ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмна платформа для відображення курсів криптовалют та аналітичної інформації»

на здобуття освітнього ступеня бакалавра

зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)

освітньо-професійної програми Інформаційні системи та технології
(назва)

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

_____ Максим КОВАЛЕНКО
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. ІСД-42
Максим КОВАЛЕНКО
Ім'я, ПРІЗВИЩЕ

Керівник: Іван ШАХМАТОВ
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти бакалавр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
_____Каміла СТОРЧАК
«____» _____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Коваленко Максим Олексійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Програмна платформа для відображення курсів криптовалют та аналітичної інформації.

керівник кваліфікаційної роботи Іван ШАХМАТОВ
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «___» _____ 20__ р. № ___

2. Строк подання кваліфікаційної роботи «___» _____ 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, основи розробки додатків під Windows платформу, методології аналізу даних, властивості мережевого протоколу HTTP, методи інтеграції зі сторонніми API.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Пошук та аналіз існуючих сервісів для отримання інформації криптовалют

2. Створення структури програмної платформи та її оформлення

3. Розробка додатку під платформу Windows

4. Тестування та перевірка функціоналу додатку

5. Перелік ілюстративного матеріалу: презентація

6. Дата видачі завдання «27» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір відповідної науково-технічної літератури	28.02-02.03.2025	
2	Дослідження існуючих програмних засобів	03.03-08.03.2025	
3	Дослідження існуючих API	09.03-23.03.2025	
4	Створення структури програмної платформи	24.03-03.04.2025	
5	Розробка додатку відповідно до структури	04.04-25.04.2025	
6	Тестування додатку	26.04-12.05.2025	
7	Оформлення роботи: вступ, висновки, реферат	13.05-20.05.2025	
8	Розробка демонстраційних матеріалів	21.05-26.05.2025	

Здобувач вищої освіти _____ Максим КОВАЛЕНКО
(підпис) (Ім’я, ПРІЗВИЩЕ)

Керівник
кваліфікаційної роботи _____ Іван ШАХМАТОВ
(підпис) (Ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 57 стор., 27 рис., 1 табл., 21 джерел.

Метою роботи – створити метод збору актуальної інформації курсу криптовалюти, реалізувати його у вигляді програмного засобу та дослідити ефективність.

Об'єкт дослідження – процес збору та агрегації даних про курс криптовалюти.

Предмет дослідження – методи і програмний засіб збору даних про курс криптовалюти

Короткий зміст роботи: Криптовалюти стали важливим інструментом у сучасній економіці, їхня роль значно зросла завдяки розвитку інформаційних технологій та глобальній цифровізації. Ринок криптовалют привертає увагу як професійних трейдерів так і початківців, що створює попит на ефективні засоби моніторингу, аналізу та прогнозування. Реалізований метод збору даних про курси криптовалют дозволяє отримувати актуальну інформацію, зменшувати ризики та полегшувати процес торгів на криптовалютних біржах для недосвідчених користувачів. У результаті виконаної роботи створено ефективний програмний продукт для моніторингу криптовалют, який відповідає сучасним вимогам. Програмний засіб забезпечує зручний інтерфейс, швидку обробку даних та доступ до інформації в режимі реального часу, що сприяє обґрунтованому прийняттю інвестиційних рішень.

КЛЮЧОВІ СЛОВА: КРИПТОВАЛЮТА, WPF, .NET, COINMARKETCAP, API, C#, БД, DeFi

ЗМІСТ

ВСТУП

Актуальність теми. Гроші завжди відігравали ключову роль у суспільстві людей через потребу здійснювати комерційні та торговельні операції. Розвиток інформаційних технологій спровокував появу нових фінансових інструментів, таких як криптовалюти, які базуються на технології блокчейн. З моменту свого виникнення криптовалютний ринок зацікавлює багатьох професійних трейдерів і початківців, які шукають практичні моделі прогнозування. Для прийняття коректних рішень у цій сфері необхідно використовувати різні методи аналізу, враховуючи специфіку ринку криптовалют.

Економіка ґрунтується не лише на фактичних даних, але й на очікуваннях і прогнозах, що охоплюють політичну ситуацію, розвиток технологій, стан багатьох економічних секторів, погодні умови, видобуток корисних копалин, а також вартість різноманітних активів і валют. Для успішної діяльності на ринку необхідно мати актуальну інформацію та своєчасні прогнози, які допоможуть передбачати майбутні зміни та обирати оптимальні стратегії.

У 21 столітті, що стало епохою інформаційних технологій, почали виникати перші криптовалютні біржі - платформ для торгівлі криптовалютами. Сьогодні існує понад 2500 різних криптовалют, а сім з них мають капіталізацію понад мільярди доларів. Незважаючи на скептицизм деяких експертів, криптовалютні біржі надають потужні можливості для заробітку. Існують два основні підходи до торгівлі криптовалютами: криптообмінники та криптовалютні біржі. Криптообмінники виступають стороною в торгах, тоді як криптовалютні біржі забезпечують платформу для торгівлі між користувачами. Тому найактуальнішим є потреба в швидкому доступі до актуальної інформації про курси криптовалют.

Метою роботи – створити метод збору актуальної інформації курсу криптовалют, реалізувати його у вигляді програмного забезпечення та дослідити його ефективність.

Завданнями бакалаврської роботи є

1) проаналізувати становлення криптовалют та засади їх функціонування;

2) розглянути принципи роботи криптовалют черз криптовалютні біржі

3) проектування системи обробки та збору даних про курси криптовалют.

Об’єкт дослідження – процес збору та агрегації даних про курс криптовалюти.

Предмет дослідження – програмна платформа збору та відображення даних про курс криптовалют.

Методи дослідження. Було застосовано як загальнонаукові, так і спеціальні методи, що дозволяють комплексно розв’язати проблемні завдання у вибраному напрямі дослідження ринку криптовалют.

Наукова новизна – розробка додатку для обробки даних з криптовалютних бірж з метою їх об'єднання та подальшої аналітичної обробки.

Практична цінність отриманих результатів полягає в тому, що розроблений метод збору даних про курс криптовалют забезпечив отримання актуальної інформації, зниження ризиків та спрощення процесу торгівлі для недосвідчених користувачів. Було здійснено програмну реалізацію цього методу, яка може використовуватися для аналізу та прогнозування курсу криптовалют.

Апробація результатів та публікації:

1) Коваленко М. О. «ІОТ ДЛЯ РОЗУМНИХ МІСТ ТА ПРОМИСЛОВОСТІ» Тези доповіді на V Міжнародну науково-технічну конференцію «Сучасний стан та перспективи розвитку IoT» 18 квітня 2024 року.
2) Коваленко М. О. «ЗАСТОСУВАННЯ BIG DATA В РІЗНИХ СФЕРАХ» Тези доповіді на VI Міжнародну науково-технічну конференцію «Сучасний стан та перспективи розвитку IoT» 15 квітня 2025 року.

Структура роботи:

Розділ 1 Визначення, принципи та функціонування криптовалют. Детально описує основні принципи та функціонування криптовалют, їхні переваги та недоліки.

Розділ 2 Інструменти розробки. Присвячено інструментам розробки програмного забезпечення, зокрема графічному фреймворку WPF, платформі .NET Framework та мові програмування C#. Розглянуто популярні API для роботи з криптовалютами.

Розділ 3 Програмна реалізація. Описує програмну реалізацію та тестування проекту.

1 ВИЗНАЧЕННЯ, ПРИНЦИПИ ТА ФУНКЦІОНУВАННЯ КРИПТОВАЛЮТ

1.1 Криптовалюта та її використання в світі

Криптовалюти – це цифрові активи, які застосовують методи криптографії для забезпечення своєї безпеки. Ці активи переважно використовуються для покупки товарів та послуг, але деякі також мають спеціальні правила для користувачів. Криптовалюти не мають власної внутрішньої вартості і не можуть бути обміняні на інші товари, такі як золото. Відмінною рисою є те, що вони не контролюються жодною центральною установою і поки що не визнані як офіційний засіб платежу.

На сьогоднішній день, кількість користувачів криптовалют сягає понад 100 мільйонів по всьому світу, більшість з яких інвестують у ці активи. Загалом, потреба у криптовалютах не є високою, оскільки національні валюти ефективно виконують свої функції.

Перш за все, криптовалюти дозволяють здійснювати покупки в інтернеті без необхідності вказувати особисті дані, проте це не забезпечує повну анонімність. Кожен користувач має анонімний псевдонім, який може бути відстежений правоохоронними органами. В умовах зростаючих побоювань щодо крадіжки особистих даних, криптовалюти можуть надати деякі переваги з точки зору конфіденційності.

Другою важливою перевагою є відсутність фінансових посередників, що знижує транзакційні витрати для користувачів. Зниження ризику розголошення конфіденційної інформації з банків стимулює багатьох вибирати криптовалюти як альтернативу традиційним фінансовим системам.

Третє, криптовалюти можуть забезпечувати додаткові переваги, як-от обмежене право власності або право голосу в управлінні організацією, що фінансується через криптовалюту. Також можливо, що вони надають права на

частку у фізичних активах, наприклад, у власності на нерухомість чи мистецькі твори.

1.2 Загальна характеристика криптовалюти

Обіг криптовалют є децентралізованим вони не підтримуються або не контролюються центральним органом влади. Замість цього вони працюють через мережу комп'ютерів. Такі ринки дозволяють торгувати криптовалютами на біржах та зберігати їх у електронних гаманцях.

Технологія блокчейну є основою більшості сучасних криптовалют. Вона базується на постійно оновлюваній публічній "книзі" транзакцій. Ця технологія дозволяє покупцям та продавцям взаємодіяти напряму, а також надає доступ до записів взаємодіючих сторін. Однією з основних переваг є те, що ця "книга" не контролюється жодним центральним органом і для реєстрації транзакцій потрібно підтвердження від обох сторін. Крім того, блокчейн зберігається у різних формах, таких як файли або бази даних та містить ланцюг взаємопов'язаних блоків, кожен з яких посилається на попередній блок за допомогою хеш-коду, що генерується криптографічним алгоритмом.

Ян Ланскі, чеський дослідник у галузі криптовалют, визначає криптовалюту як систему, яка має відповідати шести критеріям:

1. Система не потребує центрального керівного органу; її стабільність забезпечується за допомогою децентралізованого консенсусу.
2. Система реєструє криптовалютні одиниці та їх власності.
3. Система встановлює правила створення нових одиниць криптовалюти. Вона визначає умови їхнього створення та правила за допомогою яких визначається власник нових одиниць.
4. Право власності на одиниці криптовалюти може бути доведено тільки за допомогою криптографічних методів.

5. Система дозволяє здійснювати транзакції, в ході яких змінюється власник криптовалютних одиниць. Транзакцію може ініціювати тільки особа, яка наразі володіє цими одиницями.

6. У випадку подання двох різних інструкцій про передачу одних і тих самих одиниць, система реалізує тільки одну з них.

Ідея криптовалюти зародилася в роботах американського криптографа Девіда Чаума, який у 1983 році представив концепцію криптографічних електронних грошей, названих ecash. До числа піонерів також входять Вей Дай, Нік Сабо та Адам Бек, які розробили основні принципи блокчейну, що стали основою для усіх сучасних криптовалют.

Перша широко відома криптовалюта – біткоїн, була створена в 2008-2009 роках невідомою організацією під псевдонімом Сатосі Накамото. Науково-технічний розвиток та швидка діджиталізація внесли свій вклад у розвиток криптографії та становлення криптовалют як нової форми фінансових інструментів. Також на розвиток вплинув бум майнінгу 2018-2020 років та пандемія COVID-19, що призвів до складної економічної ситуації.

1.3 Принцип роботи криптовалюти. Переваги та недоліки

В основі більшості криптовалют лежить технологія блокчейн, яка забезпечує безпечний та децентралізований спосіб здійснення транзакцій. У цій системі покупці та продавці взаємодіють напряму без посередників. Записи про всі транзакції зберігаються в постійно оновлюваній базі даних, яка називається блокчейном. Кожен користувач володіє унікальним криптографічним ключем, що гарантує безпеку транзакцій та контроль над цифровими активами. Транзакції відбуваються за згодою обох сторін і записуються до блокчейну. Цей запис не підлягає зміні, оскільки кожен блок пов'язаний з попереднім за допомогою криптографічного хешу.

Нові блоки додаються до ланцюжка послідовно, а їх додавання ґрунтується на принципі консенсусу. Це означає, що більшість учасників мережі повинні підтвердити дійсність транзакцій, перш ніж вони будуть включені до блокчейну.

Основні операції з криптовалютами:

1) Зберігання

Криптовалюта може зберігатися в електронних гаманцях (програмних або апаратних) чи на криптобіржах.

2) Обмін

Обмін криптовалют можливий як на інші цифрові активи, так і на фіатні валюти через криптобіржі.

3) Транзакції

Криптовалюта використовується для здійснення платежів і переказів коштів.

4) Майнінг – це процес створення нових блоків і підтвердження транзакцій у блокчейні. Майнери отримують винагороду у вигляді криптовалюти.

Технологія блокчейн та криптовалюти мають ряд переваг, таких як:

1) Децентралізація

2) Прозорість

3) Безпека

4) Низькі комісії

Важливо зазначити, що криптовалюти також мають певні ризики, пов'язані з їх волатильністю та спекулятивним характером. Перед тим, як інвестувати в криптовалюти, важливо ретельно вивчити цю сферу та розуміти всі пов'язані з нею ризики.

