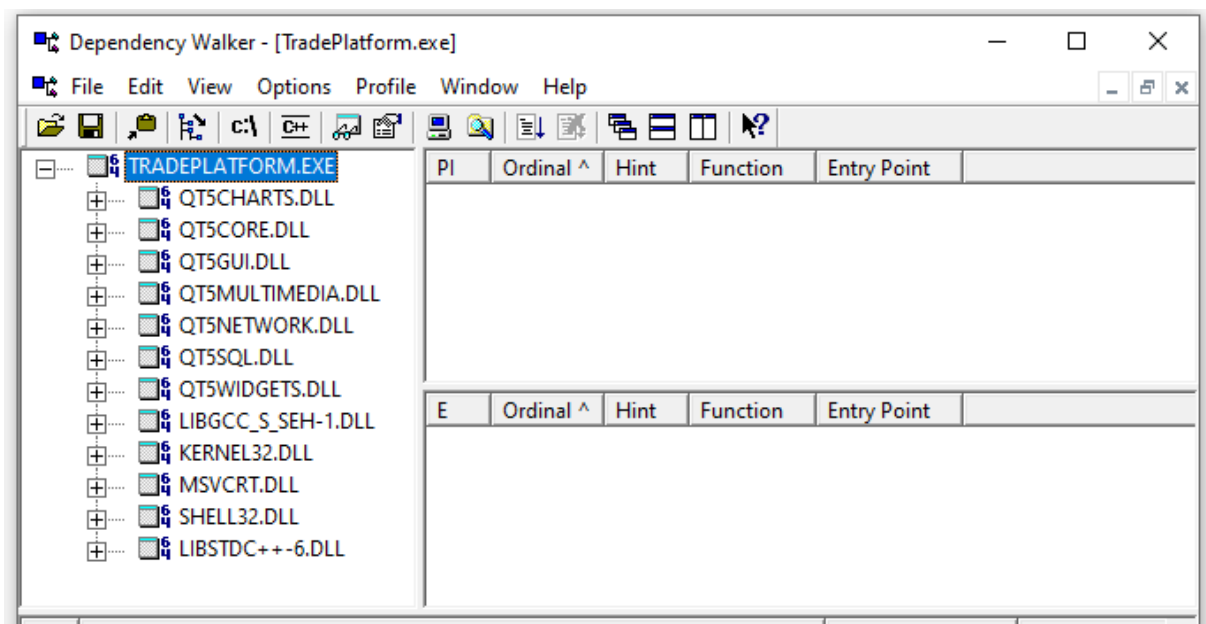


Рисунок 4.1 – Аналіз залежностей виконуваного файлу

Для використання розробленого програмного забезпечення, необхідно запустити виконуваний файл TradePlatform.exe.

4.2.2 Вхідні й вихідні дані

При запуску програми система використовує існуючі файли cryptocurrencies.txt та database.db. При цьому, у разі відсутності файлу бази даних, він буде створений автоматично. Під час використання застосунку, дані криптовалютного ринку також є вхідними та потребують деякої обробки перед використанням. Уся надана інформація надходить у вигляді файлів розширення JSON з певною структурою відповіді на конкретний запит.

До вихідних даних належать графіки, таблиця та діаграма, сформовані в результаті обробки даних. Створений обліковий запис, зміна балансу на рахунку, здійснені торгові операції, що записані до бази даних також відносяться до вихідних даних.

4.2.3 Повідомлення системи

Під час взаємодії з системою, у разі необхідності, програма повідомляє користувачу результат власних дій, при цьому відтворюється звук, що означає «успіх» та «помилка» у відповідних випадках. Нижче, в таблиці 4.1 приведено всі повідомлення, що можуть бути отримані під час експлуатації застосунку.

Таблиця 4.1 – Повідомлення системи

Текст	Причина появи	Можливі подальші дії
1	2	3
Не знайдено користувача з таким ім'ям!	База даних не містить користувача з таким ім'ям	Створити новий акаунт або увійти в існуючий
Користувач з таким ім'ям вже існує!	База даних вже містить користувача з таким ім'ям	Увійти в існуючий обліковий запис
Створено нового користувача: user	Обліковий запис створений та готовий до використання	Вхід в існуючий обліковий запис
Введіть число криптовалюти!	Купівля та продаж без вказування числа монет	Вказати додатне число криптовалюти
У вас недостатньо коштів!	Недостатньо грошей для операції купівлі монет	Продати наявну криптовалюту

Продовження табл. 4.1

1	2	3
У вас недостатньо криптовалют!	Недостатньо грошей для операції продажу монет	Зменшити обсяг для продажу монет
Неможливо переглянути портфель! Відсутнє підключення до мережі Інтернет!	Неможливо отримати дані про зміну цін за день, неможливо відобразити «Портфоліо»	Відновити підключення до мережі «Інтернет»

4.3 Тестування програмного забезпечення

Після розробки програми, необхідно виконати тестування, що відображатиме стан програмного забезпечення під час експлуатації. Було вирішено провести функціональне та нефункціональне тестування для повного огляду можливостей та пошуку недоліків чи уразливостей в роботі системи.

4.3.1 Функціональне тестування програми

Для перевірки відповідності створеного функціоналу з вимогами, поданими у пункті 2.1, необхідно провести функціональне тестування програми. Перевірка має здійснюватися за такими варіантами проведеного тестування:

- очікуваний результат;
- успішний сценарій;
- невдалий сценарій.

Функціональне тестування проводитиметься на машині з роздільною здатністю монітору 1366 × 768 px, процесором з частотою 2.5 ГГц під управлінням операційної системи Windows 10.

Результати функціонального тестування занесено нижче, в таблицю 4.2.

Таблиця 4.2 – Тестування функціональних вимог

Функціональна вимога	Дія користувача	Результат		
		Очікуваний результат	Успішний сценарій	Невдалий сценарій
1	2	3	4	5
Можливість реєстрації	Натискання кнопки «Створити новий обліковий запис»	Створено новий обліковий запис з демо рахунком	Введено коректний нікнейм та створено обліковий запис	Обліковий запис вже існує
Можливість входу в існуючий обліковий запис	Натискання кнопки «Увійти в особистий кабінет»	Перехід до форми «Головне вікно»	Введено коректне ім'я, перехід до форми «Головне вікно»	Облікового запису не існує
Купівля криптовалюти	Натискання кнопки «Купити»	Здійснено операцію купівлі	Введено число, достатнє для купівлі	Недостатньо коштів
Продаж криптовалюти	Натискання кнопки «Продаж»	Здійснено операцію продажу	Введено число криптовалюти, достатнє для купівлі	Недостатньо криптовалюти
Перемикання торгових періодів	Перемикання кнопки з вказаним періодом	Оновлено графік за новим періодом	Створено графік за новим періодом	При відсутності підключення неможливо обрати період
Можливість прокрутки графіку зміни ціни	Використання миші (колесо, ліва кнопка)	Зміщення графіку по осі X	Здійснено прокрутку, змінено масштаб графіку	Невдалого сценарію не виявлено

Продовження табл. 4.2

1	2	3	4	5
Зміна поточної монети для торгових операцій	Перемикання в таблиці «Обрана монета»	Визначено нову поточну монету для торгових операцій	Оновлено графік, встановлено ціну монети	При відсутності підключення неможливо змінити монету
Перегляд даних свічки при наведенні	Наведення миші на обрану свічку	Відображення надпису з даними про свічку	Знайдено свічку та встановлено дані про неї	Невдалого сценарію не виявлено
Перегляд таблиці портфоліо	Відкриття форми «Портфоліо»	Відображення таблиці портфоліо	Перехід до форми «Портфоліо»	При відсутності підключення неможливо відкрити форму
Перегляд кругової діаграми розподілу криптовалюти	Відкриття форми «Кругова діаграма»	Відображення кругової діаграми	Перехід до форми «Кругова діаграма»	Невдалого сценарію не виявлено

Наведена таблиця показує, що головною причиною неуспішного сценарію перевірки функціональних вимог є повна відсутність або низький рівень підключення до мережі «Інтернет».

Проведене тестування свідчить про коректну роботу усіх функцій, при цьому, під час тестування деяких функціональних вимог не було виявлено невдалих сценаріїв, а отже, розроблена система є досить надійною.

Далі буде виконано нефункціональне тестування, що означає перевірку властивостей системи, які не відповідають функціональності системи, це означає виконання перевірки того, як система працює, а не конкретна поведінка цієї системи.

