s

**Пояснювальна записка  
до дипломного проєкту**

на тему: Веб-застосунок для відстеження цін на криптоактиви з оповіщенням про зміну ціни

КПІ.ІТз-9104.045440.02.81

Київ – 2023

Зміст

[Вступ 5](#_Toc136732410)

[1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 7](#_Toc136732411)

[1.1 Загальні положення 7](#_Toc136732412)

[1.2 Змістовний опис і аналіз предметної області 9](#_Toc136732413)

[1.3 Аналіз існуючих технологій та успішних IT-проєктів 11](#_Toc136732414)

[1.3.1 Аналіз відомих алгоритмічних та технічних рішень 11](#_Toc136732415)

[1.3.2 Аналіз допоміжних програмних засобів та засобів розробки 13](#_Toc136732416)

[1.3.3 Аналіз відомих програмних продуктів 16](#_Toc136732417)

[1.4 Аналіз вимог до програмного забезпечення 19](#_Toc136732418)

[1.4.1 Розроблення функціональних вимог 26](#_Toc136732419)

[1.4.2 Розроблення нефункціональних вимог 31](#_Toc136732420)

[1.5 Постановка задачі 31](#_Toc136732421)

[Висновки до розділу 34](#_Toc136732422)

[2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 35](#_Toc136732423)

[2.1 Моделювання та аналіз програмного забезпечення 35](#_Toc136732424)

[2.2 Архітектура програмного забезпечення 44](#_Toc136732425)

[2.3 Конструювання програмного забезпечення 47](#_Toc136732426)

[2.4 Аналіз безпеки даних 54](#_Toc136732427)

[Висновки до розділу 54](#_Toc136732428)

[3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 56](#_Toc136732429)

[3.1 Аналіз якості ПЗ 56](#_Toc136732430)

[3.2 Опис процесів тестування 57](#_Toc136732431)

[3.3 Контрольний приклад 65](#_Toc136732432)

[Висновки до розділу 70](#_Toc136732433)

[4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 72](#_Toc136732434)

[4.1 Розгортання програмного забезпечення 72](#_Toc136732435)

[4.2 Підтримка програмного забезпечення 73](#_Toc136732436)

[Висновки до розділу 73](#_Toc136732437)

[ВИСНОВКИ 75](#_Toc136732438)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 76](#_Toc136732439)

[ДОДАТКИ 79](#_Toc136732440)

Перелік умовних позначень

|  |  |  |
| --- | --- | --- |
| API | – | Application programming interface, прикладний програмний Інтерфейс |
| IT | – | Інформаційні технології |
| ERD | – | Entity-Relation diagram |
| OC | – | Операційна система. |
| БД | – | База даних. |

Вступ

Світова економіка зацікавлена в криптовалютах останній час, привертаючи інтерес компаній, трейдерів та інвесторів. Однак ринок криптоактивів, як відомо, є непередбачуваним та високо волатильним, що створює дилему при прийнятті інвестиційних рішень. Відсутність актуальної інформації про ринок може призвести до інвестиційних помилок та втрат активів.

Зростаюча популярність криптовалют і сектор фінансових технологій, що постійно розвивається, стимулює попит на доступні веб-додатки, які стежують за цінами на цифрові активи і сприяють прийняттю обґрунтованих рішень. Гучні імена у сфері фінтеху та криптовалют активно намагаються вирішити цю проблему, але єдиного рішення, яке б задовольнило потреби всіх користувачів, не існує.

Розробка веб-додатку, який дозволяє здійснювати колективний аналіз змін вартості криптоактивів у рамках створення аналітично-трейдингової платформи, може принести значну користь різним секторам. Запропонований проект спрямований на створення такого додатку, що допоможе користувачам відстежувати ціни на різні криптоактиви. Це може бути особливо корисно для індивідуальних та корпоративних інвестицій, торгівлі криптовалютою та аналізу ринку.

Індивідуальні та корпоративні інвестиції, аналітичні агентства, регулятори, що регулюють ринок криптоактивів та торгівлю криптовалютою - можливі сфери застосування розроблених веб-додатків. Легка інтеграція таких інструментів у робочі процеси дозволяє значно підвищити ефективність інвестицій і знизити ризик втрат. В кінцевому підсумку, вони сприяють більш швидкому і якісному прийняттю рішень в сфері торгівлі криптовалютою.

Щодо моніторингу та аналізу криптовалют, то створення веб-додатку, який не тільки забезпечує відстеження цін на криптовалюту, але й дозволяє здійснювати аналіз їх змін, видається високоефективним рішенням. Така ініціатива є важливою для оптимізації аналізу ринку та підвищення якості прийняття рішень.

# АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Загальні положення

У сучасному світі криптовалютні активи стали важливим елементом фінансових операцій, що відкриває нові можливості для інвестицій та заробітку. Щоб використовувати криптовалюти не потрібна ідентифікація, що дає змогу використовувати їх анонімно. В Україні можливість отримувати платежі від анонімних користувачів допомогає волонтерам на денних засадах. [1]

Ринок криптовалют характеризується високою волатильністю, що робить його привабливим для інвесторів, але водночас ускладнює прийняття рішень щодо купівлі та продажу активів. Отже, актуальним є розробка аналітично-трейдингової платформи.

**Аналітично-трейдингова платформа** - це спеціалізований веб-ресурс або програмне забезпечення, що надає користувачам інструменти для здійснення торгівлі криптоактивами та проведення глибокого аналізу ринку. Ця платформа може включати в себе ряд функцій, таких як відстеження актуальних цін криптовалют, отримання сповіщень про зміни цін, автоматичний аналіз та інтерпретація даних ринку, а також різноманітні інструменти для ефективного прийняття рішень щодо купівлі та продажу активів.

**Криптовалюта** – це цифровий або віртуальний актив, який використовує криптографічні методи для забезпечення безпеки транзакцій, контролю створення нових одиниць та верифікації передачі активів та зберігається з використанням технології блокчейн. Прикладом таких криптографічних методів виступають алгоритми «Proof Of Work» [2] або «Proof of Stake» [3].

**Криптовалютний** **ринок** – це організований торгівельний майданчик, що займається укладанням між учасниками ринку угод з куплі та продажу криптоактивів та інших цифрових активів, що можуть бути пов’язані з блокчейн технологіями. Прикладами таких ринків є «Binance», «CoinBase» та інші.

**Блокчейн** [4] – це розподілена база даних, що дозволяє передавати дані у відкритому вигляді та зростає зберігаючи порядок блоків. Гарантує надійність даних використання хешу блоку у наступному блоку, що дозволяє перевірити цепочку записів на дотримання правил.

Аналітично-трейдингова платформа повинна задовольняти різноманітні потреби користувачів, які займаються трейдингом криптоактивами. Ключові функції та інструменти платформи повинні надати зручність та ефективність при здійсненні торгівельних операцій і проведенні глибокого аналізу ринку. Користувачам має бути доступне відстеження актуальних цін криптовалют, отримання сповіщень про зміни цін, автоматичний аналіз та інтерпретація даних ринку, а також різноманітні інструменти для ефективного прийняття рішень щодо купівлі та продажу активів. Графіки цін, індикатори технічного аналізу, можливість створення та відстеження портфеля, а також інструменти для автоматизації торгівлі є важливими компонентами, які забезпечують користувачам повний контроль та можливість прийняття обґрунтованих рішень на основі аналізу ринкових даних. З цього випливає, що актуальність даних, графіки та сповіщення про зміну цін є чи не основними чинниками аналітично-трейдингової платформи.

Сповіщення є важливою складовою аналітично-трейдингової платформи, яка надає користувачам можливість колективного реагування та бути завжди в курсі важливих змін на ринку криптоактивів. Система сповіщень на платформі пропонує різні канали комунікації, такі як електронна пошта, повідомлення в месенджери, для зручності користувачів. Користувачі можуть налаштовувати персоналізовані параметри сповіщень.

Наявність сповіщень допомагає користувачам не пропустити важливі можливості та події на ринку, а також швидко реагувати на зміни ситуації. Відправлення сповіщень про зміну ціни допомагає забезпечити своєчасну інформацію та підтримку користувачів у прийнятті обґрунтованих рішень щодо їх інвестицій та трейдингових стратегій та запобігає втратам на ринку.

Тим не менш розробка аналітично трейдингової платформи є комплексною задачою і є за рамками дипломного проекту. Але з вище сказано було сформульовано основні чинники що має мати дана платформа. Предметною областю даного дипломно проекту буде розробка веб-застосунка для відстеження цін на криптоактиви з оповіщенням про зміну ціни, що міг би стати важливою складовою більшої аналітично-трейдингової платформи. Основними аспектами цієї предметної області буде збір та аналіз даних про криптовалюти з багатьох джерел даних для забеспечення їх актуальності, розробка користувацького інтерфейсу, реалізація гнучкої системи сповіщень про зміну вартості ціни на криптоактиви з врахуванням побажань користувачів, та забеспечення безпеки та конфіденційності інформації що надають нам користувачі.

Розвиток даної предметної області відбувається під впливом технологічних нововведень, таких як блокчейн-технології, та спрямований на забезпечення ефективного інструментарію для аналітичної та трейдингової діяльності на ринку криптоактивів

## Змістовний опис і аналіз предметної області

Наразі існує ряд ІТ-рішень, зі схожою предметною областю, таких як мобільні додатки, веб-сайти, біржі та аналітичні платформи. Проте, ці рішення мають певні недоліки, зокрема:

* Недостатня швидкість реагування на зміни цін, що призводить до пропущених можливостей та втрат інвестицій.
* Обмежена кількість криптоактивів, які можуть бути відстежені.
* Відсутність індивідуальних налаштувань сповіщень та аналізу ринку.
* Проблеми з безпекою користувацьких даних та конфіденційність інформації.

Для поліпшення ситуації з ІТ-розробками у предметній області можна розглянути наступні шляхи:

* Застосування передових алгоритмів та технологій для швидкого відстеження змін цін криптоактивів.
* Розширення списку доступних криптовалют для відстеження та аналізу.
* Розробка зручного користувацького інтерфейсу з можливістю індивідуальних налаштувань сповіщень та аналізу ринку.
* Забезпечення безпеки користувацьких даних та конфіденційності інформаці ї за допомогою сучасних методів шифрування та аутентифікації.

У рамках даного дипломного проекту обрано шлях розробки веб-застосунка для відстеження цін на криптоактиви з оповіщенням про зміну ціни. Вибір обумовлений наступними причинами:

* Веб-додатки є доступними для широкого кола користувачів, незалежно від їх операційної системи чи пристрою.
* Швидке відстеження змін цін криптоактивів та надсилання сповіщень дозволить користувачам вчасно реагувати на ринкові зміни.
* Врахування індивідуальних налаштувань сповіщень та аналізу ринку забезпечить користувачам гнучкість вибору та контроль над отриманою інформацією.
* Забезпечення безпеки користувацьких даних та конфіденційності інформації є важливим аспектом розвитку програмного забезпечення та довіри користувачів.

Отже, розробка веб-додатку, який відповідає потребам і вимогам користувачів допоможе забезпечити швидке та якісне аналітичне рішення в галузі криптовалютного трейдингу своїм користувачам.

## Аналіз існуючих технологій та успішних IT-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації аналізу цін на криптоактиви. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

### Аналіз відомих алгоритмічних та технічних рішень

При розробці веб-додатку для аналізу вартості криптоактивів, важливо розглянути наявні алгоритми та архітектурні рішення, які допоможуть вирішити проблеми та задачі, пов'язані з цією розробкою.

#### Аналіз архітектури додатка

Серед технічних рішень ми розглянемо наступні:

1. Мікросервісна архітектура: ця архітектура передбачає розбиття системи на невеликі незалежні компоненти, які відповідають за окремі функції. Мікросервіси спрощують розробку, тестування та розгортання програмного забезпечення, а також дозволяють легко масштабувати систему.
2. Монолітна архітектура - це підхід до розробки програмного забезпечення, при якому всі компоненти системи об'єднуються в одному кодовому репозиторії та розгортаються як єдиний застосунок. У монолітній архітектурі різні частини системи, такі як фронтенд, бекенд, база даних та інші сервіси, взаємодіють безпосередньо один з одним. Це забезпечує простоту розробки, але може призвести до збільшення складності та меншої гнучкості при розширенні та масштабуванні системи.

#### Аналіз математичних алгоритмічних рішень

Веб застосунок буде агрегувати дані з багатьох джерел, що виявляє необхідність в обчисленні середніх значень. Серед доступних варіантів є наступні алгоритми:

1. Розрахунок середнього значення: цей алгоритм полягає в сумуванні всіх значень цін на криптоактив і поділі суми на кількість значень. Це найпростіший метод, але може бути вразливим до винятково високих або низьких значень, які можуть з'явитися через проблеми на окремих джерелах.
2. Розрахунок медіани: медіана - це середнє значення, яке розділяє відсортований набір даних на дві рівні частини. Цей алгоритм менш чутливий до винятково високих або низьких значень і надає більш стабільний результат.
3. Розрахунок квартилів: квартилі - це значення, які розділяють відсортований набір даних на чотири рівні частини. Вони використовуються для визначення розподілу даних та можуть допомогти виявити відхилення або аномалії в даних. Зазвичай використовують 0.25 та 0.75 квартиль, а також 0.5 квартиль, що і є медіаною.

#### Підсумок

Після порівняння алгоритмів та архітектур, було вирішено вибрати мікросервісну архітектуру, оскільки вона спрощує розробку на бекенді і дозволяє швидку адаптацію до змін та масштабування системи.

З алгоритмічної точки зору, було вибрано квартилі та медіани, а не середнє значення, тому що ці алгоритми менш чутливі до проблем в окремих джерелах даних, що можуть вплинути на середнє значення.

Отже, було обрано мікросервісну архітектуру для спрощення розробки бекенду та алгоритми квартилів та медіан для точного відображення даних про ціни криптовалют.

### Аналіз допоміжних програмних засобів та засобів розробки

Проаналізуємо відоме на сьогодні допоміжні програмні засоби та засоби розробки, що допоможуть у реалізації веб-застосунку для відстеження цін на криптоактиви з оповіщенням про зміну ціни.

#### Аналіз допоміжних програмних засобів для Back-end частини

Для розкобки мікросервісів Back-End частини, необхідно визначити мову програмування та пакетний менеджер для швидкої та безпечної розробки. Ми розглянемо наступні популярні мови:

1. Rust [5] – мова програмування відома своєю безпекою та ефективною роботою у високонагруженій середі. Rust пропонує систему контролю доступу до пам'яті, яка запобігає небезпечним ситуаціям, таким як зчитування неправильних даних та змагання за ресурси. Проте розробка на ній та поріг входу є вище чим у більш високорівневих аналогів
2. Go [6] - мова програмування, розроблена Google, що відрізняється швидкістю роботи та простотою синтаксису. Є дуже популярною технологією за свою легкість розробки та швидку компіляцію.
3. PHP [7] - популярна мова програмування для розробки веб-додатків. Останнім часом втрачає свою популярність, проте все ще пропонує зручну та легку взаємодію.

Вибір зупинився на Rust через його безпекові гарантії та продуктивність, що забезпечує вищу надійність розробленого веб-додатку, що є критично у сфері фінансів. Для встановлення додаткових залежностей буде використовувати пакетний менеджер Cargo [8].

#### Аналіз допоміжних програмних засобів для Front-End частини

Для Front-End частини, необхідно також розглянути можливі варіанти. На жаль, на даний момент вибір є тільки серед JavaScript фреймворків, так як WebASM технології все ще не користуються великою користувацькою підтримкою, тому пропоную розглянути саме фреймворки. Серед який є наступні:

1. React [9]- популярний фреймворк, що дозволяє створювати інтерактивні веб-додатки з легкими компонентами. Нажаль, має досить посередню швидкодію в порівнянні з конкурентами.
2. Angular [10] - потужний фреймворк, розроблений Google, який підтримує структурований підхід до розробки. Останніми роками втрачає свою популярність на ринку.
3. Svelte [11] - інноваційний фреймворк, який компілює код компонентів безпосередньо в оптимізований JavaScript, що покращує швидкість роботу додатку.

Було вибрано JavaScript з фреймворком React для розробки фронтенду через його широку підтримку, гнучкість та легкість інтеграції з різними бекенд-технологіями. Для передачі даних у режимі реального часу між бекендом та фронтендом будуть використовуватися GraphQL [12] оркестратор. Для встановлення залежностей буде використовуватися yarn [13] пакетний менеджер.

