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 СИСТЕМИ ПІДТРИМКИ ТОРГІВЛІ КРИПТОАКТИВАМИ. СПЕЦ. ЧАСТИНА: РОЗРОБКА МОДУЛЯ АНАЛІЗУ ДАНИХ КРИПТОАКТИВІВ МОВОЮ C#**»

Виконав: студент 4 курсу, групи ПД–43
спеціальності
121 Інженерія програмного забезпечення
Яковчук В.А.
(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

ЯКОВЧУКА ВАСИЛЯ АНДРІЙОВИЧА
(прізвище, ім‘я, по батькові)

1.Тема роботи: «Розробка програмного забезпечення системи підтримки торгівлі криптоактивами. Спец. частина: Розробка модуля аналізу 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 застосуванням аналізу даних
3.2 Практичний досвід аналізу даних
3.3 Концепція побудови веб-додатків
4. Перелік демонстраційних матеріалів
4.1 Тема дипломної роботи
4.2 Мета роботи. Об’єкт дослідження. Предмет дослідження
4. Результат дослідження розробки підсистеми для аналізу даних
4.4 Результат дослідження фреймворків для веб-розробки
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 _____ Яковчук В.А.
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Текстова частина бакалаврської роботи: 48 с., 1 табл., 14 рис., 25 джерел.

Об'єкт дослідження – процес виявлення торгових сигналів на криптовалютному ринку.

Предмет дослідження – модуль аналізу даних мовою C# для аналізу історичних цінових даних та виявлення торгових можливостей.

Мета роботи – автоматизація процесу аналізу даних за допомогою розробленої системи підтримки торгівлі мовою C#.

Дипломна робота є результатом дослідження, яке було проведено з метою побудови та оцінки сервісу для отримання прибутку від внутрішньо- та міжбіржового арбітражу криптовалюти.

На початковому етапі дослідження ми вивчили існуючі на цей момент стратегії взаємодії з криптовалютою та підходи до арбітражу. Оцінити фундаментальні поняття і стратегії арбітражної торгівлі вдалося завдяки практиці, вивченню літературних джерел і наукових статей. Деякі з цих методів стали основою для подальшого розвитку сервісу.

Далі були розглянуті платформи та інструменти, які зараз використовуються трейдерами для дослідження ринку та здійснення операцій з криптовалютами. Вивчили їхні можливості, переваги та недоліки. В результаті ми змогли виявити недоліки та потреби трейдерів, які необхідно було задовольнити за допомогою сервісу.

На основі дослідження виявлених потреб було розроблено технічне завдання для автоматизованого рішення для арбітражу криптовалюти. Серед інших важливих функцій, сервіс повинен вміти аналізувати криптовалютні пари, збирати та обробляти дані про обсяги торгів, відбирати пари на основі критеріїв прибутковості та показувати результати трейдерам.

Потім був створений і запущений в експлуатацію сервіс, який мав усі зазначені можливості. Для створення сервісу та його інтеграції було застосовано сучасні технології та програмні інструменти. Для досягнення

найвищої продуктивності та надійності прототип пройшов тестування та оптимізацію.

В останньому розділі дослідження була проведена оцінка ефективності сервісу. Потенційна прибутковість та ризики використання сервісу для арбітражу були визначені шляхом аналізу історичних даних та симуляції торгових операцій на різних криптовалютних парах. Отримані результати дозволяють зробити висновки щодо ефективності та корисності сервісу для отримання прибутку від криптовалютного арбітражу.

Галузь використання – фінансовий сектор і конкретно торгівля криптовалютами, що буде використовуватися трейдерами, інвесторами та іншими учасниками ринку криптоактивів для отримання аналітичних даних, а також генерації торговельних сигналів.

Ключові слова: SQL, арбітраж, веб-додаток, автоматизація, фреймворк, Framework, C#, розробка.

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
DTO	-	Data Transfer Object
MySQL	-	My Structured Query Language

ВСТУП

Криптовалюти, як важлива складова фінансової індустрії, привертають увагу інвесторів та трейдерів з усього світу. Однак для прийняття розумних торгових рішень потрібні потужні аналітичні інструменти через складність та обсяг даних, пов'язаних з криптографічними активами.

Метою проєкту є створення модуля, який запропонує аналіз даних, пов'язаних з криптоактивами, використовуючи технічний аналіз, статистику та машинне навчання. Модуль надасть користувачам доступ до більш широкого спектра інструментів для аналізу та прогнозування поведінки криптоактивів, що дозволить їм приймати обґрунтовані торгові рішення.

Основні вимоги до проєкту - масштабованість, надійність і безпека даних. Планується розширення функціональності модуля, що дозволить користувачам налаштовувати й використовувати різні алгоритми та стратегії аналізу. Також буде приділено увагу гнучкому інтерфейсу, який спростить використання модуля та забезпечить зручний доступ до результатів аналізу даних.

Даний проєкт надасть користувачам потужний інструмент для аналізу криптоактивів, який допоможе їм приймати обґрунтовані рішення при торгівлі на ринку криптовалют і відкриє нові можливості в цьому швидкозростаючому сегменті фінансового світу.

Мета роботи – автоматизація процесу аналізу даних за допомогою розробленої системи підтримки торгівлі мовою C#.

Об'єкт дослідження – процес виявлення торгових сигналів на криптовалютному ринку.

Предмет дослідження – модуль аналізу даних мовою C# для аналізу історичних цінових даних та виявлення торгових можливостей.

Опираючись на поставлену мету, були поставлені наступні задачі:

1. Проаналізувати предметну область серверної розробки.
2. Створити та налаштувати базу даних та створити з’єднання .

3. Розробити модель для розрахунку доступного обсягу монет криптовалютної пари.

4. Розробити алгоритм виявлення внутрішньобіржових пар.

5. Розробити алгоритм виявлення міжбіржових пар.

6. Розробити фільтрацію низьколіквідних та низькоприбуткових пар.

7. Провести тестування серверної сторони.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проєкту

Криптовалюти - це фінансові інструменти, які набувають все більшої популярності та впливу в сучасному світі. Для того, щоб приймати мудрі рішення і використовувати всі переваги цього ринку, все більше трейдерів та інвесторів шукають ефективні методи аналізу даних з криптоактивів.

Ми усвідомлюємо цінність аналізу даних у фінансовій індустрії й те, як він впливає на ефективність торгівлі. Тому в роботі зосереджуємося на створенні надійного, ефективного і зрозумілого модуля аналізу даних криптоактивів, щоб користувачі могли отримувати корисну інформацію і приймати важливі рішення.

Важливість проєкту обумовлена прагненням заповнити прогалину на ринку надійних інструментів для аналізу даних про криптоактиви. Новий модуль аналізу даних і система підтримки криптотрейдингу надають трейдерам доступ до більш широкого спектра інструментів, які допомагають у визначенні трендів, прогнозуванні цін і ефективному виконанні угод.

Завдяки нашій ініціативі користувачі можуть підвищити свій успіх у торгівлі криптовалютою, знизити ризики та приймати більш обґрунтовані рішення. Ви можете сконцентруватися на важливих частинах ринку, використовуючи машинне навчання, передовий технічний аналіз і статистику, щоб отримати глибокі знання, аналізуючи дані про криптовалютні активи.

Наші дослідження є дуже актуальними, враховуючи стрімке зростання популярності криптовалют та попит на надійні аналітичні інструменти серед трейдерів. Це відповідає вимогам сучасної фінансової системи та створює нові можливості для торгівлі та інвестування на величезному світовому ринку криптоактивів.

Крім того, цей проєкт робить більше, ніж просто аналізує дані, пов'язані з криптовалютами. Наша мета - розробити масштабовану систему підтримки

криптотрейдингу, яка зможе задовольнити зростаючі вимоги клієнтів і змінюватися разом з ринком.

У зв'язку з необхідністю проєкту, ми маємо намір збільшити швидкість і ефективність системи, підтримувати додаткові криптоактиви, розширити функціональність, а також забезпечити безпеку і надійність при обробці та зберіганні користувацьких даних. Одним із завдань роботи є надати більш широкий спектр інструментів, які допоможуть трейдерам відстежувати складну поведінку криптовалют, прогнозувати майбутню поведінку і приймати обґрунтовані рішення на основі зібраної інформації.

Ініціатива проєкту спрямована на те, щоб полегшити торгівлю криптовалютами та надати трейдерам ресурси, необхідні для прийняття мудрих рішень.

З огляду на важливість проєкту, варто відзначити наступні додаткові моменти:

1. Збільшення обсягів торгівлі криптоактивами: Ринок криптовалют розширюється і стає все більш активним, щодня відбувається безліч угод. Тому потрібні потужні інструменти аналізу даних, щоб допомогти трейдерам робити мудрий вибір в умовах жорсткої конкуренції.

2. Велика кількість доступних криптоактивів: Ринок криптовалют є домом для сотень різних активів, кожен з яких має свої особливості. Для ефективного дослідження цього розмаїття активів необхідні потужні інструменти та алгоритми, які можуть враховувати унікальні характеристики кожного криптовалютного активу.

3. Турбулентність ринку: Ціни на ринку криптовалют, як відомо, нестабільні та схильні до раптових, різких змін. Для здійснення прибуткових операцій на турбулентному ринку трейдерам потрібні інструменти аналізу даних, які допоможуть їм відстежувати коливання цін, виявляти тенденції та прогнозувати майбутні зміни.

4. Необхідність розуміти ризики: Існують певні фінансові ризики, пов'язані з використанням криптовалют, зокрема волатильність цін,

регуляторна невизначеність та кібербезпека. Трейдери можуть зрозуміти ці ризики, оцінити їх і вжити необхідних заходів для мінімізації майбутніх втрат, проаналізувавши дані про криптоактиви.

5. Впровадження штучного інтелекту: Автоматизація аналізу даних про криптовалютні активи за допомогою штучного інтелекту та машинного навчання дає змогу отримувати більш точні та швидкі результати. Ці технології можна використовувати в ефективному та адаптивному контексті завдяки створенню модуля аналізу даних на C#.

6. Персоналізація : оскільки кожен трейдер має різні вимоги та методи роботи. Цей проєкт спрямований на створення інструментів, які дозволять користувачам обирати необхідні дані та метрики, налаштовувати аналітичні параметри та задовольняти свої специфічні потреби.

7. Регуляторне середовище: Правова система постійно змінюється і схильна до регулювання криптоіндустрії. Трейдери можуть вносити необхідні корективи та запобігати будь-яким проблемам, маючи доступ до найсвіжішої інформації про регуляторні зміни, такі як нові закони, правила і вимоги, завдяки модулю аналітики даних про криптоактиви.

8. Глобальний характер ринку: Ринок криптовалют є міжнародним, торгівля відбувається на багатьох біржах, розташованих у різних країнах. Поєднуючи дані з багатьох джерел і бірж, модуль аналізу даних про криптоактиви може дати трейдерам повну картину ринку і можливість виявити хороші торгові можливості.

Створення програмного забезпечення для системи підтримки торгівлі криптовалютами та модуля для аналізу даних про криптовалюти має важливе значення в сучасному фінансовому ландшафті. Ефективні методи аналізу, прогнозування та вибору часу для торгівлі криптовалютами необхідні через їх зростаючу популярність та значення на сьогоднішній день. Користувачі системи отримають доступ до інструментів, необхідних для ефективного аналізу ринку, прогнозування цін та управління активами після створення модуля аналізу даних. До цього модуля можуть бути включені функції збору

та обробки даних, реалізації алгоритмів технічного аналізу, прогнозування цін та візуалізації даних. Створення такого модуля має сенс у світлі зростаючої потреби в аналітичних інструментах для прибуткової торгівлі криптовалютами.

1.2 Масштабування

Розширення масштабу, функціональності та ресурсів проєкту для задоволення зростаючих потреб і вимог називається масштабуванням. Це може означати обробку більшого обсягу даних, додавання нових функцій, обслуговування більшої кількості користувачів, підвищення продуктивності та створення умов для майбутнього розвитку. В умовах зростання обсягу та складності завдань масштабування проєкту допомагає забезпечити стабільне та ефективне функціонування системи:

1. Розширення функціональності

Масштаб проєкту може бути збільшений шляхом надання модулю аналізу даних більшої функціональності. Для більш ретельного та глибокого дослідження ринку криптоактивів можна впровадити нові аналітичні інструменти, індикатори, графіки та алгоритми.

