

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

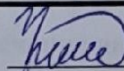
Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

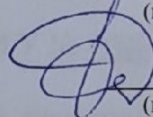
на тему

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ АНАЛИЗА И
ОБРАБОТКИ ДАННЫХ

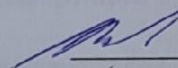
Студент

 03.06.2026 Ульченко Фёдор Сергеевич
(подпись, дата) (ФИО полностью)

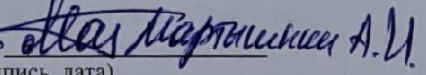
Руководитель

 03.06.2026 Федюнин Р.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

 03.06.2026 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент

зав. кафедрой Программирования ПЭВМ 05.06.2026  Мартынов А.И.
(должность, место работы) (подпись, дата)
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 № 14)

Заведующий кафедрой

(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой

отлично

(протокол заседания ГЭК от № 4)

Секретарь ГЭК

(подпись)

Е.И. Гурин
(фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

« 20 » 09 20 24 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Ульченко Фёдор Сергеевич гр.24ВВИм1 факультета ВТ
2. Тема работы «Программное обеспечение анализа и обработки данных»
Тема утверждена приказом ПГУ № 929/0 от "20" сентября 2024г.
3. Руководитель магистерской диссертации к.т.н., доцент Федюнин Р.Н.

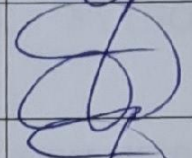
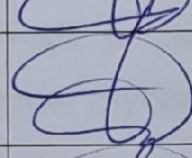
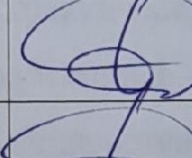
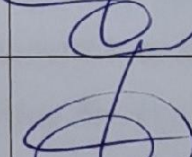
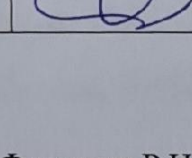
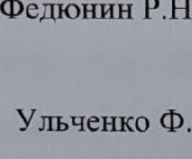
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки: разработка программного обеспечения анализа и обработки данных. Разрабатываемая система должна выполнять автоматическое прогнозирование цены биткоина на горизонт до 30 дней, с предоставлением прогноза пользователю через телеграм-бота.

5. Перечень подлежащих разработке вопросов

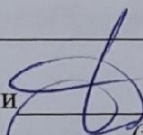
Требуется произвести анализ предметной области и существующих методов прогнозирования криптовалют. Провести сравнительный анализ нейросетевых архитектур (RNN, LSTM, Transformer, LLM/Agent) для прогнозирования временных рядов. Выполнить проектирование и обучение LSTM-модели на исторических данных цены биткоина. Разработать модуль торговой стратегии на основе прогнозов LSTM. Спроектировать и реализовать серверную часть системы (REST API, модули данных, прогнозирования и управления пользователями). Разработать клиентскую часть в виде Telegram-бота с реализацией инлайн-меню, обработки команд и интеграции с платёжной системой. Осуществить контейнеризацию системы с использованием Docker и развертывание на VPS. Провести тестирование разработанной системы, анализ точности прогнозов и эффективности торговой стратегии.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	11%	29.11.2024- 29.01.2025	
2.	Подбор и анализ продуктов аналогичного назначения	11%	29.01.2025- 30.03.2025	
3.	Разработка и анализ требований	22%	30.03.2025- 30.05.2025	
4.	Проектирование и реализация системы	27%	01.06.2025- 01.09.2025	
5.	Развёртывание системы	9%	02.09.2025- 02.10.2025	
6.	Тестирование	9%	02.10.2025- 02.12.2025	
7.	Анализ полученных результатов	5%	01.01.2026- 25.02.2026	
8.	Оформление пояснительной записки	6%	01.03.2026- 20.05.2026	

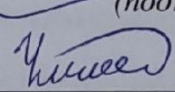
Дата выдачи задания " 20 " 09 2024 г.

Руководитель магистерской диссертации


(подпись, дата)

Федюнин Р.Н.

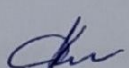
Задание к исполнению принял студент


(подпись, дата)

Ульченко Ф.С.

Работу к защите допустить

Декан факультета

 08.06.2020
(подпись, дата)

Фионова Л.Р.

РЕФЕРАТ

Пояснительная записка содержит 98 листов, 17 рисунков, 7 таблиц, 25 источников, 4 приложения.

АНАЛИЗ ДАННЫХ, ПРОГНОЗИРОВАНИЕ ВРЕМЕННЫХ РЯДОВ, КРИПТОВАЛЮТА, BITCOIN, LSTM, РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ, TELEGRAM-BOT, RUP, UML, VPS, DOCKER, PYTHON, TENSORFLOW, OPTUNA

Целью работы является разработка программного обеспечения анализа и обработки данных, которое представляет из себя интеллектуальную систему прогнозирования цены биткоина на основе разработанной и обученной LSTM модели. Предоставление прогнозов через Telegram-бота. Научная новизна исследования заключается в комплексном подходе, сочетающем глубокое обучение для многошагового прогнозирования криптовалюты и практическую реализацию в виде легковесного серверного решения с контейнеризацией.

В ходе исследования проведён сравнительный анализ нейросетевых архитектур, обоснован выбор многослойной LSTM, спроектирована, разработана и обучена LSTM модель для прогнозирования цены биткоина. Разработаны REST API и Telegram-бот. Проект реализован на Python с применением библиотек TensorFlow/Keras, Pandas, scikit-learn, Optuna, python-telegram-bot.

Полученные результаты показывают, что LSTM-модель достигает RMSE 1 177 USD и MAPE 3% на тестовой выборке. Предложенный подход может служить основой для создания коммерческих систем автоматизированной торговли криптовалютами и раннего оповещения трейдеров.

					ПГУ 1.090401.26.001 ПЗ					
Изм.	Лист	№ докум	Подпись	Дата	Программное обеспечение анализа и обработки данных] Пояснительная записка			Лит.	Лист	Листов
Разраб.		Ульченко Ф.С.		03.06.20						
Провер.		Федюнин Р.Н.		03.06.20					4	69
Н. Контр.		Бычков А.С.		03.06.20				ФВТ, каф. ВТ гр. 24ВВИм1		
Утв.		Митрохин М.А.		03.06.20						

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	9
1. Анализ предметной области и требований	11
1.1. Анализ предметной области и постановка задачи	11
1.2. Анализ продуктов аналогичного назначения	11
1.2.1. Bybit Aurora AI	11
1.2.2. 3Commas	12
1.2.3. GNYaiBOT	13
1.2.4. Сравнение аналогов	14
1.3. Анализ целевой аудитории	15
1.4. Анализ функциональных требований	16
1.5. Анализ нефункциональных требований	21
1.6. Сравнительный анализ нейросетевых архитектур	22
1.7. Подробный анализ архитектуры LSTM	25
2. Проектирование системы	29
2.1. Выбор технологий и инструментальных средств разработки	29
2.2. Проектирование архитектуры программного обеспечения	32
2.3. Проектирование серверной части системы	33
2.4. Проектирование LSTM-модели	33
2.5. Проектирование клиентской части системы	37
3. Реализация системы	40
3.1. Реализация получения и предобработки данных	40
3.2. Разработка и обучение LSTM-модели	40
3.3. Разработка модуля торговой стратегии	42
3.4. Разработка Telegram-бота	43
3.5. Развёртывание разрабатываемой системы	45
3.5.1. Выбор подходящего VPS	47
3.5.2. Контейнеризация системы с использованием Docker	48
4. Тестирование программного продукта	52
4.1. Тестирование серверной части приложения	52

4.1.1. Функциональное тестирование серверной части	52
4.1.2. Модульное тестирование серверной части	52
4.2. Тестирование клиентской части приложения	55
4.2.1. Функциональное тестирование клиентской части	55
4.2.2. Модульное тестирование клиентской части	56
4.3. Метрики кода	57
4.3.1. Метрики размера программы	59
4.3.2. Метрики Холстеда	60
ЗАКЛЮЧЕНИЕ	65
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	68
ПРИЛОЖЕНИЕ А – Листинг программы	71
ПРИЛОЖЕНИЕ Б – Результаты тестирования серверной части	86
ПРИЛОЖЕНИЕ В – Результаты тестирования клиентской части	89
ПРИЛОЖЕНИЕ Г – Презентация	93

ВВЕДЕНИЕ

В современном мире стремительного развития цифровых технологий и финансовых рынков системы анализа данных и генерации индикаторов для торговли приобретают всё большую популярность. Эти системы, основанные на сложных алгоритмах машинного обучения, призваны предоставлять трейдерам надежные прогнозы и сигналы, минимизируя влияние человеческого фактора и способствуя принятию обоснованных решений.

Согласно Указу Президента Российской Федерации от 9 мая 2017 г. № 203 «О Стратегии развития информационного общества в Российской Федерации на 2017–2030 годы», одной из ключевых национальных целей является формирование цифровой экономики, в которой данные в цифровой форме выступают важнейшим фактором производства, обеспечивая повышение конкурентоспособности страны и экономический рост. Данная стратегия нашла продолжение в национальном проекте «Экономика данных и цифровая трансформация государства», направленном на ускорение цифровой трансформации всех сфер экономики, включая в том числе финансовый сектор [1].

Также одной из приоритетных задач является ускоренное внедрение цифровых технологий в экономике и социальной сфере. Это направление получило дальнейшее развитие в национальном проекте «Экономика данных и цифровая трансформация государства» на 2025–2030 годы, который предусматривает комплексную модернизацию экономических процессов, включая финансовый сектор, через внедрение искусственного интеллекта, анализ больших данных и обеспечение технологического суверенитета. Общий объем финансирования проекта превышает 1 трлн рублей, что подчеркивает стратегический приоритет данных направлений для повышения конкурентоспособности страны.

Криптовалютный рынок, характеризующийся высокой волатильностью и круглосуточной торговлей, предоставляет уникальные возможности для применения современных методов анализа данных. По состоянию на 2026

год глобальная капитализация рынка криптовалют превышает 2,6 трлн долларов США, при этом капитализация криптовалюта составляет около 1,5 трлн долларов, что подтверждает его ведущую роль на рынке [2]. Высокая волатильность делает традиционные методы прогнозирования недостаточно эффективными и актуализирует применение рекуррентных нейронных сетей типа LSTM.

Согласно аналитическим материалам Банка России за 2024 год, доля алгоритмической торговли на организованных площадках РФ превысила 45%, что подтверждает объективный структурный сдвиг в сторону автоматизированных решений [3].

Параллельно международные исследования, включая систематический мета-анализ в авторитетном рецензируемом журнале Expert Systems with Applications за 2023 год, демонстрируют, что архитектуры глубокого обучения, в частности рекуррентные нейронные сети, обеспечивают статистически значимое улучшение точности краткосрочных прогнозов по сравнению с традиционными эконометрическими моделями [4].

При этом отечественный стандарт ГОСТ Р 71476-2024 прямо указывает на необходимость обеспечения воспроизводимости и интерпретируемости алгоритмических решений, что определяет методологический вектор настоящего исследования [5].

Опираясь на приведённые выше факторы можно сделать вывод о том, что разработка данного программного обеспечения весьма актуальна и перспективна в условиях современного общества

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Целью настоящей работы является разработка модульного программного обеспечения анализа и обработки данных, реализующего прогнозирование цены криптовалюты с помощью LSTM-модели и доставку результатов пользователям через Telegram-бот. Обычным пользователям предоставляется ежедневный сигнал (buy/sell/hold), а премиум-пользователям расширенный прогноз на горизонте до трёх месяцев с указанием потенциальных точек входа и выхода из позиций.

Для достижения цели необходимо решить следующие основные задачи:

- исследовать особенности временных рядов цен криптовалюты и обосновать выбор архитектуры LSTM для многошагового прогнозирования,
- разработать требования к функциональным и нефункциональным характеристикам системы,
- спроектировать модульную архитектуру программного обеспечения,
- реализовать компоненты системы, включая предобработку данных, обучение и применение LSTM-модели, формирование торговой стратегии и интеграцию с Telegram-ботом,
- провести комплексное тестирование разработанного ПО и оценить качество получаемых прогнозов.

1.2 Анализ продуктов аналогичного назначения

В настоящее время существует целый ряд сервисов, использующихся трейдерами в торговле. Рассмотрим наиболее известные и произведём для них сравнительный анализ.

1.2.1 Bybit Aurora AI

В основе ИИ Аврора лежит анализ исторических рыночных данных за четыре дня, который позволяет определить оптимальные параметры стратегии, которые используются для создания бот-стратегий с интеллектуальными параметрами, включая фьючерсного grid-бота, фьючерсного Мартингейла и спотового grid-бота [6].