#### Аналіз бази даних

В проекті буде використовуватись база даних, тому необхідно проаналізувати та обрати базу даних, що буде мати високу швидкодію та безпекові гарантії. Для зберігання та обробки даних ми розглядаємо такі системи баз даних, як PostgreSQL [14], Microsoft SQL Server [15] та MongoDB [16]:

1. PostgreSQL - відкрита система об'єктно-реляційної бази даних з високою продуктивністю та надійністю, підтримкою розширеного набору вбудованих функцій та активною спільнотою.
2. Microsoft SQL Server - комерційна система реляційної бази даних з відмінною продуктивністю, безпекою та інтеграцією з іншими продуктами Microsoft, але з обмеженнями щодо платформи та ліцензійних умов.
3. MongoDB - відкрита система баз даних NoSQL з орієнтацією на документи, що пропонує гнучкість та масштабованість, але може бути менш підходящою для структурованих даних та реляційних запитів.

Було обрано PostgreSQL як систему баз даних для нашого проекту через її високу продуктивність, надійність, підтримку розширеного набору вбудованих функцій та активну спільноту. Також ця система не має обмежень щодо ліцензування та не потребує використання Windows системи, що дає можливість гнучкої розробки. Ці фактори забезпечують оптимальне рішення для обробки та зберігання даних про криптоактиви та оповіщення про зміну ціни.

#### Аналіз засобів розробки

Для успішної розброки та тестування програмного забеспечення виникає потреба в використанні інтегрованих середовищ розробки. Ми розглянемо такі середовища програмування, як Visual Studio Code [17], Sublime Text [18] та Jetbrains IDE [19].

1. Visual Studio Code (VSCode) - легкий, але потужний текстовий редактор від Microsoft з широким набором розширень для різних мов програмування та технологій.
2. Sublime Text - швидкий текстовий редактор з гнучкими налаштуваннями та підтримкою багатьох мов програмування.
3. Jetbrains IDE - потужні комерційні середовища розробки від JetBrains, що має вбудовану підтримку для різних мов програмування та фреймворків, але може бути дещо важким для системи.

Ми вибрали VSCode через його легкість, широку підтримку мов програмування та гнучкість налаштувань. Також важливим фактором є безкоштовність програмного забезпечення. Для керування залежностями буде використовуватися Nix [20] cистемний менеджер.

#### Підсумок

У підсумку, для реалізації нашого веб-додатку ми обрали наступні технології та інструменти:

Мова програмування Rust для бекенду через її безпечність, продуктивність та надійність.

GraphQL для передачі даних на клієнтську сторону.

JavaScript та React для розробки фронтенду, оскільки вони дозволяють створювати швидкі та гнучкі інтерфейси користувача.

PostgreSQL як систему баз даних через її високу продуктивність, надійність та підтримку розширеного набору вбудованих функцій.

Visual Studio Code як редактор коду, завдяки його легкості, широкій підтримці мов програмування та розширюваності.

Вибір цих технологій базується на їх перевагах, а також на порівняльному аналізі з альтернативними рішеннями. Таким чином, ми забезпечуємо розвиток ефективного, безпечного та легко супроводжуваного веб-додатку для відстеження цін на криптоактиви з оповіщеннями про зміну ціни.

### Аналіз відомих програмних продуктів

У цьому розділі ми розглянемо готові програмні продукти у нашій предметній області, які частково або повністю реалізують функціональність, описану у технічному завданні. Ми порівняємо наш дипломний проект з CoinMarketCap [21] та Binance [22], використовуючи таблицю 1.3 для порівняння функціональності та особливостей

Таблиця 1.1 – Порівняння з аналогом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функціонал | Дипломний проєкт(CoinSight) | CoinMarketcap | Binance | Пояснення |
| Відстеження цін на криптоактиви | Так | Так | Так | Всі продукти мають функціонал відстеження цін на криптоактиви |
| Сповіщення про зміну ціни | Розширена функціональність, що надає можливість налаштувати зміну ціни не тільки загальної, а й на певній платформі. | Ні | Так, можливість встановити сповіщення щодо зміни ціни на їх платформі | Дипломний проект та Binance мають сповіщення про зміни цін |
| Аналітика по криптоактивам (медіани, відображення історії) | Так (Медіана, квартилі, історія) | Так (Історія, загальний об’єм, капіталізаця) | Так (Медіани, квартилі, історія, загальний об’єм, капіталізації) | Всі проекти реалізовують математичну складову у певному виді |

Продовження таблиці 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Підтримка декількох джерел даних | Так | Так | Ні | Бінанс відображає ціну на їх платформі, та не використовує інші джерела |
| Можливість покупки | Ні в рамках дипломного проекту, але так в рамках аналітично-трейдингової платформи | Ні | Так | Binance надає змогу користувачам робити покупки на їх сервісі |
| UI/UX | Гарний UI/UX | Гарний UI/UX | Перегружений | Бінансом зачасту тяжко користуватись через перегружений інтерфейс різними кнопками та сторінками |

У підсумку, дипломний проект CoinSight має значну цінність через наступні переваги:

1. Розширена функціональність сповіщень про зміну ціни: CoinSight надає можливість налаштувати зміну ціни не тільки загальної, а й на певній платформі, що відрізняє його від CoinMarketCap та робить його більш гнучким, ніж Binance.
2. Аналітика по криптоактивам: CoinSight пропонує медіани, квартилі та історію, що дозволяє користувачам отримувати більш глибокий аналіз криптоактивів порівняно з CoinMarketCap.
3. Підтримка декількох джерел даних: На відміну від Binance, CoinSight підтримує дані з різних платформ, що дозволяє користувачам отримувати більш точну та повну інформацію про ринок.
4. Гарний UI/UX: CoinSight пропонує зручний інтерфейс та хороший дизайн, що робить його легшим у використанні порівняно з перегруженим інтерфейсом Binance.

Ці переваги роблять дипломний проект CoinSight цінним та конкурентоспроможним серед інших продуктів на ринку.

## Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є можливість аналітики цін та встановлення вподобань щодо нотифікацій про зміну цін, більше функцій можна побачити на рисунку 1.1.

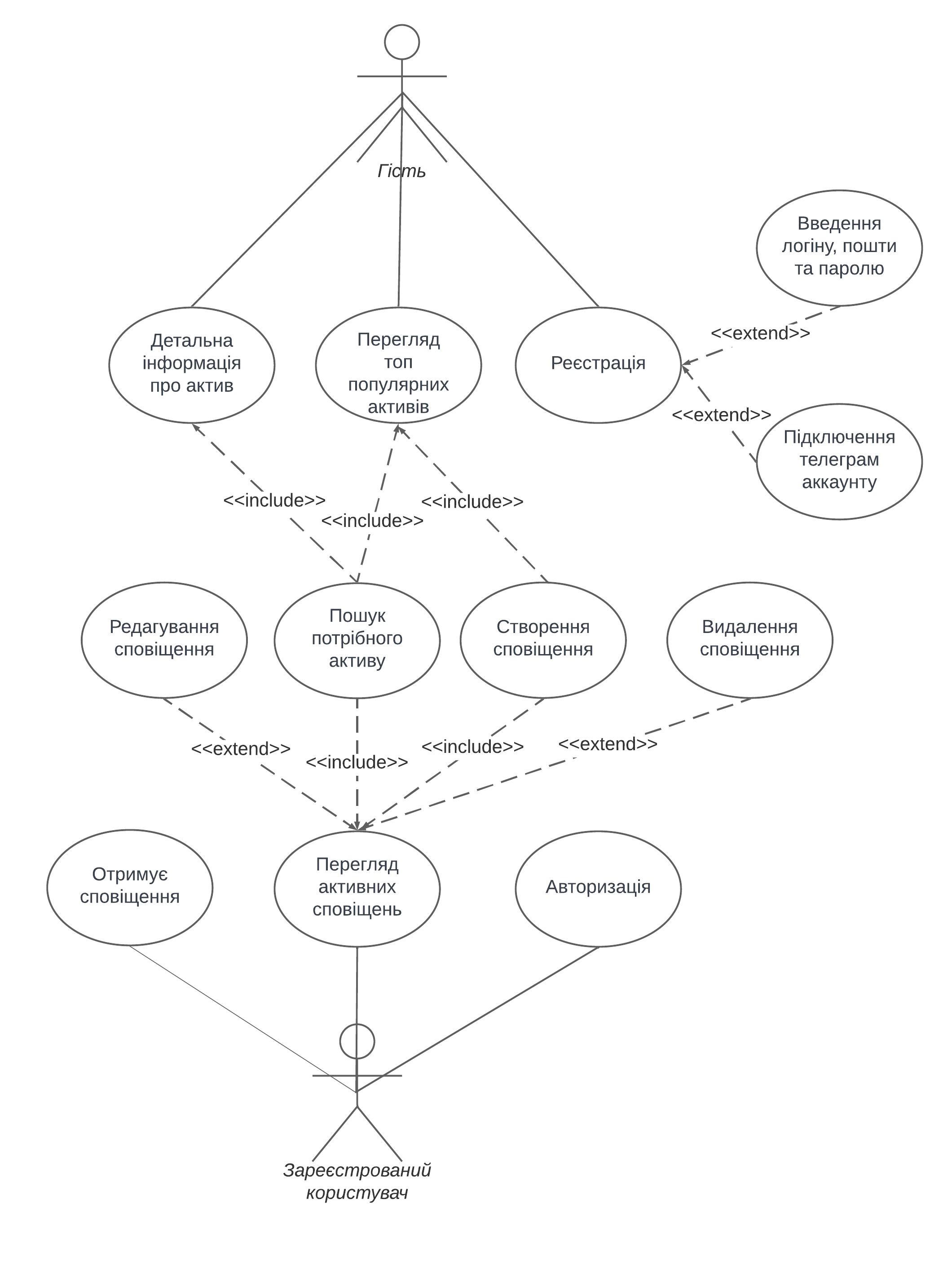


Рисунок 1.1 – Діаграма варіантів використання

В таблицях 1.2 - 1.10 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

|  |  |
| --- | --- |
| Use case name | Реєстрація користувача |
| Use case ID | UC-01 |
| Goals | Реєстрація нового користувача в системі |
| Actors | Гість (незареєстрований користувач) |
| Trigger | Користувач бажає зареєструватися |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: пошта користувача, пароль в системі, та його повтор для підтвердження, а також чек бокс для підтвердження умов сервісу. Після заповнення даних користувача натискає кнопку реєстрації. Після цього з’являється повідомлення про успішну реєстрацію, і користувач перенаправляється на сторінку входу. |
| Extension | В випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом. |
| Post-Condition | Створення облікового запису, перехід на сторінку входу |

Таблиця 1.3 - Варіант використання UC-2

|  |  |
| --- | --- |
| Use case name | Авторизація користувача |
| Use case ID | UC-02 |
| Goals | Авторизація користувача в системі |
| Actors | Гість (незареєстрований користувач) |
| Trigger | Користувач бажає увійти в свій аккаунт |
| Pre-conditions | - |
| Flow of Events | Користувач вводить свою електронну пошту та пароль у відповідні поля, а потім натискає кнопку "Увійти". Якщо облікові дані введені правильно, користувач буде перенаправлений на свою особисту панель управління. |

Продовження таблиці 1.3

|  |  |
| --- | --- |
| Extension | Якщо введені облікові дані невірні, відображається повідомлення про помилку, і користувач має можливість ввести дані ще раз. |
| Post-Condition | Користувач входить в систему і перенаправляється на головну сторінку. |

Таблиця 1.4 - Варіант використання UC-3

|  |  |
| --- | --- |
| Use case name | Перегляд топ 5 популярних активів |
| Use case ID | UC-03 |
| Goals | Переглянути топ-5 популярних криптовалютних активів |
| Actors | Гість, зареєстрований користувач |
| Trigger | Користувач хоче бачити топ-5 популярних активів чи/або користувач зайшов на вебсайт |
| Pre-conditions | - |
| Flow of Events | Користувач потрапляє на головну сторінку додатку, де відображаються топ-5 популярних криптовалютних активів з їхніми поточними цінами та ринковими даними |
| Extension | - |
| Post-Condition | Користувач отримав доступ до ринкових даних щодо 5 популярних криптовалютних активів. |

Таблиця 1.5 - Варіант використання UC-4

|  |  |
| --- | --- |
| Use case name | Пошук потрібного активу |
| Use case ID | UC-04 |
| Goals | Знайти конкретний криптовалютний актив |
| Actors | Гість, зареєстрований користувач |
| Trigger | Користувач хоче знайти певний актив |
| Pre-conditions | - |

Продовження таблиці 1.5

|  |  |
| --- | --- |
| Flow of Events | Користувач вводить назву або символ потрібного криптовалютного активу в рядок пошуку. По мірі введення відображається список відповідних активів. Користувач вибирає потрібний актив зі списку, щоб отримати доступ до детальної інформації про нього. |
| Extension | Якщо відповідні активи не знайдено, відображається повідомлення про те, що результатів не знайдено. |
| Post-Condition | Користувач знаходить і обирає потрібний актив. Його перенаправляє на сторінку з детальною інформацією про цей актив. |

Таблиця 1.6 - Варіант використання UC-5

|  |  |
| --- | --- |
| Use case name | Відкриття детальної інформації про актив |
| Use case ID | UC-05 |
| Goals | Переглянути детальну інформацію про конкретний криптовалютний актив |
| Actors | Гість, зареєстрований користувач |
| Trigger | Користувач хоче переглянути детальну інформацію про актив |
| Pre-conditions | Користувач шукав або вибрав певний актив |
| Flow of Events | Користувач натискає на потрібний актив з результатів пошуку або зі списку популярних активів. Після цього він перенаправляється на сторінку з деталями активу, де відображається вичерпна інформація, включаючи ціну, історичні дані, графіки та іншу релевантну інформацію. |
| Extension | - |
| Post-Condition | Користувач переглядає детальну інформацію про обраний криптовалютний актив |

Таблиця 1.7 - Варіант використання UC-6

|  |  |
| --- | --- |
| Use case name | Налаштування сповіщення про зміну ціни |

Продовження таблиці 1.7

|  |  |
| --- | --- |
| Use case ID | UC-06 |
| Goals | Налаштувати сповіщення для певного криптовалютного активу про зміну його ціни |
| Actors | Зареєстрований користувач |
| Trigger | Користувач хоче отримувати сповіщення про зміну цін |
| Pre-conditions | Користувач увійшов до системи |
| Flow of Events | Користувач переходить на сторінку деталей активу і натискає кнопку "Налаштувати сповіщення". З'являється панель налаштувань сповіщення, де користувач може задати умови для сповіщення, наприклад, відсоток або абсолютне значення зміни ціни. Також є можливість обрати конкретне джерело з доступних варіантів. Встановивши бажані умови, користувач натискає кнопку "Зберегти". |
| Extension | - |
| Post-Condition | Налаштовується сповіщення, і користувач отримає оповіщення при виконанні заданих умов |

Таблиця 1.8 - Варіант використання UC-7

|  |  |
| --- | --- |
| Use case name | Редагування сповіщення |
| Use case ID | UC-07 |
| Goals | Редагування існуючого сповіщення |
| Actors | Зареєстрований користувач |
| Trigger | Користувач хоче змінити існуюче сповіщення |
| Pre-conditions | Користувач увійшов до системи та має принаймні одне активне сповіщення |
| Flow of Events | Користувач переходить на свою персональну інформаційну панель, де відображаються всі активні сповіщення. Користувач обирає сповіщення, яке він хоче відредагувати, і натискає кнопку "Редагувати". Користувач змінює налаштування сповіщення за бажанням і натискає кнопку "Зберегти". |

Продовження таблиці 1.8

|  |  |
| --- | --- |
| Extension | - |
| Post-Condition | Сповіщення оновлюється з урахуванням нових налаштувань |

Таблиця 1.9 - Варіант використання UC-8

|  |  |
| --- | --- |
| Use case name | Видалення сповіщення |
| Use case ID | UC-08 |
| Goals | Видалити існуюче сповіщення |
| Actors | Зареєстрований користувач |
| Trigger | Користувач хоче видалити існуюче сповіщення |
| Pre-conditions | Користувач увійшов до системи та має принаймні одне активне сповіщення |
| Flow of Events | Користувач переходить на свою персональну інформаційну панель, де відображаються всі активні сповіщення. Користувач вибирає сповіщення, яке він хоче видалити, і натискає кнопку "Видалити". З'являється діалогове вікно підтвердження, і користувач підтверджує видалення |
| Extension | - |
| Post-Condition | Вибране сповіщення видаляється з акаунта користувача |

Таблиця 1.10 - Варіант використання UC-9

|  |  |
| --- | --- |
| Use case name | Перегляд та налаштування активних сповіщень |
| Use case ID | UC-09 |
| Goals | Переглядати та надати можливість налаштувати сповіщення |
| Actors | Зареєстрований користувач |
| Trigger | Користувач хоче переглядати активні сповіщення та керувати ними |
| Pre-conditions | Користувач увійшов до системи |

Продовження таблиці 1.10

|  |  |
| --- | --- |
| Flow of Events | Користувач переходить на особисту інформаційну панель, де відображаються всі активні сповіщення. Користувач може переглянути деталі кожного сповіщення, відредагувати налаштування або видалити сповіщення за потреби. Крім того, він може створювати нові сповіщення з цієї панелі. Користувач може налаштувати методи якими буде доставлене повідомлення. |
| Extension | Якщо сповіщення користувача не знайдено, відображається сторінка з повідомленням, що сповіщень не знайдено та пропонує створити нове. |
| Post-Condition | Користувач може переглядати та керувати своїми активними сповіщеннями відповідно до своїх уподобань |

### Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.2 наведено загальну модель вимог, а в таблицях 1.11 – 1.26 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити в таблиці 1.27.