2. Підтримка модуля криптовалютних активів

Модуль аналізу даних повинен бути адаптивним і швидко адаптуватися до нових активів, коли з'являються нові криптовалюти та блокчейн-ініціативи. Як результат, проєкт повинен бути структурно підготовлений для включення та підтримки широкого спектра криптовалют без необхідності суттєвих модифікацій коду.

3. Масштабування на рівні користувача

Якщо проєкт буде частиною торгової платформи, дуже важливо, щоб він міг масштабуватися на рівні користувача. Це означає, що система повинна підтримувати кілька одночасних користувачів і надавати кожному з них миттєвий доступ до аналітичних даних.

4. Розподілена архітектура

Для забезпечення масштабованості проєкту важливо враховувати розподілену архітектуру, яка дозволяє розміщувати багато компонентів модуля на різних серверах або в хмарних налаштуваннях. Це гарантуватиме адаптивність та масштабованість системи відповідно до вимог користувачів та навантаження.

5. Підтримка різних джерел даних

Ринок цифрових активів може отримувати інформацію з різних джерел, включаючи біржі, блокчейни, новинні сайти тощо. Модуль повинен мати можливість взаємодіяти з багатьма джерелами та надавати уніфікований формат даних для подальшого аналізу, щоб гарантувати ретельність і якість аналізу даних.

6. Розширення географічного охоплення

Масштабність географічного охоплення проєкту дозволяє розширювати його доступність і вплив на міжнародному рівні, ураховуючи різні регіони, ринки та місцеві особливості.

7. Масштабованість інтеграцій

Проєкт повинен бути обладнаний для підключення до різних платформ та інструментів, що використовуються в екосистемі торгівлі криптовалютою. Деякі приклади включають торгові платформи, біржі, платіжні методи та портфельні менеджери. Функціональність проєкту буде розширена, а його ефективна взаємодія з іншими системами буде забезпечена шляхом забезпечення сумісності та готовності до інтеграції.

8. Масштабність безпеки

Масштабовані методи захисту даних, контролю доступу та моніторингу подій безпеки необхідні проєкту через зростаючі ризики для кібербезпеки. Для успіху проєкту важливо, щоб дані користувачів були надійними, конфіденційними та точними.

Розширення функціональності, підтримка додаткових цифрових активів, масштабування на рівні користувача, розподілена архітектура, підтримка

різноманітних джерел даних, географічне покриття, масштабовані інтеграції та безпека - все це стає можливим завдяки масштабованості проєкту. Забезпечуючи масштабованість, можна збільшити пропускну здатність і ефективність системи, щоб краще задовольняти зростаючі потреби користувачів і ринку криптовалют.

1.3 Перелік аналогів

Для ефективного виконання криптоарбітражних угод та моніторингу ринку існує ряд автоматизованих сервісів, які надають учасникам можливість отримувати розширену інформацію, аналізувати ринкові дані та виконувати угоди на різних криптобіржах. В даній дипломній роботі наведено такі аналоги, як: Coingecko, Cryptohopper та CoinArbitrageBot і їх можливо порівняти таким чином:

1. coingecko.com

Coingecko - це онлайн-платформа, що надає користувачам інформацію про криптовалюти. Вона функціонує як агрегатор даних про криптовалютні активи, збираючи інформацію з різних бірж та інших джерел. Він є популярним сервісом серед трейдерів, інвесторів та криптовалютних ентузіастів, які шукають надійну та детальну інформацію про криптовалюти та їх ринок.

Перейти до ознайомлення сервісу можна з головної сторінки проєкту (рис. 1.1), вибравши криптовалюту, яку потрібно проаналізувати (рис. 1.2).

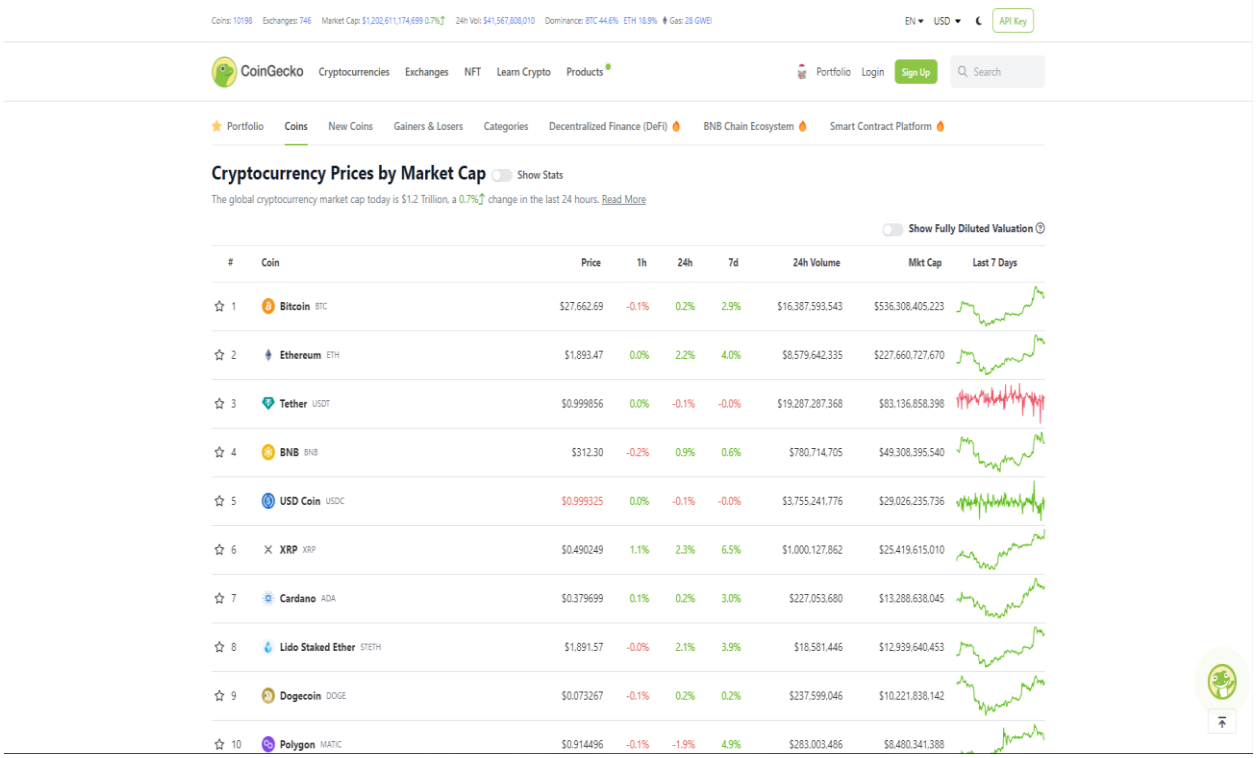


Рисунок 1.1 — Головна сторінка

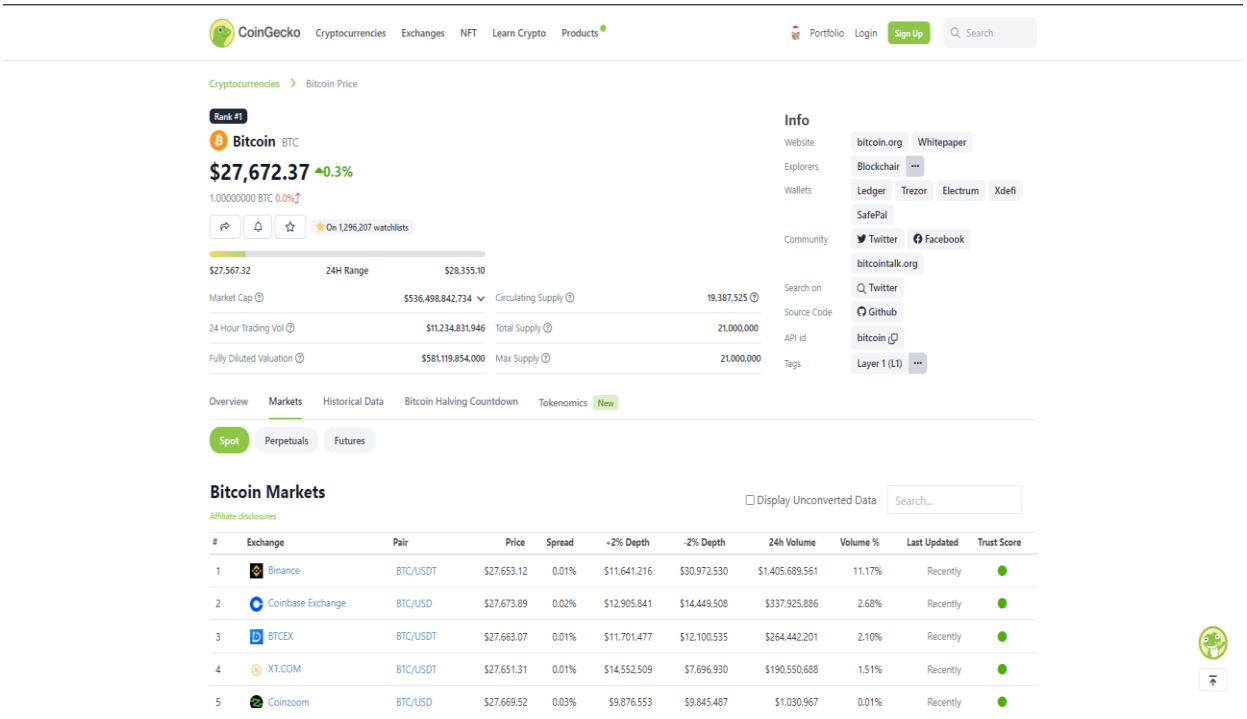


Рисунок 1.2 — Доступні ринки для монети Bitcoin

Сервіс має такі недоліки, як:

- Неточність ціни.

- Довге оновлення даних ціни.
- Дані про ліквідність відсутні.
- Немає даних про об’єми.
- Відсутність внутрішньобіржового арбітражу.
- Неможливо проаналізувати прибуткові торгові пари.

2. cryptohopper.com

Cryptohopper - це автоматизована платформа для торгівлі криптовалютами. Вона пропонує послуги автоматичного трейдингу, де торгові рішення приймаються на основі аналізу ринку та стратегій, встановлених користувачем.

Cryptohopper надає зручні інструменти для автоматизованої торгівлі криптовалютами та може бути корисним для трейдерів з різним рівнем досвіду.

Сервіс має зручний інтерфейс та стислість інформації на якій можна ознайомитись на головній сторінці (рис. 1.3).