4.3.2 Нефункціональне тестування програми

Оскільки застосунок передбачає одну цільову ОС, було вирішено провести конфігураційне тестування для виявлення проблем несумісності програми з різними конфігураціями платформи Windows. Найпопулярнішими версіями цільової ОС є Windows 7, 10 та 11 версії [25]. Розроблене програмне забезпечення зможе працювати на 95% обчислювальних машин, на яких встановлена ОС Windows при умові позитивного результату тестування.

Програма використовує криптографічну бібліотеку OpenSSL, яка надає окремі динамічні бібліотеки для 32 та 64 бітних версій ОС Windows, тому збірки розроблюваної програми відрізняються.

Для перевірки працездатності програми, було підготовлено та укомплектовано усіма необхідними dll, що використовувалися при розробці, два архіва – для 32 бітної та 64 бітної цільової ОС відповідно.

У зв'язку з тим, що остання 11 версія компанії Microsoft не передбачає 86 бітної архітектури, тестування проводитиметься лише на 64 бітній версії найновішого продукту, при цьому, буде перевірено на коректність роботи 32 бітні версії Windows 7 та Windows 10.

Розроблене програмне забезпечення має пройти тестування за параметрами, вплив на роботу системи яких є найвищим.

В першу чергу, необхідно перевірити коректність роботи застосунку при різній швидкості підключення до мережі «Інтернет», так як розроблений функціонал цілком залежить від наявності підключення, при цьому можливе підключення за допомогою різних типів з'єднання. Ще одним параметром є показник максимально можливої роздільної здатності дисплея, що коректно відображатиме користувацький інтерфейс. Врешті, коректність роботи програми має бути перевірено на різних версіях цільової ОС Windows.

Нижче, до таблиці 4.3 занесено розглянуті параметри, що перевірятимуться під час конфігураційного тестування.

Таблиця 4.3 – Конфігураційне тестування програми

Операційна система	Розрядність системи	Роздільна здатність	Тип з'єднання	Швидкість з'єднання	Результат
Windows 7	32-bit	800 × 600 px	Ethernet	40 Mbit	Пройдено
Windows 7	64-bit	1600 × 900 px	Ethernet	40 Mbit	Пройдено
Windows 10	64-bit	1920 × 1080 px	Wi-Fi	32 Mbit	Пройдено
Windows 10	64-bit	1366 × 768 px	Mobile	4 Mbit	Не пройдено
Windows 10	32-bit	1024 × 768 px	Mobile	10 Mbit	Пройдено
Windows 11	64-bit	2440 × 1440 px	Wi-Fi	37 Mbit	Не пройдено

В ході тестування програмного забезпечення було виявлено, що при низькій швидкості з'єднання з мережею Інтернет, система відображає анімацію завантаження замість графіка зміни ціни монети (рис. 4.2). При цьому, ціна поточної монети постійно оновлюється.

Отже, для зручного використання програмного забезпечення необхідне підключення зі швидкістю щонайменше 5 Мбіт/сек.

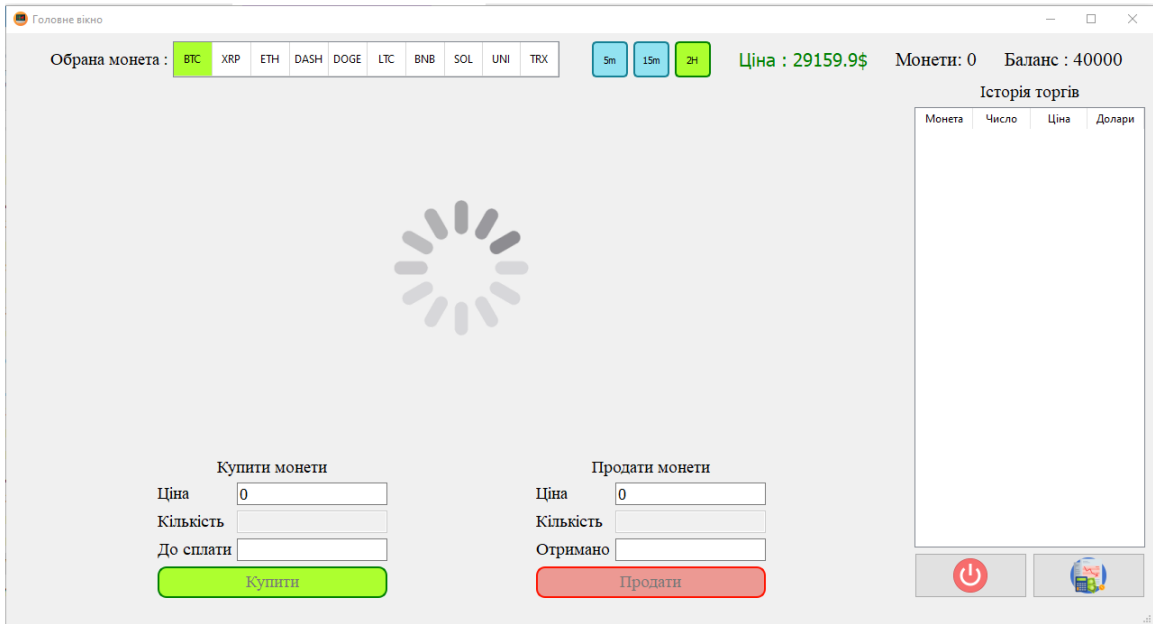


Рисунок 4.2 – Робота застосунку при низькій швидкості підключення

Також було визначено, що при досить високій роздільній здатності монітору, а саме 2440×1440 px, такі елементи, як графік ціни монети та таблиця історії операцій мають непропорційні розміри відносно одне одного (рис.4.3). Це виникає тому, що розміри елементів керування прив'язуються до піксельних показників, використовуючи різні елементи розтягування на формі.



Рисунок 4.3 – Вигляд застосунку при роздільній здатності 2440×1440 px

Таким чином, проведене конфігураційне тестування свідчить про те, що розроблене програмне забезпечення може працювати на різних версіях операційної системи Windows, з мінімальною швидкістю підключення до мережі Інтернет 5 Mbit/s і роздільною здатністю щонайменш 800 × 600px. Програмне забезпечення не викликало аварійних завершень, так як система передбачає достатньо різних перевірок для запобігання збоїв, викликаних відсутністю підключення до мережі "Інтернет", відсутності файлу бази даних, та інших можливих ситуацій.

4.4 Висновки за розділом 4

В даному розділі було визначено умови застосування програми, спосіб звернення до неї а також дані, що подаються системі та локально зберігаються.

Було проведено тестування, завдяки якому перевірено працездатність застосунку при різних конфігураціях цільової ОС Windows. Перевірено відповідність між реальною поведінкою реалізованого функціоналу і очікуваною поведінкою відповідно до специфікації і функціональних вимог.

Було описано всі можливі повідомлення системи, що можуть бути отримані при експлуатації розробленого застосунку.

ВИСНОВКИ

У результаті виконання дипломної кваліфікаційної роботи бакалавра було створено криптовалютну біржу з демонстраційним рахунком для здобуття базових навичок торгівлі криптовалютою, спостереження за рухом вартості цифрових активів та аналізу результатів власних дій та рішень.

Для набуття програмою особливостей, що вирізнятимуть її серед існуючих програм, було розглянуто існуючі рішення та проаналізовано причини проблеми відсутності інструментів для систематизації навичок та вмінь у віртуальних рахунках найпопулярніших трейдингових платформах. Було описано функціональні вимоги, основою яких є базова функціональність аналогів.

Обраною мовою програмування є мова C++, зважаючи на її переваги у продуктивності при розробці систем з постійним та динамічним оновленням даних та їх обробкою. Для побудови та відображення діаграми та графіку використано модуль «Qt Charts», наданий середовищем Qt Creator.

Для отримання даних криптовалютного ринку, виражених у числових показниках, було обрано сервіс обміну криптовалют Poloniex.

Визначено архітектуру системи та шаблони, побудовано діаграму класів, залежності та усі зв'язки між ними, згідно яких розроблено програму. Створена функціональна схема та схема прецедентів визначили модель взаємодії користувача з готовою системою.

Після цього, було спроектовано та розроблено графічний інтерфейс, визначено його кольорову гаму. Налаштовано криптографічну бібліотеку OpenSSL та реалізовано весь необхідний функціонал. Для зручної експлуатації застосунку, описано всі можливі повідомлення системи, що можуть бути отримані при взаємодії з програмою.