1.4 Види та найпопулярніші криптовалюти

Станом на квітень 2024 року загальна капіталізація ринку криптовалют сягає 2,51 трильйона доларів США. Цей показник значно виріс з моменту його першого перетину позначки 1 мільярд доларів у 2013 році, досягнувши максимуму майже 3 трильйонів наприкінці 2021 року.

Більшу частину ринку (близько 85%) складають 10 найпопулярніших криптовалют. Нижче представлені деякі з них з їхніми ключовими характеристиками:

1. Біткоїн (капіталізація ≈ 1,3 трлн $) – це перша та найвідоміша криптовалюта, розроблена у 2008 році. Вона має обмежену кількість у 21 мільйон монет, що робить її стійкою до інфляції. Зараз видобуто 19 мільйонів біткоїнів, а з часом винагорода за майнінг зменшується. Обробка одного блоку біткоїну займає близько 10 хвилин.

2. Ethereum (капіталізація ≈ 342,64 млрд $) – це платформа, створена у 2015 році Віталієм Бутеріним. Вона відома підтримкою смарт-контрактів – самовиконуваних угод із визначеними умовами. Смарт-контракти Ethereum дозволяють автоматизувати різні процеси, зокрема фінансові операції, системи голосування та децентралізовані додатки (DApps). Вони створюються мовою програмування Solidity, а час обробки одного блоку становить 2,5 хвилини.

3. Tether (капіталізація ≈ 161 млрд $) – це стейблкоїн, тобто криптовалюта, курс якої прив’язаний до вартості певного активу, зокрема долара США. Створений у 2014 році, Tether забезпечує стабільність купівельної спроможності та високу ліквідність. Він функціонує на різних блокчейн-платформах, таких як Bitcoin, Ethereum і Tron. Найпопулярнішою версією є USDT, який підтримує співвідношення 1:1 до долара США. Компанія Tether Limited заявляє, що кожен випущений USDT забезпечений еквівалентною сумою в доларах США.

4. BNB або Binance Coin (капіталізація 67,7 млрд $) - це криптовалюта, розроблена у 2017 році найбільшою криптобіржею Binance. Спочатку BNB був токеном на базі Ethereum, проте згодом перейшов на власний блокчейн Binance Smart Chain. Власники BNB отримують знижки на комісії та бонуси при торгівлі на біржі Binance. Крім того, періодично проводиться спалювання токенів, що зменшує їхню кількість в обігу та потенційно підвищує їхню вартість.

5. USD Coin (капіталізація ≈ 52 млрд $) - ще один стейблкоїн, вартість якого прив’язана до долара США у співвідношенні 1:1. Його було засновано у 2018 році компанією Circle за підтримки криптобіржі Coinbase. USDC працює на таких

блокчейнах, як Ethereum, Algorand та Solana. Circle заявляє, що кожен випущений USDC повністю забезпечений доларовими резервами або іншими схваленими фінансовими активами.

1.5 Програмні засоби прогнозування та моніторингу курсу криптовалют

Trader є системою, яка призначена для прогнозування змін валютних курсів на ринку Forex. Система здійснює аналіз історичних торгових даних, зокрема максимальних і мінімальних цін, цін закриття та щоденних обсягів угод. Аналіз базується на застосуванні різноманітних алгоритмів, зокрема трьох видів ковзних середніх (лінійне, експоненціальне, і з ваговими коефіцієнтами), MACD-гістограм, а також таких індикаторів, як Williams R%, OBV, RSI, CandleSticks та Point & Figure. Користувачам надається можливість 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.

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).

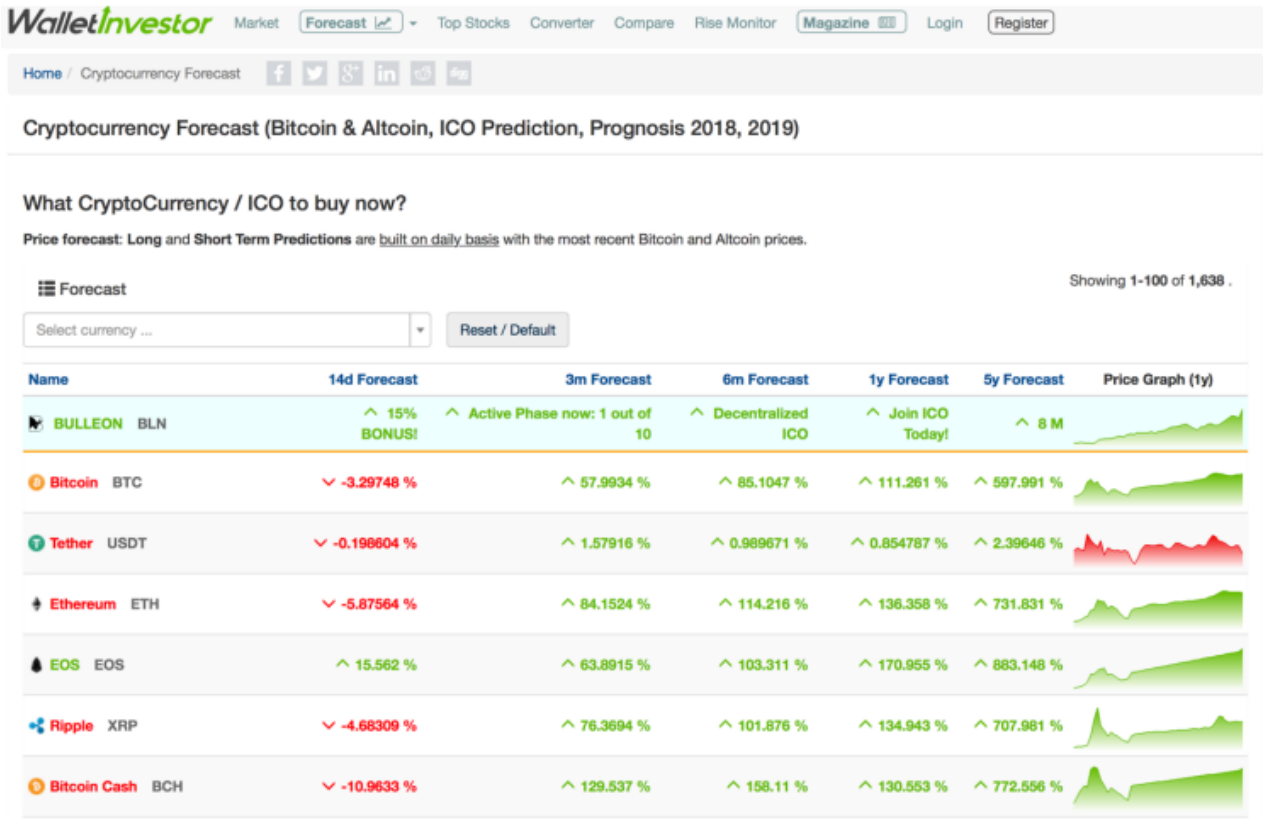


Рис. 1.1 Walletinvestor (Офіційний сайт Walletinvestor https://walletinvestor.com/markets)

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).

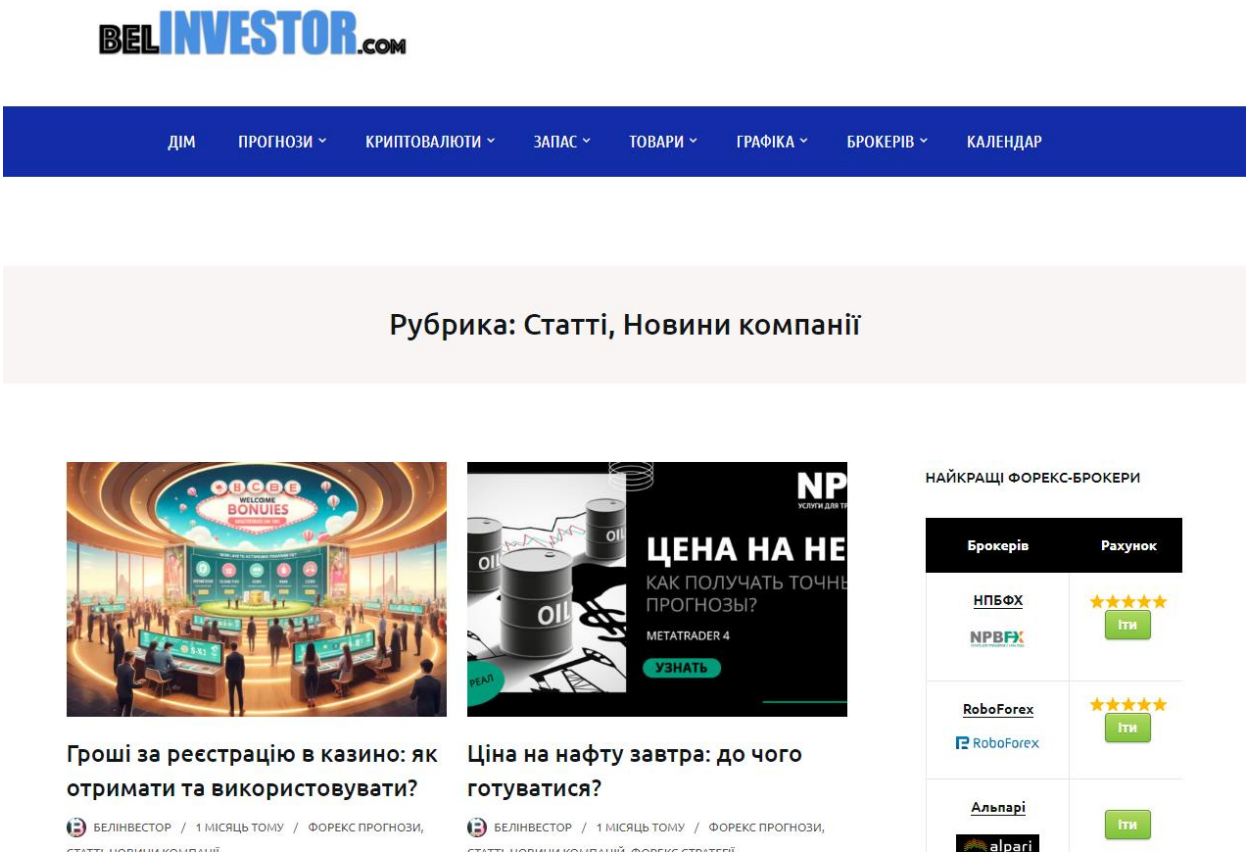


Рис. 1.2 Belinvestor (Офіційний сайт Belinvestor https://belinvestor.com/)

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).

Додатковим сигналом на користь зростання курсу BTC/USD стане тест зони підтримки на графіку ціни монети. Другим сигналом є відскок від низхідної межі бичачого каналу. Раніше був отриманий слабкий сигнал на продаж криптовалюти Bitcoin. Сигнал сформувався за рахунок перетину сигнальних ліній на рівні 67505. Найближча зона опору для BTC/USD на графіку розташована на позначці 70035. Зона підтримки біткоіни розташована за адресою 68590.

Рис. 1.3 Приклад прогнозу на сайті Belinvestor (Офіційний сайт Belinvestor https://belinvestor.com/)

NeuroShell – це комплексний набір нейронних мереж, спеціпльно розроблених і навчених спеціально для прогнозування курсів валют на фінансових ринках. Система використовує принцип "чорної скриньки", що означає, що користувачеві не потрібно розуміти алгоритми або деталі обробки даних. Мінімалістичний інтерфейс спрощує налаштування параметрів і отримання результатів, що робить програму доступною для користувачів із будь-яким рівнем досвіду у фінансових операціях (рис. 1.4).

