Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

**Кваліфікаційна робота бакалавра**

на тему : «Розробка програмного продукту для моніторингу криптовалют»

Виконав: студент групи ІПЗ20-1
Спеціальність 121 «Інженерія програмного забезпечення»
Суслов Давід Валерійович
(прізвище та ініціали)
Керівник к.ф.-м.н., доцент Лебідь О. Ю.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
в.о. завідувача кафедри кібербезпеки та інформаційних технологій
(посада)
к.т.н., доцент кафедри кібербезпеки та інформаційних технологій Прокопович - Ткаченко Д.І.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

# АНОТАЦІЯ

*Суслов Д.В.* Розробка програмного продукту для моніторингу криптовалют.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2024.

Дана кваліфікаційна робота присвячена розробці програмного забезпечення для моніторингу криптовалют. Криптовалюти стали важливим інструментом у сучасній економіці. Їхня роль значно зросла завдяки розвитку інформаційних технологій та глобальної діджиталізації. Ринок криптовалют привертає увагу як професійних трейдерів, так і початківців, що створює попит на ефективні засоби моніторингу, аналізу та прогнозування.

Розробка застосунку проводилась з використанням сучасних технологій програмування таких як .NET 6, WPF, CoinGecko API. Створений програмний продукт реалізує такі основні функції як перегляд курсів криптовалют, автоматичне побудова графіків, отримання прогнозів та інформаційних матеріалів, підбір кращих курсів для купівлі та продажу. Додаток забезпечує зручний інтерфейс, швидку обробку даних та доступ до інформації в режимі реального часу, що сприяє обґрунтованому прийняттю інвестиційних рішень.

Розроблене програмне забезпечення надає користувачам можливість переглядати історичні дані про курси криптовалют, фільтрувати та сортувати дані за різними критеріями, а також отримувати розсилку з прогнозами та інформацією про зміни на ринку. Це дозволяє користувачам зменшувати ризики та полегшувати процес торгів на криптовалютних біржах.

Ключові слова: програмне забезпечення, криптовалюта, криптобіржі, WPF, .NET, CoinMarketCap API

# ABSTRACT

*Suslov D.V.* Development of a Software Product for Cryptocurrency Monitoring. Qualification work for obtaining a bachelor's degree in the specialty 121 «Software engineering». – University of Customs and Finance, Dnipro, 2024.

This bachelor's qualification work is dedicated to the development of a software product for cryptocurrency monitoring. Cryptocurrencies have become an important tool in the modern economy. Their role has significantly increased due to the development of information technologies and global digitalization. The cryptocurrency market attracts the attention of both professional traders and beginners, creating a demand for effective monitoring, analysis, and forecasting tools.

The application development was carried out using modern programming technologies such as .NET 6, WPF, CoinGecko API. The created software product implements such basic functions as viewing cryptocurrency rates, automatic chart building, obtaining forecasts and informational materials, selecting the best rates for buying and selling. The application provides a user-friendly interface, fast data processing, and real-time information access, which contributes to well-founded investment decisions.

The developed software tool enables users to view historical data on cryptocurrency rates, filter and sort data by various criteria, and receive newsletters with forecasts and market updates. This allows users to reduce risks and facilitate the trading process on cryptocurrency exchanges.

Keywords: Software, cryptocurrency, crypto exchanges, WPF, .NET, CoinMarketCap API.

ЗМІСТ

# ВСТУП

*Актуальність теми.* Гроші завжди мали вирішальне значення в суспільному житті сприяючи комерційним та торговельним обмінам. З розвитком інформаційних технологій з'явились нові фінансові інструменти – криптовалюти, що базуються на блокчейн технології. Від часу своєї появи, криптовалютний ринок став магнітом для професійних інвесторів та новачків, які прагнуть знайти ефективні методики прогнозування. Ринок вимагає глибокого аналізу та розуміння його унікальних аспектів з метою успішного інвестування.

Економічна діяльність ґрунтується не лише на реальних даних, а й на прогнозах, що включають політичні, технологічні зміни, аграрний сектор, погоду та добувну промисловість, а також ринкові вартості активів і валют. Для роботи на ринку необхідно мати доступ до найактуальнішої інформації та точних прогнозів, що допоможуть передбачити майбутні тренди та вибрати вигідні стратегії.

У XXI столітті, еру інформаційних технологій, виникнення криптовалютних бірж стало логічним кроком. Сьогодні існує понад 2500 криптовалют, сім з яких володіють мільярдними капіталізаціями. Все більше компаній і фінансових установ інтегрують криптовалюти у свої бізнес-процеси, що сприяє їх подальшому поширенню та розвитку. З кожним роком з’являються нові криптовалюти та вдосконалюються технології, що лежать в їх основі, роблячи їх більш доступними і функціональними для широкого кола користувачів. Станом на квітень 2024 року, загальна вартість ринку криптовалют досягла 2,51 трильйонів доларів США. Цей показник зазнав значного зростання з часу, коли вперше перевищив 1 мільярд доларів у 2013 році, і досяг майже 3 трильйонів наприкінці 2021 року.

Незважаючи на скепсис деяких фахівців, вони пропонують значні можливості для інвестицій. Існують два основні типи торгівлі криптовалютами:

1) криптообмінники – це компанія, що виступає як контрагент;

2) криптовалютні біржі – платформи, що дозволяють користувачам торгувати між собою.

Тому доступ до своєчасної інформації про курси криптовалют є вкрай важливим.

Враховуючи актуальність питання яке розглядається, використання сучасних технологій програмування для прогнозування та моніторингу курсу криптовалют є актуальним. Серед відомих систем можна виділити Trader, яка є системою, яка дозволяє прогнозувати коливання валютних курсів на ринку Forex; Belinvestor — це веб-додаток, який надає прогнози криптовалют у формі новинної стрічки; Walletinvestor — це веб-додаток, що дозволяє вибрати конкретну криптовалюту для відстеження або перегляду інформації про всі доступні криптовалюти у вигляді таблиці; Belinvestor — це веб-додаток, який надає прогнози криптовалют у формі новинної стрічки; NeuroShell - це комплексний набір нейронних мереж, розроблених спеціально для прогнозування валютних курсів на фінансових ринках; Elliott Wave Analyser Professional 6.0 — це програмний продукт, призначений для аналізу валютного ринку з використанням теорії хвиль Елліотта (ТХЕ) та стандартних алгоритмів технічного аналізу тощо. Але вони мають свої обмеження та недоліки і здебільшого є платними. Саме тому тема кваліфікаційної роботи, яка полягає в розробці власного програмного забезбечення для моніторингу криптовалют є актуальною.

*Метою роботи* є автоматизація процесу збору актуальної інформації курсу криптовалюти.

У відповідності до мети роботи в кваліфікаційній роботі ставились та вирішувались *наступні завдання*:

1) проаналізувати сучасний стан криптовалют та засади їх функціонування;

2) розглянути принципи роботи криптовалют черз криптовалютні біржі;

3) дослідити програмні реалізації збору інформації про криптовалюти;

4) проаналізувати засоби збору даних;

5) розробити структуру додатку;

6) розробити програмний застосунок;

7) провести тестування розробленого програмного забезпечення.

*Методи та технології дослідження* – методи аналізу, синтезу, узагальнення, технології розробки Desktop-додатків, проектування інтерфейсів.

*Об'єкт дослідження* – розробка програмного забезпечення для збору та агрегації даних про курс криптовалюти.

*Предмет дослідження* – розробка Desktop-додатку для збору даних про курс криптовалюти.

*Структура роботи* - робота складається з вступу, трьох розділів, списку використаних джерел з 20 найменувань, 21 рисунків

# РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1. Криптовалюта та її використання в світі

Криптовалюти — цифрові валюти, які використовують криптографічні методи для забезпечення безпеки. Ці активи зазвичай використовуються для купівлі товарів і послуг, деякі з них мають унікальні правила для своїх користувачів. Вони поза контролем центральних установ і досі не прийняті як законний платіжний засіб.

Станом на сьогодні, понад 100 мільйонів людей у світі користуються криптовалютами, головним чином з інвестиційною метою, попит на криптовалюти не дуже високий порівняно з національними валютами, які добре виконують свої функції.

Перш за все криптовалюти забезпечують здійснення онлайн-покупок без необхідності вказувати особисті дані, хоча це не гарантує повної анонімності. Користувачі мають анонімні псевдоніми, які можуть бути відстежені владними органами. В контексті зростаючих занепокоєнь щодо безпеки особистих даних, криптовалюти можуть пропонувати певні переваги для збереження конфіденційності.

Друга велика перевага полягає в усуненні фінансових посередників, що знижує транзакційні витрати. Що значно зменшує комісії або взагалі може бути відсутня  це спонукає підприємців обрати криптовалюти як альтернативу традиційним фінансовим системам.

Третє деякі криптовалюти можуть пропонувати додаткові привілеї, такі як обмежені права власності або голосування у керівництві організації, яка фінансується через криптовалюту, або права на частку у фізичних активах, таких як нерухомість або твори мистецтва [1].

## 1.2. Загальна характеристика криптовалюти

Обіг криптовалюти є децентралізованим і не підпорядковується жодній центральній установі. Вони функціонують через розподілену мережу комп'ютерів, дозволяючи здійснювати торгівлю на спеціалізованих біржах та зберігати криптовалютні активи в цифрових гаманцях. Технологія блокчейн, яка є фундаментом більшості криптовалют, представляє собою відкритий реєстр транзакцій, що постійно оновлюється. Вона дозволяє покупцям та продавцям безпосередньо взаємодіяти та забезпечує доступ до записів транзакцій без втручання центральних авторитетів. Кожен блок у ланцюзі містить посилання на попередній блок за допомогою хеш-коду, створеного за допомогою криптографічного алгоритму.

Чеський дослідник Ян Ланскі визначає криптовалюту як систему, яка повинна відповідати шести критеріям:

1. Відсутність необхідності в центральному управлінні; стабільність забезпечується за допомогою децентралізованого консенсусу.
2. Запис та реєстрація криптовалютних одиниць та прав власності на них.
3. Правила створення нових одиниць криптовалюти, їх умови та методи визначення власників.
4. Можливість доведення власності на одиниці лише з використанням криптографічних методів.
5. Виконання транзакцій, що змінюють власника криптовалютних одиниць, з ініціативи поточного власника.
6. У разі подвійної інструкції про передачу одних і тих же одиниць система реалізує лише одну з них.

Концепцію криптовалюти започаткував Девід Чаум у 1983 році зі своєю ідеєю криптографічних електронних грошей "ecash". Внесок також зробили Вей Дай, Нік Сабо та Адам Бек, розвиваючи основні принципи технології блокчейн, які лягли в основу всіх сучасних криптовалют. Біткоїн, створений у 2008-2009

роках особою або групою під псевдонімом Сатосі Накамото, став першою широко відомою криптовалютою. Його створення було значним кроком вперед у розвитку цифрових фінансових інструментів. Завдяки науково-технічному прогресу та швидкій цифровізації, криптовалюти почали відігравати важливу роль у сучасній економіці.

Однією з ключових подій в історії криптовалют став бум майнінгу в 2018-2020 роках, коли видобуток криптовалют став надзвичайно популярним. Пандемія COVID-19 також мала значний вплив на розвиток криптовалют, оскільки економічна нестабільність підштовхнула багатьох інвесторів до пошуку альтернативних фінансових інструментів [3].

### 1.3. Принцип роботи криптовалюти. Переваги та недоліки

Більшість криптовалют функціонують на основі блокчейну-технології, що забезпечує безпечний і децентралізований механізм проведення транзакцій. У такій системі учасники здійснюють обмін напряму без участі посередників. Інформація про усі транзакції накопичується в блокчейні – базі даних, що регулярно оновлюється. Кожен користувач має унікальний криптографічний ключ, який забезпечує безпеку його транзакцій і дає контроль над його цифровими активами. Згоду сторін є обов'язковою для внесення транзакцій у блокчейн і змінити вже записану інформацію неможливо, оскільки кожен блок криптографічно зв'язаний з попереднім [4].

Нові блоки додаються до ланцюга відповідно до принципу консенсусу, коли більшість учасників мережі має погодитися на додавання транзакцій.

Основні можливості криптовалют включають:

1. Зберігання – можливість тримати криптовалюту у цифрових гаманцях або на криптобіржах.
2. Обмін – можна обмінювати криптовалюти між собою або на фіатні гроші через криптобіржі.

3. Транзакції – використання криптовалюти для різноманітних платежів та переказів коштів.

4. Майнінг – процес генерації нових блоків та верифікації транзакцій в блокчейні, за що майнери отримують винагороду у вигляді криптовалюти [2].

Переваги блокчейну та криптовалют включають децентралізацію, прозорість, безпеку та низькі комісії. Однак існують ризики, пов'язані з волатильністю і спекулятивним характером криптовалют, на треба звернути увагу перед інвестуванням.

## 1.4. Види та найпопулярніші криптовалюти

Станом на квітень 2024 року, загальна вартість ринку криптовалют досягла 2,51 трильйонів доларів США. Цей показник зазнав значного зростання з часу, коли вперше перевищив 1 мільярд доларів у 2013 році, і досяг майже 3 трильйонів наприкінці 2021 року. Основну частку ринку, приблизно 85%, займають десять найпопулярніших криптовалют. Ось декілька з них з їхніми основними особливостями:

1) Bitcoin (BTC):

- Капіталізація: ≈ $1,3 трлн
- Перша та найвідоміша криптовалюта (2008)
- Обмежена пропозиція: 21 мільйон монет (19 мільйонів вже видобуто)
- Стійка до інфляції, дефляційна
- Майнінг займає близько 10 хвилин на блок [8]

2) Ethereum (ETH):

- Капіталізація: ≈ $342,64 млрд
- Платформа (2015) з підтримкою смарт-контрактів
- Смарт-контракти: автоматизація процесів, DApps
- Програмування на Solidity
- Час обробки блоку: 2,5 хвилини

3) Tether (USDT):

- Капіталізація: ≈ $161 млрд
- Стейблкоїн, прив'язаний до долара США (1:1)
- Заснований у 2014 році для стабільності та ліквідності
- Працює на блокчейнах Bitcoin, Ethereum, Tron
- Tether Limited стверджує про 1 USD резерву на кожен USDT

4) BNB (Binance Coin):

- Капіталізація: ≈ $67,7 млрд
- Криптовалюта біржі Binance (2017)
- Спочатку токен Ethereum, потім перехід на Binance Smart Chain
- Знижки та бонуси на Binance для власників BNB
- Періодичне спалювання токенів для зменшення пропозиції

5) USD Coin (USDC):

- Капіталізація: ≈ $52 млрд
- Стейблкоїн, прив'язаний до долара США (1:1)
- Заснований у 2018 році Circle та Coinbase
- Працює на Ethereum, Algorand, Solana
- Circle стверджує про резерв у доларах США або "схвалені інвестиції" на кожен USDC [6, 7]

## 1.5. Програмні застосунки для прогнозування та моніторингу курсу криптовалют

Trader є системою, яка дозволяє прогнозувати коливання валютних курсів на ринку Forex. Система аналізує попередні дані торгівлі, включаючи максимальні, мінімальні ціни, ціну закриття та обсяги угод за день. Аналіз базується на застосуванні різноманітних алгоритмів, зокрема трьох видів ковзних середніх (лінійне, експоненціальне, і з ваговими коефіцієнтами), MACD-гістограм, а також таких індикаторів, як RSI, OBV, Williams R%, CandleSticks та Point & Figure. Користувачі мають змогу створювати власні формули для аналізу,

а також застосовувати одні індикатори до інших, що є корисним при розрахунку MACD-гістограм. Однак система має недоліки, такі як неоптимальна зручність для користувачів [5].

Walletinvestor — це веб-додаток, що дозволяє вибрати конкретну криптовалюту для відстеження або перегляду інформації про всі доступні криптовалюти у вигляді таблиці. Додаток пропонує прогнози на різні періоди, від двох тижнів до п'яти років. Відвідувачі сайту можуть також ознайомитись з поточними курсами популярних криптовалют, переглядати історичні дані, лінки на біржі, прогнози від інших компаній, та навіть здійснити покупку криптовалюти через сайт. Методи, які використовуються для створення прогнозів, не розкриваються через комерційну таємницю (рис. 1.1) [10].

