МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОЙ СИСТЕМЫ ДЛЯ АНАЛИЗА КРИПТОВАЛЮТНОГО РЫНКА: СЕРВЕРНАЯ ЧАСТЬ И АЛГОРИТМЫ АНАЛИЗА АРБИТРАЖНЫХ СИТУАЦИЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Севастьянова Владимира Александровича

Научный руководитель
доцент _____ Б. А. Филиппов

Зав.кафедрой,
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С развитием IT-технологий сформировалась альтернатива банковской системе — криптовалютный рынок, с каждым годом набирающий всё большую популярность. Как и любая крупная и сложная система, он имеет свои особенности, преимущества и недостатки. И из таких недостатков системы, как, например, разница курсов между одними и теми же валютными парами на разных платформах в один и тот же момент времени, или особенностей, проявляющихся в том числе в виде зависимости одних курсов от других, пытаются извлечь выгоду трейдеры. Первое из явлений получило название арбитраж, а второе — корреляционный анализ. Оба рода деятельности пользуются популярностью и являются востребованными как среди специалистов в этой области, так и среди начинающих трейдеров или предпринимателей.

Но с развитием рынка, увеличением числа бирж и валют на них, ручной анализ становится всё более затруднительным и малоэффективным, а также повышается риск допустить различного рода ошибки, обусловленные человеческим фактором. Немаловажным аспектом является и скорость принятия решения, а в условиях роста конкуренции этот фактор и вовсе становится критическим, что, в свою очередь, может привести к ещё большему числу ошибок, необдуманных действий и стрессу. Таким образом, проблема автоматизации описанных аналитических процессов и их компоновки в единый удобный программный инструмент является актуальной.

Целью дипломной работы является реализация серверной части и алгоритмов анализа арбитражных ситуаций, обеспечивающих оперативную доставку актуальных данных до пользователя, в рамках разработки клиент-серверной системы для анализа криптовалютного рынка. Для достижения цели были поставлены следующие задачи:

1. изучение и анализ предметной области и существующих решений, выработка требований к приложению и его отличительных черт;
2. выбор стека технологий и инструментов для разработки;
3. проектирование архитектуры системы и базы данных;
4. разработка алгоритма нахождения арбитражных цепочек с учётом ликвидности;
5. разработка оркестратора периодических процессов;
6. проектирование API и реализация backend части приложения.

1 Рассмотрение предметной области и существующих аналогов

1.1 Предметная область

1.1.1 Криптовалюты и блокчейны

Криптовалюта — цифровая или виртуальная форма денег, использующая криптографию для обеспечения безопасных транзакций и управления созданием новых единиц. Она не имеет физического аналога, и её ценность и собственность управляются распределенным реестром, таким как *блокчейн*.

Блокчейн — это распределенная база данных, которая записывает транзакции через сеть компьютеров. Вместо централизованного хранения данных блокчейн использует концепцию *децентрализации*, где копии базы данных хранятся на множестве устройств, называемых узлами. Эти узлы работают вместе для подтверждения и записи новых транзакций в блоки, образуя из них цепь, известную как блокчейн. Работа устройств, решающих криптографические задачи для обеспечения работы блокчейна, оплачивается криптовалютой, взимаемой в качестве комиссии за совершение транзакции.

Сеть является инфраструктурой, которая поддерживает функционирование блокчейна и криптовалют, включая в себя все узлы или компьютеры, которые участвуют в этом процессе. При этом сеть может поддерживать сразу несколько блокчейнов.

1.1.2 Криптовалютные кошельки и биржи

Криптовалютный кошелек — это цифровой инструмент, который позволяет пользователям хранить, отправлять и получать криптовалюты и предоставляют инструменты для взаимодействия с блокчейном.

Криптовалютная биржа (криптобиржа, биржа) — это платформа, на которой пользователи могут покупать, продавать, обменивать криптовалюты, а также торговать традиционными валютами против криптовалют. Существуют два основных типа бирж: централизованные (CEX) и децентрализованные (DEX); но в рамках проекта используются только первые.

Централизованные биржи (CEX) — управляются централизованными организациями, которые *контролируют средства пользователей*, обеспечивают механизмы безопасности, обработку транзакций и обслуживание клиентов.

1.1.3 Биржевой стакан и spread

При торговле на бирже пользователь взаимодействует с *биржевым стаканом*.