Рисунок 1.2 – Модель вимог у загальному вигляді

Таблиця 1.11 – Функціональна вимога FR-1

|  |  |
| --- | --- |
| Назва | Реєстрація користувача |
| Опис | Система повинна надавати можливість реєстрації користувачеві шляхом введення пошти, паролю, підтвердження паролю. |

Таблиця 1.12 – Функціональна вимога FR-2

|  |  |
| --- | --- |
| Назва | Авторизація користувача |
| Опис | Система повинна надавати можливість авторизації користувачеві шляхом введення пошти та паролю. |

Таблиця 1.13 – Функціональна вимога FR-3

|  |  |
| --- | --- |
| Назва | Перегляд топ-5 найпопулярніших активів |
| Опис | Система повинна відображати топ-5 найпопулярніших активів на основі ринкової капіталізації або обсягу торгів. |

Таблиця 1.14 – Функціональна вимога FR-4

|  |  |
| --- | --- |
| Назва | Відкриття інформації про актив |
| Опис | Система повинна надавати детальний огляд кожного активу, включаючи його поточну ціну, історичні цінові дані та іншу відповідну інформацію. |

Таблиця 1.15 – Функціональна вимога FR-5

|  |  |
| --- | --- |
| Назва | Агрегація даних з різних джерел |
| Опис | Система повинна отримувати дані про активи з різних джерел і відображати їх на платформі, надаючи більш повне уявлення про ринок. |

Таблиця 1.16 – Функціональна вимога FR-6

|  |  |
| --- | --- |
| Назва | Оновлення даних |
| Опис | Система повинна регулярно оновлювати інформацію про активи, щоб надавати користувачам найактуальніші дані |

Таблиця 1.17 – Функціональна вимога FR-7

|  |  |
| --- | --- |
| Назва | Пошук інформації про активи |
| Опис | Система повинна відображати різну інформацію про активи, таку як поточна ціна, історичні цінові дані та іншу релевантну інформацію |

Таблиця 1.18 – Функціональна вимога FR-8

|  |  |
| --- | --- |
| Назва | Пошук активів |
| Опис | Система повинна надавати функцію пошуку, яка дозволяє користувачам знаходити конкретні активи на основі їхніх назв або символів |

Таблиця 1.19 – Функціональна вимога FR-9

|  |  |
| --- | --- |
| Назва | Налаштування сповіщення про зміну ціни |
| Опис | Система повинна дозволяти зареєстрованим користувачам створювати сповіщення про зміну цін, вказуючи відсоток або абсолютну величину зміни |

Таблиця 1.20 – Функціональна вимога FR-10

|  |  |
| --- | --- |
| Назва | Налаштувати сповіщення про зміну ціни на певному джерелі |
| Опис | Система повинна дозволяти зареєстрованим користувачам створювати сповіщення про зміну ціни в певному джерелі, такому як біржа або платформа |

Таблиця 1.21 – Функціональна вимога FR-11

|  |  |
| --- | --- |
| Назва | Редагування сповіщення |
| Опис | Система повинна дозволяти зареєстрованим користувачам змінювати налаштування існуючих сповіщень, наприклад, змінювати поріг зміни ціни або джерело |

Таблиця 1.22 – Функціональна вимога FR-12

|  |  |
| --- | --- |
| Назва | Видалення сповіщення |
| Опис | Система повинна дозволяти зареєстрованим користувачам видаляти існуючі сповіщення |

Таблиця 1.23 – Функціональна вимога FR-13

|  |  |
| --- | --- |
| Назва | Перегляд сповіщень |
| Опис | Система повинна дозволяти зареєстрованим користувачам переглядати створені сповіщення |

Таблиця 1.24 – Функціональна вимога FR-14

|  |  |
| --- | --- |
| Назва | Користувацькі налаштування |
| Опис | Система повинна дозволяти зареєстрованим користувачам керувати своїми особистими налаштуваннями, такими як налаштування сповіщень та іншими параметрами, пов'язаними з профілем |

Таблиця 1.25 – Функціональна вимога FR-15

|  |  |
| --- | --- |
| Назва | Безпека системи |
| Опис | Система повинна забезпечувати безпечне зберігання даних користувачів, таких як адреси електронної пошти та паролі, а також запобігати несанкціонованому доступу до них. |

Таблиця 1.26 – Функціональна вимога FR-16

|  |  |
| --- | --- |
| Назва | Хешування паролів з використанням солі |
| Опис | Система повинна хешувати користувацькі паролі з використанням солі для безпечного зберігання конфіденційних даних. |

Таблиця 1.27 – Матриця трасування вимог

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | FR1 | FR2 | FR3 | FR4 | FR5 | FR6 | FR7 | FR8 | FR9 | FR10 | FR11 | FR12 | FR13 | FR14 | FR15 | FR16 |
| UC1 | х |  |  |  |  |  |  |  |  |  |  |  |  |  | х | х |
| UC2 |  | х |  |  |  |  |  |  |  |  |  |  |  |  | х | х |
| UC3 |  |  | х | х | х | х | х |  |  |  |  |  |  |  |  |  |
| UC4 |  |  |  | х | х | х | х | х |  |  |  |  |  |  |  |  |

Продовження таблиці 1.27

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| UC5 |  |  |  | х | х | х | х |  |  |  |  |  |  |  |  |  |
| UC6 |  |  |  |  |  |  |  |  | х | х |  |  |  |  | х |  |
| UC7 |  |  |  |  |  |  |  |  |  |  | х |  |  |  | х |  |
| UC8 |  |  |  |  |  |  |  |  |  |  |  | х |  |  | х |  |
| UC9 |  |  |  |  |  |  |  |  |  |  |  |  | х | х | х |  |

### Розроблення нефункціональних вимог

У цьому розділі будуть описані нефункціональні вимоги для проекту CoinSight.

1. Надійність. Система повинна забезпечувати стабільну роботу і забезпечення доступності даних користувачам.
2. Продуктивність. Система повинна оптимізувати запити до бази даних та використання ресурсів, щоб забезпечити швидке завантаження сторінок і відгук системи на запити користувачів.
3. Сумісність. Система повинна бути сумісною з різними браузерами та операційними системами. Також вона повинна підтримувати взаємодію з різними джерелами даних про криптоактиви.
4. Безпека. Система повинна забезпечувати безпеку даних користувачів і системи в цілому. Це означає, що повинні бути реалізовані заходи щодо авторизації, захисту від несанкціонованого доступу та витоку даних.

## Постановка задачі

У цьому розділі буде описано, що саме планується реалізувати у дипломному проекті CoinSight.

Мета. Покращити та забеспечити гнучкі та надійні сповіщення користувачам щодо змін актуальної ціни на ринку криптоактивів для надання можливості користувачам приймати обґрунтовані рішення щодо криптовалютних активів та їхнього портфелю.

Цілі:

1. розробити систему, яка дозволяє користувачам аналізувати криптоактиви, відображати їх ціну та історію ціни;
2. реалізувати можливість налаштовувати сповіщення про зміну цін, зокрема зміну ціни на конкретних джерелах данних;
3. забезпечити підтримку декількох джерел даних для отримання надійної та актуальної інформації про криптоактиви;
4. реалізувати зручний та інтуїтивно зрозумілий інтерфейс для користувачів з різним рівнем досвіду в криптовалютному просторі;
5. забезпечити безпеку даних користувачів та стабільність роботи системи;

Завдання:

1. Описати предметне середовище та постановити задачі;
2. спроектувати функціональні вимоги та сценарії використання застосунку;
3. спроектувати нефункціональні вимоги застосунку;
4. спроектувати архітектуру застосунку;
5. спроектувати програмні компоненти архітектури застосунку;
6. спроектувати інформаційні компоненти архітектури застосунку;
7. реалізувати компоненти застосунку;
8. провести тестування продукту;
9. впровадити систему та забеспечети її підтримку;

Функціональні завдання:

1. реєстрація та авторизація користувачів:
   1. Створення облікового запису користувача з унікальним ідентифікатором та паролем;
   2. Збереження особистої інформації користувача для зв'язку та повідомлень;
2. відображення ціни та історії ціни криптоактивів:
   1. виведення поточної ціни криптоактивів у режимі реального часу;
   2. відображення історії ціни криптоактивів за певний період (наприклад, останні 24 години, останній тиждень);
3. налаштування сповіщень про зміну цін:
   1. можливість встановлення порогових значень для криптоактивів, при перетині яких користувач отримує сповіщення;
   2. налаштування джерел даних, з яких відбувається моніторинг зміни цін;
4. підтримка декількох джерел даних:
   1. інтеграція з різними криптовалютними біржами та іншими джерелами для отримання актуальної інформації про ціни криптоактивів;
5. забеспечення безпеку даних:
   1. зберігання даних в безпечній формі, що забеспечить високий рівень безпеки користувацьких даних

У результаті розробки та впровадження програмного забезпечення CoinSight для дипломного проекту має бути створено рішення, яке допомагає користувачам аналізувати вартість криптоактивів, налаштовувати сповіщення про зміни цін та отримувати актуальну інформацію з різних джерел. Таким чином, програмний продукт допоможе користувачам приймати обґрунтовані рішення щодо криптовалютних активів та їхнього портфелю.

## Висновки до розділу

У даному розділі було виконано декілька ключових завдань, пов'язаних з вивченням та аналізом предметної області, а також формуванням вимог до програмного забезпечення CoinSight для дипломного проекту.

* У розділі 1.1 було надано огляд сучасного стану ринку криптовалют та висвітлено актуальність теми дипломної роботи.
* У розділі 1.2 була виконана детальна характеристика та аналіз основних аспектів предметної області, що стосуються криптовалют та їх відстеження.
* У розділі 1.3 було проведено порівняльний аналіз існуючих програмних продуктів, які пропонують функціонал для відстеження криптоактивів, а також виділено переваги та недоліки кожного з них.
* У розділі 1.4 було виконано аналіз функціональних та нефункціональних вимог до програмного забезпечення CoinSight. Було сформовано відповідні варіанти використання системи та визначено ключові характеристики продукту.
* У розділі 1.5 було сформульовано мету, основні цілі та завдання, які необхідно вирішити в результаті розробки програмного забезпечення для дипломного проекту.

На основі проведеного аналізу та вивчення предметної області у даному розділі, було сформовано чітке розуміння вимог до програмного забезпечення CoinSight та сформовано основу для подальшого проектування, розробки та тестування системи у наступних розділах диплом

# МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Моделювання та аналіз програмного забезпечення

В наступних етапах розробки веб-платформи для моніторингу криптоактивів, необхідно провести аналіз та побудувати відповідну модель. Ці дії дозволять чітко визначити та структурувати процеси, що відповідатимуть вимогам, висунутим вище. Для наочного представлення бізнес-процесів була використана нотація BPMN [23].

BPMN - це стандартний спосіб побудови діаграм, які послідовно відображають кроки бізнес-процесів у планованому веб-сервісі від початку до завершення. Завдяки BPMN, усі користувачі мають можливість ясно розуміти ключові бізнес-процеси, які реалізовані в програмному забезпеченні.

В рисунках 2.1 – 2.8 будуть наведені основні бізнес процеси веб застосунку щодо аналізу зміни ціни на криптоактиви в рамках розробки аналітично-трейдингової платформи.

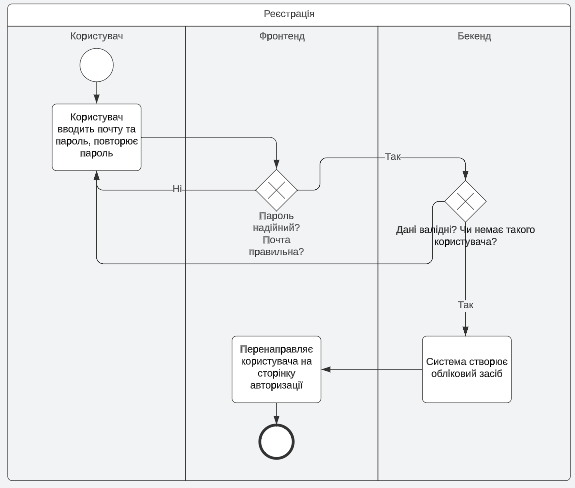


Рисунок 2.1 – Схема бізнес процесу реєстрації

Опис послідовності реєстрації облікового запису користувача:

* користувач переходить на сторінку реєстрації;
* користувач вводить свою електронну адресу та двічі пароль;
* якщо введені дані не коректні користувач з такою поштою вже існує в системі, то користувачу відображається помилка;
* якщо введені дані коректні, система створює обліковий засіб та перенаправляє користувача на сторінку входу;

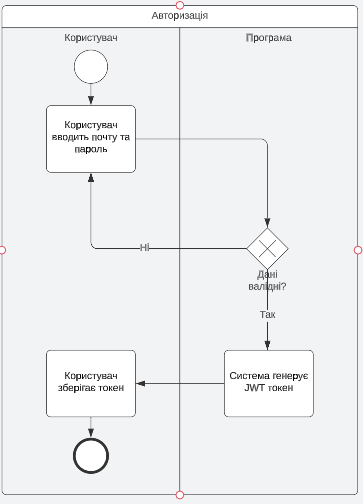


Рисунок 2.2 – Схема бізнес процесу авторизації

Опис послідовності авторизації в обліковий запис користувача:

* користувач переходить на сторінку авторизації;
* користувач вводить свою електронну адресу та пароль;
* якщо введені дані не співпадають з даними в системі, користувачу відображається повідомлення про помилку та пропонує ввести знову;
* якщо введені дані коректні, система генерує JWT токен та повертає користовачу;

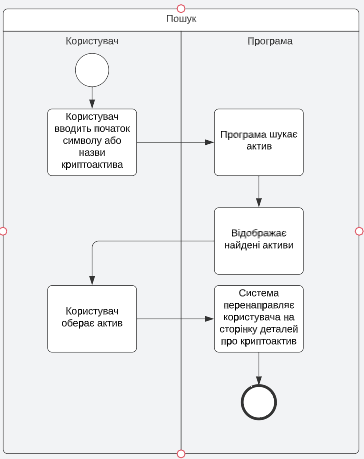


Рисунок 2.3 – Схема бізнес процесу пошуку

Опис послідовності пошуку криптоактива в веб додатку:

* користувач вводить дані в стрічку пошуку;
* програма шукає та повертає користувачу знайдені активи;
* якщо не знайдено, з’являється повідомлення, що дані відсутні, в іншому випадку відображають топ 10 активів;
* користувач може клікнути по активу, щоб перейти на сторінку актива;