Перейдя в раздел инструменты и выбрав категорию торговых ботов пользователь попадает на страницу, где может воспользоваться помощью ИИ для подбора оптимальной торговой стратегии, чем функционал и ограничивается.

Страница подбора стратегии при помощи «Aurora AI» на криптобирже «Bybit» представлена на рисунке 1.

Подберите идеальный торговый бот

1. Выберите актив

2. Выберите стиль торговли

3. Выберите направление рынка

Подобрать бот

Лучшее совпадение

Спотовый grid-бот
BTC/USDT Grid-бот «Купить дешево»

Почему именно этот бот?

Текущая цена
76,826.6 -2.03%

Ценовой диапазон
57,600.5 - 69,120.6 USDT

Начать сейчас

Сравнить все боты →

Рисунок 1 – Страница подбора стратегии при помощи «Aurora AI» на криптобирже «Bybit»

В качестве достоинства можно выделить бесплатность.

К недостаткам сервиса относятся: маленькая выборка данных для анализа, и отсутствие четкого прогноза.

1.2.2 3Commas

3Commas – ведущий разработчик программного обеспечения для криптотрейдинга, предлагающий ИИ-боты для торговли криптовалютам [7].

Пройдя авторизацию пользователь попадает на главную страницу приложения, с которой есть доступ ко всему функционалу: создание разных ботов, просмотр своих торговых позиций на графике. Навигация на сайте устроена с помощью боковой панели, в которой размещены вкладки приложения.

Страница выбора и создания бота при помощи сервиса «3Commas» представлена на рисунке 2.

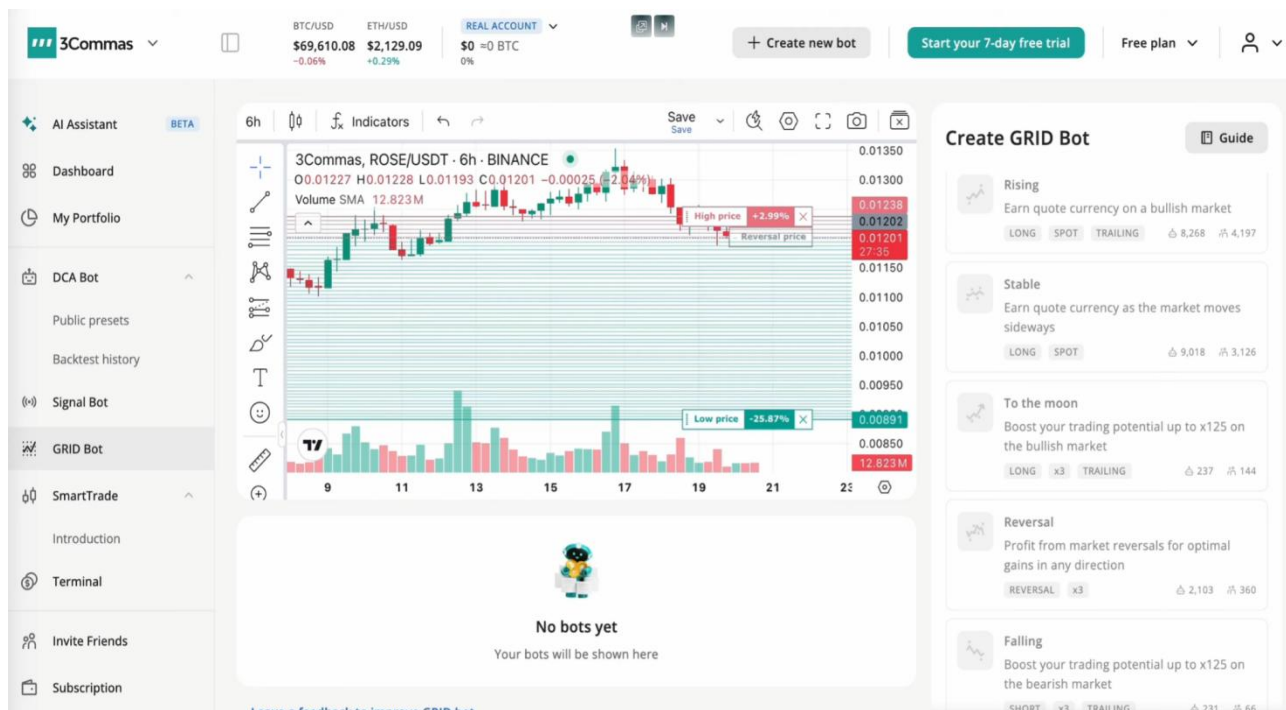


Рисунок 2 – Страница создания бота на сайте «3Commas»

В качестве достоинства можно выделить большой выбор инструментов.

К недостаткам мобильного приложения относится отсутствие бесплатной версии, урезанный функционал бюджетных и дороговизна продвинутых подписок.

1.2.3 GNYaiBOT

На официальном сайте данный проект представлен следующим образом: GNYaiBOT сделает ведущие ИИ-инсайты в криптоиндустрии наиболее доступными, предоставляя пользователям возможность запрашивать нашу ИИ через личные сообщения в Telegram [8].

Начав диалог с ботом пользователь попадает в главное меню, из которого есть доступ ко всему функционалу: Навигация в приложении устроена с помощью меню, выводящегося после каждого ответа бота.

Главное меню и весь основной функционал «GNYaiBOT» представлены на рисунке 3.

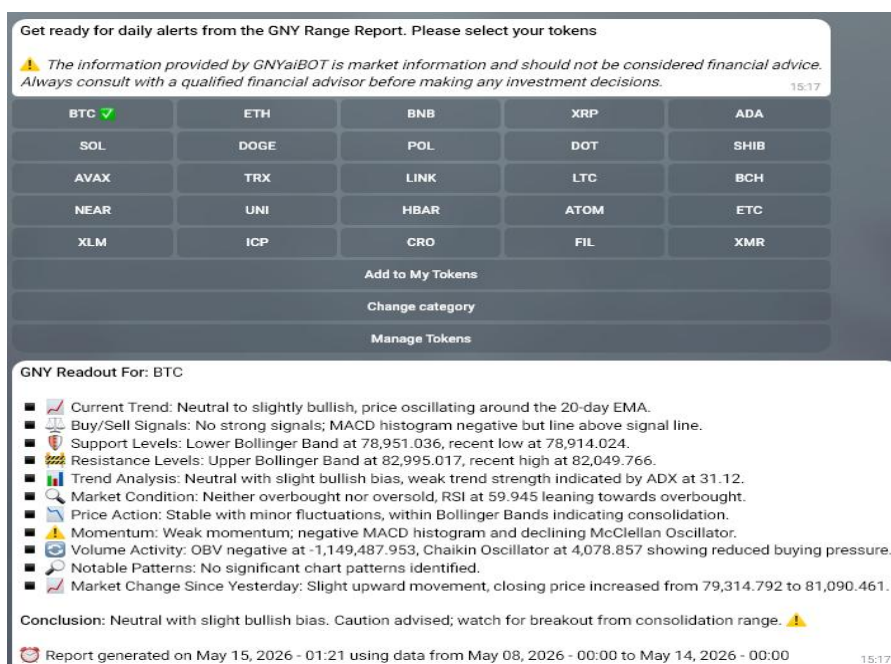


Рисунок 3 – Главное меню и функционал «GNYaiBOT»

К достоинствам бота относятся бесплатность и наличие прогноза.

К недостаткам бота относятся: прогноз формируемый только на текущий момент времени, малый выборка анализируемых данных, скудный функционал.

1.2.4 Сравнение аналогов

Сравнительный анализ рассмотренных выше систем схожего назначения представлен в таблице 1.

Таблица 1 – Сравнение аналогов

Рассматриваемая система	Прогноз	Горизонт прогноза	Горизонт анализа	Бесплатная версия	Стоимость премиум версии
Bybit Aurora AI	Нет	Нет	4 дня	Да	Нет
3Commas	Нет	Нет	30 дней	Нет	20-200\$/30дней
GNYaiBOT	Простой ТА	1 день	7 дней	Да	Нет
Разрабатываемое приложение	Полный прогноз	До 30 дней	13 лет	Да	20\$/30дней

Таким образом, в результате анализа систем аналогичного назначения было принято решение о разработке собственного приложения, которое будет лишено недостатков, выявленных у рассмотренных выше систем.

1.3 Анализ целевой аудитории

Целевой аудиторией разрабатываемого программного обеспечения являются участники криптовалютного рынка, заинтересованные в получении автоматизированных аналитических сигналов и прогнозов для поддержки принятия торговых решений.

Согласно данным на 2025–2026 годы, общее количество владельцев криптовалют в мире превысило 700 миллионов человек и продолжает активно расти. Наиболее активной возрастной группой являются пользователи от 25 до 34 лет, составляющие около 31 % от общего числа криптоинвесторов. Следующей значимой группой является категория 35–44 лет (около 23 %), часто обладающая более крупными портфелями [9].

Основными представителями целевой аудитории выступают:

- розничные трейдеры и индивидуальные инвесторы, регулярно совершающие операции с криптовалюта и другими криптовалютами,
- среднесрочные инвесторы, заинтересованные в получении не только ежедневных сигналов, но и расширенных прогнозов на горизонте до трёх месяцев.

Значительная часть целевой аудитории обладает базовыми или средними знаниями в области технического анализа и машинного обучения, однако испытывает потребность в надежном автоматизированном инструменте, снижающем влияние эмоциональных факторов при торговле.

Важной особенностью аудитории является разделение на обычных и премиум-пользователей. Обычные пользователи нуждаются в простых и понятных ежедневных сигналах (buy/sell/hold). Премиум-пользователи, готовые оплачивать подписку, ожидают более глубокий анализ, включая многошаговые прогнозы и выявление потенциальных точек входа и выхода из позиций.

Таким образом, разрабатываемое программное обеспечение ориентировано на удобство использования, оперативность предоставления

информации и дифференциацию функционала в зависимости от уровня доступа пользователя.

1.4 Анализ функциональных требований

Система может применяться частными трейдерами для улучшения своих торговых результатов. В базовом режиме программное обеспечение позволяет получить прогноз цены криптовалюты на следующий день. Так же, в рамках ежемесячной платной подписки, предоставляется прогноз на больший промежуток времени, вплоть до 30 дней.

Обычный прогноз выдаётся мгновенно и не требует каких-либо дополнительных условий – он доступен любому пользователю, начавшему диалог. Чтобы получить расширенный прогноз на несколько дней вперёд, трейдер должен оформить платную подписку. Оплата производится через платёжный сервис ЮKassa; после успешного платежа статус пользователя в системе меняется на «премиум».

Премиум-пользователь может не только запрашивать прогноз на период до 30 дней, но и самостоятельно выбирать нужный горизонт прогнозирования (например, 7, 14 или 30 дней).

Функционально подписка реализована как временное право: по истечении оплаченного периода система автоматически переводит пользователя обратно в статус «обычный», блокируя доступ к премиум-прогнозу и возможности изменения горизонта. При этом все ранее сделанные настройки (например, выбранный горизонт) сохраняются, но становятся недоступными до следующего продления подписки. Пользователь в любой момент может повторно оплатить подписку, после чего срок продлевается на следующие 30 дней от текущей даты.

Таким образом, разработанное ПО предоставляет трейдеру гибкий инструмент для принятия решений – от краткосрочного сигнала на следующий день до среднесрочного прогноза на месяц вперёд.

В результате анализа требований задания была создана диаграмма вариантов использования (рисунок 4).



Рисунок 4 – Диаграмма вариантов использования

Для каждого прецедента диаграммы, представленной выше, была составленная подробная спецификация.

Составленная спецификация для прецедента «Получить обычный прогноз» приведена ниже на рисунке 5.

Наименование: Получить обычный прогноз
Назначение: предоставление обычного прогноза пользователю
Актёры: обычный пользователь/премиум пользователь
Предусловие: открыт диалог с ботом
Основной сценарий: <ol style="list-style-type: none"> 1. Пользователь выбирает в меню вариант с обычным прогнозом. 2. Телеграм-бот отсылает запрос на сервер. 3. Сервер возвращает прогноз. 4. Телеграм-бот выводит обычный прогноз.
Постусловие: прогноз предоставлен

Рисунок 5 – Спецификация варианта использования «Получить обычный прогноз»

Далее приведена спецификация для прецедента «Оплатить подписку» (рисунок 6).

Наименование: Оплатить подписку
Назначение: предоставление премиум прогноза пользователю
Актёры: обычный пользователь/премиум пользователь
Предусловие: открыт диалог с ботом, подписка оплачена
Основной сценарий: 1. Пользователь выбирает в меню вариант с оплатой подписки. 2. Открывается окно платежной системы. 3. Пользователь проводит оплату. 4. Система меняет статус пользователя.
Постусловие: подписка оплачена, пользователю доступен премиум функционал

Рисунок 6 – Спецификация варианта использования «Оплатить подписку»

Также составим спецификацию прецедента «Получить премиум прогноз» (рисунок 7).

