ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА WEB ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ КРИПТОВАЛЮТ З ВИКОРИСТАННЯМ PYTHON, HTML, CSS**»

Виконав: студент 5 курсу, групи ППЗ–51

спеціальності

121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Патока В. В.

(прізвище та ініціали)

Керівник Аверічев І. М.

(прізвище та ініціали)

Рецензент Зінченко О. В.

(прізвище та ініціали)

Нормоконтроль Трінтіна Н. А.

(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти «Бакалавр»

Спеціальність підготовки - 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
О.В. Негоденко
«____» _____2023 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА
ПАТОКИ ВЛАДИСЛАВА ВОЛОДИМИРОВИЧА

(прізвище, ім‘я, по батькові)

1.Тема роботи: «Розробка web застосунку для аналізу криптовалют з використанням Python, HTML, CSS».

Керівник роботи: Аверічев І. М., к.е.н, доцент кафедри ІПЗ,

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

затверджені наказом вищого навчального закладу від «24» лютого 2023 року протокол №26.

2. Строк подання студентом роботи «01» червня 2023 року

3. Вхідні дані до роботи:

3.1. Середовище розробки PyCharm Community Edition 2023.1

3.2. Методи створення web застосунку

3.3. Binance API

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

4.1. Аналіз та порівняння існуючих аналогів

4.2. Дослідження програмних засобів для реалізації web застосунка

4.3. Програмна реалізація web застосунка

4.4. Приклади використання та тестування системи

4.5. Висновки

5. Перелік графічного матеріалу

5.1. Титульний слайд

5.2. Мета, об’єкт та предмет дослідження

5.3. Задачі дипломної роботи

5.4. Аналіз аналогів

5.5. Вимоги до додатку

5.6. Програмні засоби реалізації

5.7. Схеми роботи web застосунку

5.8. Екранні форми web застосунку

5.9. Висновки

6. Дата видачі завдання: «25» лютого 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	25.02 - 01.03	Виконано

2	Аналіз криптовалютного аналізу	02.03 - 05.03	Виконано
3	Дослідження архітектури web застосунків	06.03 - 16.03	Виконано
4	Розробка алгоритмів роботи web застосунку	17.03 - 25.03	Виконано
5	Реалізація програмного продукту	26.03 - 09.05	Виконано
6	Висновки, оформлення роботи	10.05 - 18.05	Виконано
7	Попередній захист роботи	19.05	Виконано
8	Подання роботи в деканат	01.06	Виконано

Студент _____ Патока В. В.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Аверічев І. М.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 57 сторінок, 53 рисунки, 13 джерел.

Ключові слова: Python, Django, PyCharm, web застосунок, криптовалюти, аналіз.

Об`єкт дослідження – процес аналізу криптовалют.

Предмет дослідження – програмне забезпечення для аналізу криптовалют.

Мета роботи – спрощення процесу аналізу криптовалют за рахунок використання web застосунку засобами Python, HTML, CSS.

Для реалізації поставленої мети потрібно вирішити наступні завдання:

1. Проаналізувати технічні засоби, що використовуються для розробки web застосунків та обрати найоптимальніший варіант який підійде для автоматизації процесу аналізу криптовалют.
2. Розробити вимоги до web застосунку на основі аналізу переваг та недоліків існуючих аналогів.
3. Спроектувати та розробити власний web застосунок на основі аналізу потреб користувачів.
4. Провести тестування роботоздатності та ефективності web застосунку.

Практичне значення отриманих результатів полягає у розробці web застосунку для аналізу криптовалют з використанням фреймворку Django та мови програмування Python.

В роботі розглянуто всі процеси створення web застосунку, досліджено можливості технічних засобів для реалізації такого типу застосунку.

Розроблено модель застосунку, функціональні вимоги до аналізу криптовалют та підібрані найефективніші методи реалізації цього процесу.

Галузь використання – web застосунок може використовувати будь-який користувач на просторі інтернету.

ЗМІСТ

Стор.

ВСТУП

Криптовалюта - це цифровий актив, що використовує криптографію для захисту відповідних транзакцій та контролю створення нових одиниць. Криптовалюта може бути використана як засіб обміну інформацією між двома сторонами без втручання посередників, таких як банки чи інші фінансові установи.

Базується на технології блокчейн, яка забезпечує безпечну та необоротну запис інформації про кожну транзакцію. Кожна транзакція відправляється до мережі, де вона підтверджується майнерами за допомогою різних математичних алгоритмів. Зберігається в електронних гаманцях, які забезпечують конфіденційність та безпеку власних даних та грошових коштів.

Найбільш відомою криптовалютою є Bitcoin, але на сьогодні на ринку існує більше 4 тисяч різних валют з різними функціями та метою використання. Криптовалюта може бути використана як інвестиційний інструмент, як засіб оплати за товари та послуги, а також як засіб переказу коштів без втручання посередників. Проте, використання криптовалюти пов'язане з певними ризиками та необхідністю розуміння основ криптографії та блокчейн технологій.

Популярність криптовалют різко зросла в останні роки, але їх недостатня стабільність ускладнюють аналіз. Вивчення та оцінка даних про криптовалютний ринок для виявлення тенденцій і потенційного прибутку активів - ось що означає аналіз криптовалют. Інвестори можуть використовувати його для прийняття розумних рішень і зниження ризиків. Особливістю аналізу криптовалюти є те, що він є високий волатильним і не піддається традиційним методам аналізу фінансового ринку. Крім того, інформація про ринок криптовалюти часто недоступна, що ускладнює аналіз. Щоб стати фахівцем в цьому потрібні величезні знання та досвід, щоб спростити цей процес, його можна оптимізувати та автоматизувати.

Таким чином, розробка web застосунків, для аналізу криптовалюти є **актуальною задачею** для фахівців у галузі сучасних інформаційних технологій.

Об’єкт дослідження - процес аналізу криптовалют.

Предмет дослідження - програмне забезпечення для аналізу криптовалют.

Мета роботи - спрощення процесу аналізу криптовалют за рахунок використання web застосунку засобами Python, HTML, CSS.

Для реалізації поставленої мети потрібно вирішити наступні завдання:

1. Проаналізувати теоретичні дослідження в сфері автоматизованих систем аналізу криптовалют.
2. Визначити переваги та можливості сучасних програмних засобів.
3. Дослідити алгоритми аналізу криптовалютного ринку та визначити найефективніший з них.
4. Спроектувати та розробити застосунок для автоматизації процесу аналізу криптовалютної пари.

Практичне значення отриманих результатів полягає у розробці web застосунку для аналізу криптовалют з використанням фреймворку Django та мови програмування Python.

В роботі розглянуто всі процеси створення web застосунку, досліджено можливості технічних засобів для реалізації такого типу застосунку.
Розроблено модель застосунку, функціональні вимоги до аналізу криптовалют та підібрані найефективніші методи реалізації цього процесу.

Галузь використання – web застосунок може використовувати будь-який користувач на просторі інтернету.

1 АНАЛІЗ ПРОЦЕСУ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КРИПТОВАЛЮТНОГО АНАЛІЗУ

1.1 Типи аналізу криптовалют та їх особливості

Криптовалютний ринок можна аналізувати за допомогою різних методів, таких як технічний аналіз, фундаментальний аналіз, соціальний аналіз та інші. Графіки руху цін, обсяги торгів та інші технічні індикатори аналізуються за допомогою технічного аналізу. Фінансові показники та фактори, що впливають на ціни криптовалюти, знаходяться в центрі фундаментального аналізу. Соціальна аналітика вивчає поведінку користувачів криптовалюти, їхні настрої та очікування. Цільова аудиторія продукту – інвестори та трейдери криптовалюти. Хочуть отримати глибше розуміння ринку та його потенційних прибутків, щоб приймати кращі інвестиційні рішення.

Вимоги до вирішення завдань аналізу криптовалют включають:

1. Надійність і точність: інвесторам і трейдерам потрібна точна та достовірна інформація про ринки криптовалют, щоб приймати зважені рішення.
2. Швидкість: ринки криптовалют можуть швидко змінюватися, тому швидке отримання та обробка інформації має першочергове значення.
3. Наявність інформації про ринки криптовалют може бути важко зібрати, тому вкрай важливо мати доступ до різних джерел і ресурсів, щоб отримати повну картину ринку.

Ринок криптовалют досить новий та мінливий, тому інформація про нього може бути нестабільною та суперечливою, криптоаналіз обмежений. Тому для отримання повної картини ринку, ефективним інструментом для інвесторів і трейдерів, для зменшення ризику, потрібно приймати більш обґрунтовані інвестиційні рішення.

Існує кілька основних способів аналізу криптовалюти, зокрема:

1. Технічний аналіз: цей підхід аналізує графіки цін і обсяги торгів, щоб визначити тенденції та можливі точки входу та виходу. Він використовує різні інструменти та індикатори, такі як ковзні середні, MACD, RSI та інші.
2. Фундаментальний аналіз: цей підхід базується на аналізі фундаментальних факторів, таких як фінансовий стан компанії, новини та події в галузі та економіці загалом. Чи є криптовалюта довгостроковою інвестицією.
3. Соціальний аналіз: цей підхід використовує дані із соціальних мереж та інших джерел для аналізу громадської думки та настроїв щодо певної криптовалюти. Це допомагає зрозуміти, які новини, анонси та інші події впливають на ціни криптовалюти.

Зараз, існують алгоритмічні торгові стратегії, засновані на штучному інтелекті та машинному навчанні. Для аналізу та прогнозування ціни криптовалюту використовують різноманітні алгоритми та інструменти. Найважливішим аспектом аналізу криптовалюти є розуміння ризиків і можливостей, пов’язаних з інвестиціями в криптовалюту. Інвестори та трейдери повинні бути готові до втрат та збалансувати свої ризики зі своїми цілями та стратегією.

Криптографічний аналіз вимагає:

1. Надійність даних: аналітики повинні мати доступ до точних і актуальних даних про ціни та обсяги торгів на ринку криптовалют.
2. Розуміння ринку: аналітики повинні розуміти особливості ринку криптовалют, такі як висока волатильність і швидкі зміни цін.
3. Компетентність: аналітики повинні мати досвід роботи з криптовалютами та розуміти різні методи та підходи до аналізу цін.
4. Швидкість і ефективність. Аналіз потрібно проводити швидко та ефективно, оскільки ринок криптовалют може змінюватися за лічені секунди.

Високий ризик і невизначеність на ринку криптовалют існує завдяки відсутності нормативної бази. Обмеженість ефективності технічного аналізу та обсягу даних присутня на нестабільних ринках.

Аналіз рівнів і щільностей, для торгівлі на валютному ринку, вважаються найоптимальнішими.

1.2 Аналіз щільності та рівнів спротиву і підтримки

Технічний аналіз «Торгівля за рівнями та щільністю» — являє собою один із підходів, який дозволяє визначати області підтримки та опору на цінових графіках, використовуючи їх для входу та виходу з ринку.

Ідея технічного аналізу полягає у визначенні рівнів, на яких ціна трималася протягом певного періоду часу, використовуючи їх для прогнозування подальшого руху ціни. Зони підтримки та опору, визначаються на основі історичних даних графіка цін.

Індикатори щільності, Volume Profile, вказують, в яких обсягах відбувається торгівля на різних цінових рівнях. Вони допомагають визначити області, де більшість учасників ринку мають відкриті позиції та рівні які вважаються значущими для ринку. Завдяки використання зон підтримки та опору, для визначення точок входу та виходу, трейдери зменшують свої ризики. Крім того, вони можуть використовувати індикатори щільності, щоб визначити потенційну силу руху ціни на ринку.

Однак, як і будь-який інший метод аналізу, технічний аналіз «Торгівля за рівнями та щільністю» може мати свої обмеження та ризики. Наприклад, врахування поточних ринкових умов, таких як новини чи геополітичні події, може мати вирішальне значення для належної оцінки ризиків і можливостей. Крім того, історичні дані на цінових графіках не завжди можуть бути індикатором майбутніх рухів цін, оскільки на ринку можуть статися непередбачувані події, які можуть вплинути на поведінку ринку.

Трейдери та інвестори, які зацікавлені в торгівлі на ринках криптовалют, можуть мати різний рівень досвіду та знань технічного аналізу, але всі вони можуть використовувати цей метод для покращення своїх торгових показників.

Основні вимоги до продукту, який використовує технічний аналіз «Торгівля за рівнями та щільністю», включають:

1. Наявність інструментів для визначення зон підтримки та опору на цінових графіках;
2. Можливість використання індикаторів щільності для визначення потенційної сили руху ціни на ринку;
3. Наявність інструментів аналізу цінових графіків, які допомагають трейдерам зрозуміти, як ціни змінюються з часом;
4. Можливість використовувати історичні дані для визначення рівня цін і тенденцій.

Обмеження цього методу аналізу можуть включати:

1. Неможливість з точністю передбачити майбутні зміни цін, оскільки на ринку можуть статися непередбачувані події, які можуть істотно вплинути на рух цін.
2. Можливість відсутності достовірного статистичного забезпечення деяких показників, що може призвести до невірної оцінки поточної ситуації на ринку.

Для ефективної оцінки ризиків та можливостей потрібно використовувати метод технічного аналізу.

1.3 Аналіз існуючого програмного забезпечення для криптовалютного аналізу

Онлайн-ресурси для аналізу криптовалют поділяються на наступні типи:

1. Ресурси для фундаментального аналізу криптовалют;
2. Ресурси для пошуку формацій на графіку валютної пари;
3. Ресурси з новинами пов'язані зі світом криптовалют;

4. Ресурси для збору даних та пошуку ущільнень в стакані.

1.3.1 Онлайн-ресурси для фундаментального аналізу

Онлайн-ресурси для фундаментального аналізу - один з найпопулярніших методів аналізу, за рахунок того що не потрібно знати багато специфічної термінології і мати досвіду торгівлі. Основний принцип це базове розуміння фінансового сегменту в світі.

Аналіз відбувається порівнянням популярності валюти, зміни її вартості за первний період, загальний обсягів грошей в валюті, ринкової капіталізація, циркуляційного запасу, детальну інформацію про кожну криптовалюту та її технологію.

Один із найвідоміших ресурсів для фундаментального аналізу це CoinMarketCap рисунок 1.3.1. Для більш детального аналізу, ви можете використовувати інші засоби, такі як порівняння криптовалют за різними показниками, детальну інформацію про кожну криптовалюту та її технологію, аналіз новин та інших факторів, які можуть впливати на ціну криптовалюти. Ви також можете переглядати інформацію про популярні криптобіржі, де торгуються криптовалюти, та обсяги торгів, що відбуваються на цих біржах.

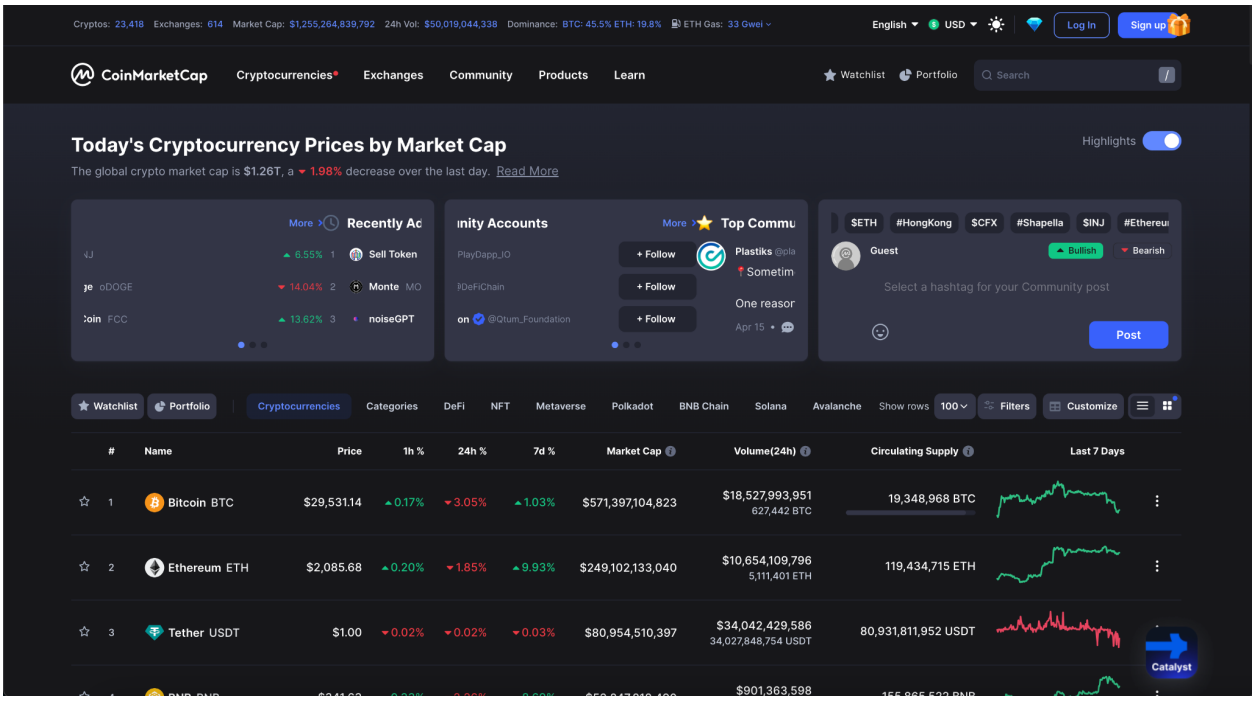


Рисунок 1.3.1 – Приклад екранної форми онлайн-ресурсу CoinMarketCap

Загалом, CoinMarketCap є потужним інструментом для фундаментального аналізу криптовалют та моніторингу їх ринку. Однак, слід пам'ятати, що аналіз ринку криптовалют може бути складним і вимагати багато досліджень та аналізу різних джерел інформації.