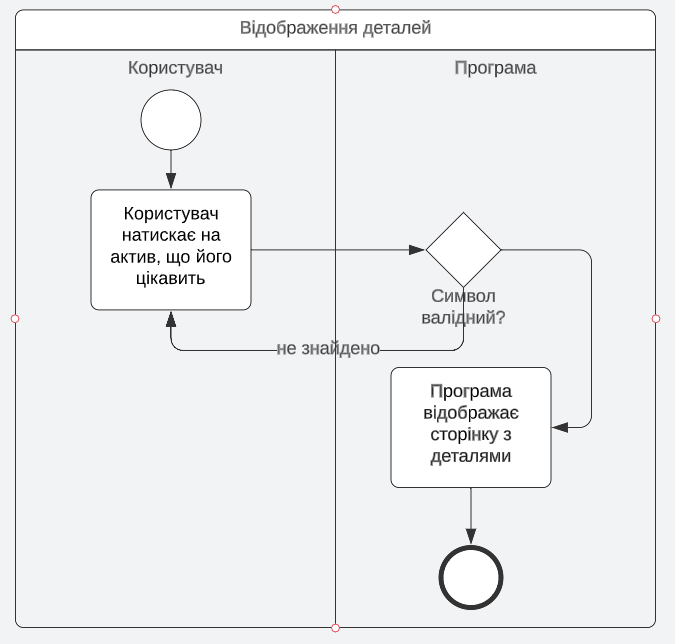


Рисунок 2.4 – Схема бізнес процесу відображення деталей щодо криптоактиву

Опис послідовності відкриття деталей про криптоактив:

* користувач обирає цікавий йому криптоактив та натискає на нього на головній сторінці чи в пошуку;
* користувач переходить на сторінку з інформацією;

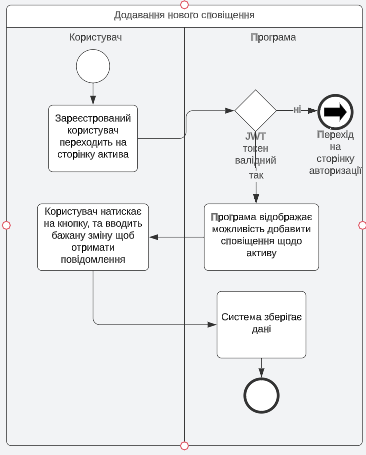


Рисунок 2.5 – Схема бізнес процесу добавлення нового сповіщення

Опис послідовності додавання нового сповіщення для зареєстрованого користувача:

* користувач переходить на сторінку криптоактиву;
* система перевіряє чи користувач авторизований, якщо так, показує йому кнопку добавлення нового сповіщення;
* користувач заповнює деталі новостворюваного сповіщення
* система зберігає дані;

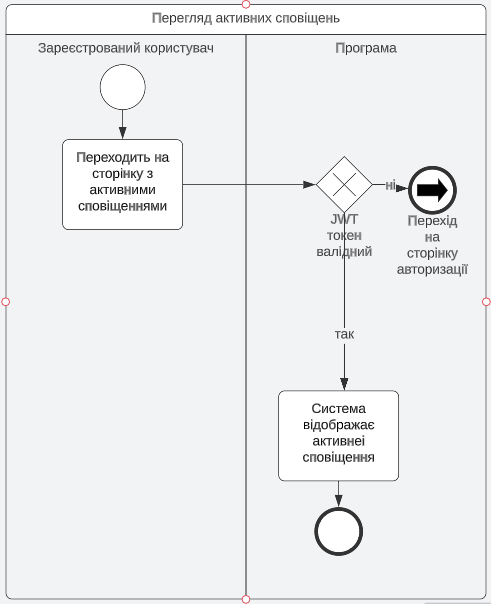


Рисунок 2.6 – Схема бізнес процесу перегляду активних сповіщень

Опис послідовності перегляду активних сповіщень:

* користувач заходить на сторінку з активними сповіщеннями;
* система валідує що користувач авторизований і токен валідний;
* система відображає активні сповіщення;



Рисунок 2.7 – Схема бізнес процесу видалення активного сповіщення

Опис послідовності перегляду активних сповіщень:

* зареєстрований користувач заходить на сторінку з активними сповіщенню;
* користувач натискає червоний хрестик на сповіщенню, що хоче видалити;
* система запитує підтвердження;
* користувач підтверджує;
* система видаляє сповіщення;

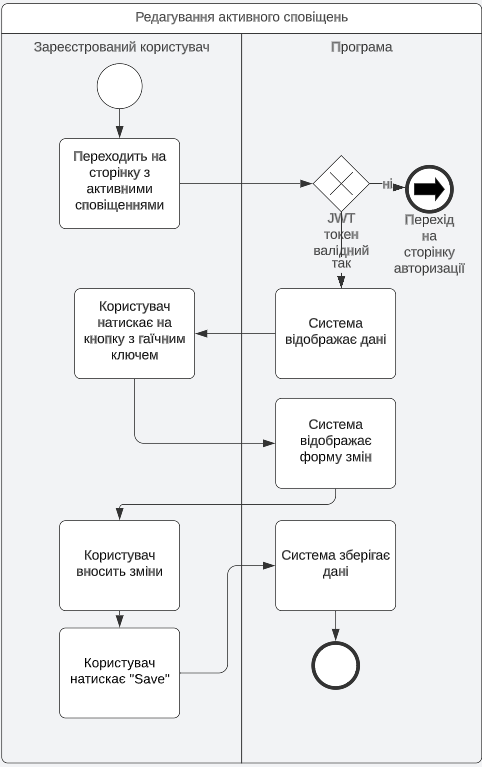


Рисунок 2.8 – Схема бізнес процесу редагування активного сповіщення

Опис послідовності перегляду активних сповіщень:

* зареєстрований користувач заходить на сторінку з активними сповіщенню;
* користувач натискає на кнопку з гаїчним ключем на сповіщенню, що хоче відредагувати;
* система відображає форму змін;
* користувач вводить нові дані, та натискає кнопку “Save”;
* система зберігає зміни;

## Архітектура програмного забезпечення

В цьому розділі буде розглянути високорівневу архітектуру веб застосунку для аналізу вартості криптоактивів в рамках розробки аналітично-трейдингової платформи. В рисунку 2.9 відображена високорівнева архітектура додатку. В рисунку 2.10-2.11 показано ERD [24] діаграми баз даних.

Для програмного забезпечення CoinSight було обрано мікросервісну архітектуру, яка дозволяє легко масштабувати та розширювати систему. Кожен мікросервіс виконує свою власну завдання, і вони взаємодіють між собою через визначені інтерфейси та протоколи.

Для вирішення задачі буде розроблено наступні мікросервіси:

* GraphQL API Оркестратор: Цей мікросервіс відповідає за обробку запитів від клієнтів. Він використовує GraphQL для обміну даними, що дозволяє клієнтам отримувати лише ті дані, які їм потрібні.
* Менеджер користувацьких даних: Цей мікросервіс керує користувацькими обліковими записами, включаючи реєстрацію, вхід, обробку персональних даних користувачів.
* Менеджер користувацьких сповіщень: Цей сервіс відповідає за надсилання повідомлень користувачам про зміну цін на криптоактиви.
* Збирачі даних щодо криптовактивів: Дана група мікросервісів збирають дані про ціни з різних джерел.
* Агрегатор цін: Цей сервіс об'єднує дані, отримані від Price Collectors, та забезпечує одну консолідовану ціну.
* Клієнт: Даний сервіс надає користувачам можливість взаємодіяти з системою шляхом веб інтерфейсу.

Ця архітектура дозволяє системі бути гнучкою та стійкою до помилок, так як кожен мікросервіс може бути розгорнутий, масштабований та відновлений незалежно.

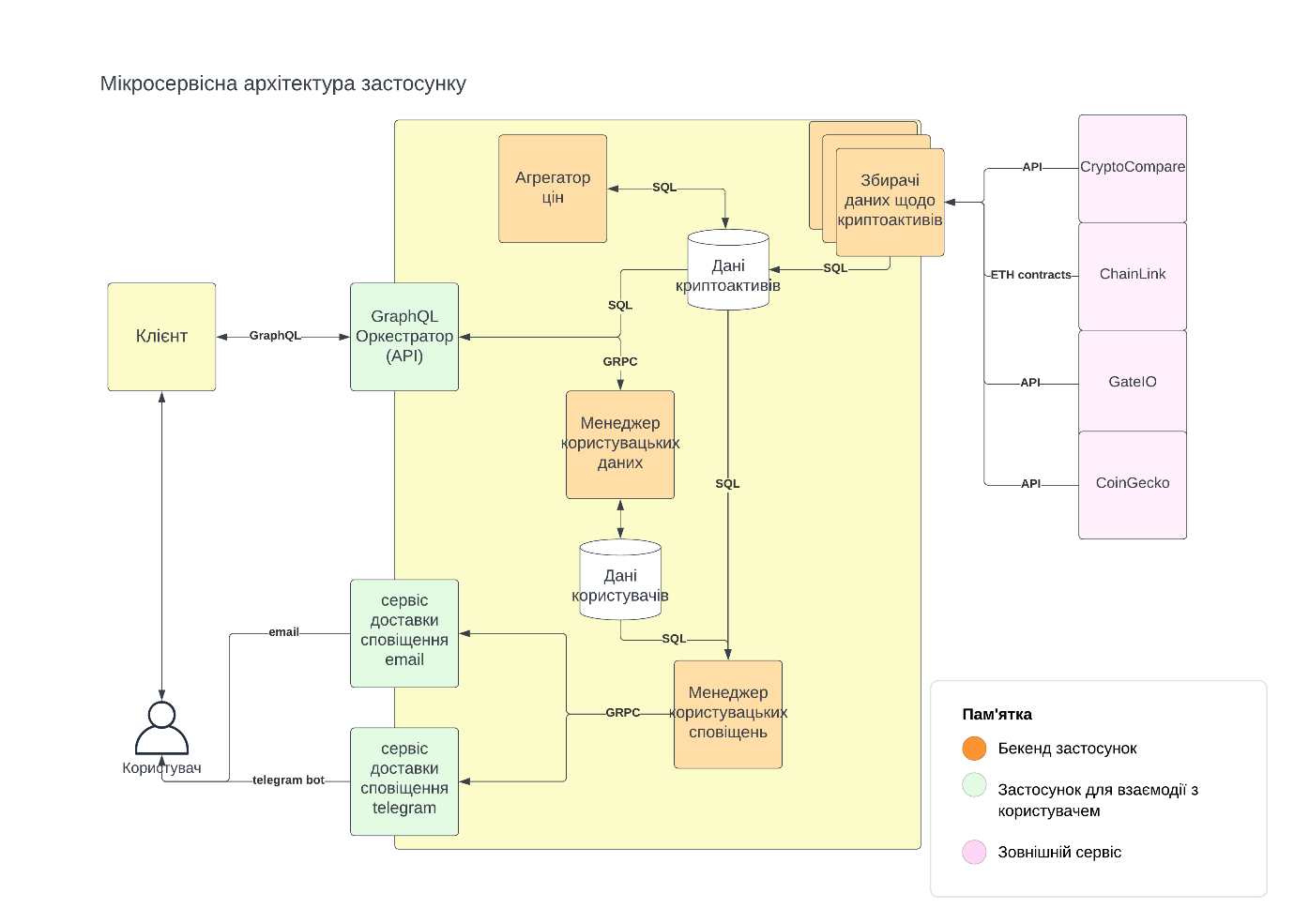


Рисунок 2.9 – Мікросервісна архітектура CoinSight

Також для розробки цього рішення є необхідним використанням двух баз даних. Одна з них відповідатиме за збереження користувацької інформації, а інша буде адмініструвати історичні та актуальні дані щодо вартості криптоактивів. Рисунки 2.10 і 2.11 ілюструють ERD діаграми вищеназваних баз даних.

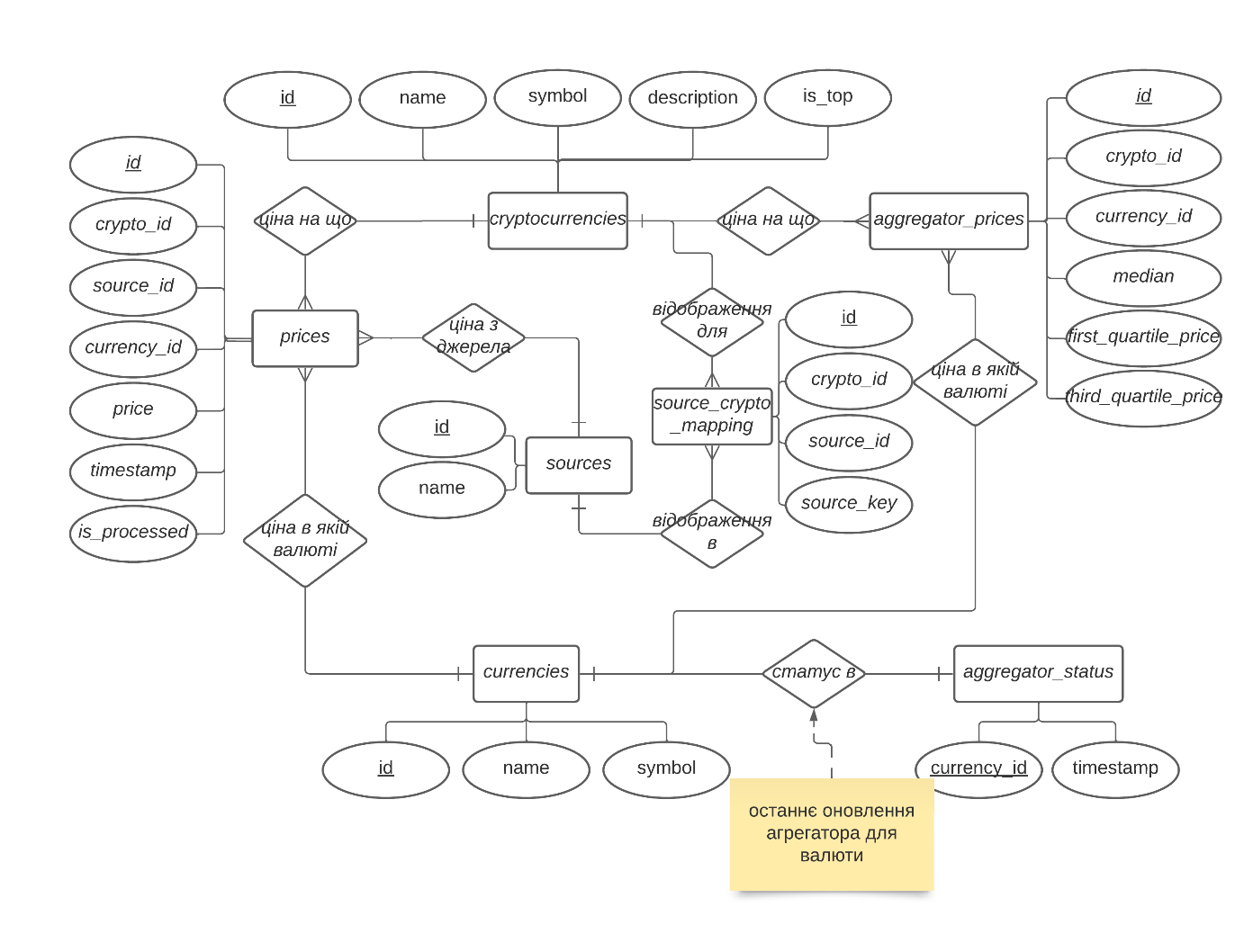


Рисунок 2.10 – ERD діаграма бази даних про історичні та актуальні дані криптоактивів

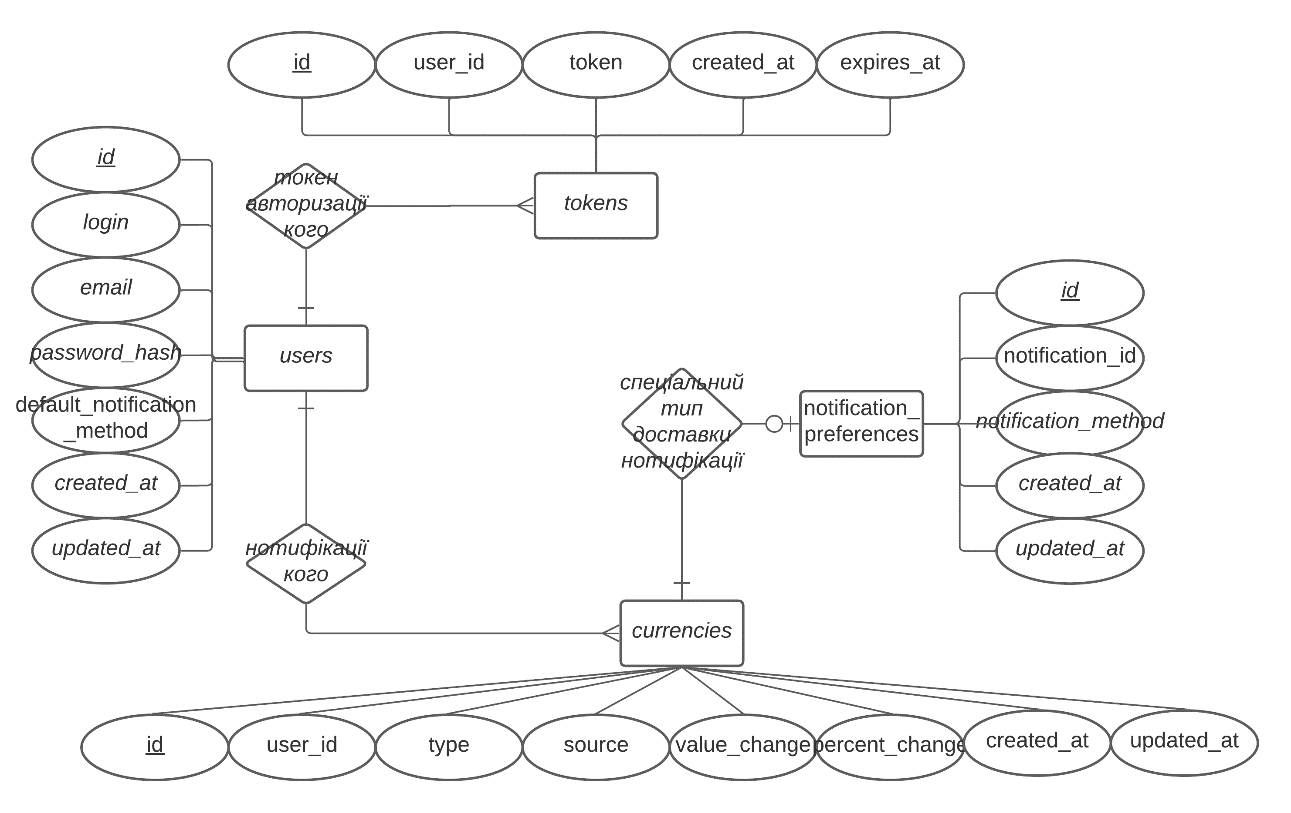


Рисунок 2.11 – ERD діаграма бази даних користувацьких даних

## Конструювання програмного забезпечення

У процесі проектування програмного забеспечення було виявлено проблему пов’язану з використанням різних джерел інформації. Ці джерела могли мати різні ключі для однакових криптовалют. Враховуючи велику кількість джерел інформації та необхідність надійної та справжньої інформації нашим користувачам, ця проблема ставала все більш важливою.