Наименование: Получить премиум прогноз
Назначение: предоставление премиум прогноза пользователю
Актёры: премиум пользователь
Предусловие: открыт диалог с ботом, подписка оплачена
Основной сценарий: 1. Пользователь выбирает в меню вариант с премиум прогнозом. 2. Система проверяет статус пользователя. 3. Телеграм-бот отсылает запрос на сервер. 4. Сервер возвращает прогноз. 5. Телеграм-бот выводит премиум прогноз.
Постусловие: прогноз предоставлен

Рисунок 7 – Спецификация варианта использования «Получить премиум прогноз»

Составим спецификацию прецедента «Выбрать горизонт прогнозирования» (рисунок 8).

Наименование: Выбрать горизонт прогнозирования
Назначение: предоставление премиум прогноза пользователю
Актёры: премиум пользователь
Предусловие: открыт диалог с ботом, подписка оплачена
Основной сценарий: <ol style="list-style-type: none"> 1. Пользователь выбирает в меню вариант с изменением горизонта прогнозирования. 2. Система проверяет статус пользователя. 3. Телеграм-бот отправляет запрос на сервер. 4. Сервер сохраняет изменения горизонта прогнозирования.
Постусловие: горизонт прогнозирования изменен

Рисунок 8 – Спецификация варианта использования «Выбрать горизонт прогнозирования»

Для некоторых из рассмотренных выше вариантов использования было принято решение разработать диаграммы последовательности и деятельности, с целью рассмотреть данные варианты использования подробнее.

Для варианта использования «получить обычный прогноз» была составлена диаграмма последовательности, представленная на рисунке 9.

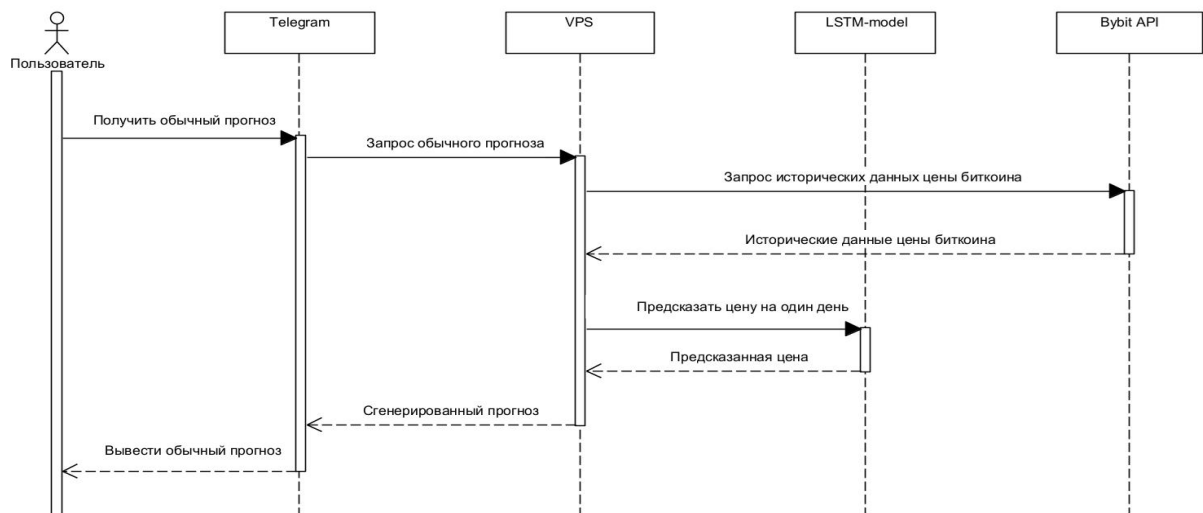


Рисунок 9 – Диаграмма последовательности варианта использования «получить обычный прогноз»

Данная диаграмма последовательности демонстрирует взаимодействие компонентов системы. Пользователь отправляет запрос через Telegram-бота на получение прогноза. VPS-сервер сначала запрашивает исторические данные о цене криптовалюты через Bynbit API, затем обращается к LSTM-

модели с запросом на генерацию прогноза. После получения данных нейронная сеть генерирует прогноз цены на один день. Результат передается обратно через VPS в Telegram и выводится пользователю в виде готового прогноза.

Для варианта использования «получить премиум прогноз» составлена диаграмма деятельности, представленная на рисунке 10.

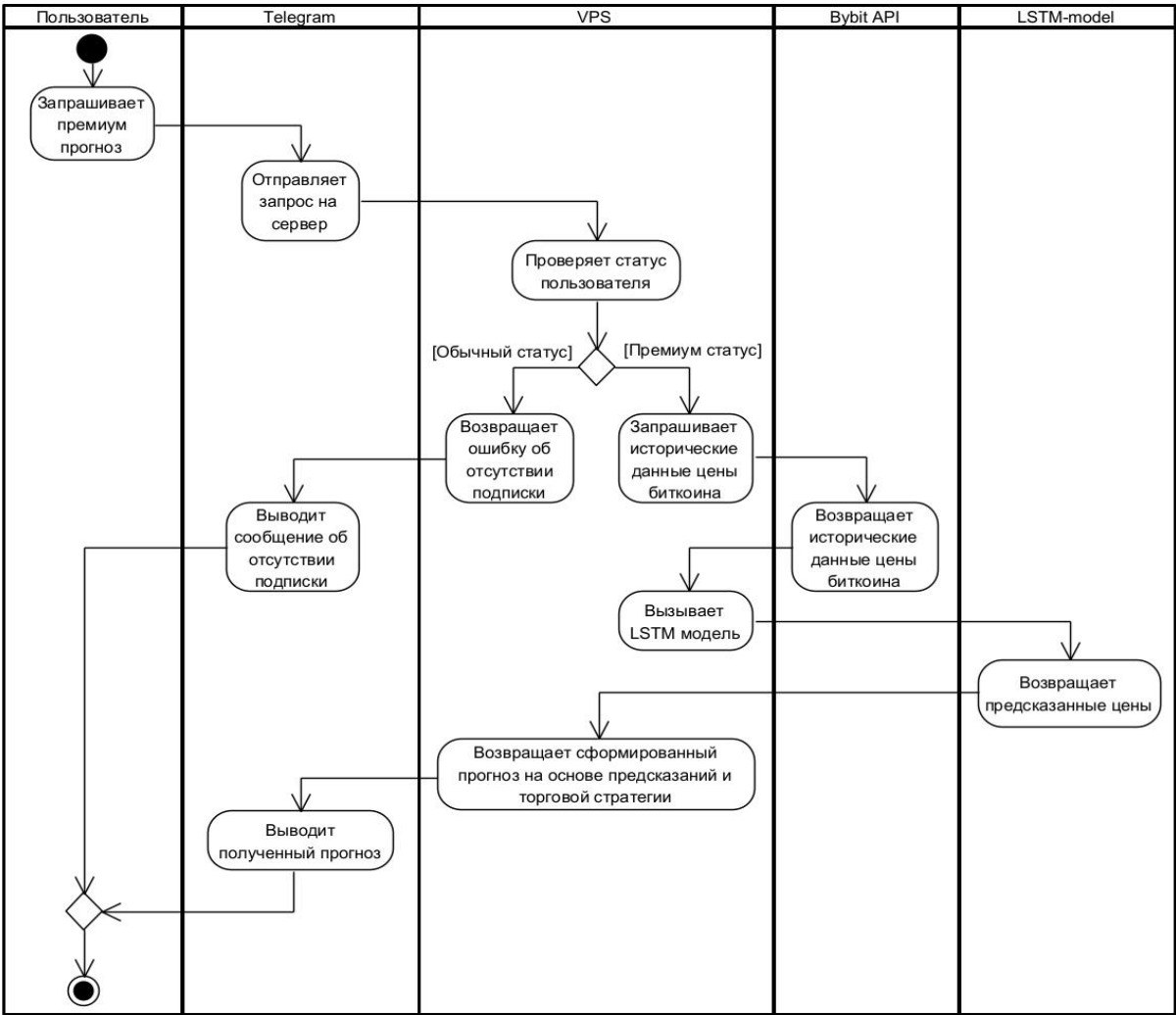


Рисунок 10 – Диаграмма деятельности варианта использования «получить премиум прогноз»

В результате анализа функциональных требований были успешно выявлены основные функции разрабатываемого приложения. При этом для разрабатываемого приложения были построены диаграммы вариантов использования, деятельности и последовательности, а также для некоторых выявленных вариантов использования были составлены спецификации.

1.5 Анализ нефункциональных требований

Помимо функциональных возможностей, к разрабатываемому программному обеспечению предъявляется ряд нефункциональных требований, определяющих качество, надежность и удобство эксплуатации системы.

К основным нефункциональным требованиям относятся:

- надежность и устойчивость работы,
- производительность,
- масштабируемость,
- удобство использования,
- поддерживаемость,
- точность прогнозирования
- кроссплатформенность и доступность.

Ниже приведено подробное описание каждого из вышеприведённых нефункциональных требований.

Система должна обеспечивать стабильную работу в условиях высокой волатильности криптовалютного рынка. Все ключевые модули должны проходить обязательное модульное тестирование, а критически важные компоненты – независимую верификацию перед интеграцией.

Генерация ежедневного прогноза и торгового сигнала должна выполняться в течение разумного времени (не более 2–3 минут с момента запуска). Генерация расширенного премиум-прогноза на горизонт до трёх месяцев не должна превышать 5 минут.

Архитектура системы должна позволять в дальнейшем расширять функциональность, в частности, добавлять поддержку других криптовалют, увеличивать горизонт прогнозирования и подключать дополнительные источники данных без значительной переработки ядра системы..

Интерфейс взаимодействия с пользователем (Telegram-бот) должен быть интуитивно понятным, обеспечивать четкое и структурированное

представление информации. Сообщения должны содержать понятные торговые сигналы, ключевые метрики прогноза и рекомендации без избыточной технической терминологии.

Программное обеспечение должно обладать модульной архитектурой, хорошо структурированным кодом и достаточным уровнем документирования, что обеспечит возможность дальнейшего развития и сопровождения системы.

LSTM-модель должна демонстрировать приемлемое качество прогнозирования на тестовых данных, оцениваемое с помощью метрик MAE, RMSE и MAPE. Особое внимание уделяется качеству многошагового (multi-step) прогнозирования.

Пользовательский интерфейс в виде Telegram-бота должен быть доступен на всех современных мобильных устройствах и операционных системах, поддерживающих приложение Telegram.

Соблюдение указанных нефункциональных требований позволит создать надежное, удобное и перспективное для дальнейшего развития программное обеспечение, соответствующее современным стандартам качества информационных систем.

1.6 Сравнительный анализ нейросетевых архитектур

В области прогнозирования финансовых временных рядов за последние годы сформировалось несколько основных подходов, основанных на нейросетевых архитектурах. Каждый из них имеет свои сильные и слабые стороны, что требует обоснованного выбора применительно к решаемой задаче – прогнозированию цены криптовалюты на горизонте до 30 дней.

Одним из первых типов нейронных сетей, применявшихся для последовательных данных, стали классические рекуррентные нейронные сети (Recurrent Neural Networks, RNN). Их ключевая особенность – наличие циклических связей, позволяющих теоретически учитывать предыдущие состояния при обработке текущего входа. Однако на практике RNN страдают от проблемы затухающих (vanishing) и взрывающихся (exploding) градиентов,

что делает обучение на длинных последовательностях крайне затруднительным. Как показало исследование Ardiansyah и соавторов (2025), LSTM демонстрирует превосходство над RNN по точности прогнозирования цен криптовалют, подтверждённое снижением значений MAPE и RMSE [10].

Архитектура LSTM (Long Short-Term Memory) была специально разработана для преодоления ограничений классических RNN. Введение вентилей забывания, входа и выхода позволяет сети избирательно сохранять или отбрасывать информацию на протяжении длительных временных интервалов. Для задачи прогнозирования цены криптовалюты этот механизм оказывается критически важным, поскольку рыночные тренды могут формироваться под влиянием событий, произошедших за много дней до текущего момента. Как отмечается в ряде исследований – модели на основе LSTM превосходили по точности прогноза стандартные статистические модели ARIMA и ETS [11]. Сравнение LSTM с ARIMA на данных криптовалюты показывает, что LSTM лучше справляется с улавливанием нелинейных зависимостей, присущих криптовалютному рынку.