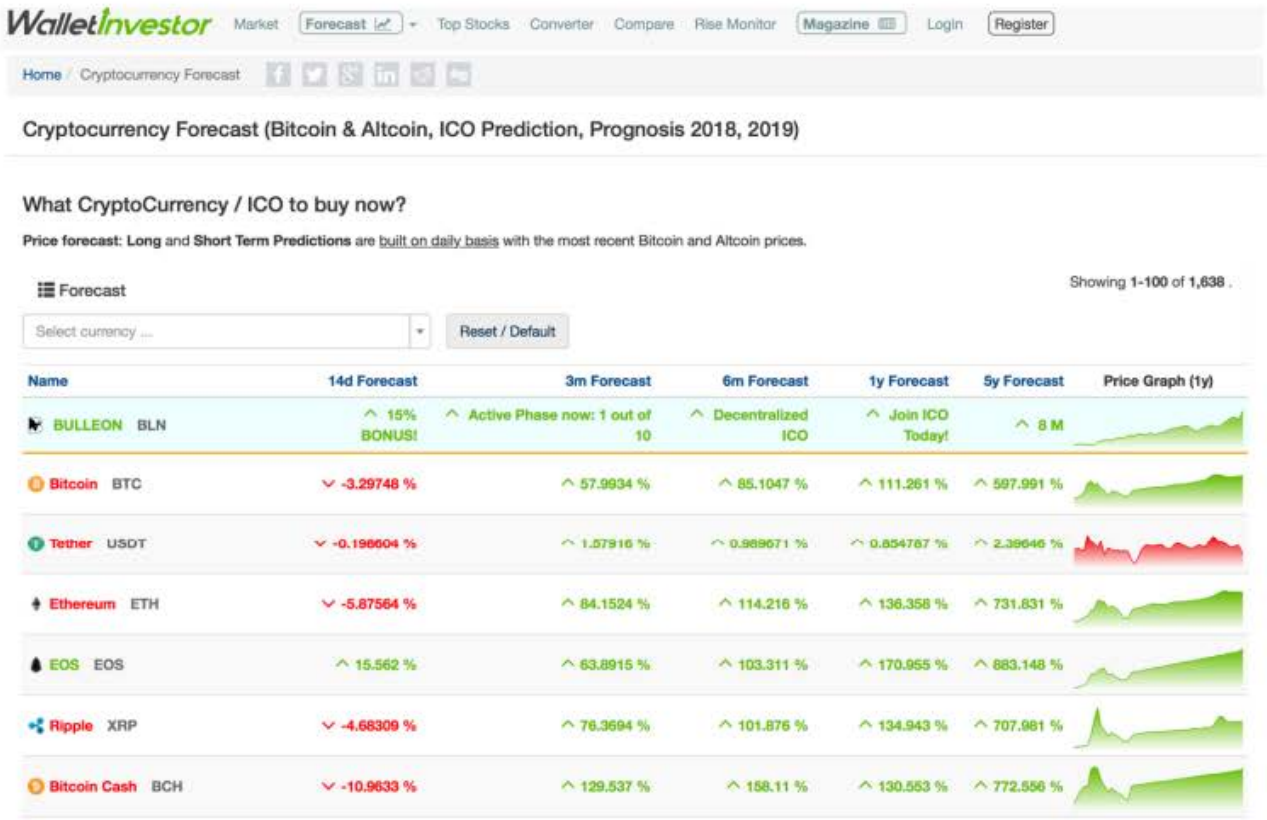



Рисунок 1.1 – Walletinvestor

Belinvestor — це веб-додаток, який надає прогнози криптовалют у формі новинної стрічки. Особливість сайту полягає в тому, що пости старіші за три

місяці автоматично видаляються, що унеможливлює перегляд старих прогнозів та оцінку їх точності. Нові прогнози створюються досвідченими фахівцями, але їх ефективність залишається низькою через передбачуваність тенденцій, які вони висвітлюють (рис 1.2) [11].

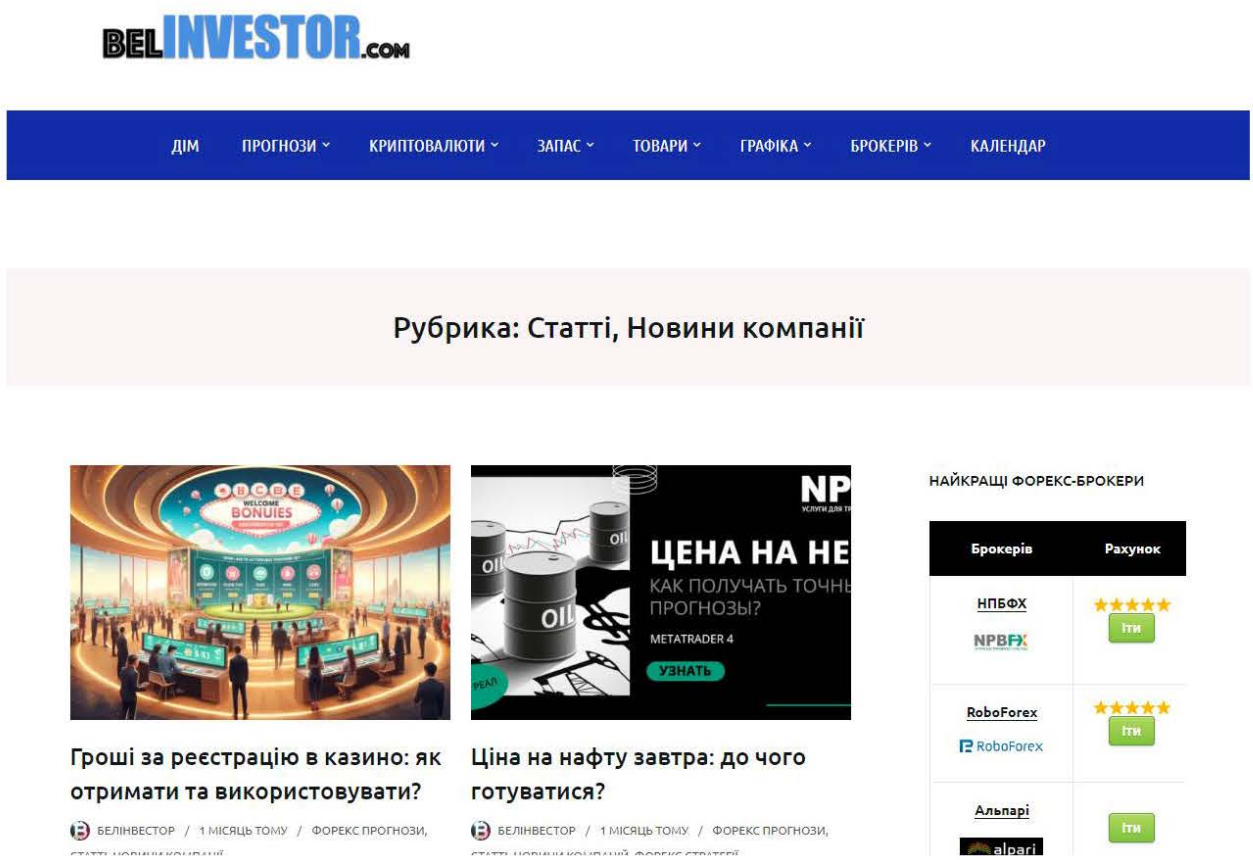



Рисунок 1.2 – Belinvestor

Кожен прогноз представлений у вигляді окремої статті, яка містить історію конкретної криптовалюти та аналіз чинників, які можуть вплинути на її курс. В основному, прогнози базуються на фундаментальному аналізі, виконаному командою професіоналів. Деталі методів, які використовують експерти для створення прогнозів, на сайті не розкриваються (рис. 1.3).


Рисунок 1.3 – Приклад прогнозу на сайті Belinvestor

NeuroShell - це комплексний набір нейронних мереж, розроблених спеціально для прогнозування валютних курсів на фінансових ринках. Основний принцип роботи цієї системи базується на концепції "чорної скриньки". Це означає, що користувачам не потрібно знати внутрішні алгоритми або деталі обробки даних, щоб користуватися програмою. Завдяки мінімалістичному інтерфейсу, NeuroShell є доступним і зрозумілим для користувачів з різним рівнем досвіду у фінансових операціях. Користувачі можуть легко налаштовувати параметри та отримувати прогнозні результати (рис. 1.4) [12].

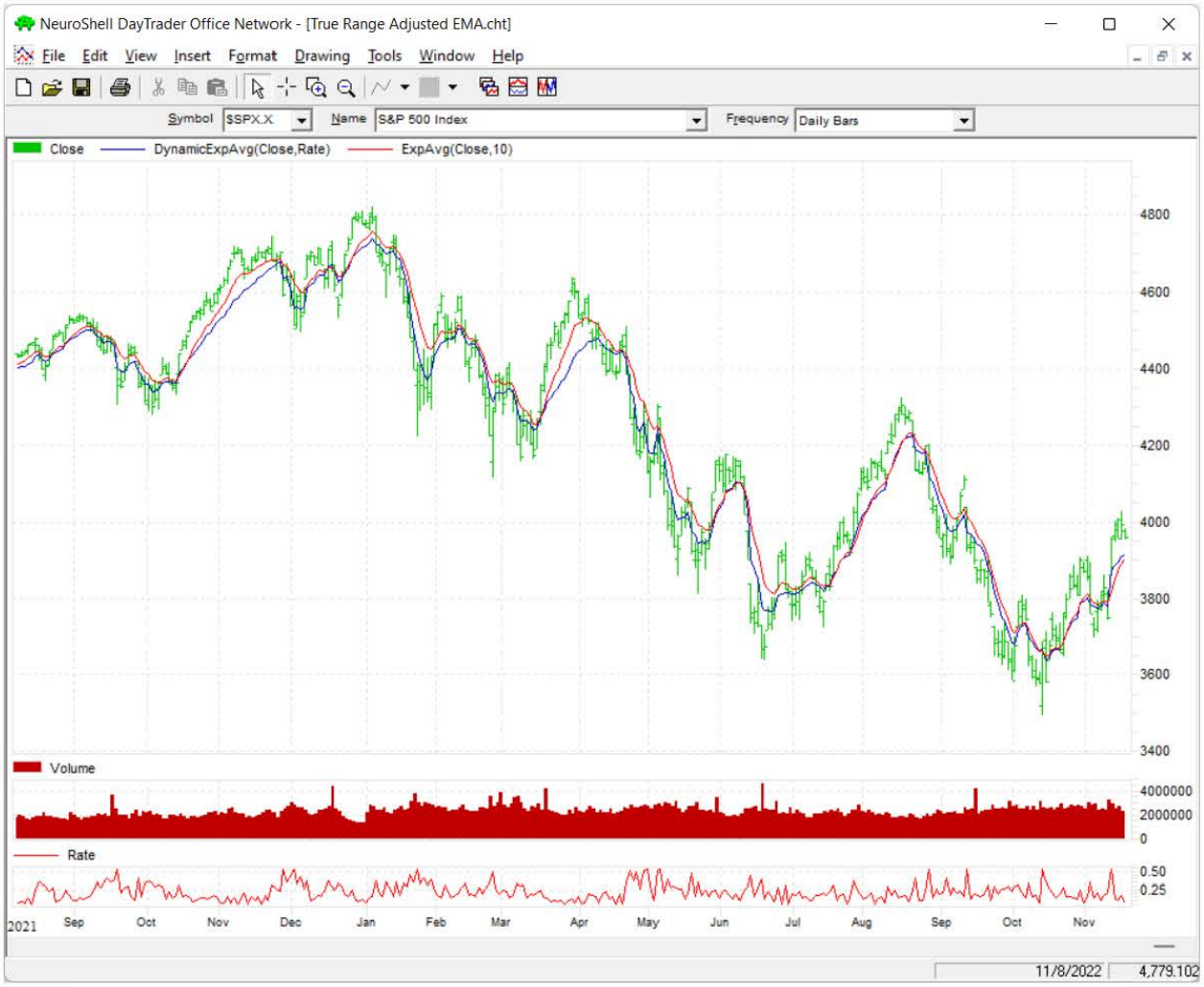



Рисунок 1.4 – Приклад прогнозу за допомогою додатку NeuroShell Day Trader

Однією з ключових переваг NeuroShell Day Trader є впровадження оптимізаційних методів, заснованих на принципах генетичних алгоритмів. Цей підхід значно покращує процес підготовки нейронних мереж, забезпечуючи вибір оптимальних параметрів для індикаторів та обробки даних з різних входів мережі. Генетичні алгоритми допомагають знаходити найкращі рішення для налаштування мереж, що в свою чергу підвищує точність прогнозування.

Основною архітектурою, яка використовується в NeuroShell Day Trader, є багатошаровий персептрон. Цей вид нейронної мережі складається з кількох шарів нейронів, що дозволяє моделі ефективно обробляти та аналізувати складні фінансові дані. Багатошаровий персептрон здатний навчатися та робити

прогнози, використовуючи різноманітні індикатори та вхідні дані, що забезпечує гнучкість та точність торгової системи.

NeuroShell Day Trader зосереджений на побудові ефективної торгової системи, яка може використовувати як традиційні індикатори, так і прогнозовані значення, отримані за допомогою нейронних мереж. Це дозволяє трейдерам приймати більш обґрунтовані рішення на основі складного аналізу та прогнозування ринкових умов. Завдяки цьому підходу користувачі можуть поліпшити свої торгові стратегії та досягати кращих результатів на фінансових ринках.

Elliott Wave Analyser Professional 6.0 — це програмний продукт, призначений для аналізу валютного ринку з використанням теорії хвиль Елліотта (ТХЕ) та стандартних алгоритмів технічного аналізу. У 1930 році Ральф Елліотт виявив, що емоційний стан натовпу впливає на валютні курси, і цей вплив описується певними зразками, відомими як хвилі Елліотта (рис. 1.5) [13].

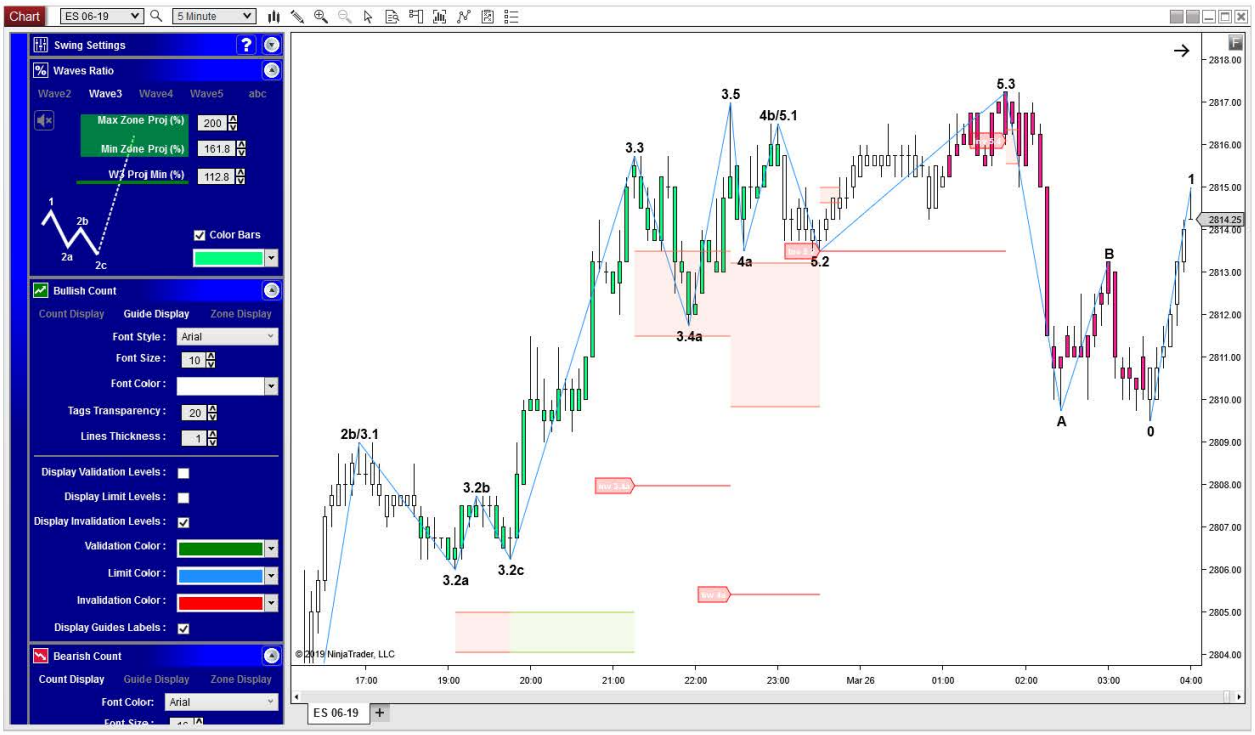



Рисунок 1.5 – Приклад прогнозу за допомогою додатку Elliott Wave Analyser Professional

Програма Elliott Wave Analyser Professional дозволяє виділяти незакінчені зразки з часових рядів, які можуть відповідати хвилям Елліотта. З огляду на вивчену поведінку стандартних хвиль Елліотта, можна прогнозувати подальший розвиток цих незакінчених зразків. Для кожного зразка система обчислює коефіцієнт Goodness, який визначає ступінь відповідності зразка теоретичному аналогу. Важливим аспектом аналізу є кількість міток, які апроксимують досліджуваний зразок. Користувач може налаштовувати щільність розподілу цих міток.