Биржевой стакан (order book) — это список всех открытых ордеров (заявок) на покупку и продажу криптовалют на бирже. Каждый ордер содержит информацию о цене и количестве актива, который кто-то хочет купить или продать. Предложения в стакане разделены на две группы:

1. **Ордеры на продажу (asks)** — верхняя, красная часть стакана, в ней представлены ордеры на продажу актива, упорядоченные по возрастанию цены.
2. **Ордеры на покупку (bids)** — нижняя, зелёная часть стакана, в ней представлены ордеры на покупку актива, упорядоченные по убыванию цены.

Курс криптовалюты — текущая цена, по которой криптовалюта может быть куплена или продана на бирже за другую валюту. Курсы могут изменяться в реальном времени и зависят от спроса и предложения на рынке.

Разрыв цен (spread) — минимальное расстояние между текущим спросом и предложением, иначе говоря, разница между самым низким аском и самым высоким бидом.

1.2 Межбиржевой арбитраж и проблемы его автоматизации

Межбиржевой арбитраж криптовалюты — это процесс покупки криптовалюты на одной бирже по более низкой цене и последующей её продажи на другой бирже по более высокой цене в как можно более короткий срок для получения прибыли от разницы в ценах.

Арбитражная цепочка — замкнутая упорядоченная последовательность переводов одной валюты в другую посредством покупок и продаж, а также переводов с одной биржи на другую, в результате которой можно увеличить количество изначальной валюты.

Несмотря на простоту идеи, в стратегии межбиржевого арбитража есть важные тонкости, которые нужно учитывать при автоматизации, и игнорирование которых может привести к неверным результатам или серьёзным проблемам в их использовании:

- ликвидность;
- совпадение имён валют;
- совпадение сетей валют;

— время выполнения межбиржевого перевода;
— ввод/вывод интересующей валюты на торгуемых биржах;
— комиссии.

1.2.1 Ликвидность

Ликвидность — это мера того, насколько легко можно купить или продать актив на рынке без значительного воздействия на его цену. Высокая ликвидность означает, что актив можно легко обменивать без больших изменений цены, в то время как низкая ликвидность может привести к более крупным изменениям цен при совершении сделок. Криптовалюты с большим объемом торгов обычно имеют более высокую ликвидность.

Отсюда получается, что, с одной стороны, спреды, на которых построен арбитраж, будут чаще возникать на низколиквидных курсах, а с другой стороны, чтобы этот спред реализовать, нужна как можно более высокая ликвидность.

1.2.2 Совпадение имён валют

Основным критерием для сопоставления валют между собой является их *тикер*.

Тикер — это уникальное буквенное обозначение определенного актива, которое используется для удобства торговли и идентификации на рынке.

Однако существуют случаи, когда тикеры различных валют могут совпадать. Поэтому важно смотреть не только на тикер валюты, но и на её полное имя, а также сеть, которой она принадлежит.

1.2.3 Совпадение сетей валют

Существует ещё одна, не менее распространённая ситуация, когда одна и та же валюта одного и того же проекта на разных биржах торгуется в разных сетях. В таком случае, перевод купленной на одной бирже валюты на другую является либо невозможным, либо сильно затруднённым.

1.2.4 Время выполнения межбиржевого перевода

Чем дольше трейдер "прогоняет" цепочку, тем больше вероятность, что спред будет урегулирован до него. А самыми долгими операциями в цепочках являются межбиржевые переводы, которые могут занимать минуты времени и

выполняться тем дольше, чем выше загруженность сети. И так как оценка загруженность сети — задача нетривиальная, то наиболее верный путь — сокращение числа межбиржевых переходов в цепочке.

1.2.5 Ввод/вывод интересующей валюты на торгуемых биржах

На возникновение арбитражных ситуаций реагируют не только трейдеры, но и сами криптобиржи, не желающие работать себе в убыток. Ведь когда кто-то приобретает, другой теряет, в данном случае — биржи. В результате этого биржи пытаются препятствовать реализации спреда посредство следующих ограничений:

- закрытие ввода/вывода того или иного актива;
- взимание крупной комиссии за совершении.

1.2.6 Комиссии

За совершение сделок в биржевом стакане или перевод средств между биржами взимается комиссия, и если первой можно в большинстве случаев (в зависимости от сети) пренебречь, ввиду их незначительности, то второй — не стоит, особенно в случае небольших спредов. Тогда можно не только потерять значительную часть прибыли, но и вовсе прогнать цепочку себе в убыток.

1.3 Аналоги

1.3.1 Сканер Arbitoring

Онлайн-платформа «Arbitoring» представляет собой сканер связок для арбитража на различных платформах. Несмотря на большой охват бирж, обменников и банков, а также удобные инструменты анализа данных, имеет серьёзные недостатки, такие как высокая задержка при предоставлении и обновлении цепочек и отсутствие учёта всех вышеперечисленных тонкостей.