1.3.2 Онлайн-ресурси для пошуку формацій на графіку валютної пари

Онлайн-ресурси для пошуку формацій на графіку валютної пари такж являється одним із найпопулярніших видів аналізу. TradingView рисунок 1.3.2, найвідоміший ресурс, надає доступ до різноманітних графіків, які дозволяють відстежувати зміну ціни криптовалюти протягом різних періодів часу. Ви можете використовувати інструменти лінійного та свічкового графіків, щоб візуально відобразити та аналізувати цінову динаміку.

TradingView надає багато різноманітних інструментів для аналізу графіків на різних ринках, включаючи криптовалютний ринок. Ось кілька з них:

1. Таймфрейм - це період часу, за який відображається цінова інформація на графіку. Ресурс надає широкий вибір таймфреймів, від однієї хвилини до

місяця. Вибір таймфрейму залежить від вашого торгівельного стилю, стратегії та інтервалу часу, в якому ви плануєте здійснювати угоди.

2. Лінії підтримки та опору, індикатори технічного аналізу, свічкові графіки та інші. Використовуйте ці інструменти, щоб аналізувати цінову динаміку та відшукувати можливі можливості для угод.
3. Відслідковування тенденцій - це важлива частина аналізу графіків на TradingView. Ви можете використовувати інструменти, такі як лінії тенденції, щоб відслідковувати тенденції руху цін та розуміти, де можуть бути можливості для входу чи виходу з ринку.
4. Обсяг торгів - це кількість криптовалюти, яка торгується за певний період часу. Обсяг може допомогти вам зрозуміти, наскільки сильної є підтримка або опір на різних рівнях ціни, а також виявити можливість зміни тренду. Дозволяє аналізувати обсяги та відображати їх на графіку, що допоможе зрозуміти, що відбувається на ринку.
5. Індикатори технічного аналізу, які можуть допомогти вам зрозуміти цінову динаміку на ринку та знайти можливості для входу чи виходу з позиції. Наприклад, RSI, MACD, Bollinger Bands та інші. Вибір індикаторів залежить від вашого торгівельного стилю та стратегії.

Однак, важливо пам'ятати, що аналіз графіків - це складний процес, який вимагає багато часу, досвіду та розуміння ринку.

Рисунок 1.3.2 – Приклад екранної форми онлайн-ресурсу TradingView

1.3.3 Онлайн-ресурси з новинами пов'язані зі світом криптовалют

Одним з інформаційних онлайн-ресурсів пов'язаний зі світом криптовалют є український проект GagarinNews рисунок 1.3.3. Цей портал можна використовувати для отримання актуальної інформації про новини та події, що відбуваються в криптовалютному світі. Тут надається велика кількість інформації про криптовалюти, ринки та проекти, що допоможуть вам розуміти, що відбувається в цій галузі.

Ось кілька способів, які можна використовувати для аналізу криптовалюти на GagarinNews:

1. Огляд новин. Знайдете актуальні новини та статті про криптовалютні ринки та проекти. Відвідуючи портал регулярно, ви можете отримувати свіжу інформацію та розуміти, які події впливають на ціни на ринку.
2. Аналіз ринку. Предоставляє корисну інформацію про ціни на криптовалюти та інші показники ринку, такі як капіталізація, обсяги, графіки та інші. Використовуючи ці дані, ви можете зрозуміти, які криптовалюти можуть

бути вигідними для інвестування.

3. Аналіз проектів. Надає велику кількість інформації про різні проекти в галузі блокчейнів та криптовалют. Ви можете ознайомитися зі списком проектів, їхніми особливостями, командою розробників та іншими деталями. Це може допомогти вам визначити потенційно перспективні проекти для інвестування.
4. Аналіз експертів. Надає можливість читати інтерв'ю з експертами у галузі криптовалют та блокчейнів. Ці експерти можуть дати свої прогнози та рекомендації щодо інвестування в криптовалюти та проекти.
5. Аналіз соціальних мереж. Відстежує популярність криптовалют та блокчейн-проектів у соціальних мережах, таких як Twitter, Reddit та інші. Використовуючи ці дані, ви можете отримати уявлення про те, які криптовалюти є популярними серед користувачів та як це може вплинути на ціни на ринку.

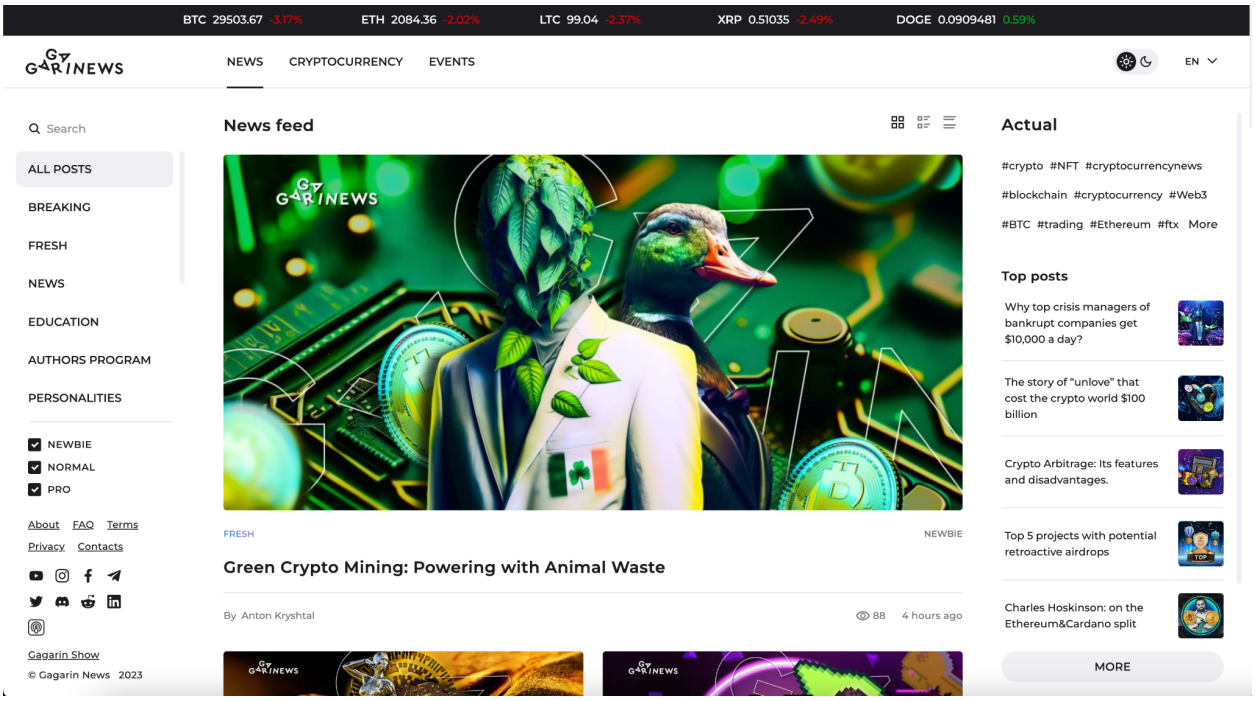


Рисунок 1.3.3 – Приклад екранної форми новинного порталу GagarinNews

Загалом, GagarinNews може бути корисним інструментом для отримання

інформації про криптовалюти та блокчейн-проекти, що може допомогти вам приймати обґрунтовані рішення щодо інвестування в цю галузь.

1.3.4 Онлайн-ресурси для збору даних та пошуку ущільнень в стакані

Winscreener рисунок 1.3.4, онлайн-ресурс для збору даних та пошуку ущільнень (лімітних заявок на купівлю та продаж).

Основні можливості ресурсу:

1. Аналіз графіків. Надає користувачам можливість переглядати графіки цін на різних криптовалютах. Користувачі можуть встановлювати фільтри за часом, відображати індикатори та іншу інформацію, щоб аналізувати графіки.
2. Скрінінг. Ресурс дозволяє користувачам знайти потенційно вигідні криптовалюти за різними параметрами, такими як ринкова капіталізація, ціна, обсяг торгів та інше.
3. Лімітні заявки. Дозволяє користувачам шукати лімітні заявки на різних криптовалютних біржах. Користувачі можуть встановлювати фільтри за ціною, кількістю токенів.

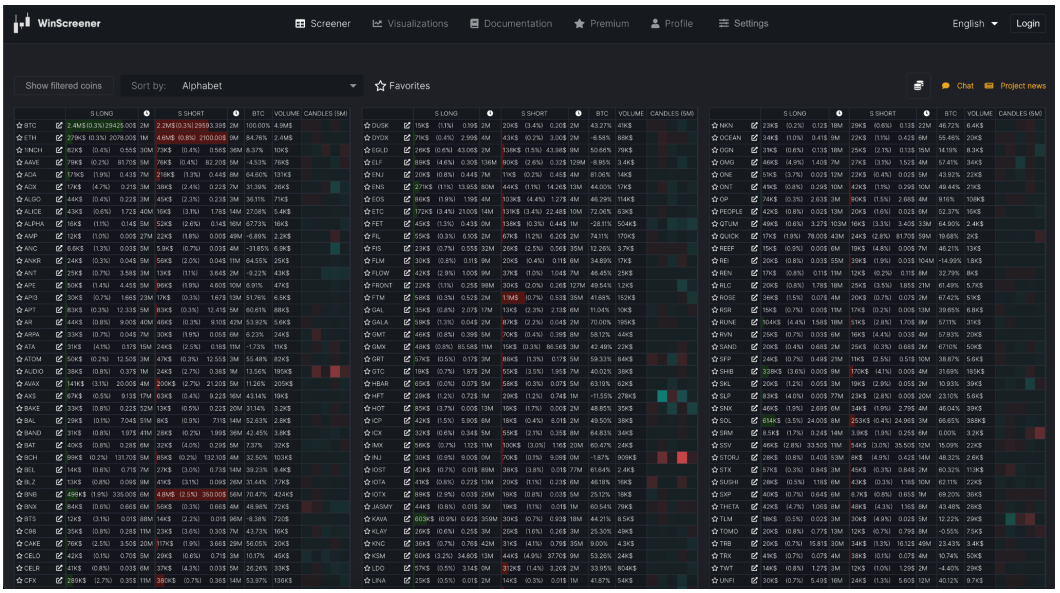


Рисунок 1.3.4 – Приклад екранної форми онлайн-ресурсу Winscreener

Загалом, Winscreener може бути корисним інструментом для трейдерів, які працюють з криптовалютними активами. Ресурс надає користувачам різноманітні можливості для аналізу ринку та пошуку вигідних можливостей.

1.4 Таблиця порівняння онлайн ресурсів

Таблиця 1.4 - таблиця порівняння аналогів

Показник	CoinMarketCap	TradingView	GagarinNews	WinScreener	Cevaluation
Зміна вартості	+	+	-	-	+
Таймфрейми	-	+	-	+	+
Лінії підтримки і опори	-	+	-	+	+
Тенденції	+	+	-	+	+
Лімітні заявки	-	-	-	+	+
Власний гаманець	-	-	-	-	+

2 ТЕХНОЛОГІЇ РЕАЛІЗАЦІЇ

2.1 Вимоги до веб застосунку

Особливості криптовалютного аналізу та існуючих засобів, дозволяє сформулювати вимоги до майбутнього програмного продукту. Застосунок для аналізу криптовалют потрібно реалізувати в вигляді веб-додатку та забезпечити наступний функціонал:

1. можливість відслідковувати зміни валютних пар;
2. можливість виводу графіків змін в різних періодах часу;
3. можливість пошуку ущільнень на цінах криптовалют;
4. засоби для аналізу криптовалютного гаманця користувача (додавання гаманця, видалення гаманця, відстеження змін в ньому);
5. можливість реєстрації користувача;
6. засоби для роботи з даними користувача (редагування псевдоніма користувача, редагування пароля від аккаунта, редагування API ключів, редагування підключеного телеграм акаунту);
7. можливість надавати щоденний звіт стосовно гаманця користувача в телеграмі;

Додаток для аналізу криптовалют розробляється в вигляді веб-застосунку. Вибір мови програмування для розробки залежить від специфіки криптовалютного ринку, а саме:

1. Швидкість обробки великої кількості інформації;
2. Можливістю обробляти велику кількість;
3. Можливість побудови інтерактивних графіків;
4. Обробки запитів через API;
5. Взаємодія з базою даних;

2.2 Python

В якості мови розробки вибрано Python. Він є однією з найпопулярніших мов програмування в світі, завдяки своїй простоті, елегантності та широкій спільноті розробників. Python можна використовувати для створення різноманітних веб-застосунків, включаючи веб-сайти, веб-сервери, веб-додатки та інші.

Для роботи з API використовується requests - це влаштована бібліотека в Python, яка дозволяє здійснювати HTTP-запити та отримувати відповіді від сервера. Завдяки простому та зрозумілому API, бібліотека requests дозволяє легко взаємодіяти з web-ресурсами, зокрема з API веб-сервісів та іншими web-додатками. За допомогою requests можна відправляти HTTP-запити, такі як GET, POST, PUT, DELETE та інші, а також виконувати запити з різними параметрами, такими як параметри запиту, заголовки, файли тощо.

2.3 Django

Одним з найбільш популярних фреймворків для створення веб-застосунків на Python являється Django. Він надає розробникам зручний інтерфейс для створення веб-додатків, забезпечуючи високу продуктивність, безпеку та масштабованість. Фреймворк розроблено з урахуванням концепцій Model-View-Controller (MVC) і містить вбудований ORM (Object-Relational Mapping), який дозволяє працювати з базами даних, такими як MySQL, PostgreSQL та SQLite, без прямої роботи з SQL.

Для розробки веб-додатків на Django використовуються широкі можливості вбудованого адміністративного інтерфейсу, який дозволяє швидко та легко зберігати, оновлювати та відображати дані. Django також надає широкі можливості для роботи з формами, авторизації користувачів, базами даних, включаючи підтримку різних СУБД

Фреймворк має відкритий вихідний код і має велику спільноту розробників, яка надає багато пакетів і додаткових функцій для розробки веб-додатків. Django також підтримує шаблони та мову розмітки HTML для створення динамічних сторінок. Django дозволяє легко створювати повнофункціональні веб-програми, які можна масштабувати та підтримувати протягом тривалого часу.

2.4 SQLite

SQLite - це компактна, вбудовувана, реляційна база даних, яка зберігається в одному файлі на диску і має певні переваги, такі як:

1. Дуже проста для використання. Ви можете створювати таблиці, вставляти дані, оновлювати та видаляти їх, виконувати запити та інше за допомогою звичайного SQL.
2. Працює дуже швидко, оскільки вона зберігається в одному файлі на диску.
3. Надійна. Так як зберігається в одному файлі, немає потреби у додаткових налаштуваннях чи утриманні сервера баз даних. Крім того, вона використовує механізм транзакцій для забезпечення цілісності даних.
4. Вільно розповсюджувана база даних та не потребує ніякої ліцензії або плати за використання.