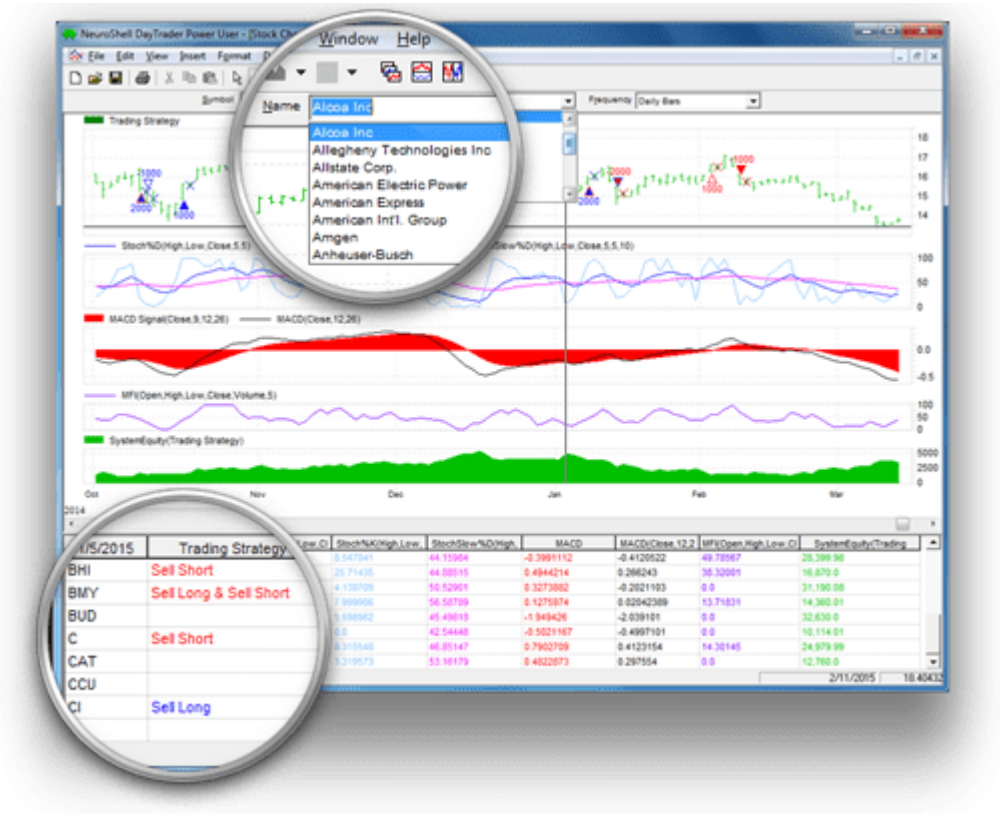


Рис. 1.4 NeuroShell (Офіційний сайт NeuroShell https://try.neuroshell.com/index/)

Ще одна перевага NeuroShell Day Trader – містить оптимізаційні методи, засновані принципами генетичних алгоритмів. Це впровадження дозволило покращити процес підготовки нейронних мереж завдяки вибору найкращих параметрів для індикаторів і обробці даних з різних входів мережі. Основною архітектурою, що використовується в NeuroShell Day Trader, це багатошаровий персептрон. Система орієнтована на створення торгової стратегії, яка поєднує індикатори та прогнозовані значення, отримані за допомогою нейронних мереж.

Elliott Wave Analyser Professional – це програмне забезпечення для аналізу валютного ринку, що базується на теорії хвиль Елліотта (ТХЕ) та стандартних алгоритмах технічного аналізу. Фінансист Ральф Елліотт виявив, що настрій натовпу впливає на валютні курси ще у 1930 році, і цей вплив проявляється у певних патернах, відомих як хвилі Елліотта (рис. 1.5).

Програма дозволяє виділяти неповні зразки з часового ряду, які потенційно відповідають хвилям Елліотта. З огляду на те, що поведінка стандартних хвиль Елліотта вивчена, ми можемо спрогнозувати подальший розвиток обраних зразків.

На кожний зразок система обчислює Goodness коефіцієнт, який визначає ступінь відповідності зразка теоретичному аналогу. Важливими параметрами аналізу є мітки та їх кількість, які апроксимують досліджуваний зразок, і користувач може налаштовувати щільність їх розподілу.

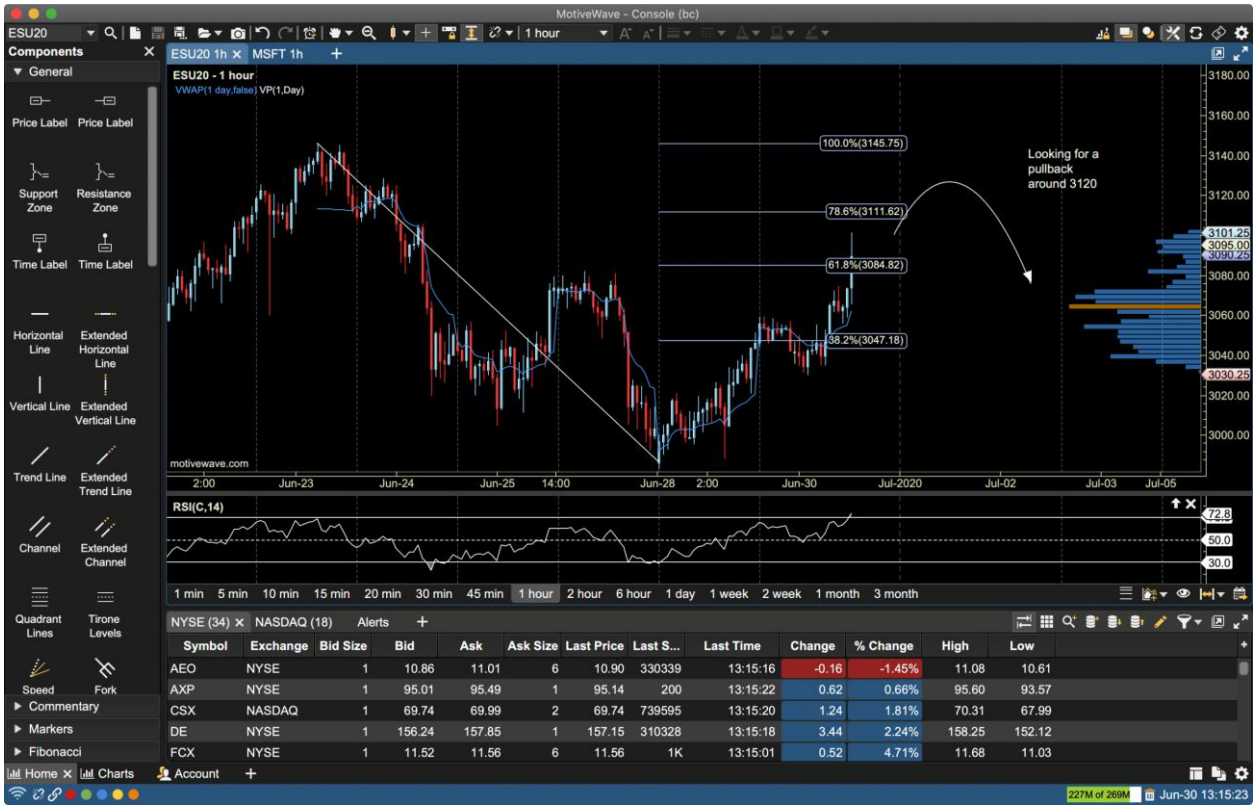


Рис. 1.5 Elliott Wave Analyser Professional (Офіційний сайт Elliott Wave Forecast https://elliottwave-forecast.com/)

Після аналізу кожної хвилі система генерує сигнали відкриття або закриття коротких і довгих позицій на ринку. Дані про курси можна згрупувати, виділяючи кольором валюти із сигналами на відкриття або закриття позицій. Користувачі можуть також налаштовувати параметри критеріїв закриття і відкриття позицій.

При запуску в режимі онлайн програма автоматично перераховує хвилі з періодичністю, яку задає користувач. Користувач може написати власний алгоритм індикатора, і він буде відображатися на тому ж графіку, що і вихідні дані. Доступна детальна довідкова система з теоретичною інформацією про принцип Елліотта та розділ Guided Tour, який веде користувача через всі етапи аналізу.

TXE дозволяє отримати довгострокові прогнози, але менш ефективний для короткострокових і середньострокових прогнозів через слабку кореляцію трендів з намірами гравців.

Ainet використовує нейронні мережі щоб прогнозувати події. Програма здебільшого не придатна для аналізу часових рядів, хоча показує хороші результати в завданнях з інтерполяції даних. До основних параметрів аналізу входить штрафний коефіцієнт, який автоматично оптимізується програмою.

Як вихідні дані використовується прямокутна матриця з повними даними і матриця з тими ж стовпцями, але з відсутніми даними. Програма намагається передбачити значення відсутніх даних. Недоліком програми є те, що вона працює за принципом "чорної скриньки", що ускладнює розуміння користувачем процесів, які відбуваються в нейронних мережах для побудови прогнозів.

1.6 Висновок до першого розділу

Криптовалюти набувають все більшої популярності, що підтверджується значним зростанням ринкової капіталізації. Найбільш відомими криптовалютами є Bitcoin, Ethereum, Tether та інші, кожна з яких має свої унікальні характеристики та сфери застосування.

Технологія блокчейн, яка лежить в основі більшості криптовалют, забезпечує безпеку та прозорість транзакцій, дозволяючи відстежувати всі операції в режимі реального часу.

Вони характеризуються децентралізованою структурою, що дозволяє уникнути контролю з боку центральних органів. Це забезпечує користувачам певні переваги, зокрема зниження транзакційних витрат та підвищення конфіденційності.

Проте криптовалюти також мають свої недоліки, зокрема високу волатильність та ризики, пов'язані зі спекулятивним характером. Перед інвестуванням необхідно ретельно вивчити ринок і усвідомити можливі ризики.

Також було розглянуто програмні засоби для прогнозування та моніторингу курсу криптовалют, які надають користувачам можливість аналізувати ринок та приймати обґрунтовані інвестиційні рішення. Незважаючи на різноманітність таких засобів, слід враховувати їхні обмеження та недоліки.

Отже, криптовалюти представляють складну та динамічну сферу, що потребує глибокого розуміння технічних, економічних та правових аспектів для їх ефективного використання та інвестування.

2 ІНСТРУМЕНТИ РОЗРОБКИ

2.1 Засоби розробки

2.1.1. Windows Presentation Foundation

Windows Presentation Foundation (WPF) — це потужний графічний фреймворк від Microsoft, призначений для розробки настільних додатків під операційну систему Windows. Вперше представлений у 2006 році як частина .NET Framework 3.0, WPF надає розробникам великий арсенал інструментів для створення складних інтерфейсів користувача, підтримки анімації, зв'язування даних та багато іншого.

Рис. 2.1 WPF

Для опису інтерфейсу користувача в додатках WPF використовується XAML – мова розмітки. Використання XAML дозволяє розробникам створювати інтерфейси у декларативному стилі, що спрощує процес розробки та полегшує співпрацю між розробниками та дизайнерами. Дозволяє створювати та застосовувати стилі і шаблони до інтерфейсних елементів., що дає можливість налаштовувати зовнішній вигляд об’єктів без зміни їх функціональності, сприяючи створенню уніфікованих і добре організованих інтерфейсів. Візуальні елементи інтерфейсу визначаються у XAML, тоді як логіка програми пишеться 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.

WPF поєднує в собі потужний рендеринг DirectX з вбудованими можливостями для роботи з мультимедіа та анімацією, роблячи його ідеальним інструментом для створення вражаючих та динамічних інтерфейсів користувачів.

Переваги WPF

1. Гнучкість дизайну

Завдяки стилям, шаблонам та анімаціям WPF дозволяє створювати кастомізовані та складні інтерфейси користувача.

2. Шкальованість

Векторна графіка забезпечує високу чіткість інтерфейсу на будь-якому екрані з будь-яким розширенням.

3. Можливість повторного використання

Стилі, шаблони та ресурси можуть бути визначені один раз і використані у багатьох місцях програми.

4. Інтеграція з іншими технологіями .NET

Легка інтеграція з іншими компонентами .NET, такими як LINQ, Entity Framework тощо.

Недоліки WPF

1. Продуктивність

На старих або слабких системах WPF може споживати багато ресурсів через свою графічну складність.

2. Складність засвоєння технології

Вивчення WPF є складним для новачків через його глибину та гнучкість.

3. Підтримка тільки Windows

WPF призначений виключно для розробки додатків під Windows, що обмежує його використання для крос-платформенних додатків.

2.1.2. Платформа розробки .Net Framework

.NET Framework — це одна з найпопулярніших платформ для розробки додатків, створених компанією Microsoft. Вона надає розробникам інструменти та сервіси, необхідні для створення різноманітних програмних засобів, від веб-сайтів до складних корпоративних систем. Платформа складається з великої бібліотеки класів (FCL) та середовища виконання загальної мови (CLR), яке забезпечує управління пам’яттю, безпеку та виконання коду. Це дозволяє розробникам писати програми, додатки та веб-сайти різними мовами програмування, таких як C#, F#, VB.NET та інші, і бути впевненими, що їхні програми будуть працювати стабільно та ефективно.

Рис. 2.2 .Net Framework

Архітектура .NET Framework включає середовище виконання загальної мови (CLR), яке відповідає за управління виконанням програм. CLR забезпечує автоматичне керування пам'яттю, безпеку коду та обробку винятків, що значно полегшує життя розробників. Бібліотека класів (FCL) надає великий набір багаторазових класів, інтерфейсів та типів, які спрощують розробку додатків. Це включає компоненти для роботи з колекціями, файловою системою, мережею, базами даних та забезпеченням безпеки.

.NET Framework має кілька ключових особливостей, які роблять його привабливим для розробників. По-перше, підтримка багатьох мов програмування дозволяє вибирати ту мову, яка найкраще відповідає вимогам конкретного

завдання. Це сприяє гнучкості та спрощує інтеграцію різних компонентів системи. По-друге, з виходом .NET Core, розробникам надали можливість створювати кросплатформені додатки, які можуть працювати на Windows, Linux та macOS. Це значно розширило можливості використання .NET у різних середовищах.

Однією з головних переваг використання .NET Framework є його розширювана екосистема, яка включає широкий спектр інструментів та бібліотек, підтримуваних спільнотою. Інтеграція з іншими продуктами Microsoft, такими як Azure, SQL Server та Visual Studio, також додає значну цінність для розробників. Крім того, високий рівень безпеки, забезпечений вбудованими механізмами, робить .NET Framework надійним вибором для створення додатків, де безпека має критичне значення. Незважаючи на деякі обмеження, такі як підтримка класичного .NET Framework лише на Windows та великий розмір інсталяційного пакету, платформа залишається одним з основних інструментів для багатьох розробників у всьому світі.

