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 ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМА ПІДТРИМКИ ТОРГІВЛІ КРИПТОАКТИВАМИ. СПЕЦ ЧАСТИНА: РОЗРОБКА МОДУЛЯ АГРЕГАЦІЇ ДАНИХ МОВОЮ C#**»

Виконав: студент 4 курсу, групи ПД–44
спеціальності
121 Інженерія програмного забезпечення
Столяр П.Ю.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Нормоконтроль _____
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти «Бакалавр»
Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення

Негоденко В.В.
“ ___ ” _____ 2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

СТОЛЯРА ПАВЛА ЮРІЙОВИЧА
(прізвище, ім‘я, по батькові)

1.Тема роботи: «Розробка програмного забезпечення системи підтримки торгівлі криптоактивами. Спец. частина: Розробка модуля агрегації даних мовою C#»
Керівник роботи: Жебка В.В., д.т.н., доц., зав. кафедри ТЦР
(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26
2. Строк подання студентом роботи «1» червня 2023 року
3. Вихідні дані до роботи:
3.1 Науково-технічна література з питань, пов‘язаних із застосування API та парсинг
3.2 Практичний досвід збору та стандартизації за допомогою API та парсингу
3.3 Концепція побудови веб-додатку
4. Перелік демонстраційних матеріалів
4.1 Тема дипломної роботи
4.2 Мета роботи. Об’єкт дослідження. Предмет дослідження
4.3 Результат дослідження розробки підсистеми даних для стандартизації обчислень
4.5 Результати дослідження та опис реалізації програми
4.6 Апробація результатів дослідження
4.7 Висновки
5. Дата видачі завдання 25.02.2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	25.02.23-26.02.23	Виконано
2	Аналіз та дослідження існуючих аналогів	27.02.23-05.03.23	Виконано
3	Моделювання об’єкта проєктування	06.03.23-26.03.23	Виконано
4	Дослідження програмних засобів	27.03.23-29.03.23	
5	Розробка веб-додатку	01.04.23-09.05.23	Виконано
6	Тестування	10.05.23-13.05.23	Виконано
7	Вступ, реферат, висновки	14.05.23-20.05.23	Виконано
8	Розробка демонстраційних матеріалів	21.05.23-25.05.23	Виконано
9	Попередній захист роботи	26.05.23	Виконано
10	Здача роботи	01.06.23	Виконано

Студент _____ Столяр П.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Жебка В.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 51 с., 1 табл., 13 рис., 54 джерел.

Об'єкт дослідження: збір інформації з бірж через API та парсинг.

Предмет дослідження: модуль агрегації даних мовою C# для збору, обробки та об'єднання даних.

Мета роботи: автоматизація збору даних з торгового ринку криптовалют за допомогою модулю агрегації даних мовою C#.

В ході цієї дипломної роботи було проведено дослідження і створено сервіс, спрямованого на заробіток від внутрішньобіржового та міжбіржового арбітражу криптовалюти. Основною метою цього сервісу була розробка ефективного автоматизованого інструменту, який допомагає користувачам виявляти можливості, що зможуть зробити їх успішними на ринку.

На першому етапі дослідження було проведено аналітичну роботу з вивчення основ і особливостей внутрішньобіржового та міжбіржового криптовалютного арбітражу. Були досліджені різні стратегії та тактики, що використовуються трейдерами для отримання прибутку від розбіжностей в цінах на різних біржах. Також було вивчено інструменти та сервіси, які допомагають у здійсненні арбітражних операцій.

Другий етап дослідження був присвячений створенню самого сервісу. Було розроблено архітектуру та функціонал сервісу, включаючи можливість досліджувати криптовалютні пари на різних біржах та збирати необхідні дані про ціни та обсяги торгівлі. Інтеграція з API бірж була ключовим компонентом розробки, оскільки це дало доступ до точних та актуальних ринкових даних.

Створення алгоритмів і моделей для оцінки даних і пошуку потенційно прибуткових перспектив було частиною третього етапу дослідження. Сучасні методи статистичного аналізу та машинного навчання були використані для прогнозування цін і визначення цінових коливань на декількох біржах. Результати цього аналізу допомогли визначити саме ті комбінації монетних пар, які можуть бути прибутковими.

Тестування та оптимізація сервісу були останніми етапами проєкту. В експериментах використовувалися реальні дані, і результати показали, наскільки ефективно сервіс може заробляти на ринку. Переваги та ефективність розробленого сервісу були підтверджені, коли результати були порівняні з альтернативними рішеннями, які вже використовуються.

Галузь використання – трейдери, інвестори та інші учасники ринку криптовалют, які потребують автоматизації аналізу ринку та обробки необхідної інформації.

Ключові слова - веб-додаток, фреймворк, автоматизація, розробка, проєктування, Framework, C#, ASP.NET, агрегація.

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ, УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

MVC	-	Model-View-Controller
SQL	-	Structured Query Language
HTML	-	HyperText Markup Language
.NET	-	Network Enabled Technology
ASP.NET	-	Active Server Pages.NET.
ADO.NET	-	ActiveX Data Objects .NET
IL	-	Intermediate Language
JSON	-	JavaScript Object Notation
URL	-	Uniform Resource Locator
LINQ	-	Language Integrated Query
JWT	-	JSON Web Token
OAuth	-	Open Authorization
IIS	-	Internet Information Services
API	-	Application Programming Interface
IDE	-	Integrated Development Environment
TFS	-	Team Foundation Server
ORM	-	Object-Relational Mapping
GIT	-	Global Information Tracker
MySQL	-	My Structured Query Language
DOM	-	Document Object Model
XML	-	eXtensible Markup Language
CSV	-	Comma-Separated Values
W3C	-	World Wide Web Consortium
ISO	-	International Organization for Standardization
CSS	-	Cascading Style Sheets

ВСТУП

У сучасному фінансовому світі, криптовалюти відіграють значну роль, відкриваючи нові можливості для заробітку на фінансових ринках. Зокрема, зросла популярність внутрішньо- та міжбіржового криптовалютного арбітражу, коли трейдери використовують коливання цін на кількох біржах для заробітку. Однак, процедура проведення операцій на ринку може бути складною і трудомісткою.

В рамках даної дипломної роботи я маю можливість запропонувати сервіс, який спрощує та автоматизує внутрішньобіржовий та міжбіржовий криптовалютний арбітраж, надаючи трейдерам практичний та ефективний інструмент для отримання прибутку. Цей сервіс поєднує в собі сучасні аналітичні інструменти, швидкість обробки даних та точність аналізу, що робить його незамінним помічником для користувачів.

Основна мета сервісу - оцінити криптовалютні пари, що торгуються на різних біржах, зібрати дані про їх ціни та обсяги торгів, а також виявити можливі можливості для прибуткових арбітражних угод. Сервіс показує тільки ті криптовалютні пари з різницею в ціні, які дозволяють отримувати прибуток завдяки високоточному аналізу та автоматичній фільтрації ринку.

Сервіс використовує складні моделі та алгоритми для пошуку можливих арбітражних угод. Щоб знайти найбільш прибуткові шанси, він вивчає величезну кількість даних, включаючи ціни на криптовалюту, обсяги торгівлі та ліквідність.

Зручність використання даного сервісу - одна з його основних переваг. Щоб гарантувати, що навіть клієнти з невеликим досвідом зможуть швидко переглядати й використовувати всі можливості сервісу, було приділено особливу увагу створенню інтуїтивно зрозумілого і зручного інтерфейсу.

Ми хочемо ретельно протестувати сервіс і впровадити його для використання в режимі реального часу по завершенню цієї дипломної роботи. Ми переконані, що наш сервіс швидко зарекомендує себе як обов'язковий ресурс для інвесторів і трейдерів, які бажають успішно здійснювати операції на ринку криптоактивів.

Завдяки цьому сервісу, користувачі зможуть отримати переваги арбітражної торгівлі криптовалютою, забезпечуючи собі стабільний та прибутковий рівень доходу. Ми віримо, що наша розробка стане цінним інструментом для професіоналів та новачків у світі криптовалюти.

Об'єкт дослідження: збір інформації з бірж через API та парсинг.

Предмет дослідження: модуль агрегації даних мовою C# для збору, обробки та об'єднання даних.

*Мета роботи:*автоматизація збору даних з торгового ринку криптовалют за допомогою модулю агрегації даних мовою C#.

У результаті поставленої мети були зроблені наступні завдання:

1. Проаналізувати предметну область збору даних.
2. Проаналізувати існуючі аналоги.
3. Під’єднати збір даних за допомогою API.
4. Розробити підсистему даних для стандартизації обчислень.
5. Провести тестування модуля збору даних та агрегації даних.
6. Розробити для фронтенду дизайн, форми та авторизацію.
7. Розробити фільтр результатів по параметрах.
8. Провести тестування веб-додатку на адаптивність.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проєкту

Неможливо недооцінити важливість сервісу автоматизованого криптовалютного арбітражу в сучасному фінансовому світі. Висока волатильність криптовалют і можливість заробити на різниці в ціні монет між біржами привертають увагу трейдерів та інвесторів завдяки зростаючій популярності.

Непередбачуваність і швидкий рух цін характеризують сучасний ринок криптоіндустрії. Завдяки цим сприятливим умовам трейдери можуть скористатися різницею в цінах між різними ринками на одній чи різних біржах, використовуючи метод арбітражу. Однак через велику кількість ринків, торгових пар і швидких змін цін на ринку криптовалют може бути складно знайти та скористатися можливостями арбітражу. Самостійно відстежувати та аналізувати такі великі обсяги даних може бути дуже складно і забирати багато часу.

У цій ситуації трейдерам та інвесторам можуть допомогти автоматизовані сервіси криптовалютного арбітражу. Ці сервіси збирають і аналізують дані з численних бірж і внутрішньобіржових пар, використовуючи передові технології, такі як штучний інтелект і машинне навчання. Вони надають клієнтам практичні методи виявлення цінових розривів, змін в обсягах торгів та інших ознак арбітражних можливостей. Ці послуги дають трейдерам конкурентну перевагу на ринку, допомагаючи їм приймати швидкі та ефективні рішення щодо торговельної діяльності.

Можливість скористатися перевагами внутрішнього та міжбіржового арбітражу є однією з ключових особливостей, які роблять послуги автоматизованого криптовалютного арбітражу актуальними. Ці послуги дають трейдерам можливість скористатися різницею в цінах, розділяючи пари на біржі на основі ціни, обсягу та інших факторів. Це особливо важливо з огляду на високий рівень волатильності на ринку криптоактивів, де раптові зміни цін можуть сприяти гарному заробітку чи великій втраті.

Однак цінність автоматизованих сервісів не обмежується потенційною фінансовою вигодою. Вони також допомагають знизити ризик і підвищити ефективність торгівлі. Трейдери можуть отримувати більш точну і неупереджену інформацію для прийняття рішень завдяки постійному моніторингу ринку і швидкому реагуванню на зміни. Автоматизовані системи можуть швидко виявляти та аналізувати зміни в ціні та обсягах торгів. Це дозволяє уникнути затримок і помилок, які можуть виникнути при ручному перегляді та обробці ордерів.

Важливо відзначити, що використання послуг автоматизованого криптовалютного арбітражу пов'язане з певними ризиками. На результати торгівлі можуть впливати волатильність ринку, технічні проблеми, цінові аномалії та інші зміни. З цієї причини трейдерам необхідно мати чітке уявлення про правила, алгоритми та обмеження сервісу. Крім того, важливо мати надійний план управління ризиками та постійно вдосконалювати свої торгові навички.

Тому автоматизовані послуги криптовалютного арбітражу є важливими, оскільки вони дозволяють трейдерам швидко та ефективно отримувати доступ до інформації, оцінювати величезні обсяги даних і надають можливості для отримання прибутку від різниці в цінах.

Вони допомагають поліпшити процес прийняття рішень і автоматизувати торгівлю, дозволяючи трейдерам зосередитися на управлінні ризиками та плануванні стратегії. Однак, перш ніж користуватися цими послугами, важливо зважити всі переваги та ризики, а також розробити власний торговий план. Автоматизовані сервіси мають потенціал для зростання популярності та ефективності, оскільки технології розвиваються, а трейдери та інвестори продовжують вдосконалювати свої продукти.

1.1.1 Популярність автоматизованих сервісів

Одним із сучасних ринків з найшвидшим зростанням і найбільшою волатильністю є криптовалютна індустрія. Цей ринок пропонує безліч можливостей для трейдерів та інвесторів, але через свою складність і динамічність потребує нових стратегій для заробітку. Ефективне використання можливостей

ринку в цій ситуації забезпечує сервіс криптовалютного арбітражу, який оцінює різницю в цінах на різних біржах і внутрішньобіржових парах.

1. Прибутковість криптовалютного арбітражу

Арбітражна торгівля криптовалютами може принести значні прибутки трейдерам та інвесторам. Через значну волатильність ринку криптовалют часто виникають розбіжності в цінах між біржами та між валютними парами. Криптовалютний арбітраж може швидко і точно виявити ці розбіжності, надаючи трейдерам можливість отримати прибуток, купуючи криптовалюту на одному ринку дешевше і продаючи на іншому ринку дорожче.

Безперебійне виконання транзакцій має вирішальне значення для успіху криптовалютного арбітражу. За допомогою арбітражного сервісу трейдери можуть легко отримати актуальну інформацію про ціни та обсяги торгів на багатьох біржах. Це забезпечує максимальну вигоду від різниці в цінах, запобігаючи затримкам і упущенням в арбітражній діяльності.

2. Ефективність аналізу та виявлення можливостей

З огляду на те, як швидко змінюється ринок, ручне вивчення ринку і пошук арбітражних можливостей може бути дорогим задоволенням. Ефективність пошуку і виявлення арбітражних можливостей значно підвищується за допомогою даного сервісу, який автоматизує процес оцінки цін і обсягів торгів на численних біржах. Це дозволяє трейдерам швидко реагувати на ринкові коливання і здійснювати вигідні угоди вчасно.

Сервіс криптовалютного арбітражу робить обробку величезних обсягів даних простою і швидкою. Ви можете швидко аналізувати дані з багатьох бірж і надавати трейдерам найсвіжішу інформацію про арбітражні можливості за допомогою автоматизованої системи збору та аналізу даних. В результаті трейдери можуть приймати більш точні та ефективні рішення та економити свій час.

3. Ринкові тенденції та розвиток криптовалютного сектору

Криптоіндустрія постійно розширюється і змінюється. З появою все більшої кількості бірж і торгових пар з'являється все більше перспектив для криптовалютного арбітражу. Це сприяє створенню економічно вигідних умов для

трейдерів, які прагнуть отримати прибуток на ринку. Ви можете швидко та ефективно знайти та скористатися цими новими ринковими можливостями за допомогою послуги криптовалютного арбітражу. Трейдери можуть бути в курсі останніх подій на ринку і швидко реагувати на них за допомогою системи аналізу даних.

4. Конкурентний ринок та необхідність інновацій

Арбітраж криптовалют продовжує привертати все більше уваги трейдерів та інвесторів. Оскільки вони пропонують швидкий і простий доступ до можливостей арбітражу, автоматизовані сервіси криптоарбітражу стають все більш популярними. Щоб відповідати очікуванням конкурентного ринку, це мотивує постачальників таких послуг постійно розвивати та вдосконалювати свої пропозиції.

Створення нових функцій і креативних рішень для послуг криптовалютного арбітражу зараз є вирішальними критеріями успіху. Використання передових технологій, таких як штучний інтелект, машинне навчання та блокчейн, може підвищити точність аналізу даних, гарантувати швидкість операцій, а також безпеку та конфіденційність управління активами.

За вище вказаною інформацією зрозуміло, що криптовалютний арбітраж є актуальним для внутрішньобіржових та міжбіржових транзакцій. Завдяки своїй прибутковості, ефективності у виявленні арбітражних можливостей, розширенню криптовалютної індустрії та потребі в інноваціях на ринку, де панує жорстка конкуренція, він є важливим компонентом будь-якого успішного торгового та інвестиційного плану. Трейдери можуть отримувати вигоду від різниці в цінах на різних біржах і внутрішньобіржових парах, використовуючи послуги автоматизованого криптовалютного арбітражу, що дає їм конкурентну перевагу в криптовалютній індустрії, яка постійно розвивається.