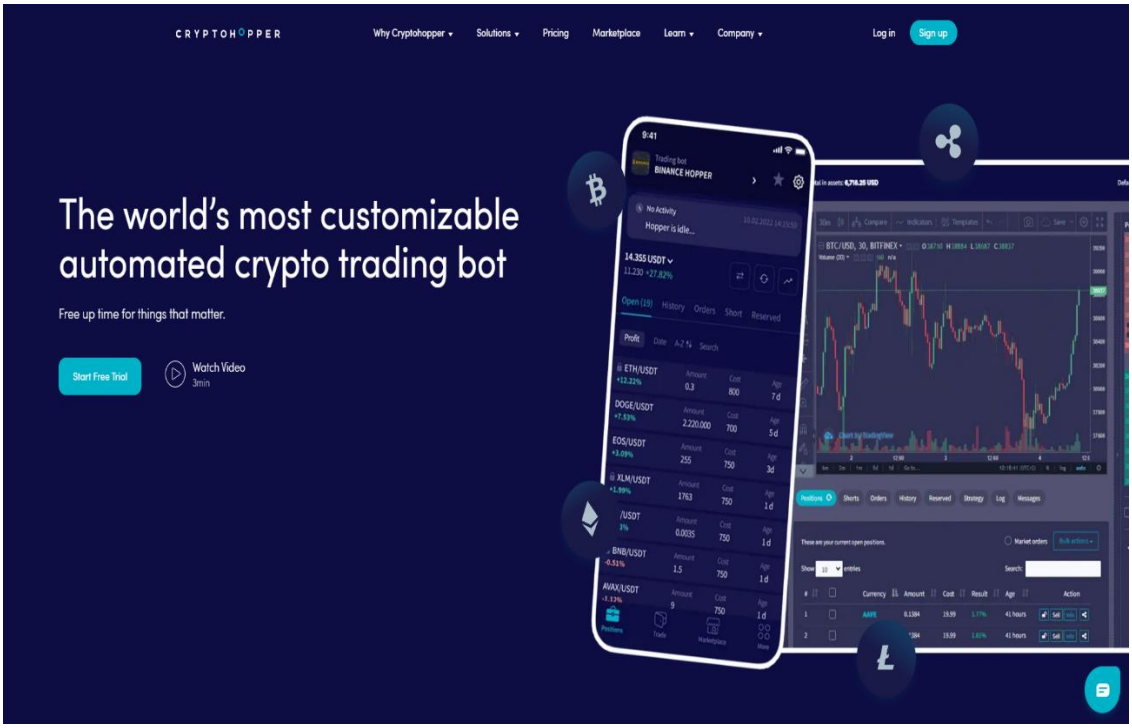


Рисунок 1.3 — Головна сторінка

Проте, для заробітку на арбітражі криптовалюти має такі недостатки:

- Міжбіржовий арбітраж: Сервіс непризначений для міжбіржового арбітражу.
- Вимоги до налаштувань: Cryptohopper потребує від користувача налаштування стратегій та параметрів для автоматичної торгівлі. Це може бути складним для новачків, які можуть потребувати часу та знань, щоб належним чином налаштувати платформу.
- Неможливо проаналізувати прибуткові торгові пари.

3. coinarbitragebot.com

CoinArbitrageBot.com - це веб-сервіс, спеціально розроблений для надання інструментів та інформації, необхідних для виконання арбітражу криптовалют між різними біржами. Він допомагає трейдерам та інвесторам знайти та використати можливості для арбітражу, що виникають на ринку криптовалют.

Для того, щоб ознайомитись з інформацією по торговим парам потрібно з головної сторінки (рис. 1.4) перейти на вкладку “Tool - Arbitrage”(рис 1.5), де ми зможемо вибрати доступні біржі (рис 1.6).

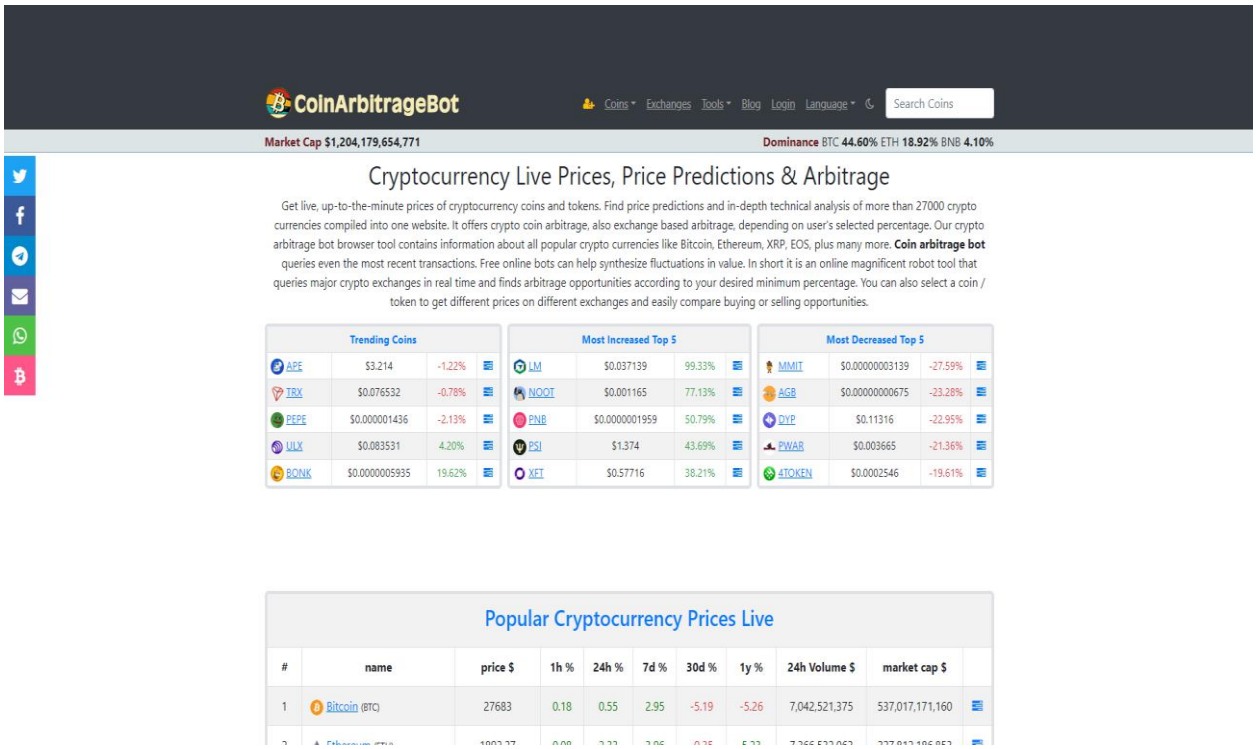


Рисунок 1.4 — Головна сторінка

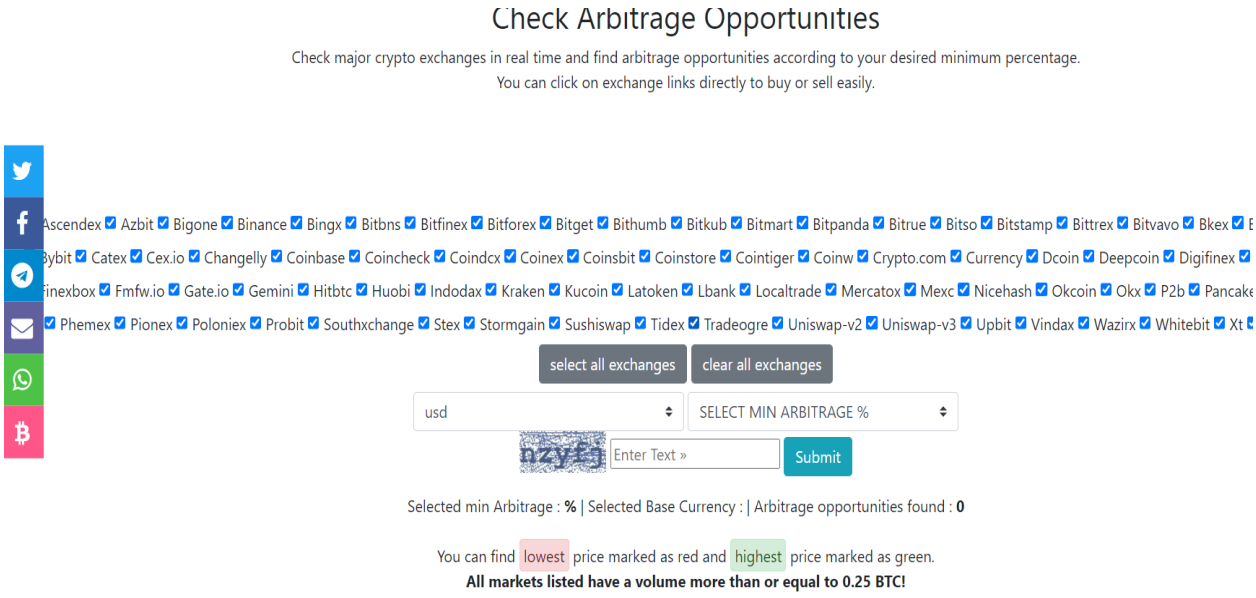


Рисунок 1.5 — Вкладка “Arbitrage”

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

Сервіс має такі недоліки, як:

- Оновлення ціни: Інформація ціни криптовалюти чи пари не актуальна з інформацією на ринку із-за довгого оновлення.
- Відсутність внутрішньобіржового арбітражу.
- Галузь використання: В загалом сервіс використовують торгові боти. Для заробітку користувачеві, не буде надано актуальної інформації.
- Відсутня ліквідність.

За результатами порівняння трьох сервісів - Coingecko, Cryptohopper та CoinArbitrageBot.com, можна зробити наступні висновки (табл. 1):

Coingecko є популярною платформою для отримання інформації про криптовалюти, проте має деякі недоліки, такі як неточність ціни, відсутність даних про ліквідність та об'єми, а також відсутність внутрішньобіржового арбітражу.

Cryptohopper є автоматизованою платформою для торгівлі криптовалютами, яка пропонує зручні інструменти для автоматичної торгівлі. Однак, він не підтримує міжбіржовий арбітраж та вимагає від користувачів налаштування стратегій та параметрів.

CoinArbitrageBot.com спеціально розроблений для арбітражу криптовалют між різними біржами. Він надає інструменти для моніторингу,

але не актуальний для користувачів, має недоліки в оновлені ціни та функціоналу.

				
	CoinArbitrageBot	Cryptohopper	Coingecko	**Arbity**
Платформи	Windows, Linux	Windows, Linux	Web	**Web**
Рівень складності	Складний	Складний	Простий	**Простий**
Внутрішньобіржовий арбітраж	Відсутній	Присутній	Відсутній	**Присутній**
Міжбіржовий арбітраж	Присутній	Відсутній	Присутній	**Присутній**
Інтервал оновлення інформації	~3хв	~10хв	~5хв	**60с**
Врахування ліквідності	Немає	Немає	Немає	**Є**
Автоторгівля	Присутня	Присутня	Відсутня	**Відсутня**

Таблиця 1 — Перегляд аналогів

2 ДОСЛІДЖЕННЯ ІНСТРУМЕНТІВ РОЗРОБКИ

2.1 Мова програмування C#

C# (Сі-шарп) - це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft у 2000 році. Вона базується на платформі .NET (в тому числі на Common Language Runtime - CLR), яка забезпечує виконання коду C# на різних платформах, до прикладу Windows.

C# була створена з метою розробки програмного забезпечення для платформи Microsoft для створення Windows-додатків, веб-додатків, серверних додатків та інших програм. Вона підтримує об'єктно-орієнтований підхід до програмування, що дозволяє розділити реалізацію в сутності, тому пропонує багатий набір функціональних можливостей для розробників. [1]

Мова C# має багато потужних можливостей, серед яких:

1. Збирання сміття (garbage collection): У C# пам'ять автоматично зберігається, а за позбавлення від непотрібної інформації відповідає збирач сміття. Це спрощує розробку і позбавляє від багатьох проблем з управлінням пам'яттю, на відміну від C++.
2. Обробка винятків: C# надає механізми для елегантної обробки винятків, що дозволяє забезпечити стійкість програми до виліту.
3. Об'єктно-орієнтоване програмування: Щоб структурувати та організувати свій код, використовувати принципи ООП, такі як успадкування, поліморфізм, класи та об'єкти в C#.
4. Великий обсяг бібліотек: Фреймворк The.NET включає декілька стандартних бібліотек для C#, 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.

5. Багатопотоковість: За допомогою надійних засобів багатопотокового програмування C#, можливо створювати паралельні та асинхронні процеси, які відповідають потребам сучасних додатків і підвищують продуктивність. [2]

На момент написання цієї дипломної роботи найновішою версією мови була C# 9.0, яка дебютувала в листопаді 2020 року разом з .NET 5.0.

Початок C# можна простежити до оголошення Microsoft у 1999 році про створення абсолютно нової мови програмування для платформи .NET. Андерс Гейлсберг (Anders Geilsberg) та його група керували створенням C#. У 2002 році платформа .NET та перша версія C#, версія 1.0, стали доступними.