2.1.3. Мова програмування C#

C# є однією з найпопулярніших мов програмування, розробленою компанією Microsoft. Вона була створена як частина ініціативи .NET і призначена для розробки програмного забезпечення, яке буде працювати на платформі .NET Framework. З моменту свого випуску в 2000 році, C# швидко здобув визнання та популярність серед розробників завдяки своїй потужності, гнучкості та легкості у використанні.

Однією з основних переваг C# є його багатофункціональність. Мова підтримує як об'єктно-орієнтоване програмування, так і компонентне програмування, що дозволяє розробникам створювати складні додатки з модульною архітектурою. C# також надає засоби для роботи з асинхронним програмуванням, що дозволяє ефективно обробляти багатозадачні операції та покращувати продуктивність додатків. Мова підтримує сильну типізацію, що зменшує кількість помилок на етапі компіляції та підвищує надійність коду.

Архітектура мови C# інтегрована з .NET Framework, що забезпечує розробникам доступ до великої бібліотеки класів і API, які спрощують розробку

додатків. Це включає все: від роботи з базами даних і веб-службами до управління мережею та безпекою. Завдяки цьому, розробники можуть легко використовувати готові компоненти і зосередитися на бізнес-логіці своїх додатків, замість витрачати час на розробку основних функцій з нуля.

C# також відзначається своєю підтримкою кросплатформеності з виходом .NET Core і пізніше .NET 5/6, що дозволило розробникам створювати додатки для Windows, Linux і macOS з єдиною кодовою базою. Це відкриває нові можливості для створення універсальних додатків, які можуть бути розгорнуті в різних середовищах без значних змін у коді.

Завдяки своїй потужності, простоті використання та широким можливостям, C# є одним з головних інструментів для розробки сучасних додатків. Незалежно від того, чи працюєте ви над створенням великих корпоративних систем, мобільних додатків або ігор. Має всі необхідні засоби для реалізації будь-яких ідей. Мова продовжує розвиватися, отримуючи нові функції та вдосконалення, що робить її актуальною і привабливою для сучасних розробників.

2.1.4 База даних LiteDB

LiteDB — це безсерверна база даних, що постачається в одній маленькій збірці (< 450 Кб), повністю написаній у керованому коді .NET C#. Вона призначена для легковагових додатків, які потребують локального зберігання даних без складного налаштування серверів. LiteDB не потребує встановлення окремого серверного процесу, як-от SQL Server або MongoDB, і працює безпосередньо у вашому додатку.

Рис. 2.3 LiteDB

LiteDB вперше з’явилася у 2015 році, створена бразильським розробником Фабіо Галло. Її головна мета — надати простий та ефективний засіб для локального збереження даних у .NET-додатках, подібно до SQLite, але з NoSQL-підходом. Вона поступово еволюціонувала, і останні версії отримали кращу продуктивність, підтримку шифрування, транзакцій та інші покращення.

Переваги LiteDB

1. Простота використання

LiteDB дуже легко інтегрується у .NET-додатки. Для початку роботи достатньо просто підключити бібліотеку та створити файл бази даних. Немає потреби у складному налаштуванні сервера, встановленні додаткового ПЗ чи конфігурації мережевого підключення.

2. Малий розмір

Розмір бібліотеки LiteDB становить усього кілька мегабайт, що значно менше, ніж у багатьох реляційних СУБД або NoSQL-систем.

3. Висока продуктивність

Оскільки LiteDB працює безпосередньо з файлом, операції читання і запису відбуваються швидко. База оптимізована для роботи зі структурами BSON, що забезпечує високу швидкість обробки документів у порівнянні з традиційними реляційними базами даних.

Недоліки LiteDB

1. Не підходить для великих обсягів даних

LiteDB добре працює з невеликими та середніми обсягами інформації (до кількох гігабайт). Якщо база перевищує цей обсяг, можуть виникати проблеми з продуктивністю, оскільки LiteDB працює як файлова база даних і не має розподіленого зберігання.

2. Обмежена підтримка складних запитів

На відміну від реляційних баз даних, LiteDB не підтримує складні операції, такі як JOIN між таблицями. Це означає, що якщо потрібно працювати з пов’язаними даними, доведеться реалізовувати їхню обробку вручну.

3. Проблеми з одночасним доступом

Хоча LiteDB підтримує багатопотоковість, його продуктивність значно знижується при одночасному доступі кількох потоків до бази. Це пояснюється тим, що LiteDB використовує файлову блокування (file locking) для забезпечення цілісності даних.

2.2 Типи API

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.

Однією з найбільш популярних архітектур API є REST (Representational State Transfer), що широко використовується сучасними веб-додатками, такими як Netflix, Uber та Amazon. Для ефективної роботи з REST API важливо забезпечити дотримання кількох принципів, зокрема:

- Відсутність збереження стану – архітектура клієнт-сервер без збереження стану.
- Багаторівнева архітектура створення систем з різними рівнями взаємодії, де кожен рівень може бути незалежно модифікований.

Іншим відомим форматом API є SOAP (Simple Object Access Protocol), який активно використовувався з кінця 1990-х років для мережевого взаємодії. Однак, через свої жорсткі обмеження та високі витрати, багато розробників переходять до використання REST, який надає більш гнучкі та ефективні рішення.

RPC (Remote Procedure Call) є однією з найраніших форм API, що дозволяє клієнту виконувати код на сервері. RPC може використовувати формати як XML (XML-RPC), так і JSON (JSON-RPC) для кодування викликів. Водночас, інтерфейси RPC можуть бути складними у оновленні через тісний зв'язок між компонентами.

Також існують спеціалізовані API, наприклад CORBA, які вирішують специфічні задачі і можуть не підходити для загального використання.

Наразі REST API вважається найкращим стандартом для розробки мережевих додатків, що ефективно взаємодіють один з одним.

2.2.1. CoinGecko API

CoinGecko API – це потужний інструмент для розробників, що надає доступ до широкого спектру даних про криптовалюти та ринки в режимі реального часу. 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, забезпечуючи своїх користувачів найсвіжішою інформацією про ринок криптовалют (рис. 2.4).

Рис. 2.4 CoinGecko API

Крім поточних ринкових даних, CoinGecko API також надає доступ до історичних даних, що дозволяє аналізувати тренди та робити прогнози. Це робить API надзвичайно корисним для аналітиків та трейдерів, які прагнуть зрозуміти динаміку ринку та приймати обґрунтовані рішення. Дані про криптовалютні біржі, включаючи обсяги торгів та інші показники, також доступні через API, що допомагає відстежувати активність на різних платформах та оцінювати їхній вплив на ринок.

Однією з унікальних можливостей CoinGecko API є доступ до соціальних даних та сигналів, таких як активність у соціальних мережах (Twitter, Reddit) та розробницькі активності на GitHub. Ці дані можуть вказувати на зміну настроїв інвесторів і впливати на ринкові ціни, що робить їх важливим інструментом для аналізу. Крім того, API надає інформацію про ринки децентралізованих фінансів (DeFi), включаючи дані про токени DeFi, обсяги ліквідності та інші важливі показники.

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 всебічне розуміння ринку. Регулярне оновлення даних у режимі реального часу гарантує, що користувачі завжди мають доступ до найактуальнішої інформації. CoinGecko API надає інструменти для створення додатків, які можуть успішно конкурувати на швидкозростаючому ринку криптовалют.

2.2.2 CoinMarketCap API

CoinMarketCap API є високопотужним інструментом, який надає розробникам детальні дані про ринки криптовалют. Забезпечує доступ до великої кількості інформації, включаючи ціни на криптовалюти, ринкову капіталізацію, обсяги торгів та інші важливі показники, які оновлюються в реальному часі. API стала надзвичайно популярною серед фінансових аналітиків, інвесторів та розробників, які прагнуть інтегрувати надійні дані про криптовалюти у свої додатки або платформи (рис. 2.5).

Рис. 2.5 CoinMarketCap API

Основною перевагою CoinMarketCap API є її здатність забезпечувати вичерпну інформацію про всі аспекти ринку криптовалют. Вона включає деталізовані дані про кожну криптовалюту, такі як її 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 і високофункціональним інструментом для отримання детальної інформації про ринки криптовалют. Надає розробникам доступ до різноманітних даних, таких як ціни криптовалют в реальному часі, історичні цінові тренди, обсяги торгів та інші ринкові показники. Завдяки здатності агрегувати дані з багатьох бірж, CryptoCompare API забезпечує всебічний огляд ринку, що робить її цінним інструментом для трейдерів, аналітиків та ентузіастів криптовалют (рис. 2.6).

Рис. 2.6 CryptoCompare API

Однією з головних переваг CryptoCompare API є її здатність надавати високоточні дані в реальному часі. Це включає поточні ціни на криптовалюти, обсяги торгів та інші ринкові метрики, що допомагає користувачам залишатися в курсі останніх змін і приймати обґрунтовані рішення. Крім того, доступ до історичних даних дозволяє аналізувати минулі ринкові тренди і передбачати можливі майбутні зміни, що є важливим для стратегічного планування та аналізу.

CryptoCompare API також відрізняється своєю здатністю надавати соціальні індикатори, які відображають активність та настрої користувачів у соціальних мережах, таких як Twitter та Reddit. Ці дані дозволяють користувачам оцінювати

вплив громадської думки на ринок криптовалют, що може бути корисним для виявлення трендів та передбачення ринкових рухів. Соціальні індикатори додають додатковий вимір до традиційного ринкового аналізу, надаючи більш повну картину ринкових умов.

API надає детальну інформацію про криптовалютні біржі, включаючи обсяги торгів, ліквідність і доступні торгові пари. Ці дані допомагають користувачам вибирати найкращі платформи для торгівлі і розуміти динаміку різних ринків. Детальна інформація про конкретні монети, їхні технічні характеристики та ринкову капіталізацію також доступна, що робить API корисним для глибокого аналізу конкретних криптовалют.

Загалом, CryptoCompare API є потужним інструментом для всіх, хто займається криптовалютами, надаючи широкий спектр даних для всебічного аналізу ринку. Її гнучкість і надійність роблять її ідеальною для використання в різних додатках і сервісах, від торгових платформ до аналітичних інструментів, забезпечуючи користувачам необхідні ресурси для успішної роботи в динамічному світі криптовалют.

2.3 Порівняльний аналіз різних API

У сучасному світі криптовалют наявність надійних і детальних даних є критично важливою для аналізу ринку та прийняття обґрунтованих рішень. Три з найбільш популярних API для отримання інформації про криптовалюти – CoinGecko, CoinMarketCap і CryptoCompare – надають різноманітні дані, включаючи ціни в реальному часі, історичні дані, ринкову капіталізацію та багато іншого. Нижче представлена порівняльна таблиця 2.1, яка детально описує функціональні можливості кожного з цих API, щоб допомогти користувачам вибрати найкращий інструмент для їхніх потреб у роботі з криптовалютами.

Таблиця 2.1

Порівняння різних API

Функціональність	CoinGecko API	CoinMarketCap API	CryptoCompare API
Ціни в реальному часі	Надає актуальні дані про понад 6000 криптовалют, збираючи інформацію з численних бірж для забезпечення точності і повноти.	Забезпечує високоточні дані про ціни криптовалют з широким покриттям ринку.	Надає інформацію про ціни в реальному часі для широкого спектру криптовалют і бірж.
Історичні дані	Забезпечує детальні історичні дані, що дозволяють проводити глибокий аналіз ринкових трендів.	Надає доступ до докладних історичних записів для ретроспективного аналізу.	Пропонує багаторічні історичні дані, корисні для аналізу довгострокових трендів.
Ринкова капіталізація	Містить детальну інформацію про ринкову капіталізацію широкого спектру криптовалют.	Пропонує актуальну інформацію про ринкову капіталізацію численних криптовалют.	Надає точні дані про ринкову капіталізацію для глибокого ринкового аналізу.
Обсяги торгів	Забезпечує дані про обсяги торгів з регулярними оновленнями в реальному часі.	Включає детальні звіти про обсяги торгів на численних біржах.	Надає актуальні дані про обсяги торгів, що дозволяють оцінити ринкову активність.
Дані про біржі	Пропонує детальну інформацію про криптовалютні біржі та торгові пари.	Забезпечує велику кількість інформації про різні біржі.	Надає докладні дані про біржі, включаючи обсяги торгів і доступні торгові пари.
Соціальні індикатори	Включає дані з Twitter, Reddit та GitHub, що дозволяє	Не підтримує соціальні індикатори.	Пропонує дані з Twitter та Reddit для аналізу

	оцінювати настрої інвесторів.		громадської думки.
Дані про DeFi	Підтримує дані про DeFi токени, що дозволяє оцінювати ринок децентралізованих фінансів.	Не включає	Не включає
Деталі про монети	Надає детальну інформацію про кожну криптовалюту, включаючи технічні характеристики та ринкові показники.	Пропонує детальні дані про криптовалюти, включаючи ринкову капіталізацію і обсяги торгів.	Забезпечує докладні характеристики кожної монети для глибокого аналізу.
Агреговані дані	Не підтримує агреговані дані.	Збирає дані з багатьох джерел для точного ринкового аналізу.	Пропонує дані з різних джерел для комплексного аналізу ринку.
Доступ до API	Безкоштовний план з обмеженнями, доступні платні плани з розширеними можливостями.	Безкоштовний план з обмеженнями, доступні платні плани з розширеними можливостями.	Безкоштовний план з обмеженнями, доступні платні плани з розширеними можливостями.
Максимальна кількість запитів	До 100 запитів на хвилину на платному плані, що підходить для великих проектів.	До 10 000 запитів на місяць на платному плані, що обмежує використання в масштабних проектах.	До 50 000 запитів на місяць на платному плані, що забезпечує високу пропускну здатність.
Додаткові функції	Включає дані про категорії криптовалют та підтримку NFT, що розширює можливості аналізу.	Інтеграція з іншими продуктами CoinMarketCap, що надає додаткові	Пропонує соціальні індикатори та аналіз активності у соцмережах для комплексного

		можливості для користувачів.	ринкового аналізу.