Зростаючий інтерес до арбітражу та швидкі зміни на ринку підкреслюють, наскільки важливим є використання автоматизованих сервісів для досягнення успіху в цій галузі. Користувачі послуг криптовалютного арбітражу отримують фінансову вигоду і мають кращі можливості для використання ринкових

можливостей. Завдяки конкурентному клімату та постійному розширенню криптовалютної індустрії, послуги з таких сервісів залишатимуться вирішальними для успішної конкуренції на цьому ринку.

1.2 Масштабування

Масштабування цієї системи є важливим компонентом для її ефективного впровадження та функціонування, тому можливо підкреслити декілька важливих пунктів:

1. Інфраструктура та обчислювальні ресурси

Перш ніж розширювати свою діяльність, необхідно переконатися, що поточна інфраструктура витримає навантаження. Потрібно потужні сервери, надійна мережа та достатня потужність комп'ютерів для ефективної обробки даних.

Спочатку потрібно вибрати правильну архітектуру для серверів. Фізичні сервери або хмарні обчислювальні платформи - обидва варіанти, на які потрібно звернути увагу. Фізичні сервери можуть надати інфраструктурі виняткову продуктивність і контроль, але вони вимагають придбання дорогого обладнання та витрат на підтримку. З іншого боку, хмарні платформи пропонують адаптивність і можливість масштабування ресурсів відповідно до вимог системи. Хоча вони можуть бути більш доступними, можуть виникнути проблеми з безпекою даних і надійністю доступу до сервера. Потреби системи та бюджет визначають, чи використовувати фізичні сервери або хмарні платформи.

Друга вимога - наявність відповідної мережевої інфраструктури. Ефективний обмін даними з біржами та іншими джерелами інформації залежить від швидкої, стабільної мережі. Щоб забезпечити швидкий потік даних між системою та джерелами інформації, дуже важливо підтримувати мережеве з'єднання з низькою затримкою та високою пропускною здатністю.

Крім того, важливо мати достатню обчислювальну потужність для обробки величезних обсягів даних і виконання складних аналітичних завдань. Це може передбачати використання потужних серверів. Крім того, потрібно акцентувати

свою увагу на використання розподілених систем і паралельних обчислень, які дозволяють розподілити обробку даних між численними вузлами та скоротити час для обчислення вхідних даних.

2. Розширення бази даних

З масштабуванням сервісу, потрібно розширювати базу даних. Це тягне за собою вибір правильної архітектури бази даних для забезпечення ефективної та швидкої обробки та архівування даних. Потрібно розглянути кілька факторів, які слід враховувати при масштабуванні бази даних:

1. Використання реляційних баз даних, таких як MySQL або PostgreSQL, є одним з варіантів. Вони зберігають структуровані дані, обробляючи складні операції та пошук. Однак реляційні бази даних можуть стати проблемою, що обмежує продуктивність, оскільки обсяг даних і робоче навантаження зростають. Оскільки нереляційні бази даних, такі як MongoDB або Cassandra, пропонують гнучкість і швидку роботу з великими обсягами даних.

2. Також слід звернути увагу на впровадження розподіленої бази даних або інших розподілених систем, які дозволяють обмінюватися даними між різними вузлами, пропонуючи балансування навантаження з метою розширення сховище даних. Це дозволить підтримувати високу доступність і швидку обробку даних навіть при збільшенні обсягу та навантаження на систему.

3. Автоматизація

Масштабування системи вимагає автоматизації процесів збору та обробки даних. Нам потрібні відмінні алгоритми для автоматичного збору, обробки та виявлення арбітражних можливостей, щоб система працювала належним чином. Використання штучного інтелекту і методів машинного навчання є одним з видів автоматизації. За допомогою цих алгоритмів, які можуть бути нейронними мережами, класифікацією або кластеризацією, система може навчитися виявляти закономірності й тенденції на ринку. Ці методи дозволяють системі швидко виявляти потенційні можливості для арбітражу, автоматично аналізуючи величезні обсяги даних.

4. Безпека

Одним з найважливіших компонентів стратегії отримання прибутку від криптовалютного арбітражу є безпека. Система захищена від хакерських атак і несанкціонованого доступу до даних з урахуванням кібербезпеки.

Безпека може бути забезпечена за допомогою різних методів і технологій, включаючи шифрування даних, аутентифікацію та авторизацію, а також захист мережі та додатків. Щоб розпізнати потенційні ризики та вжити необхідних заходів для запобігання несанкціонованого доступу, необхідно регулярно оновлювати систему та здійснювати моніторинг безпеки.

5. Оптимізація

Масштабування включає важливий етап, який називається оптимізацією системи. Вона дозволяє гарантувати ідеальне використання ресурсів і підвищити продуктивність системи. Для прискорення роботи системи оптимізація може включати зменшення розміру даних та покращення алгоритмів обробки даних, кешування та інші методи.

6. Обсяг даних

Ринок криптоактивів створює велику кількість даних, включаючи інформацію про ціни, торгову активність, технологію блокчейн, новини та соціальні мережі. Модуль збору даних повинен мати можливість швидко оновлювати інформацію в режимі реального часу, ефективно обробляючи та аналізуючи величезні обсяги даних.

Іншим варіантом є використання аналізу та моніторингу продуктивності системи. Це полегшує виявлення потенційних проблем і слабких місць у дизайні системи та впровадження необхідних виправлень.

Масштабування системи для отримання прибутку від внутрішнього та міжбіржового арбітражу криптовалют є складним завданням, але за допомогою відповідної методології та зосередження уваги на ключових факторах його можна реалізувати на практиці. Ключовими компонентами для ефективного масштабування системи є інфраструктура та комп'ютерні ресурси, зростання бази даних, автоматизація, безпека та оптимізація. Враховуючи ці фактори, система буде працювати більш ефективно і надійно.

1.3 Перелік аналогів

На цей час популярними аналогами цього сервісу є cryptorank.io та arby.trade, які являють собою можливості для визначення цінових нерівностей, безпечність роботи з торговими криптоактивами, волатильність, ліквідність та динаміку ціни. Дані веб-сайти можливо порівняти таким чином:

1. Cryptorank.io

Cryptorank.io - це веб-платформа, створена для надання користувачам доступу до різноманітних аналітичних даних про ринок криптовалют. Ця платформа забезпечує користувачам інструменти для аналізу цін, оцінки проектів, отримання новин та іншої корисної інформації.

На головній сторінці проєкту (рис. 1.1) користувач може вибрати криптовалюту, з якою він бажає працювати в контексті арбітражу (рис. 1.2). Це дозволяє користувачам фокусуватись на конкретних криптовалютах та їх можливостях арбітражу.

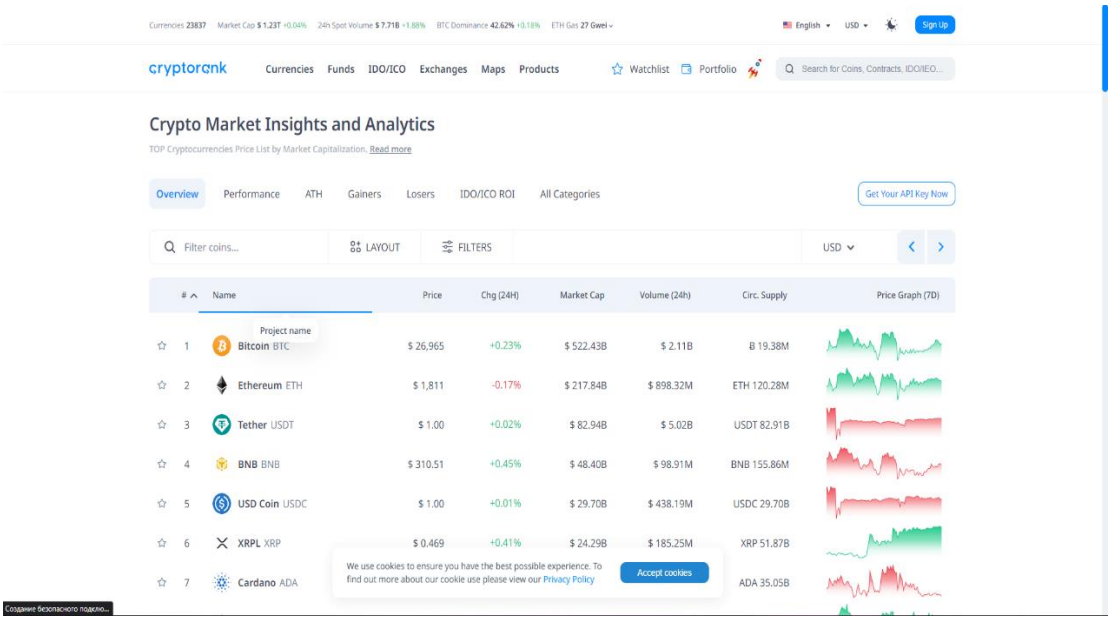


Рисунок 1.1 — Головна сторінка

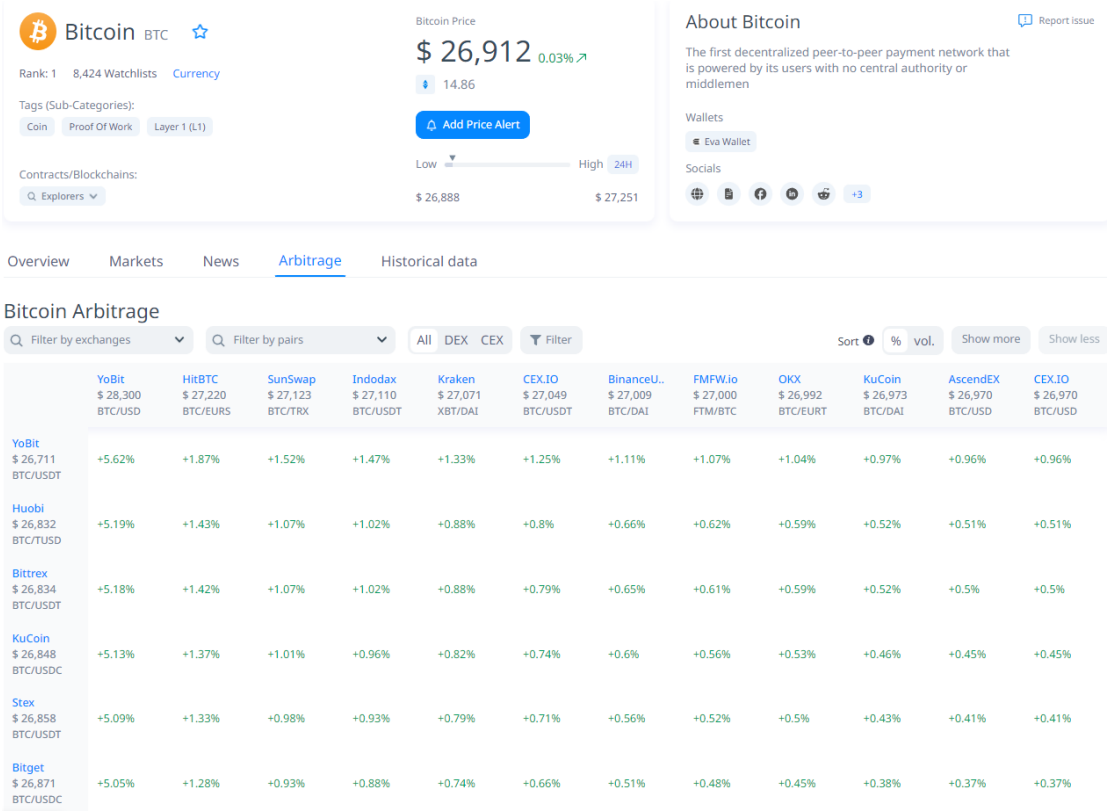


Рисунок 1.2 — Арбітраж Bitcoin

Користувачі веб-платформи Cryptorank.io отримують доступ до низки аналітичних даних щодо ринку криптовалют. Окрім новин та іншої інформації, він пропонує аналіз цін та оцінку проєктів. Інструменти для дослідження та аналізу ринку криптовалют Cryptorank.io можуть бути корисними, але сервіс не слугує для повноцінного заробітку для арбітражу криптовалюти. Даний веб-сайт показує лише теоретичний базис щодо можливостей для арбітражу, тієї чи іншої криптовалюти.

Тому даний сервіс має ряд недоліків, які перешкоджають для повноцінного криптоарбітражу:

1. Велика затримка в оновлені даних: Інформація про ціни та інші показники може оновлюватись з певною затримкою на платформі.
2. Неточність ціни: Cryptorank.io має певну неточність в відображенні цін на криптовалюти.
3. Не враховування об’ємів: Платформа може не враховувати обсяги торгів на різних біржах, що може призвести до неповного аналізу ринку та упущення можливостей для арбітражу.

4. Не враховування ліквідності: При виборі менш ліквідних криптовалют може бути складніше знайти вигідні можливості для арбітражу через Cryptorank.io.

5. Відсутність внутрішньобіржових пар: Cryptorank.io не показує арбітражні можливості між парами криптовалют в межах однієї біржі, що обмежує вибір та потенційний прибуток.

6. Неможливість подивитись перелік найвигідніших зв'язків, а тільки подивитись зв'язки до конкретної монети.

2. Arby.trade

Автоматизований сервіс для криптоарбітражу - Arby.Trade.

Він пропонує ресурси для пошуку і проведення арбітражних угод на численних біржах. Для автоматизації процесу арбітражу та отримання прибутку від різниці в цінах на різних біржах, Arby.Trade використовує алгоритми та технології. З самим сервісом можливо ознайомитись на головній сторінці проєкту (рис 1.3).

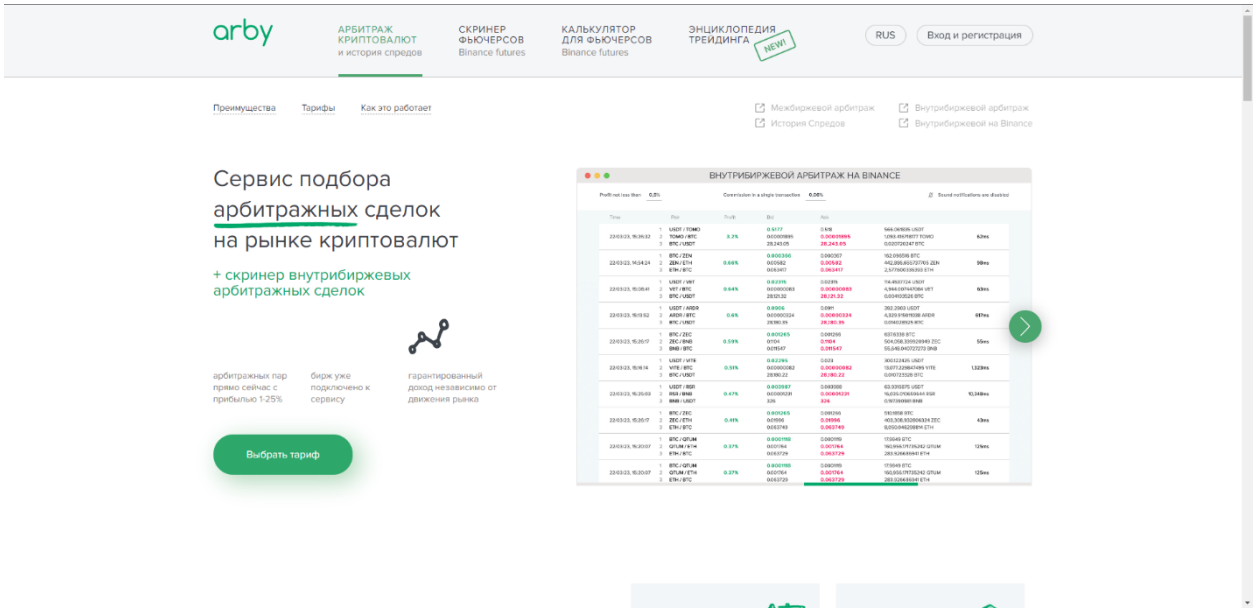


Рисунок 1.3 — Головна сторінка

Цей сервіс призначений для арбітражу криптовалют. На ньому є внутрішньо- та міжбіржові пари, але в даного сервісу існує також безліч недоліків, які напряму впливають на сам процес роботи та користування в загалом:

1. Сервіс недоступний для користувачів України.