В последние годы активно исследуются трансформеры (Transformer), которые используют механизм самовнимания и изначально были разработаны для задач обработки естественного языка. Трансформеры способны улавливать долгосрочные зависимости без рекуррентной обработки последовательности. Исследования показывают, что трансформер мог быть конкурентоспособен при прогнозировании дневной доходности акций, однако его эффективность сильно зависит от актива и горизонта.

Отдельного внимания заслуживают большие языковые модели (Large Language Models, LLM) и агентные подходы. LLM позволяют интегрировать в прогнозирование нечисловые данные – новостной фон, сообщения в социальных сетях, заявления регуляторов. Мультиагентные системы, например Multi-Agent криптовалюта Trading System, использующий DeepSeek-r1, демонстрируют способность адаптироваться к резким изменениям рыночной конъюнктуры.

Однако LLM и агентные подходы имеют существенные недостатки. Во-первых, они требуют значительных вычислительных ресурсов, вплоть до GPU-кластеров, и не всегда применимы в легковесных серверных решениях. Во-вторых, их работа основана на статистических закономерностях в текстах, а не на прямом моделировании временной динамики цены. В-третьих, для криптовалютного рынка, где основными драйверами являются технические закономерности и повторяющиеся паттерны, а не фундаментальный анализ, применение LLM может быть избыточным. Однако интеграция LLM требует сбора и обработки больших объёмов неструктурированных текстовых данных (новости, посты в социальных сетях), что выходит за рамки функциональных требований к разрабатываемой системе, ориентированной на анализ исключительно исторических цен.

Сравнительный анализ рассмотренных выше нейросетевых архитектур представлен ниже в таблице 2.

Таблица 2 – Сравнение нейросетевых архитектур

Критерий оценки	RNN	LSTM	Transformer	LLM/Agents
Преодоление затухания градиента	нет	да	да	да
Вычислительная сложность	низкая	средняя	высокая	очень высокая
Скорость обучения	высокая	средняя	низкая	очень низкая
Точность на крипто-данных	низкая	высокая	средняя	средняя

На основе проведённого сравнения для разрабатываемой системы в качестве основной архитектуры выбрана LSTM.

Данный выбор обусловлен следующими факторами:

- LSTM специально разработана для работы с долгосрочными зависимостями в последовательных данных, что соответствует природе финансовых временных рядов, где тренды формируются под влиянием событий, происходивших за дни или недели до текущего момента,

- многочисленные исследования подтверждают высокую точность LSTM при прогнозировании криптовалют, включая криптовалюта, с наилучшими показателями MAPE и RMSE по сравнению с классическими RNN,

– LSTM демонстрирует наилучшие результаты для задач прогнозирования волатильности, что критически важно для криптовалютного рынка с его высокой изменчивостью,

– Архитектура LSTM не требует таких вычислительных ресурсов, как трансформеры или LLM, что позволяет развернуть серверную часть системы на доступном оборудовании.

1.7 Подробный анализ архитектуры LSTM

Существует несколько вариаций LSTM, например:

– Vanilla LSTM (Standard LSTM) – базовая версия с одним слоем LSTM, обрабатывает входную последовательность шаг за шагом. Данная архитектура хорошо подходит для простых временных рядов, где зависимости между соседними элементами являются линейными и не требуют глубокой иерархии,

– Stacked LSTM (Deep LSTM) – несколько слоёв LSTM расположены друг над другом, что позволяет изучать более сложные зависимости в последовательностях. Каждый последующий слой работает с абстракциями, выделенными предыдущим, что особенно полезно для долгосрочных прогнозов на волатильных рынках, подобных криптовалютному,

– Bidirectional LSTM (BiLSTM) – последовательность обрабатывается в обоих направлениях, что позволяет модели понимать как прошлый, так и будущий контекст для каждого шага. Это эффективно для задач, где важна связь между всеми элементами последовательности (например, обработка естественного языка), но менее применимо для прогнозирования финансовых временных рядов, где будущее неизвестно.

В рамках данной работы было решено применять многослойную архитектуру, в связи с тем, что она показывала наилучшие результаты во множестве исследований посвящённых классификации временных рядов фондового рынка.

Теперь, для более глубокого понимания причин выбора архитектуры LSTM необходимо рассмотреть её внутреннее устройство и механизмы,

обеспечивающие её эффективность в задачах прогнозирования временных рядов.

Структура ячейки LSTM модели представлена на рисунке 11.

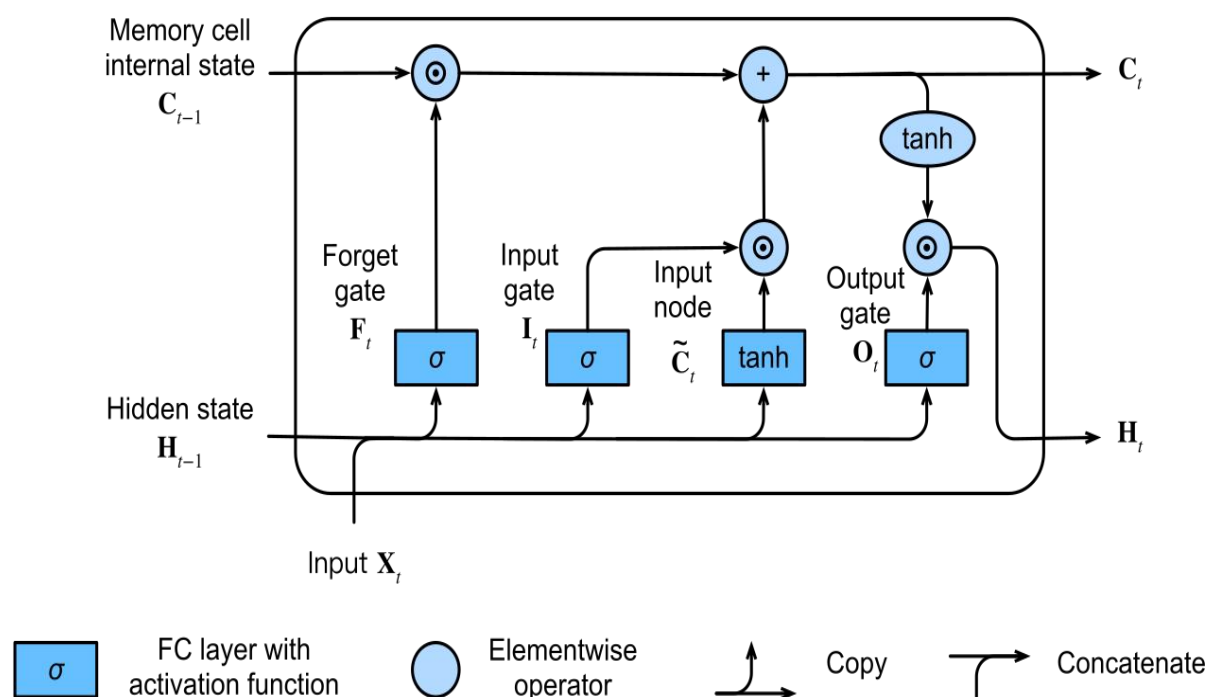


Рисунок 11 – Структура ячейки LSTM модели

Устройство и механизм работы LSTM. Архитектура LSTM была предложена Хохрайтером и Шмидхубером в 1997 году как решение проблемы затухающих градиентов, присущей классическим рекуррентным нейронным сетям. В отличие от обычного RNN, где информация передаётся через одно скрытое состояние, LSTM вводит дополнительный элемент – ячейку памяти (memory cell). Эта ячейка действует как конвейер, по которому информация может передаваться через многие временные шаги практически без изменений. Процесс обновления ячейки регулируется тремя специализированными вентилями (gates), каждый из которых представляет собой слой с сигмоидной функцией активации, возвращающей значения в диапазоне от 0 до 1:

Вентиль забывания (forget gate) определяет, какую информацию из предыдущего состояния ячейки следует отбросить. Значение, близкое к 0, означает «забыть», а близкое к 1 – «сохранить».

Входной вентиль (input gate) решает, какие новые данные следует сохранить в ячейке памяти. Сначала он определяет, какие значения обновить, затем слой гиперболического тангенса (tanh) создаёт вектор новых кандидатов.

Выходной вентиль (output gate) определяет, какая часть информации из текущего состояния ячейки попадёт в скрытое состояние (выход) сети, которое впоследствии используется для прогнозирования.

Благодаря такой архитектуре LSTM способна эффективно улавливать долгосрочные зависимости (long-term dependencies) в последовательных данных, что и делает её особенно ценной для анализа временных рядов на финансовых рынках.

Эффективность LSTM в прогнозировании криптовалют. В многочисленных исследованиях, опубликованных в 2025-2026 годах, подтверждается высокая эффективность архитектуры LSTM применительно к прогнозированию цены криптовалюты.

В исследовании Mubarak (2025), проведённом на данных о волатильности S&P 500 и криптовалюты, было установлено, что LSTM превосходит классические RNN как в условиях низкой, так и в условиях высокой волатильности. Гибридная модель GARCH-LSTM показала наилучшие результаты при прогнозировании высоковолатильных цен BTC/USD с MSE: 224 023 и MAE: 370,70.

В работе, опубликованной в 2025 году, сравнивались модели LSTM, GRU, CNN и Transformer для прогнозирования цены криптовалюты. Исследование показало, что LSTM и GRU эффективно решают проблему градиентов в RNN, повышая точность прогнозов за счёт специализированных вентильных механизмов.

Исследование 2026 года, посвящённое байесовской оптимизации LSTM моделей, продемонстрировало значительные возможности совершенствования этой архитектуры. Оптимизация LSTM на основе

технических индикаторов позволила сократить ошибку прогнозирования цены криптовалюты на 36,4% по сравнению с базовой моделью.

Сравнительный анализ, проведённый в 2026 году, показал, что LSTM и GRU обеспечивают наилучшую общую точность прогнозирования среди широкого спектра методов машинного обучения. Причём для криптовалюты LSTM является предпочтительным выбором для прогнозирования на горизонте до 30 дней, что коррелирует с задачами настоящей работы. В то же время GRU может быть более эффективна для 7-дневных прогнозов.

Таким образом, архитектура LSTM представляет собой оптимальный выбор для системы прогнозирования цены криптовалюты. Её встроенная вентильная структура позволяет эффективно моделировать нелинейные закономерности криптовалютного рынка, а многочисленные современные исследования подтверждают её превосходство над классическими методами и конкурентоспособность в сравнении с более новыми архитектурами, такими как трансформеры.

Также важно отметить совокупность научных данных подтверждающую, что двухслойная Stacked LSTM обеспечивает критически важное преимущество перед однослойными и двунаправленными моделями применительно к конкретной бизнес-задаче разрабатываемой системы. Это обусловлено её способностью эффективно извлекать иерархические временные зависимости, характерные для цен криптовалюты, при сохранении устойчивости к переобучению и приемлемых вычислительных затратах.

2 Проектирование системы

2.1 Выбор технологий и инструментальных средств разработки

В процессе разработки приложения было решено использовать методологию RUP. Разработка приложения в соответствии с методологией RUP носит итеративный характер. На каждой итерации перед разработчиками ставятся определённые цели, которые должны быть достигнуты к её концу. В результате в конце каждой итерации должна получиться промежуточная функциональная версия разрабатываемого ПО. В методологии RUP существуют 4 основных фазы, каждая из которых может включать в себя одну или несколько итераций [12]:

- фаза начала проекта – формируется общее видение проекта и экономическое обоснование,
- уточнение – анализ предметной области и построение архитектуры продукта,
- построение – реализация основного функционала продукта,
- внедрение – получение финальной версии продукта и его передача заказчику, фаза включает в себя бета-тестирование и устранение выявленных в его процессе ошибок, а также обучение пользователей.

Каждая фаза завершается вехой, на которой команда разработчиков оценивает выполнение поставленных целей и принимает критически важные решения о ходе дальнейшей работы над продуктом.

Выбор в пользу данной методологии продиктован рядом её достоинств. С одной стороны, методология RUP обеспечивает гибкость процесса разработки и позволяет быстро вносить коррективы в проект в случае выявления в нем ошибок или изменения требований к разрабатываемому приложению. Кроме того, благодаря данной методологии удаётся выявить основные риски уже на ранних стадиях разработки. При этом RUP, в отличие от некоторых других популярных методологий, например XP, не требует постоянного взаимодействия с заказчиком, которое в рамках данного проекта невозможно.

Для анализа требований, а также моделирования структуры проекта в соответствии с принципами RUP было решено использовать диаграммы на языке моделирования UML [13]. Для построения диаграмм использовалось средство UML-моделирования Visual Paradigm.