2.4 Висновок до другого розділу

Цей розділ присвячений інструментам розробки програмного забезпечення: графічному фреймворку Windows Presentation Foundation (WPF), платформі .NET Framework та мові програмування C#. 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.

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.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Аналіз вимог, проектування архітектури та формулювання завдання.

Оскільки рішення щодо проектування та розробки компонентів мають вплив на всю систему та її архітектуру, кожна підсистема повинна відповідати встановленим вимогам. Тому детальний аналіз охоплює дослідження функціональних можливостей програмного продукту, призначеного для збору, обробки та аналізу даних конкретної криптовалюти.

Технічні вимоги до зазначеного програмного забезпечення включають:

1) Забезпечення працездатності на персональних комп’ютерах зі стандартною апаратною конфігурацією.

2) Інтуїтивно зрозумілий інтерфейс та зручність у використанні.

3) Висока швидкість обробки даних і простий доступ до отриманої інформації.

4) Простота масштабування та підтримки продукту.

5) Мінімальні затрати на впровадження продукту.

Концептуальна модель являє собою комбінацію концепцій розробника та користувача. Взаємодія між акторами та прецедентами відображається на діаграмі прецедентів (рис. 3.1), яка показує всіх акторів системи та всі можливі варіанти їхніх дій.

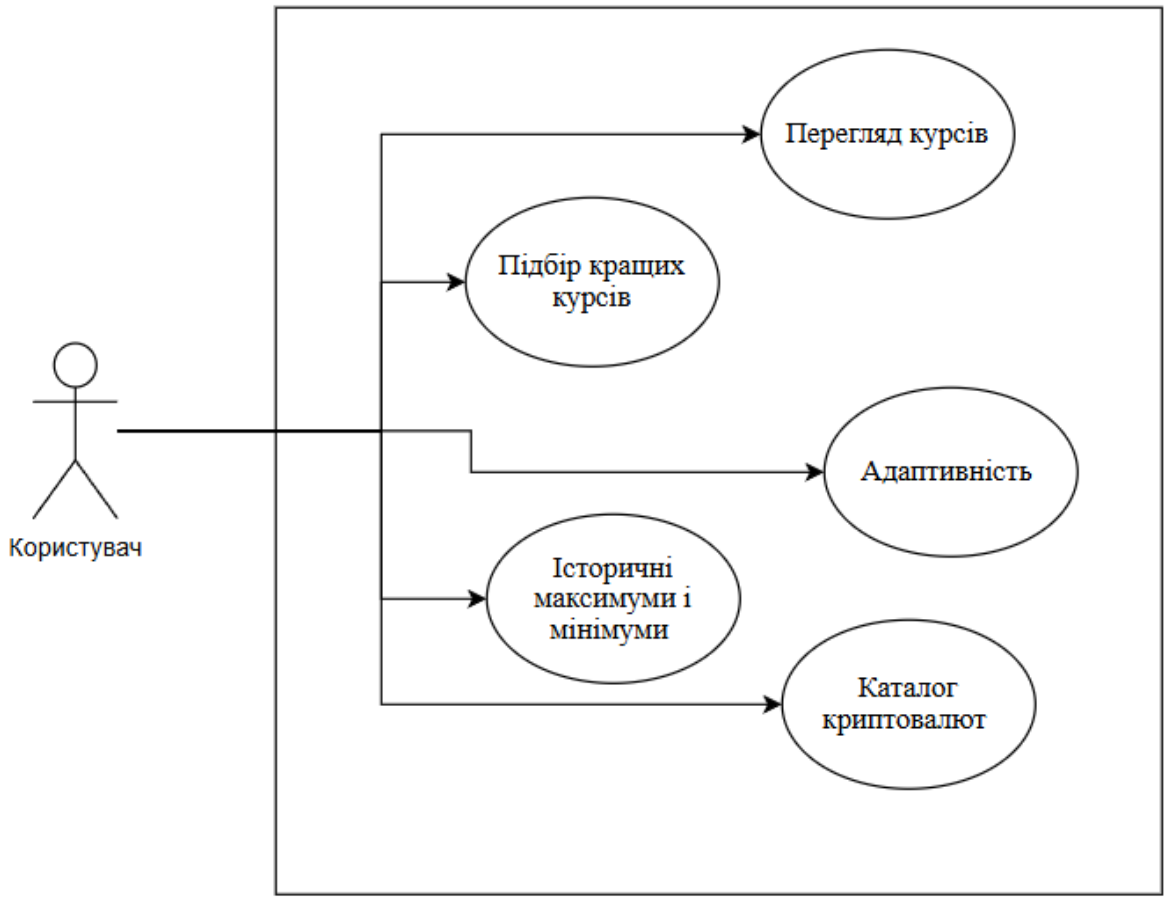


Рис. 3.1 Діаграма прецендентів

Актор "Користувач" є відвідувачем сервісу, який має доступ до таких функцій:

- Перегляд курсів – користувач маж можливість переглядати курси криптовалют за різні часові періоди та отримувати таблиці з інформацією про торги на біржах.
- Адаптивність – додаток автоматично підлаштовується під різні розміри екрану комп'ютера.
- Підбір кращих курсів – автоматичний підбір курсів, які є найбільш вигідними для покупки або продажу.
- Історичні максимуми і мінімуми – сервіс для збору історичних даних, який дозволяє переглядати максимальний ріст та спад курсів за обраний період часу.

- Каталог криптовалют – список криптовалют з інформацією про їх технічні характеристики.

Система базується на використанні .NET 6 та WPF, а дані про криптовалюти отримуються з API CoinGecko та зберігаються в локальній базі даних. Схему збереження даних зображена на діаграмі класів(рис 3.2).

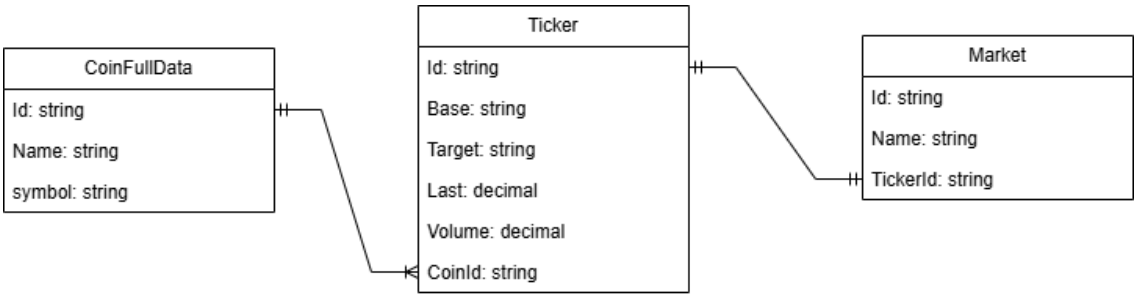


Рис 3.2 Діаграма класів

3.2 Основна логіка виконання програми

Згідно з визначеними вимогами, логіка роботи програми ґрунтується на взаємодії з користувачем, який обирає потрібні функціональні можливості для виконання в конкретний 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.

Дані про криптовалюти з'являються, коли користувач запускає програму, яка надсилає запит до API CoinGecko. На цьому етапі дані вважаються "отриманими". До моменту відображення даних на екрані користувача або оновлення даних, інформація може перебувати в стані "отримані" або перейти в стан "помилка отримання". Якщо дані отримані з помилкою, програма перевіряє в базі даних чи були ці дані отримані раніше, якщо база даних не має інформації, то життєвий цикл завершується.

Логіка отримання даних відображається на діаграмі послідовності (рис. 3.3). Якщо дані успішно отримані, вони зберігаються до локальної бази даних і далі

користувач у будь-який момент може переглянути їх або виконати фільтрацію та сортування. Якщо дані фільтруються або сортуються, вони переходять у стан "відфільтровані" або "відсортовані", відповідно. Якщо жодна з цих дій не відбулася, дані оновлюються автоматично через певний проміжок часу, повертаючись у стан "отримані".

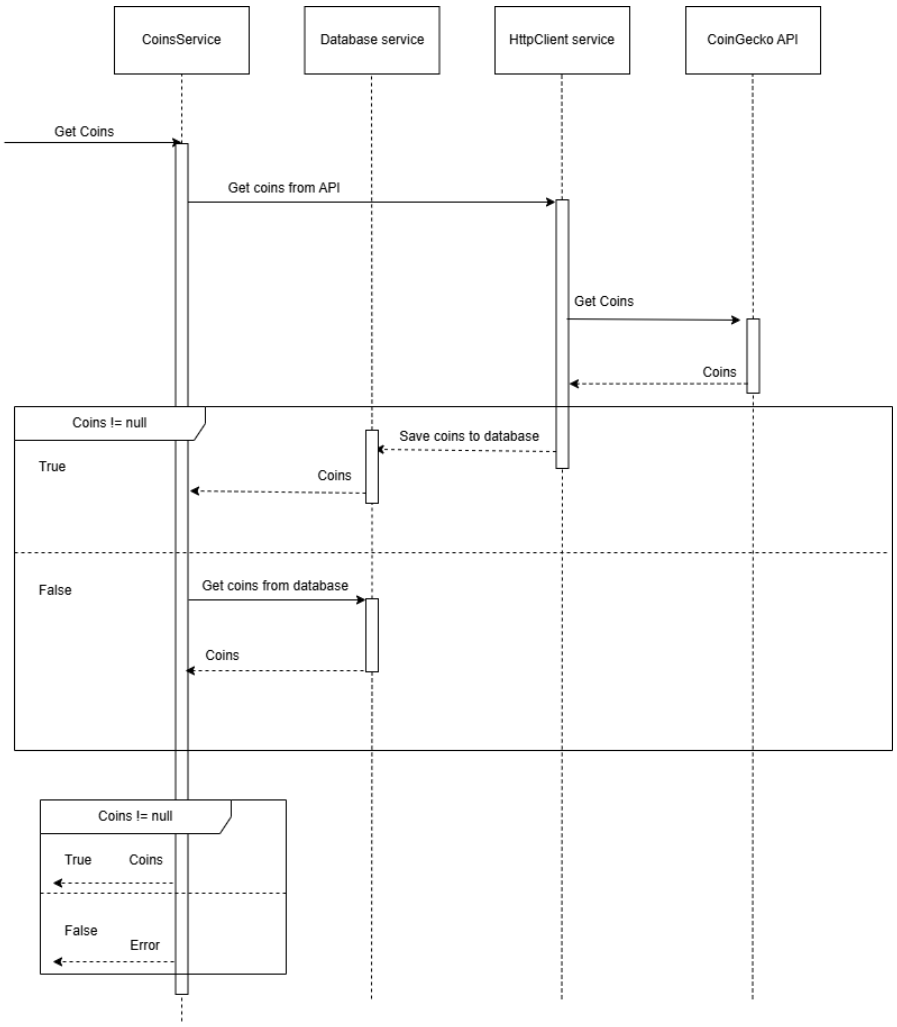


Рис 3.3 Діаграма послідовності

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. 3.4) 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()...
}
```

Рис. 3.4 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.

3. CoinView

Цей клас забезпечує основний функціонал для відображення детальної інформації про криптовалюту (рис. 3.5). 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)...
}
```

Рис. 3.5 Клас CoinView

2. ErrorDialog

Клас ErrorDialog (рис. 3.6) 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"));
    }
}
```

Рис. 3.6 Вікно ErrorDialog

4. Converters

Модуль Converters забезпечує функціонал для перетворення даних у WPF-додатку (рис. 3.7). 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
          Ссылок: 0
26    public class CurrencyVolumeToVolume ...
44
          Ссылок: 0
45    public class SupplyWithSymbol ...
63
          Ссылок: 0
64    public class SparklineToSvg ...
99
          Ссылок: 0
100   public class SparklineToBrush ...
117
          Ссылок: 0
118   public class SortToText ...
139
          Ссылок: 0
140   public class PercentageToTrend ...
152
          Ссылок: 0
153   public class PercentageToText ...
167
          Ссылок: 0
168   public class PercentageToBrush ...
180
          Ссылок: 2
181   public class DecimalToVariablePrecision ...
201
          Ссылок: 0
202   public class DecimalToVariableCurrency ...
222
          Ссылок: 0
223   public class DecimalToShortCurrency ...
244
          Ссылок: 0
245   public class DecimalToZeroCurrency ...
257
          Ссылок: 0
258   public class ChartHeightToMaxHeight ...
270
          Ссылок: 0
271   public class DayToFontWeight ...
291
          Ссылок: 0
292   public class SelectedRangeToBackground ...
314
          Ссылок: 0
315   public class DecimalDivision ...
331
          Ссылок: 0
332   public class NullToVisibilityConverter ...
344
          Ссылок: 0
345   public class CurrencyToVisibility ...
361
          Ссылок: 0
362   public class DarkModeToBrush ...
```

Рис. 3.7 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. 3.8).