Для її вирішення було прийнято рішення створити спеціальну таблицю, яка включає в себе відображення між джерелами. Таблиця "source\_crypto\_mappings" була розроблена для створення унікальних зв'язків між криптовалютами та відповідними джерелами. Вона включає в себе такі поля, як ідентифікатор, ідентифікатор криптовалюти, ідентифікатор джерела та ключ джерела.

Це дозволило нам ефективно використовувати різні джерела, незважаючи на їх внутрішню структуру та специфіку, а також оптимізувати процес обробки даних. Такий підхід покращив надійність нашого сервісу та забезпечив більш точне відображення інформації про криптовалюту з різних джерел.  
 У проекті було вирішено винести основну логіку роботи з базою даних та запитами до джерел даних в окремий клас під назвою Collector. Цей клас відповідає за збирання цін на криптоактиви. Він виконує запити до сервера та парсить отримані дані щохвилини.

Однак, оскільки джерела даних можуть мати різні підходи до збереження і структури даних, була необхідна унікальна логіка парсингу та формування запиту для кожного джерела. Для цього було створено інтерфейс CollectorHelper, який реалізований для кожного джерела окремо.

Цей інтерфейс дозволяє визначити специфічні методи для кожного джерела, які виконують необхідний парсинг та формування запиту для отримання цін криптоактивів. Кожен клас, який реалізує цей інтерфейс, має свою унікальну логіку, але відповідає загальним вимогам збирача цін.

Таким чином, шляхом винесення основної логіки роботи з базою даних та API в окремий клас Collector і використання інтерфейсу CollectorHelper для унікального парсингу та створення запитів, було досягнуто більшої гнучкості та простоти розширення системи для різних джерел даних. Кожне джерело може мати свою власну реалізацію CollectorHelper, що дозволяє ефективно збирати та обробляти дані з різних джерел в єдиному колекторі цін.

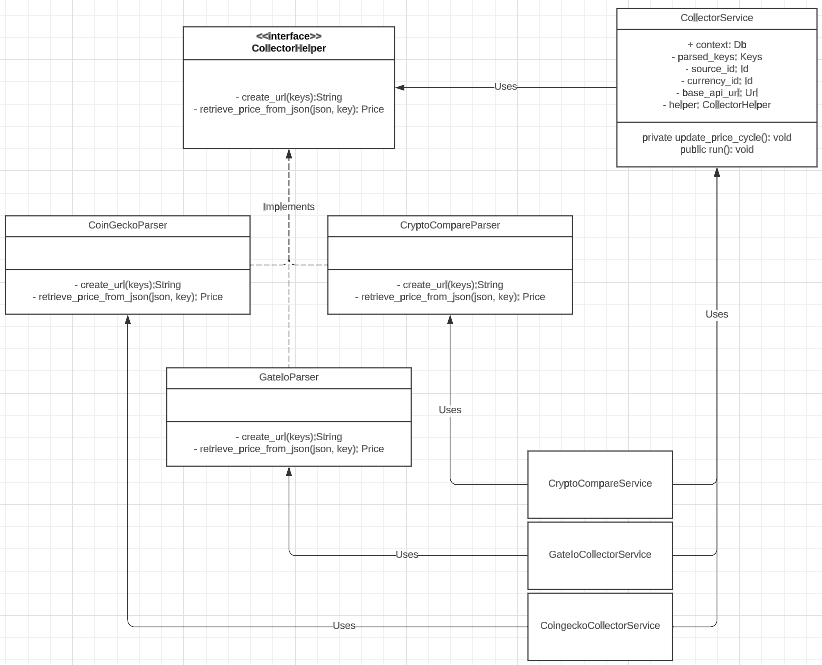


Рисунок 2.12 UML діаграма класів роботи сервісів збирання даних

Postgres було обрано в якості системи управління базами даних. Проект налічує дві бази даних, що зберігають дані про криптоактиви та персональні дані користувачів. Опис таблиць бази користувацьких даних наведено в таблицях 2.1-2.4. ERD модель бази користувацьких даних наведень в рисунку 2.11. Опис таблиць бази криптоактивів наведено в таблиця 2.5-2.11. ERD модель відображено в рисунку 2.10.

Розглянемо таблиці користувацької бази даних:

Таблиця 2.1 – Опис таблиці users

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| id | SERIAL | ідентифікаційний номер користувача |
| email | VARCHAR(255) | електронна пошта користувача |
| password\_hash | VARCHAR(255) | хеш пароля користувача |
| salt | VARCHAR(255) | сіль для хешування пароля |
| default\_notification\_preferences | VARCHAR(255) | типові налаштування повідомлень користувача |
| created\_at | TIMESTAMP | час створення запису про користувача |
| updated\_at | TIMESTAMP | час оновлення запису про користувача |

Таблиця 2.2 – Опис таблиці tokens

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| id | SERIAL | ідентифікаційний номер токена |
| user\_id | INTEGER | посилання на запис у таблиці users, що відповідає користувачу |
| token | VARCHAR(255) | JWT токен |
| created\_at | TIMESTAMP | час створення запису про токен |
| expires\_at | TIMESTAMP | час закінчення терміну дії токена |

Таблиця 2.3 – Опис таблиці notifications

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| id | SERIAL | ідентифікаційний номер сповіщення |
| user\_id | INTEGER | посилання на запис у таблиці users, що відповідає користувачу |
| type | VARCHAR(255) | тип повідомлення |
| source | VARCHAR(255) | джерело повідомлення |
| value\_change | DECIMAL | зміна значення |
| percent\_change | DECIMAL | відсоткова зміна |
| created\_at | TIMESTAMP | час створення запису про сповіщення |
| updated\_at | TIMESTAMP | час оновлення запису про сповіщення |
| is\_active | BOOLEAN | позначає, чи є сповіщення активним (true - так, false - ні) |

Таблиця 2.4 – Опис таблиці notification\_preferences

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| id | SERIAL | ідентифікаційний номер налаштування сповіщення |
| notification\_id | INTEGER | посилання на запис у таблиці notifications, що відповідає сповіщенню |
| notification\_method | VARCHAR(255) | метод повідомлення |
| created\_at | TIMESTAMP | час створення запису про налаштування сповіщення |
| updated\_at | TIMESTAMP | час оновлення запису про налаштування сповіщення |

Розглянемо таблиці бази даних про криптоактиви:

Таблиця 2.5 – Опис таблиці cryptocurrencies

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| id | SERIAL | ідентифікаційний номер криптовалюти |
| name | VARCHAR(255) | назва криптовалюти |
| symbol | VARCHAR(50) | символ криптовалюти |
| description | TEXT | опис криптовалюти |
| is\_top | BOOLEAN | позначає, чи має криптовалюта відображатись на головній (true - так, false - ні) |

Таблиця 2.6 – Опис таблиці sources

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| id | SERIAL | ідентифікаційний номер джерела |
| name | VARCHAR(255) | назва джерела |

Таблиця 2.7 – Опис таблиці currencies

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| id | SERIAL | ідентифікаційний номер валюти |
| name | VARCHAR(255) | назва валюти |
| symbol | VARCHAR(50) | символ валюти |

Таблиця 2.8 – Опис таблиці aggregator\_status

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| currency\_id | INTEGER | посилання на запис у таблиці currencies, що відповідає валюті |
| timestamp | TIMESTAMP | час останнього оновлення агрегатора даних |

Таблиця 2.9 – Опис таблиці prices

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| id | SERIAL | ідентифікаційний номер ціни |
| crypto\_id | INT | посилання на запис у таблиці cryptocurrencies, що відповідає криптовалюті |
| source\_id | INT | посилання на запис у таблиці sources, що відповідає джерелу |
| currency\_id | INT | посилання на запис у таблиці currencies, що відповідає валюті |
| price | DECIMAL(20, 8) | ціна криптовалюти в заданій валюті |
| timestamp | TIMESTAMP | час запису ціни |
| is\_processed | BOOLEAN | позначає, чи була ціна оброблена агрегатором (true - так, false - ні) |

Таблиця 2.10 – Опис таблиці source\_crypto\_mappings

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| id | SERIAL | ідентифікаційний номер зв'язку між криптовалютою та джерелом |
| crypto\_id | INT | посилання на запис у таблиці cryptocurrencies, що відповідає криптовалюті |
| source\_id | INT | посилання на запис у таблиці sources, що відповідає джерелу |
| source\_key | VARCHAR(255) | ключ у джерелі щодо криптовалюти |

Таблиця 2.11 – Опис таблиці aggregated\_prices

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| id | SERIAL | ідентифікаційний номер агрегованої ціни |
| crypto\_id | INT | посилання на запис у таблиці cryptocurrencies, що відповідає криптовалюті |
| currency\_id | INT | посилання на запис у таблиці currencies, що відповідає валюті |
| median\_price | DECIMAL(20, 8) | медіанна ціна криптовалюти в заданій валюті |
| first\_quartile\_price | DECIMAL(20, 8) | ціна першого квартилю криптовалюти в заданій валюті |
| third\_quartile\_price | DECIMAL(20, 8) | ціна третього квартилю криптовалюти в заданій валюті |
| timestamp | TIMESTAMP | час запису агрегованої ціни |

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.22.

Таблиця 2.12 – Опис утиліт

|  |  |  |
| --- | --- | --- |
| № п/п | Назва утиліти | Опис застосування |
| 1 | Visaul Studio Code | Головний редактор коду, що використовувася на проекті |
| 2 | Google Chrome | Браузер, що надає зручний інтерфейс для відладки клієнтської сторони додатку |
| 3 | DBeaver | Програмне забеспечення, що надає зручний інтерфейс роботи з різними видами баз даних |

## Аналіз безпеки даних

Використання веб-застосунку накладає обов'язок піклуватися про безпеку даних користувачів, зокрема про захист їх реєстраційних даних. Це вкрай важливо, оскільки надання безпеки - це одна з ключових вимог до будь-якої системи, що обробляє особисті дані користувачів.

В нашому застосунку основними даними для входу є електронна пошта та пароль користувача. Для забезпечення безпеки цих даних, ми використовуємо хешування паролів з використанням унікальної "солі" для кожного користувача. "Сіль" - це випадковий рядок, який додається до пароля користувача перед процесом хешування. Такий підхід робить кожен хеш пароля унікальним, навіть якщо два користувачі використовують однакові паролі. Це означає, що потенційний зловмисник не зможе використовувати готові "радужні таблиці" для швидкого вгадування паролів, а також забезпечує додаткову захищеність від атак "по сусідству".

Таким чином, поєднання криптографічного хешування і використання унікальної "солі" для кожного користувача створює міцний шар безпеки для захисту користувацьких даних у нашому веб-застосунку.

## Висновки до розділу

У даному розділі було виконано декілька ключових завдань, пов'язаних з моделюванням та конструюванням веб-застосунка для відстеження цін на криптоактиви з оповіщенням про зміну ціни.

• У розділі 2.1 було проаналізовано та змодельовано бізнес вимоги за допомогою BPMN діаграм. Дані діаграми покривають всі основні шаблони використання застосунку.

• У розділі 2.2 за результатом аналізу вимог була розроблена верхорівнева архітектура додатку та бази даних. Були створені діаграми, що облегшать підтримку та розуміння всіх деталей системи.

• У розділі 2.3 був описаний процес конструювання веб застосунку та проблеми які вдалось вирішити. Також були описані в деталях всі таблиці баз даних, що використовуються при розробці проекту.

• У розділі 2.4 була проаналізована безпекові властивості застосунку, що показало високий рівень захисту персональних даних користувачів.

В результаті детального аналізу бізнес вимог, було змодельовано процеси діяльності застосунку, сконструйована високорівнева архітектура додатку та бази даних, а також було вирішено ключові технічні проблеми. Враховуючи важливість безпеки користувачів, було також проведено аналіз безпеки застосунку. Отримані результати становлять фундамент для подальшої тестування, та впровадження системи.

# АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Аналіз якості ПЗ

У цьому підрозділі проводиться детальний аналіз якості програмного забезпечення з використанням специфічних метрик. Головною метою цього аналізу є визначення рівня ефективності, надійності та стабільності розробленого програмного забезпечення.

Усі мікросервіси бекенду для цього проекту були розроблені з використанням мови програмування Rust, відомої своєю високою продуктивністю, безпечністю та надійністю. Rust є однією з найбільш популярних мов програмування у сфері системного програмування, а також високопродуктивних веб-додатків. Використання Rust дозволяє створювати високошвидкісні та надійні мікросервіси, що забезпечують високий рівень продуктивності та безпечності.

Для забезпечення високої якості коду та виявлення потенційних проблем на ранніх етапах розробки, було використано інструменти статичного та динамічного аналізу коду - cargo clippy та cargo audit. Cargo Clippy є потужним інструментом для статичного аналізу коду, який перевіряє код на наявність загальних помилок, можливих оптимізацій та використання найкращих практик розробки на Rust.

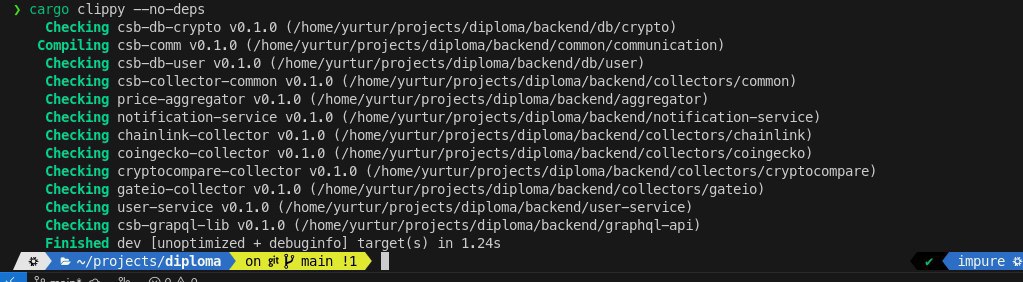


Рисунок 3.1 – Аналіз сargo-clippy

Cargo Audit з іншої сторони, є інструментом для пошуку вразливостей, який здійснює перевірку залежностей проекту на наявність відомих проблем з безпекою. Це важливо, оскільки залежності можуть мати вплив на загальну безпечність програмного забезпечення.

