МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра дискретной математики и информационных технологий

РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА ДЛЯ ОБУЧЕНИЯ КРИПТОТРЕЙДИНГУ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Ефанкина Даниила Андреевича

Научный руководитель
к.ф.-м.н., д.экон.н., профессор _____ Л. В. Кальянов

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2024

ВВЕДЕНИЕ

В современном мире финансовые рынки постоянно развиваются и усложняются, что требует от участников постоянной адаптации и обучения новым методам и инструментам. Одним из наиболее динамично развивающихся сегментов финансовых рынков является криптовалютный трейдинг. Криптовалюты, такие как Bitcoin, Ethereum и другие, стали объектом активного инвестирования и торговли, привлекая внимание как частных инвесторов, так и крупных финансовых институтов.

Однако криптотрейдинг отличается высокой степенью риска и сложностью, что требует от трейдеров глубоких знаний в области криптографии, финансовых рынков и аналитических навыков. Поэтому возникает потребность в специализированных инструментах и методах обучения, которые могли бы помочь новичкам в криптотрейдинге быстро и эффективно освоить необходимые навыки и стратегии.

На данный момент существует очень мало программных решений, которые могли бы в режиме реального времени и на основе актуальных данных помочь начинающему трейдеру освоить навыки торговли криптовалютой и приобрести полезный опыт без потери денежных средств.

Актуальность проблемы обучения криптотрейдингу обусловлена тем, что криптовалютный рынок чрезвычайно волатилен и подвержен влиянию различных факторов, таких как регуляторная политика, технологические инновации и глобальные экономические события. Это требует от трейдеров не только глубоких теоретических знаний, но и способности быстро реагировать на изменения рыночной конъюнктуры, что возможно только при наличии практического опыта.

Таким образом, разработка эффективного программного продукта для обучения криптотрейдингу является крайне важной задачей, который позволит не только снизить риски для начинающих трейдеров, но и повысить общий уровень квалификации участников криптовалютного рынка, что в свою очередь может способствовать его стабильности и устойчивому развитию. Кроме того, создание такого продукта может стать ключевым фактором в привлечении новых инвесторов и расширении рынка криптовалют, обеспечивая его рост и интеграцию в глобальную финансовую систему.

Таким образом, целью данной выпускной работы является разработ-

ка программного продукта для обучения криптотрейдингу, которое поможет приобрести практический опыт торговли без потери денежных средств.

Для достижения поставленной цели необходимо решить следующие задачи:

— Изучить особенности криптовалютного рынка;
— Изучить основы разработки клиентских и серверных приложений;
— Сформировать требования к будущему продукту;
— Смоделировать архитектуру разрабатываемого продукта;
— Реализовать серверные и клиентские приложения;
— Провести тестирование разработанного продукта.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первой главе рассматриваются все особенности криптовалютного рынка и описываются основные виды операция на финансовых рынках.

Криптовалютный рынок – это весьма специфический и динамичный сектор экономики, который привлекает внимание миллионов инвесторов и трейдеров по всему миру. Он отличается высокой волатильностью, низкой прозрачностью и отсутствием централизованной поддержки со стороны государств и финансовых учреждений.

Трейдинг – это процесс покупки и продажи активов с целью получения прибыли на основе изменений их цены. В контексте криптовалютного рынка, трейдинг подразумевает торговлю различными криптовалютами, такими как Bitcoin, Ethereum и другими, с использованием различных стратегий и инструментов для анализа рынка. Основная цель трейдера – определить оптимальные моменты для входа и выхода из сделок, чтобы максимизировать прибыль и минимизировать риски.

Рыночные и лимитные сделки – это два основных типа операций, используемых на финансовых рынках, включая криптовалютный рынок.

Рыночная сделка - это сделка с активом на рынке, в которой инвестор размещает запрос на покупку или продажу за наилучшую текущую цену. Иными словами, при рыночной сделке цена не фиксирована, а зависит от текущих условий рынка.

Лимитная сделка - это сделка с активом на рынке, в которой трейдер устанавливает цену исполнения сделки заранее. Цена актива должна достичь или превысить установленный лимит, чтобы сделка была выполнена.