Після аналізу кожної хвилі система надає сигнали входу або виходу з ринку для коротких або довгих позицій. Дані про валютні курси можна об'єднувати в групи, де валюти з сигналами входу або виходу виділяються кольором. Користувачі також можуть налаштовувати параметри критеріїв виходу і входу.

У режимі онлайн програма автоматично перераховує хвилі з періодичністю, яку задає користувач. Користувач також може написати власний алгоритм індикатора, який буде відображатися на тому ж графіку, що і вихідні дані. Програма надає детальну довідкову систему з теоретичним описом принципу Елліотта та розділом Guided Tour, який проводить користувача через всі етапи аналізу.

## 1.6. Висновок до першого розділу

Підсумовуючи можна говорити про те що криптовалюти стають все більш популярними, що підтверджується їхнім значним збільшенням ринкової капіталізації. Серед найвідоміших представників цього ринку можна виділити Bitcoin, Ethereum, Tether, кожен з яких володіє унікальними особливостями та областями застосування.

Технологія блокчейн, що є основою для більшості криптовалют, гарантує безпеку та прозорість у проведенні транзакцій, дозволяючи відслідковувати їх у реальному часі. Ця технологія характеризується децентралізацією, що виключає

контроль з боку центральних установ, забезпечуючи користувачам зниження витрат на транзакції та більшу конфіденційність.

Однак, криптовалюти не позбавлені недоліків як-от висока волатильність та ризики, що виникають через їхній спекулятивний характер. Інвестування в криптовалюти вимагає грунтовного вивчення ринку та усвідомлення потенційних ризиків.

Також розглянуті існуючі програмні програмні для прогнозування та моніторингу курсу криптовалют, які допомагають користувачам аналізувати ринок та робити обґрунтовані інвестиційні рішення. Важливо враховувати обмеження та недоліки цих інструментів.

Отже, криптовалютна сфера є складною та динамічною, вимагаючи глибокого розуміння технічних, економічних та юридичних аспектів для їх ефективного використання і інвестування.

# РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ТА МЕТОДІВ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1. Засоби Desktop-розробки

Desktop-розробка є ключовою галуззю програмного забезпечення, що охоплює створення додатків для настільних комп'ютерів. Ці програми працюють безпосередньо на операційних системах, таких як Windows, macOS або Linux, і забезпечують користувачів зручними інтерфейсами для вирішення різноманітних завдань.

Фреймворки що використовуються:

- Electron.js – дозволяє створювати кросплатформні додатки за допомогою HTML, CSS та JavaScript.
- Qt – фреймворк для створення кросплатформних додатків на C++.
- GTK – використовується для розробки додатків під Linux.
- WPF – використовується для створення застосунків для Windows.
- JavaFX – фреймворк для створення кросплатформних додатків на Java.

### 2.1.1. Windows Presentation Foundation

Windows Presentation Foundation (WPF) – це високофункціональний графічний фреймворк від Microsoft, спеціалізований на створенні настільних застосунків для Windows. Запущений у 2006 році як частина .NET Framework 3.0. WPF забезпечує розробникам комплекс інструментів для реалізації складних користувацьких інтерфейсів, анімацій, зв'язування даних та багато іншого.

WPF використовує мову розмітки XAML для опису користувацьких інтерфейсів дозволяючи розробникам виконувати роботу в декларативному стилі. Це полегшує процес розробки та взаємодію між розробниками і дизайнерами, дозволяючи застосовувати стилі та шаблони до елементів

інтерфейсу для налаштування їхнього зовнішнього вигляду без зміни функціональності, сприяючи створенню уніфікованих і добре структурованих інтерфейсів. Візуальні елементи задаються через XAML, тоді як бізнес-логіка програми розробляється на C#.

Одна з найпотужніших функцій WPF – це система зв'язування даних, яка дозволяє легко синхронізувати інтерфейс користувача з даними моделі або ViewModel. Це забезпечує гнучкість та полегшує розробку додатків, де зміни в даних автоматично відображаються в інтерфейсі [16].

WPF інтегрує потужні можливості DirectX для рендерингу з розширеними функціями для роботи з мультимедіа та анімацією, роблячи його ідеальним інструментом для створення динамічних інтерфейсів користувачів.

Переваги:

1. Гнучкість дизайну – WPF сприяє розробці індивідуальних і комплексних користувацьких інтерфейсів завдяки стилям, шаблонам та анімаціям.

2. Шкальованість – векторна графіка забезпечує чітке зображення інтерфейсу на екранах з будь-яким розширенням.

3. Можливість повторного використання – стилі, шаблони та ресурси можуть бути реалізовані один раз і використані у багатьох частинах застосунку.

4. Інтеграція з іншими технологіями .NET

Недоліки:

1. Продуктивність – на слабких системах WPF може споживати значну кількість ресурсів через свою графічну складність.

2. Складність засвоєння технології – опанування технології WPF може бути складним для новачків через його широку функціональність та гнучкість.

3. Підтримка тільки Windows – WPF призначений виключно для розробки додатків під Windows, що обмежує його використання для крос-платформених рішень.

WPF є потужним інструментом для розробки настільних застосунків з багатим і динамічним інтерфейсом користувача. Завдяки своїм широким

можливостям у сфері графіки, анімації та зв'язування даних, WPF дозволяє створювати сучасні та функціональні додатки. Однак, його використання обмежується платформою Windows, а процес навчання може бути складним для початківців. Незважаючи на це, для тих, хто працює виключно в середовищі Windows, WPF пропонує чудовий інструментарій для створення високоякісних додатків.

### 2.1.2. Платформа розробки .Net Framework

.NET Framework – це програмна платформа від Microsoft, що використовується для створення широкого спектру додатків, від веб-сайтів до складних корпоративних систем. Вона надає розробникам інструменти та бібліотеки, які значно спрощують процес розробки.

В основі .NET Framework лежать два ключових компоненти:

- Середовище виконання загальної мови (CLR) – відповідає за виконання програм, забезпечуючи автоматичне керування пам'яттю, безпеку коду та обробку винятків.
- Бібліотека класів (FCL) – надає широкий спектр готових компонентів для роботи з колекціями, файлами, мережею, базами даних та іншими функціональними можливостями.

.NET Framework має ряд суттєвих переваг, що робить її популярною серед розробників:

- Підтримка багатьох мов програмування

C#, VB.NET, F# та інші. Це дає можливість вибирати мову, яка найкраще підходить для конкретного завдання, та полегшує інтеграцію різних компонентів системи.

- Кросплатформенність

Завдяки .NET Core з'явилася можливість створювати додатки, які можуть працювати на Windows, Linux та macOS.

- Розширювана екосистема

.NET Framework має широкий спектр інструментів та бібліотек, що постійно розширюються та підтримуються спільнотою.

- Інтеграція з продуктами Microsoft

.NET Framework тісно інтегрується з іншими продуктами Microsoft, такими як Azure, SQL Server та Visual Studio, що робить розробку та розгортання програм ще зручнішим.

- Високий рівень безпеки

.NET Framework має вбудовані механізми безпеки, які роблять її надійним вибором для розробки програм, де безпека є критично важливою.

Варто зазначити й деякі обмеження .NET Framework:

- Підтримка класичного .NET Framework доступна лише на Windows.
- Інсталяційний пакет .NET Framework має досить великий розмір.

Незважаючи на ці обмеження, .NET Framework залишається одним з найпотужніших та універсальніших інструментів для розробників, пропонуючи гнучкість, продуктивність та надійність [17, 18].

### 2.1.3. Electron.js

Electron.js – це відкритий фреймворк, який дозволяє розробникам створювати кросплатформні настільні додатки за допомогою веб-технологій, таких як HTML, CSS і JavaScript. Цей фреймворк був створений розробниками GitHub для потреб внутрішніх проектів і вперше був представлений у 2013 році під назвою Atom Shell, оскільки він лежав в основі редактора коду Atom. Перейменований в Electron у 2014 році, він швидко набув популярності серед розробників завдяки своїй здатності інтегрувати веб-технології у створення нативних додатків.

Electron використовує Node.js для бекенду та Chromium для фронтенду, ефективно поєднуючи їх у єдине робоче середовище. Це дозволяє розробникам використовувати всі бібліотеки Node.js і водночас мати доступ до сучасних веб-API, які підтримуються Chromium. Додатки на Electron можуть виконувати різні

операції, такі як читання та запис локальних файлів, доступ до мережі, інтеграція з іншими програмами на комп'ютері. Відокремлені процеси рендерингу та бекенду дозволяють забезпечувати високу продуктивність та безпеку додатків.

Однією з основних переваг Electron є можливість розробки кросплатформних додатків, які з однаковою ефективністю працюють на Windows, macOS і Linux без потреби в окремій адаптації під кожну платформу. Це значно спрощує процес розробки і зменшує загальні витрати на підтримку додатків. Проте, існують також і виклики, наприклад, збільшений розмір додатків через вбудований Chromium і Node.js, що може призвести до вищих вимог до ресурсів системи. Крім того, оскільки Electron дозволяє додаткам виконувати розширені операції, такі як доступ до файлової системи, потрібно особливо уважно ставитися до аспектів безпеки.

Серед відомих додатків, створених за допомогою Electron, можна виділити Visual Studio Code, Slack, Discord, а також додаток для спілкування Skype. Всі вони показують високу продуктивність та зручність у використанні, демонструючи, що додатки на Electron можуть бути не лише функціональними, але й ефективними для кінцевого користувача. Visual Studio Code, наприклад, став стандартом серед редакторів коду завдяки своїм потужним можливостям і активній спільноті розробників, що створюють численні розширення.

Спільнота розробників активно розвивається, пропонуючи численні ресурси для підтримки та розвитку проектів. Офіційна документація надає детальну інформацію про всі аспекти роботи з фреймворком, включаючи приклади коду та готові рішення для поширених задач. Крім того, існують численні навчальні курси, блоги та форуми, де розробники можуть обмінюватися досвідом і отримувати підтримку. Також доступні численні розширення та модулі, які можуть значно розширити можливості додатків, побудованих на Electron, і прискорити процес розробки.

Electron продовжує еволюціонувати, адаптуючись до нових технологій і вимог ринку. Постійні оновлення Chromium та Node.js забезпечують додатки на найновішими можливостями веб-розробки та високою продуктивністю. Крім

того, зростає інтеграція з іншими фреймворками та інструментами, такими як React або Angular, що дозволяє створювати ще більш потужні та гнучкі додатки. Враховуючи стрімкий розвиток екосистеми та підтримку спільноти, можна очікувати, що Electron продовжить займати ключове місце в індустрії розробки настільних додатків.

Electron.js – це потужний інструмент для створення кросплатформних настільних додатків, який поєднує в собі простоту використання веб-технологій та можливості нативних додатків. Незважаючи на деякі виклики, пов'язані з продуктивністю та розміром додатків, його переваги, такі як можливість швидкої розробки та широка підтримка спільноти, роблять його популярним вибором серед розробників. З постійним розвитком і інтеграцією нових технологій, Electron.js продовжує залишатися важливим інструментом у світі desktop-розробки.

### 2.1.4. Qt

Qt – це потужний кросплатформний фреймворк, призначений для розробки додатків з графічним інтерфейсом користувача (GUI) та без нього. Створений компанією Qt Company, цей фреймворк підтримує численні операційні системи, включаючи Windows, macOS, Linux, iOS, Android, а також вбудовані системи. Завдяки своїй універсальності та багатим можливостям, Qt став популярним вибором для створення як комерційних, так і відкритих проектів у різних галузях.

Технічна архітектура Qt надає широкий спектр інструментів та бібліотек для розробки додатків. Основою фреймворку є ядро, яке забезпечує роботу з сигналами та слотами, обробку подій, багатопоточність і інші основні функції. Qt також включає модулі для роботи з графічними інтерфейсами (Qt Widgets), 3D-графікою (Qt 3D), мультимедіа (Qt Multimedia), мережами (Qt Network) та багатьма іншими аспектами. Один з ключових компонентів — Qt Quick, який

дозволяє створювати сучасні, анімовані інтерфейси за допомогою мови декларативного програмування QML (Qt Modeling Language).

Однією з головних переваг Qt є його кросплатформність. Написавши код один раз, розробники можуть запускати свої додатки на різних платформах з мінімальними змінами. Це значно економить час і ресурси, особливо для великих проектів. Крім того, Qt забезпечує високу продуктивність та стабільність, що робить його ідеальним вибором для розробки критично важливих додатків. Інтерфейси, створені за допомогою Qt, можуть мати нативний вигляд і відчуття на кожній підтримуваній платформі, що покращує користувацький досвід.

Попри всі переваги, існують також певні виклики, пов'язані з використанням Qt. Для повного використання всіх можливостей фреймворку може знадобитися значний час на вивчення його функціоналу та API. Однак, потужна та активна спільнота розробників, а також велика кількість навчальних матеріалів, документації та прикладів коду допомагають новачкам швидко освоїти роботу з Qt. Qt Company також пропонує комерційні ліцензії та професійну підтримку, що може бути корисним для великих проектів та корпоративних клієнтів.

Qt продовжує активно розвиватися, впроваджуючи нові технології та функції. Регулярні оновлення забезпечують підтримку останніх версій операційних систем і апаратного забезпечення. З урахуванням тенденцій до розвитку Інтернету речей (IoT) та вбудованих систем, Qt стає все більш важливим інструментом для розробки програмного забезпечення в цих галузях. Фреймворк також інтегрується з іншими сучасними технологіями, такими як машинне навчання та штучний інтелект, розширюючи свої можливості і сфери застосування.

Таким чином, Qt є потужним і універсальним фреймворком, який надає розробникам всі необхідні інструменти для створення сучасних і продуктивних додатків. Його кросплатформність, широкий спектр функцій і активна спільнота роблять його ідеальним вибором для різних проектів, від простих утиліт до складних корпоративних систем. Завдяки постійному розвитку та впровадженню

нових технологій, Qt залишається на передовій сучасної розробки програмного забезпечення.

### 2.1.5. GTK

GTK, або GIMP Toolkit, є одним із найпопулярніших фреймворків для розробки графічних інтерфейсів користувача (GUI) на платформі Linux, хоча він також підтримує Windows і macOS. Розроблений як відкритий проєкт, GTK спочатку був створений для використання в графічному редакторі GIMP (GNU Image Manipulation Program), звідки й отримав свою назву. Сьогодні GTK використовується у багатьох відомих додатках та середовищах робочого столу, таких як GNOME.

GTK побудований на мові програмування C, але має обгортки для багатьох інших мов, включаючи Python (через PyGTK або PyGObject), C++, JavaScript і навіть Rust. Ця універсальність дозволяє розробникам вибирати мову програмування, яка найкраще відповідає їхнім потребам, зберігаючи при цьому всі можливості фреймворку. GTK пропонує широкий набір віджетів для створення різноманітних елементів інтерфейсу, таких як кнопки, вікна, текстові поля та інші компоненти.

Однією з основних переваг GTK є його високий рівень інтеграції з Linux-системами. Він глибоко інтегрований у середовища робочого столу, такі як GNOME, що забезпечує додаткам на GTK нативний вигляд і відчуття, узгоджені з рештою системи. Крім того, GTK має відмінну підтримку міжнародних мов і наборів символів, що робить його чудовим вибором для створення додатків, які будуть використовуватися у різних регіонах світу.

Однак, як і будь-який інший фреймворк, GTK має свої виклики. Розробка на чистому C може бути складнішою порівняно з мовами вищого рівня, такими як Python або C#. Проте, завдяки обгорткам для різних мов програмування, багато цих труднощів можна обійти. Іншим викликом може бути кросплатформність: хоча GTK підтримує Windows і macOS, основна увага

приділяється Linux, і деякі функції можуть працювати не так добре на інших платформах.

GTK продовжує активно розвиватися і вдосконалюватися. Останні версії GTK приносять значні поліпшення в продуктивності та нові можливості для розробників. Фреймворк адаптується до сучасних вимог, включаючи підтримку нових стандартів графіки і сучасних мов програмування. Це забезпечує його актуальність і робить його привабливим вибором для розробників, які хочуть створювати сучасні і функціональні графічні інтерфейси.

Таким чином, GTK є потужним і гнучким інструментом для розробки GUI-додатків, особливо на платформі Linux. Його широкий набір віджетів, підтримка різних мов програмування і активний розвиток роблять його популярним вибором серед розробників, які створюють як прості утиліти, так і складні програмні комплекси. Завдяки своїй відкритості та підтримці спільноти, GTK продовжує залишатися на передовій розробки графічних інтерфейсів.

### 2.1.6. JavaFx

JavaFX — це сучасний фреймворк для створення графічних інтерфейсів користувача (GUI) на мові програмування Java. Розроблений компанією Sun Microsystems, а згодом підтримуваний Oracle, JavaFX є спадкоємцем Swing і призначений для створення більш багатих, інтерактивних і привабливих інтерфейсів. Вперше представлений у 2008 році, JavaFX зазнав значних змін та оновлень, щоб відповідати вимогам сучасної розробки додатків.