Для реализации программного обеспечения было решено использовать среду программирования Microsoft Visual Studio Code и язык программирования Python ввиду его широкого распространения в области машинного обучения, наличия мощной экосистемы библиотек и удобства разработки. Visual Studio Code предоставляет встроенную поддержку Git, терминал, отладчик и огромное количество расширений, что позволяет эффективно вести разработку, тестировать фрагменты кода и профилировать производительность модели. Выбор Python обусловлен не только доминированием в Data Science, но и возможностью быстрого прототипирования: динамическая типизация, богатый выбор фреймворков и обширная документация ускоряют итерации при разработке LSTM-модели и подборе гиперпараметров для нее.

Основными инструментальными средствами и библиотеками, использованными при разработке, являются:

- TensorFlow и Keras – для создания, обучения и инференса рекуррентной нейронной сети LSTM,
- Pandas и NumPy – для загрузки, предобработки и манипуляции с временными рядами финансовых данных,
- scikit-learn – для масштабирования данных (MinMaxScaler) и оценки качества модели,
- python-telegram-bot – для реализации клиентской части в виде Telegram-бота и отправки уведомлений,
- Matplotlib и Seaborn – для визуализации прогнозов и торговых точек,
- Optuna – для оптимизации гиперпараметров модели.

Ниже приведено краткое описание каждого из вышеперечисленных инструментальных средств.

TensorFlow представляет собой промышленный фреймворк для глубокого обучения, обеспечивающий высокую производительность при инференсе моделей на серверной стороне. Keras, встроенный в TensorFlow как высокоуровневый API, позволяет быстро конструировать и настраивать LSTM-модели с минимальным объёмом кода, что особенно ценно при итеративной оптимизации архитектуры.

Pandas предоставляет удобные структуры данных (DataFrame) для работы с временными рядами, включая загрузку CSV-файлов, обработку пропусков и сдвиги окон. NumPy лежит в основе всех численных операций и обеспечивает эффективную работу с многомерными массивами, которые подаются на вход LSTM.

Scikit-learn используется для предобработки данных (MinMaxScaler – нормализация цен в диапазон $[0,1]$, необходимая для стабильного обучения LSTM) и для расчёта метрик качества прогнозов (MSE, RMSE, MAE), что позволяет стандартизированно оценивать точность модели.

Библиотека Python-telegram-bot предоставляет полный спектр инструментов для создания Telegram-ботов: обработка команд, инлайн-клавиатуры, управление состоянием диалога. Выбрана за счёт своей зрелости, хорошей документации и совместимости с синхронным и асинхронным стилями программирования.

Matplotlib и Seaborn. Используются на этапе исследования и презентации результатов для визуализации прогнозов (наложение предсказанных цен на фактические) и отображения точек входа/выхода при тестировании торговой стратегии, что делает выводы более наглядными.

Optuna – это фреймворк для автоматической оптимизации гиперпараметров. Позволяет значительно сократить время поиска наилучшей конфигурации (число слоёв, нейронов, dropout и др.) по сравнению с ручным перебором, используя байесовские методы и механизм «pruning» для отбраковки неудачных экспериментов.

Выбор данных технологий обусловлен их высокой производительностью, открытым исходным кодом, обширной документацией и проверенной эффективностью при решении задач прогнозирования временных рядов. Модульная архитектура проекта обеспечивает независимую разработку, тестирование и сопровождение отдельных компонентов системы.

2.2 Проектирование архитектуры программного обеспечения

Было принято решение использовать в приложении двухуровневую архитектуру «клиент-сервер». Её схема представлена на рисунке 12.



Рисунок 12 – Двухуровневая архитектура «клиент-сервер»

В данной архитектуре на сервере располагаются база данных и СУБД, а также серверная часть приложения, которая отвечает за непосредственное взаимодействие с базой данных и, как правило, реализуется в виде интерфейса прикладного программирования (API).

Клиентская же часть разрабатываемого ПО обеспечивает для пользователя удобный интерфейс, позволяющий пользователю получить прогноз, а также реализует возможность удалённо обращаться к серверу. В нашем случае его роль выполняет мобильное приложение.

Взаимодействие клиента и сервера осуществляется следующим образом: клиент делает HTTP-запрос серверу. Серверная часть приложения, в свою очередь, на основе полученного запроса формирует прогноз при помощи LSTM модели и возвращает его результаты серверу, который на основе полученных результатов формирует ответ на запрос клиента. Получив ответ, клиент представляет его пользователю в удобной для того форме.

Достоинством архитектуры является то, что все данные хранятся на стороне сервера. Это позволяет отказаться от полноценного клиентского приложения, заменив его Telegram-ботом. Кроме того, ещё одним достоинством можно выделить максимально большой круг потенциальных пользователей поскольку Telegram является WEB-сервисом, а также имеет свое собственное кросс-платформенное приложение.

2.3 Проектирование серверной части системы

Серверная часть приложения представляет собой REST API. Доступ к данным осуществляется при помощи протокола HTTP с использованием его стандартных методов (GET применяется для получения данных ресурса, POST – для создания нового ресурса, PUT – для обновления существующего, DELETE – для удаления ресурса). Обращение к ресурсам происходит при помощи уникальных идентификаторов ресурсов (URI).

Серверная часть приложения включает в себя модуль доступа к данным, отвечающий за взаимодействие с Bybit API для получения и предобработки истории цен криптовалюты, а также модуль прогнозирования на основе LSTM-модели отвечающий за прогнозирование цены и модуль реализующий бизнес-логику расчёта торговой стратегии и генерации сигналов. Отдельный API-контроллер обрабатывает HTTP-запросы клиента, проверяет права доступа (обычный или премиум-пользователь) и формирует структурированные ответы.

2.4 Проектирование LSTM-модели

Исследования показывают, что двухслойные (stacked) архитектуры LSTM обеспечивают более высокую точность прогнозирования по

сравнению с однослойными, поскольку второй слой позволяет извлекать более абстрактные временные признаки, выявляя иерархические паттерны в последовательных данных [14]. В то же время дальнейшее увеличение глубины сети приводит к чрезмерному усложнению модели, росту вычислительных затрат и риску переобучения. Таким образом, оптимальным для данной задачи является использование двух LSTM-слоёв, что обеспечивает баланс между выразительной способностью сети и её устойчивостью к переобучению.

Размерность же LSTM-слоя, другими словами – количество блоков памяти, определяет способность сети хранить и обрабатывать информацию о временных закономерностях. Каждый нейрон в LSTM-слое содержит полный набор вентилей (забытия, входа и выхода) и ячейку памяти, что делает его вычислительно более ёмким по сравнению с нейроном в обычной RNN. Увеличение числа нейронов повышает выразительную способность сети, но также увеличивает количество обучаемых параметров и риск переобучения. Для прогнозирования цены криптовалюты наиболее часто применяются значения в диапазоне от 32 до 128 нейронов на слой, что в рамках данной работы будет подлежать автоматическому подбору через Optuna.

Для предотвращения переобучения в LSTM-слои встраивается механизм Dropout: во время обучения случайным образом отключается заданный процент нейронов, что вынуждает сеть не полагаться на отдельные нейроны, а распределять информационную нагрузку между ними. В качестве связующего звена между LSTM-слоями применяется полносвязный слой с ReLU (Rectified Linear Unit) – функцией активации, преобразующей выходной сигнал LSTM в нелинейное пространство признаков, которое затем подаётся на выходной слой. ReLU позволяет избежать проблемы затухающих градиентов благодаря своей линейной природе при положительных входах и способствует разреженности активаций.

Также очень немаловажной частью является оптимизатор, он определяет алгоритм обновления весов сети в процессе обучения на основе

вычисленных градиентов функции потерь. Наиболее распространённым выбором для LSTM-моделей является Adam (Adaptive Moment Estimation), сочетающий в себе идеи Momentum (учёт предыдущих градиентов для ускорения сходимости) и RMSprop (адаптивное изменение скорости обучения для каждого параметра). Adam автоматически регулирует темп обучения для каждого параметра на основе оценок первого и второго моментов градиента, что особенно ценно для нестационарных временных рядов, где оптимальная скорость обучения может меняться в процессе обучения. Исследования показывают, что Adam является предпочтительным выбором при автоматизированном поиске гиперпараметров благодаря своей устойчивости к выбору начальной скорости обучения и эффективности работы с зашумленными градиентами, характерными для финансовых данных, поэтому было принято решение использовать именно его.

Не стоит забывать и про важность подбора оптимальных гиперпараметров LSTM-модели, ведь это один из наиболее критичных и трудоёмких этапов проектирования, поскольку от правильности выбора зависит итоговая точность прогнозирования. Для решения этой задачи в рамках диссертационной работы применяется фреймворк Optuna, представляющий собой инструмент автоматизированной оптимизации гиперпараметров на основе байесовского подхода.

Традиционный подход к подбору гиперпараметров либо требует огромных вычислительных ресурсов, либо не гарантирует нахождения глобального оптимума. Optuna в отличие от них строит вероятностную модель зависимости ошибки модели от гиперпараметров и целенаправленно исследует наиболее перспективные области пространства параметров, что позволяет значительно сократить количество необходимых экспериментов. Ключевые преимущества Optuna перед альтернативными решениями (Bayesian Optimization, Hyperband) включают: возможность динамического определения пространства параметров, встроенную поддержку ранней остановки неперспективных экспериментов (pruning), эффективную работу с

категориальными и непрерывными параметрами, а также высокую масштабируемость за счёт асинхронного выполнения параллельных оптимизационных экспериментов.

Процесс оптимизации в Optuna организуется вокруг трёх ключевых понятий: `objective` (целевая функция), `trial` (одна оптимизационная итерация) и `study` (серия `trials`, образующих оптимизационный эксперимент). На каждой итерации Optuna предлагает набор гиперпараметров из заданного пространства поиска, используя алгоритм Tree-structured Parzen Estimator (TPE): строит два распределения – хороших и плохих значений ошибки, – после чего выбирает параметры, максимизирующие вероятность принадлежности к группе хороших. Процесс повторяется итеративно, постепенно сужая область поиска вокруг наилучших конфигураций.

Оптимальная конфигурация определяется по минимуму значения выбранной метрики ошибки (например, RMSE) на валидационной подвыборке.

Для количественной оценки эффективности LSTM-модели используется комплекс метрик, каждая из которых отражает различные аспекты точности прогнозирования.

Mean Squared Error (MSE) – средний квадрат разности между фактическими и прогнозируемыми значениями. Ключевое свойство MSE заключается в том, что она придаёт больший вес крупным ошибкам (благодаря возведению в квадрат), что особенно важно для прогнозирования цены криптовалюты, где значительное отклонение прогноза от фактической цены может привести к неверному торговому решению и финансовым потерям. Благодаря своей гладкости MSE широко применяется в качестве функции потерь для обучения LSTM-моделей, обеспечивая эффективную работу градиентного спуска.

Root Mean Squared Error (RMSE) – квадратный корень из MSE, возвращающий показатель ошибки в тех же единицах, что и целевая переменная (доллары США), что делает его интуитивно понятным для

интерпретации. RMSE также акцентирует внимание на крупных ошибках. RMSE помогает определить, насколько прогнозы модели близки к реальным значениям: меньшие значения указывают на лучшую точность прогнозов.

Mean Absolute Error (MAE) – среднее абсолютное отклонение прогноза от фактического значения. В отличие от MSE и RMSE, MAE не возводит ошибки в квадрат, поэтому она менее чувствительна к выбросам и даёт более робастную оценку «типичной» ошибки модели. Особенно это ценно для криптовалютного рынка, где резкие ценовые скачки могут искажать метрики, основанные на квадрате ошибки. Однако MAE имеет и недостаток: её недифференцируемость в нуле делает её менее удобной в качестве функции потерь при обучении.

На этапе обучения LSTM-модели в качестве функции потерь используется MSE, обеспечивающая гладкий градиент для эффективной оптимизации. Для мониторинга процесса обучения и ранней остановки (early stopping) применяется MAE, которая даёт более робастную оценку качества при изменяющихся рыночных условиях. Финальная оценка обученной модели на тестовой выборке производится с использованием RMSE как наиболее интерпретируемой метрики, позволяющей сравнить точность прогнозирования с альтернативными подходами.

Такой многоуровневый подход к оценке качества позволяет комплексно анализировать поведение модели и обеспечивает объективность при выборе оптимальной конфигурации гиперпараметров. По итогам оптимизации значения выбранных метрик будут сопоставлены с базовыми моделями – статистическими и машинного обучения – для подтверждения эффективности предложенного подхода.

2.5 Проектирование клиентской части системы

Клиентская часть разрабатываемого программного обеспечения представляет собой Telegram-бота, выполняющего ежедневную рассылку прогнозов стоимости криптовалюты, а также обладающего дополнительным

функционалом, доступным через inline-меню, располагающееся под сообщениями бота.