2. Ризик змін умов та обмежень: сервіс Arby.Trade залишає за собою право змінювати умови використання або встановлювати обмеження для користувачів у будь-який момент. Це може вплинути на заробіток і можливості арбітражу.

3. Обмежена підтримка бірж.

4. Слабка ліквідність: При використанні менш ліквідних ринків арбітражу, може бути складніше знайти вигідні можливості через Arby.trade.

5. Нестабільний міжбіржовий арбітраж: Це може відбуватися через коливання цін на криптовалюти та затримки в виконанні угод між різними біржами.

3. CoinArbitrageBot

Сайт CoinArbitrageBot.com був створений з метою надання ресурсів і знань, необхідних для проведення криптовалютного арбітражу на різних біржах. Він дозволяє користувачам знаходити та використовувати арбітражні можливості, що з'являються на торговому ринку криптовалют. Користувачі цього веб-сервісу отримують доступ до широкого спектра даних, включаючи біржові ціни криптовалют на різних біржах. Існування розбіжностей в цінах на одну і ту ж криптовалюту на декількох біржах полегшує виявлення можливостей для арбітражу. Для того, щоб ознайомитися з інформацією про торгові пари, необхідно на головній сторінці (рис. 1.4) перейти на вкладку "Arbitrage" (рис. 1.5). Далі вибрати зі списку доступні біржі для аналізу (мал. 1.6).

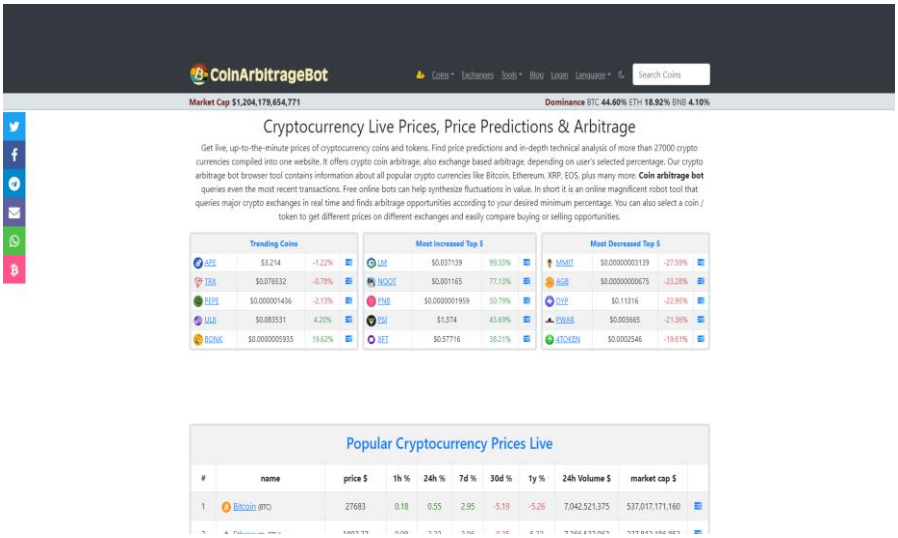


Рисунок 1.4 — Головна сторінка

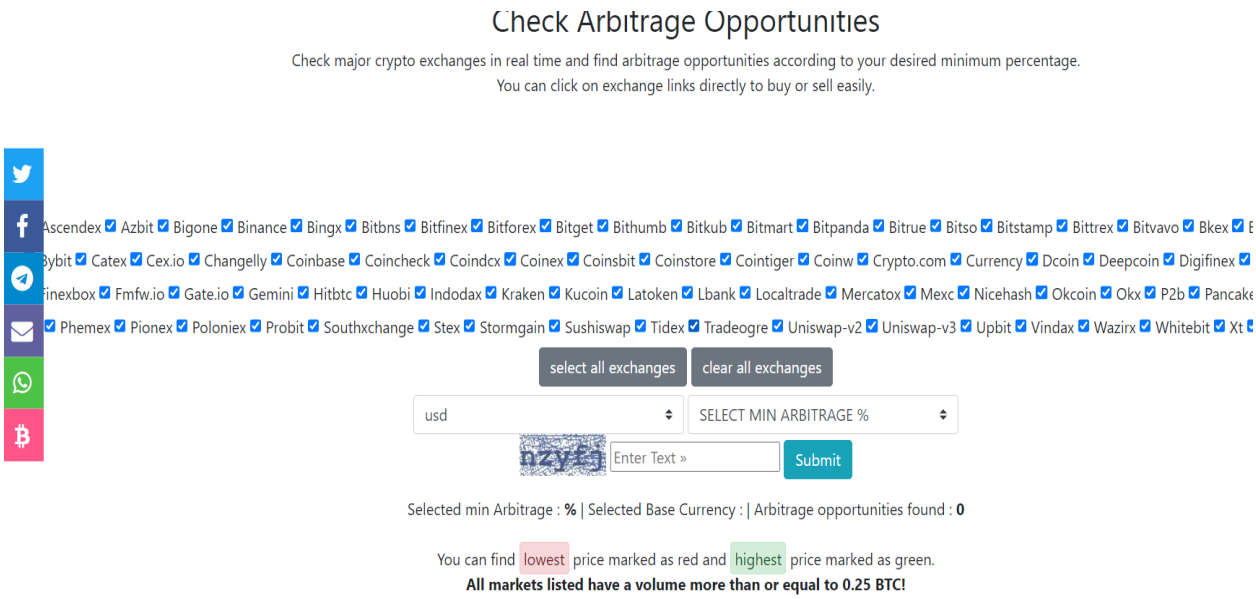


Рисунок 1.5 — Вкладка “Arbitrage”

Coin	ascendex	azbit	bigone	binance	bingx	bitbns	bitfinex	bitforex	bitget	bithumb	bitkub	bitmart	bitpanda	bitrue	bitso
XLM5S/USD	0.21315	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MBS/USD	0	0	0	0	0	0	0	0.025749	0	0	0	0	0	0	0
JUP/USD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
KAI/USD	0	0	0	0	0	0	0	0.0046	0	0	0	0	0	0	0
FRIN/USD	0.001984	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BTCUP/USD	0	0	0	5.585	0	0	0	0	0	0	0	0	0	0	0
RSR3S/USD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
XEC/USD	0.0000251	0	0.00002532	0.00002525	0	0	0	0	0	0.00002546	0	0	0	0.00002528	0
ZAX/USD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TKO/USD	0	0	0	0.3182	0	0	0	0	0	0	0	0	0	0.318	0
SOLO/USD	0	0	0	0	0	0	0	0.13617	0	0	0	0	0	0.1319	0
DPET/USD	0	0	0	0	0	0	0	0.02	0	0	0	0	0	0	0
GRT3L/USD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SAITAMA/USD	0	0	0	0	0	0	0	0	0.0009317	0	0	0	0	0	0
EDU3L/USD	0.46705	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DYDX5L/USD	0.61289	0	0.92744	0	0	0	0	0	0	0	0	0	0	0	0
TRA/USD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Рисунок 1.6 — Інформація по криптовалютнім парам після вибору фільтра

CoinArbitrageBot.com є сервісом розробленим для використання арбітражу криптовалют між різними біржами. Він надає інструменти для моніторингу, однак не є актуальним для користувачів через свої недоліки в оновленні цін та функціоналу.

Загалом, Cryptorank.io не надає прямих міжбіржових арбітражних послуг, а замість цього фокусується на наданні аналітичних даних, які можуть бути корисними для особистих арбітражних рішень. Він допомагає досліджувати та аналізувати ринок криптовалют, але не надає безпосередньо можливостей для виконання міжбіржових арбітражних угод.

Натомість, Arby.trade є автоматизованим сервісом, спеціалізованим саме на криптовалютному арбітражі. Він надає засоби для виконання арбітражних угод на різних біржах за допомогою алгоритмів та технологій. Проте, Arby.trade також має низку недоліків та обмежень, з якими стикаються користувачі, зокрема початківці. Ці недоліки включають обмежену підтримку бірж, проблеми з актуальністю даних та обмежену актуальність проєкту. Крім того, доступ до Arby.trade може бути обмежений для користувачів з певних країн.

Отже, при виборі між Cryptorank.io, Arby.trade та CoinArbitrageBot важливо враховувати їхні особливості та обмеження (табл. 1.7).

	Cryptorank	ArbyTrade	Coinarbitragebot	**Arbity**
Платформи	Web	Web	Windows, Linux	**Web**
Складність використання	Простий	Складний	Складний	**Простий**
Внутрішньобіржовий арбітраж	Відсутній	Присутній	Відсутній	**Присутній**
Міжбіржовий арбітраж	Присутній	Присутній	Присутній	**Присутній**
Інтервал оновлення даних	~10хв	60с	~3хв	**60с**
Врахування ліквідності	-	+	-	**+**
Автоторгівля	Відсутня	Відсутня	Присутня	**Відсутня**

Таблиця 1 — Аналіз аналогів

2 ДОСЛІДЖЕННЯ ІНСТРУМЕНТІВ РОЗРОБКИ

2.1 Мова програмування C#

Компанія Microsoft створила об'єктно-орієнтовану мову програмування C# (C-sharp) у 2000 році. Вона базується на платформі .NET, яка включає Common Language Runtime (CLR), яка забезпечує виконання коду C# на різних платформах, зокрема на Windows. [1]

Windows-додатки, веб-додатки, серверні додатки та інші програми можуть бути створені за допомогою мови програмування C#, яка була розроблена спеціально для платформи Microsoft. Вона підтримує об'єктно-орієнтоване програмування, що дозволяє розділити реалізацію на сутності та надає розробникам безліч можливостей. Мова C# має багато потужних можливостей, серед яких:

1. Об'єктно-орієнтоване програмування: C# дозволяє створювати класи, об'єкти, спадкування, поліморфізм та інші концепції ООП для структурування та організації коду.
2. Збирання сміття (garbage collection): На відміну від C++, C# автоматично зберігає пам'ять і відповідає за утилізацію непотрібних об'єктів за допомогою збирача сміття. Це спрощує розробку та усуває різноманітні проблеми управління пам'яттю.
3. Багатопотоковість: C# має потужні засоби для роботи багатопотокового програмування, які дозволяють розробляти паралельні та асинхронні процеси. Це підвищує продуктивність і задовольняє вимоги сучасних додатків.
4. Обробка винятків C# має методи для витонченого управління винятковими ситуаціями, що дозволяє вам переконатися, що ваш додаток є стійким до збоїв.
5. Великий обсяг бібліотек: C# має велику кількість стандартних бібліотек, які включаються в платформу .NET, в тому числі бібліотеки для роботи

з рядками, колекціями, вводу/виводу, мережами та багато іншого. Існують також сторонні бібліотеки, які розширюють можливості мови та пропонують інструменти розробки для декількох додаткових платформ, включаючи Linux, MacOS, Android, iOS та інші. [2]

Остання версія C# на момент написання дипломної роботи була C# 9.0, яка була випущена разом з .NET 5.0 у листопаді 2020 року.

Історія C# починається у 1999 році, коли Microsoft оголосила про розробку нової мови програмування, яка була спрямована на розробку програмного забезпечення для платформи .NET. C# була розроблена під керівництвом Андерса Хейлсберга і його команди. Перша версія C# (1.0) була випущена разом з платформою .NET у 2002 році. [3]

C# має автоматизовану систему керування пам'яттю, тому розробникам не потрібно вручну виконувати дії з керування пам'яттю, такі як очищення пам'яті (збірка сміття), про яку було сказано раніше. [1] Автоматично визначаючи непотрібні речі, збирач сміття звільняє пам'ять, яку вони займають. Це запобігає численним типовим помилкам керування пам'яттю, таким як джерела пам'яті та неналежне використання звільненої оперативної пам'яті. Також є можливість керувати оперативною пам'яттю вручну. Якщо потрібно більш вільно задавати алгоритм роботи програми, може знадобитися керування пам'яттю.

У мові C# компіляція відбувається у два етапи. Спочатку програма створюється у вихідних файлах, які містять код C#, написаний розробником. Потім використовується компілятор для компіляції цих файлів, створюючи файл проміжної мови (IL) або виконуваний код. Коли програма виконується, IL-файли згодом компілюються в машинний код, або виконуваний код. Цей метод дозволяє запускати програму на будь-якій .NET-сумісній платформі. Однією з переваг такої компіляції є те, що готовий продукт працює швидше і використовує менше пам'яті. Недоліком є те, що, хоча кешування використовується для прискорення роботи програми, при першому запуску вона працює повільніше. [2] [4]

Одним з основних відмінностей між C# і C++ є те, що C# є керованою мовою коду, а C++ - некерованою являється однією з головних відмінностей між ними. У

C# управління пам'яттю автоматизовано, а в C++ за управління пам'яттю відповідає розробник. Асинхронне програмування, делегати, події, LINQ (Language Integrated Query) та інші високорівневі функції включені в C#.

Завдяки впливу C++ на обидві мови, C# та Java мають схожий синтаксис. Проте є декілька відмінностей. C# підтримує властивості (properties), які дозволяють змінювати значення приватних полів через методи доступу get і set. Крім того, лямбда-вирази в C# спрощують створення анонімних функцій. Хоча збір сміття є властивістю обох мов, у C# він є більш досконалим. [2] [5]

Основні бібліотеки C# включаються в платформу .NET і надають широкий набір функціональності для розробки різноманітних застосунків. Деякі з них включають:

1. System: У цій бібліотеці містяться класи і типи для основних системних функцій, таких як керування файлами, введення/виведення, процеси, потоки, мережі та інші системні ресурси.

2. System. Collections: Ця бібліотека пропонує набір структур даних, включаючи списки, стеки, черги, хеш-таблиці та інші, які допомагають в організації та управлінні колекціями даних.

3. System.Linq: Ця бібліотека надає можливість працювати з мовою LINQ (Language Integrated Query), яка дозволяє виконувати структуровані та декларативні операції з даними та запити.

4. System. Threading: Ця бібліотека містить класи для роботи з паралельним програмуванням, синхронізацією потоків, багатопотоковістю та іншими завданнями, пов'язаними з багатопотоковим середовищем.

5. System.Net: Ця бібліотека пропонує класи для взаємодії з мережевими завданнями, включаючи веб-запити, сокети, протоколи та інше.

6. System.IO: Ця бібліотека містить класи для обробки завдань вводу-виводу, таких як взаємодія з файлами, потоками, каталогами та іншими системними ресурсами вводу-виводу.

7. System. Data: Ця бібліотека містить класи для підключення до баз даних, виконання запитів та маніпулювання даними. Вона надає доступ до набору

інструментів для роботи з реляційними базами даних під назвою ADO.NET (ActiveX Data Objects).

8. System. Security: Ця бібліотека містить класи для роботи з безпекою, шифруванням, підписанням даних та іншими питаннями, пов'язаними з безпекою в програмах.

9. System. Web: Ця бібліотека містить класи для створення веб-додатків, зокрема для взаємодії з веб-сервером, обробки запитів і відповідей, керування сеансами, навігації за URL-адресами та виконання інших завдань, пов'язаних з Інтернетом.

10. System. WindowsForms: Використовуючи технологію Windows Forms, ця бібліотека пропонує класи для побудови графічних інтерфейсів користувача. Ви можете створювати вікна, кнопки, тексти, таблиці та інші компоненти інтерфейсу за допомогою цієї бібліотеки.

11. System. Drawing: Ця бібліотека пропонує класи для малювання та взаємодії з графікою, наприклад, для створення малюнків, зміни кольорів, малювання ліній, фігур та інших графічних операцій.

Це лише деякі з основних бібліотек, які пропонує C#. Крім того, значна кількість зовнішніх бібліотек розширює можливості мови та пропонує інструменти для інших сфер розробки, включаючи графіку, обробку даних, мережеве програмування, машинне навчання тощо. [6] [8]

Популярні сторонні бібліотеки C# включають Entity Framework для роботи з базами даних, ASP.NET для створення онлайн-додатків, Xamarin для створення мобільних додатків та Newtonsoft.Json для роботи з форматом JSON та багато інших.