JavaFX пропонує широкий набір компонентів та API для створення різноманітних елементів інтерфейсу, таких як кнопки, текстові поля, таблиці, графіки, анімації та мультимедіа. Однією з головних особливостей JavaFX є використання FXML — декларативної мови на основі XML для опису інтерфейсів. Це дозволяє розробникам відокремити логіку додатка від його візуального представлення, що спрощує процес розробки і підтримки коду. Крім

того, JavaFX підтримує CSS для стилізації інтерфейсів, що дозволяє легко змінювати зовнішній вигляд компонентів.

Однією з важливих переваг JavaFX є його потужні можливості для створення графіки та анімацій. Використовуючи JavaFX, розробники можуть створювати складні анімації, тривимірну графіку, а також інтегрувати мультимедійний контент у свої додатки. Це робить JavaFX ідеальним вибором для створення інтерактивних додатків, ігор та інших програм, які вимагають високого рівня графічної взаємодії. Крім того, JavaFX надає інструменти для роботи з аудіо та відео, що розширює його можливості у створенні мультимедійних додатків.

Ще однією перевагою JavaFX є його кросплатформність. Як і інші рішення на основі Java, додатки, створені з використанням JavaFX, можуть працювати на будь-якій платформі, що підтримує Java Virtual Machine (JVM). Це включає Windows, macOS, Linux та навіть мобільні платформи, такі як Android та iOS, хоча останні вимагають додаткових інструментів для повної підтримки. Завдяки цьому розробники можуть створювати кросплатформні додатки з єдиною базою коду, що значно спрощує процес розробки та підтримки.

Однак, JavaFX також має свої виклики. Попри всі його можливості, JavaFX не завжди є найкращим вибором для простих додатків через свою відносну складність порівняно зі Swing. Крім того, хоч JavaFX активно розвивається, його використання все ще менш поширене у порівнянні з іншими фреймворками, такими як Electron або навіть Swing, що може обмежити доступність ресурсів та прикладів.

На майбутнє, JavaFX продовжує активно розвиватися, з новими релізами та оновленнями, що додають нові функції та покращення. Спільнота JavaFX залишається активною, і багато розробників вибирають цей фреймворк для створення сучасних, інтерактивних додатків. Завдяки своїм багатим можливостям для створення графіки та мультимедіа, а також підтримці сучасних підходів до розробки, JavaFX залишається потужним інструментом для розробки графічних інтерфейсів користувача.

Отже, JavaFX є сучасним і потужним фреймворком для створення графічних інтерфейсів на Java, пропонуючи багатий набір компонентів, потужні можливості для створення графіки та анімацій, а також кросплатформну підтримку. Незважаючи на деякі виклики, пов'язані з його складністю та меншою поширеністю, JavaFX продовжує залишатися важливим інструментом для розробників, які прагнуть створювати сучасні та інтерактивні додатки.

## 2.2. API для моніторингу криптовалют

Кожен API має набір протоколів для взаємодії, що визначають як системи спілкуються одна з одною. Ці протоколи доповнюють, а не замінюють інші протоколи або розширення API. Наприклад, HTTP/HTTPS використовується для надсилання запитів через команду GET, а відповіді також передаються через той же протокол. Проте, формати запитів та адресація, куди ці запити надсилаються, представляють собою два різні аспекти.

### 2.2.1. CoinGecko API

CoinGecko API є потужним інструментом для розробників, який надає доступ до широкого спектру даних про криптовалюти та їх ринки у реальному часі. Він дозволяє отримувати актуальну інформацію про ціни, ринкову капіталізацію, обсяги торгів та інші важливі показники для понад 6000 криптовалют. Користувачі можуть легко інтегрувати ці дані у свої додатки або вебсайти, щоб забезпечити своїх користувачів оновленою інформацією про ринок криптовалют.

API пропонує цінні дані про криптовалютні біржі, включаючи обсяги торгів, ліквідність та активність користувачів. Завдяки можливості отримувати доступ до історичних даних криптовалют дозволяє проводити ретроспективний аналіз, оцінювати вплив минулих подій та робити обґрунтовані прогнози щодо майбутнього ринку.

CoinGecko API виходить за рамки традиційних ринкових даних, надаючи доступ до соціальних даних та сигналів з таких платформ, як Twitter, Reddit та GitHub. Ці дані дають змогу дослідникам виявляти зміни настроїв інвесторів, оцінювати вплив новин та подій на ринок та передбачати потенційні зміни ціни [6].

Зростаючий ринок децентралізованих фінансів (DeFi) охоплений CoinGecko API, що пропонує дані про токени DeFi, обсяги ліквідності, протоколи DeFi та інші важливі показники. Ці дані дозволяють дослідникам досліджувати цю динамічну сферу, оцінювати ризики та можливості та аналізувати вплив DeFi на загальний ринок криптовалют.

Загалом, CoinGecko API є надзвичайно корисним ресурсом для всіх, хто працює з криптовалютами. Його безкоштовний план з обмеженим доступом дозволяє легко розпочати роботу, а широкий спектр доступних даних забезпечує глибоке розуміння ринку. Регулярне оновлення даних у реальному часі забезпечує користувачам постійний доступ до актуальної інформації.

### 2.2.2. CoinMarketCap API

CoinMarketCap API є потужним інструментом, який надає розробникам детальні дані про ринки криптовалют. Він забезпечує доступ до великої кількості інформації: ціни, ринкову капіталізацію, обсяги торгів та інші важливі показники; яка оновлюється в реальному часі. API стала є популярною серед фінансових аналітиків, інвесторів та розробників, які прагнуть інтегрувати надійні дані про криптовалюти у свої додатки або платформи [14].

Основною перевагою CoinMarketCap API є її здатність надавати вичерпну інформацію про всі аспекти ринку криптовалют, такі як поточна ціна в різних валютах, історичні дані, процентні зміни в ціні, доступність на різних біржах та багато іншого. Така широта і глибина даних роблять її ідеальним ресурсом для проведення ретельного ринкового аналізу та прийняття обґрунтованих інвестиційних рішень.

Крім того, CoinMarketCap API пропонує високий рівень налаштування запитів, що дозволяє користувачам отримувати саме ту інформацію, яка їм потрібна. Це може включати специфічні дані для певних криптовалют, агрегацію даних за вказаний період часу, а також можливість слідкувати за ринковими трендами та коливаннями. API також підтримує різні рівні доступу, що забезпечує гнучкість і масштабованість для індивідуальних потреб користувачів, від особистих проєктів до великих комерційних застосувань.

На додаток до ринкових даних, CoinMarketCap API також надає доступ до соціальних показників і рейтингів, які можуть вказувати на настрої інвесторів та споживачів щодо певних криптовалют. Це включає інформацію про кількість дописів у соціальних мережах, коментарі та інші важливі метрики, які допомагають виявляти тенденції та патерни в поведінці ринку.

Завдяки своїй повноті, точності та широкому спектру функцій, CoinMarketCap API є незамінним інструментом для будь-якого розробника або аналітика, який працює з криптовалютами. Окрім базових даних про ціни та обсяги торгів, API також включає інформацію про ринкову капіталізацію, домінування криптовалют на ринку, та дані про біржі, такі як обсяги торгів на конкретних платформах, комісії та інші деталі. Це дозволяє користувачам отримувати всебічне уявлення про ринок і приймати більш обґрунтовані рішення.

API CoinMarketCap також забезпечує високу швидкість та надійність даних, що особливо важливо для додатків, які потребують актуальної інформації в режимі реального часу. Це включає торгові боти, фінансові інформаційні панелі, аналітичні інструменти та мобільні додатки, які потребують точних і швидких оновлень. Підтримка великої кількості одночасних запитів робить цю API ідеальною для використання в масштабованих системах та сервісах.

Для розробників API пропонує добре документовані endpoints, що спрощує інтеграцію та використання. Документація включає детальні описи кожного запиту, приклади коду та інструкції для різних мов програмування, що значно

полегшує роботу з API. Це дозволяє навіть новачкам швидко розпочати роботу та ефективно використовувати ресурси API.

### 2.2.3. CryptoCompare API

CryptoCompare API є універсальним інструментом для отримання детальної інформації про ринки криптовалют, надаючи доступ до різноманітних даних, таких як поточні ціни криптовалют, обсяги торгів і ринкові метрики в реальному часі. Це дозволяє трейдерам, аналітикам і ентузіастам швидко реагувати на ринкові зміни і приймати обґрунтовані рішення. Крім того, доступ до історичних даних дає можливість аналізувати минулі ринкові тренди і прогнозувати можливі майбутні зміни, що є важливим для стратегічного планування [15].

Однією з важливих переваг CryptoCompare API є можливість отримувати соціальні індикатори, що відображають активність і настрої користувачів у соціальних мережах, таких як Twitter та Reddit. Ці дані дозволяють оцінювати вплив громадської думки на ринок криптовалют, додаючи новий вимір до традиційного ринкового аналізу. Соціальні індикатори допомагають виявляти тренди та передбачати ринкові рухи, що може бути корисним для трейдерів і аналітиків.

API також надає детальну інформацію про криптовалютні біржі, включаючи обсяги торгів, ліквідність і доступні торгові пари. Це допомагає користувачам вибирати найкращі платформи для торгівлі та розуміти динаміку різних ринків. Інформація про конкретні монети, їхні технічні характеристики та ринкову капіталізацію робить API корисним для глибокого аналізу конкретних криптовалют. Загалом, CryptoCompare API є цінним інструментом для всебічного аналізу ринку криптовалют, забезпечуючи гнучкість і надійність для різних додатків і сервісів.

## 2.3. Висновок до другого розділу

Цей розділ присвячений інструментам розробки програмного забезпечення, як Windows Presentation Foundation (WPF), платформа .NET Framework, мова програмування C#, та фреймворк Electron.js. Було детально описані їхні функціональні можливості, переваги та недоліки, а також розглянуто типи API для роботи з криптовалютами.

WPF пропонує розробникам великий арсенал інструментів для створення складних інтерфейсів користувача, підтримки анімації та зв'язування даних. Використання мови розмітки XAML полегшує процес розробки та сприяє співпраці між розробниками та дизайнерами. Основні переваги WPF включають гнучкість дизайну, шкальованість та можливість повторного використання стилів та шаблонів. Водночас, технологія може бути ресурсомісткою та складною для засвоєння новачками.

Платформа .NET Framework є універсальним інструментом для розробки різноманітних програм, забезпечуючи підтримку багатьох мов програмування та пропонуючи великий набір класів та бібліотек. Вона підтримує створення кросплатформених додатків, що робить її привабливою для розробників. Основні недоліки включають обмеження підтримки лише на Windows та великий розмір інсталяційного пакету.

CoinGecko API, CoinMarketCap API та CryptoCompare API надають широкий спектр даних про ринки криптовалют, включаючи ціни в реальному часі, історичні дані, ринкову капіталізацію, обсяги торгів та соціальні індикатори. Кожен з цих API має свої переваги та обмеження, що дозволяє користувачам вибирати найбільш відповідний інструмент для своїх потреб.

# РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Аналіз вимог, проектування архітектури та формулювання завдання.

Оскільки рішення щодо проектування та реалізації розроблюваних компонентів впливають на всю систему, кожна підсистема повинна відповідати вимогам. Тому детальний аналіз включає дослідження функціональних можливостей програмного продукту, призначеного для збору, обробки та аналізу даних певної криптовалюти.

Технічні вимоги до програмного продукту включають:

1. Сумісність з персональними комп'ютерами зі стандартним набором компонентів.
2. Зручність та інтуїтивність у користуванні.
3. Високу швидкість обробки даних та доступ до інформації в режимі реального часу.
4. Легкість масштабування та обслуговування.

Концептуальна модель представляє собою поєднання бачення розробника та користувача. Взаємодія між акторами та прецедентами зображена на діаграмі прецедентів (рис. 3.1). Ця діаграма відображає всіх учасників цієї інтелектуальної системи та всі можливі сценарії їхніх дій.

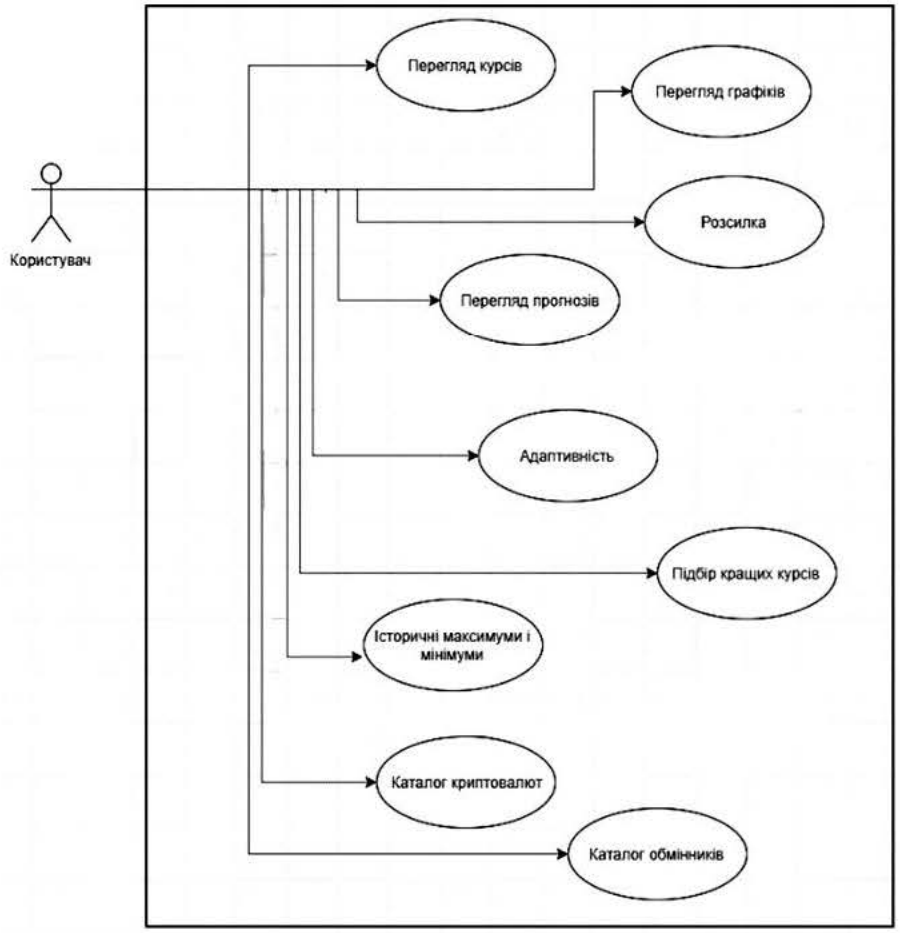



Рисунок 3.1 – Діаграма прецендентів

Актор "Користувач" є відвідувачем сервісу, який має можливість користуватися наступними функціями: перегляд курсів, перегляд графіків, перегляд прогнозів, отримання розсилки, вибір мови інтерфейсу, адаптивність, підбір кращих курсів, історичні максимуми і мінімуми, каталог бірж, обмінників та криптовалют.

- Перегляд курсів – користувач може переглядати курси за різні періоди часу, а також отримувати таблиці зі списком торгів на біржах.
- Перегляд графіків – автоматична побудова графіків за різні періоди часу.
- Розсилка – користувач може підписатися на отримання прогнозів та інформаційних матеріалів з тематики криптовалют.

- Адаптивність – додаток автоматично адаптується під різні розширення екрана комп'ютера та телефону.
- Підбір кращих курсів – автоматичний підбір курсів, за якими вигідно купувати та продавати.
- Історичні максимуми і мінімуми – міні-сервіс для збору історії курсів, який дозволяє переглядати максимальний ріст та спад курсів за певний проміжок часу.
- Каталог обмінників – список обмінників, де можна обміняти валюту за заявками.
- Каталог криптовалют – список криптовалют із їхніми технічними характеристиками.

Система базується на використанні .NET 6 та WPF, а дані про криптовалюти отримуються з API CoinGecko.

## 3.2. Основна логіка виконання програми

Згідно сформованих вимог логіка виконання програми заснована на взаємодії з користувачем, який обирає необхідні функціональні можливості для виконання в потрібний момент. Головною сутністю предметної області є криптовалюти, оскільки проблема стосується їх моніторингу та порівняння. На прикладі відображення даних можна продемонструвати, як реалізується логіка виконання програми.

Коли користувач запускає програму, вона відправляє запит до API CoinGecko, і в цей момент дані вважаються "отриманими". До моменту, коли дані відображаються на екрані користувача або оновлюються, вони можуть перебувати у стані "отримані" або "помилка отримання". У випадку отримання даних із помилкою, їх обробка припиняється.