Проведене тестування функціональних вимог та здатність розробленого застосунку працювати при різних конфігураціях цільової ОС Windows показали, що розроблений застосунок готовий до використання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. State of crypto in Ukraine [Electronic resource]. – Access mode: <https://triple-a.io/crypto-ownership-ukraine-2022>.
2. Most expensive ad themes on Telegram [Electronic recourse]. – Access mode: <https://cointelegraph.com/news/crypto-spawns-most-expensive-adverts-on-telegram-in-2021-data>.
3. Підписання закону «Про віртуальні активи» [Електронний ресурс]. – Режим доступу: <https://minfin.com.ua/2022/03/16/82176211>.
4. Інтерв'ю з головою криптобіржи «Binance» [Електронний ресурс]. – Режим доступу: <https://ain.ua/ru/2021/02/11/interview-binance-2021>.
5. Трейдер [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Трейдер>.
6. Damodaran, A. Investment Philosophies: Successful Strategies and the Investors Who Made Them Work, 2nd Edition [Text] / A. Damodaran - New Jersey: John Wiley & Sons, 2012. – 121p.
7. Why most of day traders lose money [Electronic resource]. – Access mode: <https://www.daytradetheworld.com/trading-blog/why-day-traders-lose-money>.
8. Percentage of day Traders Make Money [Electronic resource]. – Access mode: <https://www.cnn.com/2020/11/20/attention-robinhood-power-users-most-day-traders-lose-money.html>.
9. Report from S&P Dow Jones Indices [Electronic resource]. – Access mode: <https://www.spglobal.com/spdji/en/documents/spiva/spiva-us-mid-year-2020.pdf>.
10. Історії людей, що випадково розбагатіли на криптовалюті [Електронний ресурс]. – Режим доступу: <https://laba.ua/blog/3309-7-istoriy-o-lyudyah-kotorye-razbogateli-na-kriptovalyute>.
11. Binance Trade Volume, Trade Pairs & Trust Score [Electronic resource]. – Access mode: <https://www.coingecko.com/en/exchanges/binance>.

12. Ninja Trader 8 - [Electronic resource]. – Access mode: <https://ninjatrader.com/support/helpguides/nt8/nt%20helpguide%20english.html?ninjaascrip.htm>.

13. Японські свічки [Electronic resource]. – Access mode: https://uk.wikipedia.org/wiki/Японські_свічки.

14. What is cryptocurrency Tether? [Electronic resource]. – Access mode: <https://kriptomat.io/cryptocurrencies/tether/what-is-tether>.

15. Час Unix [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Час_Unix.

16. Supported Platforms & Languages for Software Development | Qt [Electronic resource]. – Access mode: <https://www.qt.io/product/supported-platforms-languages>.

17. Common Patterns | Qt 6 Book [Electronic resource]. – Access mode: <https://www.qt.io/product/qt6/qml-book/ch06-controls-common-patterns>.

18. Model/View Programming | Qt Widgets 5.15.13 [Electronic resource]. – Access mode: <https://doc.qt.io/qt-5/model-view-programming.html>.

19. Thelin, J. Foundations of Qt Development [Text] / J. Thelin - New York: Apress, 2007. – 371p.

20. Oliveira, C. Practical C++ Financial Programming [Text] / C. Oliveira - New York: Apress, 2015. – 91p.

21. Nison, S. Beyond Candlesticks: New Japanese Charting Techniques Revealed [Text] / S. Nison – New York: John Wiley & Sons, 1994. – 249p.

22. SSL Certificate Update News – Poloniex [Electronic resource]. – Access mode: <https://support.poloniex.com/hc/en-us/articles/4402512146839-SSL-Certificate-Update>.

23. Round-off-error [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Round-off_error.

24. What Every Computer Scientist Should Know About Floating-Point Arithmetic [Electronic resource]. – Access mode: https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html.

25. Desktop Windows Version Market Share Ukraine | Statcounter Global Stats [Electronic resource]. – Access mode: <https://gs.statcounter.com/os-version-market-share/windows/desktop/ukraine>.
26. Introduction – Poloniex API Reference [Electronic resource]. – Access mode: <https://docs.legacy.poloniex.com>.
27. Qt for Windows – Deployment [Electronic resource]. – Access mode: <https://doc.qt.io/qt-6/windows-deployment.html>.
28. Dependency Walker (depends.exe) Home Page [Electronic resource]. – Access mode: <http://www.dependencywalker.com>
29. QNetworkAccessManager Class | Qt Network [Electronic resource]. – Access mode: <https://doc.qt.io/qt-6/qnetworkaccessmanager.html#details>.
30. Тестування програмного забезпечення [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення#Нефункціональне_тестування.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення трейдингової платформи з демонстраційним рахунком застосовується для отримання базових навичок управління криптовалютним портфоліо, спостереження за рухом зміни вартості цифрових монет та перегляду успішності власних дій, торгових стратегій та ідей.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Розробка криптовалютної біржі з демо рахунком», затверджене наказом Національного університету «Запорізька політехніка» № 87 від 4 квітня 2023 р.

A.2 Призначення розробки

Програмне забезпечення призначене для відпрацьовування торгових стратегій, перегляду успішності власних рішень та аналізу ситуації на криптовалютному ринку в режимі «онлайн».

Результатом використання даної програми є підвищення практичних навичок та опанування стратегії торгування тими чи іншими активами кінцевого користувача.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Криптовалютна біржа з демо рахунком повинна задовольняти такі функціональні характеристики:

- створення облікового запису;
- вхід в існуючий обліковий запис;

- відображення кругової діаграми розподілення монет у відсотках;
- відображення таблиці з портфоліо криптовалют, їх наявності, зміна вартості за останню добу, загальна вартість кожної з куплених монет;
- відображення графіка зміни вартості кожної монети (за поточною, обраною монетою);
- відображення графіка зміни вартості за годинами, хвилинами;
- можливість прокручувати графік для перегляду попередніх позицій криптовалюти за визначеним інтервалом;
- масштабування графіку та автоматичне оновлення останньої свічки;
- відображення торгових операцій користувача;
- відображення балансу в доларах;
- купівля та продаж обраної монети.

A.3.2 Вимоги до інтерфейсу програми

Програмне забезпечення у вигляді криптовалютної біржі повинне забезпечувати оптимальне відображення усіх форм, незалежно від роздільної здатності монітору, що відображає програму, а також мати чіткий, інтуїтивно зрозумілий інтерфейс.

A.3.3 Вимоги до надійності

Програмне забезпечення у вигляді криптовалютної біржі повинно містити достатньо перевірок для виявлення помилок та збійних ситуацій, забезпечуючи при цьому коректну подальшу роботу програми.

A.3.4 Вимоги експлуатації

Під час роботи програми необхідно забезпечити безперебійне енергопостачання комп'ютера, завершення роботи з програмою слід

завершувати за допомогою спеціального елемента керування на головній формі.

Обов'язковою вимогою є наявність компонентів «Microsoft Visual C++ Redistributable Packages». Даний набір компонентів включає деякі важливі DLL-бібліотеки, які використовуватимуться програмою.

А.3.5 Вимоги до складу та параметрів технічних засобів

Для експлуатації програмного забезпечення необхідно використовувати апаратне забезпечення, мінімальні параметри якого зазначені нижче:

- операційна система Windows 7;
- процесор з тактовою частотою 1 ГГц;
- роздільна здатність дисплею 800 × 600;
- оперативна пам'ять обсягом не менше 2 Гб;
- наявність пристроїв введення інформації;
- 1 Гб вільного місця на диску.

А.3.6. Вимоги до маркування й пакування

Програмне забезпечення у вигляді криптовалютної біржі може бути записане на USB-флеш-накопичувач, оптичний, жорсткий магнітний диск. На упаковці повинна бути зазначена назва програми – «Криптовалютна біржа з демо рахунком».

А.3.7 Порядок контролю та приймання

Дипломна робота бакалавра повинна бути виконана у терміни, зазначені календарним планом, допущена до захисту керівником, консультантом та нормоконтролером, затверджена завідувачем кафедри та захищена у присутності членів екзаменаційної комісії.