Завдяки широкому спектру можливостей, C# використовується для розробки в найпопулярніших сферах, включаючи веб-додатки, мобільні додатки, десктопні додатки, ігри, системи управління базами даних, хмарні рішення та багато іншого. Це основна мова для платформи .NET, яка має велику спільноту розробників, функціонуючу екосистему та підтримку від Microsoft.

Загалом, C# є потужним інструментом для розробки програмного забезпечення, пропонуючи зручний синтаксис, багату функціональність, великий вибір бібліотек та підтримку великої спільноти розробників. C# є ключовим компонентом платформи .NET, і завдяки своїй універсальності та портативності є однією з найпопулярніших мов програмування в сучасному програмуванні.

2.2 Framework ASP.NET Core

ASP.NET Core - це відкритий веб-фреймворк, розроблений компанією Microsoft. Він був розроблений, щоб забезпечити кращу продуктивність і кросплатформну роботу, тобто функціонувати не тільки на Windows, але й на Linux та інших серверних системах. Він є наступником оригінального фреймворку ASP.NET. [6] [8]

Основна мета ASP.NET Core - надати розробникам інструменти, необхідні для створення передових онлайн-додатків, які можуть бути розгорнуті як на локальних серверах, так і в хмарних інфраструктурах і працювати на будь-якій платформі (Windows, macOS і Linux). Фреймворк пропонує потужні можливості для створення веб-сервісів, мікросервісів, веб-додатків, API та багато іншого. [9]

Основні особливості ASP.NET Core:

1. Кросплатформність: ASP.NET Core може функціонувати на різних операційних системах, що дає розробникам свободу вибору системи, яка найкраще відповідає їхнім потребам.

2. Модульність: Модульна архітектура фреймворку дозволяє використовувати лише необхідні компоненти, що призводить до зменшення розміру проєкту та підвищення його продуктивності.

3. Висока продуктивність: Завдяки оптимізації управління пам'яттю, масштабованості та ефективному використанню ресурсів, ASP.NET Core забезпечує високу швидкість роботи.

4. Підтримка облікових записів користувачів: У фреймворк інтегрована підтримка облікових записів користувачів, включаючи використання файлів

cookie, JWT, OAuth та інших протоколів для аутентифікації та авторизації користувачів.

5. Вбудована підтримка сучасних технологій: ASP.NET Core має вбудовану підтримку сучасних технологій, таких як WebSockets, SignalR (для роботи в режимі реального часу), Web API (для розробки API), Dependency Injection (вбудована підтримка управління залежностями), асинхронність та багатопоточність, що робить його бажаним варіантом для створення сучасних веб-додатків.

6. Простота розгортання: ASP.NET Core дозволяє розгортати веб-додатки за допомогою готових пакетів, які включають всі необхідні залежності можуть бути запущені на різних серверах, включаючи Internet Information Services (IIS), Apache і Nginx.

7. Висока продуктивність: Завдяки оптимізованій обробці запитів та масштабованому дизайну фреймворку ви можете отримати високу швидкість роботи навіть під великим навантаженням. Вбудовані оптимізації дозволяють максимально ефективно використовувати ресурси сервера, використовуючи при цьому менше оперативної пам'яті. [10] [11]

Важливо відзначити наступне порівняння з аналогами, такими як Java та C++:

1. Java: Хоча популярні веб-фреймворки, такі як ASP.NET Core та Java, використовують мову програмування Java, C# не підтримується. Завдяки розгалуженій екосистемі Java та великій кількості бібліотек і фреймворків, її адаптивність гарантована. Перевага ASP.NET Core, з іншого боку, полягає в тому, що вона включає вбудовані інструменти для розробки онлайн додатків, а також доступ до найновіших технологій.

2. C++: є мовою загального призначення, яка надає високий рівень контролю над ресурсами та продуктивністю. Оскільки C++ є мовою нижчого рівня, ніж C#, розробникам доступні додаткові можливості оптимізації та управління пам'яттю. Недоліком цього є те, що розробка на C++ може стати більш трудомісткою, складною та чутливою до безпеки.Остання версія ASP.NET Core на момент написання дипломної роботи (вересень 2021) - ASP.NET Core 6.0. [10] [11]

Зазвичай Microsoft випускає нові версії фреймворку з певною регулярністю, пропонуючи оновлення, нові функціональність та виправлення помилок.

З точки зору історії ASP.NET Core, він був представлений Microsoft у 2014 році як спосіб для програмістів перейти від старого ASP.NET до більш сучасного та модульного фреймворку. У 2015 році була опублікована початкова версія ASP.NET 5. З виходом ASP.NET Core 1.0 у 2016 році з'явилася нова модульна архітектура та кросплатформна підтримка. Згодом були опубліковані версії 2.0, 2.1, 2.2 і 3.0, які принесли з собою вдосконалення і нові функції. Більш детальна інформація про новіші версії може бути недоступною через вік моделі знань та дату останніх змін. [8]

ASP.NET Core надає вбудовану підтримку для реалізації MVC (Model-View-Controller) структури, вона є одним з найпоширеніших шаблонів розробки веб-додатків. MVC дозволяє розподілити логіку програми на три основних компоненти: Модель (Model), Представлення (View) і Контролер (Controller).

1. *Модель (Model)* представляє дані та бізнес-логіку додатка. Вона відповідає за зберігання стану додатку, доступ до даних, валідацію та обробку бізнес-правил. До моделі можуть бути включені класи, структури, бази даних, сервіси та інші елементи, необхідні для обробки даних.

2. *Представлення (View)* відповідає за інформування користувача про інформацію. Воно надає дані моделі браузера в зрозумілому форматі. Як представлення, може використовуватися візуальний інтерфейс, HTML-сторінка, шаблон, JSON-відповідь або будь-який інший спосіб представлення інформації.

3. *Контролер (Controller)* відповідає за обробку запитів, що надходять від користувачів, керування потоками додатка, а також взаємодію з моделлю та представленням. Прийняття запитів, виконання необхідних завдань, доступ до моделі для отримання даних і вибір правильного представлення для показу користувачеві результату - все це завдання, які виконує контролер. [12]

Як результат, фреймворк MVC дозволяє розподіляти завдання між компонентами програми, що покращує контроль і масштабованість. Оскільки

кожен компонент слугує окремій меті, розробники можуть зосередитися на різних компонентах системи незалежно один від одного.

2.3 Середовище розробки

Microsoft розробила Visual Studio як інтегроване середовище розробки (IDE) для розробки програмного забезпечення. Воно пропонує зручний інтерфейс і різноманітні інструменти для створення багатьох типів програм, зокрема настільних, мобільних і хмарних рішень, а також веб-сайтів. [13]

Visual Studio надає програмістам можливість писати код, налагоджувати його, тестувати, аналізувати продуктивність і розгортати програми. Вона підтримує широкий спектр мов програмування, включаючи Python, JavaScript, Visual Basic, F#, C# та багато інших. Крім того, Visual Studio взаємодіє з іншими інструментами та технологіями, включаючи контроль версій, бази даних, хмарні платформи та інші сервіси Microsoft, щоб спростити процес розробки.

Команда Microsoft розробила і створила Visual Studio. З моменту випуску початкової версії у 1997 році Microsoft часто випускала оновлення та нові версії з оновленнями та новими функціями. на момент написання бакалаверської роботи у вересні 2021 року, останньою версією була Visual Studio 2019.

Visual Studio рекомендується для розробки з використанням платформи ASP.NET та мови програмування C#. Щоб спростити розробку на цих платформах, Visual Studio пропонує спеціалізовані шаблони проєктів, засоби налагодження, розширення для підтримки ASP.NET та C#, інтеграцію зі сховищами даних та багато інших можливостей. [14]

Використання Visual Studio для роботи з ASP.NET і C# має кілька переваг:

1. Інтегроване середовище розробки: Visual Studio надає повноцінне інтегроване середовище для розробки на платформі ASP.NET і мові програмування C#. Воно має всі необхідні компоненти, інструменти та налаштування для ефективної розробки додатків.

2. Шаблони проєктів: Для створення веб-додатків на фреймворку ASP.NET Visual Studio пропонує великий вибір шаблонів проєктів. Ці шаблони дозволяють розпочати розробку з нуля або на основі шаблону, автоматично налаштувавши базову структуру проєкту та включивши ключові компоненти.

3. Інструменти налагодження: Для налагодження коду Visual Studio містить безліч ефективних інструментів. Виконання програми можна відстежувати крок за кроком, перевіряти значення змінних, використовувати точки зупинки та багато іншого. Це дозволяє ефективно налагоджувати код і виправляти помилки.

4. Підтримка IntelliSense: Завдяки функції IntelliSense у Visual Studio варіанти коду, методи та властивості автоматично відображаються під час введення. В результаті робота з синтаксисом мови спрощується, а кількість помилок зменшується.

5. Інтеграція зі сховищами даних: Team Foundation Server (TFS) та Git - це дві системи контролю версій, інтегровані з Visual Studio. Це спрощує розподілену розробку, взаємодію з командою розробників та підтримку версій вашого коду.

6. Розширення та плагіни: Visual Studio підтримує розширення і плагіни, які дозволяють розширити функціональність IDE. Існує велика кількість розширень, створених спільнотою розробників, які додають нові можливості, інструменти та підтримку для різних технологій.

7. Підтримка ASP.NET і C#: Спеціалізовані інструменти та функції пропонує Visual Studio для розробки на платформі ASP.NET та мові програмування C#. Це редактор коду з підсвічуванням синтаксису, підтримка IntelliSense, інструменти для створення та підтримки веб-сторінок, використання баз даних, налагодження та інші функції, створені спеціально для цих технологій.

8. Інтеграція з сервером Інтернету Іспанії: Visual Studio має підтримку для інтеграції з сервером Інтернету Іспанії (ASP.NET), що є платформою розробки веб-додатків і веб-сайтів. Використовуючи Visual Studio для роботи з ASP.NET, можливо розробляти веб-додатки з використанням ASP.NET MVC, ASP.NET Web Forms, ASP.NET Web API та інших технологій. За допомогою Visual Studio можна

швидко створювати складні веб-додатки, підтримувати моделі даних та інтегрувати їх з базами даних.

9. Розробка на мові C#: Основним середовищем розробки для мови програмування C# є Visual Studio. Microsoft створила потужну мову програмування C#, яка часто використовується для створення широкого спектра додатків, включаючи настільні, мобільні та веб-додатки. Окрім редактора коду з підсвічуванням синтаксису, IntelliSense, інструментів налагодження коду та підтримки розширень, Visual Studio пропонує повну підтримку C#.

10. Спільна робота та командна розробка: Інструменти для командної розробки та співпраці вже включено до складу Visual Studio. Контроль версій, коментарі, спільний доступ до завдань та інші функції спрощують роботу з іншими розробниками над одним проєктом та забезпечують хорошу комунікацію та співпрацю. [15]

Використання Visual Studio для роботи з ASP.NET і C# дозволяє ефективно розробляти веб-додатки та програми на основі цих технологій. Visual Studio надає багато інших переваг, таких як:

1. Підтримка інструментів тестування: Інструменти тестування коду, такі як модульні тести, функціональні тести та тести продуктивності, вже включені до складу Visual Studio. Це дає змогу на ранніх стадіях знаходити проблеми у вашому додатку та виправляти їх, щоб забезпечити якісний результат.

2. Візуальний редактор і дизайнер інтерфейсу: Візуальні редактори та дизайнери, що постачаються з Visual Studio, дозволяють розробляти та змінювати інтерфейс користувача вашої програми. Без необхідності писати код вручну ви можете легко створювати веб-сторінки, додавати елементи керування, змінювати стилі та інші елементи інтерфейсу.

3. Підтримка розгортання та публікації: Програми можна легко публікувати та розгортати на різних платформах і службах, зокрема на веб-серверах або хмарних платформах Azure. Це спрощує розгортання додатків і пропонує практичні інструменти для керування програмним забезпеченням у реальних умовах.

4. Широкі можливості розширення: Існує велика спільнота розробників, яка працює над різними плагінами та розширеннями для Visual Studio. Додаючи нові інструменти, шаблони проєктів та розширення для роботи зі сторонніми бібліотеками та сервісами, ви можете розширити можливості Visual Studio, спростивши свою роботу та надавши більш індивідуалізоване середовище розробки. [13] [14]

Отже, Visual Studio є потужним і необхідним інструментом для розробки програмного забезпечення, особливо при використанні ASP.NET та C#. Розробники мають доступ до широкого спектра можливостей та інструментів у цьому всеохоплюючому, інтегрованому середовищі, що допомагає підвищити ефективність та якість розробки. Visual Studio швидко перетворюється на найважливіший інструмент для створення додатків на основі ASP.NET і C# завдяки своїм розширеним можливостям розробки, інтегрованим інструментам, підтримці мов програмування та іншим функціям. Вона полегшує проєктування, тестування, налагодження та розгортання веб-додатків та додатків на C#, допомагаючи програмістам створювати ефективне та якісне програмне забезпечення.

2.4 Збір даних

Збір даних в Інтернеті є широким поняттям і охоплює різні методи та технології для отримання інформації з веб-ресурсів. Декілька з них:

1. HTML-парсинг: За допомогою цього методу можна витягувати дані з HTML-сторінок. Для розбору HTML-коду і вилучення необхідних даних з тегів, класів, ідентифікаторів тощо часто використовують бібліотеки або інструменти на кшталт Beautiful Soup.

2. API-запити: Деякі веб-ресурси пропонують програмний інтерфейс, що дозволяє розробникам отримувати доступ до даних серверів безпосередньо. Можливо отримати доступ до структурованих даних через API, які часто повертаються у форматі JSON або XML.

3. Ручний збір даних: При використанні цього методу дані з веб-сторінок необхідно вручну копіювати й вставляти в електронні таблиці або текстові файли. Коли автоматичний збір даних не є ані практичним, ані ефективним, його використовують.

Можливості збору даних з веб-ресурсів:

1. Маркетингові дослідження: Компанії можуть аналізувати ринок, вивчати споживачів, відстежувати конкурентів та отримувати цінну інформацію, збираючи дані з онлайн-ресурсів.

2. Аналіз соціальних мереж: Збір даних з відомих соціальних мереж, таких як Facebook, Twitter та Instagram, дозволяє вивчати поведінку користувачів та досліджувати їхні вподобання, інтереси та погляди. Це може бути корисно компаніям, які хочуть створити цілеспрямовані рекламні кампанії, розробити нові продукти або вдосконалити вже існуючі.

3. Моніторинг цін та конкурентів: Використовується, щоб відстежувати ціни конкурентів, знаходити найкращі пропозиції та аналізувати зміни цін у часі, збираючи дані з веб-сайтів, які продають товари чи послуги. Компанії можуть використовувати цю інформацію для встановлення конкурентних цін і прийняття тактичних рішень.

4. Наукові дослідження та аналітика: Дослідження, аналіз тенденцій і статистика можуть отримати користь від збору даних з онлайн-джерел. Для побудови статистичних моделей, прогнозування тенденцій або вивчення поведінки користувачів дослідники можуть збирати дані з різних джерел.

5. Розвиток програмного забезпечення: Збір даних з веб-ресурсів може бути важливим кроком при розробці програмного забезпечення або веб-додатків. [16]

Дані, зібрані з онлайн-джерел, можна використовувати для створення баз даних, додавання до них інформації або для зв'язку з іншими програмами через API.

Ці методи і технології використовуються для збору інформації з Інтернету з різних джерел і для різних цілей. Їх можна використовувати для різних цілей, включаючи дослідження, аналіз ринку, моніторинг конкуренції, отримання

поточних даних для прийняття управлінських рішень, розробку індивідуальних послуг та багато іншого.