2.5 Python-telegram-bot

Доставки важливих сповіщень, для зручності клієнта, використовують бібліотеку python-telegram-bot. Завдяки якій створюються телеграм-боти в мові програмування Python. Основна мета полягає в тому, щоб допомогти розробникам створювати швидкі та ефективні програмні рішення з використанням. Надає зручний інтерфейс для роботи з API Telegram, що дозволяє легко налаштовувати та розширювати функціональність бота.

2.6 HTML

Мова HTML (Hypertext Markup Language) є стандартною мовою розмітки для створення веб-сторінок. Теги та атрибути використовуються для структурування та форматування веб-сторінок. За допомогою HTML ви можете вставляти зображення, аудіо та відео, створювати гіперпосилання та форми введення. Теги HTML описують кожен елемент веб-сторінки, наприклад зображення, відео, форми тощо.

Атрибути в тегах HTML визначають додаткові відомості про елемент, такі як його розміри, колір, адреса посилання тощо. Веб-сторінку також можна стилізувати та взаємодіяти з нею за допомогою HTML, CSS і JavaScript.

2.7 Jinja2

Jinja2 шаблонизатор для мови програмування Python, який дозволяє вбудовувати в HTML-документи динамічний контент, який генерується за допомогою Python. Jinja2 підтримує вставку змінних, конструкції умов, циклів та інші елементи, які дозволяють генерувати різні типи контенту на стороні сервера. Загалом Jinja2 добре підходить для створення веб-додатків на Python, зокрема для побудови сайтів та веб-додатків з великою кількістю динамічного контенту. Використовуючи Jinja2, можна швидко та ефективно створювати шаблони.

2.8 CSS

Мова CSS (каскадні таблиці стилів) описує вигляд і форматування веб-сторінок HTML. За допомогою CSS ви можете відокремити структуру веб-сторінки від її візуального вигляду. Зменшує дублювання коду та спрощує процес зміни зовнішнього вигляду сторінки без зміни її структури. CSS складається з правил, які описують стиль веб-сторінки. Кожне правило

складається з селектора та блоку властивостей. Селектор вибирає, яка частина HTML-коду буде використана в стилі. Блок властивостей містить список параметрів, які виглядають як вибраний елемент. Також підтримує вкладення, що дозволяє визначати стиль для елементів всередині інших елементів. Крім того, CSS підтримує використання класів та ідентичностей, що дозволяє додатково диференціювати та застосовувати стилі до конкретних елементів.

CSS — потужний інструмент для оформлення веб-сторінок. Забезпечує зручність і читабельність, а також дозволяє розробникам привернути увагу до ключових частин веб-сторінок.

2.9 JavaScript

JavaScript — мова програмування, яка використовується для створення динамічних веб-сторінок і взаємодії з користувачем на сторінці. Дозволяє додавати різні функції до веб-сторінок, такі як анімація, зміна вмісту без перезавантаження сторінки, перевірка форм, перевірка даних і багато іншого. Він є однією з найпопулярніших мов програмування веб-додатків. Його можна використовувати як на стороні клієнта (у браузері), так і на стороні сервера (з Node.js). Існує велика кількість бібліотек і фреймворків JavaScript, які зменшують обсяг коду, який ви пишете, і спрощують розробку веб-додатків.

Мова програмування з динамічним типом даних, яка може працювати з об’єктами, масивами, рядками, числами, логічними значеннями та функціями. Іншими важливими особливостями мови є анонімні функції, обробка подій і закриття. JavaScript є дуже важливою мовою програмування для веб-розробки, яка дозволяє створювати динамічні та інтерактивні веб-сторінки.

2.10 PyCharm

PyCharm — інтегроване середовище розробки (IDE) для Python. Він розроблений JetBrains і містить різні інструменти для розробки, тестування та налагодження програм на Python.

PyCharm надає широкі можливості для розробки програм на Python, включаючи автоматичне визначення типу, рефакторинг, підсвічування синтаксису, автозаповнення коду, інтегрований налагоджувач, аналіз коду тощо. Він також має вбудовані інструменти для роботи з системами контролю версій, такими як Git, і дозволяє легко взаємодіяти з віртуальними середовищами та залежностями.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ WEB ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ КРИПТОВАЛЮТ

3.1 Налаштування програмного середовища та створення структури застосунку

Web застосунок для аналізу криптовалют, основний функціонал якого полягає в наданні користувачу інформацію про конкретну валютну пару, буде розроблений з використанням фреймворку Django.

Для початку створимо новий проект в нашій IDE (PyCharm), та сформуємо початковий каркас нашого застосунку на Django. Для цього в Django є початковий список команд для реалізації цього процесу, це називають шаблоном. Потрібно в терміналі нашого проекту написати команду «django-admin startproject crypto_analitic_site», де «crypto_analitic_site» - назва нашого проекту, і отримаємо базову архітектуру рисунок 3.1.1.

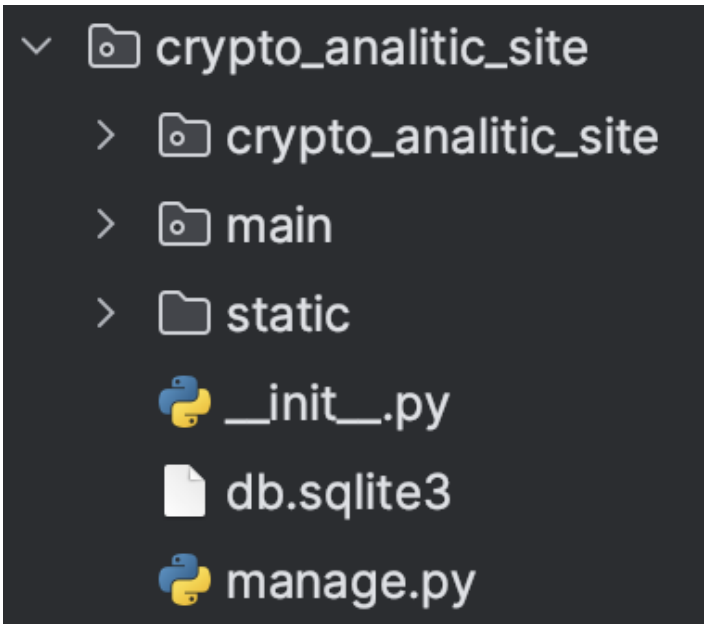


Рисунок 3.1.1 – базова архітектура проекту

Насамперед для реалізації застосунку потрібно створити логіку взаємодії користувача з сервісом та загальну структуру такого застосунку, які блоки можна в цьому процесі виділити та відокремити для більш доступної і швидкої системи для обробки інформації і взаємодією з нею, в потрібній нам формі, рисунок 3.1.2.

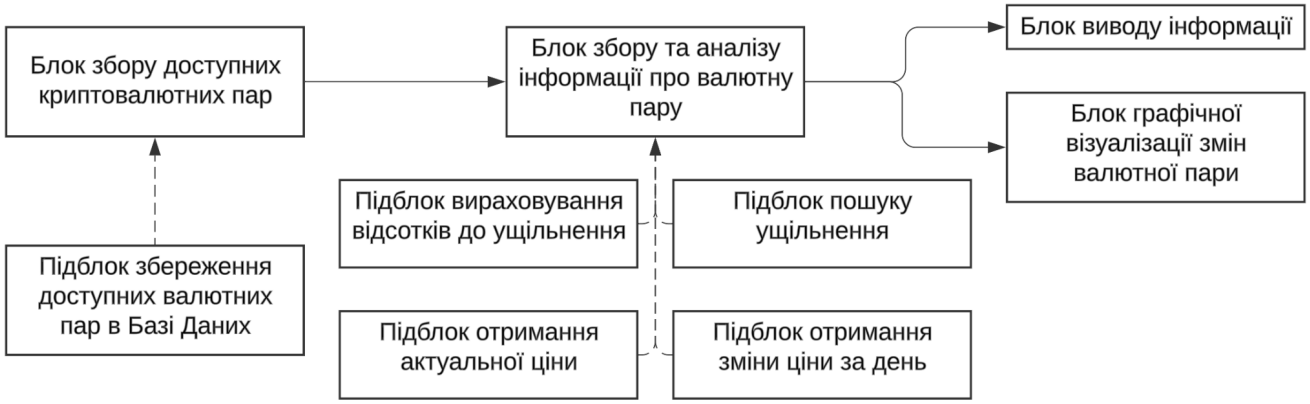


Рисунок 3.1.2 – схема структури системи

Також дуже важливим моментом є взаємодії різних видів користувачів з майбутньою системою та можливі варіанти доступу до певного функціоналу з умовами що для звичайного користувача не буде доступний весь можливий функціонал рисунок 3.1.3.

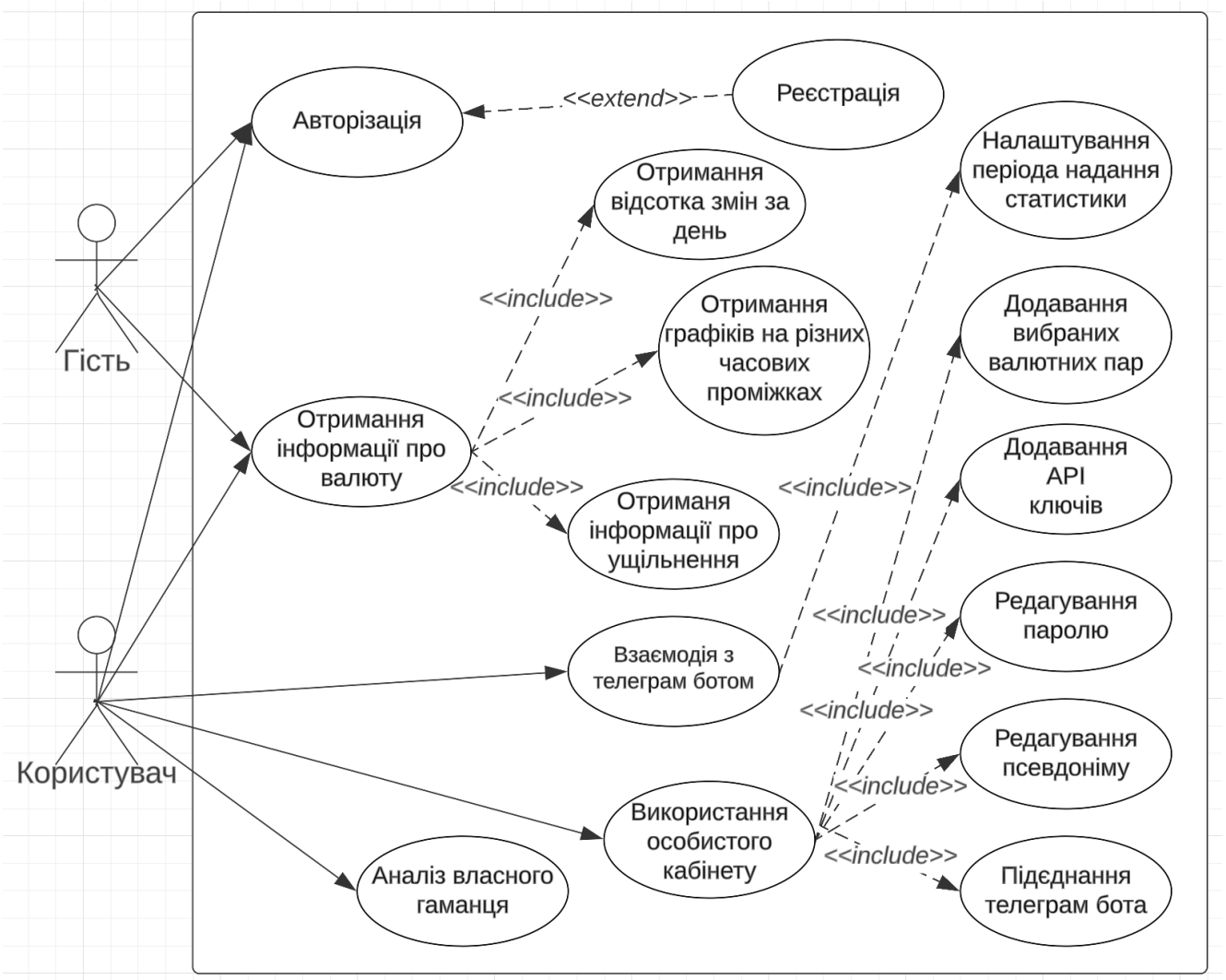


Рисунок 3.1.3 – діаграма прецедентів

3.2 Розробка скриптів роботи з криптовалютою

Наступним етапом розробки програмного забезпечення буде створення самого головного, а саме, скриптів для аналізу криптовалют, для цього в папці нашого проекту «main» створимо додаткову папку «scripts», в якій будуть знаходитися наші основні скрипти, поділивши їх на 3 файли:

1. «token_info.py» - скрипт при взаємодією з яким можливо отримувати всю потрібну нам інформацію про валюту.
2. «user_token_info.py»- скрипт при взаємодією з яким можливо отримувати всю потрібну нам інформацію про користувача та його активи.

3. «trading_pairs.py» - скрипт при взаємодією з яким можливо отримувати інформацію про валютні пари та їх зміни.

В кожному із цих скриптів створено головний клас через який відбувається взаємодія з його полями і методами в частинах програмного застосунку, де це нам потрібно. Створено діаграму цих класів для загального розуміння системи рисунок 3.2.1.

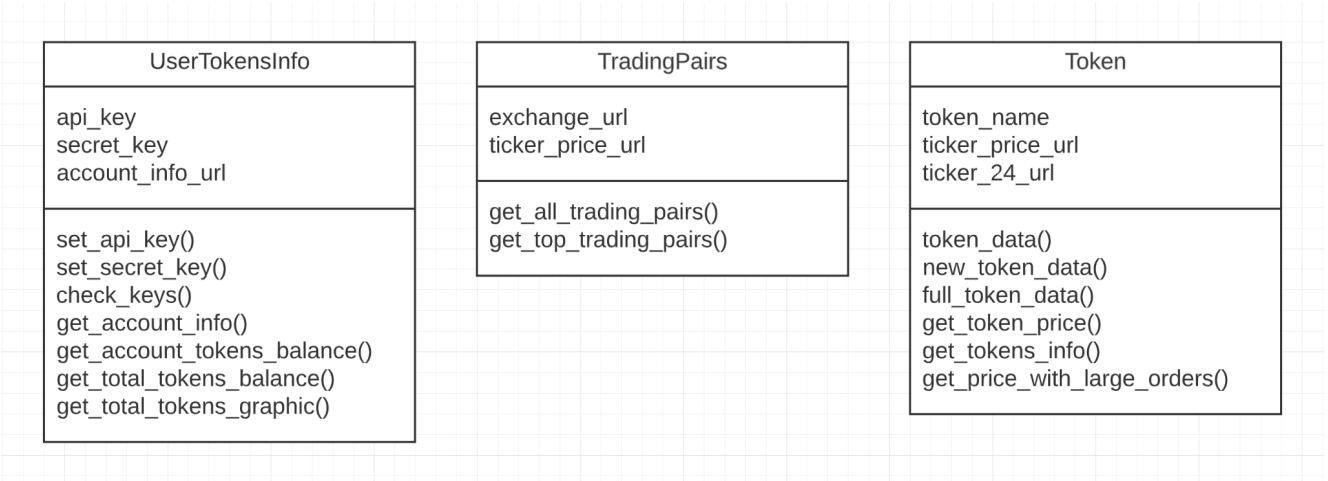


Рисунок 3.2.1 – діаграма класів

Вся логіка з отриманням інформації про криптовалюти завязана на Binance API, що є вільно доступним API, від найвідомішої криптовалютної біржі.