ДОДАТОК Б
Текст програми

Б.1 Файл TradePlatform.pro

```

QT += core gui charts sql multimedia
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets network
CONFIG += c++17

# You can make your code fail to compile if it uses deprecated
APIs.

# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables
all the APIs deprecated before Qt 6.0.0

SOURCES += \
    api/apiaddressbuilder.cpp \
    api/apiservice.cpp \
    api/apiservicerresponse.cpp \
    candles/candlegraphbuilder.cpp \
    candles/candlesticklist.cpp \
    candles/candlesticklistbuilder.cpp \
    candles/chartview.cpp \
    main.cpp \
    piechart/drilldownchart.cpp \
    piechart/drilldownslice.cpp \
    storage/database.cpp \
    tablemodel/tablemodeltradehistory.cpp \
    utils/moneyconverter.cpp \
    utils/sound.cpp \
    utils/txtreader.cpp \
    visitor/user.cpp \
    window/loginwindow.cpp \
    window/piechartwindow.cpp \
    window/portfoliowindow.cpp \
    window/tradewindow.cpp

```

```

HEADERS += \
    api/apiaddressbuilder.h \
    api/apiservice.h \
    api/apiserviceresponse.h \
    candles/candlegraphbuilder.h \
    candles/candlesticklist.h \
    candles/candlesticklistbuilder.h \
    candles/chartview.h \
    piechart/drilldownchart.h \
    piechart/drilldownslice.h \
    storage/database.h \
    tablemodel/tablemodeltradehistory.h \
    utils/moneyconverter.h \
    utils/sound.h \
    utils/txtreader.h \
    visitor/user.h \
    window/loginwindow.h \
    window/piechartwindow.h \
    window/portfoliowindow.h \
    window/tradewindow.h \
    window/ui_loginwindow.h \
    window/ui_piechartwindow.h \
    window/ui_portfoliowindow.h \
    window/ui_tradewindow.h

```

```

FORMS += \
    window/loginwindow.ui \
    window/piechartwindow.ui \
    window/portfoliowindow.ui \
    window/tradewindow.ui

```

```

# Default rules for deployment.
qnx: target.path = /tmp/${TARGET}/bin
else: unix:!android: target.path = /opt/${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

INCLUDEPATH += ../dlls/openssl_x32
INCLUDEPATH += ../dlls/openssl_x64

DISTFILES += \
    images/arrow-left.png \
    images/arrow-right.png \
    images/candlestick.png \
    images/chart.png \
    images/economics.png \
    images/exit.png \
    images/green_triangle.png \
    images/icon.ico \
    images/icon.png \
    images/loading.gif \
    images/red_triangle.png \
    sound/click-error.wav \
    sound/closing_effect.wav \
    sound/interface-hover.wav \
    sound/interface_click.wav \
    sound/joystick_trigger.wav \
    sound/process_complete.wav

```

Б.2 Файл main.cpp

```

#include "window/loginwindow.h"
#include <QApplication>
#include <utils/sound.h>
int main(int argc, char *argv[]) {

```

```

    QApplication a(argc, argv);
    a.setWindowIcon(QIcon("./images/icon.png"));
    Sound *sound = new Sound();
    LoginWindow loginWindow(sound);
    loginWindow.show();
    return a.exec();
}

```

Б.3 Файл apiaddressbuilder.h

```

#ifndef APIADDRESSBUILDER_H
#define APIADDRESSBUILDER_H

#include <QString>
#include <ctime>
#include <QDebug>

class ApiAddressBuilder {
private:
    ApiAddressBuilder();
public:
    static QString getChartData(QString currency_pair, QString
period, QString limit);
    static QString getPriceCryptoPair(QString pair);
    static QString getDataAllCryptoPairs();
};

#endif // APIADDRESSBUILDER_H

```

Б.4 Файл apiservice.h

```

#ifndef APISERVICE_H
#define APISERVICE_H

#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QNetworkReply>

```

```

#include <QtNetwork>

class ApiService
{
private:
    ApiService(){};
    static QNetworkAccessManager *accessManager;
public:
    static QJsonDocument MakeRequest(QString requestStr);
};

#endif // APISERVICE_H

```

Б.5 Файл apiservicerresponse.h

```

#ifndef APISERVICERESPONSE_H
#define APISERVICERESPONSE_H
#include <QJsonDocument>
class ApiServiceResponse // объект сервиса
{
private:
    QJsonDocument response;
public:
    ApiServiceResponse(QJsonDocument response_service) : response
(response_service) { };
    QJsonDocument get_response();
};

#endif // APISERVICERESPONSE_H

```

Б.6 Файл apiservicerresponse.cpp

```

#include "apiservicerresponse.h"
QJsonDocument ApiServiceResponse::get_response() {
    return response;
}

```

```
}
```

Б.7 Файл `apiaddressbuilder.cpp`

```
#include "apiaddressbuilder.h"
```

```
QString ApiAddressBuilder::getChartData(QString currency_pair,
QString period, QString limit) {
    return QString("https://api.poloniex.com/markets/")
        + currency_pair + QString("/candles?")
            + QString("interval=") + period
            + QString("&limit=")
            + limit;
}
```

```
QString ApiAddressBuilder::getPriceCryptoPair(QString pair) {
    return QString("https://api.poloniex.com/markets/") +
        pair + QString("/markPrice");
}
```

```
QString ApiAddressBuilder::getDataAllCryptoPairs() {
    return QString("https://api.poloniex.com/markets/")
        + QString("price");
}
```

Б.8 Файл `apiservice.cpp`

```
#include "apiservice.h"
#include <qeventloop.h>
#include <QDebug>

QNetworkAccessManager* ApiService::accessManager = new
QNetworkAccessManager();

QJsonDocument ApiService::MakeRequest(QString requestStr) {
    QNetworkRequest request((QUrl(requestStr)));
```

```

        request.setTransferTimeout(3000);
        request.setHeader(QNetworkRequest::ContentTypeHeader,
"application/json");
        QNetworkReply *reply = accessManager->get(request);
        while (!reply->isFinished()) {
            // QEventLoop::ExcludeUserInputEvents
            QApplication->processEvents();
        }
        QByteArray response_data = reply->readAll();
        reply->deleteLater();
        return QJsonDocument::fromJson(response_data);
    }

```

Б.9 Файл `tablemodeltradehistory.h`

```

#ifndef TABLEMODELTRADEHISTORY_H
#define TABLEMODELTRADEHISTORY_H

#include <QSqlTableModel>
class TableModelTradeHistory : public QSqlTableModel {
    Q_OBJECT
public:
    explicit TableModelTradeHistory(QObject *parent = nullptr);
    QVariant data(const QModelIndex &idx, int role) const;
};

#endif // TABLEMODELTRADEHISTORY_H

```

Б.10 Файл `tablemodeltradehistory.cpp`

```

#include "tablemodel/tablemodeltradehistory.h"
#include <QBrush>

TableModelTradeHistory::TableModelTradeHistory(QObject *parent)

```

```

        : QSqlTableModel{parent} {}

QVariant TableModelTradeHistory::data(const QModelIndex &idx, int
role) const {
    if(role == Qt::BackgroundRole) {
        if(QSqlTableModel::data(this->index(idx.row(),
1)).toString() == "sell") {
            return QColor(255, 142, 136);
        }
        else if(QSqlTableModel::data(this->index(idx.row(),
1)).toString() == "buy") {
            return QColor(173, 253, 135);
        }
    }
    else if(role == Qt::DisplayRole) {
        return QSqlTableModel::data(idx);
    }
    return QVariant();
}

```

Б.11 Файл database.h

```

#ifndef DATABASE_H
#define DATABASE_H
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QPair>
#include <QVector>

class Database
{
public:
    static void initDatabase();
    static void openDatabase();
    static void closeDatabase();

```

```

    static bool userIsExist(QString user_name);
    static bool accountIsCreated(QString user_name);
    static void createEmptyFieldsPortfolio(QString user_name);
    static bool rewriteBalance(QString user_name, double balance);
    static double getBalance(QString user_name);
    static void createAllTables();
private:
    Database(){};
    static QSqlDatabase db;
    static void createFields(QSqlQuery query);
};
#endif // DATABASE_H

```

Б.12 Файл database.cpp

```

#include "storage/database.h"
#include <QDir>
#include "qvariant.h"
#include <QDebug>
#include "utils/txtreader.h"
#include <QFileInfo>
#include <QSslSocket>
#include <QMessageBox>
#include <utils/moneyconverter.h>

QSqlDatabase Database::db = QSqlDatabase::addDatabase("SQLITE");

void Database::initDatabase() {
    db.setDatabaseName("./db/database.db");
}

void Database::openDatabase() {
    initDatabase();
    bool db_file_exist = QFileInfo::exists("./db/database.db");
}

```

```

db.open();
if(db.isOpen()) {
    qDebug() << "database is open now!";
}
Else {
    qDebug() << "database can not open!";
}
if(db_file_exist == false) {
    createAllTables();
}
}

void Database::closeDatabase() {
    db.close();
    qDebug() << "database was closed!";
}

bool Database::userIsExist(QString user_name) {
    QSqlQuery query;
    query.exec("SELECT * from User");
    while(query.next()) {
        QString temp_name = query.value(0).toString();
        qDebug() << "db:" + temp_name;
        if(user_name == temp_name) {
            return true;
        }
    }
    return false;
}