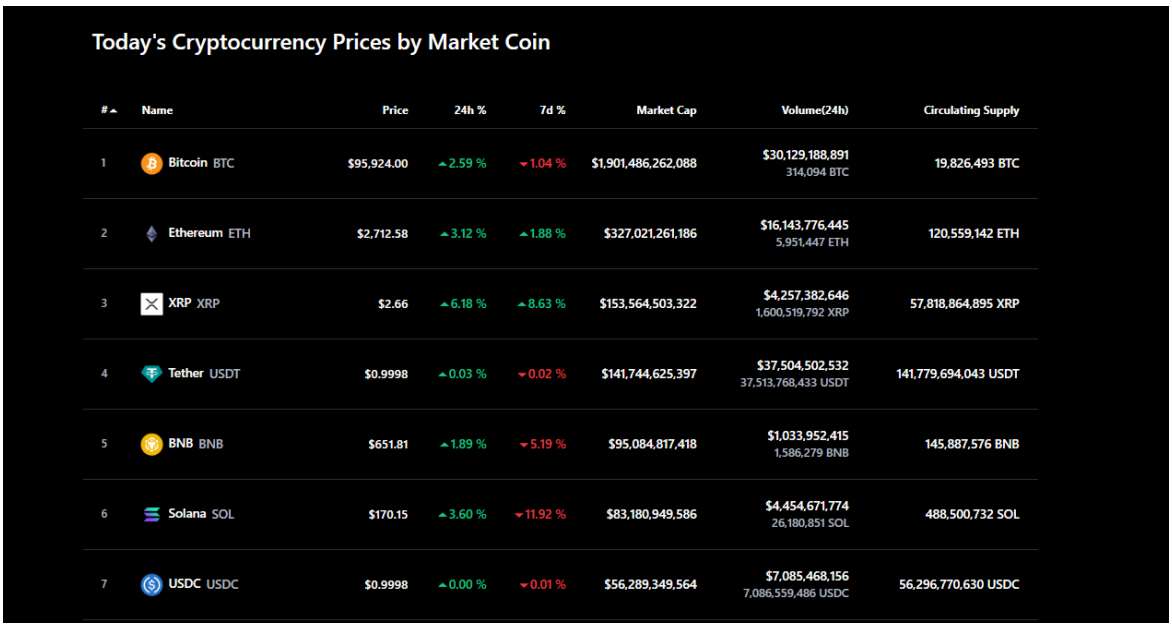


Рис. 3.8 Головне вікно додатку

Це вікно запускається на старті програми якщо маємо Інтернет-з’єднання та не вичерпали ліміт безкоштовної кількості запитів. Якщо немає мережі маємо такий вигляд вікна (рис. 3.9).

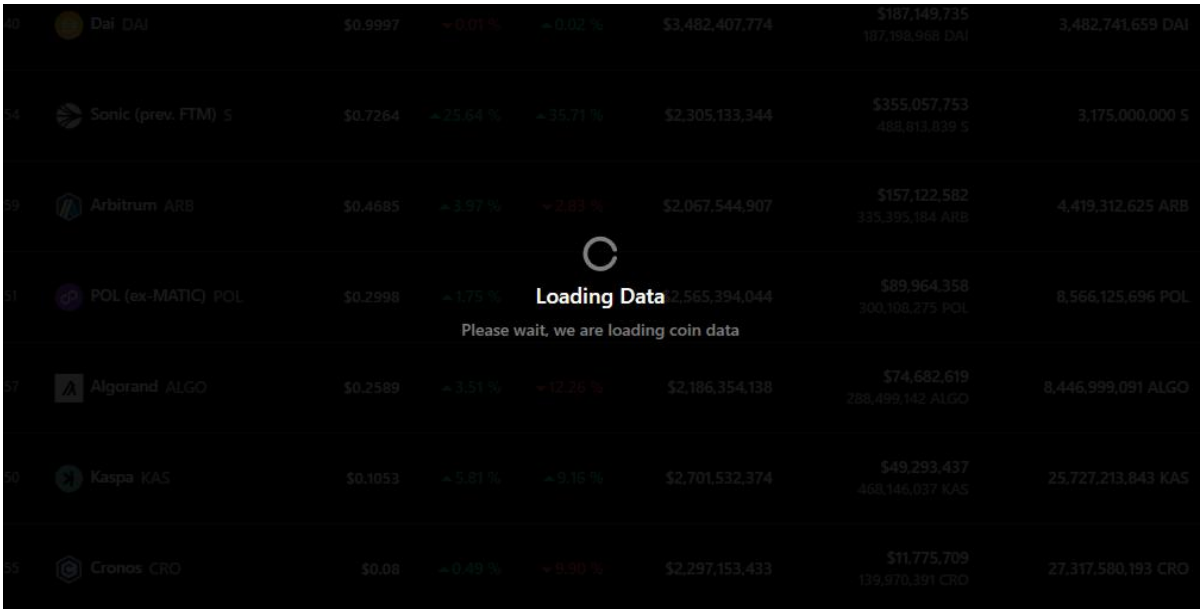


Рис. 3.9 Відсутня мережа

Якщо перевищили ліміт запитів до API, то отримаємо таку вигляд вікна (рис. 3.9)

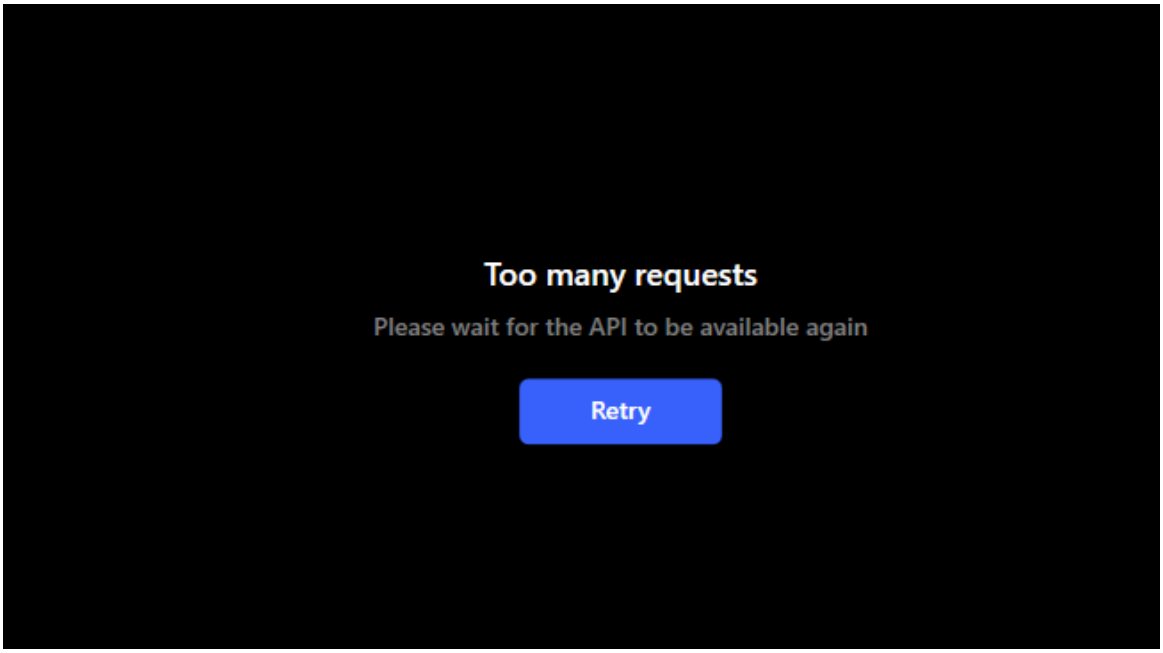


Рис. 3.10 Перевищений ліміт запитів

Для знаходження бажаної криптовалюти можна використовувати навігацію у вигляді кнопок «назад» та «вперед» (рис. 3.11).

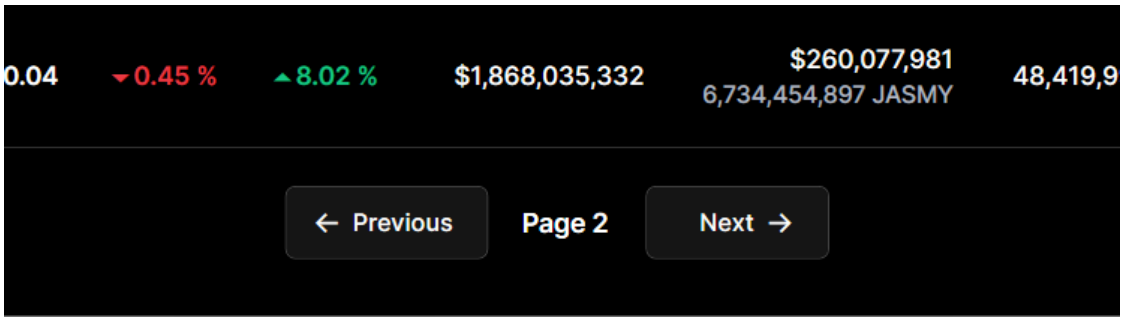


Рис. 3.11 Навігація в таблиці

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 3.12).

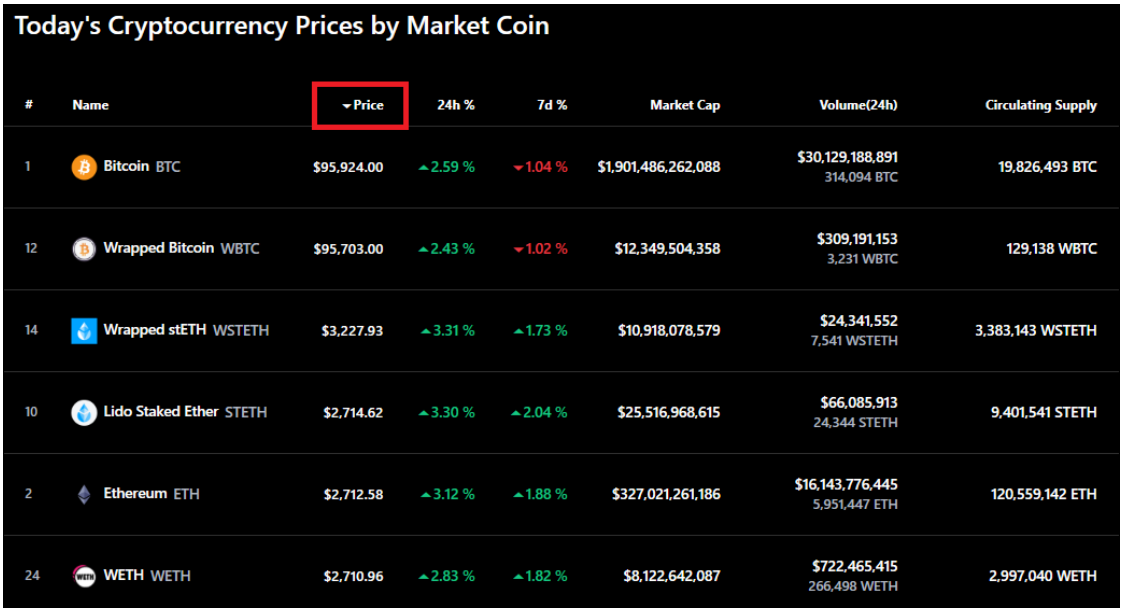


Рис. 3.12 Фільтри в таблиці

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).