Якщо дані отримані без помилок, користувач має змогу в будь-який час переглянути їх, а також виконати фільтрацію та сортування. Після фільтрації чи сортування стан даних змінюється на "відфільтровані" або "відсортовані"

відповідно. Якщо фільтрація чи сортування не проводилися, дані автоматично оновлюються через певний проміжок часу і повертаються до стану "отримані".

## 3.3. Розробка проекту

Для реалізації додатку реалізуємо його сторінки. В проект входять такі вікна:

1. HomeView
2. ErrorDialog
3. CoinView

Далі розглянемо кожен окремо.

1. HomeView

Клас HomeView (рис. 3.2) забезпечує основний функціонал для перегляду та взаємодії з даними про криптовалюти, надаючи користувачам зручний інтерфейс для отримання актуальної інформації про ринок.

Основні методи:

1) KeyExtractor(string property) – повертає функцію для отримання значення властивості CoinMarkets на основі назви поля.

2) UpdateSortParams(string property, bool reverse) – оновлює параметри сортування (поле та напрямок) і повертає функцію для витягування значення відповідної властивості.

3) SortObservableCollection(string property) – сортує колекцію Coins на основі заданого поля.

4) GetCoins(int size, int page) – викликає API для отримання списку криптовалют, вказуючи кількість записів на сторінку і номер сторінки.

5) SelectCoin(object sender, MouseButtonEventArgs e) – обробляє подію вибору криптовалюти, відкриваючи вікно з детальною інформацією про обрану криптовалюту.

```
public partial class HomeView
{
    Ссылок: 2
    private CoinGeckoClient Api { get; } = new();
    Ссылок: 3
    public ObservableCollection<CoinMarkets> Coins { get; set; } = new();
    Ссылок: 1
    public decimal MarketCapChangePercentage24h { get; set; }
    Ссылок: 1
    public RelayCommand<string> Sort { get; set; }
    Ссылок: 1
    public AsyncRelayCommand<Currency> SelectCurrency { get; set; }
    Ссылок: 1
    public AsyncRelayCommand DismissErrorCommand { get; set; }
    Ссылок: 3
    public string SortedBy { get; set; } = "#";
    Ссылок: 5
    public ListSortDirection SortDirection { get; set; } = ListSortDirection.Ascending;
    Ссылок: 8
    public int CurrentPage { get; set; } = 1;
    Ссылок: 5
    public Visibility LoadingVisibility { get; set; } = Visibility.Visible;
    Ссылок: 0
    public double BackButtonOpacity => CurrentPage == 1 ? 0 : 1;

    Ссылок: 1
    public static List<Currency> Currencies ...

    Ссылок: 4
    public Currency SelectedCurrency { get; private set; } = Currencies.FirstOrDefault();
    Ссылок: 6
    public bool CurrencyDropdownOpen { get; set; }
    Ссылок: 1
    public bool CoinsLoaded { get; set; }

    Ссылок: 1
    private static Func<CoinMarkets, object> KeyExtractor(string property)...

    Ссылок: 2
    private Func<CoinMarkets, object> UpdateSortParams(string property, bool reverse)...

    Ссылок: 1
    private void SortObservableCollection(string property)...

    Ссылок: 1
    private Task<List<CoinMarkets>> GetCoins(int size, int page)...

    Ссылок: 1
    private void SelectCoin(object sender, MouseButtonEventArgs e)...

    Ссылок: 1
    private void ToggleCurrencyDropdown(object sender, MouseEventArgs e)...

    Ссылок: 1
    private async Task OnSelectCurrency(Currency currency)...

    Ссылок: 5
    private async Task FetchData(int page = 1)...

    Ссылок: 1
    private async void PreviousPage(object sender, RoutedEventArgs ea)...

    Ссылок: 1
    private async void NextPage(object sender, RoutedEventArgs ea)...

    Ссылок: 0
    public HomeView()...
}
```

Рисунок 3.2 – клас HomeView

6) ToggleCurrencyDropdown(object sender, MouseEventArgs e) – перемикає стан відкриття/закриття випадаючого меню валют.

7) OnSelectCurrency(Currency currency) – асинхронно обробляє вибір нової валюти, оновлюючи дані та змінюючи культуру відображення.

8) FetchData(int page = 1) – асинхронно отримує та оновлює дані про криптовалюти, відображаючи індикатор завантаження під час процесу.

9) PreviousPage(object sender, RoutedEventArgs ea) – переходить на попередню сторінку даних криптовалют, якщо поточна сторінка не є першою.

10) NextPage(object sender, RoutedEventArgs ea) – переходить на наступну сторінку даних криптовалют, обмежуючи максимальну кількість сторінок до 100.

2. CoinView

Цей клас забезпечує основний функціонал для відображення детальної інформації про криптовалюту (рис. 3.3). Він включає методи для завантаження даних про ціну та об'єми торгів, побудови графіків, відображення діапазонів часу та форматування даних. Клас також обробляє події користувача, такі як рух миші над графіком, для динамічного відображення відповідних даних.

Методи класу CoinView:

1) FormatMixed(decimal number, CultureInfo culture) – форматує число відповідно до заданої культури.

2) FetchChart(bool initial = false) – асинхронно отримує та обробляє дані графіка для вибраної криптовалюти і діапазону часу.

3) FetchData() – асинхронно отримує основні дані про криптовалюту.

4) StringToTextBlocks(string input) – перетворює HTML-строку в колекцію текстових блоків.

5) Loading(bool initial = false) – відображає індикатор завантаження.

6) DoneLoading(bool shouldClose = false) – приховує індикатор завантаження або закриває вікно.

7) Error() – відображає повідомлення про помилку.

8) Конструктор CoinView(string coin, string currency) – ініціалізує новий екземпляр класу, встановлює шрифт, запускає завантаження даних.

9) ChartMouseMove(object sender, MouseEventArgs e) – обробляє рух миші над графіком, відображаючи відповідні значення.

10) FollowOnTwitter(object sender, MouseButtonEventArgs e) – відкриває сторінку CoinMarketCap в Twitter у веб-браузері.

```
public partial class CoinView
{
    Ссылок: 3
    public string Currency { get; set; }
    Ссылок: 2
    public CoinMarkets CoinData { get; private set; }
    Ссылок: 1
    public Thickness YAxisMargin { get; private set; }
    Ссылок: 1
    public Thickness YOpenMargin { get; private set; }
    Ссылок: 1
    public string YPointPrice { get; private set; }
    Ссылок: 2
    public Point ChartPoint { get; set; }
    Ссылок: 2
    private Dictionary<int, Point> PathPoints { get; set; } = new();
    Ссылок: 1
    public ObservableCollection<string> DescriptionRows { get; } = new();
    Ссылок: 3
    public Visibility LoadingVisibility { get; set; } = Visibility.Collapsed;
    Ссылок: 2
    public double LoadingOpacity { get; set; } = 1;
    Ссылок: 4
    public ObservableCollection<string> Timesteps { get; } = new();
    Ссылок: 8
    public TimeRange TimeRange { get; set; } = new(TimeSpan.FromDays(30), "30D");
    Ссылок: 0
    public List<TimeRange> TimeRangeNames ...
    Ссылок: 0
    public string FormattedOpen => FormatMixed(Open, CultureInfo.CurrentCulture);
    readonly DecimalToVariablePrecision _variablePrecisionConverter = new();
    Ссылок: 1
    public AsyncRelayCommand<TimeRange> ChangeTabCommand { get; }
    Ссылок: 2
    public RelayCommand DismissErrorCommand { get; private set; }

    Ссылок: 1
    public PriceData PriceAtPoint...

    Ссылок: 0
    public string ChartPointFill => PriceAtPoint?.Price >= Open ? "#16C784" : "#EA3943";

    Ссылок: 0
    public string Sparkline...

    Ссылок: 0
    public Brush Stroke...

    Ссылок: 0
    public Brush Fill...

    Ссылок: 0
    public List<string> YPrices...

    Ссылок: 0
    public string VolumeSparkline...

    private const decimal _hResolution = 360;
    private const decimal _wResolution = 740;
    private readonly CoinGeckoClient _api = new();
    private readonly string _coin;
    private PriceData _priceAtPoint;
    Ссылок: 10
    private MarketChartById Data { get; set; }
    Ссылок: 3
    private Dictionary<decimal, PriceData> PriceData { get; set; } = new();
    Ссылок: 4
    private decimal Step { get; set; } = 2.75m;
    Ссылок: 9
    private decimal Min { get; set; }
    Ссылок: 9
    private decimal Max { get; set; }
    Ссылок: 6
    private decimal Open { get; set; }
    private bool ResyncChart { get; set; }
    Ссылок: 2
    private TimeRange OldTimeRange { get; set; } = new(TimeSpan.FromDays(30), "30D");

    Ссылок: 3
    string FormatMixed(decimal number, CultureInfo culture)...

    Ссылок: 2
    private async Task FetchChart(bool initial = false)...

    Ссылок: 1
    private async Task FetchData()...

    Ссылок: 1
    private static IEnumerable<string> StringToTextBlocks(string input)...

    Ссылок: 1
    private void Loading(bool initial = false)...

    Ссылок: 3
    private void DoneLoading(bool shouldClose = false)...

    Ссылок: 1
    private void Error()...

    Ссылок: 1
    public CoinView(string coin, string currency)...

    Ссылок: 1
    private void ChartMouseMove(object sender, MouseEventArgs e)...

    Ссылок: 1
    private void FollowOnTwitter(object sender, MouseButtonEventArgs e)...
}
```

Рисунок 3.3 – клас CoinView

3. ErrorDialog

Клас ErrorDialog (рис. 3.4) створює вікно діалогового повідомлення про помилку в додатку. Клас успадковується від Primitives.Window, що дозволяє йому мати всі функції стандартного вікна у WPF. При ініціалізації класу

викликається метод InitializeComponent для налаштування інтерфейсу, а також встановлюється шрифт "Inter" для всіх текстових елементів у вікні. Шрифт завантажується з локальної директорії додатка, що забезпечує консистентний зовнішній вигляд.

```
namespace CoinMarketCap.Views;

using System.IO;
using System.Windows.Media;

Ссылок: 2
public partial class ErrorDialog : Primitives.Window
{
    Ссылок: 0
    public ErrorDialog()
    {
        InitializeComponent();
        FontFamily = new FontFamily(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Fonts", "#Inter"));
    }
}
```

Рисунок 3.4 – Вікно ErrorDialog

4. Converters

Модуль Converters забезпечує функціонал для перетворення даних у WPF-додатку (рис. 3.5). Дозволяючи відображати дані у відповідних форматах, залежно від контексту та вимог інтерфейсу користувача. Підтримуються різні сценарії, такі як: форматування чисел, обчислення відсотків, відображення спарклайнів, керування видимістю елементів.

Основні класи модулю:

1) Клас ToUpper – перетворює рядок на верхній регістр.

2) Клас CurrencyVolumeToVolume – обчислює об'єм торгів в одиницях криптовалюти, форматуючи його з символом криптовалюти.

3) Клас SupplyWithSymbol – відображає обсяг обігу криптовалюти разом з її символом.

```
12    public class ToUpper ...
25
26    public class CurrencyVolumeToVolume ...
44
45    public class SupplyWithSymbol ...
63
64    public class SparklineToSvg ...
99
100   public class SparklineToBrush ...
117
118   public class SortToText ...
139
140   public class PercentageToTrend ...
152
153   public class PercentageToText ...
167
168   public class PercentageToBrush ...
180
181   public class DecimalToVariablePrecision ...
201
202   public class DecimalToVariableCurrency ...
222
223   public class DecimalToShortCurrency ...
244
245   public class DecimalToZeroCurrency ...
257
258   public class ChartHeightToMaxHeight ...
270
271   public class DayToFontWeight ...
291
292   public class SelectedRangeToBackground ...
314
315   public class DecimalDivision ...
331
332   public class NullToVisibilityConverter ...
344
345   public class CurrencyToVisibility ...
361
362   public class DarkModeToBrush ...
```

Рисунок 3.5 – Модуль Converters

4) Клас SparklineToSvg – перетворює дані спарклайну в формат SVG для відображення графіку.

5) Клас SparklineToBrush – визначає колір спарклайну залежно від зміни ціни (зелений для зростання, червоний для спаду).

6) Клас SortToText – форматує текст для сортування стовпців з відповідними символами напрямку сортування.

7) Клас PercentageToTrend – визначає тренд (зростання чи спад) на основі відсоткового значення.

8) Клас PercentageToText – форматує відсоткове значення з відповідними символами напрямку (стрілки).

9) Клас PercentageToBrush – визначає колір на основі відсоткового значення (зелений для зростання, червоний для спаду).

10) Клас DecimalToVariablePrecision – форматує число з різною кількістю десяткових знаків залежно від його величини.

11) Клас DecimalToVariableCurrency – форматує число як валюту з різною кількістю десяткових знаків залежно від його величини.

12) Клас DecimalToShortCurrency – форматує число як скорочену валюту (з використанням тисяч, мільйонів, мільярдів).

13) Клас DecimalToZeroCurrency – форматує число як валюту без десяткових знаків.

14) Клас ChartHeightToMaxHeight – збільшує висоту графіку на певну величину.

15) Клас DayToFontWeight – визначає товщину шрифту на основі формату дати.

16) Клас SelectedRangeToBackground – визначає колір фону залежно від вибраного діапазону часу та темного/світлого режиму.

17) Клас DecimalDivision – ділить два числа і форматує результат.

18) Клас NullToVisibilityConverter – визначає видимість елементу залежно від його значення (при null елемент прихований).

19) Клас CurrencyToVisibility – визначає видимість елементу залежно від вибраної валюти.

20) Клас DarkModeToBrush – визначає колір залежно від темного/світлого режиму.

## 3.4. Тестування роботи додатку

WPF-додаток є платформою, яка забезпечує доступ до можливості моніторингу цін на криптовалюти. Користувачі зможуть знаходити валюту що їх цікавлять.

Для виконання додатку потрібно забезпечити наступні умови:

- операційна система – Windows 11,
- оперативна пам’ять – 2 Гб;
- платформа – 64-розрядна;
- жорсткий диск – 1 Гб вільного простору.

Наступним кроком є тестування програми, перевірка всіх її функції та коректної роботи інтерфейсу. Спочатку потрібно завантажити програму та запустити її. Після цього користувач повинен спостерігати наступне вікно (рис. 3.6).

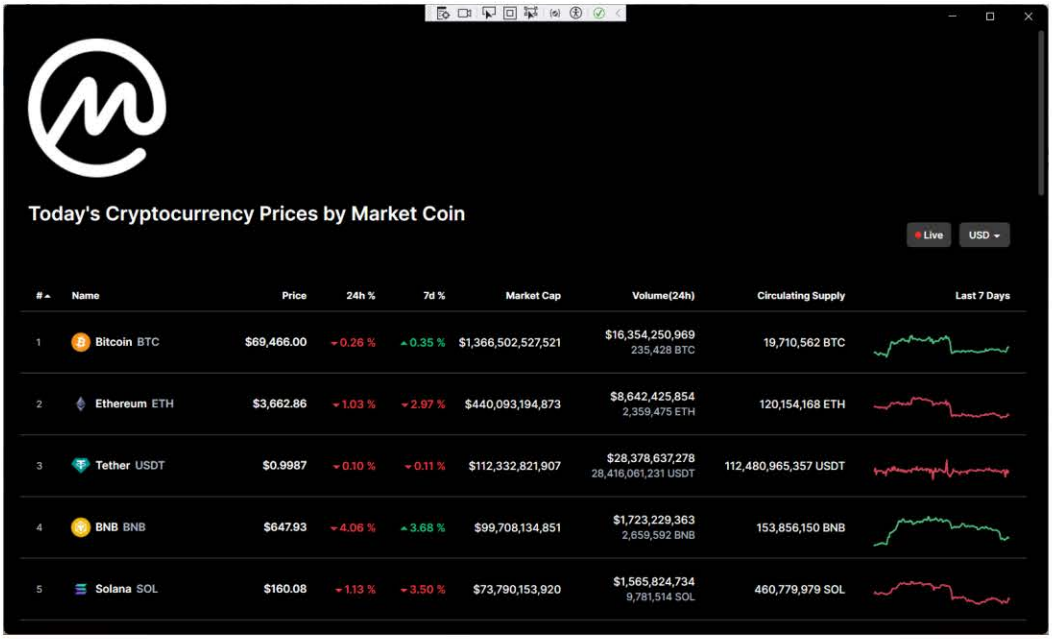



Рисунок 3.6 – Головне вікно додатку

Це вікно запускається на старті програми якщо маємо Інтеренет-з’єднання та не вичерпали ліміт безкоштовної кількості запитів. Якщо немає мережі маємо такий вигляд вікна (рис. 3.7 ).