Також варто враховувати та дотримуватись етичні аспекти збору даних з веб-ресурсів. Ось деякі рекомендації, які варто враховувати:

1. Дотримуватись політики конфіденційності: Перед тим, як збирати дані з веб-ресурсу потрібно обов'язково ознайомитись із заявою про конфіденційність на веб-сайті. Щоб не порушувати закон і права інших користувачів, важливо дотримуватися обмежень і вказівок, встановлених власником ресурсу.
2. Використовувати публічно доступні дані: Збирати лише ті дані, які є публічно доступними або які можуть бути отримані через легальні методи. Уникати порушення авторських прав або незаконного доступу до приватних даних.
3. Обмежити частоту запитів: Використовувати розумні інтервали між запитами під час збору даних з веб-сайтів, щоб запобігти надмірному навантаженню на сервери, на яких розміщені ці ресурси. Інтенсивні, повторювані запити можуть призвести до блокування IP-адреси або ускладнити доступ до даних.
4. Документувати джерело даних: Використовуючи дані, завжди потрібно посилатися на них там, де це потрібно. Це дозволить іншим підтвердити точність і надійність даних, які використовуються.
5. Уникати персональної ідентифікації: Уникати збору особистої інформації з веб-сайтів без належного дозволу. Це стосується особистої інформації про користувачів, такої як імена, адреси, адреси електронної пошти тощо. [17]

2.5 Веб-парсинг

Веб-парсинг (web parsing) - це процес автоматичного збору та аналізу даних з веб-сайтів. Він використовується для отримання структурованої інформації з веб-сторінок, такої як текст, таблиці, зображення, посилання, ціни, оголошення, новини й багато іншого. [18]

Взагалі, веб-парсинг використовується для різних цілей, таких як:

1. Збір даних: Веб-парсинг дозволяє автоматично збирати дані з веб-сайтів без необхідності ручного копіювання та вставлення. Це корисно для збору інформації для досліджень, моніторингу конкурентів, скрапінгу новин, отримання фінансових даних тощо.

2. Аналіз даних: Зібрані дані можливо досліджувати, щоб виявити важливі особливості, збирати статистику, проводити бенчмаркінг тощо. Аналізуючи дані про конкурентів, компанії можуть вивчати їхні продукти, витрати, маркетингові стратегії та інші фактори.

3. Моніторинг змін: Веб-парсинг може бути використаний для моніторингу змін на веб-сайтах. Наприклад, електронні магазини можуть парсити ціни конкурентів для автоматичного оновлення своїх цін, або веб-сторінки новин можуть парсити зміни у статтях для надсилання сповіщень про оновлення.

4. Побудова додатків та послуг: Веб-парсинг є важливою технікою для розробки різноманітних додатків та послуг. Наприклад, пошукові системи, агрегатори новин, сервіси моніторингу соціальних мереж тощо базуються на парсингу для отримання та обробки веб-інформації. [19]

У процесі веб-парсингу використовуються різні техніки та інструменти, такі як:

1. HTTP-запити: Програма виконує HTTP-запити до веб-сайтів, щоб отримати вихідний код HTML або інші структуровані дані під час розбору веб-сторінок.

2. HTML-аналіз: Щоб виділити основні елементи, такі як теги, класи, ідентифікатори тощо, вхідний HTML-код обробляється. Для цього можна використовувати регулярні вирази, DOM-дерева або спеціалізовані інструменти для розбору HTML.

3. Розбір JSON або XML: Якщо дані на веб-сторінці представлені у форматі JSON або XML, то вони можуть бути розібрані для отримання необхідних даних. Для цього використовуються бібліотеки або вбудовані засоби для роботи з JSON або XML.

4. Обробка даних: Після отримання даних вони можуть пройти подальшу обробку, фільтрацію або конвертацію. Це може включати видалення зайвих символів, зміну формату, підрахунок статистики тощо.

5. Збереження даних: Отримані дані можуть бути збережені у відповідному форматі, такому як база даних, CSV-файл, текстовий файл або спеціалізований формат для подальшого використання. [20]

Для веб-парсингу використовуються різні техніки, такі як регулярні вирази, DOM-парсинг, XPath або використання спеціалізованих бібліотек та інструментів, таких як BeautifulSoup або Scrapy. Ці інструменти допомагають швидше та зручніше отримувати дані з веб-сторінок і здійснювати подальшу обробку. Ось детальніше про кожну з них:

1. Регулярні вирази (Regular Expressions): Регулярні вирази - це шаблони, які використовуються для пошуку та вилучення певних даних з тексту. Вони дозволяють шукати та витягувати дані на основі заздалегідь визначених критеріїв форматування. Щоб витягти певні рядки, числа, URL-адреси або інші шаблони з HTML-коду веб-сторінки, скористайтеся цією технікою.

2. DOM-парсинг (Document Object Model): Дані витягуються зі структурованого HTML-коду веб-сторінки за допомогою синтаксичного аналізу DOM. DOM, або об'єктна модель документа, - це представлення веб-сторінки браузером. Ви можете отримати доступ до елементів сторінки, витягти їхній вміст, отримати атрибути та дослідити структуру DOM за допомогою синтаксичного аналізу DOM. Цей метод корисний, коли вам потрібно отримати доступ до певних компонентів сторінки, таких як заголовки, посилання або таблиці.

3. XPath: XPath - це мова запитів, яка використовується для перегляду і вибору елементів у XML-подібних документах, включаючи HTML. За допомогою XPath можна явно вибирати елементи на основі їхньої структури, властивостей або значень. За допомогою XPath можна визначити точний маршрут до будь-яких потрібних компонентів сторінки та отримати їхній вміст.

4. Використання бібліотек та інструментів: Процес синтаксичного аналізу веб-сторінок можна спростити за допомогою різноманітних бібліотек та

інструментів. BeautifulSoup, який пропонує зручний інтерфейс для роботи з HTML-кодом веб-сторінок, є одним з найвідоміших інструментів у цій категорії. Ви можете використовувати теги, класи, ідентифікатори та інші властивості для пошуку елементів. [18]

2.6 API

API (Application Programming Interface) - це набір правил та протоколів, які дозволяють різним програмним додаткам взаємодіяти між собою. Вона визначає, як різні компоненти програмного забезпечення повинні спілкуватись, як передавати дані та які операції можна виконувати. [21]

Для того, щоб забезпечити інтеграцію між різними системами та сервісами, використовуються інтерфейси API. Розробники можуть використовувати функції та дані інших програм або веб-сервісів для власних цілей, отримавши доступ до них через API. API надає програмам стандартизовані засоби для обміну даними, що дозволяє їм розширювати свою функціональність і взаємодіяти з іншими системами. [21] [22]

Метод, який використовується для отримання даних, відрізняється між веб-парсингом та API, що є однією з ключових відмінностей. Аналізуючи та витягуючи різні частини з HTML-коду онлайн-сторінок, веб-парсинг використовується для безпосереднього збору даних. З іншого боку, API надає програмному забезпеченню унікальний інтерфейс для отримання даних безпосередньо від власника даних. [22]

Отже, основні переваги використання API порівняно з веб-парсингом:

1. Структуровані дані: API надає дані в структурованому форматі, що полегшує їх обробку та інтеграцію з іншими системами.

2. Автоматизований доступ: API дозволяє отримати доступ до даних безпосередньо інтерфейсу API без необхідності аналізувати та витягувати дані з веб-сторінок.

3. Оновлення в реальному часі: Багато API надають дані в режимі реального часу, що дозволяє отримувати актуальну інформацію миттєво, без необхідності періодичного парсингу веб-сторінок.

4. Документація та підтримка: Більшість API надають детальну документацію та приклади використання, що полегшує їхню інтеграцію. Також, зазвичай, є підтримка розробників, які можуть відповісти на питання та допомогти розв'язати проблеми. [23]

Коли доступ до публічного API неможливий або він має обмежену функціональність чи відображення даних, виникає потреба у використанні веб-парсингу. За таких обставин веб-парсинг дозволяє отримати доступ до більшої кількості даних, які можуть бути присутніми лише на онлайн-сторінках. З іншого боку, веб-аналіз може бути складнішим і вимагати більшого обсягу обробки HTML. [23]

Загалом, веб-парсинг і API доповнюють один одного і можуть використовуватися залежно від вимог конкретного проєкту. Коли доступ до API доступний, це, як правило, швидший і надійніший спосіб отримання даних. Однак веб-парсинг можна використовувати для вилучення необхідних даних з веб-сторінок, якщо доступ до API неможливий або якщо потрібен більш гнучкий спосіб вилучення даних. [24] [25]

Крім того, веб-парсинг може бути використаний для різних сценаріїв, які не обмежені доступом до даних. Деякі з них включають:

1. Моніторинг: Веб-парсинг може бути використаний для постійного моніторингу веб-сторінок і отримання оновленої інформації. Наприклад, коли потрібно парсити ціни товарів на різних веб-сайтах та отримувати повідомлення або зберігати дані, коли ціни змінюються.

2. Агрегація даних: Веб-парсинг дозволяє збирати дані з різних джерел та агрегувати їх в одне місце. Наприклад, коли необхідно зібрати новини з різних новинних веб-сайтів і створити свій власний новинний агрегатор.

3. Аналіз даних: Парсинг веб-сторінок дозволяє отримувати структуровані дані, які можуть бути використані для аналізу та виявлення корисних

залежностей. Наприклад, якщо потрібно зібрати дані про продажі певних товарів з різних інтернет-магазинів та виконати аналіз ринку.

4. Інтеграція зі сторонніми системами: Веб-парсинг дозволяє отримати дані з веб-сторінок та інтегрувати їх зі своєю власною системою або додатком. Наприклад, якщо потрібно парсити інформацію про користувачів з соціальних медіа та імпортувати їх до своєї системи управління користувачами.

5. Машинне навчання та обробка природної мови: Веб-парсинг може бути використаний для збору даних, які можуть бути використані для тренування моделей машинного навчання або обробки природної мови. Наприклад, коли потрібно парсити веб-сторінки з відгуками про товари та використовувати ці дані для побудови моделей аналізу настрою або рекомендаційних систем.

6. Перевірка наявності даних: Web-парсинг використовується для оновлення або перевірки доступності даних на онлайн-сайтах. Наприклад, коли потрібно регулярно перевіряти певні дані, такі як наявність товару або ціни на готельні номери, а потім надсилати сповіщення або вживати необхідних заходів у відповідь на результати парсингу. [21]

Електронна комерція, банківська справа, маркетинг, дослідження та багато інших сфер використовують веб-парсинг. Це дає вам можливість отримати актуальну, багату та впорядковану інформацію з веб-сторінок, яку ви можете використовувати для автоматизації операцій, покращення процесів прийняття рішень та отримання конкурентних переваг.

В API Є кілька переваг порівняно з веб-парсингом: [23]

1. Простота використання: API часто включають документацію, доступ до організованих даних і попередньо визначений набір запитів. В результаті можливо швидко отримати потрібні дані без необхідності створювати власні процедури парсингу.

2. Швидкість та ефективність: API можуть пропонувати спеціалізовані пошукові запити для вилучення потрібних вам даних. Як результат, потрібно надсилати менше даних, а інформація обробляється швидше та ефективніше.

3. Оновлення та стабільність: Розробники часто підтримують та оновлюють API, забезпечуючи стабільну роботу та доступ до найсвіжіших даних. Веб-парсинг дозволяє веб-сторінкам змінюватися, що вимагає регулярних змін у процедурах парсингу.

4. Законність: Доступ до даних за допомогою API може бути більш коректним з моральної та юридичної точки зору. Деякі веб-сайти можуть мати правила або умови надання послуг, які забороняють автоматичний синтаксичний аналіз сторінок.

5. Доступ до спеціалізованих функцій: Деякі API можуть надавати користувачам доступ до спеціалізованих інструментів або послуг, які недоступні через веб-парсер. Наприклад, способи оплати, географічна інформація, обробка зображень тощо. [20] [23]

З огляду на ці переваги, отримання даних з веб-ресурсів за допомогою API є більш практичним, надійним і ефективним методом. Синтаксичний аналіз веб-сторінок все ще залишається ефективним методом, особливо за відсутності доступних API.

2.7 Фронтенд

Фронтенд – це частина веб-розробки, який відповідає за розробку користувацького інтерфейсу веб-сайту або веб-додатку. Все, що користувач бачить і з чим взаємодіє в браузері або на мобільному пристрої, розробляється і реалізується в цій галузі.

Основна мета фронтенду - запропонувати практичний і привабливий користувацький досвід. Це досягається шляхом створення та використання компонентів, включаючи шаблони, стилі, анімацію та функціональність, які сприяють успішній взаємодії користувача з веб-додатком або веб-сайтом.[27]

Основні завдання фронтенд-розробника включають:

1. Розробка інтерфейсу користувача (UI): Фронтенд-розробник створює макети та шаблони, які визначають, як веб-сторінка буде виглядати для

користувачів. Це стосується розташування елементів, кольорової гами, вибору шрифтів та інших естетичних факторів.

2. Розробка користувацького досвіду (UX): Враховує, як користувач буде взаємодіяти з веб-сайтом або веб-додатком. Вони переконуються, що потреби користувача задовольняються, а інтерфейс є простим, зрозумілим та інтуїтивно зрозумілим.

3. Розробка функціональності: Фронтенд-розробник реалізує логіку і функціональність веб-сайту або веб-додатку на практиці, щоб користувачі могли взаємодіяти з різними аспектами. Це може передбачати використання кнопок, форм та інших інструментів.

4. Веб-оптимізація: Покращення швидкості та функціональності веб-додатку або веб-сайту є обов'язком фронтенд-розробника. Щоб гарантувати швидке завантаження веб-сайтів і покращити користувацький досвід, це включає використання найкращих зображень, кешування ресурсів, мінімізацію та об'єднання файлів.

5. Кросбраузерна сумісність: Фронтенд-розробник переконується, що веб-сайт або веб-додаток працює на багатьох платформах і веб-браузерах. Щоб переконатися, що він відображається і функціонує однаково як у широко використовуваних браузерах, так і в менш поширених, вони тестують і модифікують код.

6. Зв'язок з сервером: Фронтенд-розробник полегшує комунікацію між сервером і клієнтом (браузером). Користувачі надсилають запити до сервера, який згодом обробляє отримані дані й відображає їх на веб-сторінках.

7. Тестування та налагодження: Фронтенд-розробник проводить тестування, знаходить і виправляє помилки та вразливості на веб-сайті або в онлайн-додатку. Щоб переконатися, що інтерфейс функціонує належним чином, вони використовують різноманітні інструменти та процедури, включаючи інспектор браузера та тестування на різних пристроях та роздільній здатності екрана.

8. Веб-аналітика: Для збору інформації про поведінку користувачів під час взаємодії з веб-додатком або веб-сайтом фронтенд-розробник встановлює і

використовує аналітичний код. Це дає детальну інформацію про відвідувачів, сторінки, які вони переглядають, дії, які вони виконують, та інші аналітичні дані, які допомагають аналізувати продуктивність і покращувати користувацький досвід.

9. Інтеграція з іншими сервісами: Фронтенд-розробник може інтегрувати веб-сайт або онлайн-додаток з різноманітними іншими сервісами, включаючи системи управління контентом, соціальні мережі та платіжні системи. Це дає можливість розширити функціональність і гарантувати зручність для користувача.

10. Підтримка та оновлення: З часом фронтенд-розробник відповідає за підтримку та оновлення веб-сайту або онлайн-додатку. Це включає в себе модернізацію відповідно до нових стандартів і технологій, усунення помилок і розширення функціональності.

Для того, щоб користувачеві було легко взаємодіяти з веб-додатком або веб-сайтом, використовується інтерфейс, який створює привабливий і функціональний користувальницький інтерфейс. Фронтенд-розробник створює естетично привабливий дизайн, застосовує візуальні ефекти та впорядковує елементи для створення привабливого та інтуїтивно зрозумілого інтерфейсу.[27][28]

2.8 HTML

HTML (HyperText Markup Language) - це різновид мови розмітки, що використовується для створення та представлення веб-сторінок. Визначає семантику і структуру матеріалу на веб-сторінках, включаючи заголовки, абзаци, списки, посилання тощо.