В скрипті «token_info.py» знаходиться клас «Token», що містить в собі назву токена що передається при ініціалізації об'єкта цього класу, посилання на API для отримання ціни та зміни валютної пари за день рисунок 3.2.2. Методи для отримання інформацію загальну про токен який ініціалізувати, токен який можна передати, повну інформацію про зміни за день і актуальну ціну токену рисунок 3.2.3. Також присутні більш розширені методи вже для фільтрації потрібної нам інформації такої як отримання словника де ключем являється валютна пара, а значенням її актуальна ціна рисунок 3.2.4. І метод в якому відбувається сам процес пошуку ущільнень на валютній парі за рахунок того що дістається в заданому ліміті з кожної позначки ціни кількість виставлених монет на продаж та купівлю,

після чого в циклі підраховується яка сума в доларах складають лімітні заявки на конкретній ціни, і якщо ця сума більше ста тисяч доларів то зупиняється цикл і повертає на якій ціні є таке ущільнення, після чого вираховується на скільки ціна має змінитися в відсотках щоб досягти цього ущільнення. І в результаті цей цикл повертає список значень в вигляді ціна на продаж, відсоток до ціни продажу, ціна на купівлю, відсоток до ціни на купівлю, рисунок 3.2.5.

```python
class Token:

    def __init__(self, token_name: str = None):
        self.token_name = token_name
        self.ticker_price_url = 'https://api.binance.com/api/v3/ticker/price'
        self.ticker_24_url = 'https://api.binance.com/api/v3/ticker/24hr'
```

Рисунок 3.2.2 – клас «Token» та його об'єкти

```python
1 usage
@property
def token_data(self):
    response = requests.get(self.ticker_price_url, params={'symbol': self.token_name})
    return response.json()

1 usage
def new_token_data(self, token):
    response = requests.get(self.ticker_24_url, params={'symbol': token})
    return response.json()

1 usage
@property
def full_token_data(self):
    response = requests.get(self.ticker_24_url)
    return response.json()

2 usages
@property
def get_token_price(self):
    return self.token_data['price']
```

Рисунок 3.2.3 – методи для отримання несортованої інформації, класу «Token»

```python
def get_tokens_info(self, tokens_list: list[str]):
    tokens_with_prices = {}
    data = self.full_token_data
    for token in tokens_list:
        try:
            token_data = list(filter(lambda x: x['symbol'] == token, data))[0]
            token_price = round(float(token_data['askPrice']), 2)
            token_changes = round(float(token_data['priceChangePercent']), 2)
            tokens_with_prices[token] = [token_price, token_changes]
        except IndexError:
            continue
    return tokens_with_prices
```

Рисунок 3.2.4 – методи для отримання валютної пари та її ціни, класу «Token»

```python
def get_price_with_large_orders(self, pair):
    url = f'https://api.binance.com/api/v3/depth?symbol={pair}&limit=10000'
    response = requests.get(url).json()

    sell = None
    buy = None
    sell_percent = None
    buy_percent = None
    token_current_price = float(self.get_token_price)
    token_data = self.new_token_data(pair)
    token_changes_for_day = round(float(token_data['priceChangePercent']), 2)

    for bid in response['bids']:
        price, quantity = bid
        if float(quantity) * float(price) > 1000000:
            sell = round(float(price), 2)
            break

    for ask in response['asks']:
        price, quantity = ask
        if float(quantity) * float(price) > 1000000:
            buy = round(float(price), 2)
            break

    if buy:
        buy_percent = round(((buy - token_current_price) / token_current_price) * 100, 2)
    if sell:
        sell_percent = round(((token_current_price - sell) / token_current_price) * 100, 2)
    return [sell, sell_percent, buy, buy_percent, token_changes_for_day]
```

Рисунок 3.2.5 – методи для отримання несортованої інформації, класу «Token»

В скрипті «user_token_info.py» знаходиться клас «UserTokensInfo», що містить в собі API ключ, секретний ключ і посилання на API для отримання інформації про акаунт користувача на біржі рисунок 3.2.6. Реалізовані сетери для об'єктів цього класу рисунок 3.2.7. Метод для перевірки валідності ключа та секретного ключа, цей метод створений для перевірки валідності цих даних при спробі користувача підключити свій акаунт з біржі до сайту рисунок 3.2.8. Метод для отримання списку всіх валют що має користувач в своєму гаманці на біржі, та їх актуальної вартості і допоміжний метод до цього що додатково підсумовує ці значення та отримує повну вартість гаманця рисунок 3.2.9. Також метод що створює графік змін гаманця користувача, який приймає дату та суму гаманця, будує графік та повертає його в бітовому вигляді рисунок 3.2.10.

```
class UserTokensInfo:

    def __init__(self):
        self.api_key = None
        self.secret_key = None
        self.account_info_url = 'https://api.binance.com/api/v3/account'
```

Рисунок 3.2.6 – клас «UserTokensInfo» та його об'єкти

```
3 usages
def set_api_key(self, api_key):
    self.api_key = api_key
    return self

3 usages (2 dynamic)
def set_secret_key(self, secret_key):
    self.secret_key = secret_key
    return self
```

Рисунок 3.2.7 – сетери класу «UserTokensInfo»

```python
def check_keys(self, api_key, secret_key):
    self.set_api_key(api_key).set_secret_key(secret_key)

    params = {'timestamp': str(int(time.time() * 1000))}
    signature = hmac.new(
        self.secret_key.encode('utf-8'), urlencode(params).encode('utf-8'), hashlib.sha256
    ).hexdigest()
    params['signature'] = signature
    headers = {'X-MBX-APIKEY': self.api_key}
    response = requests.get(self.account_info_url, headers=headers, params=params)
    if str(response) == '<Response [200]>':
        return True
    else:
        return False
```

Рисунок 3.2.8 – метод для валідації ключів, класу «UserTokensInfo»

```python
2 usages (2 dynamic)
@property
def get_account_tokens_balance(self):
    account_balances = self.get_account_info['balances']
    balances = {}

    for balance in account_balances:
        if float(balance['free']) > 0 or float(balance['locked']) > 0:
            token_name = balance['asset'] + 'USDT'
            try:
                token_price = Token(token_name).get_token_price
            except KeyError:
                continue
            try:
                balance_value = (float(balance['free']) + float(balance['locked'])) * float(token_price)
                balances[balance['asset']] = round(balance_value, 2)
            except KeyError:
                continue
    return balances

2 usages (2 dynamic)
@staticmethod
def get_total_tokens_balance(token_balances):
    total = 0
    for token, value in token_balances.items():
        total += value
    return total
```

Рисунок 3.2.9 – методи для отримання та підрахунку гаманця користувача, класу «UserTokensInfo»

```
2 usages (2 dynamic)
@staticmethod
def get_total_tokens_graphic(totals):
    prices = []
    dates = []
    for col in totals:
        prices.append(col['total'])
        dates.append(str(col['date'])[5:])

    fig, ax = plt.subplots()
    ax.plot(dates, prices)
    ax.tick_params(labelsize=8)
    plt.xticks(rotation=60)

    buf = BytesIO()
    fig.savefig(buf, format='png')
    buf.seek(0)

    string = base64.b64encode(buf.read())
    graphic = '' + urllib.parse.quote(string)
    return graphic
```

Рисунок 3.2.10 – метод для отримання графіку змін, класу «UserTokensInfo»

В скрипті «trading_pairs.py» знаходиться клас «TradingPairs», що містить в собі посилання на API для отримання назв валютних пар що зараз торгуються на біржі та посилання що повертає зміни валютної пари за день рисунок 3.2.11. Метод, що повертає список всіх валютних пар до долара рисунок 3.2.12. Та метод що повертає топ пять валютних пар що більше всього змінились в вартості за сьогодні рисунок 3.2.13.

```
class TradingPairs:

    def __int__(self):
        self.exchange_url = 'https://fapi.binance.com/fapi/v1/exchangeInfo'
        self.ticker_price_url = 'https://api.binance.com/api/v3/ticker/24hr'
```

Рисунок 3.2.11 – клас «TradingPairs» та його об'єкти

```python
@property
def get_all_trading_pairs(self):
    response = requests.get(self.exchange_url)
    data = response.json()
    trading_pairs = []

    for symbol in data['symbols']:
        if (
            symbol['contractType'] == 'PERPETUAL' or
            symbol['contractType'] == 'CURRENT_QUARTER' or
            symbol['contractType'] == 'NEXT_QUARTER'
        ):
            if symbol['symbol'][-4:] == 'USDT':
                trading_pairs.append(symbol['symbol'])
    return trading_pairs
```

Рисунок 3.2.12 – метод для отримання валютних пар, класу «TradingPairs»

```python
@property
def get_top_trading_pairs(self):
    tickers = []
    for pair in self.get_all_trading_pairs:
        params = {'symbol': pair}
        response = requests.get(self.ticker_price_url, params=params)
        ticker = response.json()
        try:
            ticker['priceChangePercent']
        except KeyError:
            continue
        tickers.append(ticker)

    top_pairs = sorted(tickers, key=lambda ticker: float(ticker['priceChangePercent']), reverse=True)[:5]
    return [(pair['symbol'], pair['priceChangePercent']) for pair in top_pairs]
```

Рисунок 3.2.13 – метод для отримання топ валютних пар, класу «TradingPairs»

3.3 Створення бази даних

В Django створена досить зручна взаємодія з Базою Даних. Створення таблиць відбувається звичайним створенням класів з назвою таблиці та об'єкти цього класу стають полями в таблиці. Для коректної роботи цього проекту була розроблена схема БД рисунок 3.3.1. Яка вміщує в собі чотири таблиці:

1. «User» - таблиця що містить в собі дані про користувача рисунок 3.3.2:
   - 1.1. «id» - унікальний ідентифікатор об'єкта таблиці.
   - 1.2. «email» - електронна пошта користувача зареєстрованого в системі.
   - 1.3. «password» - пароль від аккаунта користувача.
   - 1.4. «nickname» - ім'я або унікальний псевдонім користувача в системі.
   - 1.5. «api_key» - унікальний API ключ користувача від біржі Binance.
   - 1.6. «secret_key» - ключ аутентифікації для API.
   - 1.7. «last_login» - дата та час останнього візиту на web застосунок.
2. «FavoriteTokens» - таблиця що містить улюблені валютні пари для кожного користувача рисунок 3.3.3:
   - 2.1. «id» - унікальний ідентифікатор об'єкта таблиці.
   - 2.2. «favorite_token» - назва валютної пари що є улюбленою для заданого користувача.
   - 2.3. «user» - посилання на користувача з таблиці користувачів.
3. «Total» - таблиця що містить загальну суму криптовалютного гаманця користувача рисунок 3.3.4:
   - 3.1. «id» - унікальний ідентифікатор об'єкта таблиці.
   - 3.2. «date» - дата запису в таблицю.
   - 3.3. «total» - сума криптовалютного гаманця.
   - 3.4. «user» - посилання на користувача з таблиці користувачів.
4. «TokenName» - таблиця що містить назви криптовалютних пар що торгуються на біржі 3.3.5:
   - 4.1. «id» - унікальний ідентифікатор об'єкта таблиці.

4.2. «token_name» - унікальна назва валютної пари.

Для роботи з Базою такого вигляду потрібно зробити міграції. Міграції - це файли Python, які містять код для створення, зміни або видалення таблиць та полів бази даних. Саме вони конвертують наші класи в звичайний вигляд Бази даних. Для цього потрібно виконати два кроки, створити файл міграції і мігрувати цей файл, для цього в Django є спеціальні команди які потрібно прописати в терміналі:

1. «python manage.py makemigrations» - створення файлу міграції.
2. «python manage.py migrate» - міграція в БД.

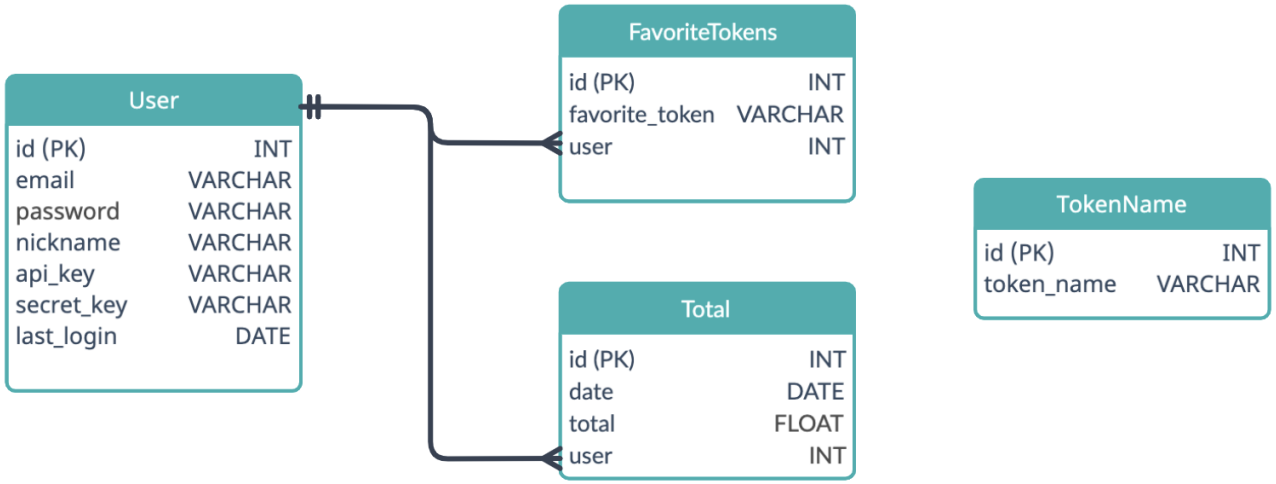


Рисунок 3.3.1 – схема бази даних

```python
class User(models.Model):
    email = models.EmailField('User Email')
    password = models.CharField('Password', max_length=100)
    nickname = models.CharField('Nickname', max_length=50)
    api_key = models.CharField('API Key', max_length=100)
    secret_key = models.CharField('Secret Key', max_length=100)
    tg_account = models.CharField('Telegram Account', max_length=50, null=True, blank=True)
    last_login = models.DateTimeField(null=True, blank=True)

    class Meta:
        app_label = 'main'

    1 usage (1 dynamic)
    def check_password(self, raw_password):
        print(raw_password, self.password)

        return raw_password == self.password
```

Рисунок 3.3.2 – «User» таблиця в Базі Даних

```python
2 usages
class FavoriteTokens(models.Model):
    favorite_token = models.CharField('Favorite Token', max_length=20)
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    class Meta:
        app_label = 'main'
```

Рисунок 3.3.3 – «FavoriteTokens» таблиця в Базі Даних

```python
8 usages
class Total(models.Model):
    date = models.DateField('Date', auto_now_add=True)
    total = models.DecimalField('Total', max_digits=10, decimal_places=2)
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    class Meta:
        app_label = 'main'
```

Рисунок 3.3.4 – «Total» таблиця в Базі Даних

```
8 usages
class TokenName(models.Model):
    token_name = models.CharField('Token Name', max_length=20)

    class Meta:
        app_label = 'main'
```

Рисунок 3.3.5 – «TokenName» таблиця в Базі Даних

3.4 Розробка форм заповнення для web застосунку

В Django форми створюються в окремому файлі під назвою «forms.py», кожна форма описується як окремий клас в якому можливо задавати модель БД, до якої ця форма привязана і які саме поля для заповнення вона в собі містить. Також в середині класа можна відразу прописувати додаткові віджети. Віджети в формах - це вставки що містять в собі стиль кожного поля що буде створений в формі та їх унікальні властивості. Для проекту було створено шість форма, а саме:

1. «LoginForm» - форма що в має поля електронної пошти та паролю від аккаунта, рисунок 3.4.1. Створена для входження в систему зареєстрованого користувача.
2. «RegistrationForm» - форма що має поля нікнейму, електронної пошти, паролю та повторення паролю від аккаунта, рисунок 3.4.2. Створена для реєстрації нового користувача в системі.
3. «EditUserPasswordForm» - форма що має поля старого паролю, нового паролю, та повторення нового паролю, рисунок 3.4.3. Створена для того щоб редагувати пароль від аккаунта користувача.
4. «EditUserNicknameForm» - форма що має поле нікнейму, рисунок 3.4.4. Створена для редагування нікнейму користувача.

5. «EditUserTGForm» - форма що має поле телеграм акааунту, рисунок 3.4.5. Створена для додавання чи редагування телеграм аккаунта користувача на який будуть приходити сповіщення.
6. «APIKeysForm» - форма що має поля API ключа, секретного ключа і пароля від аккаунта, рисунок 3.4.6. Створена форма для того щоб можна було додати свої ключі або замінити їх за необхідністю.