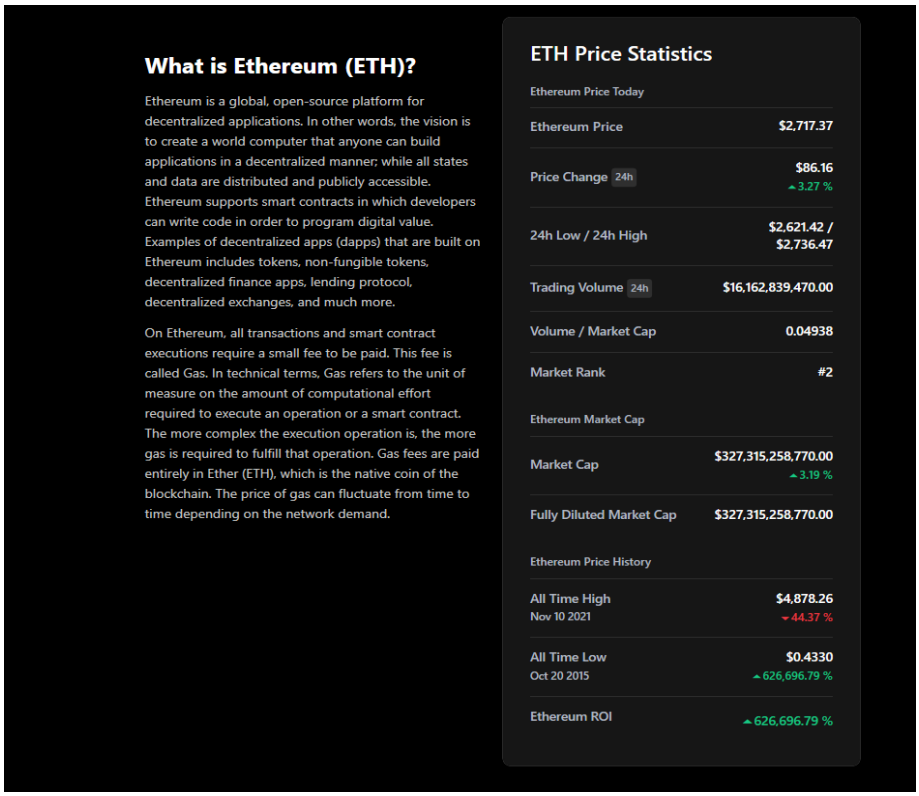


Рис. 3.13 Вікно з детальною інформацією про обрану криптовалюту

У вікні маємо короткий опис криптовалюти (рис 3.14)

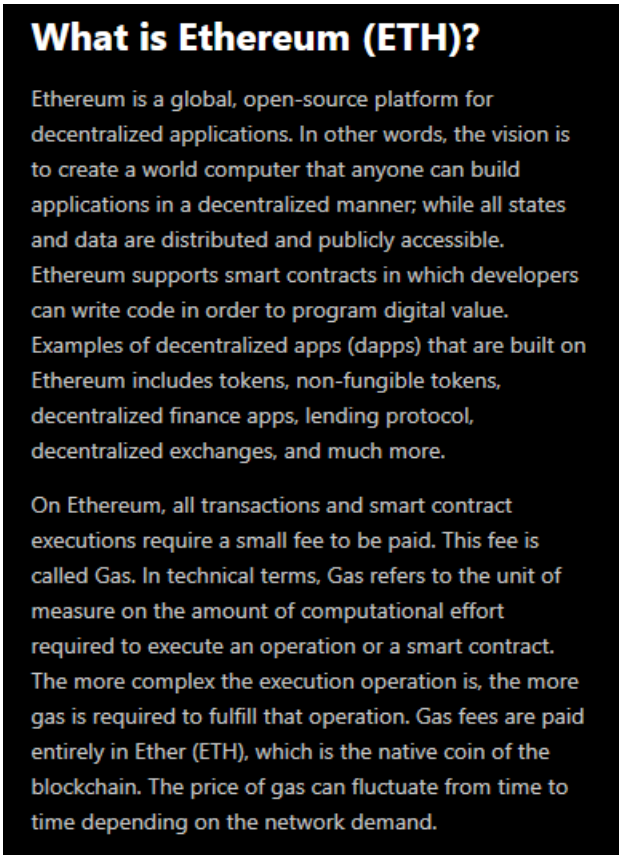


Рис. 3.14 Опис обраної криптовалюти

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

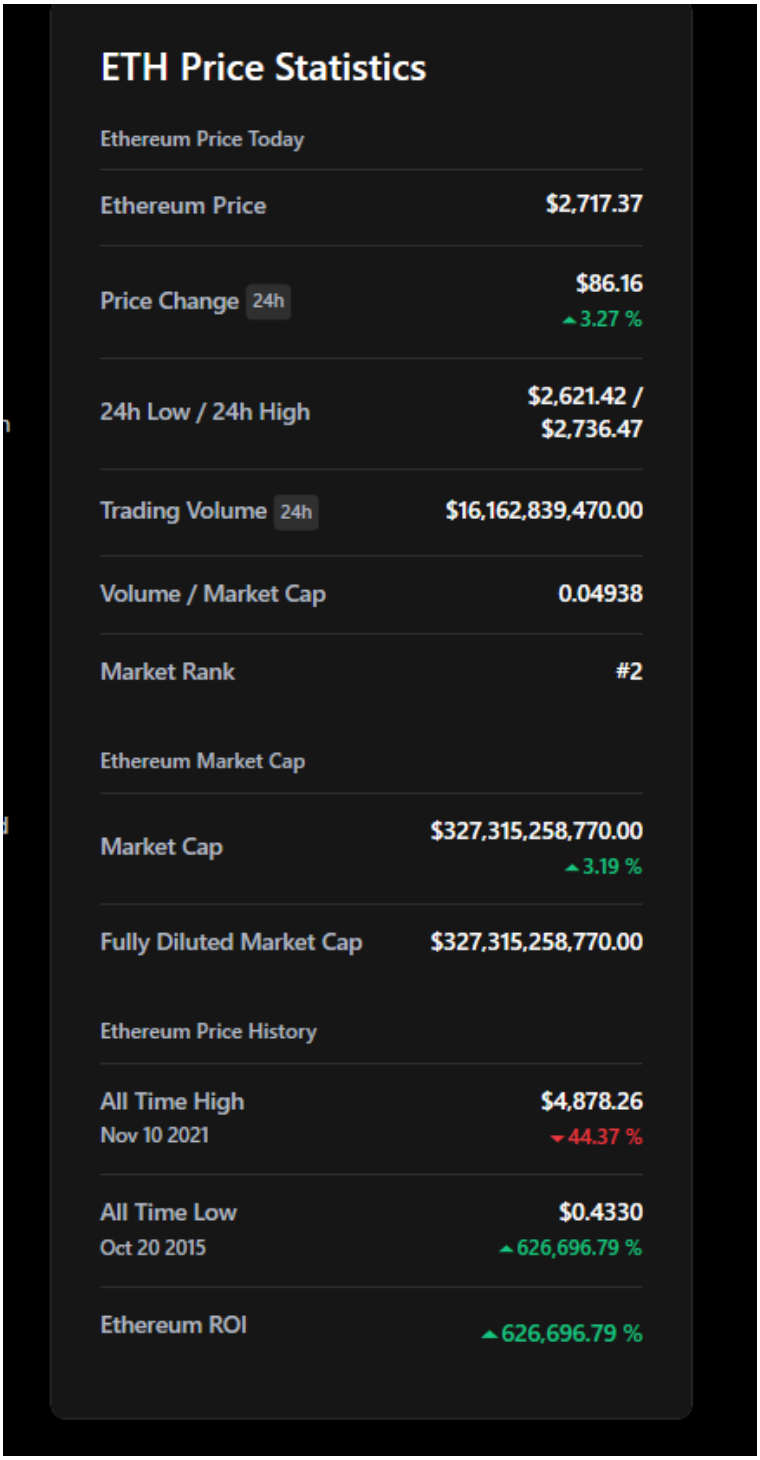
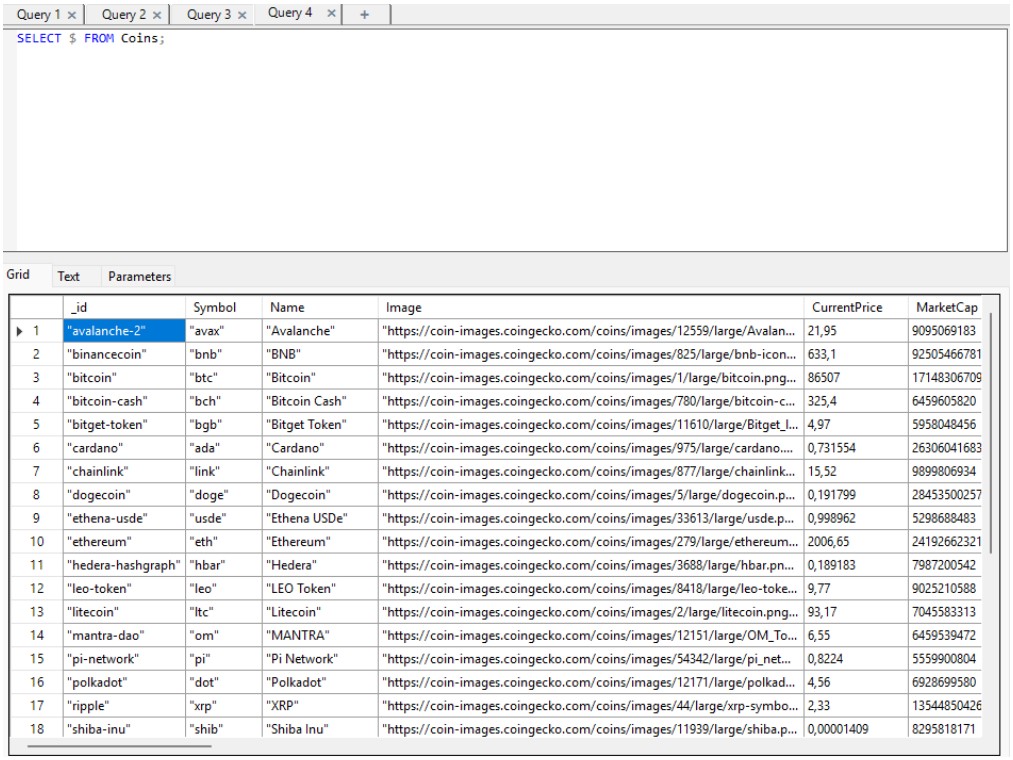


Рис. 3.15 Статистика по валюті

Всі успішно отримані дані були збережені до локальної бази даних(рис 3.16).



	_id	Symbol	Name	Image	CurrentPrice	MarketCap
1	"avalanche-2"	"avax"	"Avalanche"	"https://coin-images.coingecko.com/coins/images/12559/large/Avalan...	21,95	9095069183
2	"binancecoin"	"bnb"	"BNB"	"https://coin-images.coingecko.com/coins/images/825/large/bnb-icon...	633,1	92505466781
3	"bitcoin"	"btc"	"Bitcoin"	"https://coin-images.coingecko.com/coins/images/1/large/bitcoin.png...	86507	17148306709
4	"bitcoin-cash"	"bch"	"Bitcoin Cash"	"https://coin-images.coingecko.com/coins/images/780/large/bitcoin-c...	325,4	6459605820
5	"bitget-token"	"bgb"	"Bitget Token"	"https://coin-images.coingecko.com/coins/images/11610/large/Bitget_l...	4,97	5958048456
6	"cardano"	"ada"	"Cardano"	"https://coin-images.coingecko.com/coins/images/975/large/cardano....	0,731554	26306041683
7	"chainlink"	"link"	"Chainlink"	"https://coin-images.coingecko.com/coins/images/877/large/chainlink...	15,52	9899806934
8	"dogecoin"	"doge"	"Dogecoin"	"https://coin-images.coingecko.com/coins/images/5/large/dogecoin.p...	0,191799	28453500257
9	"ethena-usde"	"usde"	"Ethena USDe"	"https://coin-images.coingecko.com/coins/images/33613/large/usde.p...	0,998962	5298688483
10	"ethereum"	"eth"	"Ethereum"	"https://coin-images.coingecko.com/coins/images/279/large/ethereum...	2006,65	24192662321
11	"hedera-hashgraph"	"hbar"	"Hedera"	"https://coin-images.coingecko.com/coins/images/3688/large/hbar.pn...	0,189183	7987200542
12	"leo-token"	"leo"	"LEO Token"	"https://coin-images.coingecko.com/coins/images/8418/large/leo-toke...	9,77	9025210588
13	"litecoin"	"ltc"	"Litecoin"	"https://coin-images.coingecko.com/coins/images/2/large/litecoin.png...	93,17	7045583313
14	"mantra-dao"	"om"	"MANTRA"	"https://coin-images.coingecko.com/coins/images/12151/large/OM_To...	6,55	6459539472
15	"pi-network"	"pi"	"Pi Network"	"https://coin-images.coingecko.com/coins/images/54342/large/pi_net...	0,8224	5559900804
16	"polkadot"	"dot"	"Polkadot"	"https://coin-images.coingecko.com/coins/images/12171/large/polkad...	4,56	6928699580
17	"ripple"	"xrp"	"XRP"	"https://coin-images.coingecko.com/coins/images/44/large/xrp-symbo...	2,33	13544850426
18	"shiba-inu"	"shib"	"Shiba Inu"	"https://coin-images.coingecko.com/coins/images/11939/large/shiba.p...	0,00001409	8295818171

Рис 3.16 Збережені дані валют

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 розроблено ефективне програмне забезпечення для збору даних про курс криптовалют, який відповідає сучасним вимогам. Додаток пропонує зручний, мінімалістичний та зрозумілий інтерфейс, швидку обробку даних та надає доступ до інформації в реальному часі. Завдяки використанню API CoinGecko, користувачі мають можливість отримувати актуальні дані про курси криптовалют, їхні графіки та історичні максимуми і мінімуми.

Основними функціями додатку є перегляд курсу криптовалют, автоматична побудова графіків, фільтрація та сортування даних, підбір кращих курсів для купівлі та продажу, а також можливість підписки на розсилку прогнозів. Додаток адаптовано для роботи на різних розширеннях екрану, що забезпечує зручність використання.

Технологічна основа додатку включає використання .NET 6 та WPF, що забезпечує стабільність та високу продуктивність програмного забезпечення. Проведене тестування підтвердило надійність та безпеку додатку, виявлені дефекти були успішно усунені. У підсумку, створено надійний інструмент для збору даних про курс криптовалют, який дозволяє користувачам приймати обґрунтовані інвестиційні вибори та отримувати актуальну інформацію про ринок криптовалют.

ВИСНОВОК

У процесі роботи були вивчені основні принципи та механізми функціонування криптовалют, їхні переваги та недоліки. Також було розглянуто сучасні технології, методи розробки програмного забезпечення та відповідні інструменти, які забезпечують ефективну роботу з криптовалютами. Зокрема, було досліджено різні API для отримання даних про криптовалюти, а також методи їх обробки та аналізу.