1.3.2 Платформа HaasOnline

Англоязычная платформа «HaasOnline» используется для алгоритмической торговли и арбитража на криптовалютных рынках. Нацелена на опытных трейдеров, так как требует знание алгоритмического рынка и опыт в программировании, необходимый для раскрытия всех возможностей системы.

1.3.3 Платформа Coinigy

Англоязычная платформа «Coinigy» предоставляет доступ ко множеству бирж и инструментов для трейдинга, а также позволяет пользователям отслеживать цены на различных биржах и совершать сделки прямо с платформы. Подойдёт как для новичков, так и для профессионалов, учитывает многие важные особенности рынка, но имеет мало инструментов анализа данных и требует предоставления API-ключей для осуществления работы с биржами.

1.4 Отличительные черты разрабатываемого продукта

Для обеспечения конкурентоспособности, разрабатываемая система должна обладать следующими характеристиками:

- портативность, достигаемая посредством мобильного приложения;
- удобный и понятный интерфейс;
- предоставление информации о курсах с множества бирж;
- низкий порог вхождения;
- частота обновления данных;
- предоставление информации и инструментов для анализа данных;
- кросс-корреляционный анализ;
- учёт принадлежности валют определённым сетям;
- учёт ликвидности;
- учёт открытости ввода/вывода на биржах;
- учёт комиссий;
- отсутствие сбора данных об аккаунтах пользователя на биржах.

Первые три пункта будут выполнены другим разработчиком в смежной дипломной работе.

2 Описание используемых технологий

2.1 СУБД PostgreSQL

PostgreSQL — объектно-реляционная система управления базами данных, широко известная своей надежностью, масштабируемостью, производительностью, лаконичным синтаксисом и поддержкой сложных запросов. Также PostgreSQL активно развивается и поддерживается сообществом разработчиков, имеет обширное комьюнити и большое количество литературы. Для взаимодействия именно с этой СУБД написано много библиотек на языке Python.

2.2 Язык программирования Python

Для разработки серверной части системы использован язык программирования **Python**. Но при работе с ним следует помнить о таких недостатках, как сравнительно *низкая скорость работы* и ограничения, накладываемые на многопоточность механизмом GIL, ограничивающим выполнение Python-кода только одним потоком в данный момент времени, даже если приложение работает на многопроцессорной системе.

2.3 Библиотека asyncio

asyncio — библиотека, предоставляющая средства для асинхронного программирования с использованием синтаксиса асинхронных/ожидаемых ключевых слов `async` и `await`. Позволяет создавать асинхронные функции и корутины, которые могут выполняться параллельно без блокировки основного потока выполнения. Это особенно полезно при работе с сетевыми и I/O-операциями.

2.4 Библиотека aiohttp

aiohttp — библиотека, которая предоставляет инструменты для разработки асинхронных HTTP-серверов и клиентов. Она хорошо интегрируется с Asyncio и позволяет создавать веб-приложения и микросервисы с поддержкой асинхронных HTTP-запросов и ответов.

Для организации большого числа запросов и их обработки можно использовать *асинхронные циклы* — структуры управления выполнением кода, позволяющие запускать и продолжать выполнение I/O-задач, собранных в итерируемом объекте или генераторе, параллельно, не блокируя поток.

2.5 Библиотека asyncpg

Библиотека **asyncpg** представляет собой специализированный асинхронный PostgreSQL-драйвер для Python, оптимизированный для использования с asyncio. Она обеспечивает высокую производительность и является одной из наиболее эффективных асинхронных библиотек для работы с PostgreSQL.

Также библиотека asyncpg обеспечивает защиту от SQL-инъекций за счёт использования подготовленных запросов (prepared statements) и поддержки параметризованных запросов.

2.6 Фреймворк FastAPI и библиотека Pydantic

FastAPI и **Pydantic** являются двумя популярными инструментами в современной разработке веб-приложений на Python, особенно для построения высокопроизводительных серверных решений.

Основные особенности и преимущества FastAPI:

1. **Асинхронность** — фреймворк полностью асинхронен и построен на стандартах ASGI (Asynchronous Server Gateway Interface), что означает, что он может обрабатывать асинхронные запросы и предлагает значительное улучшение производительности при работе с I/O-операциями, такими как запросы к базам данных и вызовы к API.
2. **Автоматическая документация** — FastAPI автоматически генерирует документацию для API (с использованием Swagger и ReDoc), что делает его идеальным выбором для быстрого прототипирования и упрощения командного взаимодействия.