У перші роки існування Всесвітньої павутини Тім Бернерс-Лі вперше представив HTML у 1991 році. Базові елементи розмітки, які дозволили створювати посилання на сторінки, були включені в початкову версію HTML, або HTML 1.0. [29]

З випуском нових версій HTML з часом зростав і розвивався. Остання стабільна версія, HTML5, була опублікована у вересні 2021 року, з моменту

написання цієї бакалаврської роботи. У HTML5 включено багато нових можливостей, зокрема розширена підтримка графіки, геолокації, форм, мультимедійних компонентів (відео та аудіо) та багато іншого. Крім того, він пропонує допомогу в адаптивному дизайні та мобільній розробці. HTML є стандартом, тому W3C (World Wide Web Consortium), організація, яка встановлює стандарти для веб-технологій, постійно оновлює та вдосконалює його. [30]

2.9 CSS

Мова стилів CSS (Cascading Style Sheets - каскадні таблиці стилів) використовується для визначення зовнішнього вигляду та форматування веб-сторінок, створених за допомогою HTML або інших мов розмітки. CSS дає веб-дизайнерам можливість визначати зовнішній вигляд елементів на сторінці, включаючи їхні кольори, шрифти, розміри, відступи, розміщення та інші естетичні характеристики.

Робоча група CSS Консорціуму Всесвітньої павутини (W3C) та Міжнародної організації зі стандартизації (ISO) створила і розвинула CSS. Початковий стандарт CSS, відомий як CSS1, був оприлюднений у грудні 1996 року. CSS2 і CSS3 з'явилися після нього в 1998 і 1999 роках відповідно.

На момент написання цієї бакалаврської роботи CSS3 був останньою стабільною версією, яка була доступна. Набір модулів, відомий як CSS3, розширює можливості CSS і надає користувацькому інтерфейсу нові можливості. Ці уроки охоплюють широкий спектр тем веб-дизайну, включаючи градієнти, медіа-запити, тіні, селектори, анімацію, розміщення та багато іншого. CSS3 також підтримує адаптивний дизайн, який дозволяє веб-сторінкам підлаштовуватися під різні розміри екранів і пристрої. [31]

CSS3 також включає ряд додаткових можливостей, які значно розширюють можливості веб-дизайну. Деякі з цих можливостей включають:

1. Анімація: CSS3 дозволяє створювати кілька анімаційних ефектів без використання JavaScript. Розробники можуть створювати витончені та привабливі

анімовані переходи на веб-сайтах за допомогою ключових кадрів, трансформацій та переходів.

2. Трансформації: Елементи на веб-сторінці можна трансформувати, використовуючи складні можливості CSS3. Можливо змінювати розмір, обертати, зміщувати й нахиляти елементи за допомогою трансформацій. Це покращує взаємодію з користувачем на веб-сайті й дозволяє створювати інтригуючі візуальні ефекти.

3. Гнучке розміщення елементів: CSS3 відкриває нові можливості для точного позиціювання елементів на веб-сторінці. Нові функції, такі як Flexbox і Grid, дозволяють розробникам легко налаштовувати розташування елементів у веб-документі. Це усуває проблеми з попередніми методами позиціонування і спрощує розробку адаптивних макетів.

4. Тіні та градієнти: CSS3 усуває вимогу до зображень створювати складні тіні та градієнти. Нові атрибути, такі як box-shadow і linear-gradient, дозволяють розробникам створювати потужні градієнти та тіні, які покращують зовнішній вигляд компонентів сторінки.

5. Медіа-запити: CSS3 дозволяє задавати стилі, які будуть використовуватися відповідно до різних характеристик пристрою або екрану. Це дозволяє розробляти адаптивні веб-сторінки, які адаптуються до розміру екрана, типу пристрою та інших факторів, змінюючи свій вигляд і поведінку.

6. Шрифти та текстові ефекти: CSS3 пропонує складні інструменти для зміни вигляду тексту на веб-сторінках. Розробники можуть створювати текст і застосовувати ефективні тіні, використовуючи такі атрибути, як @font-face, text-shadow і text-overflow.

7. Додаткові можливості: CSS3 також має ряд інтригуючих можливостей, таких як можливість працювати з анімованими переходами, численні фони, адаптивні межі, псевдоелементи і псевдокласи, розширене форматування таблиць та багато іншого. [32]

Важливо пам'ятати, що CSS3 - це набір вимог, і не кожен браузер може реалізувати всі його можливості. Проте, велика кількість сучасних браузерів

підтримує більшість основних можливостей CSS3, що дозволяє програмістам використовувати їх для створення естетично привабливих і цікавих онлайн-інтерфейсів.

2.10 JavaScript

Високорівнева інтерпретована мова програмування JavaScript (JS) використовується для створення динамічного контенту для веб-сайтів. Вона була розроблена для спілкування з користувачами та управління поведінкою веб-сторінки на стороні клієнта.

Брендан Айх розробив JavaScript у 1995 році, коли працював у компанії Netscape. У 1997 році версія мови ECMAScript 1 (ES1), яка є оригінальною, була стандартизована. Офіційна специфікація JavaScript, відома як ECMAScript, описує синтаксис і можливості мови.

ECMAScript 3 (ES3), ECMAScript 5 (ES5), ECMAScript 6 (ES6) та інші версії - це лише деякі з численних еволюцій, яких зазнав JavaScript за ці роки. Кожна наступна версія надає мові більше можливостей, покращує синтаксис і надає розробникам більше контролю та свободи.

Остання версія JavaScript на момент написання цієї бакалаврської роботи була опублікована у вересні 2021 року і називається ECMAScript 2022 (ES2022). Оператор matchAll(), рядкові методи, що дозволяють використовувати регулярні вирази, функції, пов'язані з пропорціями, оператори порівняння множин та інші нововведення також є частиною цієї версії. [33]

Поява динамічних веб-сторінок, односторінкових додатків (SPA), мобільних додатків, десктопних програм, інтерактивних ефектів, веб-ігор та деяких інших додатків зробила JavaScript найважливішим компонентом веб-розробки.

Здатність JavaScript взаємодіяти з HTML і CSS є однією з його важливих особливостей; це дозволяє створювати динамічний інтерфейс і миттєво змінювати вміст веб-сайтів без необхідності перезавантажувати сторінку. JavaScript дає вам можливість керувати подіями, змінювати стилі, працювати з DOM (Document

Object Model) та взаємодіяти з сервером, що дозволяє створювати веб-додатки з різноманітними функціями.

Платформа Node.js також підтримує JavaScript для програмування на стороні сервера. Виконання JavaScript на сервері стає можливим завдяки Node.js, що значно розширює ваші можливості для розробки серверних додатків, API та функціональності, пов'язаної з базами даних. [34]

Однією з головних переваг JavaScript порівняно з PHP є його універсальність і широкі можливості в розробці. JavaScript може використовуватись як на стороні клієнта, так і на стороні сервера за допомогою платформи Node.js.

JavaScript є основною мовою для створення динамічних веб-сторінок і веб-додатків. З його допомогою можна змінювати вміст сторінки без перезавантаження, взаємодіяти з користувачем, керувати подіями та змінювати стилі.

Завдяки Node.js, JavaScript може бути використаний на стороні сервера для розробки серверних додатків, API і роботи з базами даних. Це дає розробникам можливість використовувати одну мову програмування як на клієнтській, так і на серверній стороні, що спрощує розробку та підтримку проектів.

JavaScript також має велику активну спільноту розробників і велику кількість доступних бібліотек і фреймворків, що сприяє швидкому розвитку проектів і полегшує роботу розробників.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКА

3.1 Опис етапів розробки програмного забезпечення

Процес розробки програмного забезпечення для системи підтримки торгівлі криптовалютами можна поділити на наступні етапи:

1. Опис вимог: Початковим кроком у процесі розробки програмного забезпечення є збір вимог до системи. Цей етап включає дослідження потреб і вимог потенційних користувачів.

2. Проєктування: Після збору вимог наступним кроком є проєктування системи. На цьому етапі розробляються високорівневий та низькорівневий дизайн системи. Високорівневий дизайн описує загальну структуру системи, включаючи функціональні модулі та їх взаємозв'язки. Низькорівневий дизайн, фокусується на конкретних деталях реалізації модулів.

3. Розробка: Після завершення проєктування іде перехід до етапу розробки, де реалізуються всі необхідні компоненти програмного забезпечення: написання коду, розробку інтерфейсу користувача та інші технічні аспекти.

4. Тестування: Після завершення розробки програмного забезпечення виконується його тестування для виявлення та усунення можливих помилок, багів і недоліків. Цей етап включає як ручне, так і автоматизоване тестування системи.

3.1.1 Опис вимог до програмного забезпечення

Перед початком розробки програмного забезпечення було встановлено основні вимоги до системи, яка підтримує торгівлю криптоактивами. Ці вимоги були визначені на підставі докладного аналізу потреб користувачів та вимог ринку. Основні вимоги до програмного забезпечення включають:

1. Автоматизувати збір даних з бірж.
2. Автоматизувати агрегацію даних.
3. Візуалізувати результат модуля аналізу даних.
4. Реалізувати фільтр торгових сигналів.

5. Висока швидкодія для забезпечення оперативності.
6. Стабільність системи для безперебійності роботи.

3.2 Реалізація

3.2.1 Автоматизований збір даних з бірж за допомогою бібліотек API та парсингу

Модуль збору даних складається з різних об'єктів, а не є єдиною цілісною одиницею. Кожна біржа представлена окремим об'єктом збору даних ([Exchanger name]DataManager), який узагальнюється за допомогою інтерфейсу IExchanger. Такий розподілений підхід був необхідний, оскільки кожна біржа має власний спосіб взаємодії з клієнтом, зображено на рисунку 3.1.

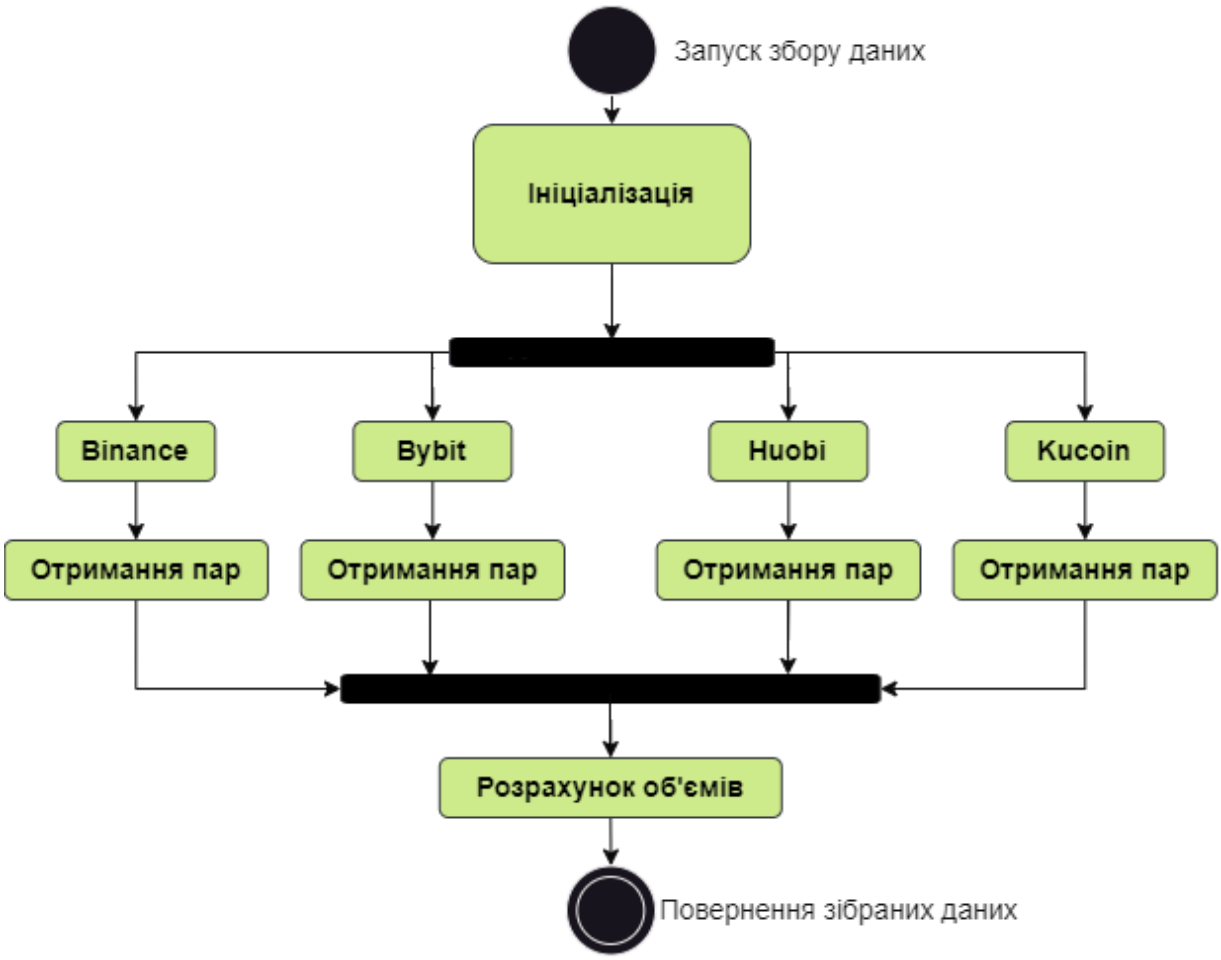


Рисунок 3.1 — Діаграма діяльності

На цей час, біржі, які підключені до проєкту, реалізовані через використання бібліотек, які взаємодіють з API цих бірж. На початку, з бази даних отримується список бірж та список торгових пар, які передаються кожному біржовому менеджеру. Після цього відбувається асинхронний етап ініціалізації у кожному менеджері, де встановлюється з'єднання з біржею за допомогою протоколу WebSocket. WebSocket є протоколом, призначеним для обміну інформацією між браузером та веб-сервером в режимі реального часу, що дозволяє в цьому випадку переглядати замовлення в реальному часі. Для прискорення підключення до бірж, кожна торгова пара завантажується паралельно.

Після успішної ініціалізації, програма переходить до етапу моніторингу даних з певним інтервалом. На цьому етапі формується загальний список торгових пар, який передається в модуль обробки та аналізу даних для подальшої обробки.

Демонстрація збору інформації з біржі Binance, пара BTC/USDT. В консолі зображений об’єм торгів за 24 години та відкриті замовлення на продаж чи купівлю (рис. 3.2).

```
BTCUSDT  Buy:27146,59$   Sell:27146,58$  Volume: 26619,41063000BTC QuoteVolume: 721813205,83$
Orders:
Buy:27146,59$ | 2,90627000BTC    Sell:27146,58$ | 5,68331000BTC
Buy:27146,60$ | 0,07400000BTC    Sell:27146,56$ | 1,80055000BTC
Buy:27146,61$ | 0,22392000BTC    Sell:27146,55$ | 0,10000000BTC
Buy:27146,63$ | 0,01576000BTC    Sell:27146,51$ | 0,21743000BTC
Buy:27146,94$ | 0,00043000BTC    Sell:27146,47$ | 1,16820000BTC
Buy:27147,01$ | 0,02900000BTC    Sell:27146,42$ | 0,09207000BTC
Buy:27147,19$ | 0,00037000BTC    Sell:27146,41$ | 1,41769000BTC
Buy:27147,25$ | 0,00043000BTC    Sell:27146,33$ | 0,00040000BTC
Buy:27147,26$ | 0,00690000BTC    Sell:27146,31$ | 0,10000000BTC
Buy:27147,56$ | 0,00043000BTC    Sell:27146,14$ | 0,10000000BTC
```

Рисунок 3.2 — Демонстрація збору інформації з біржі Binance, пара BTC/USDT

3.2.2 Вузол агрегації даних отриманих з бірж

Вузол агрегації даних, який є складовою частиною модуля збору даних, складається зі списку об'єктів DTO (Data Transfer Object). Ці об'єкти зберігають інформацію про торгову пару і є універсальними для всіх менеджерів бірж. Основним об'єктом є TradingPair.

Після отримання даних, вузол розпочинає процес завантаження цих даних в об'єкт TradingPair. Крім того, відбуваються вторинні розрахунки, такі як обчислення ціни та об'єму, а також інші обчислення. Після завершення цього процесу, об'єкт TradingPair, додається до списку, що складається з подібних об'єктів. Структурна схема зображена на рисунку 3.3.