```

Б.13 Файл sound.h

```

#ifndef SOUND_H
#define SOUND_H

```

```

#include <QtMultimedia/QSoundEffect>

class Sound {
public:
    Sound();
    void transitionOnAnotherWindow();
    void error();
    void hover();
    void simpleClick();
    void processComplete();
    void close();
private:
    QSoundEffect *soundEffect;
    QString path;
};

#endif // SOUND_H

```

Б.14 Файл sound.cpp

```

#include "utils/sound.h"
#include "QUrl"

Sound::Sound()
{
    path = "./sound/";
    soundEffect = new QSoundEffect();
    soundEffect->setVolume(0.2f);
}

void Sound::transitionOnAnotherWindow() {
    soundEffect->setSource(QUrl::fromLocalFile(path
    + "joystick_trigger.wav"));
    soundEffect->play();
}

void Sound::error() {
    soundEffect->setSource(QUrl::fromLocalFile(path
    + "click-error.wav"));
}

```

```

        soundEffect->play();}
void Sound::processComplete() {
    soundEffect->setSource(QUrl::fromLocalFile(path
    + "process_complete.wav"));
    soundEffect->play();
}

```

Б.15 Файл candlesticklistbuilder.h

```

#ifndef CANDLESTICKLISTBUILDER_H
#define CANDLESTICKLISTBUILDER_H

#include "candlesticklist.h"
#include "api/apiserviceresponse.h"
#include <QtCharts/QCandleStickSet>

class CandleStickListBuilder : public CandleStickList {
private:
    ApiServiceResponse apiServiceResponse;
public:
    CandleStickListBuilder(ApiServiceResponse response) :
apiServiceResponse(response){ }
    QList<QtCharts::QCandlestickSet*>
    get_list_candlestick() override;
};
#endif // CANDLESTICKLISTBUILDER_H

```

Б.16 Файл candlesticklistbuilder.cpp

```

#include "candlesticklistbuilder.h"
QList<QtCharts::QCandlestickSet*>
CandleStickListBuilder::get_list_candlestick() {
    QList<QtCharts::QCandlestickSet*> list;

```

```

QJsonArray jsonArray =
apiServiceResponse.get_response().array();
foreach(const QJsonValue &value, jsonArray) {
    QJsonArray nested_json = value.toArray();
    qreal low = nested_json.at(0).toString().toDouble();
    qreal high = nested_json.at(1).toString().toDouble();
    qreal open = nested_json.at(2).toString().toDouble();
    qreal close = nested_json.at(3).toString().toDouble();
    qreal date = nested_json.at(12).toDouble();
    list.append(this->createCandleStickSet
                (date, open, close, low, high));
}
return list;
}

```

Б.17 Файл candlegraphbuilder.h

```

#ifndef CANDLEGRAPHBUILDER_H
#define CANDLEGRAPHBUILDER_H
#include <QtCharts/QDateTimeAxis>
#include "candles/chartview.h"
#include "candles/candlesticklist.h"
class CandleGraphBuilder {
public:
    CandleGraphBuilder();
    void refresh_graph_builder();
    void createNewChart();
    void addAllCandleStickSets
        (QList<QtCharts::QCandlestickSet*> list);
    void insertLastCandle(qreal timestamp, qreal open,
        qreal close, qreal high, qreal low);
    ChartView* getGraphChartView();
    CandleStickList *CandleStickList;
private:

```

```

        void initAcmeSeries();
        void initAxes();
    };
#endif // CANDLEGRAPHBUILDER_H

```

Б.18 Файл candlegraphbuilder.cpp

```

#include "candlegraphbuilder.h"
#include <QDebug>

CandleGraphBuilder::CandleGraphBuilder() {
    chart = nullptr;
    oldChart = nullptr;
    chartView = new ChartView();
    CandleStickList = nullptr;
    chartView->setRenderHint(QPainter::Antialiasing);
    initAcmeSeries();
}

void CandleGraphBuilder::initChartSettings() {
    chart->legend()->setVisible(true);
    chart->legend()->setAlignment(Qt::AlignBottom);
}

void CandleGraphBuilder::createNewChart() {
    oldChart = chart;
    chart = new QtCharts::QChart();
    chart->addSeries(acmeSeries);
    initChartSettings();
    initAxes();
    chartView->setChart(chart);
    chartView->update();
    candle->setPos(105, 45);
    candle->setFont(font);
    chart->scene()->addItem(candle);}

void CandleGraphBuilder::initAxes() {
    axisValue = new QValueAxis;

```

```

    axisDate = new QDateTimeAxis;
    chart->addAxis(axisValue, Qt::AlignLeft);
    chart->addAxis(axisDate, Qt::AlignBottom);
    acmeSeries->attachAxis(axisDate);
    acmeSeries->attachAxis(axisValue);
    axisDate->setFormat("dd.MM-hh:mm");
    axisDate->setTitleText("Дата");
    axisValue->setTitleText("Цена");
}

ChartView* CandleGraphBuilder::getGraphChartView() {
    return chartView;
}

void CandleGraphBuilder::refresh_graph_builder() {
    acmeSeries->clear();
    chartView->list_candlestick_set.clear();
}

void CandleGraphBuilder::refreshLastCandle
(qreal open, qreal close, qreal high, qreal low) {
    chartView->list_candlestick_set.last()->setClose(close);
    chartView->list_candlestick_set.last()->setHigh(high);
    chartView->list_candlestick_set.last()->setOpen(open);
    chartView->list_candlestick_set.last()->setLow(low);
    chartView->chart()->update();
    chartView->update();
}

```

Б.19 Файл tradewindow.h

```

#ifndef TRADEWINDOW_H
#define TRADEWINDOW_H
#include "api/apiservice.h"
#include "api/apiaddressbuilder.h"
#include "candles/candlegraphbuilder.h"
#include "window/portfoliowindow.h"

```

```

#include "tablemodel/tablemodeltradehistory.h"
#include "api/apiservicerresponse.h"

#include <QMainWindow>
#include <QGridLayout>
#include <QSqlTableModel>
#include <QMessageBox>
#include "visitor/user.h"
#include "utils/sound.h"

enum Interval { FIVE_MINUTES, FIFTEEN_MINUTES, TWO_HOURS };

QT_BEGIN_NAMESPACE
namespace Ui { class TradeWindow; }
QT_END_NAMESPACE

class TradeWindow : public QMainWindow
{
    Q_OBJECT

public:
    TradeWindow(User *user, QWidget *parent = nullptr);
    ~TradeWindow();
    void InitTimers();
    void setDefaultSettings();
    void setCurrentInterval(Interval interval);
    void changeCryptoPair(QString pair);
    void stopAllRequests();
    void startAllRequests();
    void getSoundObj(Sound *sound);
private:
    Ui::TradeWindow *ui;
    PortfolioWindow *portfolioWindow;
    QLabel *label_process;

```

```

    bool no_internet_connection;
    double last_price = 0;
signals:
    void sendSoundObj(Sound *sound);
public slots:
    void getChartGeneral();
    void getChartData5Minutes();
    void getChartData15Minutes();
    void getChartData2Hours();
    void getLastCandle();
    void setLastCandle(QJsonDocument document);
    void drawDiagram();
    void reDrawCandleChart(ApiServiceResponse response);
private slots:
    void closeEvent(QCloseEvent *event) override;
    void showEvent(QShowEvent *event) override;
    void hideEvent(QHideEvent *event) override;
    void closeApp();
    void clearFields();
    void init_table_coins();
    void on_btn_5_minutes_clicked();
    void on_btn_15_minutes_clicked();
    void on_btn_2_hours_clicked();
    void on_table_coins_cellClicked(int row, int column);
    void update_balance_label();
    void setDiragramInsteadLoading();
    void initLoading();
    void disabledBuySellAction();
    void enableBuySellAction();
    void on_table_coins_cellEntered(int row, int column);
};
#endif // TRADEWINDOW_H

