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 КЛИЕНТСКОЙ ЧАСТИ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЧЕСКОЙ ТОРГОВЛИ КРИПТОВАЛЮТОЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Пичугиной Дарьи Николаевны

Научный руководитель
доцент, к. ф.-м. н, доцент _____ Ю. Н. Кондратова

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

Саратов 2025

ВВЕДЕНИЕ

Развитие криптовалютного рынка сопровождается постоянным увеличением объёма данных и числа пользователей, вовлечённых в торговлю цифровыми активами. Для эффективного взаимодействия с этим рынком всё чаще используются автоматизированные системы, обеспечивающие анализ арбитражных возможностей и управление торговыми роботами.

Однако наличие качественной серверной части не гарантирует удобства работы конечного пользователя. Важной составляющей любого программного продукта является клиентская часть, которая обеспечивает визуальное представление данных, доступ к функциональности системы и взаимодействие с пользователем.

Актуальность данной работы обусловлена необходимостью создания современного пользовательского интерфейса для проекта «Arbitoring», реализуемого компанией ООО «Интеллектуальные решения» — аккредитованной IT-компанией, основанной в 2019 году. Интерфейс должен быть адаптивным, мультиязычным, обеспечивать надёжную визуализацию финансовых данных и стабильное взаимодействие с серверной частью.

Целью данной работы является реализация пользовательского интерфейса клиент-серверного приложения, предназначенного для анализа арбитражных ситуаций и управления роботами на криптовалютном рынке с использованием современных веб-технологий для компании ООО «Интеллектуальные решения».

Поставленная цель определила следующие **задачи**:

- рассмотреть стек технологий, предопределённый в рамках проекта;
- реализовать ключевые пользовательские интерфейсы;
- реализовать поддержку мультиязычного интерфейса (русский и английский);
- организовать клиент-серверное взаимодействие с использованием HTTP и WebSocket;
- визуализировать данные для наглядного представления;
- обеспечить отзывчивость и адаптивность интерфейса для разных устройств.

Теоретическая значимость заключается в освоении современных подходов к разработке веб-интерфейсов, взаимодействию с сервером и использованию библиотек для визуализации и управления состоянием. Реализация проекта

способствовала углублению понимания архитектуры клиент-серверных приложений.

Практическая значимость состоит в создании рабочего интерфейса, внедрённого в деятельность компании ООО «Интеллектуальные решения» и используемого в проекте «Arbitoring» для анализа арбитражных ситуаций и управления торговыми роботами на криптовалютном рынке. Разработанное решение способствует повышению удобства взаимодействия с системой, обеспечивает наглядное отображение аналитических данных и успешно применяется в текущей работе над проектом.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и трех приложений. Общий объем работы – 52 страницы, из них 42 страницы — основное содержание, включая 23 рисунка, цифровой носитель в качестве приложения, список использованных источников информации — 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Описание используемых технологий» посвящён инструментам и библиотекам, применённым при разработке клиентской части веб-приложения. В разделе рассмотрены ключевые технологии, обеспечивающие построение интерфейса, взаимодействие с сервером, визуализацию данных, работу с формами и поддержку мультиязычности.

Основой клиентской части выступает фреймворк Next.js, построенный на базе библиотеки React. Его преимущества включают автоматическую маршрутизацию на основе файловой структуры, поддержку серверного рендеринга (SSR) и статической генерации (SSG), типизацию с использованием TypeScript, а также возможность динамической загрузки компонентов. Благодаря этим особенностям Next.js позволяет создавать масштабируемые и производительные веб-приложения, соответствующие современным требованиям, включая SEO-оптимизацию.

Для построения интерфейса используется библиотека Material UI (MUI), основанная на концепции Material Design от Google. MUI предоставляет готовые компоненты (кнопки, формы, таблицы и др.), которые легко настраиваются и интегрируются. В проекте применён механизм ThemeProvider для централизованного управления темами оформления. Это обеспечило визуальную целостность и адаптивность интерфейса на различных устройствах.

Для отображения аналитических данных применена библиотека ECharts, позволяющая строить интерактивные графики — линейные, столбчатые, круговые и диаграммы рассеяния. Графики поддерживают масштабирование, подсказки, выделение областей и взаимодействие с пользователем. Гибкость настройки и высокая производительность делают ECharts подходящим решением для визуализации большого объёма торговой информации.