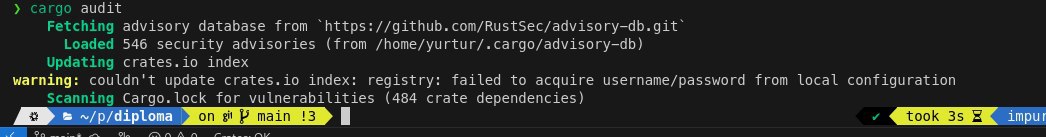


Рисунок 3.2 – Результати аналізу сargo audit

При тестуванні фронтенду особлива увага була приділена якості коду та його відповідності сучасним стандартам розробки. Для цього було використано інструмент статичного аналізу коду ESLint. ESLint є одним з найбільш популярних інструментів для перевірки якості коду в JavaScript-спільноті.

ESLint аналізує код на наявність помилок, потенційних проблем, недотримання встановлених стилів кодування та використання найкращих практик розробки.

Застосування цих інструментів дозволило виявити та виправити декілька потенційних проблем, що покращило якість та надійність програмного забезпечення. Зокрема, ці інструменти допомогли знайти та виправити помилки, які могли б не бути виявлені без їх використання.

Таким чином, цей аналіз якості програмного забезпечення був важливою частиною процесу розробки, що сприяло створенню надійного, ефективного та високоякісного веб-застосунку.

## Опис процесів тестування

Тестування - це важлива частина процесу розробки програмного забезпечення, яка не лише допомагає забезпечити високу якість кінцевого продукту, але і може допомогти виявити та вирішити проблеми на ранніх стадіях розробки. Тестування може бути автоматизовано, але мануальне тестування є важливою складовою процесу, особливо для складних систем, які вимагають великої кількості контекстуальної та виключної логіки.

Мануальне тестування важливе для певних сценаріїв, де автоматизовані тести можуть не враховувати всі можливі випадки використання. Мануальне тестування включає в себе набір дій, які виконуються тестувальником для перевірки функціональності та взаємодії різних частин системи.

Протягом процесу тестування було проведено серію мануальних тестів, результати частини яких представлені у вигляді таблиць 3.1 - 3.12 Кожна таблиця описує конкретний тест, включаючи модуль, який тестується, початковий стан системи, вхідні дані для тесту та опис проведення тесту.

Таблиця 3.1 – Тест.1.1

|  |  |
| --- | --- |
| Тест | Реєстрація |
| Модуль | Реєстрація користувача |
| Номер тесту | 1.1 |
| Початковий стан | Гість знаходиться на сторінці реєстрації |
| Вхідні дані | Електронна пошта, пароль, підтвердження паролю, логін |
| Опис проведення тесту | У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 8 до 64 символів, який містить хоча б одну англійську літеру, одне число і один спеціальний символ, підтвердження паролю, яке співпадає з раніше введеним паролем. Після цього натискається кнопка підтвердження реєстрації |
| Очікуваний результат | Користувач зареєстрований, отримує підтвердження про реєстрацію на електронну пошту |
| Фактичний результат | Користувач зареєстрований, отримує підтвердження про реєстрацію на електронну пошту |

Таблиця 3.2 – Тест 1.2

|  |  |
| --- | --- |
| Тест | Авторизація |
| Модуль | Авторизація користувача |
| Номер тесту | 1.2 |
| Початковий стан | Гість знаходиться на сторінці авторизації |
| Вхідні дані | Електронна пошта та пароль, що відповідають зареєстрованому користувачу |
| Опис проведення тесту | У відповідні поля вводяться електронна пошта та пароль, що відповідають зареєстрованому користувачу. Після цього натискається кнопка "Вхід" |
| Очікуваний результат | Користувач успішно авторизований в системі, відображається персональна сторінка користувача |
| Фактичний результат | Користувач успішно авторизований в системі, відображається персональна сторінка користувача |

Таблиця 3.3 – Тест 1.3

|  |  |
| --- | --- |
| Тест | Перегляд топ активних криптовалют |
| Модуль | Головна сторінка |
| Початковий стан | Користувач (зареєстрований або гість) знаходиться на головній сторінці додатку |
| Номер тесту | 1.3 |
| Вхідні дані | - |
| Опис проведення тесту | Користувач переглядає знаходиться на головній сторінці, та переглядає топ криптовалют |
| Очікуваний результат | Відображається список активних криптовалют з актуальною інформацією про кожну з них |
| Фактичний результат | Відображається список активних криптовалют з актуальною інформацією про кожну з них |

Таблиця 3.4 – Тест 1.4

|  |  |
| --- | --- |
| Тест | Пошук криптовалюти |
| Модуль | Пошук криптовалюти |
| Номер тесту | 1.4 |
| Початковий стан | Користувач (зареєстрований або гість) знаходиться на головній сторінці додатку |
| Вхідні дані | Назва або символ криптовалюти |
| Опис проведення тесту | Користувач вводить у поле пошуку назву або символ криптовалюти |
| Очікуваний результат | Відображається результати пошуку з актуальною інформацією про знайдену криптовалюту |
| Фактичний результат | Відображається результати пошуку з актуальною інформацією про знайдену криптовалюту |

Таблиця 3.5 – Тест 1.5

|  |  |
| --- | --- |
| Тест | Перегляд детальної інформації про криптовалюту |
| Модуль | Детальна інформація про криптовалюту |
| Початковий стан | Користувач (зареєстрований або гість) знаходиться на сторінці конкретної криптовалюти |
| Номер тесту | 1.5 |
| Вхідні дані | - |
| Опис проведення тесту | Користувач переглядає детальну інформацію про вибрану криптовалюту, яку він може обрати з головної сторінки або пошуку |

Продовження таблиці 3.5

|  |  |
| --- | --- |
| Очікуваний результат | Відображається детальна інформація про вибрану криптовалюту, включаючи медіану, перший третій квартиль та графіки цін в джерелах та аналітиці. |
| Фактичний результат | Відображається детальна інформація про вибрану криптовалюту, включаючи медіану, перший третій квартиль та графіки цін в джерелах та аналітиці. |

Таблиця 3.6 – Тест 1.6

|  |  |
| --- | --- |
| Тест | Створення нотифікації на зміну ціни криптовалюти |
| Модуль | Нотифікації |
| Номер тесту | 1.6 |
| Початковий стан | Зареєстрований користувач знаходиться на своїй сторінці нотифікацій |
| Вхідні дані | Назва нотифікації, назва криптовалюти, назва джерела, тип зміни відсотковий чи фактичний, значення типу |
| Опис проведення тесту | Користувач натискає кнопку добавлення нового сповіщення, вводить назву нотифікації, назву криптовалюти, назву джерела, обирає тип зміни та вводить значення |
| Очікуваний результат | Нотифікація успішно створена і з'являється у списку нотифікацій користувача |
| Фактичний результат | Нотифікація успішно створена і з'являється у списку нотифікацій користувача |

Таблиця 3.7 – Тест 1.7

|  |  |
| --- | --- |
| Тест | Створення нотифікації на зміну ціни криптовалюти на сторінці криптоактиву |

Продовження таблиці 3.7

|  |  |
| --- | --- |
| Модуль | Нотифікації |
| Номер тесту | 1.7 |
| Початковий стан | Зареєстрований користувач знаходиться на сторінці криптоактивів |
| Вхідні дані | Назва нотифікації, назва джерела, тип зміни відсотковий чи фактичний, значення типу |
| Опис проведення тесту | Користувач натискає кнопку добавлення нового сповіщення, система автоматичо підставляє назву криптовалюти з сторінки, вводить назву нотифікації, назву джерела, обирає тип зміни та вводить значення |
| Очікуваний результат | Нотифікація успішно створена і з'являється у списку нотифікацій користувача |
| Фактичний результат | Нотифікація успішно створена і з'являється у списку нотифікацій користувача |

Таблиця 3.8 – Тест 1.8

|  |  |
| --- | --- |
| Тест | Редагування створеної нотифікації |
| Модуль | Нотифікації |
| Номер тесту | 1.8 |
| Початковий стан | Зареєстрований користувач знаходиться на сторінці нотифікацій і має хоча б одну створену нотифікацію |
| Вхідні дані | Нові значення назви нотифікації, назви криптовалюти, назви джерела, тип зміни відсотковий чи фактичний, значення типу |
| Опис проведення тесту | Користувач вибирає нотифікацію для редагування, вносить нові дані і натискає кнопку “Ok” |

Продовження таблиці 3.8

|  |  |
| --- | --- |
| Очікуваний результат | Зміни в нотифікації успішно збережені і відображаються в списку нотифікацій користувача |
| Фактичний результат | Зміни в нотифікації успішно збережені і відображаються в списку нотифікацій користувача |

Таблиця 3.9 – Тест 1.9

|  |  |
| --- | --- |
| Тест | Видалення створеної нотифікації |
| Модуль | Нотифікації |
| Номер тесту | 1.9 |
| Початковий стан | Зареєстрований користувач знаходиться на сторінці нотифікацій і має хоча б одну створену нотифікацію |
| Вхідні дані | - |
| Опис проведення тесту | Користувач вибирає нотифікацію для видалення і натискає кнопку “Delete” |
| Очікуваний результат | Нотифікація успішно видалена і більше не відображається в списку нотифікацій користувача |
| Фактичний результат | Нотифікація успішно видалена і більше не відображається в списку нотифікацій користувача |

Таблиця 3.10 – Тест 1.10

|  |  |
| --- | --- |
| Тест | Перегляд створених нотифікацій |
| Модуль | Нотифікації |
| Номер тесту | 1.10 |
| Початковий стан | Зареєстрований користувач знаходиться на сторінці нотифікацій |

Продовження таблиці 3.10

|  |  |
| --- | --- |
| Вхідні дані | - |
| Опис проведення тесту | Користувач переходить на сторінку “Notifications” |
| Очікуваний результат | Користувач бачить список створених нотифікацій |
| Фактичний результат | Користувач бачить список створених нотифікацій |

Таблиця 3.11 – Тест 1.11

|  |  |
| --- | --- |
| Тест | Отримування сповіщення на почту |
| Модуль | Нотифікації |
| Номер тесту | 1.11 |
| Початковий стан | У зареєстрованого користувача є активна нотифікація, умови якої щойно виконалися |
| Вхідні дані | - |
| Опис проведення тесту | Користувач перевіряє свою поштову скриньку |
| Очікуваний результат | Користувач бачить нове сповіщення про зміну ціни криптовалюти, відповідно до умов нотифікації |
| Фактичний результат | Користувач бачить нове сповіщення про зміну ціни криптовалюти, відповідно до умов нотифікації |

Таблиця 3.12 – Тест 1.12

|  |  |
| --- | --- |
| Тест | Отримування сповіщення в телеграм |
| Модуль | Нотифікації |
| Номер тесту | 1.11 |
| Початковий стан | У зареєстрованого користувача увімкнене отримування сповіщення в телеграм та є активна нотифікація, умови якої щойно виконалися |

Продовження таблиці 3.12

|  |  |
| --- | --- |
| Вхідні дані | - |
| Опис проведення тесту | Користувач перевіряє свою телеграм сторінку |
| Очікуваний результат | Користувач бачить нове сповіщення про зміну ціни криптовалюти, відповідно до умов нотифікації |
| Фактичний результат | Користувач бачить нове сповіщення про зміну ціни криптовалюти, відповідно до умов нотифікації |

## Контрольний приклад

Контрольний приклад є важливою складовою процесу тестування. Він допомогає перевірити що базовий користувацький шлях є вірним і не несе помилок.

Для використання створеного додатку може бути достатньо використання без реєстрації.

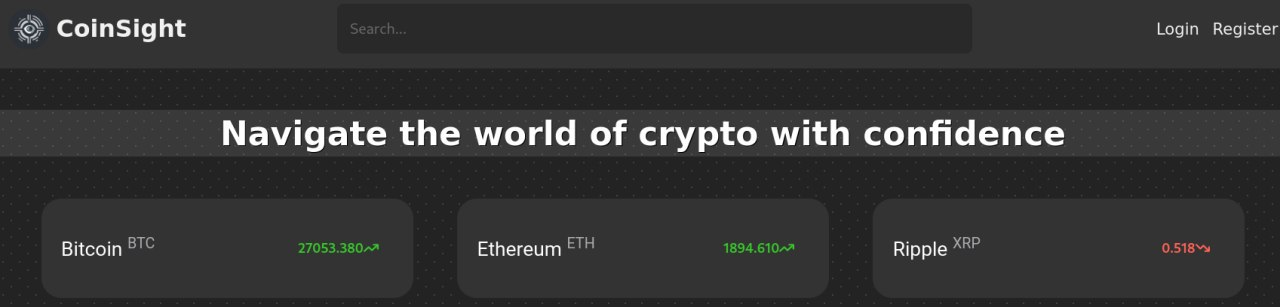


Рисунок 3.3 – Гість знаходиться на головній сторінці

Зазвичай, користувачу цікаво перевірити деталі щодо певного криптоактиву.

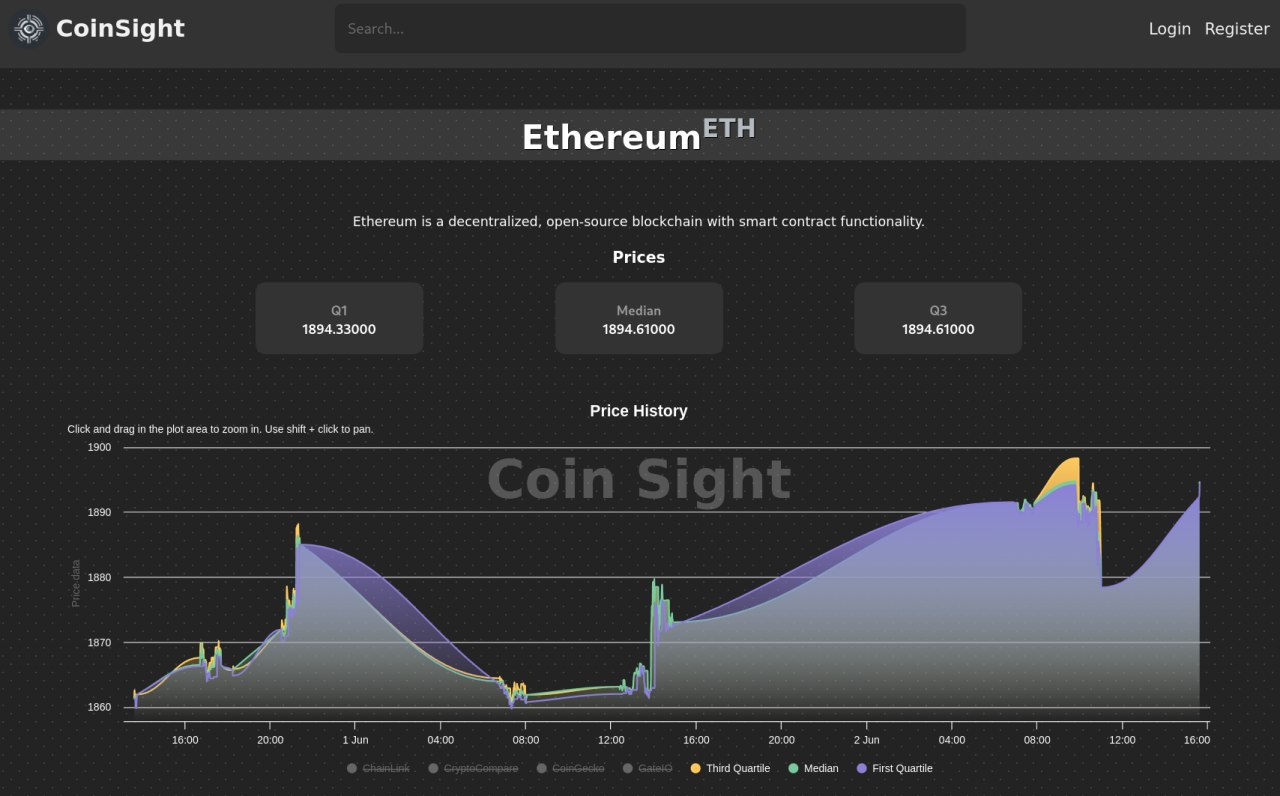


Рисунок 3.4 – Гість на сторінці криптоактиву

Після успішного використання додатку протягом деякого часу, користувач може захотіти зареєструватися, щоб почати використовувати додатковий функціонал сповіщень. Для реєстрації була використана тимчасова пошта.