Во второй главе рассматриваются основы разработки клиент-серверных приложений, описываются основные архитектуры программного обеспечения, рассматриваются этапы разработки программных продуктов, а также проводится сравнение и описание популярных языков программирования и различных инструментов для разработки клиент-серверных приложений.

Клиент-серверная архитектура является одной из наиболее распространенных архитектурных моделей для построения современных информационных систем. Она основана на разделении функций между двумя типами узлов: клиентами и серверами. Клиенты представляют собой устройства или программы, которые запрашивают информацию или услуги у серверов, кото-

рые обрабатывают эти запросы и предоставляют соответствующие ресурсы. Сервер в свою очередь может быть подключен к базе данных для взаимодействия с ней.

При разработки ПО очень важно делить весь процесс на этапы, основными этапами при разработки клиент-серверных приложения являются:

1. Составление требований к разрабатываемому продукту: на этом этапе определяются все требования, которым должен удовлетворять будущий продукт, функциональные требования описывают то, какие функции и задачи должно выполнять приложение, а нефункциональные требования определяют такие характеристики приложения, как производительность, масштабируемость, безопасность;
2. Проектирование системы для продукта: на данном этапе происходит разработка архитектуры приложения, которая определяет структуру приложения, включая его компоненты, связи между ними, протоколы коммуникации;
3. Разработка: это важный этап в процессе создания программного продукта, на нем разработчики пишут код;
4. Тестирование: данный этап играет ключевую роль в разработке программного обеспечения, поскольку его целью является проверка системы на ошибки и дефекты, которые могут повлиять на его работу в реальной среде.

Разработка серверных приложений связана с использованием специальных инструментов, которые помогают разработчикам создавать высокопроизводительное, масштабируемое и безопасное программное обеспечение. Основными инструментами для разработки серверных приложений являются:

1. Базы данных: программное обеспечение, которое сохраняет данные в структурированной форме и обеспечивает доступ к ним через язык запросов. Для разработки серверных приложений используются различные системы управления базами данных (СУБД), такие как MySQL, PostgreSQL, Oracle и MS SQL Server;
2. Web-серверы: программное обеспечение, которое обрабатывает запросы от клиентов, полученные через протокол HTTP или HTTPS. Часто используемые web-серверы включают Apache HTTP Server, Nginx и Microsoft IIS;

3. Фреймворки: коллекции библиотек и инструментов, которые используются для упрощения разработки серверных приложений;
4. Средства логирования: инструменты, которые помогают разработчикам отслеживать и анализировать логи серверов;
5. Системы контроля версий: представляют собой инструменты для контроля изменений, что позволяет разработчикам организовать совместную работу и управление кодом.

JavaScript является одним из самых важных языков программирования в современной веб-разработке. Он широко применяется для создания интерактивных веб-страниц, одностраничных приложений, игр, мобильных приложений и многого другого. JavaScript является основой всего веб-программирования, поскольку он обеспечивает динамическое взаимодействие с пользователем, обработку событий, анимацию и многое другое.

В третьей главе рассматривается разработка серверных приложений на языке Java, а также дополнительные инструменты и технологии для упрощения и ускорения разработки.

Java является одним из наиболее популярных языков программирования для серверного развития. Это связано с тем, что Java обладает несколькими уникальными свойствами, которые делают его идеальным для создания серверных приложений с высокой производительностью и масштабируемостью. Spring Framework является одним из ключевых фреймворков для серверной разработки на языке Java. Этот фреймворк был создан для упрощения и ускорения процесса разработки, а также для создания высокоэффективных и масштабируемых приложений. Рассмотрим некоторые из важных особенностей разработки на Java Spring. Spring Framework состоит из множества модулей, каждый из которых решает определенные задачи и позволяет разработчикам создавать приложения быстро, эффективно и без лишнего кода. Основные модули Spring Framework:

1. Spring Data JPA: данный модуль упрощает процесс взаимодействия с базой данных, предоставляет разработчикам набор абстракций, которые позволяют работать с базами данных без необходимости писать крупный объем кода;
2. Spring Web: данный модуль предоставляет разработчикам гибкие и эффективные способы разделения компонентов приложения и обеспечи-

вает удобный способ управления HTTP-запросами и ответами.