Рисунок 3.4.1 – клас форми «LoginForm»

Рисунок 3.4.2 – клас форми «RegistrationForm»

```
3 usages
class EditUserPasswordForm(ModelForm):
    old_password = CharField(widget=PasswordInput(attrs={
        'class': 'form-control mb-3',
        'placeholder': 'Account Password'
    }))
    repeat_password = CharField(widget=PasswordInput(attrs={
        'class': 'form-control mb-3',
        'placeholder': 'Repeat New Password'
    }))

    class Meta:
        model = User
        fields = ['password']
        widgets = {
            'password': PasswordInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'New Password'
            }),
        }
```

Рисунок 3.4.3 – клас форми «EditUserPasswordForm»

```
class EditUserNicknameForm(ModelForm):
    class Meta:
        model = User
        fields = ['nickname']
        widgets = {
            'nickname': TextInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'New Nickname'
            })
        }
```

Рисунок 3.4.4 – клас форми «EditUserNicknameForm»

```python
class EditUserTGForm(ModelForm):
    class Meta:
        model = User
        fields = ['tg_account']
        widgets = {
            'tg_account': TextInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'my_telegram without @'
            })
        }
```

Рисунок 3.4.5 – клас форми «EditUserTGForm»

```python
3 usages
class APIKeysForm(ModelForm):
    password = CharField(widget=PasswordInput(attrs={
        'class': 'form-control mb-3',
        'placeholder': 'Account Password'
    }))

    class Meta:
        model = User
        fields = ['api_key', 'secret_key']
        widgets = {
            'api_key': TextInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'Binance API Key'
            }),
            'secret_key': PasswordInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'Binance Secret Key'
            }),
        }
```

Рисунок 3.4.6 – клас форми «APIKeysForm»

3.5 Реалізація шаблонів web сторінок

Для створення HTML документів та роботи з ними в Django потрібно в середині створеного проекта створити папку «templates», фреймворк сам розуміє що там лежать саме файли для генерації web сторінки. Всередині цієї папки створимо ще одну задовільну папку, назвемо її «main», після цього є можливість

створювати шаблони, вийшло шість сторінок сайту, тобто шість шаблонів, рисунок 3.5.1.

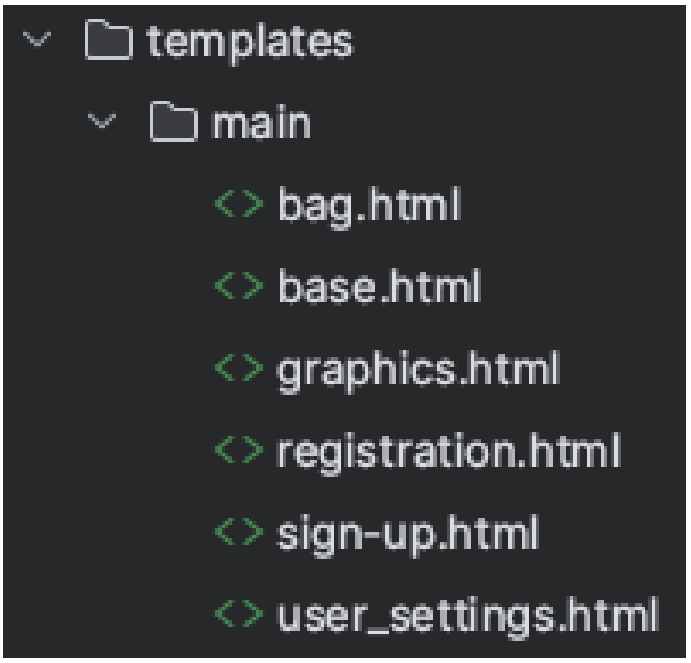


Рисунок 3.5.1 – папка із HTML шаблонами

Є головний шаблон від якого наслідуються всі інші, і цей шаблон має назву «base.html» всередині кожного із шаблонів використовується шаблонизатор Jinja2, завдяки якому можна робити багато зручних використання одного і того самого блоку коду. Весь базовий повторюваний код, такий як навігаціона полоска зверху, лежить в «base.html» і відокремлений блок контенту що змінюється на кожній сторінці позначається з використанням Jinja2 кодом, рисунок 3.5.2. Також за допомогою нього можна використовувати умови, в випадку навігаційної полоски перевіряємо чи користувач що потрапив на сторінку починав сесію з сервером, якщо так, то демонструємо нікнейм, і натиснувши на нього, користувач може переміститися на сторінку редагування профілю, рисунок 3.5.3. Також цикли напряму в HTML документі, при виклику генерації сторінки можемо передавати список або словник на сторінку, і на сторінці вже маніпулювати цими даними, нариклад на головній сторінці виводимо багато однакових елементів, в яких

змінюється тільки контент, а сам елемент має одні і ті самі характеристики, щоб зменшити код і постійно не повторювати використовуються цикл, який проходить по кожному елементу зі словника що передається і створює об'єкти на сторінці, рисунок 3.5.4. Ще додано додатковий блок, що відповідає за назву сторінки, який так само в залежності від сторінки динамічно змінює назву, рисунок 3.5.5.

Рисунок 3.5.2 – блок контента в файлі «base.html»

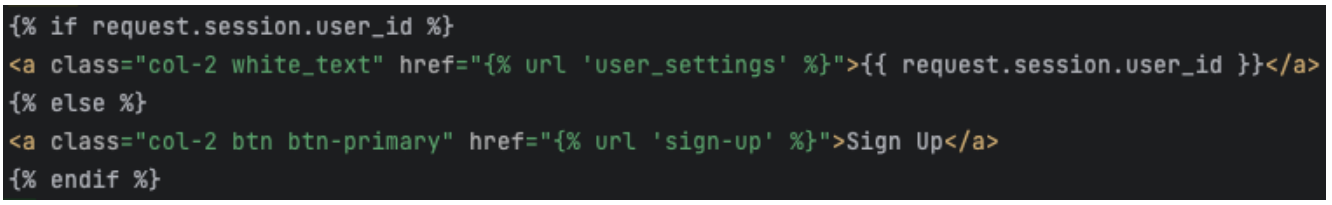


Рисунок 3.5.3 – використання умов з бібліотеки Jinja2

Рисунок 3.5.4 – використання циклів з бібліотеки Jinja2

Рисунок 3.5.5 – блок назви сторінки в файлі «base.html»

Всі наступні шаблони ініціалізуються в середині документу, рисунок 3.5.6, головний шаблон на який посилаються, і все що залишається, лише передавати нові блоки з контентом, назвою сторінки що відповідають за це.

```
{% extends 'main/base.html' %}
```

Рисунок 3.5.6 – наслідування з головного файлу «base.html»

Сторінка входу в системі містить в собі блок контенту що складається зі форми яка була створена в минулому пункті, і додаткові умови, якщо сторінка прийме змінну помилки, створить елемент і попередить користувача, що ввів не валідні дані, також виставляємо поля форми що передаємо, де це потрібно, рисунок 3.5.7.

```
{% block content %}
<body class="text-center bg-dark">
    <form class="form-signin mx-auto col-5" method="post">
        <div class="form-group">
            <h1 class="h3 mb-3 font-weight-normal white_text">Please sign in</h1>
             {% csrf_token %}
             {% if error %}
                 <div class="alert alert-danger">
                     {{ error }}
                 </div>
             {% endif %}
             {{ loginform.email }}
             {{ loginform.password }}
            <button class="btn btn-lg btn-primary btn-block" type="submit">Login</button>
            <a href="{% url 'registration' %}" class="btn btn-lg btn-primary btn-block" type="submit">Registration</a>
        </div>
    </form>
</body>
{% endblock %}
```

Рисунок 3.5.7 – сторінка входу в аккаунт «sign-up.html»

Сторінка реєстрації має схожу структуру як і сторінка логіна, але містить в собі форму реєстрації та всі поля що відносяться до неї, також умава на перевірку наявності помилок, рисунок 3.5.8.

```
{% block content %}
<body class="text-center bg-dark">
    <form class="form-signin mx-auto col-5" method="post">
        <div class="form-group">
            <h1 class="h3 mb-3 font-weight-normal white_text">Registration</h1>
            {% csrf_token %}
            {% if error %}
                <div class="alert alert-danger">
                    {{ error }}
                </div>
            {% endif %}
            {{ regform.nickname }}
            {{ regform.email }}
            {{ regform.password }}
            {{ regform.repeat_password }}
            <button class="btn btn-lg btn-primary btn-block" type="submit">Registrate</button>
        </div>
    </form>
</body>
{% endblock %}
```

Рисунок 3.5.8 – сторінка реєстрації нового аккаунту «registration.html»

Сторінка налаштування профілю користувача екранізує форму редагування нікнейма користувача, рисунок 3.5.9, форму редагування телеграм акаунту, рисунок 3.5.10, форму редагування пароля, рисунок 3.5.11 та форму редагування API ключів, рисунок 3.5.12. Також від кожної з цих форм виводиться помилка якщо щось пішло не так, і повідомлення про успішні зміни.

```
<form class="form-signin mx-auto col-5" method="post">
    <div class="form-group">
        <h1 class="h3 mb-3 font-weight-normal white_text">Edit nickname</h1>
        {% csrf_token %}
        {% if nickname_error %}
            <div class="alert alert-danger">
                {{ nickname_error }}
            </div>
        {% endif %}
        {{ edit_nickname_form.nickname }}
        <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
    </div>
</form>
```

Рисунок 3.5.9 – форма зміни нікнейму в «user_settings.html»

```
<form class="form-signin mx-auto col-5" method="post">
    <div class="form-group">
       <h1 class="h3 mb-3 font-weight-normal white_text">Edit telegram account</h1>
        {% csrf_token %}
        {% if tg_account_error %}
            <div class="alert alert-danger">
                {{ tg_account_error }}
            </div>
        {% endif %}
        {{ edit_tg_account_form.tg_account }}
       <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
   </div>
</form>
```

Рисунок 3.5.10 – форма зміни телеграмм аккаунту в «user_settings.html»

```
<form class="form-signin mx-auto col-5" method="post">
    <div class="form-group">
       <h1 class="h3 mb-3 font-weight-normal white_text">Edit password</h1>
        {% csrf_token %}
        {% if password_error %}
            <div class="alert alert-danger">
                {{ password_error }}
            </div>
        {% endif %}
        {{ edit_password_form.old_password }}
        {{ edit_password_form.password }}
        {{ edit_password_form.repeat_password }}
       <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
   </div>
</form>
```

Рисунок 3.5.11 – форма зміни пароля в «user_settings.html»

```
<form class="form-signin mx-auto col-5" method="post">
    <div class="form-group">
       <h1 class="h3 mb-3 font-weight-normal white_text">Edit Binance API keys</h1>
        {% csrf_token %}
        {% if key_error %}
            <div class="alert alert-danger">
                {{ key_error }}
            </div>
        {% endif %}
        {{ keys_form.api_key }}
        {{ keys_form.secret_key }}
        {{ keys_form.password }}
       <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
   </div>
</form>
```

Рисунок 3.5.12 – форма зміни API ключів в «user_settings.html»

Сторінка детальної інформації про токен та аналітичні дані стосовно нього, на цю сторінку можна потрапити тільки з головної сторінки при натисканні на якийсь з валютних пар на ній. Ця сторінка виводить інформацію про зміни валютної пари, стоїть умова якщо знайшлося ущільнення, то його виведе на екран, рисунок 3.5.13. Також нижче розташований цикл що за допомогою JavaScript виводить чортири графіка змін валюти за певний проміжок часу, щоб не повторювати код чотири рази, він теж винесений в цикл, рисунок 3.5.14.

Сторінка аналізу криптовалютного гаманця користувача містить в собі дані по результатам аналізу гаманця на біржі. Виводиться побудований графік по результатам змін, це фото документ що передається в бітовому вигляді і вимальовується відразу на сторінці. Також виводиться загальна вартість гаманця користувача і в циклі створюються елементи які в собі містять валютні пари в цьому гаманці і їх вартість на поточний момент, рисунок 3.5.15

```
<h4 class="white_text" style="text-align: center;">Graphics {{ token }} in different timeframes </h4>
<h5 class="white_text" style="margin-bottom: 20px;">Change price today: {{ token_changes.4 }}%</h5>
{% if token_changes.0 %}
    <h5 class="white_text">Support level on the price: {{ token_changes.0 }}$</h5>
    <h5 class="white_text" style="margin-bottom: 20px;">To support level: {{ token_changes.1 }}%</h5>
{% endif %}
{% if token_changes.2 %}
    <h5 class="white_text">Resistance level on the price: {{ token_changes.2 }}$</h5>
    <h5 class="white_text">To resistance level: {{ token_changes.3 }}%</h5>
{% endif %}
```

Рисунок 3.5.13 – умови відображення ущільнень в «graphics.html»

```
{% for name, val in time_frame.items %}
<div class="col-6">
    <h5 class="white_text mt-4" style="...">Timeframe: {{ name }}</h5>
    <div class="tradingview-widget-container">
        <div id="tradingview_{{ name }}"></div>
        <script type="text/javascript">
            var container = document.getElementById("tradingview_{{ name }}");
            container.style.height = (window.innerHeight * 0.4) + "px";

            window.addEventListener('resize', function(event){
                container.style.height = (window.innerHeight * 0.5) + "px";
            });
        </script>
        <script type="text/javascript" src="https://s3.tradingview.com/tv.js"></script>
        <script type="text/javascript">
            new TradingView.widget({
                  "autosize": true,
                  "symbol": "BINANCE:{{ token }}",
                  "interval": "{{ val }}",
                  "timezone": "Etc/UTC",
                  "theme": "dark",
                  "style": "1",
                  "locale": "en",
                  "toolbar_bg": "#f1f3f6",
                  "enable_publishing": true,
                  "hide_top_toolbar": true,
                  "hide_legend": true,
                  "save_image": false,
                  "container_id": "tradingview_{{ name }}"
                });
        </script>
    </div>
</div>
{% endfor %}
```

Рисунок 3.5.14 – цикл відображення графіків в «graphics.html»

```
{% block content %}
<body class="bg-dark">
    <div class="container">
        <h5 class="white_text" style="...">Bag Changes</h5>
        <div style="text-align:center;">
            <img class="mb-4" src="{{ graphic }}">
        </div>
        <h5 class="white_text mb-2" style="...">Total: {{ total }}$</h5>
        <h5 class="white_text" style="...">Tokens in Binance spot bag:</h5>
        {% for token, amount in tokens_with_amounts.items %}
            <h6 class="white_text" style="...">{{ token }} = {{ amount }}$</h6>
        {% endfor %}
    </div>
</body>
{% endblock %}
```

Рисунок 3.5.15 – вигляд шаблону «bag.html»

3.6 Розробка головних функцій системи

Фінальним етапом реалізації веб застосунку є збір всього розробленого в повноцінну працюючу систему, саме для цього в фреймворку Django інує файл «views.py». В цьому файлі створюється обробник запитів, приписується привязку конкретного інтернет адресу до конкретного шаблону і додається обробка та передача потрібної інформації.