Щодо чистки пам'яті, яку згадували раніше(garbage collection) в C# використовується механізм автоматичного управління пам'яттю, а це означає, що розробникам не потрібно вручну виконувати операції з управлінням пам'яттю. Збирач сміття автоматично виявляє невикористані об'єкти та звільняє пам'ять, яку вони займають. Це дозволяє уникнути багатьох типових помилок, пов'язаних з управлінням пам'яттю, таких як витоки пам'яті та некоректне використання звільненої пам'яті. При бажанні, можна вручну керувати пам’яттю. Керування пам’яттю може знадобиться, якщо потрібно більш гнучко прописати алгоритм роботи програми. [3]

Компіляція в мові C# відбувається у два етапи. Спочатку розробник пише програму у вихідних файлах, які містять код на мові C#. Потім ці файли компілюються за допомогою компілятора, що перетворює їх у виконуваний код або у файл з проміжним представленням (Intermediate Language - IL). Файли з IL підлягають подальшій компіляції в машинний код або у виконуваний код під час запуску програми. Цей процес дозволяє виконувати програму на будь-якій платформі, на якій доступна технологія .NET. Серед плюсів такої компіляції, те що кінцевий файл займає менше пам’яті та швидше виконується. Серед мінусів, те що перший запуск програма працює повільніше через те, що працює процес кешування для подальшого пришвидшення роботи. [3] [4]

Якщо порівнювати з Java, C# має схожий синтаксис, оскільки на обидві мови вплинув C++. Однак, є кілька відмінностей. C# підтримує властивості (properties), які дозволяють змінювати значення приватних полів через методи доступу get і set. Крім того, C# має лямбда-вирази, які дозволяють зручно визначати анонімні функції. В обох мовах є збирання сміття, але в C# воно є більш розширеним. [5]

Платформа The.NET функціонує з основними бібліотеками C#, які пропонують широкий спектр можливостей для створення різноманітних додатків. Деякі з них включають в себе:

1. System. Threading: Ця бібліотека містить класи для обробки багатопотоковості, паралельного програмування, синхронізації потоків та інших дій, пов'язаних з багатопотоковим середовищем.

2. System. Collections: Списки, стеки, черги, хеш-таблиці та інші структури даних доступні в цій бібліотеці, щоб допомогти в управлінні та організації колекцій даних.

3. System.Linq: Виконує структуровані та декларативні операції з даними та запити, використовуючи підтримку цієї бібліотеки для мови LINQ (Language Integrated Query).

4. System.Net: До цієї бібліотеки включено класи для взаємодії з мережевими операціями, такими як веб-запити, сокети, протоколи тощо.

5. System. WindowsForms: Ця бібліотека надає класи для створення графічних інтерфейсів користувача за допомогою технології 6.

6. Windows Forms. Використовується, щоб створювати такі елементи інтерфейсу, як вікна, кнопки, повідомлення, таблиці тощо.

7. System.IO: До цієї колекції включено класи для керування діями вводу/виводу, такими як взаємодія з файлами, потоками, каталогами та іншими системними ресурсами вводу/виводу.

8. System. Web: Ця бібліотека містить класи для створення веб-додатків, які можуть з'єднуватися з веб-сервером, обробляти запити та

відповіді, керувати сеансами, переходити за URL-адресами та виконувати інші операції, пов'язані з Інтернетом.

9. System. Data: Ця бібліотека містить класи для встановлення з'єднань з базами даних, виконання запитів та роботи з даними. Вона надає користувачам доступ до набору інструментів управління реляційними базами даних, відомих як ADO.NET (ActiveX Data Objects).

10. System.Drawing: у цій бібліотеці доступні класи, а також інформація про те, як створювати малюнки, змінювати кольори, малювати лінії та форми та виконувати інші графічні операції.

11. System. Security: Ця бібліотека містить класи для вирішення проблем, пов'язаних з безпекою в додатках, таких як шифрування даних, підписання даних та інші проблеми, пов'язані з безпекою.

12. System: Класи і типи для взаємодії з файлами, вводом/виводом, процесами, потоками, мережами та іншими системними ресурсами включені в цю бібліотеку. [6]

Це лише деякі з основних бібліотек, які надає C#. Крім того, значна кількість сторонніх бібліотек розширює функціональність мови та надає інструменти для додаткових сфер розробки, таких як графіка, обробка даних, мережеве програмування, машинне навчання тощо.

Entity Framework для роботи з базами даних, ASP.NET для створення онлайн-додатків, Xamarin для створення мобільних додатків та Newtonsoft - все це відомі сторонні бібліотеки C#. JSON, серед іншого, для роботи з форматом JSON.

Завдяки широкому набору функцій C# використовується для розробки в найпопулярніших сферах, включаючи онлайн-додатки, мобільні додатки, десктопні додатки, ігри, системи управління базами даних, хмарні рішення та багато іншого. Вона слугує базовою мовою для платформи .NET, що підтримується Microsoft, яка має значну спільноту розробників та екосистему. [5] [6]

Загалом, C# є потужним інструментом розробки програмного забезпечення, який пропонує зручний синтаксис, глибоку функціональність, широкий спектр бібліотек та підтримку значної спільноти розробників. Завдяки своїй адаптивності та портативності, C# є однією з найпопулярніших мов програмування в сучасному програмуванні та важливою частиною фреймворку .NET.

2.2 Framework ASP.NET Core

Корпорація Майкрософт створила веб-фреймворк з відкритим вихідним кодом, відомий як ASP.NET Core. Він був створений, щоб запропонувати чудову продуктивність і крос-платформну функціональність, що означає, що він може працювати на Linux та інших серверних операційних системах на додаток до Windows. Він прийшов на зміну першій версії фреймворку ASP.NET.

Основна мета ASP.NET Core - надати програмістам інструменти, необхідні для створення складних веб-додатків, які можна використовувати на будь-якій платформі (Windows, macOS і Linux) і розгортати як на локальних серверах, так і в хмарних інфраструктурах. Фреймворк надає потужні інструменти для розробки веб-додатків, мікросервісів, веб-сервісів, API та багато іншого. [7]

Ключовими атрибутами ASP.NET Core є:

1. 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.

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

3. Підтримка облікових записів користувачів: Інтегрована у фреймворк, включаючи використання файлів cookie, JWT, OAuth та інших протоколів для автентифікації та авторизації користувачів.

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

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

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

7. 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 багатопотоковість, що робить його бажаним варіантом для створення сучасних веб-додатків. [8]

Щодо порівняння з аналогами, такими як Java і C++, варто відзначити наступне:

1. ASP.NET Core і Java - це два добре відомі веб-фреймворки, які використовують мову програмування Java, хоча C# не підтримується. Універсальність цієї технології забезпечується широкою екосистемою Java та великою кількістю бібліотек і фреймворків. Перевага ASP.NET Core, з іншого боку, полягає в тому, що вона має доступ до найновіших технологій і постачається з вбудованими інструментами для створення веб-додатків.

2. C++ - це мова загального призначення, яка пропонує високий рівень продуктивності та управління ресурсами. Оскільки C++ є мовою нижчого рівня, ніж C#, програмісти мають у своєму розпорядженні більше

інструментів оптимізації та управління пам'яттю. Недоліком розробки на C++ є те, що вона може ставати дедалі складнішою, тривалішою та дорожчою. [9]

На момент написання цієї дипломної роботи (вересень 2021 року) останньою версією була ASP.NET Core 6.0. Microsoft зазвичай регулярно публікує оновлені, вдосконалені та виправлені версії фреймворку.

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 крос-платформна підтримка. Згодом були опубліковані версії 2.0, 2.1, 2.2 і 3.0, які принесли з собою вдосконалення і нові функції. Більш точна інформація щодо останніх версій може бути недоступною з огляду на час, що минув з моменту останнього оновлення моделі знань. [10]

Одна з найпопулярніших парадигм розробки онлайн-додатків, структура MVC (Model-View-Controller), підтримується ядром ASP.NET Core. Модель, подання та контролер - це три основні частини, на які MVC дозволяє розділити функціональність додатка.

1. Модель (Model) представляє бізнес-логіку та дані додатка. Вона відповідає за зберігання стану додатка, 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. Поданням може бути будь-який спосіб представлення інформації, включаючи візуальний інтерфейс, HTML-сторінку, шаблон, JSON-відповідь та інші.

3. Контролер (Contoller) відповідає за відповіді на запити користувачів, керування потоками додатка та координацію з моделлю та представленням. Отримуються запити, виконуються завдання, здійснюється

доступ до моделі для отримання даних, а потім контролер обирає правильне представлення для відображення результату користувачеві. [11] [12]

Отже, MVC структура дозволяє розподілити відповідальності між компонентами додатка, що сприяє покращенню контролю та масштабування. Кожен компонент виконує свою функцію і це дозволяє розробникам працювати над окремими частинами системи незалежно один від одного.

2.3 Середовище розробки

Visual Studio - це інтегроване середовище розробки (IDE), створене компанією Microsoft для розробки програмного забезпечення. Воно надає зручний інтерфейс та широкий набір інструментів для розробки різноманітних типів додатків, включаючи веб-сайти, мобільні додатки, настільні програми та хмарні рішення.

Програмісти можуть розробляти код, тестувати його, аналізувати продуктивність та розгортати додатки за допомогою Visual Studio. 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. [13]

Visual Studio був створений і розроблений командою в Microsoft. Перша версія була випущена у 1997 році, а з того часа Microsoft регулярно випускає оновлення та нові версії з поліпшеннями та новими функціями. Найновіша версія Visual Studio на момент написання бакалаверської роботи у вересні 2021 році була Visual Studio 2019.

Для розробки на платформі ASP.NET і мові програмування C# рекомендується використовувати Visual Studio. Visual Studio надає спеціалізовані шаблони проєктів, інструменти налагодження, розширення для

підтримки ASP.NET та C#, інтеграцію зі сховищами даних та багато інших функцій, спрощуючи розробку на цих платформах. [14]

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

1. Інтегроване середовище розробки: Заснована на платформі ASP.NET та мові програмування C#, Visual Studio пропонує повне інтегроване середовище розробки. У ньому є всі елементи, ресурси та опції, необхідні для ефективної розробки додатків.

2. Шаблони проєктів: Visual Studio надає широкий вибір шаблонів проєктів для розробки веб-додатків на платформі ASP.NET. Ці шаблони автоматично налаштовують основну структуру проєкту та включають основні компоненти, що допомагають почати розробку з нуля або на основі шаблону.

3. Засоби налагодження коду: Visual Studio має потужні засоби налагодження. Ви можете використовувати точки зупинки, перевіряти значення змінних, слідкувати за виконанням програми крок за кроком та багато іншого. Завдяки цьому ви можете ефективно виправляти помилки та налагоджувати код.

4. Підтримка IntelliSense: Visual Studio містить функцію IntelliSense, яка пропонує альтернативні варіанти коду, методи та властивості під час введення коду. Це полегшує роботу з синтаксисом мови та зменшує кількість помилок.

5. Інтеграція зі сховищами даних: Visual Studio підтримує інтеграцію сховища даних з низкою систем контролю версій, включаючи Git і Team Foundation Server (TFS). Це спрощує використання розподіленої розробки, взаємодію з командою розробників та підтримку версій коду.

6. Розширення та плагіни: Можливо розширити можливості IDE за допомогою розширень та плагінів, які підтримує Visual Studio. Спільнота розробників створила величезну кількість розширень, які надають додаткові функції, інструменти та підтримку декількох технологій.

7. Підтримка ASP.NET та C#: Visual Studio пропонує спеціалізовані інструменти та можливості для роботи мовою програмування C# та платформі ASP.NET. 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 іспанським інтернет-сервером: Spanish Internet Server (ASP.NET), платформа для створення веб-сайтів та онлайн-додатків, підтримується Visual Studio. Можливо створювати веб-додатки з використанням ASP.NET MVC, ASP.NET Web Forms, ASP.NET Web API та інших технологій, використовуючи Visual Studio для роботи з ASP.NET. За допомогою Visual Studio можна швидко створювати складні веб-додатки, підтримувати моделі даних та інтеграцію з базами даних.