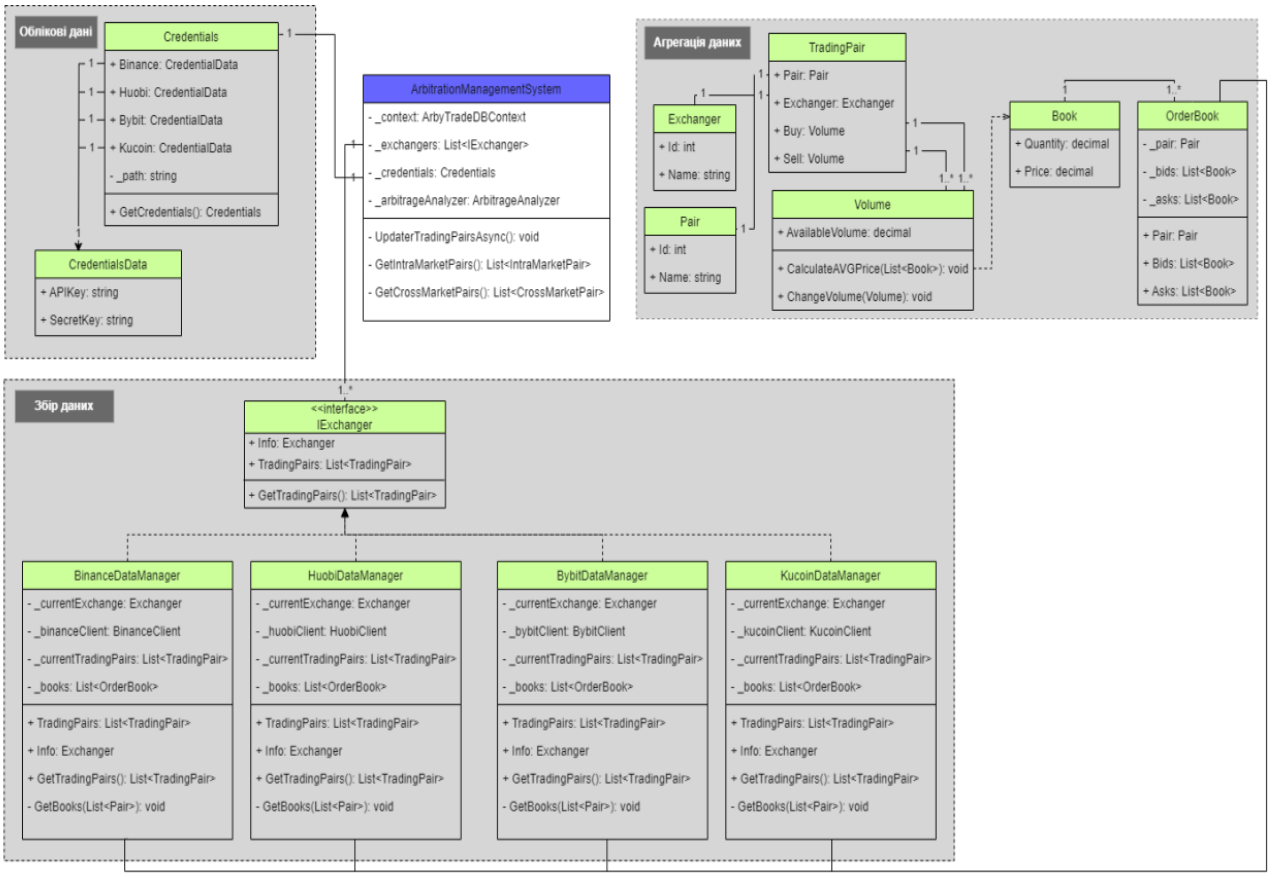


Рисунок 3.3 — Діаграма класів

Таким чином, вузол агрегації даних виконує важливу роль у модулі збору даних. Він отримує дані, перетворює їх у відповідний формат об'єкта TradingPair і проводить необхідні розрахунки. Цей вузол створює єдиний список об'єктів TradingPair, який використовується для подальшої обробки та аналізу даних.

На рисунку 3.4 продемонстровано агрегацію даних, а саме вивід даних з об’єкта TradingPair.

```
BTCUSDT Binance
Buy: price:27203,79 volume:5020527,2736965000000000
Sell: price:27098,09 volume:5038812,6942886000000000
```

Рисунок 3.4 — Агрегація даних, а саме вивід даних з об’єкта TradingPair

3.2.3 Розробка сайту

1. Створення макета сторінок: макет сайту передбачає декілька сторінок з різними розділами та елементами дизайну. Кожна сторінка створюється з використанням HTML та CSS, враховуючи дизайн, колірну схему (синя стилістика). Макет дизайну продемонстрований на рисунку 3.5.

Рисунок 3.5 — Макет дизайну

2. Використання серверного коду: ASP.NET дозволяє використовувати серверний код для динамічного генерування контенту та взаємодії з базою даних. Використовувати мову програмування C# для обробки запитів користувачів та відображення даних на сторінці (рис. 3.6).

Рисунок 3.6 — Схема веб-додатка

3. Валідація та обробка форм: ASP.NET дозволяє використовувати вбудовані інструменти для валідації введених даних та обробки форм. Це допомагає забезпечити коректну та безпечну обробку вхідних даних від користувачів.

4. Доступ до сторінок та безпека: Використано механізми для керування доступом до сторінок та забезпечення безпеки. Використано аутентифікації та авторизації.

3.2.3.1 Візуалізація результату модуля аналізу криптовалютних пар

Для представлення торгових сигналів і їх візуалізації на сторінці веб-сайту була використана таблиця, демонстрація вигляду таблиці зображення на рисунку 3.7.

Пара	Прибуток	Біржа	Купити	Продати	Об'єм USDT	10хв	1год
KAVA/USDT	11,43%	Binance -> Huobi	1,181	1,3160	33393	+0,16%	-0,93%
DOGE/BTC	6,94%	Huobi -> Binance	0,00000259	0,00000277	17440	-0.56%	-0.2%
TRX/USDT	5,1%	Huobi -> Binance	0,07681	0,0807273	23679	-0,6%	-0,5%
NEAR/USDT	2,64%	Huobi -> Binance	1,6212	1,664	10305	-0,06%	-0,49%
ZIL/USDT	2%	Binance -> Huobi	0,02342	0,2389	6305	-0,06%	-0,49%
MAGIC/USDT	1,75%	Huobi -> Kucoin	0,9309	0,9472	11670	+0,16%	-0,1%

Рисунок 3.7 — Демонстрація вигляду таблиці

Ця таблиця містить обмежену кількість даних, а саме лише 10 сигналів відображаються одночасно на сторінці. Щоб переглянути інші сигнали, користувач має можливість використовувати навігацію по сторінках.

Такий підхід до відображення даних дозволяє забезпечити зручну навігацію для користувача та уникнути перенасиченості таблиці великою кількістю сигналів. Відображаючи лише часткову кількість сигналів на сторінці, сайт забезпечує оптимальне сприйняття інформації користувачем і полегшує його орієнтацію.

Завдяки навігації по сторінках, користувач може легко переміщатись між різними наборами сигналів і переглядати їх поетапно. Це забезпечує зручну інтерактивну можливість огляду інформації.

Такий підхід до візуалізації торгових сигналів забезпечує ефективну інтерактивну взаємодію з користувачем, дозволяючи йому швидко переглядати та аналізувати обмежений набір даних на кожній сторінці та здійснювати легкий перехід між сторінками для огляду більшої кількості сигналів.

3.2.3.2 Фільтр торгових сигналів

На сторінці присутній фільтр для підбору торгових сигналів за параметрами. Цей фільтр надає користувачу можливість точно визначити критерії, за якими він хоче здійснити відбір сигналів.

Функціонал фільтрації дозволяє обрати різні параметри для відображення відповідних сигналів. Наприклад, користувач може встановити фільтр за часовим періодом, щоб показати лише сигнали, отримані в певному діапазоні дат або інтервалі часу. Також, можливо встановити фільтри за торговими парами, щоб показати лише сигнали, пов'язані з конкретними валютними парами або іншими фінансовими інструментами.

Параметри фільтра також можуть включати інші аспекти, наприклад, типи торгових сигналів (купівля або продаж), рівень ризику, рекомендації від аналітиків чи показники технічного аналізу. Користувач може вибрати один або декілька параметрів для фільтрації даних.

Після застосування фільтра, на сторінці будуть відображені лише ті сигнали, які відповідають обраним параметрам. Це дозволяє зменшити обсяг інформації та показати лише ті сигнали, які користувач вважає важливими або цікавими.

Фільтр для підбору сигналів за параметрами надає зручний і персоналізований спосіб взаємодії з великим обсягом даних про торгові сигнали. Користувач може точно визначити свої вимоги та отримати лише ті сигнали, які відповідають його уявленням про вигідні торговельні можливості.

3.3 Тестування продукту

Тестування модуля збору даних та самого сайту є важливою частиною процесу розробки, що дозволяє перевірити, чи працює програмний продукт належним чином та відповідає очікуванням.

Крім того, важливим аспектом тестування модуля збору даних є тестування на реальних даних. Це означає, що модуль випробовується з реальними даними, які

надходять з різних бірж. Це дозволяє перевірити, чи правильно обробляються та зберігаються дані, а також чи працює з'єднання через WebSocket.

Тестування самого сайту також має свої етапи. Перш за все, проводиться функціональне тестування, де перевіряється робота основних функцій сайту, таких як відображення даних, робота фільтрації, навігація по сторінках та інші.

Також, проводиться тестування коректності відображення на різних пристроях та браузерах.

В результаті фінального тестування, помилок не виявлено.

ВИСНОВКИ

Метою даної роботи стало розроблення модуля агрегації даних мовою C# для збору, об'єднання даних та обробки інформації криптовалютного ринку.

Для досягнення поставленої мети в процесі виконання роботи наступні завдання:

1. Розроблено модуль агрегації даних, що дозволяє автоматизувати збір даних з торгового ринку криптовалют.
2. Обґрунтовано використання технологій .NET та ASP.NET, що дозволяє розробити зручну та масштабовану серверну систему.
3. За результатами проведеного дослідження розроблена підсистема даних для стандартизації обчислень.
4. Розроблена система збору даних за допомогою API та парсингу.
5. Розроблено для фронтенду дизайн, форми та автентифікація.
6. Розроблено фільтр результатів по параметрах.
7. Проведено автоматизоване тестування програмного продукту, що дозволило усунути помилки.

В результаті успішного виконання завдань, що були поставлені на початку проєкту, було створено повноцінну серверну частину для інформаційної системи збору даних криптоактивів, яка готова до використання в реальних умовах та взаємодії з інтерфейсом користувача.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. C# (C-Sharp). [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/whatis/definition/C-Sharp

2. Парфьонов Ю. Е. Федорченко В. М. Лосєв М. Ю. Щербаков О. В. ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ: конспект лекцій. Харків: Вид. ХНЕУ, 2010. 314 с. Режим доступу: https://www.pdau.edu.ua/sites/default/files/node/4518/pravylaoformlennyaspyskuvykorystanyhdzherel.pdf

3. C Sharp (programming language). [Електронний ресурс]. – Режим доступу: https://codedocs.org/what-is/c-sharp-programming-language

4. Коноваленко І.В., Марущак П.О., Савків В.Б. Програмування мовою C# 7.0: навчальний посібник. Тернопіль: 2017. 302 с. – Режим доступу: https://elartu.tntu.edu.ua/bitstream/lib/22436/1/Konovalenko_I._Programuvannya_C%23_2017.pdf

5. Основи ООП. Базовий синтаксис мови C#. [Електронний ресурс]. – Режим доступу: http://iwanoff.inf.ua/oop_ua/LabTraining01.html

6. The Good and the Bad of .NET Framework Programming. [Електронний ресурс]. – Режим доступу: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-net-framework-programming/

7. Andrea Chiarelli. What is .NET? An Overview of the Platform. [Електронний ресурс]. – Режим доступу: https://auth0.com/blog/what-is-dotnet-platform-overview/

8. .NET Core Framework Complete Review. [Електронний ресурс]. – Режим доступу: https://www.instinctools.com/blog/net-core-framework-complete-review/

9. Т.А. ВАКАЛЮК. Хмарні технології в освіті: посібник. Житомир: 2016. 72 с. Режим доступу: https://lib.iitta.gov.ua/706333/1/%D0%9F%D0%BE%D1%81_%D0%A5%D0%A2%D0%9E.PDF

10. Introduction to .NET and .NET Core Framework. [Електронний ресурс]. – Режим доступу: https://www.interviewbit.com/dot-net-interview-questions/

11. Andrew Lock | .NET Escapades. [Електронний ресурс]. – Режим доступу: https://andrewlock.net/getting-started-with-asp-net-core/

12. Vlad Ostrenko. Comparing Three-Layered and Clean Architecture for Web Development. [Електронний ресурс]. – Режим доступу: https://betterprogramming.pub/comparing-three-layered-and-clean-architecture-for-web-development-533bda5a1df0

13. Visual Studio. [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Visual_Studio

14. Kolade Chris. Visual Studio vs Visual Studio Code. [Електронний ресурс]. – Режим доступу: https://www.freecodecamp.org/news/visual-studio-vs-visual-studio-code/

15. Anshul Aggarwal. Introduction to Visual Studio. [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/introduction-to-visual-studio/

16. Research ethics and data protection. [Електронний ресурс]. – Режим доступу: https://www.reading.ac.uk/research-services/research-data-management/data-management-planning/research-ethics-and-data-protection

17. 3 key steps to ethical data collection by Paul Clough. [Електронний ресурс]. – Режим доступу: https://www.tpximpact.com/knowledge-hub/insights/ethical-data-collection/

18. Web Scraping. [Електронний ресурс]. – Режим доступу: https://scrape-it.cloud/blog/web-scraping-what-it-is-and-how-to-use-it

19. Ansel Barrett. Web Scraping Techniques and Tools. [Електронний ресурс]. – Режим доступу: https://www.octoparse.com/blog/introduction-to-web-scraping-techniques-and-tools

20. Songhao Wu. Web Scraping Basics. [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/web-scraping-basics-82f8b5acd45c

21. API. [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/API

22. Application programming interface (API). [Електронний ресурс]. – Режим доступу: https://golden.com/wiki/Application_programming_interface_(API)-DAXJJ4

23. Web Scraping vs API. [Електронний ресурс]. – Режим доступу: https://www.zenrows.com/blog/web-scraping-vs-api

24. API Testing Approaches and Tools. [Електронний ресурс]. – Режим доступу: https://www.altexsoft.com/blog/api-testing/

25. Web Scraping vs API: Best Way to Get Data. [Електронний ресурс]. – Режим доступу: https://scrapeops.io/blog/the-state-of-web-scraping-2022/

26. What is Front-End Web Development? [Електронний ресурс]. – Режим доступу: https://www.trio.dev/blog/front-end-web-development

27. Front-End Development: Key Technologies and Concepts. [Електронний ресурс]. – Режим доступу: https://www.altexsoft.com/blog/front-end-development-technologies-concepts/

28. THE DIFFERENCE BETWEEN UI & FRONTEND DEVELOPERS. [Електронний ресурс]. – Режим доступу: https://pagepro.co/blog/difference-between-ui-frontend-developers/

29. HTML (Hypertext Markup Language). [Електронний ресурс]. – Режим доступу: https://pagepro.co/blog/difference-between-ui-frontend-developers/

30. HTML vs HTML5: Core Differences. [Електронний ресурс]. – Режим доступу: https://www.browserstack.com/guide/html-vs-html5

31. Importance of CSS in Web Development. [Електронний ресурс]. – Режим доступу: https://www.antino.com/blog/importance-css-web-development/

32. CSS Transitions and Animations. Motion Path Module CSS. [Електронний ресурс]. – Режим доступу: https://stfalcon.com/en/blog/post/animation-css

33. What is JavaScript? [Електронний ресурс]. – Режим доступу: https://www.scaler.com/topics/javascript/what-is-javascript/

34. Web App Development. [Електронний ресурс]. – Режим доступу: https://www.trio.dev/blog/web-app-development

ДОДАТКИ

Додаток А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення системи підтримки торгівлі криптоактивами. Спец частина: Розробка модулю агрегації даних мовою C#

Виконав студент 4 курсу
групи ПД-44
Столяр Павло Юрійович
Керівник роботи
Д.т.н, доцент, завідувач кафедри Технологій цифрового розвитку Жебка Вікторія Вікторівна

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи -** автоматизація збору даних з торгового ринку криптовалют за допомогою модулю агрегації даних мовою C#.
- **Об’єкт дослідження -** збір інформації з бірж через API та парсинг.
- **Предмет дослідження -** модуль агрегації даних мовою C# для збору, обробки та об'єднання даних.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати предметну область.
2. Проаналізувати існуючі аналоги.
3. Під’єднати збір даних за допомогою API.
4. Розробити підсистему даних для стандартизації обчислень.
5. Провести тестування модуля збору даних та агрегації даних.
6. Розробити для фронтенду дизайн, форми та авторизацію.
7. Розробити фільтр результатів по параметрах.
8. Провести тестування веб-додатку на адаптивність.