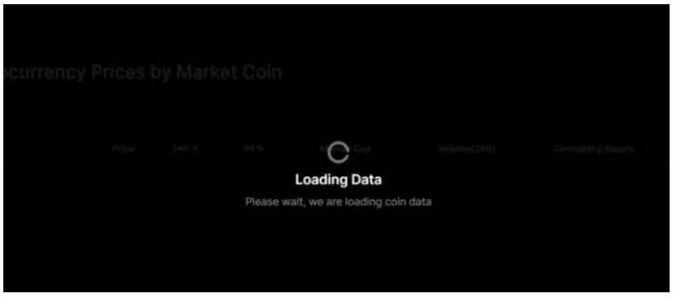



Рисунок 3.7 – Відсутня мережа

Якщо перевищили ліміт запитів до API, то отримаємо таку вигляд вікна (рис. 3.8)

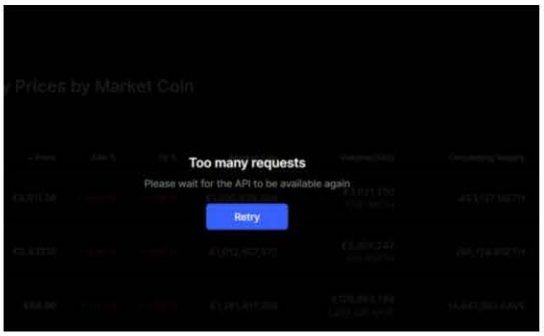
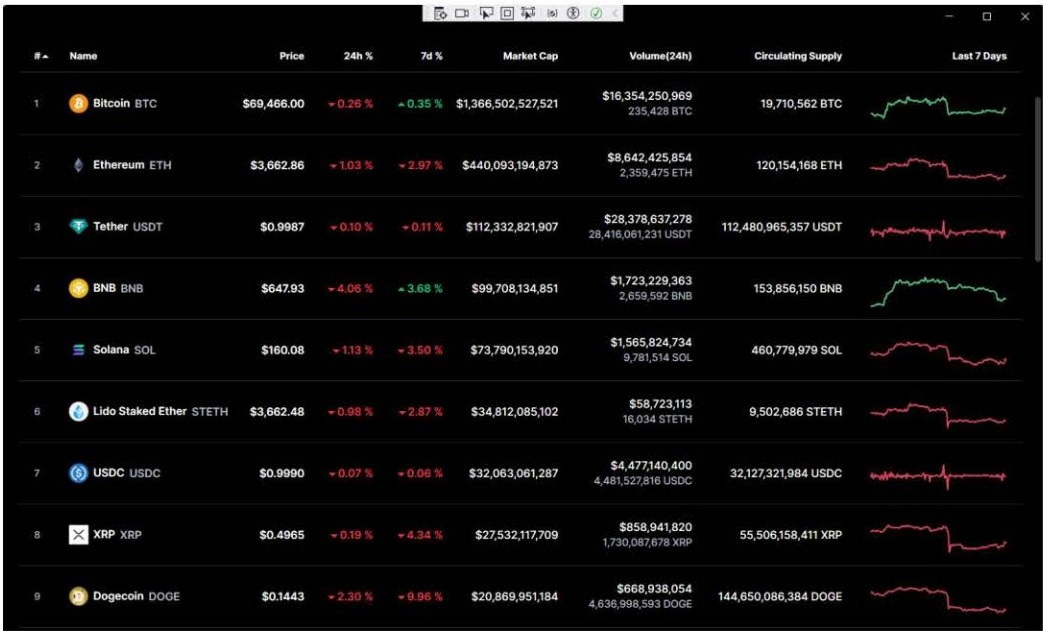



Рисунок 3.8 – Перевищений ліміт запитів

Це вікно є головним вікном програми містить в собі таблицю з коротку інформацією про криптовалюти: ціна, поведінку курсу протягом 1 доби, 7 діб; графік поведінки (рис. 3.9).



| # | Name | Price | 24h % | 7d % | Market Cap | Volume(24h) | Circulating Supply | Last 7 Days |
|---|---|---|---|---|---|---|---|---|
| 1 | Bitcoin BTC | $69,466.00 | 0.26 % | 0.35 % | $1,366,502,527,521 | $16,354,250,969 235,428 BTC | 19,710,562 BTC | |
| 2 | Ethereum ETH | $3,662.86 | 1.03 % | 2.97 % | $440,093,194,873 | $8,642,425,854 2,359,475 ETH | 120,154,168 ETH | |
| 3 | Tether USDT | $0.9987 | 0.10 % | 0.11 % | $112,332,821,907 | $28,378,637,278 28,416,061,231 USDT | 112,480,965,357 USDT | |
| 4 | BNB BNB | $647.93 | 4.06 % | 3.68 % | $99,708,134,851 | $1,723,229,363 2,659,592 BNB | 153,856,150 BNB | |
| 5 | Solana SOL | $160.08 | 1.13 % | 3.50 % | $73,790,153,920 | $1,565,824,734 9,781,514 SOL | 460,779,979 SOL | |
| 6 | Lido Staked Ether STETH | $3,662.48 | 0.98 % | 2.87 % | $34,812,085,102 | $58,723,113 16,034 STETH | 9,502,686 STETH | |
| 7 | USDC USDC | $0.9990 | 0.07 % | 0.06 % | $32,063,061,287 | $4,477,140,400 4,481,527,816 USDC | 32,127,321,984 USDC | |
| 8 | XRP XRP | $0.4965 | 0.19 % | 4.34 % | $27,532,117,709 | $858,941,820 1,730,087,678 XRP | 55,506,158,411 XRP | |
| 9 | Dogecoin DOGE | $0.1443 | 2.30 % | 9.96 % | $20,869,951,184 | $668,938,054 4,636,998,593 DOGE | 144,650,086,384 DOGE | |

Рисунок 3.9 – Таблиця з короткими відомостями про криптовалюти

Для знаходження бажаної криптовалюти можна використовувати навігацію у вигляді кнопок «назад» та «вперед» (рис. 3.10).

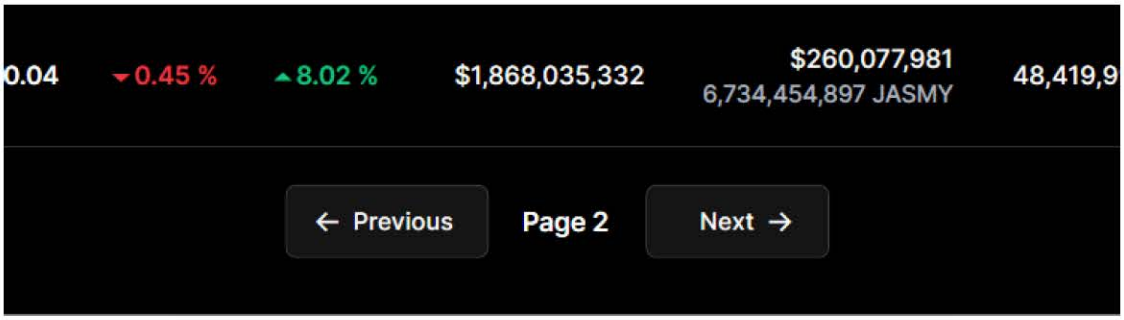



Рисунок 3.10 – Навігація в таблиці

Також таблиця має відповідні фільтри за якими можна роботи упорядкування таблиці. Для цього на верхній панелі обираємо фільтр що цікавить. Наприклад якщо хочемо упорядкувати згідно ціни (рис 3.11).

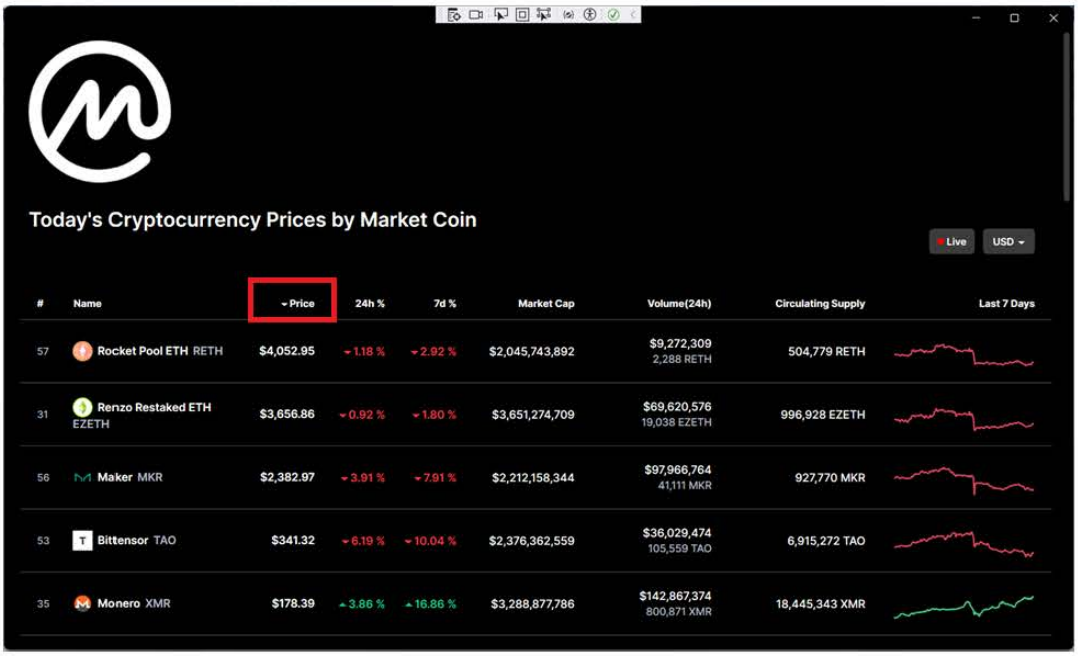



Рисунок 3.11 – Фільтри в таблиці

В таблиці можна конвертацію ціни криптовалюти відносно іншої валюти. На даний момент часу для вибору доступні американський долар та євро (рис. 3.12).

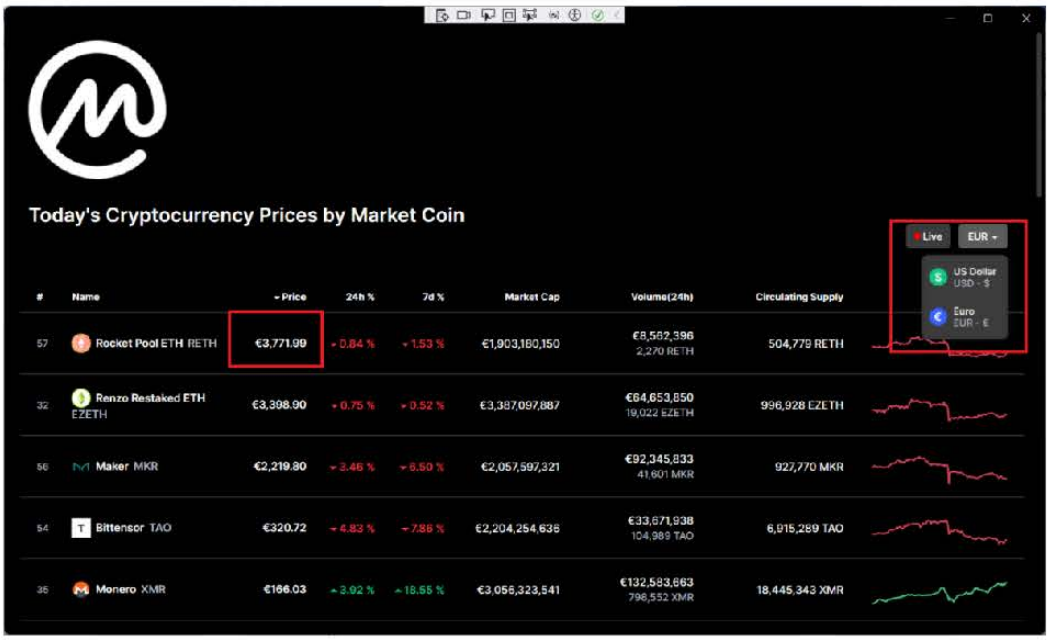



Рисунок 3.12 – Відображення конвертації криптовалюти

В таблиці можна натиснути на будь-яку криптовалюту що нас цікавить та отримати більше про неї інформації. При натисканні відкривається окреме вікно з детальною інформацією (рис. 3.13).

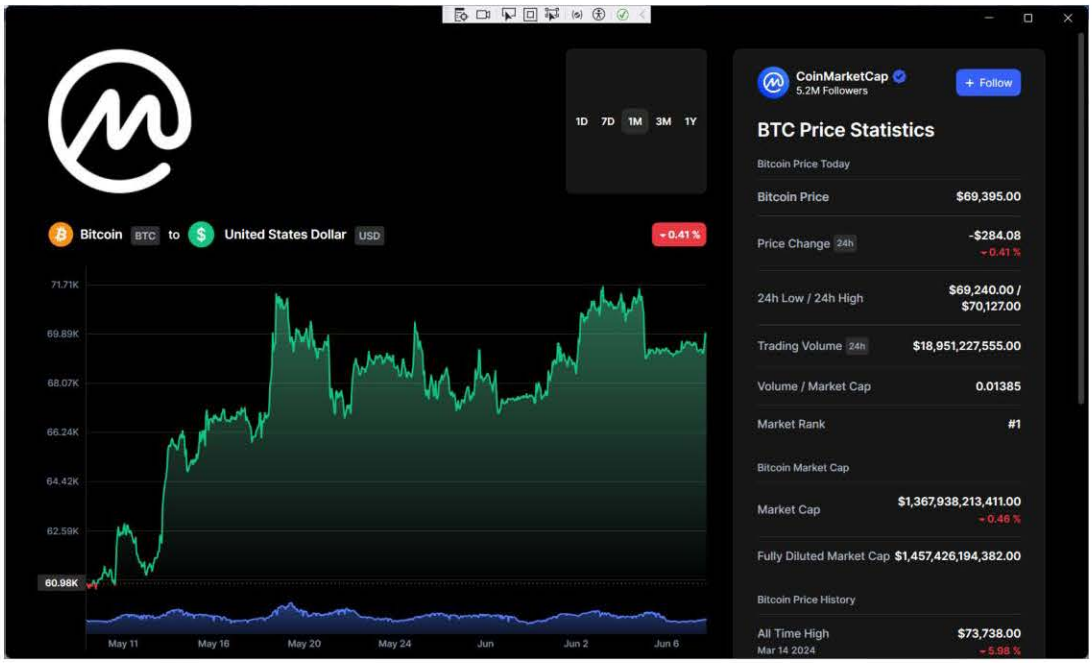



Рисунок 3.13 – Вікно з детальною інформацією про обрану криптовалюту

У вікні маємо короткий опис криптовалюти (рис 3.14)

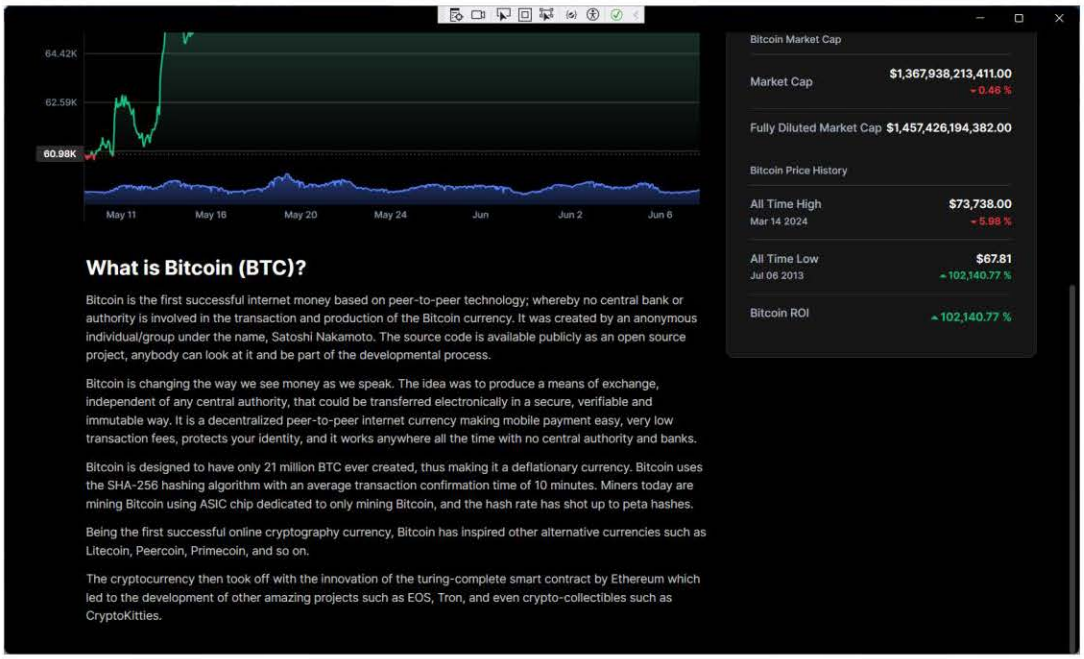



Рисунок 3.14 – Опис обраної криптовалюти

З права маємо колонку ціновою статистикою (рис. 3.15).