На основе выявленных функциональных требований был спроектирован интерфейс бота. Макет интерфейса приведен на рисунке 13.

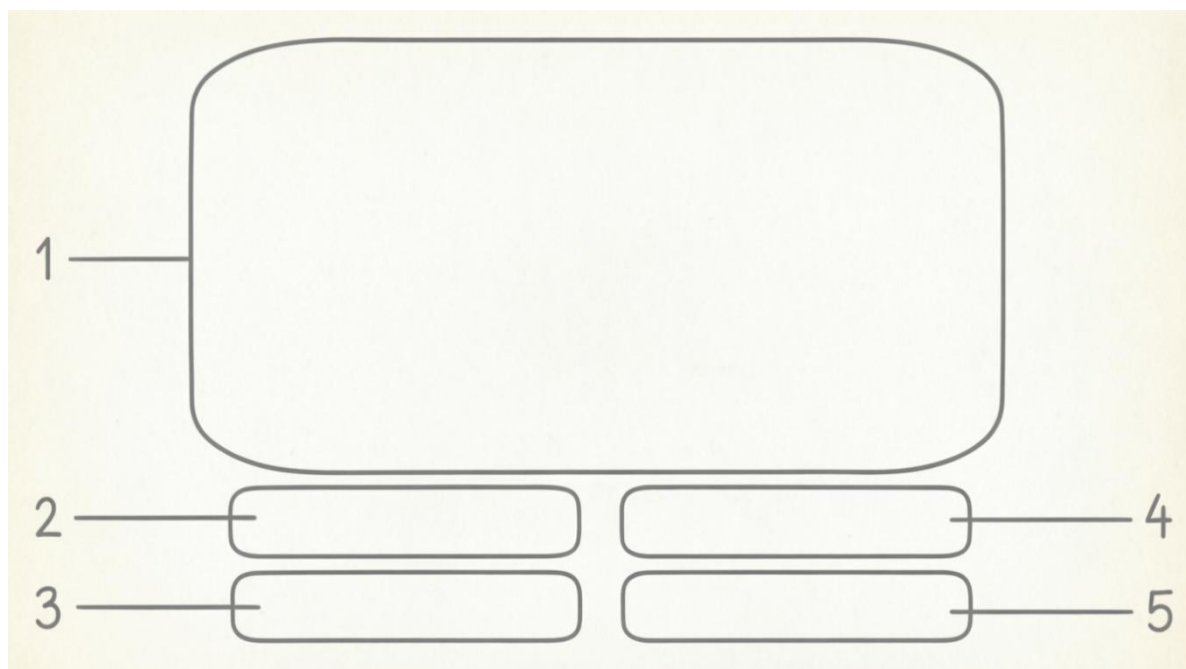


Рисунок 13 – Макет интерфейса бота

Рассмотрим основные элементы интерфейса: 1 – сообщение отправленное ботом, 2 – кнопка получения обычного прогноза, 3 – кнопка оплаты премиум-подписки, 4 – кнопка получения премиум прогноза, 5 – кнопка изменения горизонта прогнозирования.

При нажатии на кнопку «Обычный прогноз» на экран пользователю будет выведен соответствующий прогноз, актуальный на данный момент времени.

При нажатии кнопки «Оплатить подписку» пользователь будет перенаправлен в браузер на страницу платежной системы. После оплаты подписки система изменяет статус пользователя на премиум.

При нажатии на кнопку «Изменить горизонт прогноза» система проверяет, активна ли подписка у данного пользователя. Если нет, будет выведено сообщение об отсутствии премиум подписки. Иначе на экран пользователю будет выведено сообщение с предложением указать желаемый

горизонт прогнозирования. Система сохранит указанную длину и впредь будет выводить соответствующий прогноз.

При нажатии на кнопку «Премиум прогноз» система проверяет, активна ли подписка у данного пользователя. Если нет, будет выведено сообщение об отсутствии премиум подписки. Иначе на экран пользователю будет выведен соответствующий прогноз, актуальный на данный момент времени.

Проектируемый интерфейс бота полностью соответствует выявленным функциональным требованиям и обеспечивает интуитивно понятное взаимодействие пользователя с системой. Четыре основные кнопки меню покрывают все ключевые сценарии использования: получение базового прогноза, управление подпиской, настройка горизонта прогнозирования и запрос расширенного прогноза для премиум-пользователей. Логика проверки статуса подписки при нажатии на кнопки «Премиум прогноз» и «Изменить горизонт прогноза» предотвращает несанкционированный доступ к платному функционалу и своевременно информирует пользователя о необходимости оплаты. Размещение кнопок непосредственно под сообщениями бота (inline-меню) сокращает количество действий, требуемых для получения результата, и делает интерфейс удобным даже для неопытных пользователей. Таким образом, спроектированный интерфейс Telegram-бота в полной мере решает поставленные задачи, обеспечивая как быстрый доступ к обычному прогнозу, так и гибкую работу с премиум-возможностями.

3 Реализация системы

3.1 Реализация получения и предобработки данных

Источником данных выступает исторический временной ряд цен закрытия криптовалюты, полученный через BybitAPI. Для обеспечения качества входных данных выполняется проверка на наличие пропусков с последующей интерполяцией для восстановления непрерывности ряда. Масштабирование значений выполняется с помощью MinMaxScaler, приводящего все цены к диапазону $[0, 1]$ для ускорения сходимости градиентных методов оптимизации. Применяется техника скользящего окна (sliding window), с помощью которой исходный ряд преобразуется в обучающие последовательности: каждая последовательность длины L (количество предыдущих дней, используемых для предсказания) служит входом модели, а целевым значением выступает цена на следующий день (для одношагового прогноза). После преобразования выполняется разделение выборки на: обучающую, валидационную и тестовую части, в соотношении 70/20/10 соответственно. Причём для временных рядов разделение осуществляется в хронологическом порядке без перемешивания, чтобы сохранить временную структуру и избежать «утечки» будущих данных в процесс обучения.

3.2 Разработка и обучение LSTM-модели

Разработка LSTM-модели производилась в среде программирования Microsoft Visual Studio Code на языке Python с применением фреймворков TensorFlow и Keras [15, 16].

Разработанная модель LSTM имеет многослойную структуру, типичную для задач прогнозирования временных рядов. Первый слой – входной, принимающий последовательность заданной длины. Далее следуют два LSTM-слоя с 171 нейронами в каждом и функцией активации ReLU, отвечающие за извлечение временных зависимостей. Для предотвращения переобучения после каждого LSTM-слоя применяется dropout с коэффициентом 0,1411. Завершает сеть полносвязный слой (Dense) с

линейной активацией, преобразующий выходные признаки последнего LSTM-слоя в скалярное предсказание – прогнозируемую цену закрытия.

Модель компилируется с функцией потерь Mean Squared Error (MSE) , которая минимизирует среднеквадратичное отклонение предсказанных значений от фактических. Оптимизатором выступает Adam с начальной скоростью обучения 0,000756 – адаптивный метод первого порядка, хорошо зарекомендовавший себя при обучении рекуррентных сетей. Процесс обучения выполняется в течение 50 эпох с размером мини-батча 32. Для мониторинга качества используется метрика Mean Absolute Error (MAE) на валидационной выборке

Данные гиперпараметры модели были подобраны при помощи фреймворка Optuna.

Результат обучения приведен на рисунке 14.

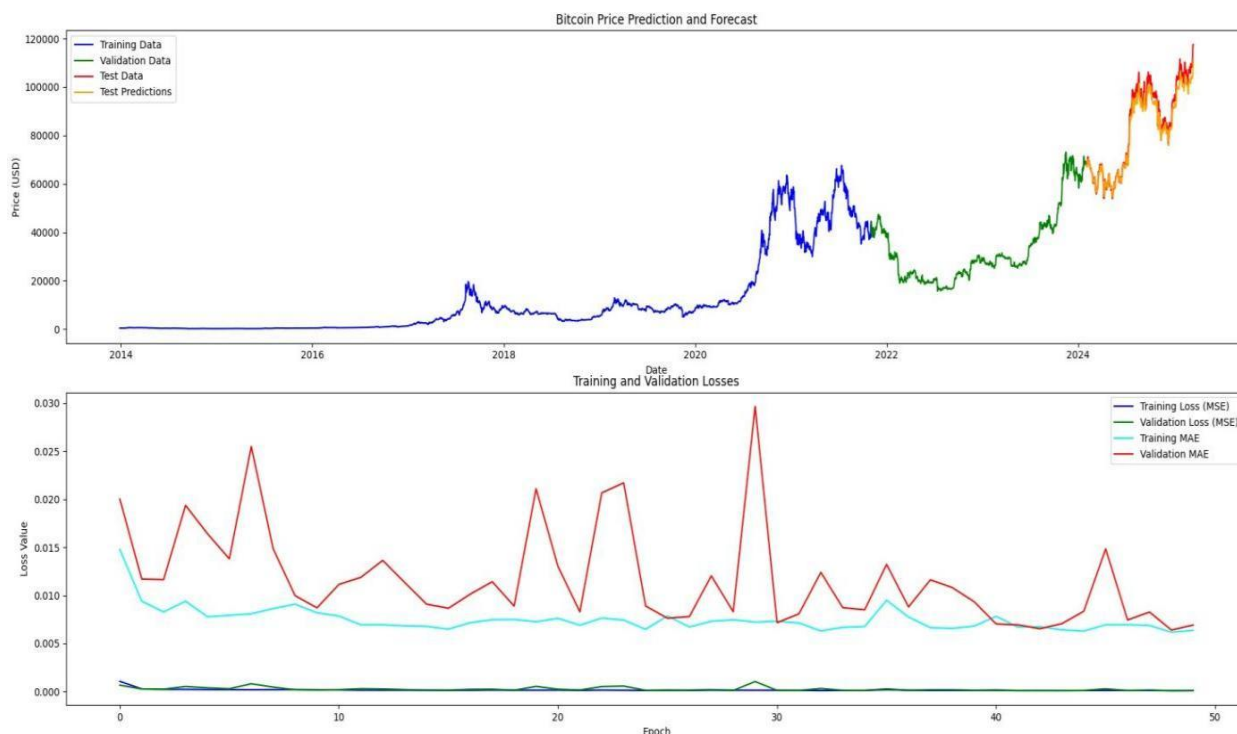


Рисунок 14 – Результат обучения LSTM-модели

Обученная модель сохраняется в формате HDF5 (Hierarchical Data Format Version 5) с расширением .h5, с показателем MAE равным 1777.

Листинг разработанной LSTM представлен в приложении А.

В серверной части системы эта модель загружается в модуль прогнозирования, где используется для инференса по запросам как от обычных, так и от премиум-пользователей.

3.3 Разработка модуля торговой стратегии

Модуль торговой стратегии является связующим звеном между прогностической LSTM-моделью и практическими торговыми решениями. Если ранее описанная модель решает задачу прогнозирования цены, то модуль торговой стратегии преобразует эти прогнозы в конкретные торговые сигналы – покупка, продажа или удержание позиции. Для криптовалютного рынка с его высокой волатильностью особенно важна тщательная настройка генерации сигналов и системы управления рисками. В данной работе разработана торговая стратегия, основанная на выходных значениях LSTM-модели, дополненная классическими фильтрами для снижения ложных срабатываний.

Обоснование выбора типа стратегии. В зависимости от рыночной конъюнктуры выделяют два основных типа торговых стратегий: следование тренду (trend-following) и возврат к среднему (mean reversion). Трендовые стратегии приносят прибыль за счёт нескольких крупных сделок в периоды устойчивого движения цены, тогда как стратегии возврата к среднему зарабатывают на многочисленных мелких сделках при колебании цены вокруг своего среднего значения. В рамках данной работы реализована трендовая стратегия, ориентированная на долгосрочные прогнозы LSTM. Рынок криптовалюты демонстрирует выраженные трендовые периоды, что делает такой подход более надёжным. При этом для снижения рисков в периоды флета (бокового движения) модуль включает механизмы фильтрации ложных сигналов.

Генерация торговых сигналов. LSTM-модель возвращает массив прогнозных цен на заданный горизонт (до 30 дней). Модуль стратегии вычисляет ожидаемое направление движения как разность между прогнозируемой ценой на последний день горизонта и текущей ценой. На

основе этого направления формируются дискретные торговые сигналы. Для снижения ложных срабатываний вследствие шума во входных данных применяется пороговая фильтрация. Сигнал покупки генерируется, если прогнозируемый рост превышает пороговое значение, сигнал продажи – если прогнозируемое падение превышает то же пороговое значение с отрицательным знаком. При отсутствии уверенного сигнала (прогнозируемое изменение не превышает порог) модуль рекомендует оставаться вне позиции. Такой пороговый подход широко применяется в практике автоматизированной торговли с использованием LSTM-моделей. Выбор значения порога осуществляется на основе максимизации коэффициента Шарпа на исторических данных, что позволяет настроить стратегию с учётом баланса между прибыльностью и риском.