Таким чином функція, що відповідає за генерацію головної сторінки, всередині себе викликає один раз на день функцію зі скриптів, що повертає список всіх доступних валютних пар на біржі. Все записується в БД, звідки дістається ці значення, до кожної валютної пари використовується ще одна функція, яка отримує до кожної валютної пари, її вартість та відсоток змін за день, після чого повертається шаблон що пов'язаний з цією сторінкою і отримані дані, рисунок 3.6.1.

```
def base(request):
    now = timezone.localtime()
    current_time = now.time()
    target_time = datetime.time(1, 0, 0)
    if current_time < target_time or TokenName.objects.count() == 0:
        if TokenName.objects.count() != 0:
            TokenName.objects.all().delete()
        for token in TradingPairs().get_all_trading_pairs:
            TokenName.objects.create(token_name=token)

    tokens_name = TokenName.objects.all().values_list('token_name', flat=True)
    tokens_with_prices = Token().get_tokens_info(tokens_name)
    return render(request, 'main/base.html', {'tokens_with_prices': tokens_with_prices})
```

Рисунок 3.6.1 – функція що обробляє головну сторінку сайту

Далі йде функція, що поєднана зі сторінкою налаштувань користувача, вона є найбільшою серед всіх, за рахунок того, що вона в собі містить дуже багато умов і шаблонів які передаються. Так як, кожну форму потрібно перевіряти на

валідацію, перевіряти значення з БД і записувати нові, а в кінці передається на сторінку великий обсяг даних, який вже потім на самій сторінці обробляється, рисунок 3.6.2.

```
content = {
    'edit_nickname_form': EditUserNicknameForm(instance=user),
    'edit_tg_account_form': EditUserTGForm(instance=user),
    'edit_password_form': EditUserPasswordForm(instance=user),
    'keys_form': APIKeysForm(instance=user),
    'nickname_error': nickname_error,
    'tg_account_error': tg_account_error,
    'password_error': password_error,
    'key_error': key_error,
    'success': success,
}
return render(request, 'main/user_settings.html', content)
```

Рисунок 3.6.2 – передача даних в шаблон сторінки редагування

Наступною йде функція, що аналізує використовуючи скрипт, задану валюту, та повертає цілий список значень різних показників, також передається словник з умовними позначеннями таймфреймів та передається на шаблон, рисунок 3.6.3.

```
def graphics(request, token):
    token_changes = Token(token).get_price_with_large_orders(token)
    content = {
        'token': token,
        'time_frame': {
            '1d': 'D',
            '1h': '60',
            '15m': '15',
            '5m': '5'
        },
        'token_changes': token_changes
    }
    return render(request, 'main/graphics.html', content)
```

Рисунок 3.6.3 – функція що обробляє сторінку з аналізом валютної пари

Функція, що обробляє сторінку з аналізом криптовалютного гаманця користувача, всередині себе обробляє багато запитів до БД та генерує за допомогою скриптів графік змін, який потім з усіма значеннями передається в шаблон, рисунок 3.6.4.

Функція, за допомогою якої користувач виходить з аккаунта, реалізована як перенаправлення на головну сторінку, і закінчення поточну сесію користувача, рисунок 3.6.5.

```
1 usage
def bag(request):
    total_db = Total()
    total_db_dates = []
    user = User.objects.filter(nickname=request.session.get('user_id')).first()
    config = UserTokensInfo().set_api_key(user.api_key).set_secret_key(user.secret_key)
    tokens_with_amounts = config.get_account_tokens_balance
    total = config.get_total_tokens_balance(tokens_with_amounts)
    graphic = config.get_total_tokens_graphic(Total.objects.filter(user=user).values('total', 'date'))

    for obj in Total.objects.filter(user=user).all():
        total_db_dates.append(obj.date.strftime('%Y-%m-%d'))

    if str(datetime.date.today()) not in total_db_dates:
        total_db.total = total
        total_db.user = user
        total_db.save()

    content = {
        'tokens_with_amounts': tokens_with_amounts,
        'total': round(total, 2),
        'graphic': graphic
    }
    return render(request, 'main/bag.html', content)
```

Рисунок 3.6.4 – функція що обробляє сторінку аналізом гаманця користувача

```
def log_out(request):
    request.session['user_id'] = None
    return redirect('home')
```

Рисунок 3.6.5 – функція що завершує сесію користувача

Функції реєстрації та входа в систему містять в собі багато перевірок на валідність даних які дістаються з БД, рисунок 3.6.6 та рисунок 3.6.7.

Крім цього, був розроблений додатковий функціонал для сповіщення користувача, завдяки телеграм-боту. Він містить в собі функціонал щоденного інформування користувача, зареєстрованого в системі, стосовно стану його крипто гаманця. Він присилає користувачеві детальну інформацію і згенерований графік змін. Також, при першому запуску, перевіряє чи ви зареєстровані в системі, завдяки тому, що бот використовує БД web застосунка. Також бот може прислати стан гаманця, за запитом, відправляючи йому потрібну команду.

```python
def sign_up(request):
    error = None
    if request.method == 'POST':
        login_form = LoginForm(request.POST)
        if login_form.is_valid():
            email = login_form.cleaned_data['email']
            password = login_form.cleaned_data['password']
            user = User.objects.filter(email=email).first()
            if user is not None:
                if user.check_password(password):
                    login(request, user)
                    request.session['user_id'] = user.nickname
                    user.last_login = timezone.now()
                    user.save()
                    return redirect('home')
                else:
                    error = 'Invalid password'
            else:
                error = f'User with email "{email}", not registered'

    content = {
        'loginform': LoginForm(),
        'error': error
    }
    return render(request, 'main/sign-up.html', content)
```

Рисунок 3.6.4 – функція що обробляє сторінку аналізом гаманця користувача

```python
def registration(request):
    error = None
    if request.method == 'POST':
        registration_form = RegistrationForm(request.POST)
        if registration_form.is_valid():
            nickname = registration_form.cleaned_data['nickname']
            email = registration_form.cleaned_data['email']
            password = registration_form.cleaned_data['password']
            repeat_password = registration_form.cleaned_data['repeat_password']
            if not User.objects.filter(nickname=nickname).first():
                if not User.objects.filter(email=email).first():
                    if password == repeat_password:
                        registration_form.save()
                        user = User.objects.filter(email=email).first()
                        login(request, user)
                        request.session['user_id'] = user.nickname
                        user.last_login = timezone.now()
                        user.save()
                        return redirect('home')
                    else:
                        error = 'Passwords do not match'
                else:
                    error = f'User with email "{email}", is registered'
            else:
                error = f'Nickname {nickname} if busy'

    content = {
        'regform': RegistrationForm(),
        'error': error
    }
    return render(request, 'main/registration.html', content)
```

Рисунок 3.6.4 – функція що обробляє сторінку аналізом гаманця користувача

ВИСНОВКИ

Отже, у даній роботі проведено розробку web застосунку для аналізу криптовалют за допомогою мови програмування Python, HTML і CSS. Насамперед, визначено які функції притаманні продуктам типу web застосунок та проаналізовано, який найбільш ефективний вид аналізу криптовалютних пар підходить для поставленої задачі. Встановлено, що використання web застосунку значно спрощує та пришвидшує процес аналізу криптовалютних пар. Після аналізу технічних вимог до програмного продукту, була визначена його архітектура, а також технічно засоби для реалізації. Обраною мовою програмування є найбільш популярна мова Python, та web фреймворк Django, для користувацького інтерфейсу було обрано класичний HTML разом з CSS та JavaScript.

Була сформована загальна схема роботи програмного продукту, принцип роботи алгоритму та створена оптимальна робоча система для швидкого виконання і зберігання результатів в БД. Розробка відбувалася спираючись на всі ці рішення і реалізовувалась крок за кроком, що можна побачити в третьому розділі. Було проведено тестування програмного продукту для переконанняів корректній роботі. В підсумку програмний продукт задовольняє потребам користувачів, і може використовуватися як допоміжний інструмент для цільової аудиторії.

Результати досліджень бакалаврської роботи апробовані на Всеукраїнській науково-технічної конференції «Застосування програмного забезпечення в інфокомунікаційних технологіях», м. Київ: ДУТ, 2023 року.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Schinckus C. Crypto-currencies Trading and Energy Consumption [Електронний ресурс] / C. Schinckus, C. Nguyen. - 2020. - Режим доступу до ресурсу: https://www.researchgate.net/publication/340015034_Crypto-currencies_Trading_and_Energy_Consumption.
2. Kim G. Crypto BERT Incorporated Trading System [Електронний ресурс] / Gyeongmin Kim. - 2023. - Режим доступу до ресурсу: https://www.researchgate.net/publication/367063009_CBITS_Crypto_BERT_Incorporated_Trading_System.
3. An INTRANET-Based Web Application for College Management System Using Python with Django Web Framework [Електронний ресурс] / [N. Reddy, D. Tej, D. Koncha та ін.]. - 2023. - Режим доступу до ресурсу: https://www.researchgate.net/publication/368719592_An_INTRANET-Based_Web_Application_for_College_Management_System_Using_Python_with_Django_Web_Framework.
4. Що таке криптовалюта та які тенденції її розвитку в Україні та світі? [Електронний ресурс]. - 2022. - Режим доступу до ресурсу: https://www.binance.com/uk-UA/blog/markets/що-таке-криптовалюта-та-які-тенденції-її-розвитку-в-україні-та-світі-4214998246849039 26.
5. Використання фундаментального і технічного аналізу у криптовалюті [Електронний ресурс]. - 2023. - Режим доступу до ресурсу: https://uain.press/articles/1732857-1732857.
6. Як влаштований ринок криптоактивів і яка його роль у фінансовій системі [Електронний ресурс]. - 2023. - Режим доступу до ресурсу: https://mind.ua/publications/20253236-otaka-teper-valyuta-yak-vlashtovanij-rinok-kriptoaktiviv-i-yaka-jogo-rol-u-finansovij-sistemi.
7. How to read cryptocurrency charts [Електронний ресурс]. - 2021. - Режим доступу до ресурсу:

https://cointelegraph.com/learn/crypto-charts-101-how-to-read-cryptocurrency-charts.

8. Technical Analysis of Digital Currencies and the Most Important Indicators for Trading [Електронний ресурс]. - 2022. - Режим доступу до ресурсу: https://www.researchgate.net/publication/365999271_Technical_Analysis_of_Digital_Currencies_and_the_Most_Important_Indicators_for_Trading.
9. Scalping Techniques in Crypto Trading [Електронний ресурс]. - 2022. - Режим доступу до ресурсу: https://pintu.co.id/en/academy/post/what-is-scalping.
10. Посібник по Django для початківців [Електронний ресурс]. - 2017. - Режим доступу до ресурсу: https://codeguida.com/post/1039.
11. Розробка веб додатків з використанням Python і Django [Електронний ресурс]. - 2021. - Режим доступу до ресурсу: https://webcase.com.ua/uk/blog/razrabotka-veb-prilozhenij-s-ispolzovaniem-python-i-django/.
12. Run tasks of manage.py [Електронний ресурс]. - 2023. - Режим доступу до ресурсу: https://www.jetbrains.com/help/pycharm/running-manage-py.html#config.
13. Writing your first Django app [Електронний ресурс]. - 2021. - Режим доступу до ресурсу: https://docs.djangoproject.com/en/4.2/intro/tutorial01/.

ДОДАТОК А

Головний файл views.py:

```
import datetime

from django.contrib.auth import login
from django.shortcuts import render, redirect, get_object_or_404
from django.utils import timezone

from .forms import LoginForm, RegistrationForm, EditUserNicknameForm, EditUserPasswordForm, APIKeysForm, EditUserTGForm
from .models import User, TokenName, Total

from .scripts import TradingPairs, Token, UserTokensInfo


def base(request):
    now = timezone.localtime()
    current_time = now.time()
    target_time = datetime.time(1, 0, 0)
    if current_time < target_time or TokenName.objects.count() == 0:
        if TokenName.objects.count() != 0:
            TokenName.objects.all().delete()
        for token in TradingPairs().get_all_trading_pairs:
            TokenName.objects.create(token_name=token)

    tokens_name = TokenName.objects.all().values_list('token_name', flat=True)
    tokens_with_prices = Token().get_tokens_info(tokens_name)
```

```
                return render(request, 'main/base.html', {'tokens_with_prices': tokens_with_prices})


    def user_settings(request):
        user = get_object_or_404(User, nickname=request.session.get('user_id'))
        tg_account_error = None
        nickname_error = None
        password_error = None
        key_error = None
        success = None
        if request.method == 'POST':
                edit_nickname_form = EditUserNicknameForm(request.POST, instance=user)
            edit_tg_account_form = EditUserTGForm(request.POST, instance=user)
                edit_password_form = EditUserPasswordForm(request.POST, instance=user)
            keys_form = APIKeysForm(request.POST, instance=user)
                                                                user_in_db = User.objects.filter(nickname=request.session.get('user_id')).first()
            if edit_nickname_form.is_valid():
                nickname = edit_nickname_form.cleaned_data['nickname']
                if not User.objects.filter(nickname=nickname).first():
                    edit_nickname_form.save()
                    user = User.objects.filter(nickname=nickname).first()
                    request.session['user_id'] = user.nickname
                    success = 'Your nickname changed'
                else:
                    nickname_error = f'Nickname {nickname} is busy'
```

```
elif edit_tg_account_form.is_valid():
    tg_account = edit_tg_account_form.cleaned_data['tg_account']
    if not User.objects.filter(tg_account=tg_account).first():
        edit_tg_account_form.save()
        user = User.objects.filter(tg_account=tg_account).first()
        success = 'Your telegram account changed'
    else:
        tg_account_error = f'Telegram account {tg_account} is busy'
elif edit_password_form.is_valid():
    old_password = edit_password_form.cleaned_data['old_password']
    password = edit_password_form.cleaned_data['password']
    repeat_password = edit_password_form.cleaned_data['repeat_password']
    if old_password == user_in_db.password:
        if password == repeat_password:
            edit_password_form.save()
            success = 'Your password changed'
        else:
            password_error = 'New passwords do not match'
    else:
        password_error = 'Password of account incorrect'
elif keys_form.is_valid():
    api_key = keys_form.cleaned_data['api_key']
    secret_key = keys_form.cleaned_data['secret_key']
    password = keys_form.cleaned_data['password']
    if not User.objects.filter(api_key=api_key).first():
        if UserTokensInfo().check_keys(api_key, secret_key):
            if password == user_in_db.password:
                keys_form.save()
                success = 'API Keys changed'
```

```
                else:
                    key_error = 'Password of account incorrect'
            else:
                key_error = 'API Keys incorrect'
        else:
            key_error = 'This api key use another user'

    content = {
        'edit_nickname_form': EditUserNicknameForm(instance=user),
        'edit_tg_account_form': EditUserTGForm(instance=user),
        'edit_password_form': EditUserPasswordForm(instance=user),
        'keys_form': APIKeysForm(instance=user),
        'nickname_error': nickname_error,
        'tg_account_error': tg_account_error,
        'password_error': password_error,
        'key_error': key_error,
        'success': success,
    }
    return render(request, 'main/user_settings.html', content)


def graphics(request, token):
    token_changes = Token(token).get_price_with_large_orders(token)
    content = {
        'token': token,
        'time_frame': {
            '1d': 'D',
            '1h': '60',
            '15m': '15',
```

```
            '5m': '5'
        },
        'token_changes': token_changes
    }
    return render(request, 'main/graphics.html', content)


def bag(request):
    total_db = Total()
    total_db_dates = []
    user = User.objects.filter(nickname=request.session.get('user_id')).first()
                                                        config =
UserTokensInfo().set_api_key(user.api_key).set_secret_key(user.secret_key)
    tokens_with_amounts = config.get_account_tokens_balance
    total = config.get_total_tokens_balance(tokens_with_amounts)
                                                        graphic =
config.get_total_tokens_graphic(Total.objects.filter(user=user).values('total', 'date'))

    for obj in Total.objects.filter(user=user).all():
        total_db_dates.append(obj.date.strftime('%Y-%m-%d'))

    if str(datetime.date.today()) not in total_db_dates:
        total_db.total = total
        total_db.user = user
        total_db.save()

    content = {
        'tokens_with_amounts': tokens_with_amounts,
        'total': round(total, 2),
```

```
        'graphic': graphic
    }
    return render(request, 'main/bag.html', content)


def sign_up(request):
    error = None
    if request.method == 'POST':
        login_form = LoginForm(request.POST)
        if login_form.is_valid():
            email = login_form.cleaned_data['email']
            password = login_form.cleaned_data['password']
            user = User.objects.filter(email=email).first()
            if user is not None:
                if user.check_password(password):
                    login(request, user)
                    request.session['user_id'] = user.nickname
                    user.last_login = timezone.now()
                    user.save()
                    return redirect('home')
                else:
                    error = 'Invalid password'
            else:
                error = f'User with email "{email}", not registered'

    content = {
        'loginform': LoginForm(),
        'error': error
    }
```

```
    return render(request, 'main/sign-up.html', content)


def log_out(request):
    request.session['user_id'] = None
    return redirect('home')


def registration(request):
    error = None
    if request.method == 'POST':
        registration_form = RegistrationForm(request.POST)
        if registration_form.is_valid():
            nickname = registration_form.cleaned_data['nickname']
            email = registration_form.cleaned_data['email']
            password = registration_form.cleaned_data['password']
            repeat_password = registration_form.cleaned_data['repeat_password']
            if not User.objects.filter(nickname=nickname).first():
                if not User.objects.filter(email=email).first():
                    if password == repeat_password:
                        registration_form.save()
                        user = User.objects.filter(email=email).first()
                        login(request, user)
                        request.session['user_id'] = user.nickname
                        user.last_login = timezone.now()
                        user.save()
                        return redirect('home')
                    else:
                        error = 'Passwords do not match'
```

```
                else:
                    error = f'User with email "{email}", is registered'
            else:
                error = f'Nickname {nickname} if busy'

        content = {
            'regform': RegistrationForm(),
            'error': error
        }
        return render(request, 'main/registration.html', content)
```