Рисунок 3.15 – Статистика по валюті

А також графічне відображення динаміки поведіки курсу криптовалюти за день, тиждень, місяць, 3 місяці та рік (рис. 3.16).

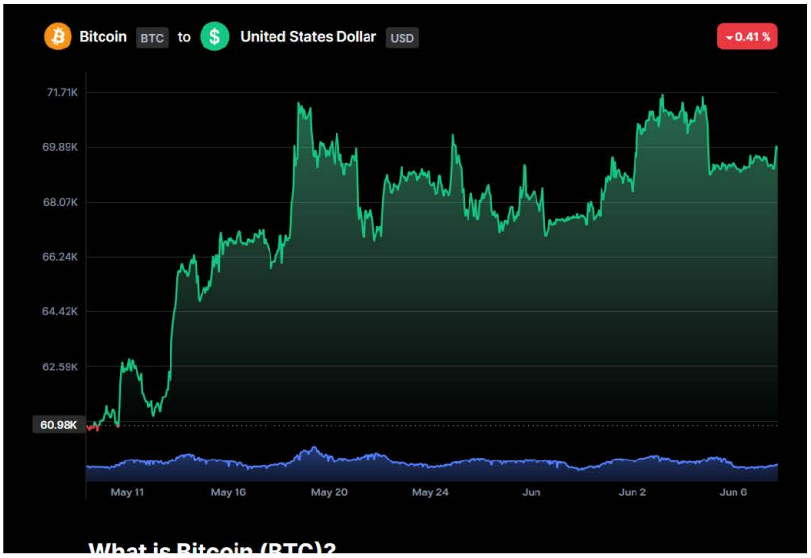



Рисунок 3.16 – Графічне відображення динаміки поведінки курсу криптовалюти

Тестування проекту відслідковування курсу криптовалют було успішно завершено. Додаток демонструє високу стабільність, безпеку та продуктивність. Інтерфейс користувача є зручним та інтуїтивно зрозумілим, що забезпечує високу задоволеність користувачів. Ніяких критичних дефектів або вразливостей не було виявлено.

## 3.5 Висновок до третього розділу

У результаті виконаної роботи було розроблено програмний продукт для моніторингу криптовалют, що має зручний і інтуїтивний інтерфей, високу швидкість обробки даних та можливістю отримання інформації у реальному часі. Використання API CoinGecko дозволяє користувачам отримувати актуальні курси криптовалют, графіки та історичні дані. Базується на використанні .NET 6 та WPF, що забезпечує його стабільність та високу продуктивність.

Основні функції додатку включають перегляд актуальних курсів криптовалют, автоматичне створення графіків, фільтрацію та сортування інформації, вибір оптимальних курсів для купівлі та продажу, а також можливість підписки на інформаційні бюлетені з прогнозами.

Проведене тестування підтвердило високу надійність та безпеку продукту, а виявлені проблеми були ефективно вирішені. Завдяки цьому, додаток є надійним інструментом для моніторингу криптовалют, який надає користувачам можливість приймати обґрунтовані інвестиційні рішення та стежити за найновішими трендами на ринку криптовалют.

# ВИСНОВОК

Кваліфікаційна робота присвячена актуальному питанню розробки програмного забезпечення для моніторингу криптовалют. У першому розділі проведено дослідження предметної області та проаналізовано сучасний стан криптовалютного ринку. Встановлено, що моніторинг та аналіз криптовалют є важливими завданнями для трейдерів та інвесторів, оскільки дозволяють зменшити ризики та підвищити ефективність торгівлі.

Технологія блокчейн, що є основою для більшості криптовалют, гарантує безпеку та прозорість у проведенні транзакцій, дозволяючи відслідковувати їх у реальному часі. Ця технологія характеризується децентралізацією, що виключає контроль з боку центральних установ, забезпечуючи користувачам зниження витрат на транзакції та більшу конфіденційність.

Однак, криптовалюти не позбавлені недоліків як-от висока волатильність та ризики, що виникають через їхній спекулятивний характер. Інвестування в криптовалюти вимагає грунтовного вивчення ринку та усвідомлення потенційних ризиків.

Основними критеріями для додатку моніторингу криптовалют є зручний та інтуїтивний інтерфейс, можливість отримання даних у реальному часі, функціональність для аналізу та прогнозування ринкових трендів, а також інтеграція з різними API для отримання даних.

В роботі були проаналізовані декілька існуючих програмних продуктів за темою кваліфікаційної роботи. Серед них Trader – система, яка дозволяє прогнозувати коливання валютних курсів на ринку Forex. Система аналізує попередні дані торгівлі, включаючи максимальні, мінімальні ціни, ціну закриття та обсяги угод за день. Walletinvestor — це веб-додаток, що дозволяє вибрати конкретну криптовалюту для відстеження або перегляду інформації про всі доступні криптовалюти у вигляді таблиці. Додаток пропонує прогнози на різні періоди, від двох тижнів до п’яти років. Belinvestor — це веб-додаток, який надає прогнози криптовалют у формі новинної стрічки. Особливість сайту полягає в

тому, що пости старіші за три місяці автоматично видаляються, що унеможливлює перегляд старих прогнозів та оцінку їх точності. NeuroShell - це комплексний набір нейронних мереж, розроблених спеціально для прогнозування валютних курсів на фінансових ринках. Основний принцип роботи цієї системи базується на концепції "чорної скриньки". Elliott Wave Analyser Professional 6.0 — це програмний продукт, призначений для аналізу валютного ринку з використанням теорії хвиль Елліотта (ТХЕ) та стандартних алгоритмів технічного аналізу.

Аналіз існуючих програмних сервісів показав, що більшість з них не відповідають усім зазначеним критеріям, що обумовлює необхідність розробки нового програмного забезпечення.

У другому розділі проаналізовано основне програмне забезпечення для розробки додатку. Було обрано .NET 6, Windows Presentation Foundation (WPF), мову програмування C#, а також CoinGecko API для отримання даних про криптовалюти. Це дозволяє реалізувати усі необхідні функції для моніторингу криптовалют, адаптувати їх під потреби користувачів та забезпечити високу продуктивність і стабільність роботи.

Розроблений програмний продукт реалізує такі функції: отримання та відображення актуальних курсів криптовалют, автоматичне створення графіків, фільтрація та сортування даних, вибір оптимальних курсів для купівлі та продажу, підписка на інформаційні бюлетені з прогнозами. Додаток забезпечує користувачів зручним інтерфейсом, швидкою обробкою даних та доступом до інформації у реальному часі.

Інтерфейс користувача є зручним та інтуїтивно зрозумілим, що забезпечує високу задоволеність користувачів. Додаток має адаптивний дизайн, що дозволяє комфортно використовувати його на різних пристроях з різними розширеннями екрану. Проведене тестування підтвердило високу надійність та безпеку продукту, що дозволяє рекомендувати його для використання трейдерами та інвесторами.

Проте існують певні недоліки та можливості для покращення. Зокрема, додаток наразі підтримує лише операційну систему Windows, тому розширення підтримки на інші платформи, такі як macOS та Linux, могло б залучити ширшу аудиторію. Також існує можливість оптимізації використання ресурсів для покращення продуктивності на старих або слабких системах. Додаткові покращення можуть включати інтеграцію інших API для отримання більш різноманітної інформації про криптовалюти, а також розширення функціоналу для підтримки нових типів даних, таких як NFT та децентралізовані фінанси (DeFi).

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adem Efe Gencer. Decentralization in Bitcoin and Ethereum Networks / Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, Emin Gün Sirer // Financial Cryptography and Data Security (FC). 2018.

2. Двуліт З. Криптовалюта: стан та тенденції розвитку. Економіка та держава, 2019

3. Кравець Д. Теоретичні та практичні аспекти ролі криптовалюти як елементу фінансових активів. Науковии вісник, 2022

4. Лапко О., Солосіч О. Технологія блокчейн: поняття, сфери застосування та вплив на підприємницький сектор. Бизнес информ, 2019, 6 (497): 77-82.

5. Мокін В. Інформаційна технологія прогнозування курсу криптовалют на основі комплексної інженерії ознак. Вісник впі.№ 2: 81-93., 2022.

6. Офіційний сайт Coingecko. [Електронний ресурс]. – Режим доступу: https://www.coingecko.com

7. Офіційний сайт Coinbase [Електронний ресурс]. – Режим доступу: https://www.coinbase.com

8. Офіційний сайт Binance [Електронний ресурс]. – Режим доступу: https://www.binance.com

9. Офіційний сайт Whitebit [Електронний ресурс]. – Режим доступу: URL: https://whitebit.com

10. Офіційний сайт Walletinvestor [Електронний ресурс]. – Режим доступу: https://walletinvestor.com/

11. Офіційний сайт Belinvestor [Електронний ресурс]. – Режим доступу: https://belinvestor.com/charts/

12. Офіційний сайт NeuroShell [Електронний ресурс]. – Режим доступу: https://try.neuroshell.com/index/

13. Офіційний сайт Elliott Wave Analyser Professional [Електронний ресурс]. – Режим доступу: https://wavebasis.com/

14. Офіційний сайт CoinMarketCap [Електронний ресурс]. – Режим доступу: https://coinmarketcap.com/

15. Офіційний сайт CryptoCompare [Електронний ресурс]. – Режим доступу: https://www.cryptocompare.com/

16. Офіційний документація WPF [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-8.0

17. Офіційний документація C# [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview

18. Офіційний документація .Net [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/ru-ru/dotnet/core/whats-new/dotnet-8/overview?WT.mc_id=dotnet-35129-website

19. UML 2.5 Diagrams Overview [Електронний ресурс]. – Режим доступу: https://www.uml-diagrams.org/uml-25-diagrams.html

20. UML Use Case Diagrams [Електронний ресурс]. – Режим доступу: https://www.uml-diagrams.org/use-case-diagrams.html

# ДОДАТОК А

1. HomeView

```
namespace CoinMarketCap.Views;

using System.IO;
using System.Windows.Media;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Globalization;
using System.Net.Http;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using CommunityToolkit.Mvvm.Input;
using CoinGecko.Clients;
using CoinGecko.Entities.Response.Coins;

public static class Extensions
{
    public static List<T> Sort<T, TKey>(
        this IEnumerable<T> list,
        Func<T, TKey> keyExtractor,
        ListSortDirection direction
    )
    {
        return direction == ListSortDirection.Ascending ?
            list.OrderBy(keyExtractor).ToList() :
            list.OrderByDescending(keyExtractor).ToList();
    }
```

```
    public static void Sort<T, TKey>(
        this ObservableCollection<T> collection,
        Func<T, TKey> keyExtractor,
        ListSortDirection direction
    )
    {
        var sorted = direction == ListSortDirection.Ascending ?
            collection.OrderBy(keyExtractor).ToList() :
            collection.OrderByDescending(keyExtractor).ToList();

        for (var i = 0; i < sorted.Count; i++)
        {
            collection.Move(collection.IndexOf(sorted[i]), i);
        }
    }
}

public record Currency(string Name, string Symbol, string Id);

public partial class HomeView
{
    private CoinGeckoClient Api { get; } = new();
    public ObservableCollection<CoinMarkets> Coins { get; set; } = new();
    public decimal MarketCapChangePercentage24h { get; set; }
    public RelayCommand<string> Sort { get; set; }
    public AsyncRelayCommand<Currency> SelectCurrency { get; set; }
    public AsyncRelayCommand DismissErrorCommand { get; set; }
    public string SortedBy { get; set; } = "#";
```

```
        public ListSortDirection SortDirection { get; set; } =
ListSortDirection.Ascending;
        public int CurrentPage { get; set; } = 1;
        public Visibility LoadingVisibility { get; set; } = Visibility.Visible;
        public double BackButtonOpacity => CurrentPage == 1 ? 0 : 1;

        public static List<Currency> Currencies { get; set; } = new()
        {
            new Currency("US Dollar", "$", "usd"),
            new Currency("Euro", "€", "eur")
        };

        public Currency SelectedCurrency { get; private set; } =
Currencies.FirstOrDefault();
        public bool CurrencyDropdownOpen { get; set; }
        public bool CoinsLoaded { get; set; }

        private static Func<CoinMarkets, object> KeyExtractor(string property) =>
property switch
        {
            "#" => c => c.MarketCapRank,
            "Name" => c => c.Name,
            "Price" => c => c.CurrentPrice,
            "24h %" => c => c.PriceChangePercentage24HInCurrency,
            "7d %" => c => c.PriceChangePercentage7DInCurrency,
            "Market Cap" => c => c.MarketCap,
            "Volume(24h)" => c => c.TotalVolume,
            "Circulating Supply" => c => c.CirculatingSupply,
            "Total Supply" => c => c.TotalSupply,
            _ => throw new NotImplementedException()
```

```
        };

        private Func<CoinMarkets, object> UpdateSortParams(string property, bool
reverse)
        {
            var lambda = KeyExtractor(property);

            if (SortedBy == property && reverse)
            {
                SortDirection = SortDirection == ListSortDirection.Ascending ?
                                        ListSortDirection.Descending :
                                        ListSortDirection.Ascending;
            }
            else if (reverse)
            {
                SortDirection = ListSortDirection.Descending;
            }
            SortedBy = property;

            return lambda;
        }

        private void SortObservableCollection(string property)
        {
            var keyExtractor = UpdateSortParams(property, reverse: true);
            Coins.Sort(keyExtractor, SortDirection);
        }

        private Task<List<CoinMarkets>> GetCoins(int size, int page)
        {
```

```
    return Api.CoinsClient.GetCoinMarkets(
        vsCurrency: SelectedCurrency.Id,
        ids: Array.Empty<string>(),
        order: "market_cap_desc",
        perPage: size,
        page: page,
        sparkline: true,
        priceChangePercentage: "24h,7d",
        category: ""
    );
}

private void SelectCoin(object sender, MouseButtonEventArgs e)
{
    LoadingVisibility = Visibility.Visible;
    var selectedCoinId = (string)((Panel)sender).Tag;

    var w = new CoinView(selectedCoinId, SelectedCurrency.Id)
    {
        //WindowState = WindowState.Minimized,
        Width = ActualWidth,
        Height = ActualHeight
    };
    w.FullyRendered += (_, _) =>
    {
        w.Show();
        w.Top += 25;
        w.Left += 25;

        // w.CenterWindowOnScreen();
```

```
        //w.WindowState = WindowState.Normal;
        LoadingVisibility = Visibility.Collapsed;
    };
}

private void ToggleCurrencyDropdown(object sender, MouseEventArgs e)
{
    CurrencyDropdownOpen = !CurrencyDropdownOpen;
}

private async Task OnSelectCurrency(Currency currency)
{
    if (SelectedCurrency == currency)
    {
        CurrencyDropdownOpen = false;
        return;
    }

    LoadingVisibility = Visibility.Visible;
    SelectedCurrency = currency;

    await Dispatcher.InvokeAsync(() =>
    {
        CurrencyDropdownOpen = false;
        var culture = currency.Id switch
        {
            "eur" => "en-IE",
            "usd" or _ => "en-US"
        };
```

```
            Thread.CurrentThread.CurrentCulture =
Thread.CurrentThread.CurrentUICulture = new CultureInfo(culture);
        });

        await FetchData(CurrentPage);
    }

    private async Task FetchData(int page = 1)
    {
        try
        {
            await Dispatcher.BeginInvoke(() =>
            {
                LoadingTitle.Text = "Loading Data";
                LoadingDescription.Text = "Please wait, we are loading coin data";
                LoadingIcon.Visibility = Visibility.Visible;
                DismissErrorButton.Visibility = Visibility.Collapsed;
                LoadingVisibility = Visibility.Visible;
            });
            var globalTask = Api.GlobalClient.GetGlobal();
            var coins = await GetCoins(size: 30, page);
            var globalData = await globalTask;
            var keyExtractor = UpdateSortParams(SortedBy, reverse: false);

            await Dispatcher.BeginInvoke(() =>
            {
                MarketCapChangePercentage24h =
globalData.Data.MarketCapChange;
                Coins.Clear();
                coins.Sort(keyExtractor, SortDirection).ForEach(c => Coins.Add(c));
```

```
                CoinsLoaded = true;
                LoadingVisibility = Visibility.Collapsed;
            });
        }
        catch (HttpRequestException)
        {
            Dispatcher.Invoke(() =>
            {
                LoadingTitle.Text = "Too many requests";
                LoadingDescription.Text = "Please wait for the API to be available
again";
                LoadingIcon.Visibility = Visibility.Collapsed;
                DismissErrorButton.Visibility = Visibility.Visible;
            });
        }
    }

    private async void PreviousPage(object sender, RoutedEventArgs ea)
    {
        if (CurrentPage == 1) return;

        var page = Math.Max(1, CurrentPage - 1);
        await FetchData(page);
        Scroller.ScrollToTop();
        CurrentPage = page;

        Scroller.Focus();
    }

    private async void NextPage(object sender, RoutedEventArgs ea)
```

```
        {
            var page = Math.Min(CurrentPage + 1, 100);
            await FetchData(page);
            Scroller.ScrollToTop();
            CurrentPage = page;
            Scroller.Focus();
        }

        public HomeView()
        {
            Thread.CurrentThread.CurrentCulture =
Thread.CurrentThread.CurrentUICulture = new CultureInfo("en-US");
            InitializeComponent();
            FontFamily = new
FontFamily(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Fonts",
"#Inter"));

            var workArea = SystemParameters.WorkArea;
            Height = workArea.Height * 0.96;
            Title = "CoinMarketCap - Homepage";

            Sort = new RelayCommand<string>(SortObservableCollection);
            SelectCurrency = new
AsyncRelayCommand<Currency>(OnSelectCurrency);
            DismissErrorCommand = new AsyncRelayCommand(() =>
FetchData(CurrentPage));
            DismissErrorButton.Visibility = Visibility.Collapsed;

            PreviewMouseDown += (_, _) =>
            {
```

```
            if (CurrencyDropdownOpen &&
                !CurrenciesButton.IsMouseOver &&
                !CurrenciesBorder.IsMouseOver)
            {
                CurrencyDropdownOpen = false;
            }
        };

        Task.Run(async () =>
        {
            await FetchData();
        });
    }
}
```