9. Розробка на мові C#: Основним середовищем розробки для мови програмування C# є Visual Studio. Компанія 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. Командна робота та співпраця: Visual Studio постачається з інструментами для командної роботи та співпраці. Контроль версій, коментарі, спільний доступ до завдань та інші функції спрощують роботу з декількома розробниками над одним проєктом і забезпечують хорошу комунікацію та співпрацю. [10] [13]

Можливо швидко створювати веб-додатки та програми на основі ASP.NET і C#, використовуючи Visual Studio для роботи з цими технологіями. Серед інших переваг використання Visual Studio можна виділити наступні:

1. Підтримка інструментів тестування: Тести продуктивності, функціональні та модульні тести вбудовані у Visual Studio для тестування

коду. Це дозволяє знаходити та виправляти дефекти програми на ранніх стадіях розробки, забезпечуючи високу якість продукту.

2. Візуальний редактор та дизайнер інтерфейсу: Visual Studio містить вбудовані візуальні редактори та дизайнери інтерфейсів, які дозволяють створювати та змінювати користувацький інтерфейс програми. Без необхідності писати код вручну, легко створювати веб-сторінки, додавати елементи керування, змінювати стилі та інші елементи інтерфейсу.

3. Підтримка розгортання та публікації: Visual Studio спрощує розгортання та публікацію додатків на різноманітних сервісах та платформах, включаючи веб-сервери та хмару Azure. Це полегшує розгортання додатків і пропонує практичні інструменти для управління програмним забезпеченням в реальних умовах.

4. Розширюваність: Значна спільнота розробників Visual Studio створює широкий спектр плагінів та розширень. Додаючи додаткові інструменти, шаблони проєктів, розширення для роботи зі сторонніми бібліотеками та інші функції, можливо розширити можливості Visual Studio, що спростить роботу і створить більш індивідуалізоване середовище розробки. [15]

Таким чином, Visual Studio є потужним і необхідним інструментом для розробки програмного забезпечення, особливо при використанні ASP.NET і 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 розробникам створення ефективного та якісного програмного забезпечення, спрощуючи створення, тестування, налагодження та розгортання веб-додатків та додатків на C#.

2.4 Бази даних

База даних (англ. database) - це структурована колекція даних, яка використовується для ефективного зберігання, організації та пошуку великих обсягів даних. Бази даних використовуються для різних цілей, включаючи:

1. Зберігання інформації: Використовуючи бази даних можливо зберігати багато видів даних в організованому вигляді, включаючи текст, числа, фотографії, аудіо та багато іншого. Вони дають практичний спосіб впорядкувати дані для зберігання та легкого доступу до них, коли вони будуть потрібні.
2. Обробка та аналіз даних: Робота з даними, включаючи сортування, фільтрацію, обчислення, статистичний аналіз і підготовку звітів, можлива завдяки базам даних. Це дає можливість виконувати цілий ряд дій, від простих пошукових запитів до складних аналітичних процесів.
3. Керування даними: Бази даних пропонують прості засоби для додавання, оновлення та видалення даних, що дозволяє ефективно керувати ними. Використовувати їх, щоб встановити правила цілісності, гарантувати безпеку даних і керувати доступом до них.
4. Доступ до даних: Завдяки базам даних кілька користувачів або програм можуть отримати доступ до даних одночасно. Вони дають змогу запитувати дані та обмінюватися інформацією між кількома програмами. [16]

Бази даних були розроблені у 1960-х роках. Реляційна модель баз даних, яка є однією з найпоширеніших моделей, була запропонована Едгаром Кодом в IBM у 1970-х роках. Реляційні бази даних використовують табличну структуру для збереження даних, де дані організовані у вигляді таблиць зі стовпцями та рядками, а взаємозв'язки між таблицями встановлюються за допомогою ключів. Цей підхід став популярним і отримав широке використання у багатьох сферах, включаючи бізнес, науку та інформаційні системи. [16]

Основні типи баз даних включають:

1. Реляційні бази даних: Ці бази даних побудовані на реляційній парадигмі, де дані розташовані в таблицях зі стовпчиками й рядками. Ці бази даних дозволяють користувачам запитувати та змінювати дані за допомогою SQL.

2. Нереляційні бази даних: Ці бази даних не використовують реляційну парадигму. Вони також можуть використовувати інші моделі, такі як графові, ключ-значення, стовпчикові та документальні моделі. Для роботи з великими обсягами даних і віддаленого зберігання, нереляційні бази даних набули популярності.

3. Ієрархічні бази даних: Ці бази даних налаштовані як батьківські та дочірні вузли в ієрархії. Вони часто використовуються в системах управління даними, таких як системи управління файлами, де ієрархічна структура є природною.

4. Мережеві бази даних: Кожен запис може мати кілька батьківських і дочірніх записів завдяки моделі зв'язків між записами, яку вони використовують. Хоча сьогодні вони зустрічаються рідше, цей тип баз даних широко використовувався в 1960-х і 1970-х роках. [17]

2.5 SQL

SQL (Structured Query Language) - це мова запитів до баз даних, яка використовується для зберігання, керування та зміни даних. Доступ до баз даних з даними, організованими у вигляді таблиць зі стовпчиками та рядками, є стандартним для SQL.

Існує кілька способів використання SQL, зокрема

1. Створення баз даних: SQL дозволяє створювати таблиці, стовпці, індекси та зв'язки між таблицями, а також структуру бази даних.

2. Відбір даних і створення запитів: SQL дозволяє створювати запити для вибору даних з бази даних. Можливі як прості запити (наприклад, ті, що

просто вибирають всі записи з певної бази даних), так і складні запити (які використовують критерії, сортують, групують і об'єднують дані з багатьох таблиць).

3. Додавання, оновлення та видалення даних: SQL дозволяє додавати нові записи до таблиць, оновлювати вже існуючу інформацію та видаляти записи, які більше не потрібні.

4. Керування базами даних: SQL пропонує команди для створення, видалення або зміни структури бази даних, а також для управління користувачами та надання дозволів на доступ.

IBM розробила SQL у 1970-х роках. Структурована англійська мова запитів, або SEQUEL, була первісною назвою мови SQL до того, як її перейменували на SQL. В цей час існує кілька реалізацій SQL, доступних від різних постачальників баз даних. SQL є стандартом ANSI (Американського національного інституту стандартів). [18]

На момент написання цієї дипломної роботи, SQL:2016 був останньою версією стандарту SQL. Реалізації SQL, що використовуються різними базами даних, можуть відрізнятися і включати додаткові можливості та розширення.

Існує два основних способи використання SQL для зв'язку з C# та базами даних стосовно Visual Studio та баз даних:

1. Використання ADO.NET: ADO.NET (ActiveX Data Objects.NET) - це набір класів і частин, які працюють з середовищем .NET для підключення до баз даних, виконання SQL-запитів і зміни даних. Для встановлення з'єднань з базами даних, виконання запитів та отримання результатів у C#-сумісному форматі у Visual Studio використовується ADO.NET.

2. Застосування стратегій ORM: Об'єктно-орієнтована взаємодія з базами даних стає можливою завдяки технології ORM (Object-Relational Mapping - об'єктно-реляційне відображення). Можливо прив'язувати об'єкти C# до таблиць бази даних і автоматично створювати SQL-запити, використовуючи ORM-рішення, такі як Entity Framework або NHibernate. В результаті, робота з базою даних значно спрощується, оскільки можливо

взаємодіяти з даними у вигляді об'єктів, а не вручну створювати складні SQL запити. [19]

Широка функціональність роботи з базами даних, яку пропонує Visual Studio, включає можливість конструювання з'єднань з базами даних, створення і запуск запитів, графічне проєктування схеми бази даних і генерування коду для взаємодії з базою даних. Visual Studio дозволяє легко взаємодіяти з базою даних, створювати SQL запити, будувати моделі даних і налагоджувати запити.

Взаємодія SQL, C# та Visual Studio відкриває широкі можливості для створення додатків для роботи з базами даних. Бази даних спрощують створення, зберігання та оновлення даних, а також виконання складних пошуків, керування транзакціями та захист даних. Поєднуючи SQL з C# та Visual Studio, можливо створювати потужні програми, які ефективно працюють зі збереженими даними. [18] [19]

Якщо порівнювати SQL і бази даних з іншими методами зберігання даних, то можна виділити наступні їхні ключові переваги::

1. Структуроване зберігання даних: Бази даних пропонують систематизований спосіб зберігання даних, де можна вказати таблиці, поля та зв'язки між ними. Таким чином, є можливість ефективно керувати даними та організовувати їх.

2. Можливість виконання складних запитів: SQL підтримує цілий ряд операцій з базами даних, таких як пошук, фільтрація, сортування, об'єднання та агрегування даних. Це дозволяє швидко та ефективно отримувати доступ до потрібних даних з бази даних.

3. Масштабованість: Бази даних здатні зберігати та швидко отримувати величезні обсяги даних. Їх можна розширювати для використання у великих корпоративних середовищах або розподілених системах.

4. Безпека даних: Контроль доступу, шифрування та системи резервного копіювання - це лише деякі з інструментів, які бази даних пропонують для захисту даних. Це сприяє цілісності та конфіденційності даних. [20]

SQL можна використовувати в C# за допомогою рішень ADO.NET або ORM, вбудованих SQL-запитів або інших методів. Є можливість використовувати бібліотеки, наприклад, бібліотеки для взаємодії з базами даних C#. Ці бібліотеки містять класи та методи для створення з'єднань з базами даних, виконання SQL-запитів, отримання результатів і роботи з даними.

Основні кроки для взаємодії з базою даних у C# та Visual Studio через SQL можуть включати наступні дії:

1. Підключення до бази даних: Ви маєте використати відповідну бібліотеку, таку як для створення підключення до бази даних. Ви повинні вказати параметри підключення, такі як адреса сервера, ім'я бази даних, облікові дані тощо.

2. Виконання SQL-запитів: Після успішного підключення ви можете виконувати SQL-запити до бази даних. Використовуйте класи та методи, надані бібліотекою, для виконання запитів. Наприклад, ви можете використовувати *SqlCommand* для створення об'єкта запиту та викликати методи, такі як *ExecuteReader* для отримання результатів запиту.

3. Обробка результатів: Після виконання запиту ви можете обробити отримані дані. Використовуйте відповідні методи для отримання значень з результатів запиту та їх подальшої обробки у C#.

4. Закриття підключення: Після завершення роботи з базою даних важливо закрити підключення, щоб звільнити ресурси. Потрібно використовувати метод *Close* або *Dispose* на об'єкті підключення для закриття з'єднання. [20] [21]

2.6 Entity Framework

Група технологій платформи .NET під назвою Entity Framework забезпечує об'єктно-орієнтоване програмування та взаємодію з базами даних

на основі моделей. Вона функціонує як об'єктно-реляційний маппер (ORM), який пропонує ефективний підхід до взаємодії з даними.

Microsoft створила Entity Framework, яка була вперше доступна у 2008 році. Це розвиток попередньої технології ADO.NET, покликаний полегшити створення додатків, пов'язаних з базами даних.

Entity Framework Core 6.0, яка призначена для роботи на багатоплатформному фреймворку NET Core, була останньою версією на момент написання цієї бакалаврської роботи. Зараз він підтримує ряд баз даних, має додаткові функції та підвищену швидкість роботи. [22]

Переваги використання Entity Framework в порівнянні з іншими видами зберігання даних включають:

1. Зручна робота з даними: Entity Framework надає високорівневий інтерфейс для взаємодії з базами даних, що спрощує розробку додатків. Розробники можуть працювати з даними за допомогою об'єктно-орієнтованої моделі, замість написання складних SQL-запитів.

2. Автоматична генерація SQL-запитів: Entity Framework автоматично генерує SQL-запити на основі виразів LINQ (Language-Integrated Query) або методів запитів. Це дозволяє розробникам зосередитись на бізнес-логіці, а не на написанні складних запитів.

3. Маппінг об'єктів на таблиці бази даних: Entity Framework надає можливість визначити відповідність між об'єктами.