Файл зі скриптом token_info.py:

```
import requests


class Token:

    def __init__(self, token_name: str = None):
        self.token_name = token_name
        self.ticker_price_url = 'https://api.binance.com/api/v3/ticker/price'
        self.ticker_24_url = 'https://api.binance.com/api/v3/ticker/24hr'

    @property
    def token_data(self):
        response = requests.get(self.ticker_price_url, params={'symbol': self.token_name})
        return response.json()

    def new_token_data(self, token):
```

```
    response = requests.get(self.ticker_24_url, params={'symbol': token})
    return response.json()

@property
def full_token_data(self):
    response = requests.get(self.ticker_24_url)
    return response.json()

@property
def get_token_price(self):
    return self.token_data['price']

def get_tokens_info(self, tokens_list: list[str]):
    tokens_with_prices = {}
    data = self.full_token_data
    for token in tokens_list:
        try:
            token_data = list(filter(lambda x: x['symbol'] == token, data))[0]
            token_price = round(float(token_data['askPrice']), 2)
            token_changes = round(float(token_data['priceChangePercent']), 2)
            tokens_with_prices[token] = [token_price, token_changes]
        except IndexError:
            continue
    return tokens_with_prices

def get_price_with_large_orders(self, pair):
    url = f'https://api.binance.com/api/v3/depth?symbol={pair}&limit=10000'
    response = requests.get(url).json()
```

```
        sell = None
        buy = None
        sell_percent = None
        buy_percent = None
        token_current_price = float(self.get_token_price)
        token_data = self.new_token_data(pair)
        token_changes_for_day = round(float(token_data['priceChangePercent']), 2)

        for bid in response['bids']:
            price, quantity = bid
            if float(quantity) * float(price) > 1000000:
                sell = round(float(price), 2)
                break

        for ask in response['asks']:
            price, quantity = ask
            if float(quantity) * float(price) > 1000000:
                buy = round(float(price), 2)
                break

        if buy:
            buy_percent = round(((buy - token_current_price) / token_current_price) * 100,
2)
        if sell:
            sell_percent = round(((token_current_price - sell) / token_current_price) * 100,
2)
        return [sell, sell_percent, buy, buy_percent, token_changes_for_day]
```

Файл зі скриптом trading_pairs.py:

```
import requests


class TradingPairs:

    def __int__(self):
        self.exchange_url = 'https://fapi.binance.com/fapi/v1/exchangeInfo'
        self.ticker_price_url = 'https://api.binance.com/api/v3/ticker/24hr'

    @property
    def get_all_trading_pairs(self):
        response = requests.get(self.exchange_url)
        data = response.json()
        trading_pairs = []

        for symbol in data['symbols']:
            if (
                symbol['contractType'] == 'PERPETUAL' or
                symbol['contractType'] == 'CURRENT_QUARTER' or
                symbol['contractType'] == 'NEXT_QUARTER'
            ):
                if symbol['symbol'][-4:] == 'USDT':
                    trading_pairs.append(symbol['symbol'])
        return trading_pairs

    @property
    def get_top_trading_pairs(self):
        tickers = []
```

```
        for pair in self.get_all_trading_pairs:
            params = {'symbol': pair}
            response = requests.get(self.ticker_price_url, params=params)
            ticker = response.json()
            try:
                ticker['priceChangePercent']
            except KeyError:
                continue
            tickers.append(ticker)

         top_pairs = sorted(tickers, key=lambda ticker: float(ticker['priceChangePercent']),
reverse=True)[:5]
        return [(pair['symbol'], pair['priceChangePercent']) for pair in top_pairs]
```

Файл зі скриптом user_token_info.py:

```
import base64
import hashlib
import hmac
import time
import urllib
from io import BytesIO
from urllib.parse import urlencode
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

import requests

from .token_info import Token
```

```
class UserTokensInfo:

    def __init__(self):
        self.api_key = None
        self.secret_key = None
        self.account_info_url = 'https://api.binance.com/api/v3/account'

    def set_api_key(self, api_key):
        self.api_key = api_key
        return self

    def set_secret_key(self, secret_key):
        self.secret_key = secret_key
        return self

    def check_keys(self, api_key, secret_key):
        self.set_api_key(api_key).set_secret_key(secret_key)

        params = {'timestamp': str(int(time.time() * 1000))}
        signature = hmac.new(
                            self.secret_key.encode('utf-8'),   urlencode(params).encode('utf-8'),
hashlib.sha256
        ).hexdigest()
        params['signature'] = signature
        headers = {'X-MBX-APIKEY': self.api_key}
        response = requests.get(self.account_info_url, headers=headers, params=params)
        if str(response) == '<Response [200]>':
```

```
            return True
        else:
            return False

    @property
    def get_account_info(self):
        if self.api_key and self.secret_key:
            params = {'timestamp': str(int(time.time() * 1000))}
            signature = hmac.new(
                            self.secret_key.encode('utf-8'),  urlencode(params).encode('utf-8'),
hashlib.sha256
            ).hexdigest()
            params['signature'] = signature
            headers = {'X-MBX-APIKEY': self.api_key}
                        response  =  requests.get(self.account_info_url,  headers=headers,
params=params)
            return response.json()
        else:
            print('Set API key and Secret key')
            return None

    @property
    def get_account_tokens_balance(self):
        account_balances = self.get_account_info['balances']
        balances = {}

        for balance in account_balances:
            if float(balance['free']) > 0 or float(balance['locked']) > 0:
                token_name = balance['asset'] + 'USDT'
```

```
            try:
                token_price = Token(token_name).get_token_price
            except KeyError:
                continue
            try:
                        balance_value = (float(balance['free']) + float(balance['locked'])) *
float(token_price)
                balances[balance['asset']] = round(balance_value, 2)
            except KeyError:
                continue
        return balances

    @staticmethod
    def get_total_tokens_balance(token_balances):
        total = 0
        for token, value in token_balances.items():
            total += value
        return total

    @staticmethod
    def get_total_tokens_graphic(totals):
        prices = []
        dates = []
        for col in totals:
            prices.append(col['total'])
            dates.append(str(col['date'])[5:])

        fig, ax = plt.subplots()
        ax.plot(dates, prices)
```

```
        ax.tick_params(labelsize=8)
        plt.xticks(rotation=60)

        buf = BytesIO()
        fig.savefig(buf, format='png')
        buf.seek(0)

        string = base64.b64encode(buf.read())
        graphic = '' + urllib.parse.quote(string)
        return graphic
```

Файл з моделью БД models.py:

```
from django.db import models



class User(models.Model):
    email = models.EmailField('User Email')
    password = models.CharField('Password', max_length=100)
    nickname = models.CharField('Nickname', max_length=50)
    api_key = models.CharField('API Key', max_length=100)
    secret_key = models.CharField('Secret Key', max_length=100)
        tg_account = models.CharField('Telegram Account', max_length=50, null=True,
blank=True)
    last_login = models.DateTimeField(null=True, blank=True)

    class Meta:
        app_label = 'main'

    def check_password(self, raw_password):
```

```
        print(raw_password, self.password)

        return raw_password == self.password


class TokenName(models.Model):
    token_name = models.CharField('Token Name', max_length=20)

    class Meta:
        app_label = 'main'


class FavoriteTokens(models.Model):
    favorite_token = models.CharField('Favorite Token', max_length=20)
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    class Meta:
        app_label = 'main'


class Total(models.Model):
    date = models.DateField('Date', auto_now_add=True)
    total = models.DecimalField('Total', max_digits=10, decimal_places=2)
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    class Meta:
        app_label = 'main'
```

```
class TelegramChats(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    chat_id = models.CharField('Chat id', max_length=20)

    class Meta:
        app_label = 'main'
```

Файл з формами forms.py:

```
from .models import User
from django.forms import ModelForm, EmailInput, PasswordInput, TextInput, CharField


class LoginForm(ModelForm):
    class Meta:
        model = User
        fields = ['email', 'password']
        widgets = {
            'email': EmailInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'Email address'
            }),
            'password': PasswordInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'Password'
            }),
        }
```

```
class RegistrationForm(ModelForm):
    repeat_password = CharField(widget=PasswordInput(attrs={
        'class': 'form-control mb-3',
        'placeholder': 'Repeat Password'
    }))

    class Meta:
        model = User
        fields = ['nickname', 'email', 'password']
        widgets = {
            'nickname': TextInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'Nickname'
            }),
            'email': EmailInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'Email address'
            }),
            'password': PasswordInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'Password'
            }),
        }


class EditUserPasswordForm(ModelForm):
    old_password = CharField(widget=PasswordInput(attrs={
        'class': 'form-control mb-3',
        'placeholder': 'Account Password'
```

```
    }))
    repeat_password = CharField(widget=PasswordInput(attrs={
        'class': 'form-control mb-3',
        'placeholder': 'Repeat New Password'
    }))

    class Meta:
        model = User
        fields = ['password']
        widgets = {
            'password': PasswordInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'New Password'
            }),
        }


class EditUserNicknameForm(ModelForm):
    class Meta:
        model = User
        fields = ['nickname']
        widgets = {
            'nickname': TextInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'New Nickname'
            })
        }

class EditUserTGForm(ModelForm):
```

```
    class Meta:
        model = User
        fields = ['tg_account']
        widgets = {
            'tg_account': TextInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'my_telegram without @'
            })
        }


class APIKeysForm(ModelForm):
    password = CharField(widget=PasswordInput(attrs={
        'class': 'form-control mb-3',
        'placeholder': 'Account Password'
    }))

    class Meta:
        model = User
        fields = ['api_key', 'secret_key']
        widgets = {
            'api_key': TextInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'Binance API Key'
            }),
            'secret_key': PasswordInput(attrs={
                'class': 'form-control mb-3',
                'placeholder': 'Binance Secret Key'
            }),
```

```
}
```

Файл з шаблоном base.html:

```
<!doctype html>
<html lang="en">
{% load static %}
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
        content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>{% block title %}Main{% endblock %}</title>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
  <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
</head>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
      <a class="navbar-brand" href="/">Cevaluation</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExample07" aria-controls="navbarsExample07" aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarsExample07">
        <ul class="navbar-nav mr-auto">
          <li class="nav-item">
```

```
                    <a class="nav-link" href="{% url 'bag' %}">Wallet analysis<span class="sr-only">(current)</span></a>
                </li>
            </ul>
            {% if request.session.user_id %}
                    <a class="col-2 white_text" href="{% url 'user_settings' %}">{{ request.session.user_id }}</a>
            {% else %}
            <a class="col-2 btn btn-primary" href="{% url 'sign-up' %}">Sign Up</a>
            {% endif %}
        </div>
    </div>
</nav>
{% block content %}
<body class="bg-dark">
   <div class="container">
      <div class="row justify-content-center align-items-center">
      {% for token, info in tokens_with_prices.items %}
         <a href="graphics/{{ token }}" class="col-3 m-2 custom-border">
         <div class="d-flex justify-content-between" style="text-align: center;">
            <div style="padding-top: 10px;">
               <h6 class="white_text" style="border-bottom: 1px dashed white;">{{ token }}</h6>
               <h6 class="white_text">Price: {{ info.0 }}$</h6>
            </div>
               <div style="padding-top: 10px; border-left: 1px dashed white; padding-left: 5px; display: flex; align-items: center; justify-content: center;">
                  <h5 class="{% if info.1 > 0 %}positive{% else %}negative{% endif %}" style="margin-left: 10px;">{{ info.1 }}%</h5>
```

```
            </div>
          </div>
          </a>
        {% endfor %}
        </div>
    </div>
</body>

{% endblock %}
</html>
```

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8a
btTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.min.js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK
/l8WvCWPIPm49" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/js/bootstrap.min.js"
integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6O
W/JmZQ5stwEULTy" crossorigin="anonymous"></script>
```

Файл з шаблоном сторінки з графіками graphics.html:

```
{% extends 'main/base.html' %}

{% block title %}
Graphick
{% endblock %}
```

```
{% block content %}
<body class="bg-dark">
<div class="container">
    <h4 class="white_text" style="text-align: center;">Graphics {{ token }} in different timeframes </h4>
        <h5 class="white_text" style="margin-bottom: 20px;">Change price today: {{ token_changes.4 }}%</h5>
    {% if token_changes.0 %}
        <h5 class="white_text">Support level on the price: {{ token_changes.0 }}$</h5>
            <h5 class="white_text" style="margin-bottom: 20px;">To support level: {{ token_changes.1 }}%</h5>
    {% endif %}
    {% if token_changes.2 %}
            <h5 class="white_text">Resistance level on the price: {{ token_changes.2 }}$</h5>
        <h5 class="white_text">To resistance level: {{ token_changes.3 }}%</h5>
    {% endif %}
    <div class="row justify-content-center align-items-center">
        {% for name, val in time_frame.items %}
        <div class="col-6">
            <h5 class="white_text mt-4" style="text-align: center;">Timeframe: {{ name }}</h5>
            <div class="tradingview-widget-container">
                <div id="tradingview_{{ name }}"></div>
                <script type="text/javascript">
                    var container = document.getElementById("tradingview_{{ name }}");
                    container.style.height = (window.innerHeight * 0.4) + "px";

                    window.addEventListener('resize', function(event){
```

```
                container.style.height = (window.innerHeight * 0.5) + "px";
            });
        </script>
        <script type="text/javascript" src="https://s3.tradingview.com/tv.js"></script>
        <script type="text/javascript">
            new TradingView.widget({
                "autosize": true,
                "symbol": "BINANCE:{{ token }}",
                "interval": "{{ val }}",
                "timezone": "Etc/UTC",
                "theme": "dark",
                "style": "1",
                "locale": "en",
                "toolbar_bg": "#f1f3f6",
                "enable_publishing": true,
                "hide_top_toolbar": true,
                "hide_legend": true,
                "save_image": false,
                "container_id": "tradingview_{{ name }}"
            });
        </script>
      </div>
    </div>
    {% endfor %}
  </div>
</div>
</body>
{% endblock %}
```

Файл з шаблоном сторінки аналізу гаманця bag.html:

```
{% extends 'main/base.html' %}

{% block title %}
Bag
{% endblock %}

{% block content %}
<body class="bg-dark">
   <div class="container">
      <h5 class="white_text" style="text-align: center;">Bag Changes</h5>
      <div style="text-align:center;">
         <img class="mb-4" src="{{ graphic }}">
      </div>
      <h5 class="white_text mb-2" style="text-align: center;">Total: {{ total }}$</h5>
            <h5 class="white_text" style="text-align: center;">Tokens in Binance spot bag:</h5>
      {% for token, amount in tokens_with_amounts.items %}
            <h6 class="white_text" style="text-align: center;">{{ token }} = {{ amount }}$</h6>
      {% endfor %}
   </div>
</body>
{% endblock %}
```

Файл з шаблоном сторінки логіна sign-up.html:

```
{% extends 'main/base.html' %}

{% block title %}
```

```
Login
{% endblock %}

{% block content %}
<body class="text-center bg-dark">
   <form class="form-signin mx-auto col-5" method="post">
      <div class="form-group">
         <h1 class="h3 mb-3 font-weight-normal white_text">Please sign in</h1>
         {% csrf_token %}
         {% if error %}
            <div class="alert alert-danger">
               {{ error }}
            </div>
         {% endif %}
         {{ loginform.email }}
         {{ loginform.password }}
                                 <button class="btn btn-lg btn-primary btn-block"
type="submit">Login</button>
             <a href="{% url 'registration' %}" class="btn btn-lg btn-primary btn-block"
type="submit">Registration</a>
      </div>
   </form>
</body>
{% endblock %}
```

Файл з шаблоном сторінки реєстрації registration.html:

```
{% extends 'main/base.html' %}

{% block title %}
```

```
Registration
{% endblock %}

{% block content %}
<body class="text-center bg-dark">
   <form class="form-signin mx-auto col-5" method="post">
      <div class="form-group">
         <h1 class="h3 mb-3 font-weight-normal white_text">Registration</h1>
         {% csrf_token %}
         {% if error %}
            <div class="alert alert-danger">
               {{ error }}
            </div>
         {% endif %}
         {{ regform.nickname }}
         {{ regform.email }}
         {{ regform.password }}
         {{ regform.repeat_password }}
                                 <button class="btn btn-lg btn-primary btn-block"
type="submit">Registrate</button>
      </div>
   </form>
</body>
{% endblock %}
```

Файл з шаблоном сторінки редагування профіля користувача user_settings.html:

```
{% extends 'main/base.html' %}

{% block title %}
```

```
User Settings
{% endblock %}

{% block content %}
<body class="text-center bg-dark">
    <h1 class="h3 mb-3 font-weight-normal white_text">Edit {{ request.session.user_id
}} account</h1>
    {% if success %}
        <div class="alert alert-danger">
            {{ success }}
        </div>
    {% endif %}
    <form class="form-signin mx-auto col-5" method="post">
        <div class="form-group">
            <h1 class="h3 mb-3 font-weight-normal white_text">Edit nickname</h1>
            {% csrf_token %}
            {% if nickname_error %}
                <div class="alert alert-danger">
                    {{ nickname_error }}
                </div>
            {% endif %}
            {{ edit_nickname_form.nickname }}
            <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
        </div>
    </form>

    <form class="form-signin mx-auto col-5" method="post">
        <div class="form-group">
```

```
                <h1 class="h3 mb-3 font-weight-normal white_text">Edit telegram account</h1>
        {% csrf_token %}
        {% if tg_account_error %}
          <div class="alert alert-danger">
            {{ tg_account_error }}
          </div>
        {% endif %}
        {{ edit_tg_account_form.tg_account }}
        <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
      </div>
    </form>

    <form class="form-signin mx-auto col-5" method="post">
      <div class="form-group">
        <h1 class="h3 mb-3 font-weight-normal white_text">Edit password</h1>
        {% csrf_token %}
        {% if password_error %}
          <div class="alert alert-danger">
            {{ password_error }}
          </div>
        {% endif %}
        {{ edit_password_form.old_password }}
        {{ edit_password_form.password }}
        {{ edit_password_form.repeat_password }}
        <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
      </div>
    </form>
    <form class="form-signin mx-auto col-5" method="post">
```

```
      <div class="form-group">
                <h1 class="h3 mb-3 font-weight-normal white_text">Edit Binance API keys</h1>
          {% csrf_token %}
          {% if key_error %}
            <div class="alert alert-danger">
               {{ key_error }}
            </div>
          {% endif %}
          {{ keys_form.api_key }}
          {{ keys_form.secret_key }}
          {{ keys_form.password }}
         <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
      </div>
   </form>
    <a href="{% url 'log_out' %}" class="btn btn-lg btn-primary btn-block mx-auto col-5 log_out" type="submit">Log out</a>
</body>
{% endblock %}
```

Файл з допоміжним телеграм ботом notification_bot.py

```
import os
import time
import urllib
from io import BytesIO
from pathlib import Path

from telegram.ext import Updater, CommandHandler
from datetime import time
```

```
import django
from django.conf import settings

BASE_DIR = Path(__file__).resolve().parent

settings.configure(
    DEBUG=True,
    DATABASES={
        'default': {
            'ENGINE': 'django.db.backends.sqlite3',
            'NAME': BASE_DIR / 'db.sqlite3',
        }
    }
)

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'crypto_analitic_site.settings')
django.setup()

from main.models import TelegramChats, User, Total
from main.scripts import UserTokensInfo

TOKEN = "5917187870:AAEDZrFE3rJirXdiokw0S5noAYXZX-2LO3Q"


def add_user(update, context):
    username = update.message.from_user.username
    chat_id = update.message.chat_id
    user = User.objects.filter(tg_account=username).first()
    if user:
```

```
        telegram_chat = TelegramChats.objects.filter(user=user).first()
        if not telegram_chat:
            telegram_chats = TelegramChats()
            telegram_chats.user = user
            telegram_chats.chat_id = chat_id
            telegram_chats.save()
                    update.message.reply_text(f'Hello, {user.nickname} im bot from site Cevaluation. And i will help you')
        else:
            update.message.reply_text('You now log in in system!')
    else:
        update.message.reply_text('Sorry, but you not register in our site(')


def send_graphic(update, context):
    chat_id = update.message.chat_id
    telegram_chat = TelegramChats.objects.filter(chat_id=chat_id).first()
    if telegram_chat:
                                                        config = UserTokensInfo().set_api_key(telegram_chat.user.api_key).set_secret_key(telegram_chat.user.secret_key)
        update.message.reply_photo(get_graphic(telegram_chat, config))
         update.message.reply_text(f'{round(get_total(config), 2)}, wow, this all you got in your wallet!')

def get_graphic(telegram_chat, config):
                                                    graphic = config.get_total_tokens_graphic(Total.objects.filter(user=telegram_chat.user).values('total', 'date'))
```

```
    return BytesIO(urllib.request.urlopen(graphic).read())

def get_total(config):
    tokens_with_amounts = config.get_account_tokens_balance
    return config.get_total_tokens_balance(tokens_with_amounts)

def send_message(context):
    all_users = TelegramChats.objects.all()
    for user in all_users:
        chat_id = user.chat_id
                                                                config =
UserTokensInfo().set_api_key(user.user.api_key).set_secret_key(user.user.secret_key)
            context.bot.send_message(chat_id=chat_id, text=f'{round(get_total(config), 2)}',
photo=get_graphic(user, config))

def main():
    updater = Updater(token=TOKEN, use_context=True)
    dispatcher = updater.dispatcher
    start_handler = CommandHandler('start', add_user)
    dispatcher.add_handler(start_handler)
    graphics = CommandHandler('graphic', send_graphic)
    dispatcher.add_handler(graphics)
    job_queue = updater.job_queue
    job_queue.run_daily(send_message, time(hour=12, minute=0, second=0))
    updater.start_polling()

if __name__ == '__main__':
    main()
```

ДОДАТОК Б

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web застосунку для аналізу криптовалют з використанням Python, HTML, CSS

Виконав студент 5 курсу
групи ППЗ-51
Патока Владислав Володимирович
Керівник роботи
К.т.н, доц, доцент кафедри ІПЗ Аверічев Ігор Миколайович

Київ – 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи:** спрощення процесу аналізу криптовалют за рахунок використання web застосунку засобами Python, HTML, CSS.
- **Об'єкт дослідження:** процес аналізу криптовалют**.**
- **Предмет дослідження:** програмне забезпечення для аналізу криптовалют.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати теоретичні дослідження в сфері автоматизованих систем аналізу криптовалют.
2. Визначити переваги та можливості сучасних програмних засобів.
3. Дослідити алгоритми аналізу криптовалютного ринку та визначити найефективніший з них.
4. Спроектувати та розробити застосунок для автоматизації процесу аналізу криптовалютної пари.

3

АНАЛІЗ АНАЛОГІВ

Показник	CoinMarketCap	TradingView	GagarinNews	WinScreener	Cevaluation
Зміна вартості	+	+	-	-	+
Таймфрейми	-	+	-	+	+
Лінії підтримки і опори	-	+	-	+	+
Тенденції	+	+	-	+	+
Лімітні заявки	-	-	-	+	+
Власний гаманець	-	-	-	-	+

4

ВИМОГИ ДО ДОДАТКУ

1. Можливість відслідковувати зміни валютних пар.
2. Можливість виводу графіків змін в різних періодах часу.
3. Можливість пошуку ущільнень на цінах криптовалют.
4. Засоби для аналізу криптовалютного гаманця користувача (додавання гаманця, видалення гаманця, відстеження змін в ньому).
5. Можливість реєстрації користувача.
6. Засоби для роботи з даними користувача (редагування псевдоніма користувача, редагування пароля від аккаунта, редагування API ключів, редагування підключеного телеграм акаунту).
7. Можливість надавати щоденний звіт стосовно гаманця користувача в телеграмі.

5

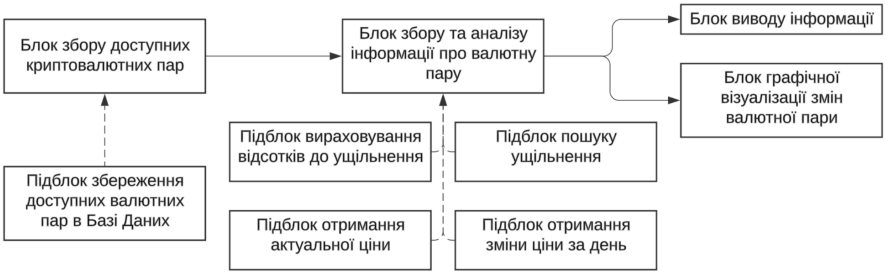
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Фреймворк для розробки веб застосунків

Шаблонізатор

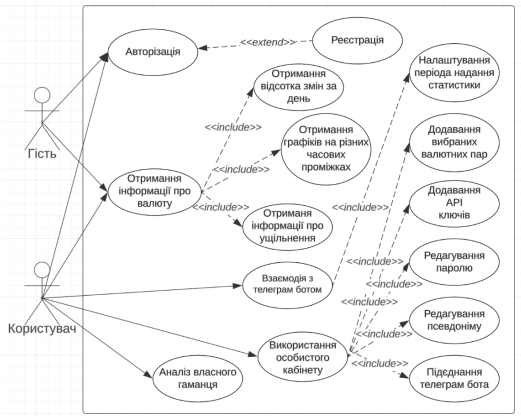
6

СХЕМА СТРУКТУРИ СИСТЕМИ



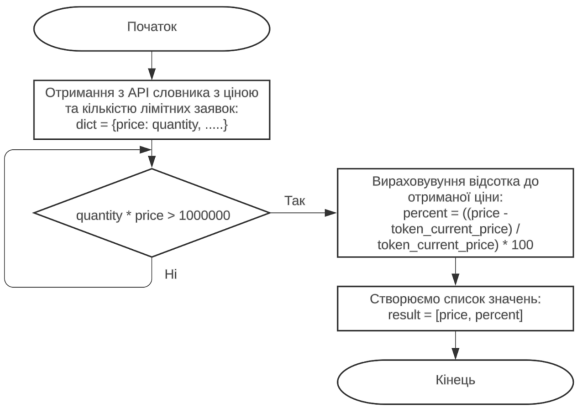
7

ДІАГРАМА ПРЕЦЕДЕНТІВ



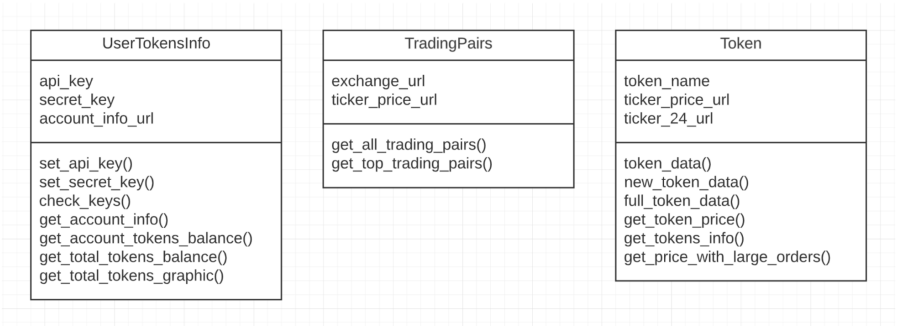
8

СХЕМА АЛГОРИТМУ АНАЛІЗУ



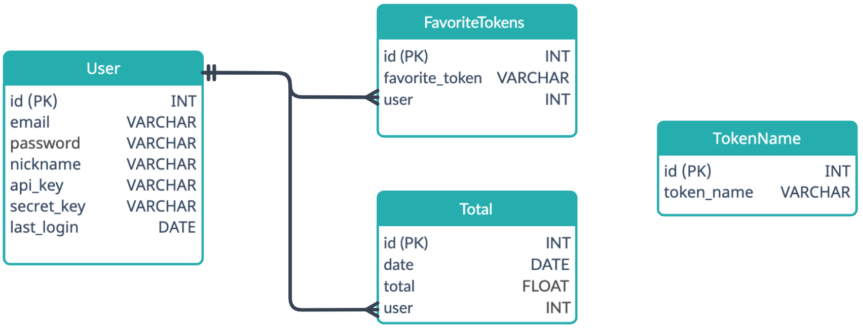
9

ДІАГРАМА КЛАСІВ



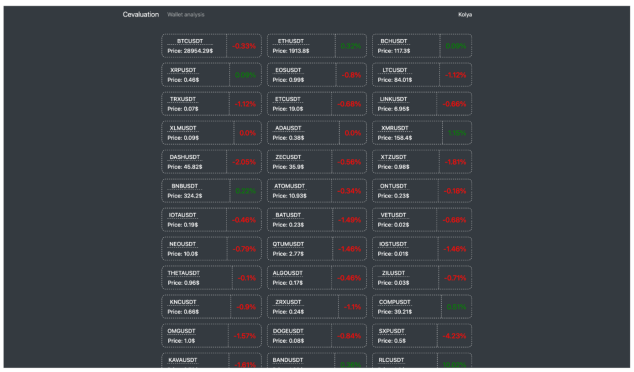
10

СХЕМА БАЗИ ДАНИХ



11

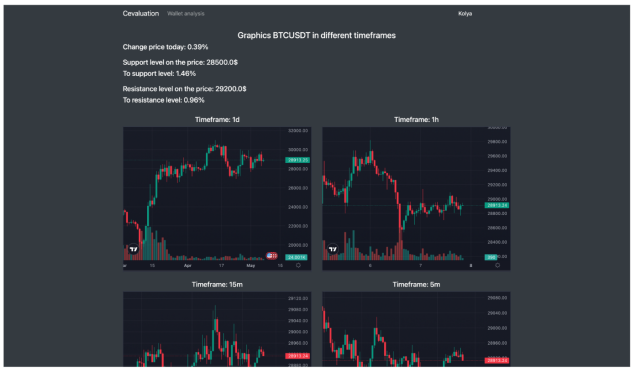
ЕКРАННІ ФОРМИ



Головний екран додатку

12

ЕКРАННІ ФОРМИ



Екран детальної інформації про валюту

13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Аверічев І.М., Патока В.В. Засоби програмної реалізації додатку для аналізу криптовалют // Всеукраїнська науково-технічна конференція “Застосування програмного забезпечення в інфокомунікаційних технологіях” - Київ: ДУТ, 2023 - 106-107 с..
2. Аверічев І.М., Патока В.В. IoT та штучний інтелект // Всеукраїнська науково-технічна конференція “Застосування програмного забезпечення в інфокомунікаційних технологіях” - Київ: ДУТ, 2023 - 108-109 с..

14

ВИСНОВКИ

1. Проаналізовано теоретичні дослідження в сфері автоматизованих систем аналізу криптовалют.
2. Визначено переваги та можливості сучасних програмних засобів.
3. Досліджено алгоритми аналізу криптовалютного ринку та визначено найефективніший з них.
4. Розроблено web застосунку для аналізу криптовалют за допомогою мови програмування Python, HTML і CSS. Визначено які функції притаманні продуктам типу web застосунок та проаналізовано, який найбільш ефективний вид аналізу краптовалютної пари підходить для поставленої задачі. Встановлено, що використання web застосунку значно спрощує та пришвидшує процес аналізу криптовалютних пар. Після аналізу технічних вимог до програмного продукту, була визначена його архітектура, а також технічно засоби для реалізації. Сформована загальна схема роботи програмного продукту, принцип роботи алгоритму та створена оптимальна робоча система для швидкого виконання і зберігання результатів в БД.

15

ДЯКУЮ ЗА УВАГУ!