2. ErrorDialog

```
namespace CoinMarketCap.Views;

using System.IO;
using System.Windows.Media;

public partial class ErrorDialog : Primitives.Window
{
    public ErrorDialog()
    {
        InitializeComponent();
        FontFamily = new
FontFamily(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Fonts",
"#Inter"));
    }
```

```
}
```

3. CoinView

```
namespace CoinMarketCap.Views;

using System;
using System.Collections.ObjectModel;
using System.Diagnostics;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;
using System.Windows.Media;
using CommunityToolkit.Mvvm.Input;
using CoinGecko.Entities.Response.Coins;
using CoinGecko.Clients;

public record PriceData(
    decimal Price,
    decimal Volume,
    decimal MarketCap,
    DateTimeOffset Timestamp,
    string Date,
    string Time
);
```

```
public record TimeRange(TimeSpan Range, string Name);

public partial class CoinView
{
    public string Currency { get; set; }
    public CoinMarkets CoinData { get; private set; }
    public Thickness YAxisMargin { get; private set; }
    public Thickness YOpenMargin { get; private set; }
    public string YPointPrice { get; private set; }
    public Point ChartPoint { get; set; }
    private Dictionary<int, Point> PathPoints { get; set; } = new();
    public ObservableCollection<string> DescriptionRows { get; } = new();
    public Visibility LoadingVisibility { get; set; } = Visibility.Collapsed;
    public double LoadingOpacity { get; set; } = 1;
    public ObservableCollection<string> Timesteps { get; } = new();
    public TimeRange TimeRange { get; set; } = new(TimeSpan.FromDays(30),
"30D");
    public List<TimeRange> TimeRangeNames { get; } = new()
    {
        new TimeRange(TimeSpan.FromDays(1), "1D"),
        new TimeRange(TimeSpan.FromDays(7), "7D"),
        new TimeRange(TimeSpan.FromDays(30), "1M"),
        new TimeRange(TimeSpan.FromDays(90), "3M"),
        new TimeRange(TimeSpan.FromDays(365), "1Y")
    };
    public string FormattedOpen => FormatMixed(Open,
CultureInfo.CurrentCulture);
    readonly DecimalToVariablePrecision _variablePrecisionConverter = new();
    public AsyncRelayCommand<TimeRange> ChangeTabCommand { get; }
```

```
        public RelayCommand DismissErrorCommand { get; private set; }

        public PriceData PriceAtPoint
        {
            get
            {
                var price = PriceData.GetValueOrDefault(decimal.Round((decimal)ChartPoint.X, 3));
                if (price is not null)
                {
                    _priceAtPoint = price;
                }

                return _priceAtPoint;
            }
        }

        public string ChartPointFill => PriceAtPoint?.Price >= Open ? "#16C784" : "#EA3943";

        public string Sparkline
        {
            get
            {
                if (Data is null || !ResyncChart) return string.Empty;

                decimal index = 0;
                var scale = _hResolution / (Max - Min);

                return Data.Prices.Aggregate("", (path, price) =>
```

```
        {
            var instruction = index == 0 ? 'M' : 'L';
            path = string.Format(
                CultureInfo.InvariantCulture,
                "{0} {1}{2} {3:0.###}",
                path,
                instruction,
                index,
                decimal.Round((Max - price?[1] ?? 0) * scale, 3)
            );
            index += Step;
            return path;
        }) +
        string.Format(
            CultureInfo.InvariantCulture,
            " {0}{1} {2:0.###}",
            'L',
            index,
            (Max - Open) * scale
        );
    }
}

public Brush Stroke
{
    get
    {
        if (Data is null) return Brushes.Transparent;

        var ratio = (double)(1 - (Open - Min) / (Max - Min));
```

```
        var green = (Color)ColorConverter.ConvertFromString("#16C784")!;
        var red = (Color)ColorConverter.ConvertFromString("#EA3943")!;

        GradientStopCollection collection = new()
        {
            new GradientStop(green, 0),
            new GradientStop(green, ratio),
            new GradientStop(red, ratio),
            new GradientStop(red, 1.0)
        };

        return new LinearGradientBrush(collection, angle: 90);
    }
}

public Brush Fill
{
    get
    {
        if (Data is null) return Brushes.Transparent;

        var ratio = (double)(1 - (Open - Min) / (Max - Min));

        var green = (Color)ColorConverter.ConvertFromString("#16C784")!;
        var red = (Color)ColorConverter.ConvertFromString("#EA3943")!;

        GradientStopCollection collection = new()
        {
            new GradientStop(green, -0.95),
```

```
            new GradientStop(Color.FromArgb(3, 131, 214, 183), ratio),
            new GradientStop(Color.FromArgb(3, 247, 153, 159), ratio),
            new GradientStop(red, 1.95)
        };

        return new LinearGradientBrush(collection, angle: 90);
    }
}

public List<string> YPrices
{
    get
    {
        if (Data is null) return new List<string>();

        var step = (Max - Min) / 6;

        return Enumerable.Range(0, 7)
            .Select(i => Max - i * step)
            .Select(v => FormatMixed(v, CultureInfo.CurrentCulture))
            .ToList();
    }
}

public string VolumeSparkline
{
    get
    {
        if (Data is null) return string.Empty;
```

```
            decimal index = 0;
            var max = Data.TotalVolumes.MaxBy(p => p[1])[1] ?? 0;
            var min = Data.TotalVolumes.MinBy(p => p[1])[1] ?? 0;

            var scale = 25 / (max - min);

            const char instruction = 'L';

            return $"M0 {(max - min) * scale} " + Data.TotalVolumes.Aggregate("",
(path, volume) =>
            {

                path = string.Format(
                    CultureInfo.InvariantCulture,
                    "{0} {1}{2} {3:0.####}",
                    path,
                    instruction,
                    index,
                    (max - volume[1]) * scale - 12
                );
                index += Step;
                return path;
            }) + $" L{index + 3} {(max - min) * scale}";
        }
    }

    private const decimal _hResolution = 360;
    private const decimal _wResolution = 740;
    private readonly CoinGeckoClient _api = new();
    private readonly string _coin;
```

```
        private PriceData _priceAtPoint;
        private MarketChartById Data { get; set; }
        private Dictionary<decimal, PriceData> PriceData { get; set; } = new();
        private decimal Step { get; set; } = 2.75m;
        private decimal Min { get; set; }
        private decimal Max { get; set; }
        private decimal Open { get; set; }
        private bool ResyncChart { get; set; }
        private TimeRange OldTimeRange { get; set; } = new(TimeSpan.FromDays(30), "30D");

        string FormatMixed(decimal number, CultureInfo culture)
        {
            return number switch
            {
                > 9_999 => number.ToString("0,.00K", culture),
                > 999 => number.ToString("N0", culture),
                _ => (string)_variablePrecisionConverter.Convert(number, typeof(string), null, culture)
            };
        }

        private async Task FetchChart(bool initial = false)
        {
            Loading(initial);
            var from = DateTimeOffset.UtcNow - TimeRange.Range;
            var to = DateTimeOffset.UtcNow;
            MarketChartById d;

            try
```

```
            {
                d = await _api.CoinsClient.GetMarketChartRangeByCoinId(
                    id: _coin,
                    vsCurrency: Currency,
                    from.ToUnixTimeSeconds().ToString(),
                    to.ToUnixTimeSeconds().ToString()
                );
            }
            catch (HttpRequestException)
            {
                DismissErrorCommand = new RelayCommand(() => DoneLoading(shouldClose: initial));
                Error();
                return;
            }

            Step = _wResolution / d.Prices.Length;

            var open = d.Prices[0][1] ?? 0;
            var max = d.Prices.MaxBy(p => p[1])[1] ?? 0;
            var min = d.Prices.MinBy(p => p[1])[1] ?? 0;

            var dateTimeCulture = new CultureInfo("en-US");

            await Dispatcher.InvokeAsync(() =>
            {
                ResyncChart = false;
                PriceData = new();

                for (var i = 0; i < d.Prices.Length; i++)
```

```
            {
                var timestamp =
DateTimeOffset.FromUnixTimeMilliseconds((long)(d.Prices[i][0] ?? 0)) +
TimeSpan.FromHours(2);
                PriceData.Add(
                    decimal.Round(i * Step, 3),
                    new PriceData(
                        d.Prices[i][1] ?? 0,
                        d.TotalVolumes[i][1] ?? 0,
                        d.MarketCaps[i][1] ?? 0,
                        timestamp,
                        timestamp.DateTime.ToString("d", dateTimeCulture),
                        timestamp.DateTime.ToString("T", dateTimeCulture)
                    )
                );
            }

            Open = open;
            Max = max;
            Min = min;
            Data = d;
            YAxisMargin = new Thickness(0, 0, 0,
(double)decimal.Round(_hResolution / (max - min) * (max - min) / 7, 3) + 2);
            YOpenMargin = new Thickness(0, (double)(_hResolution / (max - min))
* (double)(max - open) + 24, 0, 0);

            ResyncChart = true;

            PathPoints = ChartPath.Data
            .GetFlattenedPathGeometry()
```

```
.Figures.SelectMany(f => f.Segments)
.SelectMany(s => ((PolyLineSegment)s).Points)
.DistinctBy(p => (int)p.X)
.ToDictionary(p => (int)p.X, p => p);

int steps = TimeRange.Range.TotalDays switch
{
   >= 365 => 12,
   >= 7 => 7,
   1 or _ => 8
};

var timestep = (to - from) / steps;

var timeFormat = TimeRange.Range.TotalDays switch
{
   1 => "h:mm tt",
   >= 7 => "MMM d",
   _ => "T"
};

Timesteps.Clear();

var dateTimeSteps = Enumerable.Range(0, steps + 1)
                .Select(i => from + TimeSpan.FromHours(3) + i * timestep)
                .ToList();

for (var i = 0; i < dateTimeSteps.Count - 1; i++)
{
   if (TimeRange.Range >= TimeSpan.FromDays(7) &&
```

```
                    dateTimeSteps[i].Month != dateTimeSteps[i + 1].Month
                )
                {
                    Timesteps.Add(dateTimeSteps[i + 1].ToString("MMM"));
                }
                else if (TimeRange.Range == TimeSpan.FromDays(1) &&
                        dateTimeSteps[i].DayOfYear != dateTimeSteps[i + 1].DayOfYear
                )
                {
                    Timesteps.Add(dateTimeSteps[i + 1].ToString("MMM d"));
                }
                else
                {
                    Timesteps.Add(dateTimeSteps[i].ToString(timeFormat, dateTimeCulture));
                }
            }

            DoneLoading();
        });
    }

    private async Task FetchData()
    {
        var data = (await _api.CoinsClient.GetCoinMarkets(
            vsCurrency: Currency,
            ids: new[] { _coin },
            order: "market_cap_desc",
            perPage: 1,
```

```
        page: 0,
        sparkline: false,
        priceChangePercentage: "24h,7d",
        category: ""
    )).FirstOrDefault();

    Dispatcher.Invoke(() => CoinData = data);
}

private static IEnumerable<string> StringToTextBlocks(string input)
{
    var tagFound = false;
    StringBuilder sb = new();

    foreach (var t in input)
    {
        if (t == '<') tagFound = true;
        if (!tagFound) sb.Append(t);
        if (t == '>') tagFound = false;
    }

    return sb.ToString().Split("\r\n\r\n");
}

private void Loading(bool initial = false)
{
    Dispatcher.Invoke(() =>
    {
        LoadingTitle.Text = "Loading Data";
        LoadingDescription.Text = "Please wait, we are loading coin data";
```

```
            if (initial) LoadingOpacity = 1;
            else LoadingOpacity = 0.9;
            DismissErrorButton.Visibility = Visibility.Collapsed;
            LoadingIcon.Visibility = Visibility.Visible;
            LoadingVisibility = Visibility.Visible;
        });
    }

    private void DoneLoading(bool shouldClose = false)
    {
        if (shouldClose)
        {
            Close();
            return;
        }
        Dispatcher.Invoke(() =>
        {
            LoadingVisibility = Visibility.Collapsed;
        });
    }

    private void Error()
    {
        Dispatcher.Invoke(() =>
        {
            TimeRange = OldTimeRange;
            LoadingTitle.Text = "Too many requests";
            LoadingDescription.Text = "Please wait for the API to be available
again";
            LoadingIcon.Visibility = Visibility.Collapsed;
```

```
            DismissErrorButton.Visibility = Visibility.Visible;
            LoadingVisibility = Visibility.Visible;
        });
    }

    public CoinView(string coin, string currency)
    {
        InitializeComponent();
        FontFamily = new FontFamily(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Fonts", "#Inter"));

        _coin = coin;
        Currency = currency;

        ChangeTabCommand = new AsyncRelayCommand<TimeRange>(async range =>
        {
            OldTimeRange = TimeRange;
            TimeRange = range;
            await FetchChart();
        });

        DismissErrorCommand = new RelayCommand(() => DoneLoading());

        Task.Run(async () =>
        {
            var descriptionTask = _api.CoinsClient.GetAllCoinDataWithId(_coin, "false", false, false, false, false, false);
            await Task.WhenAll(FetchChart(initial: true), FetchData());
```

```
            Dispatcher.Invoke(() =>
            {
                NotifyFullyRendered();
                Title = $"CoinMarketCap - {CoinData?.Name} Price Chart";
            });

            var descriptionData = await descriptionTask;
            var descriptionRows = StringToTextBlocks(descriptionData.Description["en"]);

            Dispatcher.Invoke(() =>
            {
                foreach (var t in descriptionRows)
                {
                    DescriptionRows.Add(t);
                }
            });
        });
    }

    private void ChartMouseMove(object sender, MouseEventArgs e)
    {
        var position = e.GetPosition(ChartPath);
        var pathPoint = PathPoints.GetValueOrDefault((int)position.X);

        double width = ChartPanel.ActualWidth;
        double timestampWidth = Math.Max(XAxisTimestampPanel.ActualWidth, 150);
```

```
if (pathPoint != default)
{
    ChartPoint = pathPoint;

    var xTimestampMargin = XAxisTimestampPanel.Margin;
    xTimestampMargin.Left = Math.Clamp(
        position.X - timestampWidth / 2,
        0,
        width - timestampWidth - 6
    );
    XAxisTimestampPanel.Margin = xTimestampMargin;
}

var yPointMargin = YPointPricePanel.Margin;
var chartTooltipMargin = ChartTooltip.Margin;

if (position.X > width - 280)
{
    chartTooltipMargin.Left = position.X - 220;
}
else
{
    chartTooltipMargin.Left = position.X + 80;
}

if (position.Y < 110)
{
    chartTooltipMargin.Top = position.Y + 52;
}
else
```

```
            {
                chartTooltipMargin.Top = position.Y - 92;
            }

            HorizontalChartLine.Y1 = HorizontalChartLine.Y2 = yPointMargin.Top =
position.Y + 24;

            ChartTooltipWrapper.Margin = chartTooltipMargin;
            ChartTooltip.Margin = chartTooltipMargin;
            YPointPricePanel.Margin = yPointMargin;

            YPointPrice = FormatMixed(
                Min + (Max - Min) *
                (1 - (decimal)(position.Y / ChartPath.ActualHeight)),
                CultureInfo.CurrentCulture
            );
        }

        private void FollowOnTwitter(object sender, MouseButtonEventArgs e)
        {
            Process.Start(
                new ProcessStartInfo(
                    "cmd", $"/c start https://twitter.com/CoinMarketCap/"
                )
                { CreateNoWindow = true }
            );
        }
    }
```