4. Підтримка різних баз даних: Entity Framework підтримує різні системи управління базами даних, такі як SQL Server, MySQL, PostgreSQL, Oracle і багато інших. Це дозволяє розробникам працювати з різними базами даних без необхідності вивчати конкретні деталі реалізації кожної з них.

5. Повторне використання коду: Entity Framework дозволяє розробникам створювати загальні моделі даних, які можна використовувати в різних додатках. Це дозволяє ефективно повторно використовувати код і скорочує час розробки нових додатків.

6. Легка міграція бази даних: Entity Framework надає механізми міграції, що дозволяють зручно здійснювати зміни в схемі бази даних без необхідності вручну писати та виконувати SQL-скрипти. Це дозволяє зберігати структуру бази даних синхронізованою з моделлю даних.

7. Тестування: Entity Framework дозволяє легко виконувати тестування додатків, оскільки розробники можуть використовувати мок-об'єкти або фейкові бази даних для тестування різних сценаріїв без необхідності взаємодії з реальною базою даних.

8. Підтримка різних підходів до моделювання даних: Entity Framework підтримує як Code-First, так і Database-First підходи до розробки. Це дозволяє розробникам вибрати найбільш зручний спосіб роботи з даними, відповідний вимогам проєкту. [23] [24]

Entity Framework - це потужний інструмент для роботи з базами даних, який полегшує розробку додатків, спрощує взаємодію з даними та пропонує багато корисних функцій і переваг. Entity Framework пропонує автоматизований контроль доступу до даних, створення SQL-запитів і простіше управління базами даних, що дозволяє розробникам зосередитися на бізнес-логіці своїх додатків. Entity Framework перетворюється на потужний інструмент для розробників, які працюють зі сховищами даних, завдяки підтримці численних баз даних, простій міграції та повторному використанню коду.

Незалежно від способу моделювання даних, Entity Framework надає розробникам гнучкість і розширюваність, необхідні для ефективної роботи з базами даних. Розробники можуть швидко створювати, змінювати та взаємодіяти з базами даних у простий та ефективний спосіб, використовуючи Entity Framework у поєднанні з C# та Visual Studio.

Entity Framework вважається однією з найпопулярніших ORM-технологій для взаємодії з базами даних в середовищі .NET. Для розробників, які використовують сховища даних у своїх проєктах, її продуктивність,

адаптивність та багатофункціональність роблять її незамінним інструментом. [25]

Тому Entity Framework, SQL та бази даних є важливими інструментами для розробки програм. Вони надають вам можливість отримувати та зберігати велику кількість інформації в організованому вигляді. За допомогою надійної мови запитів SQL ви можете виконувати ряд операцій з даними, включаючи вибірку, додавання, оновлення та видалення. Одним з найефективніших ORM-рішень для впорядкування операцій з базами даних є Entity Framework. Загалом, бази даних, SQL та Entity Framework є чудовими технологіями, які допомагають розробникам легко і швидко створювати надійні та ефективні додатки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКА

3.1 Опис етапів розробки програмного забезпечення

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

1. Збір вимог: Першим кроком у процесі розробки програмного забезпечення є збір вимог до системи. Цей етап охоплює вивчення потреб і вимог потенційних користувачів системи.

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 взаємозв'язки між ними. Низькорівневий дизайн концентрується на конкретних деталях реалізації модулів.

3. Розробка: Після завершення проєктування, іде перехід до етапу розробки, де виконуються всі необхідні компоненти програмного забезпечення. Розробка може включати написання коду, створення баз даних, розробку інтерфейсу користувача та інші технічні аспекти реалізації системи.

4. Тестування: Після завершення розробки програмне забезпечення потребує тестування для виявлення та усунення можливих помилок, багів і недоліків. Цей етап включає ручне та автоматизоване тестування системи, а також валідацію функціональності та продуктивності.

3.1.1 Опис вимог до програмного забезпечення

Перед початком розробки програмного забезпечення були визначені основні вимоги до системи підтримки торгівлі криптоактивами. Вимоги були визначені на основі детального аналізу потреб користувачів та вимог ринку. Основними вимогами до програмного забезпечення є:

1. Автоматичне виявлення міжбіржових та внутрішньобіржових торгових сигналів.
2. Періодичне оновлення списку торгових зв’язків.
3. Висока швидкодія для забезпечення оперативності.
4. Стабільність системи для безперебійності роботи.

3.2 Реалізація

Система була реалізована з використанням модульного підходу, тобто була розділена на ряд невеликих модулів. Такий підхід спростив розробку та підтримку системи.

Дані представляють список об’єктів, де кожен об’єкт має такі поля:

-Pair (назва торгової пари).

-Exchanger (Назва біржі до якої відноситься ця пара).

-Buy (Ціна та об’єм замовлень на покупку).

-Sell (Ціна та об’єм замовлень продаж).

Після виклику, починається робота об’єктів виявлення міжбіржових та внутрішньобіржових торгових сигналів в асинхронному режимі (рис. 3.1). Обидва об’єкти займаються виявленням можливих зв’язків (кожен у своєму напрямку), а під час формування кожного зв’язка, розраховується прибутковість (profit). Далі, перед тим, як надати дані для інших вузлів системи, дані проходять етап сортування та фільтрації для відсіювання низьколіквідних пар. Повна структура класів візуалізована на рисунку 3.2.

Рисунок 3.1 — Діаграма діяльності модуля

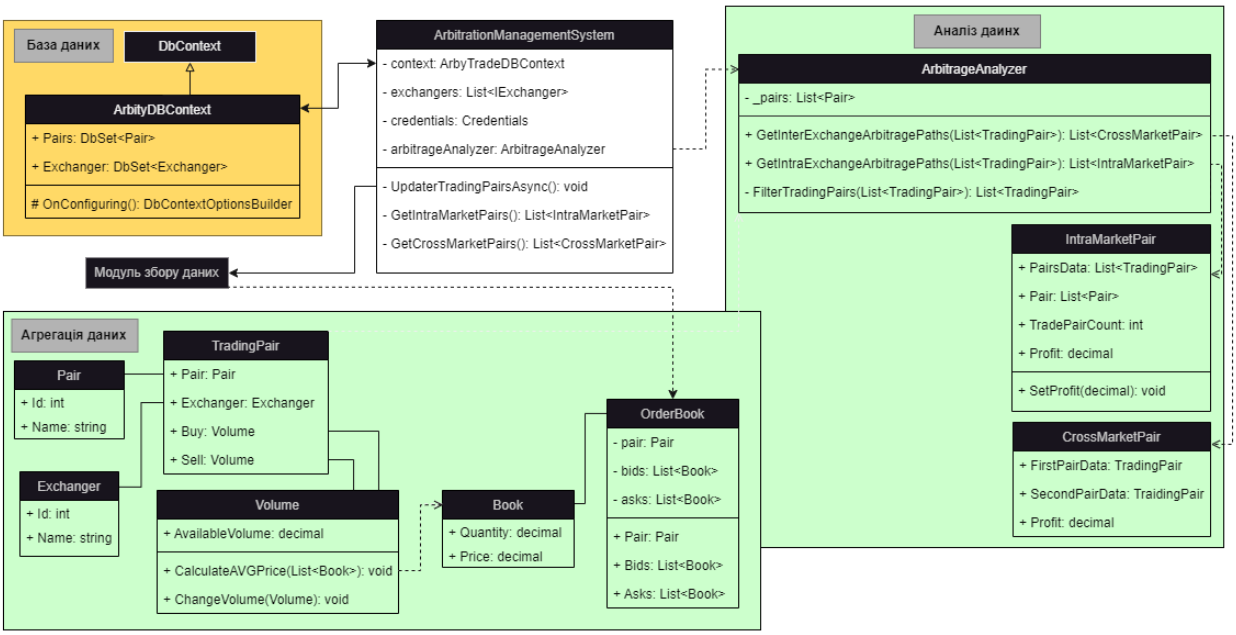


Рисунок 3.2 - Діаграма класів

3.2.1 Розрахування об’ємів.

Після агрегації даних, система переходить до обчислення торговельних можливостей для кожної пари. В цьому процесі враховуються обсяги торгів, що відбулися протягом останніх 24 годин, а також розраховується середня ціна обсягу за певний період часу.

Результатом обчислення обсягів є визначення торговельних можливостей для кожної пари з урахуванням обсягів торгів, що відбулися протягом останніх 24 годин, а також середньої ціни обсягу за визначений проміжок часу. Ці дані мають велике значення для подальшого аналізу та прийняття рішень щодо торговельної стратегії.

Після отримання результатів обчислення торговельних можливостей, система використовує ці дані для подальшого аналізу та прийняття рішень. Вона може виявити потенційно прибуткові торговельні ситуації, де різниця в цінах між парами має потенціал для отримання прибутку. Це дозволяє трейдерам розробляти ефективні торговельні стратегії, враховуючи обсяги торгів та середню ціну обсягу, що отримані в результаті обчислень (рис. 3.3).

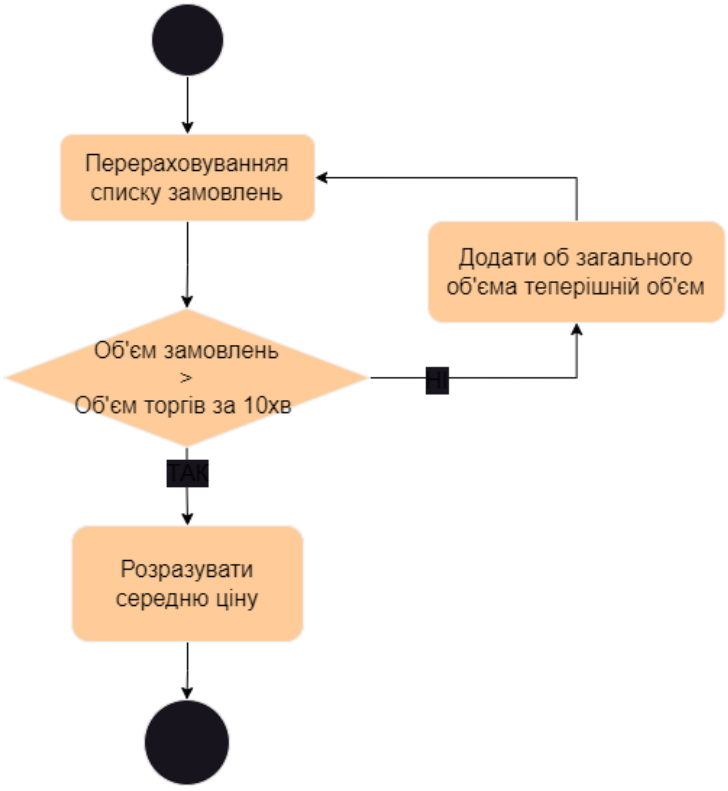


Рисунок 3.3 — Діаграма діяльності розраховування об’ємів

Отже, процес обчислення торговельних можливостей після агрегації даних є важливим кроком у підготовці до торгівлі, оскільки він надає об'єктивну інформацію про потенційні прибуткові можливості на ринку.

3.2.2 Система пошуку міжбіржових торгових сигналів

Система пошуку міжбіржових торгових сигналів виконує завдання виявлення можливостей отримання прибутку з різниці в цінах між біржами для двох монет (рис. 3.4).

Рисунок 3.4 — Демонстрації роботи на прикладі реальних даних не великої вибірки

Ця система починає свою роботу після отримання вхідних даних з модуля аналізу та агрегації даних, включаючи поточні ціни на всі активи з кожної біржі.

Алгоритм системи включає наступні кроки:

1. Вибір списку пар однакового типу: З вхідного списку торгових пар система вибирає список пар, що належать до одного типу.
2. Аналіз та вибір найбільш прибуткового варіанта: Для кожного типу пар система аналізує список і вибирає найбільш прибуткову можливість для торгівлі. Це включає порівняння ціни активу на різних біржах та визначення оптимального часу для здійснення операцій.
3. Додавання до результату: Вибрані прибуткові варіанти додаються до результату аналізу.
4. Фільтрація та відсортування низьколіквідних пар: На завершення процесу дані проходять через вузол фільтрації та відсортування для вилучення

низьколіквідних пар, які можуть бути менш придатними для торгівлі через недостатню ліквідність на ринку.