```

Б.20 Файл tradewindow.cpp

```

#include "window/tradewindow.h"
#include "window/ui_tradewindow.h"
#include "utils/txtreader.h"
#include "candles/candlesticklistbuilder.h"
#include <QSslSocket>
#include <QGridLayout>
#include "api/apiservicerresponse.h"
#include "storage/database.h"

TradeWindow::TradeWindow(User *user, QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::TradeWindow) {
    ui->setupUi(this);
    this->user = user;
    setWindowTitle("Головне вікно");
    no_internet_connection = false;
    candle_graph = new CandleGraphBuilder();
    main_layout_diagram = new QBoxLayout(QBoxLayout::RightToLeft);
    initLoading();
    QSizePolicy::Expanding);
    ui->widget->setLayout(main_layout_diagram);
    setDefaultSettings();
    init_table_coins();
    initTableTradeHistory();
    update_balance_label();
    update_coins_balance_label();
    getPriceCurrentPair();
    connect(timer_refresh_chart, &QTimer::timeout, this,
    &TradeWindow::getLastCandle);
    connect(timer_refresh_price, &QTimer::timeout, this,
    &TradeWindow::getPriceCurrentPair);

```

```

        ui->lineEdit_count_buy->setValidator(new
QRegExpValidator(QRegExp("^\\d{1,3}(\\.\\d{0,2}[1-9])?$"), this));
    }
    void TradeWindow::hideEvent(QHideEvent *event) {
        Database::closeDatabase();
    }
    void TradeWindow::showEvent(QShowEvent *event) {
        Database::openDatabase();
        startAllRequests();
    }
    void TradeWindow::closeEvent(QCloseEvent *event) {
        Database::closeDatabase();
        stopAllRequests();
    }
    void TradeWindow::stopAllRequests() {
        timer_refresh_chart->stop();
        timer_refresh_price->stop();
        qDebug() << "Stop all requests!";
    }
    void TradeWindow::startAllRequests()
    {
        timer_refresh_price->start(1000);
        timer_refresh_chart->start(5000);}
    void TradeWindow::initTableTradeHistory() {
        model = new TableModelTradeHistory(this);
        model->setTable("TradeHistory");
        model->setFilter("name = '"+ user->discoverName() + "'");
        model->select();
        //model->setHeaderData(1, Qt::Horizontal, "Дія",
Qt::DisplayRole);
        model->setHeaderData(2, Qt::Horizontal, "Монета",
Qt::DisplayRole);
        model->setHeaderData(3, Qt::Horizontal, "Число",
Qt::DisplayRole);

```

```

        model->setHeaderData(4, Qt::Horizontal, "Ціна",
Qt::DisplayRole);
        model->setHeaderData(5, Qt::Horizontal, "Долари",
Qt::DisplayRole);
        ui->tableView_trade_history->setModel(model);
        ui->tableView_trade_history->setColumnHidden(0, true);
        ui->tableView_trade_history->setColumnHidden(1, true);
        ui->tableView_trade_history->verticalHeader()-
>setVisible(false);
        ui->tableView_trade_history->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);
    }
    void TradeWindow::refreshTableTradeHistory() {
        model->select();
    }
    TradeWindow::~~TradeWindow() {
        qDebug() << "destructor TradeWindow";
        delete diagram;
        delete main_layout_diagram;
        delete candle_graph;
        delete ui;
    }

    void TradeWindow::getSoundObj(Sound *sound) {
        this->sound = sound;
    }

    void TradeWindow::changeCryptoPair(QString coin) {
        QString pair = coin + "_USDT";
        current_coin = coin;
        setCurrentPair(pair);
    }

    void TradeWindow::init_table_coins() {
        QStringList list_coins = TXTReader::getListCryptocoins();
        int number_coins = list_coins.length();

```

```

        ui->table_coins->setColumnCount(number_coins);
        ui->table_coins-
>setSelectionMode(QAbstractItemView::SingleSelection);
        ui->table_coins->insertRow(0);
        for(int i = 0; i < list_coins.length(); i++) {
            ui->table_coins->setItem(0, i, new
QTableWidgetItem(list_coins.at(i)));
        }
        ui->table_coins->setEditTriggers(0);
        ui->table_coins->horizontalHeader()->hide();
        ui->table_coins->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);
        ui->table_coins->verticalHeader()->hide();
        for(int i = 0; i < number_coins; i++) {
            ui->table_coins->item(0, i)-
>setTextAlignment(Qt::AlignCenter);
        }
        ui->table_coins->item(0, 0)->setSelected(true);
        QTableWidgetItem item;
    }

void TradeWindow::setDefaultSettings() {
    Interval temp;
    temp = Interval::FIFTEEN_MINUTES;
    setCurrentInterval(temp);
    setCurrentPair("BTC_USDT");
    current_coin = "BTC";
    InitTimers();
    getChartGeneral();
}

void TradeWindow::update_balance_label() {
    ui->balance_label->setText("Баланс : " + QString::number(user-
>discoverBalance()));
}

void TradeWindow::update_coins_balance_label() {

```

```

        int current_column = ui->table_coins->currentColumn();
        if(current_column < 0) {
            current_column = 0;
        }
        QString cryptocurrency = ui->table_coins->item(0,
current_column)->text();
        ui->coins_balance_label->setText("Монеты: " +
QString::number(user->getNumberOfCoin(cryptocurrency)));
    }
    void TradeWindow::setCurrentInterval(Interval temp){
        interval = temp;
    }
    void TradeWindow::InitTimers() {
        timer_refresh_chart = new QTimer(this);
        timer_refresh_price = new QTimer(this);
        startAllRequests();
    }

```

Б.21 Файл loginwindow.h

```

#ifndef LOGINWINDOW_H
#define LOGINWINDOW_H
#include <QWidget>
#include "window/tradewindow.h"
#include "visitor/user.h"
#include <QMessageBox>
#include "utils/sound.h"
namespace Ui {
class LoginWindow;
}
class LoginWindow : public QWidget {
    Q_OBJECT
public:
    explicit LoginWindow(Sound *sound, QWidget *parent = nullptr);

```

```

    ~LoginWindow();
    void showDialogLoading();
signals:
    void sendSoundObj(Sound *sound);
private slots:
    void closeEvent(QCloseEvent *event) override;
    void on_btn_new_account_clicked();
    void on_btn_login_clicked();
private:
    Ui::LoginWindow *ui;
    TradeWindow *tradeWindow;
    User *user;
    Sound *sound;
    QWidget *loading_window;
};
#endif // LOGINWINDOW_H

```

Б.22 Файл loginwindow.cpp

```

#include "window/loginwindow.h"
#include "window/ui_loginwindow.h"
#include "storage/database.h"
LoginWindow::LoginWindow(Sound *sound, QWidget *parent) :
    QWidget(parent),
    ui(new Ui::LoginWindow) {
    ui->setupUi(this);
    setWindowTitle("Вхід");
    user = new User();
    this->sound = sound;
    Database::openDatabase();
}
LoginWindow::~LoginWindow() {
    qDebug() << "destructor LoginWindow";
    delete ui;
}

```

```

        delete tradeWindow;
    }

void LoginWindow::closeEvent(QCloseEvent *event) {
    Database::closeDatabase();
}

void LoginWindow::showDialogLoading() {
    QMovie *movie_loading = new QMovie("./images/loading.gif");
    movie_loading->setScaledSize(QSize(75, 75));
    loading_window = new QWidget();
    loading_window->setFixedSize(75, 75);
    QLabel *label_process = new QLabel(loading_window);
    label_process->setMovie(movie_loading);
    movie_loading->start();
    loading_window->setAttribute(Qt::WA_TranslucentBackground );
    loading_window->setWindowFlags(Qt::FramelessWindowHint |
Qt::WindowStaysOnTopHint);
    loading_window->setWindowTitle("Завантаження...");
    loading_window->show();
}

void LoginWindow::on_btn_login_clicked() {
    QString user_name = ui->lineEdit_username->text();
    if(user->userIsExists(user_name)) {
        sound->transitionOnAnotherWindow();
        user->setUsername(user_name);
        user->loadUserData();
        showDialogLoading();
        tradeWindow = new TradeWindow(user);
        connect(this, &LoginWindow::sendSoundObj,
tradeWindow, &TradeWindow::getSoundObj);
        emit sendSoundObj(sound);
        this->close();
        loading_window->close();
        tradeWindow->show();
    }
}

```

```

else {
    QMessageBox message;
    message.setText("Не знайдено користувача з таким ім'ям!");
    sound->error();
    message.exec();
}
}