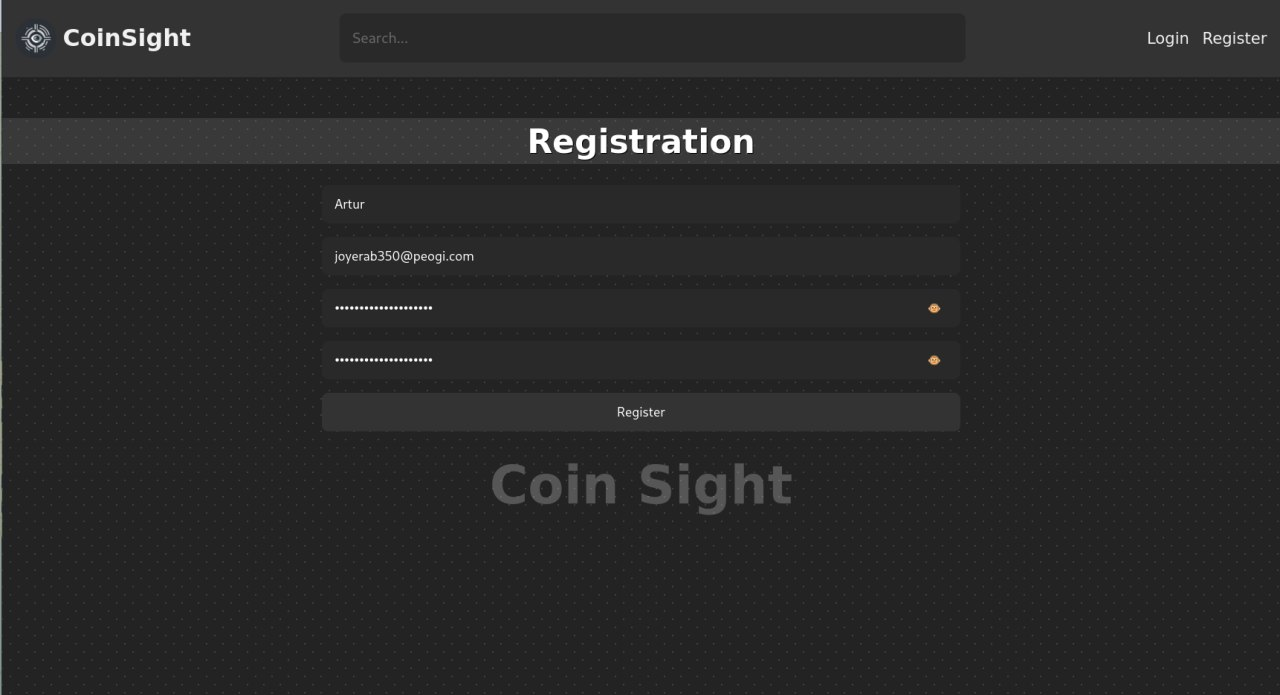


Рисунок 3.5 – Гість на сторінці реєстрації

Успішно зареєстрований користувач перенаправлається на сторінку авторизації.

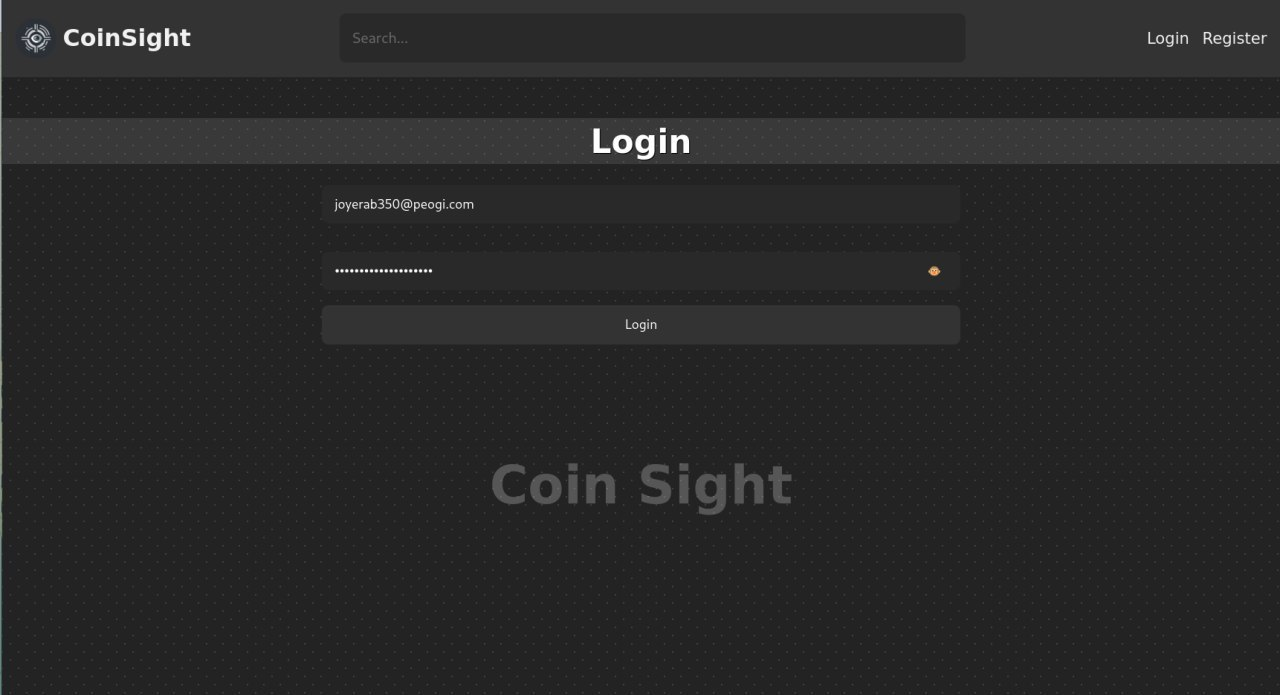


Рисунок 3.6 – Гість на сторінці авторизації

Зареєстрований користувач може захотіти створити сповіщення, щодо зміни ціни.

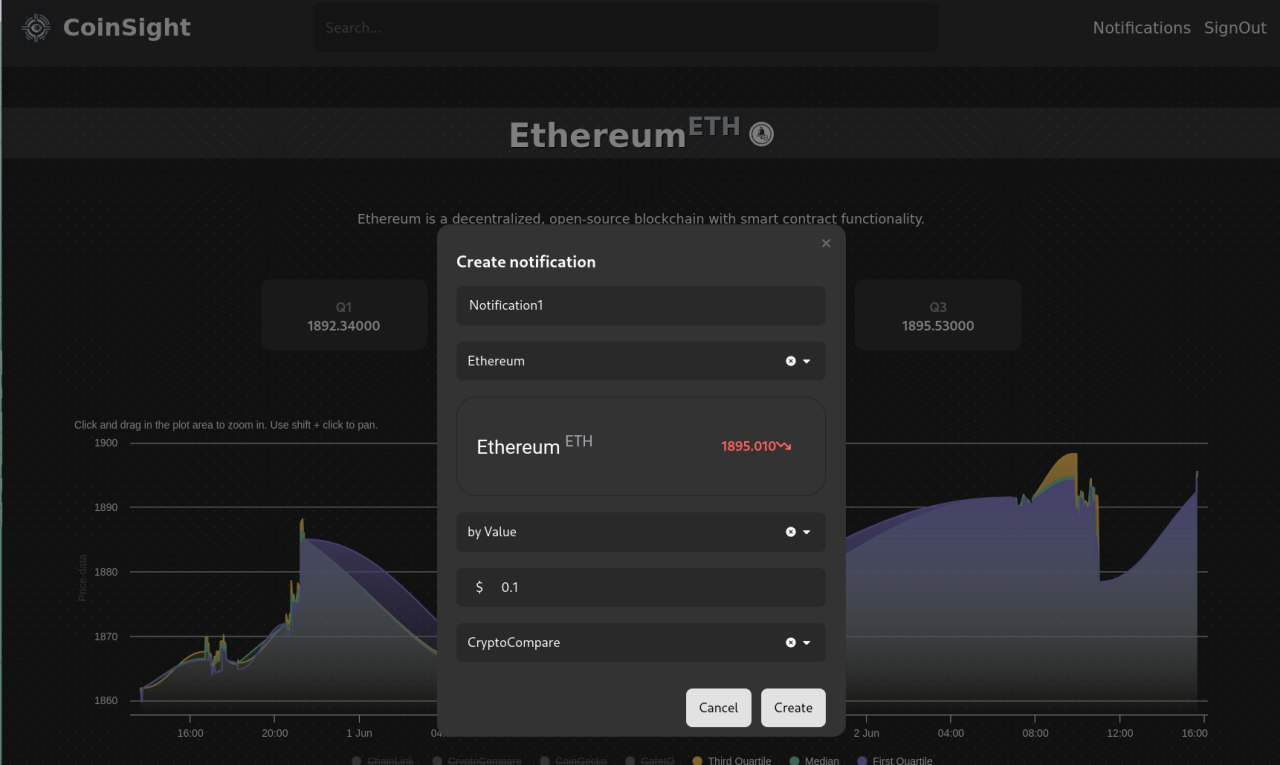


Рисунок 3.7 – Користувач створює сповіщення

Після чого користувач може бути зацікавлений переглянути активні сповіщення.



Рисунок 3.8 – Користувач переглядає активні сповіщення.

Користувач налаштував сповіщення з маленькою зміною в ціні, тому що хотів протестувати цю можливість.

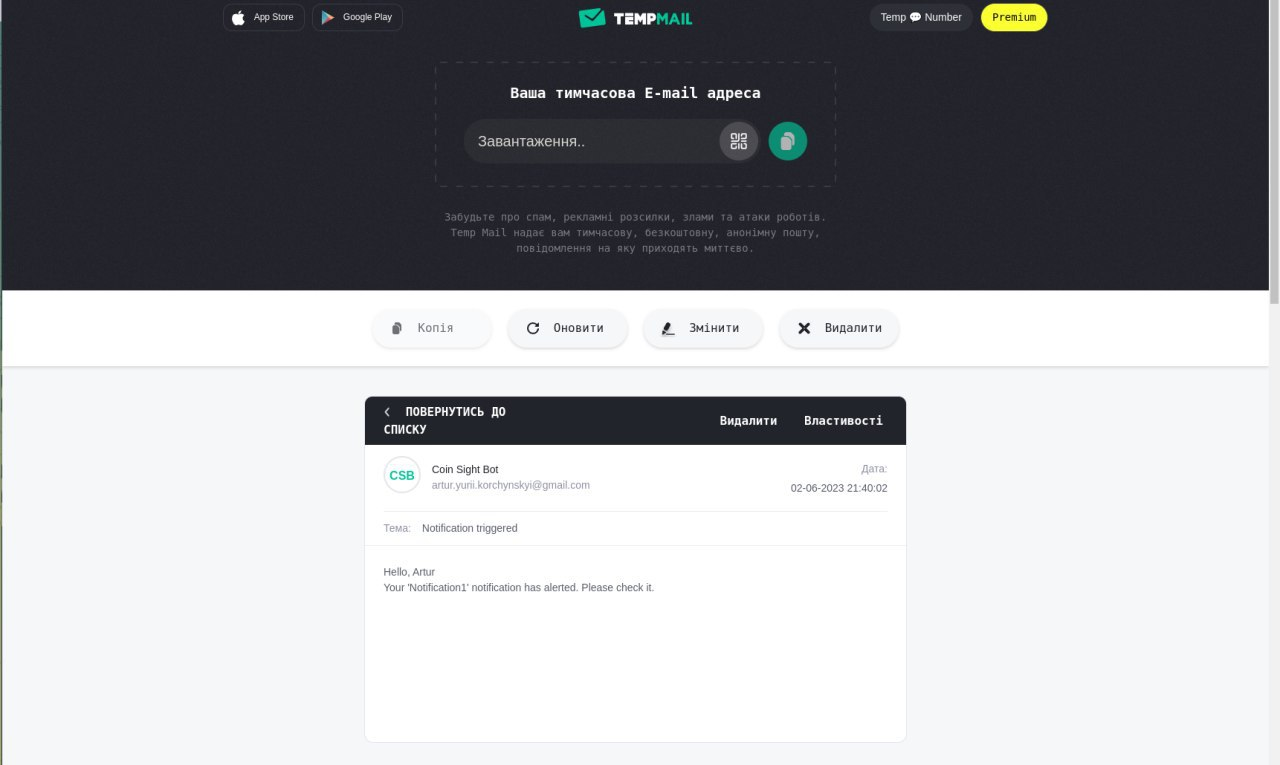


Рисунок 3.9 – Користувач отримує сповіщення на почту

Розуміючи що сповіщення працюють користувач відредагував його на коректне значення.

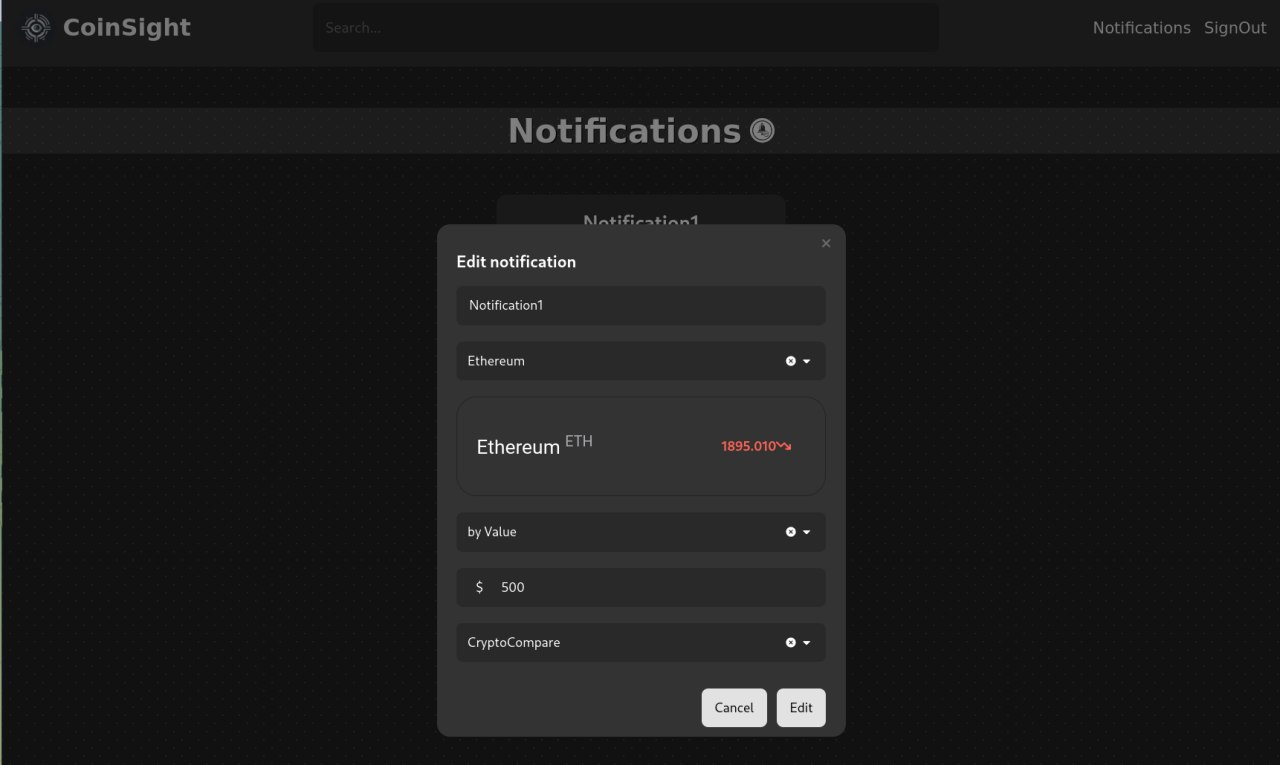


Рисунок 3.10 – Користувач редагує сповіщення

Користувач бачить зміни на сторінці нотифікацій.