Таким чином, система пошуку міжбіржових торгових сигналів виконує аналіз та вибір найбільш прибуткових можливостей зі списку пар однакового типу, а також фільтрацію та відсортування пар на основі їх ліквідності. Це допомагає знайти потенційно вигідні можливості для міжбіржової торгівлі.

3.2.3 Система пошуку внутрішньобіржових торгових сигналів

Система пошуку внутрішньобіржових торгових сигналів виконує завдання виявлення можливостей отримання прибутку шляхом використання різниці в цінах всередині однієї біржі. Ця система розпочинає свою роботу після отримання вхідних даних з модуля аналізу та агрегації даних.

Розглянемо детальніше кроки, що виконує система:

1. Генерація розширеного списку біржових пар: Система генерує розширений список пар, який включає 3-6 зв'язків, для подальшого аналізу внутрішньобіржових взаємодій. Це допомагає виявити потенційно вигідні можливості для торгівлі всередині однієї біржі.

2. Покроковий вибір зв'язку та розрахунки прибутку: Система почергово аналізує кожен зв'язок зі списку та проводить розрахунки потенційного прибутку. Це включає порівняння цін на активи, виявлення можливих арбітражних сигналів та визначення оптимального часу для здійснення операцій.

3. Звуження вибірки торгових сигналів: Після пошуку всіх доступних торгових сигналів, система звужує вибірку до вхідних значень, відкидаючи менш придатні або менш потенційно вигідні можливості.

4. Вузол фільтрації та відсортування низьколіквідних пар: На завершення процесу, дані проходять через вузол фільтрації та відсортування, який призначений для вилучення низьколіквідних пар. Це допомагає

сконцентруватися на більш ліквідних активах та забезпечує більш оптимальні умови для торгівлі (рис. 3.5).

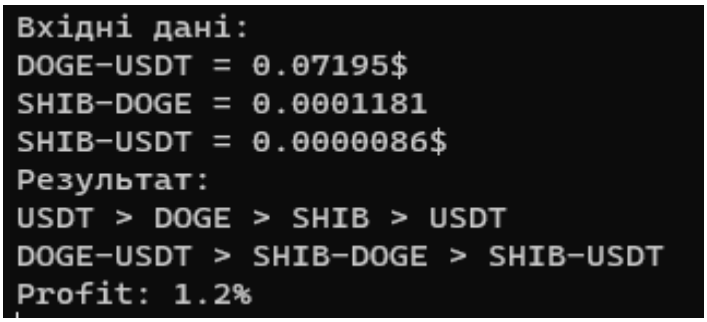


Рисунок 3.5 — Демонстрація роботи на прикладі реальних невеликої вибірки

Таким чином, система пошуку внутрішньобіржових торгових сигналів включає генерацію розширеного списку біржових пар, покроковий аналіз та розрахунки прибутку, звуження вибірки торгових сигналів та фільтрацію низьколіквідних пар для забезпечення більш ефективної торгівлі всередині біржі.

3.2.4 Вузол фільтрації низьколіквідних торгових пар

Система фільтрації виконує важливу роль у відборі торгових пар на основі різних критеріїв, таких як обсяг торгів та ліквідність ринку. Низьколіквідні торгові пари не є перспективними для торгівлі, оскільки їх купівля та продаж може бути проблематичним.

Робота системи полягає у зборі даних з різних бірж, що включають поточні ціни та обсяги торгів для всіх активів на кожній біржі. Далі система використовує спеціальні алгоритми для визначення торгових пар, які мають низьку ліквідність протягом певного періоду часу. Якщо певна пара відповідає встановленим параметрам, вона включається до фінального списку міжбіржових торгових сигналів.

Отже, система фільтрації виконує функцію фільтрування торгових пар на основі обсягу торгів та ліквідності ринку, збираючи дані з різних бірж та застосовуючи алгоритми для відбору пар з низькою ліквідністю.

3.2.5 Робота з базою даних

ASP.NET надає зручні інструменти, такі як Entity Framework, для спрощення роботи з базою даних. База даних зберігає, як клієнтські дані, так і дані про біржі та пари, які потрібно переглядати.

Етапи роботи:

1. Підключення до бази даних: У веб-додатку налаштовується підключення до бази даних, використовуючи рядок підключення, який містить інформацію про сервер бази даних, ім'я бази даних, автентифікаційні дані тощо.

2. Визначення моделей: Створюються класи-моделі, що відображають сутності бази даних. Кожен клас-модель представляє таблицю, а властивості класу відображають стовпці цієї таблиці.

3. Мапування моделей: За допомогою атрибутів або конфігураційних файлів встановлюються зв'язки між моделями та таблицями бази даних, а також визначаються правила мапування для кожного стовпця.

4. Створення контексту бази даних: Створюється клас, що наслідується від класу DbContext, який представляє контекст бази даних. У цьому класі визначаються набори моделей, які будуть використовуватися для взаємодії з базою даних.

5. Операції з базою даних: За допомогою контексту бази даних можна виконувати різні операції, такі як додавання, оновлення, видалення записів або виконання запитів до бази даних. Наприклад, можна додати новий запис до таблиці, оновити існуючий запис або виконати запит для отримання даних з бази даних.

6. Збереження змін: Після внесення змін до моделей, що відбулися під час взаємодії з базою даних, необхідно зберегти ці зміни. Це робиться викликом методу SaveChanges() у контексті бази даних.

3.3 Автоматизоване тестування продукту

Для тестування системи було використано автоматизоване тестування продукту з використанням Unit тестів. Unit тести - це метод тестування, який порівнює очікувані вихідні дані з фактичними, результатами обчислень об'єкта є міжбіржові зв'язки (рис. 3.6) та внутрішньобіржові зв'язки (рис. 3.7).

```
Вхідні дані:
A-B      Exchanger1      Buy: 1000$      Sell: 990$
A-C      Exchange1       Buy: 10$        Sell: 9$
A-B      Exchanger2      Buy: 980$       Sell: 970$
A-C      Exchange2       Buy: 11$        Sell: 10$
A-B      Exchanger3      Buy: 1010$      Sell: 1005$
A-C      Exchange3       Buy: 11$        Sell: 10.5$
Результат:
A-C      exchanger1->exchanger2  Buy:10$         Sell:10.5$     | 5%
A-B      Exchanger2->Exchanger3  Buy:980$        Sell:1005$     | 2.55%
TRUE
```

Рисунок 3.6 — Unit тестування алгоритму міжбіржового арбітражу

```
Вхідні дані:
A-B      Buy: 10000      Sell: 10000
A-C      Buy: 0.2        Sell: 0.2
C-B      Buy: 2000       Sell: 2000
Результат:
A > B > C > A
A-B > C-B > A-C
Profit: 0%

B > A > C > B
A-B > A-C > C-B
Profit: 0%
```

Рисунок 3.7 — Unit тестування алгоритму Внутрішньобіржового арбітражу

Якщо отримані дані збігаються з очікуваними, це означає, що алгоритм розрахований вірно.

У процесі автоматизованого тестування продукту було створено Unit тести, які перевіряють правильність роботи окремих компонентів або функцій системи. Ці тести виконуються автоматично і містять набір вхідних даних, на основі яких, система проводить розрахунки. Результати обчислень порівнюються з очікуваними значеннями, визначеними заздалегідь. Якщо всі вихідні дані відповідають очікуваним результатам, тест вважається пройденим, і це свідчить про правильну реалізацію алгоритмів системи.

Отже, використання автоматизованого тестування з використанням Unit тестів дозволило перевірити коректність роботи системи шляхом порівняння фактичних результатів з очікуваними значеннями, що підтверджує правильність реалізації алгоритмів системи.

3.3.1 Моніторинг активності

У рамках автоматизованого тестування системи буде здійснюватися постійний моніторинг її діяльності з метою перевірки стабільності та продуктивності. Цей процес дозволяє виявити та проаналізувати можливі проблеми, помилки та недоліки продукту.

Моніторинг діяльності системи зосереджується на регулярному спостереженні за її роботою, зборі та аналізі важливих метрик, таких як: час відгуку, завантаження ресурсів та швидкість виконання операцій. Це дозволяє виявляти можливі проблеми з продуктивністю та забезпечує стабільну роботу системи під великим навантаженням.

Крім того, автоматизоване тестування спрямоване на виявлення критичних багів, пов'язаних з синхронізацією модулів системи. Це включає виявлення та усунення проблем, пов'язаних зі збереженням та передачею даних між різними компонентами системи.

В результаті проведених тестів та аналізу, були виявлені критичні баги, пов'язані з синхронізацією модулів, які були виправлені. Крім того, була

проведена оптимізація алгоритму роботи міжбіржового та внутрішньобіржового арбітражу з метою покращення продуктивності системи. Результатом цих змін стали більш продуктивні алгоритми, які були описані в пунктах 3.2.2 та 3.2.3.

ВИСНОВКИ

Метою даної роботи стало розроблення модуля аналізу даних за допомогою розробленої системи підтримки торгівлі мовою C#.

Для досягнення постановленої мети в процесі роботи виконано наступні завдання:

1. Розроблено модуль підтримки системи торгівлі, що дозволяє автоматизувати процес аналізу даних.
2. Розроблено підсистему збереження даних.
3. Розроблено підсистему забезпечення надійності та захисту даних.
4. Розроблено циклічну систему з паралельним обчисленням.
5. Протестовано продукт на функціональність та відповідність вимогам.
6. Було проведено Unit тестування програмного продукту, що дозволило усунути помилки та оптимізувати алгоритм роботи.

Завдяки чіткому виконанню завдань, поставлених на початку роботи, в результаті було створено повноцінну частину для аналізу даних з торгового ринку, яка готова до використання в реальних умовах.

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. What Is SQL? Definition, Elements, Examples, and Uses. [Електронний ресурс]. – Режим доступу: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-sql/

19. Structured Query Language (SQL). [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/searchdatamanagement/definition/SQL

20. SQL vs. NoSQL Databases. [Електронний ресурс]. – Режим доступу: https://www.ibm.com/cloud/blog/sql-vs-nosql

21. The Ultimate Comparison of ADO.NET and Entity Framework. [Електронний ресурс]. – Режим доступу: https://blog.devart.com/ado-net-vs-entity-framework.html

22. What is Entity Framework? [Електронний ресурс]. – Режим доступу: https://www.interviewbit.com/entity-framework-interview-questions/

23. Introduction to Entity Framework. [Електронний ресурс]. – Режим доступу: https://www.partech.nl/nl/publicaties/2020/11/introduction-to-entity-framework

24. What are the advantages of using an entity framework? [Електронний ресурс]. – Режим доступу: https://www.quora.com/What-are-the-advantages-of-using-an-entity-framework-over-a-database

25. Entity Framework. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Entity_Framework

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 аналізу даних мовою C#

Виконав студент 4 курсу
групи ПД-43
Яковчук Василь Адрійович
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 процесу аналізу даних за допомогою розробленої системи підтримки торгівлі мовою C#.
- **Об’єкт дослідження -** процес виявлення торгових сигналів на криптовалютному ринку.
- **Предмет дослідження -** модуль аналізу даних мовою C# для аналізу історичних цінових даних та виявлення торгових можливостей.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати предметну область серверної розробки.
2. Створити та налаштувати базу даних та створити з’єднання.
3. Розробити модель для розрахунку доступного обсягу монет криптовалютної пари.
4. Розробити алгоритм виявлення внутрішньобіржових пар.
5. Розробити алгоритм виявлення міжбіржових пар.
6. Розробити фільтрацію низьколіквідних та низькоприбуткових пар.
7. Провести тестування серверної сторони.