```

Б.23 Файл piechartwindow.h

```

#ifndef PIECHARTWINDOW_H
#define PIECHARTWINDOW_H
#include <QMainWindow>
#include "piechart/drilldownchart.h"
#include "piechart/drilldownslice.h"
#include <QtCharts/QChartView>
#include "utils/sound.h"
namespace Ui {
class PiechartWindow;
}
class PiechartWindow : public QMainWindow {
    Q_OBJECT
public:
    explicit PiechartWindow(QWidget *parent = nullptr);
    ~PiechartWindow();
    void getCoins(QVector<QPair<QString, double>> coins);
    void closeEvent(QCloseEvent *event) override;
private:
    void initSettingsChart();
    void initSettingsPieSeries();
    void drawDiagram(QVector<QPair<QString, double>> coins);
private:
    Ui::PiechartWindow *ui;
    DrillDownChart *chart;

```

```

        QChartView *chartView;
        Sound *sound;
    };
#endif // PIECHARTWINDOW_H

```

Б.24 Файл piechartwindow.cpp

```

#include "window/piechartwindow.h"
#include "qdebug.h"
#include "window/ui_piechartwindow.h"
#include <QThread>
#include <QTimer>
#include <QtCharts/QLegend>
#include <QtCharts/QPieSeries>
#include <QtCore/QRandomGenerator>
PiechartWindow::PiechartWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::PiechartWindow) {
    ui->setupUi(this);
    setWindowTitle("Кругова діаграма");
    initSettingsChart();
    initSettingsPieSeries();
}
PiechartWindow::~PiechartWindow() {
    qDebug() <<"destructor PiechartWindow";
    delete ui;
}
void PiechartWindow::closeEvent(QCloseEvent *event) {
    sound->close();
}
void PiechartWindow::drawDiagram(
    QVector<QPair<QString, double>> coins) {
    coinSeries->clear();
    for(QPair<QString, double> coin : coins) {
        if(coin.second != 0) {

```

```

        *coinSeries << new
        DrillDownSlice(coin.second, coin.first, coinSeries);
    }
}
chart->changeSeries(coinSeries);
chartView = new QChartView(chart);
chartView->setRenderHint(QPainter::Antialiasing);
PiechartWindow::setCentralWidget(chartView);
}
void PiechartWindow::initSettingsChart() {
    chart = new DrillDownChart();
    chart->setTheme(QChart::ChartThemeLight);
    chart->setAnimationOptions(QChart::AllAnimations);
    chart->legend()->setVisible(true);
    chart->legend()->setAlignment(Qt::AlignRight);
}

```

Б.25 Файл `portfoliowindow.h`

```

#ifndef PORTFOLIOWINDOW_H
#define PORTFOLIOWINDOW_H
#include <QWidget>
#include "window/piechartwindow.h"
#include "visitor/user.h"
#include "utils/sound.h"
namespace Ui {
    class Portfoliowindow;
}
class Portfoliowindow : public QWidget
{
    Q_OBJECT
public:
    explicit Portfoliowindow(User *user, QWidget *parent =
nullptr);

```

```

~Portfoliowindow();
QVector <DatabaseCryptocoinsData> list_cryptocoins;
void getDataCryptocoinsFromAPI();
void getSoundObj(Sound *sound);
signals:
    void tradeWindowShow();
    void sendSoundObj(Sound *sound);

private slots:
    void on_tradeWindow_button_clicked();
    void showListCryptocoins();
    void initTablePortfolio();
    void initSettingTablePortfolio();
    void on_piechartWindow_button_clicked();
    void prepare_data_to_pieChart();
signals:
    void sendCoinsData(QVector<QPair<QString, double>> coins);
private:
    Ui::Portfoliowindow *ui;
    PiechartWindow *piechartWindow;
    User *user;
    Sound *sound;
};
#endif // PORTFOLIOWINDOW_H

```

Б.26 Файл portfoliowindow.cpp

```

#include "window/portfoliowindow.h"
#include "window/ui_portfoliowindow.h"
#include "api/apiaddressbuilder.h"
#include "api/apiservice.h"
#include "utils/txtreader.h"
Portfoliowindow::Portfoliowindow(User *user, QWidget *parent) :
    QWidget(parent),

```

```

        ui(new Ui::PortfolioWindow) {
            ui->setupUi(this);
            setWindowTitle("Портфолио");
            initSettingTablePortfolio();
        }
PortfolioWindow::~~PortfolioWindow() {
    delete ui;
}
void PortfolioWindow::showEvent(QShowEvent *event) {
    piechartWindow = new PiechartWindow();
    initDataCryptocoins();
}
void PortfolioWindow::getDataCryptocoinsFromAPI() {
    QString address = ApiAddressBuilder::getDataAllCryptoPairs();
    QJsonDocument json = ApiService::MakeRequest(address);
    QStringList list_coins = TXTReader::getListCryptocoins();
    QJsonArray arrayJsonCoins = json.array();
    list_cryptocoins.clear();
    foreach(const QJsonValue &coin, arrayJsonCoins) {
        for(int i = 0; i < list_coins.length(); i++) {
            if(coin.toObject().value("symbol") ==
                list_coins[i] + "_USDT") {
                DatabaseCryptocoinsData current_coin;
                current_coin.price_per_coin =
coin.toObject().value("price").toString().toDouble();
                current_coin.name =
coin.toObject().value("symbol").toString();
                current_coin.daily_change =
coin.toObject().value("dailyChange").toString().toDouble() * 100;
                list_cryptocoins.append(current_coin);
            }
        }
    }
}