2.7 Стандарт JWT, access и refresh токены, библиотека pyjwt

JWT (JSON Web Tokens) представляют собой механизм безопасной передачи информации между сторонами в виде компактного JSON-объекта, который можно подписать и, при необходимости, зашифровать. Они обычно используются для аутентификации и авторизации в приложениях, а также для обмена информацией между серверами.

Access token — JWT, на основе которого приложение идентифицирует и авторизует пользователя. Обычно имеет сильно ограниченное время жизни (например, минуту). Короткий срок жизни токена уменьшает риск злоупотребления, если токен будет скомпрометирован.

Refresh token — одноразовый токен произвольного формата, служащий для обновления access token, когда последний истекает. Имеет куда более длительное время жизни. Использование refresh token позволяет пользователю оставаться в системе дольше без необходимости повторно вводить учётные данные.

FastAPI не имеет встроенной поддержки JWT, но его можно легко интегрировать с помощью сторонних библиотек, например, pyjwt, и инструментов модуля security фреймворка FastAPI, например, класса OAuth2PasswordBearer.

2.8 Протокол WebSocket, модуль websockets

WebSocket (веб-сокеты) — независимый веб-протокол, который позволяет создавать интерактивное постоянное соединение между сервером и клиентом и обмениваться сообщениями в реальном времени. В отличие от HTTP, веб-сокеты позволяют работать с двунаправленным потоком данных, позволяя обеим сторонам посылать данные в любое время.

Для работы с веб-сокетами можно воспользоваться классом **WebSocket** модуля **fastapi.websockets** и модулем **websockets**, используемым под капотом и разработанным специально для асинхронного взаимодействия с помощью asyncio. Также библиотека поддерживает SSL/TLS.

2.9 Платформа контейнеризации Docker

Docker — это платформа для разработки, доставки и запуска приложений в изолированных контейнерах. Контейнеризация позволяет упаковать приложение со всеми его зависимостями в контейнер, который является легковесной, стандартизированной и автономной единицей, гарантирующей, что приложение будет работать одинаково в любой среде. Также Docker упрощает и автоматизирует процесс развертывания приложений с помощью Dockerfiles, благодаря чему можно легко настроить CI/CD.

3 Разработка серверной части системы и алгоритмов поиска арбитражных цепочек

3.1 Архитектура системы

Пользователь взаимодействует с функционалом системы посредством *мобильного приложения*, реализованного на React.

На сервере развёрнуты:

— Docker-контейнер *algo*, ответственный за составление арбитражных цепочек, кросс-корреляционный анализ и некоторые вспомогательные периодические процессы;

— Docker-контейнер *back*, ответственный за клиентскую часть сервера — backend — на основе FastAPI, предоставляющий API для мобильного приложения и некоторых системных вызовов из контейнера algo;

— *СУБД* — система управления базами данных PostgreSQL, необходимая для хранения данных и нормального функционирования системы.

3.2 Поиск арбитражных цепочек

3.2.1 Подготовка скраперов

В данном разделе описаны абстрактный класс скраперов и его методы, реализуемые наследниками и используемые в последующем, этап формирования скраперов и его особенности, возможность отключения скраперов в БД, а также простота добавления новых скраперов и бирж в систему, достигаемая описанным подходом.

3.2.2 Скрапинг общей информации о спотовых курсах

Скрапинг общей информации с бирж, нужные для получения торгуемых пар валют и курсов на них, а именно лучших бидов и асков, построен на основе асинхронного цикла, как и все остальные этапы скрапинга. Результат скрапинга проходит фильтрацию по белому списку валют, который хранится в БД и нужен для отсечения мусорных валют.

3.2.3 Поиск арбитражных ситуаций

Поиск арбитражных ситуаций строится на основе сопоставления цен торговых пар на разных биржах, поиске лучших и предварительном отсечении малоприбыльных. Результатом этапа являются группы, внутри которых суще-

ствуют спреды, и роль каждой биржи в группе: на какой нужно купить, на какой продать, а на какой допустимо и то, и другое.

3.2.4 Построение цепочек

Построение цепочек состоит из двух этапов:

- формирование ядра цепочки;
- расширение ядра до полноценной цепочки.

В данном разделе приводится описание обоих этапов, а также достижение минимального количества межбиржевых переходов в цепочках, что позволяет сократить время их прогона.