Работа с формами реализована с помощью библиотеки Formik, которая упрощает управление состоянием полей и обработку событий. Все формы (вход, регистрация, управление роботами и ключами) построены на базе Formik. Для валидации используется библиотека Yup, позволяющая описывать схемы проверки данных декларативно и повторно использовать их в разных частях приложения.

Важной частью интерфейса является поддержка мультиязычности, реализованная с помощью библиотеки next-intl. Локализация осуществляется ав-

томатически на основе URL, а переводы хранятся в отдельных JSON-файлах. Для получения переводов внутри компонентов применяется хук useTranslations. Такой подход обеспечивает гибкую и масштабируемую поддержку нескольких языков — в данном случае русского и английского.

Для работы с датами и форматирования временных значений используется библиотека date-fns, которая поддерживает локализацию и позволяет использовать стандартизированные форматы. В проекте реализованы обёртки для отображения дат в различных форматах и получения дней недели на русском языке.

Взаимодействие с сервером построено на базе библиотеки Axios, предоставляющей удобный API для отправки HTTP-запросов. Axios позволяет централизованно задавать базовый URL, обрабатывать ошибки и добавлять заголовки (в том числе токены авторизации) ко всем запросам. Библиотека используется во всех разделах интерфейса, за исключением случаев, когда необходимо постоянное соединение с сервером.

В таких случаях применяется WebSocket — протокол для двустороннего обмена данными в реальном времени. Он позволяет серверу отправлять данные по мере их готовности, что особенно важно при работе с графиками, где объём информации может быть значительным. В отличие от одностороннего протокола HTTP, WebSocket обеспечивает устойчивое и эффективное соединение (рис. 0.1). В проекте используется защищённый вариант WSS, что соответствует требованиям безопасности.

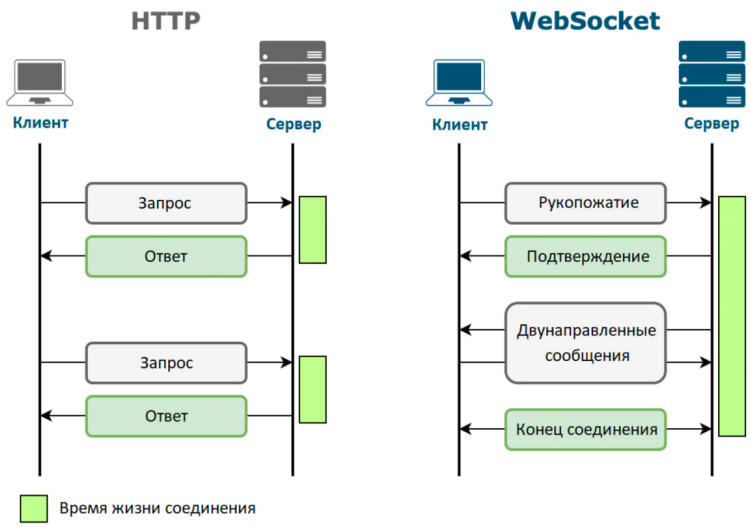

Рисунок 0.1 – Сравнительная схема работы HTTP и WebSocket

Таким образом, в первом разделе обоснован выбор технологического стека, который обеспечил реализацию функционального, гибкого и надёжного пользовательского интерфейса, способного работать с большими объёмами данных и обеспечивать удобное взаимодействие с системой.

Второй раздел «Разработка клиентской части» посвящён практической реализации пользовательского интерфейса веб-приложения для управления торговыми роботами и анализа арбитражных ситуаций на криптовалютном рынке.

В разделе последовательно рассматриваются реализованные экраны и ключевые элементы интерфейса. Работа над клиентской частью включала в себя не только верстку и стилизацию, но и полную интеграцию с серверной частью проекта, настройку визуализации данных, реализацию адаптивности, мультиязычности и взаимодействие с протоколами HTTP и WebSocket. Особое внимание было уделено пользовательскому опыту (UX): проектирование интерфейса велось с учётом потребностей конечных пользователей, их сценариев поведения и требований к удобству работы как на настольных устройствах, так и на мобильных платформах.