Було створено функціональний додаток для збору даних про курс криптовалют, який забезпечує користувачів зручним інтерфейсом та доступом до актуальної інформації. Використання сучасних технологій, таких як .NET 6 та WPF, а також інтеграція з CoinGecko API дозволили досягти продуктивності та стабільності додатку. Основні функціональні можливості включають перегляд курсів криптовалют, автоматичну побудову графіків, фільтрацію та сортування даних, підбір найвигідніших курсів.

Проведене тестування підтвердило стабільну роботоздатність додатку. Інтерфейс користувача є зручним та інтуїтивно зрозумілим, що забезпечує високу задоволеність користувачів. Окрім цього, додаток має адаптивний дизайн, що забезпечує зручне використання його на різних пристроях з різними розмірами екрану.

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 дані про децентралізовані фінанси (DeFi).

ПЕРЕЛІК ПОСИЛАНЬ

1. Decentralization in bitcoin and ethereum networks / A. E. Gencer et al. *Financial cryptography and data security*. Berlin, Heidelberg, 2018. P. 439–457. URL: https://doi.org/10.1007/978-3-662-58387-6_24

2. Двуліт З. П. Криптовалюта: стан та тенденції розвитку / З. П. Двуліт, Х. С. Передало, Р. Б. Тиліпська, Р. М. Терно, Р. І. Стибель // Економіка та держава. - 2019. - № 1. - С. 10-14. - Режим доступу: http://nbuv.gov.ua/UJRN/ecde_2019_1_4.

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

4. Лапко О. О. Технологія блокчейн: поняття, сфери застосування та вплив на підприємницький сектор / О. О. Лапко, О. С. Солосіч // Бізнес Інформ. - 2019. - № 6. - С. 77-82. - Режим доступу: http://nbuv.gov.ua/UJRN/binf_2019_6_11.

5. Troelsen A., Japikse P. Pro C# 10 with . NET 6: foundational principles and practices in programming. Apress L. P., 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 Whitebit [Електронний ресурс]. – Режим доступу: URL: https://whitebit.com

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

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

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

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

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

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

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

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

17. Офіційна 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

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

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

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

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

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).ToLi
st();
    }

    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.Current
Domain.BaseDirectory, "Fonts",
"#Inter"));

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

        Sort = new
RelayCommand<string>(SortObservableCollec
tion);
        SelectCurrency = new
AsyncRelayCommand<Currency>(OnSelectCurre
ncy);
        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.Current
Domain.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("#1
6C784")!;
            var red =
(Color)ColorConverter.ConvertFromString("#E
A3943")!;
            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("#1
6C784")!;
            var red =
(Color)ColorConverter.ConvertFromString("#E
A3943")!;

            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.GetMarketChartRangeByCoinI
d(
                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((lo
ng)(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(tim
eFormat, 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.CurrentDo
main.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.Descript
ion["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 }
        );
    }
}
```

ПРЕЗЕНТАЦІЯ

Державний університет інформаційно-комунікаційних технологій

Кафедра Інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Програмна платформа для відображення курсів криптовалют та аналітичної інформації»

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

Виконав: Коваленко М.О, ІСД-42
Науковий керівник роботи:
Викладач кафедри ІСТ, Шахматов І.О.

Київ - 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: створити метод збору актуальної інформації курсу криптовалюти, реалізувати його у вигляді програмного засобу та дослідити ефективність.

Об`єкт дослідження: процес збору та агрегації даних про курс криптовалюти.

Предмет дослідження: програмна платформа збору та відображення даних про курс криптовалюти.

Актуальність теми: З моменту свого виникнення криптовалютний ринок привертає увагу багатьох професійних трейдерів і початківців, які шукають ефективні моделі прогнозування. Для прийняття успішних рішень у цій сфері необхідно використовувати різні методи аналізу, враховуючи специфіку ринку криптовалют.

Завдання дослідження:

1. Проаналізувати становлення криптовалют та засади їх функціонування.
2. Розглянути принципи роботи криптовалют через криптовалютні біржі.
3. Спроектувати систему збору та обробки даних про курси криптовалют.
4. Розробити застосунок для обробки даних з криптовалютних бірж з метою їх об’єднання та подальшої аналітичної обробки.
5. Забезпечити можливість роботи застосунку в автономному режимі шляхом реалізації механізму кешування отриманих даних.

Графік капіталізації ринку криптовалют

(Джерело: www.tradingview.com)

АНАЛІЗ АНАЛОГІВ

Показник	Walletinvestor	Belinvestor	Elliott Wave Analyser Professional	Розроблений застосунок
Платформа	Веб-браузер	Веб-браузер	Windows	Windows
Безкоштовна версія	Частково	Частково	Ні	Так
Простота використання та інтерфейсу	Висока	Висока	Низька	Висока
Порівняння популярності валюти	Ні	Ні	Ні	Так
Цільова аудиторія	Початківці	Початківці	Професіонали	Початківці
Ціна	Платна	Платна	Платна	Безкоштовна
Динаміка зміни вартості	Так	Так	Ні	Так

5

АНАЛІЗ АНАЛОГІВ

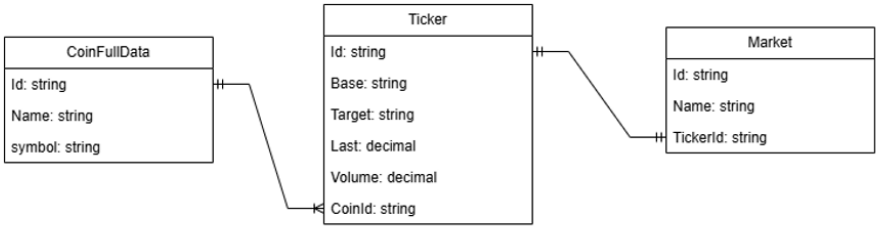
Показник	Walletinvestor	Belinvestor	Elliott Wave Analyser Professional	Розроблений застосунок
Зміна вартості за сьогодні	Так	Так	Ні	Так
Зміна вартості за тиждень	Так	Так	Ні	Так
Ринкова капіталізація	Так	Частково(не завжди доступна)	Ні	Так
Загальна інформація про валюту та її історію	Так	Обмежено	Ні	Так
Обсяг торгів за 24 години	Так	Частково	Ні	Так
Циркулярний запас	Так	Ні	Ні	Так
Історичні дані	Так	Так	Ні	Так
Кешування даних для збереження автономної роботи	Ні	Ні	Ні	Так

6

ІНСТРУМЕНТИ РОЗРОБКИ

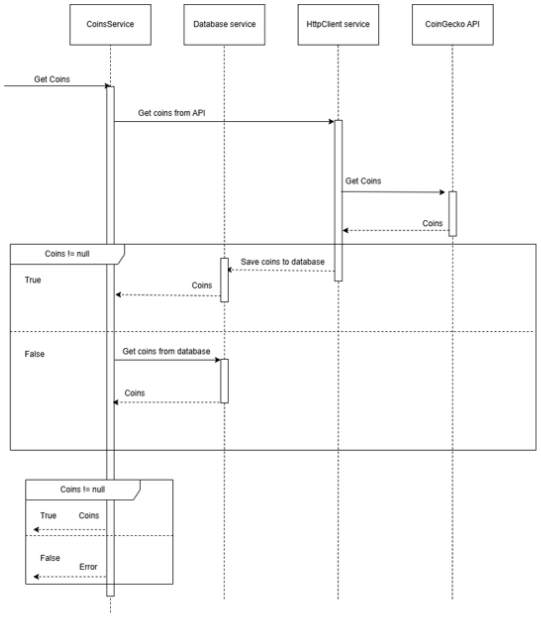
7

ДІАГРАМА КЛАСІВ



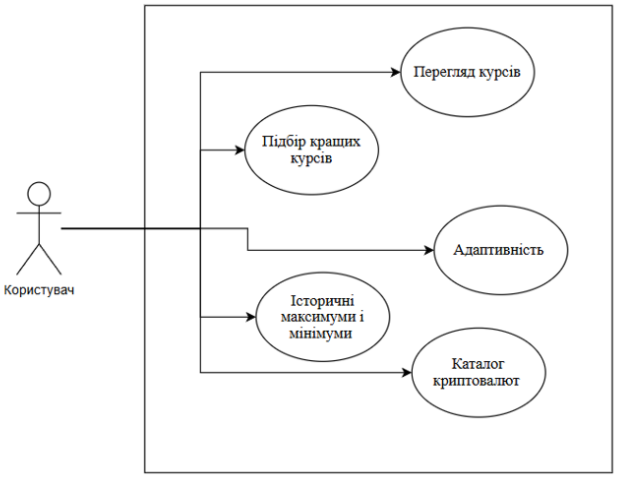
8

ДІАГРАМА ПОСЛІДОВНОСТІ



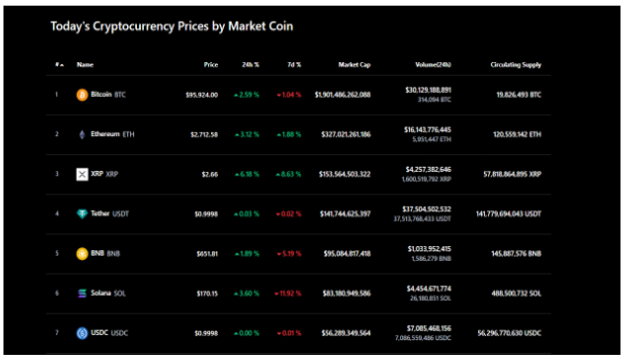
9

ЗАГАЛЬНА ДІАГРАМА ВИКОРИСТАННЯ

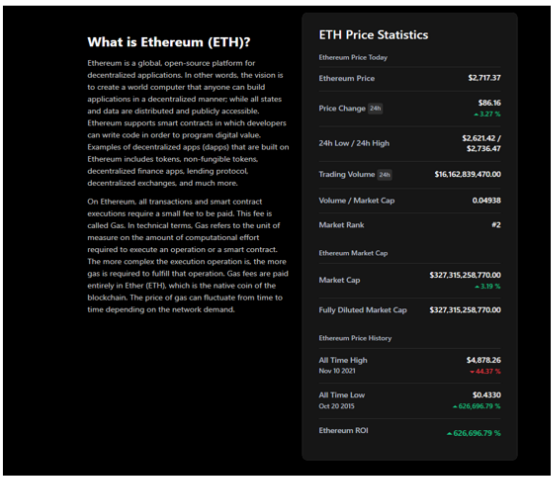


10

ДЕМОНСТРАЦІЯ РОЗРОБКИ ПЛАТФОРМИ



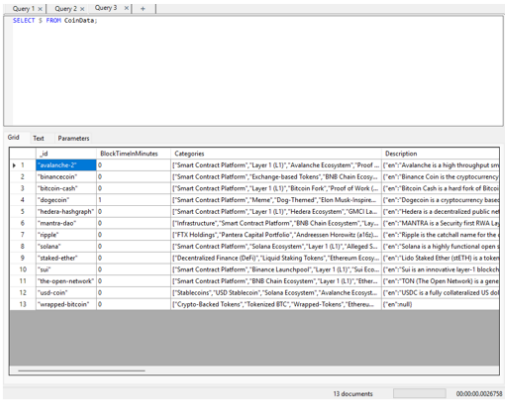
Головне вікно додатку



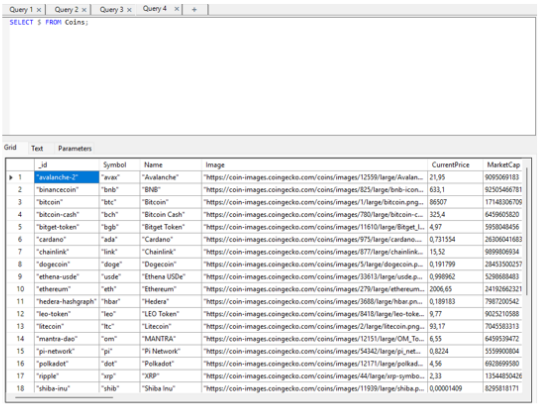
Вікно з детальною інформацією про криптовалюту

11

ДЕМОНСТРАЦІЯ РОЗРОБКИ ПЛАТФОРМИ



Збережені дані валют з описом і маркетів



Збережені дані валют з загальною інформацією

12

ВИСНОВКИ

1. Проаналізовано основні принципи та функціонування криптовалют, їхні переваги та недоліки.
2. Розглянуто сучасні технології розробки програмного забезпечення та інструменти, які забезпечують ефективну роботу з криптовалютами.
3. Досліджено різні API для отримання даних про криптовалюти, а також методи їх обробки та аналізу
4. Спроектовано та розроблено функціональний додаток для збору та відображення даних про курс криптовалют.
5. В результаті роботи вдалось створити застосунок з підтримкою автономного режиму завдяки кешуванню даних.

АПРОБАЦІЯ

1. Коваленко М. О. IoT для розумних міст та промисловості // Тези доповіді на V Міжнародну науково-технічну конференцію «Сучасний стан та перспективи розвитку IoT», 18 квітня 2024 р. – С. [142 - 143]

2. Коваленко М. О. Застосування Big Data в різних сферах // Тези доповіді на VI Міжнародну науково-технічну конференцію «Сучасний стан та перспективи розвитку IoT», 15 квітня 2025 р. – С. [Подано до друку]