```

```

void PortfolioWindow::initDataCryptocoins(){
    getDataCryptocoinsFromAPI();
    getDataCryptocoinsFromDatabase();
    initTablePortfolio();
}

void PortfolioWindow::getDataCryptocoinsFromDatabase()
{
    QVector<QPair<QString, double>> coinsDatabase = user-
>getCoinsList();
    for(int i = 0; i < list_cryptocoins.length(); i++) {
        for(int j = 0; j < coinsDatabase.length(); j++) {
            if(list_cryptocoins[i].name ==
coinsDatabase[j].first + "_USDT") {
                list_cryptocoins[i].name = coinsDatabase[j].first;
                list_cryptocoins[i].number_coins =
coinsDatabase[j].second;
            }
        }
    }
}

void PortfolioWindow::initSettingTablePortfolio() {
    ui->portfolioTableWidget->horizontalHeader()
->setSectionResizeMode(QHeaderView::Stretch);
    ui->portfolioTableWidget->
setEditTriggers(QAbstractItemView::NoEditTriggers);
    ui->portfolioTableWidget->setColumnCount(4);
    QStringList headers;
    headers << "Назва" << "Кількість" <<
"В доларах" << "Зміна за день %";
    ui->portfolioTableWidget->setHorizontalHeaderLabels(headers);
}

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна робота бакалавра
**РОЗРОБКА КРИПТОВАЛЮТНОЇ БІРЖІ З ДЕМО
РАХУНКОМ**
**DEVELOPMENT OF A CRYPTO EXCHANGE
WITH A DEMO ACCOUNT**

Виконав: ст. групи КНТ-139

Богдан ЗУЄВ

Керівник: доцент, к.т.н.

Сергій СЕРДЮК

2023

Рисунок В.1 – Слайд №1

Мета роботи, об'єкт та предмет дослідження

Об'єкт дослідження — процес розробки та впровадження криптовалютної біржі з демо рахунком.

Предмет дослідження — функціональні можливості криптовалютної біржі з віртуальним рахунком та методи взаємодії з користувачами.

Мета роботи — розробка комп'ютерного застосунку, що надає інструменти для підвищення практичних навичок та опанування стратегій торгування тими чи іншими активами кінцевого користувача.

Рисунок В.2 – Слайд №2

Завдання до роботи

- аналіз предметної області;
- аналіз програм аналогів;
- розробка технічного завдання;
- розробка моделей системи;
- проектування програмного забезпечення;
- реалізація програмного забезпечення;
- проведення тестування програми.

3

Рисунок В.3 – Слайд №3

Аналіз торгових платформ

Назва	Binance	MetaTrader 4	NinjaTrader 8
Інтерфейс (українська)	Так	Ні	Ні
Обов'язкова реєстрація	Так	Так	Так
Кількість торгових періодів	16	9	22
Демо рахунок	Так	Так	Так
Складні налаштування	Ні	Так	Так
Динамічний графік	Так	Так	Так
Перегляд портфелю	Так	Ні	Ні
Діаграми розподілу криптовалюти	Ні	Ні	Ні

4

Рисунок В.4 – Слайд №4

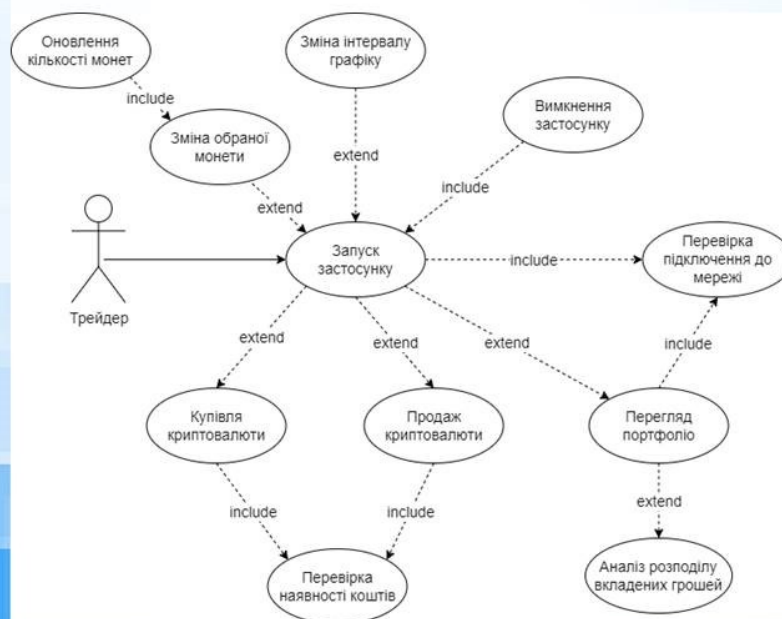
Аналіз API відомих криптобірж

Назва	Kucoin API	Poloniex API
Безкоштовне використання	Так	Так
Обмеження запитів	500 за 10 сек.	200 за сек.
Графіки зміни ціни	Так	Так
Кількість монет	понад 300	понад 600
Обов'язкове використання API ключа	Ні	Ні

5

Рисунок В.5 – Слайд №5

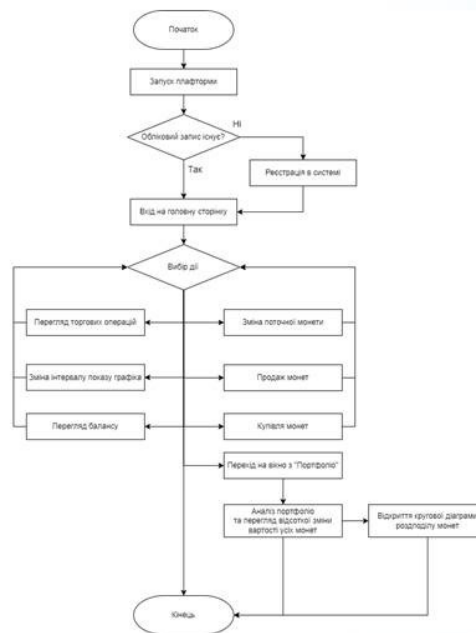
Діаграма прецедентів програмного забезпечення



6

Рисунок В.6 – Слайд №6

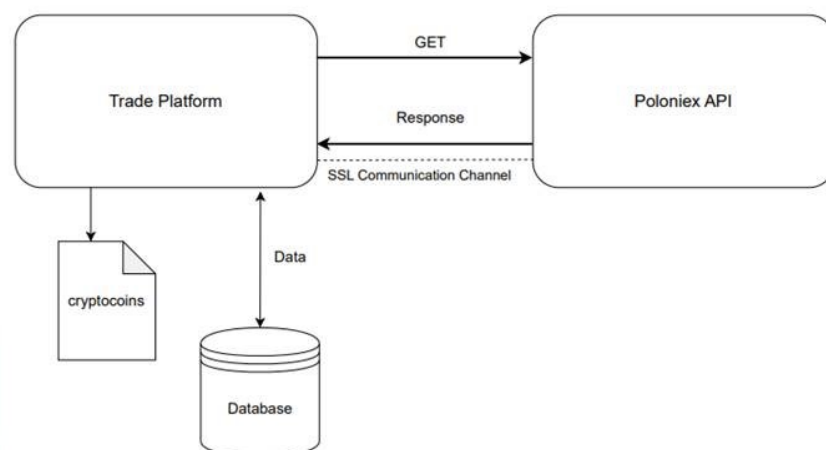
Схема алгоритму роботи програми



7

Рисунок В.7 – Слайд №7

Структурна схема компонентів системи



8

Рисунок В.8 – Слайд №8

```
classDiagram
    class User {
        text: name
        integer: total_money
    }
    class Portfolio {
        text: name
        text: cryptocurrency
        integer: number
    }
    class TradeHistory {
        text: name
        text: action
        text: cryptocurrency
        real: number
        real: price
        real: usd
    }
    User "1" -- "0..n" Portfolio
    User "1" -- "0..n" TradeHistory
```

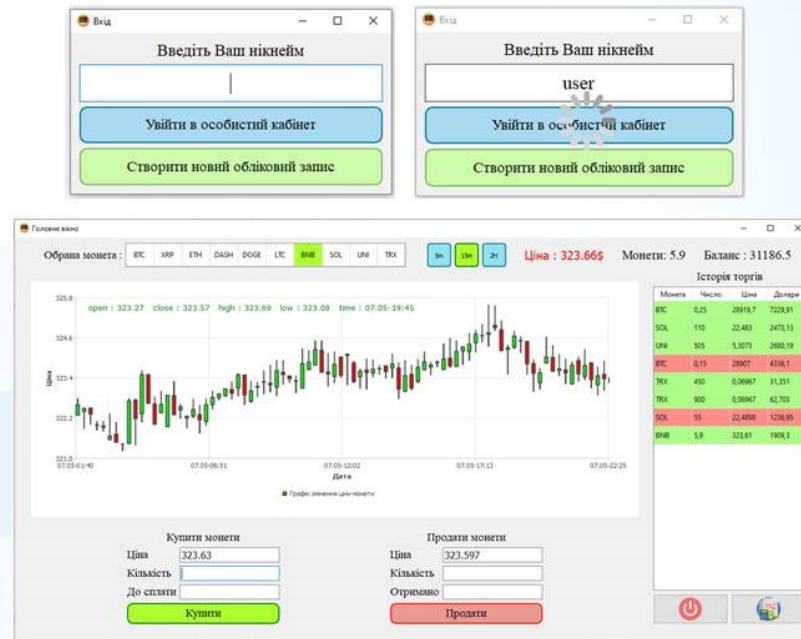
The diagram illustrates a database schema with three classes: User, Portfolio, and TradeHistory. The User class is connected to both Portfolio and TradeHistory via one-to-many relationships. The User class has attributes: text: name (primary key) and integer: total_money. The Portfolio class has attributes: text: name (primary key), text: cryptocurrency, and integer: number. The TradeHistory class has attributes: text: name (primary key), text: action, text: cryptocurrency, real: number, real: price, and real: usd. The relationships are indicated by lines with crow's foot notation: a line from User to Portfolio with a '1' at the User end and a '0..n' at the Portfolio end, and a line from User to TradeHistory with a '1' at the User end and a '0..n' at the TradeHistory end.

Рисунок В.9 – Слайд №9

[illegible]

Рисунок В.10 – Слайд №10

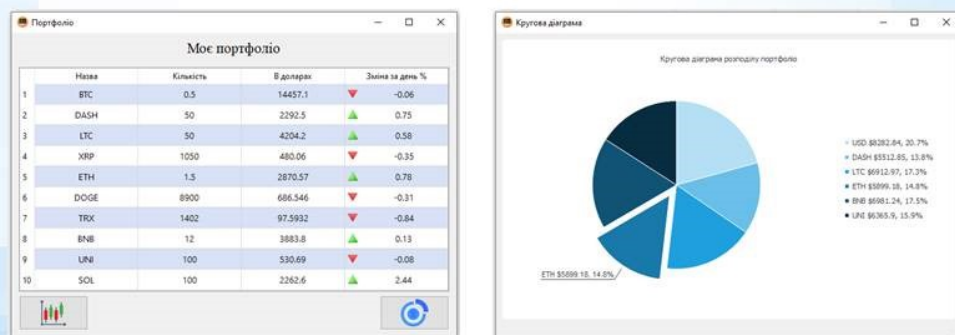
Форми входу та торгівлі



11

Рисунок В.11 – Слайд №11

Форми відображення портфолію



Табличне представлення

Секторна діаграма

12

Рисунок В.12 – Слайд №12

Висновки

- проведено аналіз предметної області;
- проведено аналіз програмних аналогів;
- розроблено технічне завдання;
- розроблено моделі взаємодії користувача з системою;
- виконано проєктування системи;
- програмно реалізовано застосунок;
- виконано тестування програми.

13

Рисунок В.13 – Слайд №13

Дякую за увагу!

14

Рисунок В.14 – Слайд №14