3

АНАЛІЗ АНАЛОГІВ

	Cryptorank	ArbyTrade	Coinarbitragebot	**Arbity**
Платформи	Web	Web	Windows, Linux	**Web**
Складність використання	Простий	Складний	Складний	**Простий**
Внутрішньобіржовий арбітраж	Відсутній	Присутній	Відсутній	**Присутній**
Міжбіржовий арбітраж	Присутній	Присутній	Присутній	**Присутній**
Інтервал оновлення даних	~10хв	60с	~3хв	**60с**
Врахування ліквідності	Немає	Є	Немає	**Є**
Автоторгівля	Відсутня	Відсутня	Присутня	**Відсутня**

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні:

1. Автоматизувати збір даних з бірж.
2. Автоматизувати агрегацію даних.
3. Візуалізувати результат модуля аналізу даних.
4. Реалізувати фільтр торгових сигналів.

Нефункціональні:

1. Висока швидкодія для забезпечення оперативності.
2. Стабільність системи для безперебійності роботи.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Kucoin.net

Bybit.net

Huobi.net

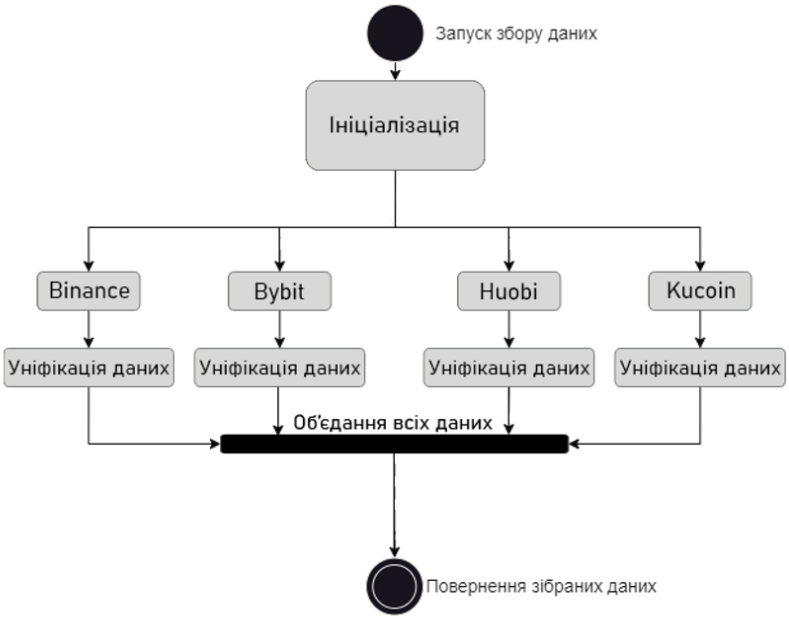
Binance.net

6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

7

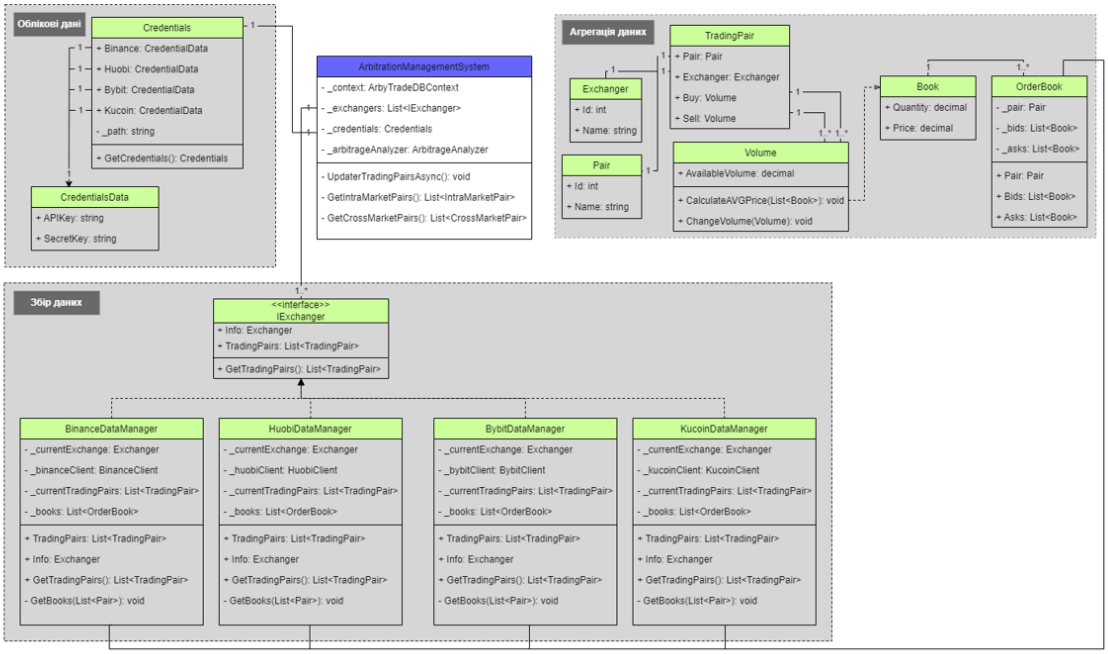
ДІАГРАМА ДІЯЛЬНОСТІ



Збір та стандартизація даних через парсинг даних з бірж

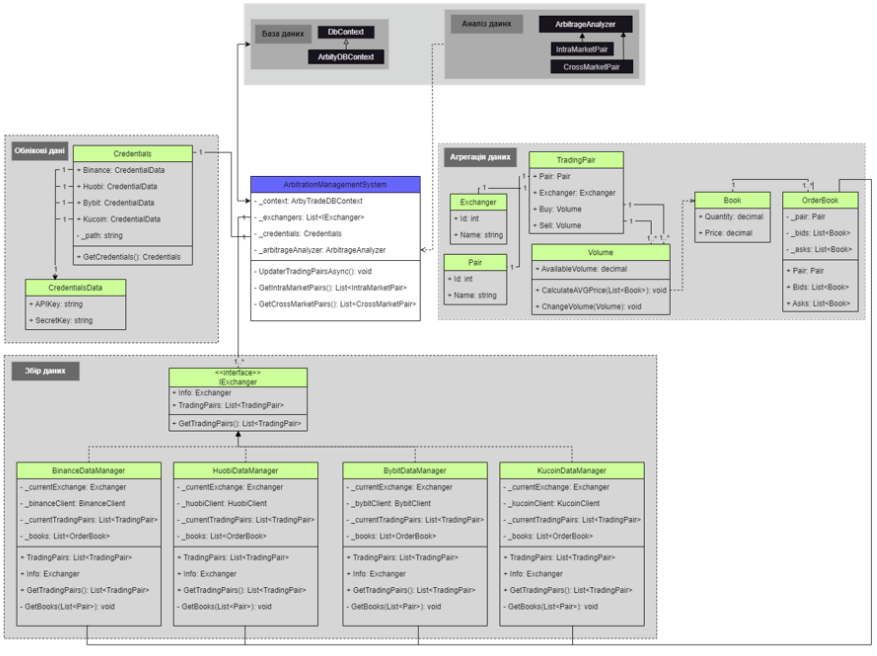
8

ДІАГРАМА КЛАСІВ



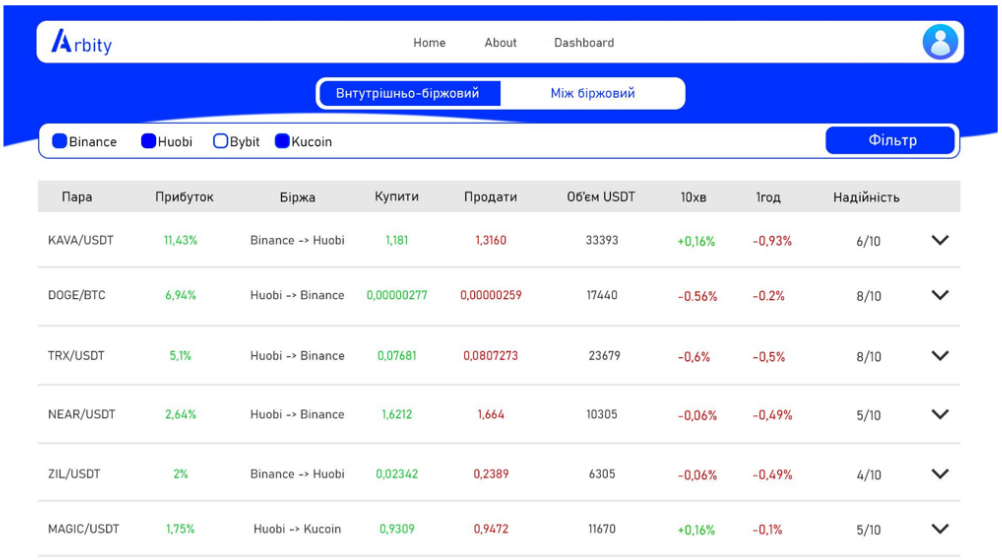
9

ДІАГРАМА ВЗАЄМОДІЇ



10

ЕКРАННІ ФОРМИ



Пара	Прибуток	Біржа	Купити	Продати	Об'єм USDT	10хв	1год	Надійність
KAVA/USDT	11,43%	Binance -> Huobi	1,181	1,3160	33393	+0,16%	-0,93%	6/10
DOGE/BTC	6,94%	Huobi -> Binance	0,00000277	0,00000259	17440	-0.56%	-0.2%	8/10
TRX/USDT	5,1%	Huobi -> Binance	0,07681	0,0807273	23679	-0,6%	-0,5%	8/10
NEAR/USDT	2,64%	Huobi -> Binance	1,6212	1,664	10305	-0,06%	-0,49%	5/10
ZIL/USDT	2%	Binance -> Huobi	0,02342	0,2389	6305	-0,06%	-0,49%	4/10
MAGIC/USDT	1,75%	Huobi -> Kucoin	0,9309	0,9472	11670	+0,16%	-0,1%	5/10

Фото сторінки торгових зв'язків

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Столяр П.Ю / Arbity / Столяр П.Ю // Університетський науково-технічний хакатон "SMART SMART IT 2023", 25.05 2023р., ДУТ. м. Київ - К. ДУТ, 2023, Сертифікат за участь Столяр Павло Юрійович в категорії "IoT".
2. Столяр П.Ю / Розробка модуля агрегації даних мовою C# для підтримки торгівлі криптовалютами / Жебка В.В, Столяр П.Ю // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії, 01-03 червня 2023р., ДУТ, м. Київ - К. ДУТ, 2023.

ВИСНОВКИ

1. Розроблено модуль агрегації даних, що дозволяє автоматизувати збір даних з торгового ринку криптовалют.
2. Обґрунтовано використання технологій .NET та ASP.NET, що дозволяє розробити зручну та масштабовану серверну систему.
3. За результатами проведеного дослідження розроблена підсистема даних для стандартизації обчислень.
4. Розроблена система збору даних за допомогою API та парсингу.
5. Розроблено для фронтенду дизайн, форми та автентифікація.
6. Розроблено фільтр результатів по параметрах.
7. Проведено автоматизоване тестування програмного продукту, що дозволило усунути помилки.

13

ДЯКУЮ ЗА УВАГУ!

Додаток Б

Налагодження WebSocket з'єднання

private void GetBooks(List<Pair> pairs)

```
        {
            int totalCount = 0;
            int count = 0;
            foreach (Pair pair in pairs)
            {
                totalCount++;
                new Thread(() => {
                    var book = new BinanceSpotSymbolOrderBook(pair.Name.Replace("-", ""));
                    var startResult = book.StartAsync().Result;
                    if (startResult.Success)
                    {
                        _books.Add(new OrderBook(pair, book));
                    }
                    count++;
                }).Start();
            }
            while (count < totalCount) { }
        }
```

Оновлення інформації про торгові криптовалютні пари

```
public List<TradingPair> GetTradingPairs()
        {
            foreach (OrderBook book in _books)
            {
                int indexOfCurrentPairData = _currentTradingPairs.FindIndex(pd => pd.Pair.Id == book.Pair.Id);
                if (indexOfCurrentPairData != -1)
                {
```

```
                    _currentTradingPairs[indexOfCurrentPairData].Buy.Change(new
Volume(book.Bids));
                    _currentTradingPairs[indexOfCurrentPairData].Sell.Change(new
Volume(book.Asks));
                }
                else
                {
                    _currentTradingPairs.Add(new TradingPair(book.Pair,
_currentExchange, new Volume(book.Bids), new Volume(book.Asks)));
                }
            }
            return _currentTradingPairs;
        }
```

Авторизаційні дані для доступу до біржі

```
public class Credentials
    {
        public CredentialsData Binance { get; set; }
        public CredentialsData Huobi { get; set; }
        private const string _path = @"D:\Credentials.json";
        private static Credentials _credentials;

        private Credentials() {}

        public static Credentials GetCredentials()
        {
            if(_credentials == null)
            {
                if (!File.Exists(_path))
                {
```

```
                Console.BackgroundColor = ConsoleColor.Red;
                Console.ForegroundColor = ConsoleColor.Black;
                Console.WriteLine($"Creadentials file not found at path {_path}");
                Console.ResetColor();
                throw new FileNotFoundException($"Creadentials file not found at path {_path}");
            }

            var jsonString = File.ReadAllText(_path);
            _credentials = JsonSerializer.Deserialize<Credentials>(jsonString);
        }
        return _credentials;
    }
}

public class CredentialsData
{
    public string APIKey { get; set; }
    public string SecretKey { get; set; }
}
```

TDO класи

```
public class Book
    {
        public decimal Quantity { get; set; }
        public decimal Price { get; set; }

        public Book() { }
        public Book(decimal quantity, decimal price)
        {
```

```
            Quantity = quantity;
            Price = price;
        }
    }
class OrderBook
    {
        private Pair _pair;
        private List<Book> _bids;
        private List<Book> _asks;
        public Pair Pair => _pair;
        public List<Book> Bids => _bids;
        public List<Book> Asks => _asks;

        public OrderBook(Pair pair, ISymbolOrderBook book)
        {
            _pair = pair;
            _bids = new List<Book>();
            foreach(var bookEntry in book.Bids.ToList())
            {
                _bids.Add(new Book(bookEntry.Quantity, bookEntry.Price));
            }
            _asks = new List<Book>();
            foreach (var bookEntry in book.Asks.ToList())
            {
                _asks.Add(new Book(bookEntry.Quantity, bookEntry.Price));
            }
        }

        public OrderBook(Pair pair, HuobiIncementalOrderBook book)
        {
```

```
            _pair = pair;
            _bids = new List<Book>();
            foreach (var bookEntry in book.Bids.ToList())
            {
                _bids.Add(new Book(bookEntry.Quantity, bookEntry.Price));
            }
            _asks = new List<Book>();
            foreach (var bookEntry in book.Asks.ToList())
            {
                _asks.Add(new Book(bookEntry.Quantity, bookEntry.Price));
            }
        }
    }
    public class TradingPair
    {
        public Pair Pair { get; set; }
        public Exchanger Exchanger { get; set; }
        public Volume Buy { get; set; }
        public Volume Sell { get; set; }
        public TradingPair() { }
        public TradingPair(Pair pair, Exchanger exchanger, Volume bids, Volume
asks)
        {
            Pair = pair;
            Exchanger = exchanger;
            Buy = bids;
            Sell = asks;
        }
    }
```