3. Spring Security: этот модуль обладает множеством функциональных возможностей, которые позволяют установить необходимые правила безопасности для приложений.

Также в Java приложения зачастую есть потребность использования баз данных для хранения информации и систем управления базами данных для взаимодействия с данными. PostgreSQL — это объектно-реляционная система управления базами данных, наиболее развитая из открытых СУБД в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных. СУБД позволяет гибко управлять базами данных.

Во четвертой главе описывается процесс разработки клиентских приложений с помощью фреймворка React и его особенности. React - один из самых популярных JavaScript фреймворков, значительно ускоряет и упрощает процесс разработки.

React использует концепцию компонентов, что делает разработку веб-приложений более модульной, гибкой и масштабируемой. Одним из ключевых преимуществ React является использование виртуального DOM, что обеспечивает более эффективное обновление пользовательского интерфейса. React сравнивает виртуальное DOM с реальным DOM и обновляет только те части интерфейса, которые действительно изменились, что значительно увеличивает производительность приложения.

Во пятой главе рассматриваются суть контейнеризации системы с помощью Docker. Docker - это платформа для контейнеризации приложений и сервисов. Он позволяет запускать приложения в изолированном окружении, которое содержит все необходимые зависимости, библиотеки и настройки. Основная цель Docker - это облегчение процесса разработки, тестирования и развертывания приложений. С помощью Docker можно создавать и управлять контейнерами, которые могут быть запущены и перемещены между разными окружениями без необходимости изменения настроек или кода приложения.

Во шестой главе описывается процесс подготовки к разработке программного продукта. На начально этапе были сформированы функциональные и не функциональные требования к разрабатываемой системе.

К системе был выдвинут следующий ряд функциональных требований:

1. Возможность регистрации и авторизации пользователей;
2. Возможность совершения демонстрационных рыночных и лимитных сделок по выбранной валютной паре;
3. Возможность анализировать рынок на основе графика курса валют;
4. Просмотр информации о балансе пользователя по каждой доступной криптовалюте;
5. Просмотр истории сделок пользователя;
6. Возможность анализировать истории сделок с помощью графических инструментов;
7. Возможность сброса баланса на аккаунте к исходному состоянию;
8. Возможность получать уведомления по лимитным сделкам через бота;
9. Предоставление базовой теории крипторынка с помощью теоритических материалов.

Из нефункиональных требований к приложению был выдвинут следующий ряд:

1. Высокая производительность, которая обеспечит моментальное исполнение сделки по заданному курсу;
2. Безопасность, которая защитит от межсайтовых атак, чтобы сервер обрабатывал запросы, только с защищенного клиента;
3. Удобство использования - интуитивно понятный и приятный интерфейс для взаимодействия с приложением.

Для серверной части системы был выбран язык Java, так как имеется большой опыт работы с данным языком программирования. Для упрощения процесса разработки также был выбран Spring Framework и его дополнительные модули:

1. Spring Data JPA - для работы с базой данных;
2. Spring Web - для обработки HTTP запросов от клиентской части;
3. Spring Security - для обеспечения безопасности и реализации регистрации/авторизации внутри приложения.

В качестве архитектуры для системы были выбраны два подхода: клиент-серверный и микросервисный. Для реализации клиентской части системы было принято решение реализации клиентского приложения с помощью JavaScript фреймворка React, так как есть опыт работы с данной технологией, к тому же к этому фреймфорку достаточно просто подключить библиотеку AnyChart

для реализации графиков и диаграмм в приложении, которые обеспечат визуальные инструменты для анализа рынка и совершенных пользователем сделок. Приложение было названо Trade-Web.

Микросервисный подход подразумевает разбиение функционала приложения между микросервисами, поэтому были спроектированы два основных серверных приложения разрабатываемой системы:

1. Trade-App - приложение для обучения трейдингу;
2. Notification-Bot - бот для отправки уведомлений пользователям через Telegram.