Модуль торговой стратегии выполняется на серверной стороне системы и вызывается после получения прогноза от LSTM-модели. Результаты расчёта сигналов могут использоваться для уведомления пользователя о потенциальных точках входа.

3.4 Разработка Telegram-бота

Telegram-бот выполняет функцию клиентской части разрабатываемой системы и обеспечивает интерфейс взаимодействия конечного пользователя с серверной инфраструктурой.

Навигация по функциональным возможностям организована через inline-меню, отображаемое под каждым сообщением бота.

Для получения обычного прогноза пользователь нажимает кнопку «Получить обычный прогноз». Бот отправляет GET-запрос на эндпоинт `/predict/basic`, передавая идентификатор пользователя. Сервер возвращает прогноз цену на следующий день, который бот оформляет в виде сообщения с округлением до двух знаков и указанием валюты (USD). При возникновении ошибки (недоступность сервера, отсутствие данных) бот уведомляет пользователя и предлагает повторить попытку позже.

При нажатии кнопки «Оплатить подписку» бот инициирует платёжный процесс. Для этого используется встроенная платёжная система Telegram Stars или формирование ссылки на внешний платёжный шлюз. После успешной оплаты бот получает уведомление (через webhook или прямое подтверждение от платёжного сервиса), отправляет PUT-запрос на сервер для обновления статуса пользователя, после чего выводит сообщение об активации премиум-доступа.

Получение премиум-прогноза доступно только пользователям, у которых в базе данных сервера установлен активный статус `is_premium = True`. При нажатии соответствующей кнопки бот отправляет запрос на эндпоинт `/predict/premium`, включая `user_id`. Сервер проверяет статус подписки и текущий горизонт прогноза, возвращает массив прогнозных цен на требуемое количество дней. Бот форматирует этот массив в виде нумерованного списка. Если подписка отсутствует, сервер возвращает код ошибки, и бот выводит сообщение с предложением оплатить подписку и кнопкой для перехода к оплате.

Также премиум-пользователь может воспользоваться пунктом меню «Изменить горизонт прогнозирования». Бот запрашивает ввод числа от 1 до 30 и ожидает текстового сообщения. После получения корректного значения бот отправляет POST-запрос на `/user/horizon` с параметрами `userId` и `newHor`. Сервер сохраняет новое значение горизонта, после чего бот подтверждает изменение.

Бот разработан с учётом возможности временных сбоев сервера или сети. При получении HTTP-статуса 5XX или таймауте соединения бот информирует пользователя о технических неполадках и предлагает повторить действие через несколько минут.

Для идентификации пользователя используется `chat_id` Telegram, который передаётся во всех запросах к серверу. Бот не хранит состояние пользователя локально, полагаясь на серверную базу данных как единственный источник истины о статусе подписки и настройках.

Реализация и развёртывание. Бот написан на Python с использованием библиотеки `python-telegram-bot`, поддерживающей асинхронную обработку сообщений и вызовов. Взаимодействие с сервером осуществляется через HTTP-клиент, а именно библиотеку `requests`. Бот может быть развёрнут на том же сервере, что и REST API, либо как отдельный контейнер. Для работы бота необходимо получить токен авторизации у BotFather в Telegram.

Таким образом, Telegram-бот выступает в роли удобного и интуитивно понятного интерфейса, скрывающего сложность вычислительных модулей и обеспечивающего доступ к прогнозам как обычным, так и премиум-пользователям в соответствии с их правами.

3.5 Развёртывание разрабатываемой системы

Диаграмма развертывания, на которой описывается физическое расположение компонентов системы, представлена на рисунке 15.

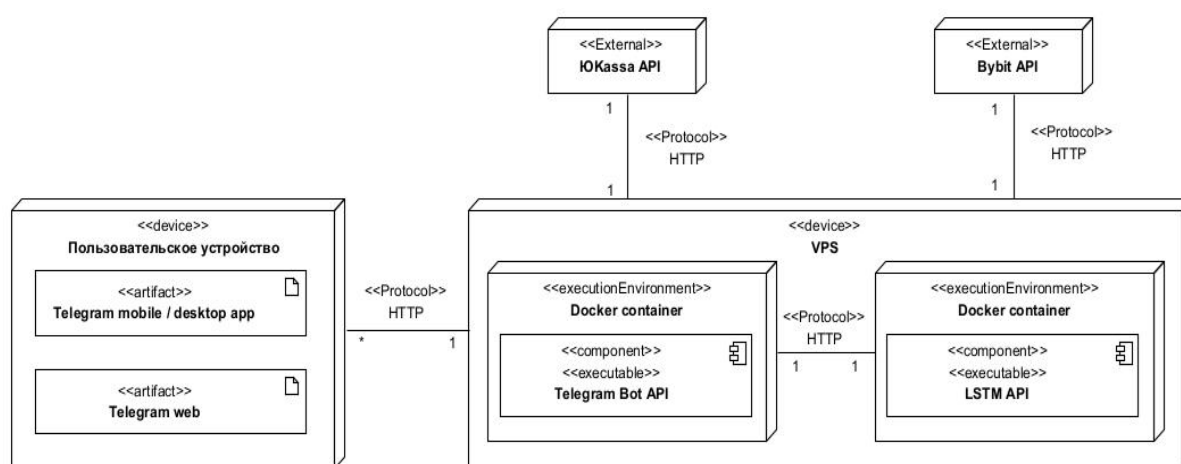


Рисунок 15 – Диаграмма развертывания мобильного приложения

Спроектированная в предыдущих разделах система имеет распределённую архитектуру, включающую несколько взаимодействующих компонентов. Диаграмма развёртывания иллюстрирует физическое размещение и связи между элементами системы. Пользователь взаимодействует с системой через один из клиентских интерфейсов Telegram – мобильное приложение, десктопную версию или веб-версию. Клиентское устройство пользователя соединяется с серверной инфраструктурой через публичную сеть Интернет по протоколу HTTP.

Серверная часть системы размещается на виртуальном выделенном сервере (VPS), расположенном за пределами Российской Федерации. Данное архитектурное решение обусловлено особенностями функционирования мессенджера Telegram на территории РФ и будет подробно обосновано в следующем подразделе.

На VPS развёрнуты два основных компонента, каждый из которых функционирует в отдельном Docker-контейнере для обеспечения изолированности, воспроизводимости и простоты управления. Первый контейнер содержит Telegram Bot API – исполнительный компонент, реализующий логику клиентской части системы. Он обрабатывает входящие сообщения и команды от пользователей, отправляет HTTP-запросы к API прогнозирования и форматирует полученные ответы. Второй контейнер включает LSTM API – сервер прогнозирования, реализующий бизнес-логику системы. Данный компонент загружает предварительно обученную LSTM-модель, предоставляет REST-эндпоинты для получения обычных и премиум-прогнозов, управляет пользовательскими данными (статус подписки, горизонт прогнозирования), хранящимися в CSV-файлах.

Контейнеры взаимодействуют между собой по протоколу HTTP внутри виртуальной сети, создаваемой Docker. Telegram Bot API выступает в роли клиента по отношению к LSTM API, отправляя запросы на соответствующие эндпоинты. Доступ к контейнерам из внешней сети осуществляется через проброшенные порты, при этом для повышения безопасности и управляемости трафиком может быть использован reverse-проxy (например, Nginx), принимающий внешние HTTP-запросы и направляющий их к соответствующему внутреннему сервису. При необходимости оба компонента могут быть объединены в один контейнер, однако раздельное развёртывание предпочтительнее с точки зрения гибкости масштабирования и управления.

Такая контейнеризованная архитектура обеспечивает полную изоляцию окружений: LSTM API может использовать одни версии библиотек

(например, TensorFlow для загрузки модели), а Telegram Bot API – другие, и их конфликты исключены. При переносе системы на другой сервер достаточно скопировать образы и конфигурационные файлы, что гарантирует идентичность поведения в средах разработки, тестирования и эксплуатации.

3.5.1 Выбор подходящего VPS

При проектировании системы критически важным является обеспечение бесперебойного взаимодействия с API Telegram. По состоянию на 2026 год в Российской Федерации действуют существенные технические ограничения в отношении мессенджера Telegram. По сообщениям ряда изданий, в апреле 2026 года уровень блокировки Telegram в России приблизился к 100%. В этих условиях развёртывание серверной части системы на хостингах, расположенных на территории России, делает её недоступной для пользователей либо приводит к значительным задержкам.

Таким образом, для обеспечения стабильной работы разрабатываемой системы требуется VPS, расположенный в юрисдикции с неблокируемым доступом к API Telegram, то есть за пределами РФ. Для размещения ботов и их подключения к API Telegram теперь необходимо использовать VPS с серверами за пределами России. Оптимальным вариантом являются зарубежные провайдеры.

Кроме того, отдельного рассмотрения требует вопрос соответствия разрабатываемой системы требованиям законодательства Российской Федерации в области персональных данных. Согласно ст. 3 Федерального закона от 27.07.2006 № 152-ФЗ «О персональных данных», персональными данными признаётся любая информация, относящаяся прямо или косвенно к определённому или определяемому физическому лицу [25]. В разрабатываемой системе хранятся три типа данных, связанных с пользователем: идентификатор пользователя в Telegram (`user_id`), статус подписки (`is_premium`) и выбранный горизонт прогнозирования. Ключевым для правовой квалификации является вопрос: позволяет ли идентификатор пользователя Telegram установить его личность? Постольку поскольку

Telegram ID не раскрывает напрямую фамилию, имя, отчество, адрес, номер телефона или иные сведения, позволяющие идентифицировать конкретного человека. Судебная практика по делам, связанным с применением Закона № 152-ФЗ, указывает, что данные, не позволяющие идентифицировать конкретного субъекта персональных данных, в терминологии Закона № 152-ФЗ персональными данными не являются. Таким образом, хранение исключительно Telegram ID, статуса подписки и горизонта прогнозирования без привязки к иным идентификационным данным не подпадает под регулирование Федерального закона № 152-ФЗ, что позволяет размещать систему на зарубежных серверах без нарушения российского законодательства.

Исходя из ранее упомянутых критериев в качестве хостинг-провайдера для развертывания системы был выбран Hetzner, а именно тарифный план CX22, физически расположенный в крупном дата-центре Германии г. Фалькенштайн. Выбранный тариф предоставляет доступ к 2 vCPU, 4 ГБ оперативной памяти и 40 ГБ SSD-диска. Стоимость аренды составляет 3,79 евро в месяц, что в 2,5 раза дешевле аналогичного предложения OVHcloud стоимостью 9,99 долларов и в 6 раз дешевле конфигурации на DigitalOcean за 24 доллара. Указанных вычислительных ресурсов достаточно для поддержки LSTM-модели, выполнения инференса и одновременной обработки до сотни запросов в минуту без деградации производительности.

3.5.2 Контейнеризация системы с использованием Docker

Docker – это программное обеспечение с открытым исходным кодом, применяемое для разработки, тестирования, доставки и запуска веб-приложений в средах с поддержкой контейнеризации. Docker предоставляет возможность упаковывать и запускать приложение в изолированной среде, называемой контейнером. Говоря простыми словами, контейнерная упаковка Docker позволяет запустить больше кода на каждом используемом сервере, что оптимизирует расход ресурсов, обеспечивает облегчённую систему

переноса приложения из среды разработки и тестирования в продуктивную среду с последующим развертыванием.

Выбор Docker в качестве платформы контейнеризации обусловлен несколькими ключевыми преимуществами. Во-первых, Docker обеспечивает стабильные среды: контейнеры гарантируют, что приложения работают в одной и той же среде независимо от того, где они развёрнуты. Это решает фундаментальную проблему расхождения сред разработки и эксплуатации – приложение, собранное в образ, работает идентично на компьютере разработчика, на сервере тестирования и в production-окружении. Во-вторых, Docker обеспечивает изоляцию контейнеров, позволяя запускать множество контейнеров одновременно на одном хосте. Все зависимости фиксируются внутри образа и не конфликтуют с другими приложениями на сервере. В-третьих, Docker упрощает масштабирование и перенос в другие среды с гарантированным сохранением стабильной работы.

Для LSTM API разработан Dockerfile, код которого представлен ниже в листинге 1.