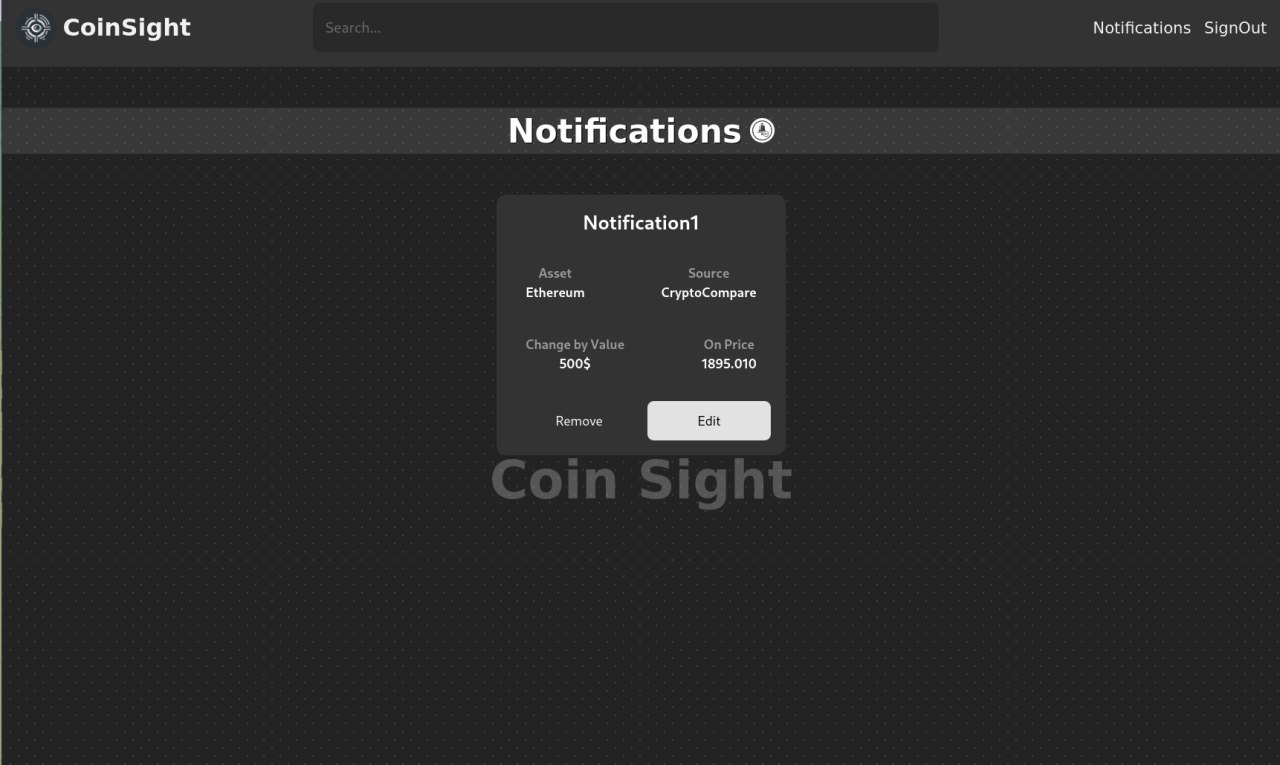


Рисунок 3.11 – Користувач бачить зміни після редагування сповіщення

Користувач передумав використовувати цей функціонал та видалив сповіщення.

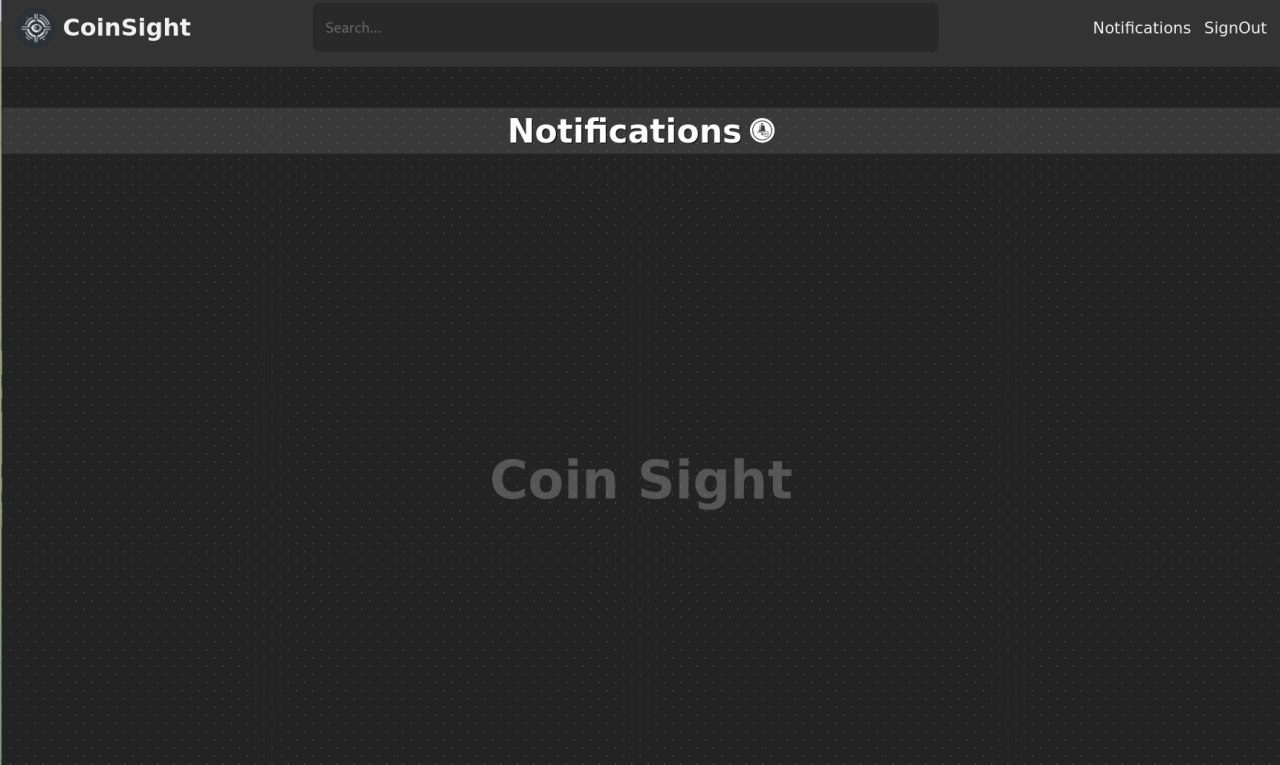


Рисунок 3.12 – Користувач видаляє сповіщення.

Отже, було протестовано середньостатичний користувацький досвід з використанням більшості функціоналу вебзастосунку.

## Висновки до розділу

У даному розділі було виконано тестування новоствореного додатку.

* У розділі 3.1 було проаналізовано створений код на факт частих помилок за допомогою лінтерів коду. Для впевненості в якості коду бекенду, написаного на Rust, ми використовували вбудований в середовище Rust статичний аналізатор Clippy. Цей інструмент допоміг нам знайти потенційні помилки, неоптимальні конструкції та підвищити загальну якість коду. Додатково, для більшої безпеки, ми використовували динамічний аналізатор Cargo Audit, що виявив потенційно небезпечні залежності. Щодо фронтенду, що був написаний на React, ми використовували статичний аналізатор коду ESLint. Він дозволив нам виявити та виправити помилки ще до запуску коду, що сприяло підтримці високого рівня якості коду та забезпеченню його чистоти
* У розділі 3.2 було проведено мануальне тестування, що дало змогу перевірити коректність роботи застосунку. Мануальне тестування додатку було проведено на основі ряду заздалегідь створених сценаріїв, що відображали основні варіанти використання додатку.
* У розділі 3.3 був наданий опис контрольного прикладу з усіма можливими розгалуженнями та особливостями.

В результаті виконання тестування було перевірено, що створений застосунок відповідає стандартам якості галузі, відповідає бізнес лозіці та варіантам використанням, та готовий до впровадження на ринок.

# ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Розгортання програмного забезпечення

Клієнтська та серверна частини програмного забезпечення були успішно розгорнуті на платформі DigitalOcean.

DigitalOcean є однією з провідних хмарних платформ, спеціалізованою на хостингуванні та керуванні інфраструктурою. Вона надає розгортання віртуальних серверів (Droplets), контейнерів (Kubernetes) та зберігання об'єктів (Spaces), сервіси моніторингу (Monitoring) та керування базами даних (Managed Databases).

Для полегшення процесу розгортання та керування інфраструктурою було використано оркестратор контейнерів docker-compose. docker-compose дозволяє упакувати програмне забезпечення та його залежності у контейнери, які можуть бути легко розгорнуті на будь-якому середовищі.

Для забезпечення ефективного спільного розроблення та контролю версій використовується розподілена система управління версіями git разом з хостинговим сервісом GitHub. Розробник може працювати над кодом одночасно, створювати гілки для нових функцій та виправлень, а потім злити зміни в гілку main. Це дозволяє відстежувати зміни, коментувати їх та спілкуватися з іншими людьми.

GitHub Actions використовуються для автоматизації процесу розгортання застосунку. Кожен новий коміт в гілці main запускає створення Docker-образів, які містять вихідний код додатку та його залежності. Після успішної збірки система автоматично підключається до віддаленого сервера та ініціює процес розгортання, що включає реліз програмного забезпечення для користувачів.

Конфігураційні файли, необхідні для розгортання, зберігаються разом з кодом проекту в розподіленій системі управління версіями git. Це забезпечує єдність коду та налаштувань, спрощує процес розгортання та забезпечує високий рівень працездатності системи.

## Підтримка програмного забезпечення

Підтримка програмного забезпечення є одним з ключових етапів у життєвому циклі будь-якого програмного продукту. Вона має на меті не тільки забезпечити працездатність системи, але й підтримувати високий рівень задоволеності користувачів, шляхом регулярного оновлення функціональності та усунення помилок.

В рамках підтримки нашого проекту на платформі DigitalOcean будуть виконуватися ряд дій, спрямованих на забезпечення безперебійної роботи та задоволення потреб користувачів. Один з основних аспектів - це моніторинг та виправлення помилок. DigitalOcean надає можливість підключення до системи та зберігання лог-файлів з усіх запущених додатків. Це дає змогу оперативно ідентифікувати помилки та починати їх усунення.

Крім того, регулярне оновлення програмного забезпечення є важливою складовою підтримки. Ці оновлення відбуваються відповідно до схеми розгортання веб-застосунку, що дозволяє користувачам отримувати їх вчасно та швидко. Актуалізація включає в себе внесення нової функціональності, виправлення помилок та покращення загальної продуктивності системи. Це дозволяє забезпечити стабільну роботу програмного забезпечення та надати користувачам доступ до нових можливостей.

Усі ці дії забезпечують ефективну підтримку програмного забезпечення, що має на меті забезпечити стабільну та надійну роботу системи, а також задоволення потреб та очікувань користувачів.

## Висновки до розділу

Підтримка програмного забезпечення є важливим аспектом у життєвому циклі програмного продукту. Дипломний проект використовує платформу DigitalOcean для розгортання програмного забезпечення. Розгортання відбувається за допомогою docker-compose, а код зберігається та відстежується за допомогою git та GitHub.

Підтримка дипломного проекту відбувається шляхом моніторингу та виправлення помилок. DigitalOcean дозволяє нам підключатись до системи та зберігати лог-файли для швидкого виявлення помилок та їх виправлення. Крім того, ми регулярно оновлюємо наше програмне забезпечення за схемою розгортання веб-застосунку, щоб користувачі могли отримувати оновлення вчасно.

Підтримка програмного забезпечення допомагає забезпечити стабільну роботу нашого продукту та задовольняти потреби користувачів. Вона виконується командою розробників та технічної підтримки, яка дотримується встановлених процедур та методів.

# ВИСНОВКИ

В результаті виконання дипломного проєкту було спроєктовано програмне рішення, що забеспечує користувачам можливість слідкувати за зміною цін на криптоактиви та мати можливість отримувати нотифікації щодо їх змін.

Застосунок був спроектований з впрахуванням передових програмних рішень у галузі криптовалют. Застосунок агрегує дані з передових технологічних гігантів у галузі криптовалютної торгівлі, що забеспечує велику актуальність данних. Застосунок доступний у всіх сучасних веб браузерах при умові доступу до мережі інтернет.

В якості середовища розробки обрано Visual Studio Code через свою доступність, легкість та швидкість роботи, а також підтримку більшості мов програмування за допомогою плагінів.

У якості БД використано PostgresSQL з відкритим програмним кодом через свої якісні переваги у швидкодії та відсутності проблем з комерційним використанням через ліцензійні перебої.

Після реалізації веб застосунка було проведено тестування коду на предмет вразливостей та відомих помилок за допомогою статичних та динамічних аналізаторів коду. Після чого було проведено мануальне тестування, що дало змогу впевнитись в коректності праці застосунку.

Наукова новизна роботи полягає в набутті подальшого розвитку системи сповіщень, що надає користувачам можливість гнучкого налаштування сповіщень щодо зміни вартості криптоактивів на різних платформах джерел даних.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Хом'яков К. Криптовалюта під час війни: миттєві донати та блокчейн-версія "Плану Маршалла". *LIGA*. URL: <https://www.liga.net/ua/economics/opinion/kriptovalyuta-vo-vremya-voyny-mgnovennye-donaty-i-blokcheyn-versiya-plana-marshala> (дата звернення: 03.06.2023).
2. Jakobsson M., Juels A. Proofs of work and bread pudding protocols. 1999. 272 p.
3. Bentov I., Gabizon A., Mizrahi A. Cryptocurrencies without proof of work. 2012.
4. Mylittlewhitepaper. Bitcoin whitepaper: a peer-to-peer electronic cash system. Mylittlewhitepaper, 2022.
5. Mozilla. Rust programming language. Rust Programming Language. URL: https://www.rust-lang.org (date of access: 03.06.2023).
6. Google. The go programming language. The Go Programming Language. URL: https://go.dev/ (date of access: 03.06.2023).
7. Lerdorf R. PHP: hypertext preprocessor. *PHP: Hypertext Preprocessor*. URL: [https://www.php.net](https://www.php.net/" \t "_blank) (date of access: 03.06.2023).
8. Mozilla. Cargo: Rust's community crate host | Rust Blog. The Rust Programming Language Blog. URL: https://blog.rust-lang.org/2014/11/20/Cargo.html#:~:text=Designed%20by%20Carl%20Lerche%20and,last%20for%20years%20to%20come (date of access: 03.06.2023).
9. Facebook. React. React. URL: https://react.dev (date of access: 03.06.2023).
10. Google. Angular. Angular. URL: https://angular.io (date of access: 03.06.2023).
11. Svelte • Cybernetically enhanced web apps. Svelte • Cybernetically enhanced web apps. URL: https://svelte.dev (date of access: 03.06.2023).
12. Facebook. GraphQL | A query language for your API. GraphQL | A query language for your API. URL: https://graphql.org (date of access: 03.06.2023).
13. Facebook. Home. Home | Yarn - Package Manager. URL: https://yarnpkg.com (date of access: 03.06.2023).
14. Stonebraker M., Lawrence A. R. The design of POSTGRES. 1986. 355 p.
15. Microsoft. Microsoft data platform. Microsoft Data Platform. URL: https://www.microsoft.com/en-us/sql-server (date of access: 01.06.2023).
16. MongoDB Inc. MongoDB: the developer data platform. MongoDB. URL: https://www.mongodb.com (date of access: 03.06.2023).
17. Microsoft. Visual studio code - code editing. redefined. Visual Studio Code - Code Editing. Redefined. URL: https://code.visualstudio.com (date of access: 03.06.2023).
18. Sublime HQ. Sublime Text - the sophisticated text editor for code, markup and prose. Sublime Text - Text Editing, Done Right. URL: https://www.sublimetext.com (date of access: 03.06.2023).
19. JetBrains s.r.o. JetBrains: essential tools for software developers and teams. JetBrains. URL: https://www.jetbrains.com (date of access: 03.06.2023). NixOS contributors. Nix & NixOS | Reproducible builds and deployments. Nix & NixOS | Reproducible builds and deployments. URL: https://nixos.org (date of access: 03.06.2023).
20. NixOS contributors. Nix & NixOS | Reproducible builds and deployments. Nix & NixOS | Reproducible builds and deployments. URL: https://nixos.org (date of access: 03.06.2023).
21. CoinMarketCap. Cryptocurrency prices, charts and market capitalizations | coinmarketcap. CoinMarketCap. URL: https://coinmarketcap.com (date of access: 03.06.2023).
22. Binance. Binance - cryptocurrency exchange for bitcoin, ethereum & altcoins. Binance. URL: https://www.binance.com/en (date of access: 03.06.2023).
23. White S. A. BPMN modeling and reference guide: Understanding and using BPMN : develop rigorous yet understandable graphical representations of business processes. Lighthouse Point, Fla : Future Strategies Inc., 2008. 225 p.
24. Thalheim B. Entity-Relationship modeling. Berlin, Heidelberg : Springer Berlin Heidelberg, 2000. URL: https://doi.org/10.1007/978-3-662-04058-4 (date of access: 03.06.2023).

# ДОДАТКИ