Во седьмой главе был рассмотрен процесс разработки серверной части программного продукта на языке Java, были описаны поэтапно все тонкости использования Spring Framework модулей.

Для управления зависимостями внутри приложения Trade-App и Notification-Bot было принято решение использовать сборщик Maven, так как имеется опыт работы с ним. Для реализации взаимодействия Java приложения с базой данных был выбран модуль Spring Data JPA для упрощения отправки запросов, перед началом использования данный модуль фреймворка требует настройки. Для каждой таблицы из базы данных был создан Java класс, к каждому полю были добавлены специальные аннотации, которые отображают данные из таблицы на конкретное поле класса. Такие классы в контексте ORM системы принято называть моделями или моделями данных.

Поиск лимитных сделок для закрытия был реализован, но скорость выполнения данного запроса не может удовлетворять главному требованию к системе - производительности. Для ускорения поиска заявок в БД было принято решение использовать индексы.

Индекс в базе данных - это структура данных, создаваемая для ускорения доступа к данным в таблице. Индекс содержит отсортированный список значений столбца(ов) таблицы, позволяя базе данных быстрее находить и извлекать необходимую информацию. Индексы являются важным аспектом производительности баз данных, поскольку они позволяют ускорить поиск и сортировку данных.

Во восьмой главе описан процесс разработки клиентской части системы, рассмотрена реализация подлючения к Keycloak и получение токена авторизации для взаимодействия с серверной частью.

Проектирование интерфейса всегда начинается с создания дизайна. Для этих целей существует множество бесплатных и продвинутый программ, например, Figma. Перед началом разработки были созданы примитивные макеты основных страниц клиентского приложения Trade-Web в программе drawio, который отображал примерный дизайн будущего приложения.

На основе прототипов были созданы макеты всех страниц приложения Trade-App. Разработанные дизайны страниц интерфейса:

1. Страница входа: интерфейс для авторизации в системе;
2. Страница регистрации: интерфейс для создания нового аккаунта в системе;
3. Страница торговли: основная часть интерфейса, которая предоставляет функционал для совершения обучающих сделок;
4. Страница анализа: интерфейс с графическим функционалом по анализу уже совершенных сделок;
5. Страница истории сделок: интерфейс для отображения всей истории сделок пользователя;
6. Страница советов по торговле: интерфейс для отображения теоретических советов для новичков в криптотрейдинге;
7. Страница личного кабинета: интерфейс для отображения информации по кошельку пользователя и токена авторизации в боте.

Для разработки React приложения была использована базовая структура проекта. Добавлены необходимые зависимости в проект. Версия NodeJS для разработки React приложения была выбрана 16.0.1, так как это последняя совместимая версия фреймворка React с библиотекой AnyChart. В приложении Trade-Web были выделены слудующие компоненты: кошелек пользователя, график для торговли, график для анализа сделок, таблица для отображения полной истории совершенных сделок, панель совершения сделок, панель календаря на страницы анализа, карточка совета с обучающей теорией, навигационная панель приложения. Для перехода между страницами в веб-приложении на React, используется React Router, который предоставляет удобные и эффективные средства навигации.

Во девятой главе описано тестирование разработанной системы.

После реализации всей системы был пройден этап тестирования. Для начала было проведено ручное тестировани. Ручное тестирование - имита-

ция действий пользователя. На этапе ручного тестирования было замечено несколько проблем с удобством использования интерфейса, эти проблемы были решены.

Интеграционное тестирование очень похоже на модульное, также для его реализации использовалась библиотека JUnit5, но его отличие в том, что оно предпологает тестирование взаимодействия различных компоненентов приложения, поэтому тестирование не ограничивалось лишь в рамках конкретного класса.

Во десятой главе рассмотрены все функциональные возможности разработанной системы, подробно описаны все элементы пользовательского интерфейса.