3

АНАЛІЗ АНАЛОГІВ

	CoinArbitrageBot	Cryptohopper	Coingecko	**Arbity**
Платформи	Windows, Linux	Windows, Linux	Web	**Web**
Рівень складності	Складний	Складний	Простий	**Простий**
Внутрішньобіржовий арбітраж	Відсутній	Присутній	Відсутній	**Присутній**
Міжбіржовий арбітраж	Присутній	Відсутній	Присутній	**Присутній**
Інтервал оновлення інформації	~3хв	~10хв	~5хв	**60с**
Врахування ліквідності	Немає	Немає	Немає	**Є**
Автоторгівля	Присутня	Присутня	Відсутня	**Відсутня**

4

ВИМОГИ ДО ДОДАТКУ

Функціональні:

1. Автоматичне виявлення міжбіржових торгових сигналів.
2. Автоматичне виявлення внутрішньобіржових торгових сигналів.
3. Періодичне оновлення списку торгових зв'язків.

Нефункціональні:

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

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

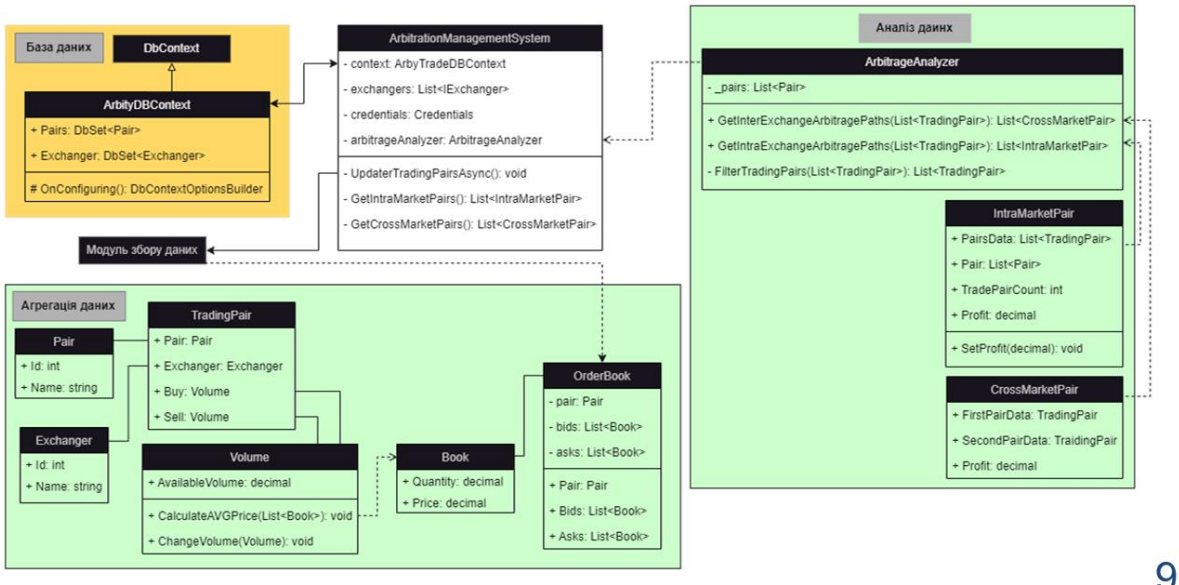
7

ДІАГРАМА ДІЯЛЬНОСТІ

Пошук міжбіржових та внутрішньобіржових арбітражних зв’язків

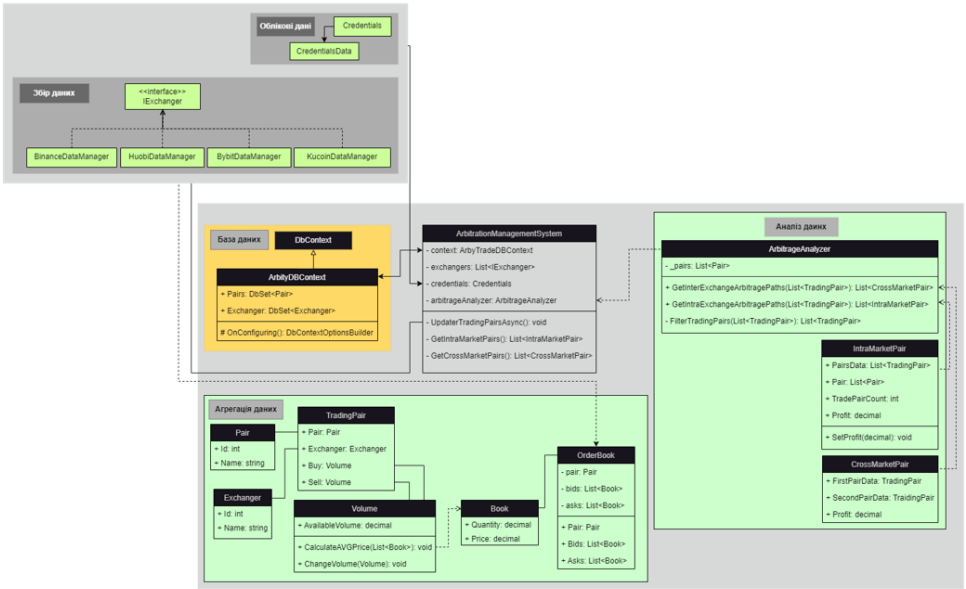
8

ДІАГРАМА КЛАСІВ



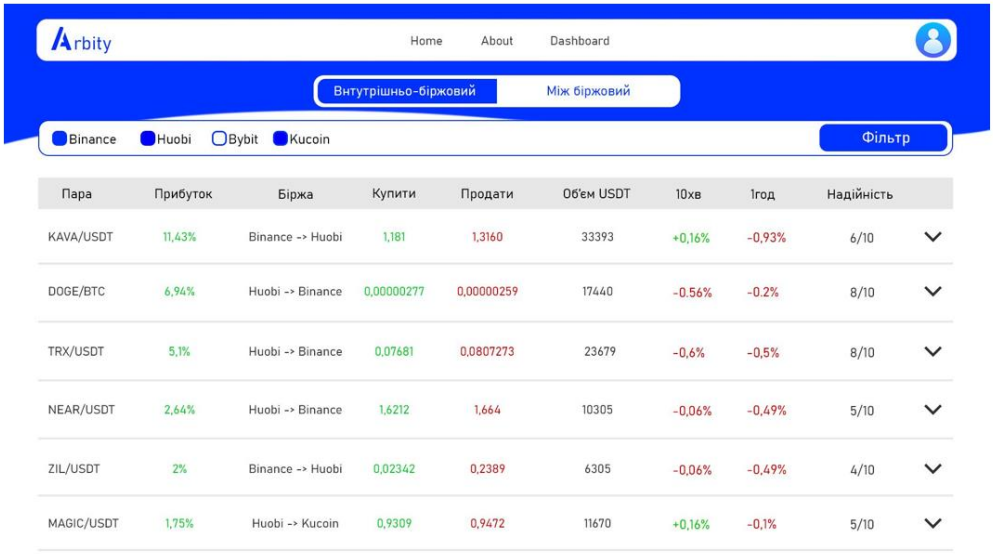
9

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. Яковчук В. А. / Arbity / Яковчук В.А // Університетський науково-технічний хакатон "SMART SMART IT 2023", 25.05 2023р., ДУТ. м. Київ - К. ДУТ, 2023, Сертифікат за участь Яковчук Василь Андрійович в категорії "IoT".
2. Яковчук В.А / РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ПІДТРИМКИ ТОРГІВЛІ КРИПТОАКТИВАМИ. СПЕЦ. ЧАСТИНА: РОЗРОБКА МОДУЛЯ АНАЛІЗУ ДАНИХ КРИПТОАКТИВІВ МОВОЮ C# / Жебка В.В, Яковчук В.А // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії, 01-03 червня 2023р., ДУТ, м. Київ - К. ДУТ, 2023.

ВИСНОВКИ

В процесі виконання дипломної роботи розроблено програмне забезпечення для автоматизованого пошуку арбітражних зв’язків на мові програмування C#.

1. Розроблено модуль підтримки системи торгівлі, що дозволяє автоматизувати процес аналізу даних.

2. Розроблено підсистему збереження даних.

3. Розроблено підсистему забезпечення надійності та захисту даних.

4. Розроблено циклічну систему з паралельним обчисленням.

6. Було проведено Unit тестування програмного продукту, що дозволило усунути помилки та оптимізувати алгоритм роботи.

13

ДЯКУЮ ЗА УВАГУ!

Додаток Б

Пошук міжбіржових зв'зяків

```
public void GetInterExchangeArbitragePaths(List<TradingPair> newPairsData)
        {
```

```
            crossMarketPairs = new List<CrossMarketPair>();
            foreach (Pair currentPair in _pairs)
            {
                List<TradingPair> currentPairsData = newPairsData.Where(p => p.Pair.Id == currentPair.Id).ToList();
                TradingPair minBuyPair = null;
                TradingPair maxSellPair = null;
                foreach (TradingPair firstPairData in currentPairsData)
                {
                    foreach (TradingPair secondPairData in currentPairsData)
                    {
                        decimal profit = firstPairData.Sell.AveragePrice - secondPairData.Buy.AveragePrice;
                        if (firstPairData.Exchanger.Id != secondPairData.Exchanger.Id && profit > 0 && ((minBuyPair == null && maxSellPair == null)
                            || profit > maxSellPair.Sell.AveragePrice - minBuyPair.Buy.AveragePrice))
                        {
                            minBuyPair = secondPairData;
                            maxSellPair = firstPairData;
                        }
                    }
                }
                if (minBuyPair != null && maxSellPair != null)
                {
                    crossMarketPairs.Add(new CrossMarketPair(minBuyPair, maxSellPair));
                }
            }
```

```
    }
```

Пошук внутрішньобіржових зв'язків

```
public void GetIntraExchangeArbitragePaths(List<TradingPair> pairsData)
    {
        intraMarketPairs = new List<IntraMarketPair>();
        List<TradingPair> expandedPairsData = new List<TradingPair>();
        foreach (TradingPair pairData in pairsData)
        {
            expandedPairsData.Add(pairData);
            string[] pairName = pairData.Pair.Name.Split('-');
            TradingPair modifiedPairData = new TradingPair
            {
                Pair = new Pair
                {
                    Name = $"{pairName[1]}-{pairName[0]}"
                },
                Exchanger = new Exchanger
                {
                    Name = pairData.Pair.Name,
                },
                Buy = new Volume
                {
                    AveragePrice = pairData.Buy.AveragePrice
                },
                Sell = new Volume
                {
                    AveragePrice = pairData.Sell.AveragePrice
                }
```

```
                };
                expandedPairsData.Add(modifiedPairData);
            }
            foreach (TradingPair firstPairData in expandedPairsData)
            {
                string[] firstNameOfCoin = firstPairData.Pair.Name.Split('-');
                List<TradingPair> secondPairsData = GetRequiredPairs(expandedPairsData, firstNameOfCoin[1]);
                foreach (TradingPair secondPairData in secondPairsData)
                {
                    string[] secondNameOfCoin = secondPairData.Pair.Name.Split('-');
                    List<TradingPair> thirdTradingPairs = GetRequiredPairs(expandedPairsData, secondNameOfCoin[1], firstNameOfCoin[0]);
                    foreach (TradingPair thirdPairData in thirdTradingPairs)
                    {
                        intraMarketPairs.Add(new IntraMarketPair(new List<TradingPair> { firstPairData, secondPairData, thirdPairData }));
                    }
                }
            }
        }
```

Обрахунок Об'ємів

```
        private void CalculateAVGPrice(List<Book> orders, decimal currentVolume = 10000)
        {
            List<decimal> quantites = new List<decimal>();
            foreach (var order in orders)
```

```
            {
                if(AvailableVolume > currentVolume)
                {
                    break;
                }
                quantites.Add(order.Quantity);
                AvailableVolume += order.Quantity * order.Price;
            }
            AveragePrice = AvailableVolume / quantites.Sum();
        }
```

З'язок з базою даних

```
class ArbyTradeDBContext : DbContext
    {
        public DbSet<Pair> Pairs { get; set; }
        public DbSet<Exchanger> Exchangers { get; set; }
        public ArbyTradeDBContext()
        {
            Pairs.ToList();
            Exchangers.ToList();
        }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\Development\Arbity\ArbyBackSystem\ArbyTradeDB.mdf;Integrated Security=True");
        }
    }
```