3.2.5 Проверка совпадения сетей валют

На данном этапе сначала запускается скрапинг необходимых данных, после чего для всех цепочек проводится сама проверка, что спредообразующий межбиржевой переход возможен — возможные сети валюты на разных биржах должны пересекаться. Некорректные цепочки удаляются. Процесс обособлен от первого этапа скрапинга ввиду необходимости отправки отдельного запроса на каждую валюту, число которых значительно сокращается после предыдущих этапов. Данное утверждение будет справедливо и для следующего этапа.

3.2.6 Пересчёт прибыльности цепочки с учётом ликвидности

Сначала для всех цепочек проводится декомпозиция — разбиение на переходы, из числа которых в последствии выбираются те (внутрибиржевые), для которых необходимо уточнить ликвидность. После чего запускаются очередной этап скрапинга и пересчёт прибыльностей цепочек для начальных сумм 100\$, 1000\$ и 10000\$. Цепочки, для которых не удалось просчитать прибыльность даже для первой суммы, удаляются.

3.2.7 Сохранение результатов в БД

Последним этапом является сохранение результатов в базу данных, а именно ликвидных курсов и цепочек. Здесь приводится описание таблиц в БД, предназначенных для хранения этой информации, и некоторых их специфических полей, важных для ведения статистики о цепочках.

По окончании всех этапов поиска цепочек, а также завершения вычислений в модуле корреляционного анализа, отправляется сигнал контейнеру back для обновления данных у пользователей по веб-сокету.

3.3 Оркестрация периодических процессов

Весь контейнер algo оркестрируется на основе асинхронности, в том числе, с использованием `asyncio.sleep()`. В случае необходимости, можно либо разделить контейнер на несколько, либо начать использовать библиотеку `multiprocessing` для реального распараллеливания процессов.

3.4 Бэкенд

В этом разделе будут рассмотрены такие ключевые моменты, как система аутентификации и авторизации, а также работа с WebSocket соединением на примере предоставления и обновления информации об арбитражных цепочках.

Структура backend'а разделена на несколько модулей с помощью компонента FastAPI `router`, позволяющего группировать связанные маршруты.

3.4.1 Аутентификация и авторизация

Система аутентификации стандартна, refresh token хранится в БД, генерация токенов производится на основе алгоритма шифрования RSA256 и генерирующихся при каждом запуске сервера публичного и приватного ключей.

Авторизация пользователей производится при каждом обращении к роутам и инициируется посредством такого инструмента, предоставляемого FastAPI, как `Depends`, позволяющего создавать зависимости между компонентами приложения.

3.4.2 Предоставление и обновление данных по веб-сокету на примере арбитражных цепочек

В программе поддерживается список всех открытых веб-сокет соединений, а также переменная `loops`, содержащая в себе всю информацию о цепочках, предоставляемую по запросу пользователей, что помогает уменьшить нагрузку на базу данных и время отклика сервера. По окончании обновления информации в контейнере algo в back поступает уведомление о том, что данные обновлены, после чего происходит обновление информации в переменной `loops`, установка события `update_event` и последующая рассылка данных по всем открытым соединениям.

Фильтрация и сортировка цепочек происходят в мобильном приложении для снижения нагрузки на сервер и увеличения быстродействия.

ЗАКЛЮЧЕНИЕ

Автоматизация различных процессов имеет множество преимуществ, например, увеличение производительности и эффективности работы, перекладывание рутинной работы с человека на вычислительные устройства, уменьшение количества ошибок и сокращение влияния человеческого фактора, возможность обрабатывать большие объёмы данных. Однако, универсального подхода к автоматизации не существует, по крайней мере на данный момент, поэтому для выполнения таких задач зачастую требуется программист.

Подводя итоги, можно сказать, что при выполнении данной дипломной работы были выполнены все поставленные задачи, при этом были освоены такие навыки, как изучение и анализ предметной области, последующее проектирование приложения и выбор подходящих для решения задачи подходов и инструментов. Во время работы были изучены такие фреймворки и библиотеки, как `FastAPI` и `asyncpg`, `pyjwt` и `websockets`, позволяющие разрабатывать производительные веб-приложения, эффективно взаимодействовать с системами управления базами данных и многое другое, а также опробованы на практике и закреплены знания о построении крупных асинхронных систем в целом.

Дальнейшая разработка проекта будет продолжаться по завершении данной дипломной работы. Например, планируется такое расширение функционала системы, как реализация модуля автоматической торговли, а также расширение списка используемых бирж, в том числе посредством добавления децентрализованных.