После удачной регистрации и авторизации пользователь попадает на основную страницу приложения. На данной странице сразу представлен функционал для совершения обучающих сделок. В центральной части интерфейса представлен график курса криптовалютной пары. График представляет из себя график японских свечей, каждая свеча характиризует поведение рынка в определенный промежуток времени. При наведении курсора на свечу появиться окно с дополнительной информацией о выбранном временном интервале. Справа от графика располагается панель для совершения новых сделок и управления графиком, на данной панеле можно выбрать доступную валютную пару и временной промежуток, то есть за какой временной интервал будет отвечать одна свечка на графике.

Также на данной панеле доступны 2 режима торговли: лимитная и рыночная сделки, перед совершением сделки необходимо выбрать ее тип. Из полей для ввода доступны: поле ввода желаемой цены продажи или преобретения актива и количество. В режиме рыночных сделок поле для ввода желаемой цены заблокировано, так как сделка будет совершена по текущему курсу. Также для удобства пользователя есть 4 кнопки, которые позволяют выбрать долю от текущего баланса, которую трейдер готов потратить на сделку. Использование данных кнопок позволит быстро принимать решение в сложных ситуациях и не вводить количество актива вручную.

В самом низу панели для торговли отображается информация по балансу выбранной криптовалютной пары, а также актуальная комиссия. Комиссия нужна для того, чтобы трейдер заранее понимал, что совершить сделку

и отменить ее, при этом не потеряв нисколько денег, на реальном криптовалютном рынке невозможно.

Одна из важных страниц подукта - это страница с анализом сделок. На данной странице можно увидеть исторический курс валюты в выбранный день, а также отметки с совершенными сделками. Это позволит анализировать совершенные сделки визуально и делать выводы о правильности принятых решений. При наведении на отметку сделки на графике можно получить дополнительную информацию о ней. Справа находится панель для выбора криптовалютной пары, а также отображается выбранный день для анализа. День можно выбрать с помощью календаря внизу страницы.

ЗАКЛЮЧЕНИЕ

Разработка масштабных и нагруженных систем - довольно сложный процесс. Однако, правильно выбрав технологии и верно выстроив архитектуру, можно создать высокопроизводительный и надежный продукт, соответствующий всем требованиям современного бизнеса и адаптированный к различным платформам и средам. Были изучены все особенности и этапы разработки серверных и клиентских приложений, рассмотрены все основные фреймворки и библиотеки для данных задач. Также были изучены популярные и современные архитектуры и принципы разработки ПО. В итоге был реализован программный продукт для обучения торговле криптовалютой. В будущем планируется дальше поддерживать данное приложение и добавлять новый функционал.

Основные источники информации:

1. Роберт Хиггинз Финансовый менеджмент. Управление капиталом и инвестициями. К., 2013. 47 с.
2. Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M. John Wiley Sons Pattern-Oriented Software Architecture: A System of Patterns. K., 1996. 476 с.
3. Официальная документация по Spring Framework [Электронный ресурс] Spring Framework. - URL: https://spring.io/docs (дата обращения: 10.12.2023)
4. Официальная документация по Spring Reactive [Электронный ресурс] Spring Framework. - URL: https://spring.io/reactive/docs (дата обращения: 14.12.2023)
5. Официальная документация по Hibernate ORM [Электронный ресурс] Hibernate. - URL: https://hibernate.org/orm/documentation (дата обращения: 05.01.2024)
6. Официальная документация по Spring Data JDBC [Электронный ресурс] Spring Data JDBC. - URL: https://docs.spring.io/spring-data/jdbc/docs/current/reference/ (дата обращения: 05.01.2024)
7. Официальная документация по Spring Data JPA [Электронный ресурс] Spring Data JPA. - URL: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/ (дата обращения: 09.01.2024)
8. Официальная документация по PostgreSQL [Электронный ресурс] PostgreSQL.

- URL: https://www.postgresql.org/docs/ (дата обращения: 04.10.2023)
9. Официальная документация по Redis [Электронный ресурс] Redis. - URL: https://redis.io/documentation (дата обращения: 12.09.2023)
10. Официальная документация по RabbitMQ [Электронный ресурс] RabbitMQ. - URL: https://www.rabbitmq.com/documentation.html (дата обращения: 21.12.2023)