В первом подразделе описана реализация форм входа и регистрации. Реализована система авторизации с валидацией данных и обработкой токенов доступа, обеспечивающих безопасное хранение и использование учётных данных. После регистрации пользователь перенаправляется на экран настроек. Все формы созданы с использованием библиотеки Formik и библиотеки Yup для валидации, что позволило централизовать обработку состояний, ошибок и проверок ввода.

Во втором подразделе рассматривается главный экран приложения, отображающий список роботов пользователя в виде слайдера. Такой подход позволил компактно и наглядно представить информацию о текущей торговой активности. Реализована система отображения карточек с основными метриками, такими как прибыль, количество сделок, эффективность и статус роботов. Визуализация сделок и баланса осуществляется с помощью библиотеки ECharts, позволяющей динамически строить графики. Пользователю предоставлены фильтры и выбор временного диапазона для настройки отображаемой информации. Администраторам доступны расширенные функции поиска роботов и фильтрации сделок.

Третий подраздел посвящён экрану управления параметрами роботов. Была разработана форма настройки параметров с пояснениями к каждому полю. Также реализованы состояния загрузки, модальные уведомления и возможность переключения между торговым и аналитическим режимами (доступно админи-

страторам).

Четвёртый подраздел — экран управления API-ключами, в котором пользователь может добавлять, редактировать и удалять ключи от криптобирж. Было реализовано динамическое обновление формы в зависимости от выбранной биржи: некоторые поля скрываются или становятся обязательными, в зависимости от требований конкретной платформы. Асинхронная проверка ключа на валидность происходит через запрос к серверу, результат отображается в виде визуальной индикации статуса (например, зелёная галочка при успехе). Вся работа с формой и её состоянием построена с помощью Formik, что обеспечивает высокую степень повторного использования компонентов и модульность кода.

Пятый подраздел посвящён экрану подписок, который позволяет пользователю выбрать уровень подписки (Starter, Basic, Advanced), а также пополнить баланс через внешнюю платёжную систему. Реализован расчет итоговой стоимости с учётом выбранного тарифа и оставшегося времени действия текущей подписки.

В шестом подразделе настроек пользователь может изменить личные данные, язык интерфейса, способ получения уведомлений и формат отображения даты. Особое внимание уделено синхронизации локализации интерфейса с серверными и клиентскими данными через библиотеку next-intl. Также реализована проверка подключения Telegram и уведомления в случае его отсутствия.

Последние два подраздела посвящены аналитическим графикам. Была реализована система выбора и построения графика по конкретной сделке с возможностью задания временного диапазона. Построенные графики синхронизированы между собой и включают режим отображения дисперсий, область действия сделки и управление округлением значений.

График по монете предназначен для администраторов и отображает все сделки по выбранной валюте за определённый период. В связи с большим объёмом данных реализовано подключение через WebSocket, а не HTTP. Разработана система отображения прогресса загрузки, интерактивную фильтрацию и отображение точек сделок с деталями при наведении. Отдельно реализована система обработки чанков, агрегации данных и восстановления соединения при ошибках.

В результате работы была создана полноценная клиентская часть для реальной платформы. Интерфейс внедрён в рабочую систему компании, соответ-

ствует требованиям, имеет надёжное взаимодействие с сервером, визуализацию аналитики и адаптацию под различные устройства и языки.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была реализована frontend-часть клиент-серверного приложения для анализа арбитражных ситуаций и управления торговыми роботами на криптовалютном рынке.

Разработанный интерфейс отвечает современным требованиям: он адаптивен, мультиязычен, обеспечивает визуализацию сложных финансовых данных и поддерживает взаимодействие с серверной частью как по HTTP, так и по WebSocket-протоколу.

Решение поставленных задач позволило создать функциональный и удобный инструмент, который может быть использован как конечными пользователями, так и специалистами в области алгоритмической торговли.

В процессе работы были получены практические навыки проектирования интерфейсов, организации клиент-серверного взаимодействия, работы с WebSocket-протоколом и библиотеками для построения интерактивных пользовательских интерфейсов.

Дальнейшая разработка проекта будет продолжаться по завершении данной дипломной работы. Например, планируется такое расширение функционала системы, как ведение реферальной программы для клиентов, а также внедрение дополнительных элементов интерфейса с пояснениями и подсказками, которые помогут пользователям лучше ориентироваться в функционале.