Листинг 1 – Dockerfile для LSTM API

```
# Базовый образ Python
FROM python:3.11-slim AS builder

WORKDIR /build

RUN apt-get update && apt-get install -y --no-install-recommends \
    gcc g++ git \
    && rm -rf /var/lib/apt/lists/*

COPY requirements.txt .
RUN pip install --no-cache-dir --prefix=/install -r \
    requirements.txt uvicorn

# Финальный образ
FROM python:3.11-slim

RUN addgroup --system appgroup && adduser --system --group \
    appuser

WORKDIR /app

COPY --from=builder /install /usr/local
```

```

# Клонирование кода из репозитория
RUN apt-get update && apt-get install -y --no-install-recommends
git \
    && git clone https://github.com/Acsious/crypto-lstm-
signals.git . \
    && apt-get remove -y git && apt-get autoremove -y \
    && rm -rf /var/lib/apt/lists/*

RUN chown -R appuser:appgroup /app
USER appuser

EXPOSE 8000

HEALTHCHECK --interval=30s --timeout=10s --start-period=5s --
retries=3 \
    CMD python -c "import requests;
requests.get('http://localhost:8000/health') || exit 1

ENV PYTHONUNBUFFERED=1 \
    PYTHONDONTWRITEBYTECODE=1

CMD ["gunicorn", "-w", "4", "-k",
"uvicorn.workers.UvicornWorker", "app.main:app", "--bind",
"0.0.0.0:8000"]

```

В начале указывается базовый образ Python, устанавливается рабочая директория, копируется файл с зависимостями из репозитория, после чего выполняются их установка и настройка Uvicorn-сервера для production-среды.

Установка зависимостей происходит через `pip install --no-cache-dir -r requirements.txt`, что позволяет избежать накопления кэша и уменьшает итоговый размер образа. Для минимизации размера образа также применяется очистка кэша пакетного менеджера apt и удаление временных файлов после установки.

Для Telegram-bot API также разработан Dockerfile, код которого представлен в листинге 2.

Листинг 2 – Dockerfile Telegram-bot API

```

FROM python:3.11-slim
WORKDIR /bot

# Копируем зависимости бота
COPY requirements.txt .
# Устанавливаем зависимости и сам интерпретатор

```

```

RUN apt-get update && apt-get install -y --no-install-recommends \
    && rm -rf /var/lib/apt/lists/* \
    && pip install --no-cache-dir -r requirements.txt

# Копируем весь код бота
COPY https://github.com/Acsious/crypto-lstm-signals
CMD ["python", "execution_module.py"]

```

Запуск контейнеров. Для удобства локальной разработки и тестирования, а также для объединения двух сервисов в одном окружении используется `docker-compose.yml`. В нём описываются оба компонента – Telegram Bot API и LSTM API – и определяется их взаимодействие через внутреннюю сеть Docker. Содержимое файла `docker-compose.yml` приведено ниже в листинге 3.

Листинг 3 – `docker-compose.yml`

```

services:
  lstm-api:
    build: ./lstm-api    # путь к папке с Dockerfile для API
    ports:
      - "8000:8000"
    restart: unless-stopped
    environment:
      - ENV=production

  telegram-bot:
    build: ./telegram-bot
    restart: unless-stopped
    environment:
      - BOT_TOKEN=${BOT_TOKEN}
      - API_URL=http://lstm-api:8000
    depends_on:
      - lstm-api

```

Использование Docker Compose гарантирует согласованное взаимодействие контейнеров по внутренней сети. Таким образом, предложенное решение минимизирует риски конфликтов библиотек, ускоряет процесс деплоя и соответствует современным практикам эксплуатации распределённых приложений.

Полный листинг клиентской части программного обеспечения представлен в приложении А.

4 Тестирование программного продукта

4.1 Тестирование серверной части приложения

4.1.1 Функциональное тестирование серверной части

Для начала необходимо убедиться в корректности работы серверной части приложения. С этой целью было произведено функциональное тестирование серверной части приложения с использованием инструментального средства Postman. Результаты тестирования функционала серверной части представлены в таблице 3. Скриншоты тестов находятся в приложении Б.

Таблица 3 – Результаты тестирования серверной части

Описание теста	Запрос		Результат
	Метод	Параметры	
Получение обычного прогноза	GET	нет	Тест пройден. Прогноз предоставлен. Код ответа HTTP – 200 (рисунок Б.1).
Оплата подписки	PUT	userId=39	Тест пройден. Подписка оплачена. Пользовательский статус успешно изменен на премиум . Код ответа HTTP – 201 (рисунок Б.2).
Получение премиум прогноза	GET	userId=39	Тест пройден. Прогноз предоставлен. Код ответа HTTP – 200 (рисунок Б.3).
Изменение горизонта прогнозирования	PUT	userId=39, newHor=30	Тест пройден. Горизонт планирования успешно изменен. Код ответа HTTP – 201 (рисунок Б.4).

Все тесты серверной части были пройдены успешно, что свидетельствует о корректности работы серверной части приложения и позволяет перейти к тестированию клиентской части разрабатываемого мобильного приложения.

4.1.2 Модульное тестирование серверной части

Модульные тесты – автоматизированные тесты, позволяющие оценить корректность работы некоторой автономной единицы программного кода путем сравнения фактического результата работы с ожидаемым. Основное достоинство модульных тестов состоит в том, что они позволяют быстро выявить и устранить ошибки, возникающие при модификации кода.

В рамках данного проекта было решено покрыть модульными тестами такой модуль системы как `strategy_module.py`, отвечающий за торговую стратегию.

Для проведения модульных тестов был разработан тестовый класс `TestStrategyModule`, код которого представлен в листинге 4.

Листинг 4 – Исходный код модульных тестов

```
import importlib
import unittest

from sqlalchemy import true
MODULE_NAME = "strategy_module"
def get_public_names(module):
    return [name for name in dir(module) if not
            name.startswith("_")]

class TestStrategyModule(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.module = importlib.import_module(MODULE_NAME)

    def test_module_import(self):
        try:
            from strategy_module import find_trade_points
            assert find_trade_points is not None
        except ImportError as e:
            self.fail(f"Не удалось импортировать strategy_module:
{e}")
        except Exception as e:
            self.fail(f"Ошибка при импорте strategy_module:
{type(e).__name__}: {e}")

    def test_profit_threshold():
        prices = [105, 100, 101, 102, 101]
        buy, sell, trades = MODULE_NAME.find_trade_points(prices,
threshold=0.015)
        assert len(trades) == 1
        assert trades[0]['profit'] >= 0.015
        assert trades[0]['buy_price'] == 100
        assert trades[0]['sell_price'] == 102

    def test_no_profitable_trades():
        prices = [100, 99, 100.5]
        buy, sell, trades = MODULE_NAME.find_trade_points(prices,
threshold=0.02)
        assert len(trades) == 0
    def test_empty_input(self):
        """Пустой массив → пустой результат"""
```

```

        buy,    sell,    trades    =    find_trade_points([],
threshold=0.01)
        self.assertEqual(buy, [])
        self.assertEqual(sell, [])
        self.assertEqual(trades, [])

    def test_short_input_less_than_3_elements(self):
        """Массив из 1-2 элементов → невозможно найти
сделку"""
        for prices in [[100], [100, 101]]:
            with self.subTest(prices=prices):
                buy,    sell,    trades    =
find_trade_points(prices, threshold=0.01)
                self.assertEqual(trades, [])

    def test_no_valid_trade_sequence(self):
        """Максимум перед минимумом – сделка невозможна"""
        prices = [100, 110, 90]    # сначала рост, потом
падение
        _,    _,    trades    =    find_trade_points(prices,
threshold=0.01)
        self.assertEqual(len(trades), 0)

```

Результат проведения модульных тестов серверной части приложения представлен на рисунке 16.

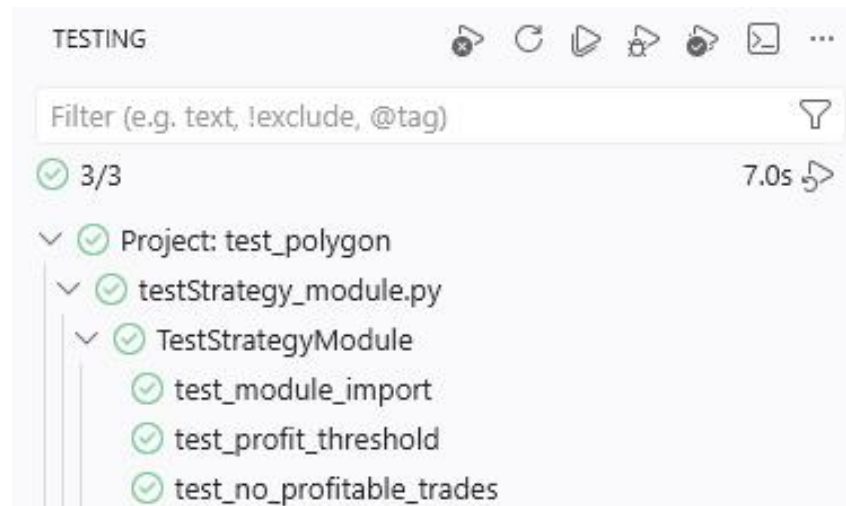


Рисунок 16 – Результаты модульного тестирования серверной части

Все тесты были успешно пройдены, что свидетельствует о корректной работе тестируемых методов.

4.2 Тестирование клиентской части приложения

4.2.1 Функциональное тестирование клиентской части

Для проверки корректности работы клиентской части приложения было проведено функциональное тестирование. Его результаты представлены в таблице 4. Скриншоты тестов находятся в приложении В.

Таблица 4 – Функциональное тестирование

Вариант использования	Тест и параметры	Результат
Получение обычного прогноза	Запросить обычный прогноз. Статус пользователя: обычный.	Тест пройден. Прогноз предоставлен (рисунок В.1).
	Запросить обычный прогноз. Статус пользователя: премиум.	Тест пройден. Прогноз предоставлен (рисунок В.2).
Оплата подписки	Оплатить подписку. Статус пользователя: обычный.	Тест пройден. Подписка успешно оплачена. Статус пользователя изменён на премиум (рисунок В.3).
	Оплатить подписку. Статус пользователя: премиум.	Тест пройден. Подписка успешно продлена. (рисунок В.4).
	Оплата прервана.	Тест пройден. Статус пользователя остался неизменным (рисунок В.5).
Получение премиум прогноза	Запросить премиум прогноз. Статус пользователя: обычный.	Тест пройден. Выведено сообщение об отсутствии подписки (рисунок В.6).
	Запросить премиум прогноз. Статус пользователя: премиум.	Тест пройден. Прогноз предоставлен (рисунок В.7).
Изменение горизонта прогнозирования	Изменить горизонт прогнозирования. Статус пользователя: обычный.	Тест пройден. Выведено сообщение об отсутствии подписки (рисунок В.8).
	Изменить горизонт прогнозирования. Статус пользователя: премиум.	Тест пройден. Горизонт прогнозирования успешно изменен (рисунок В.9).

Функциональное тестирование показало корректность работы клиентской части мобильного приложения.

4.2.2 Модульное тестирование клиентской части

В рамках данного проекта было решено покрыть модульными тестами такой модуль системы как `execution_module.py`, реализующий весь функционал Telegram-бота.

Для проведения модульных тестов был разработан тестовый класс `TestExecutionModule`, код которого представлен в листинге 5.

Листинг 5 – Исходный код модульных тестов

```
import importlib
import unittest

from sqlalchemy import true

MODULE_NAME = "execution_module"

def get_public_names(module):
    return [name for name in dir(module) if not
            name.startswith("_")]

class TestExecutionModule(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.module = importlib.import_module(MODULE_NAME)
    def test_module_imports(self):
        from execution_module import BotRouter
        self.assertTrue(callable(BotRouter))

    def test_routes_regular_forecast(self):
        callback_data = {"action": "regular_forecast", "user_id":
101}
        self.bot_router.route.return_value =
self.bot_router.handlers['regular_forecast']

        handler = self.bot_router.route(callback_data['action'])
        self.assertEqual(handler,
self.bot_router.handlers['regular_forecast'])

self.bot_router.route.assert_called_once_with('regular_forecast')

    def test_routes_premium_forecast(self):
        callback_data = {"action": "premium_forecast", "user_id":
101}
        self.bot_router.route.return_value =
self.bot_router.handlers['premium_forecast']

        handler = self.bot_router.route(callback_data['action'])
        self.assertEqual(handler,
self.bot_router.handlers['premium_forecast'])
```

```

def test_routes_set_horizon(self):
    callback_data = {"action": "set_horizon", "user_id": 101}
    self.bot_router.route.return_value =
self.bot_router.handlers['set_horizon']

    handler = self.bot_router.route(callback_data['action'])
    self.assertEqual(handler,
self.bot_router.handlers['set_horizon'])

```

Результат проведения модульных тестов клиентской части приложения представлен на рисунке 17.

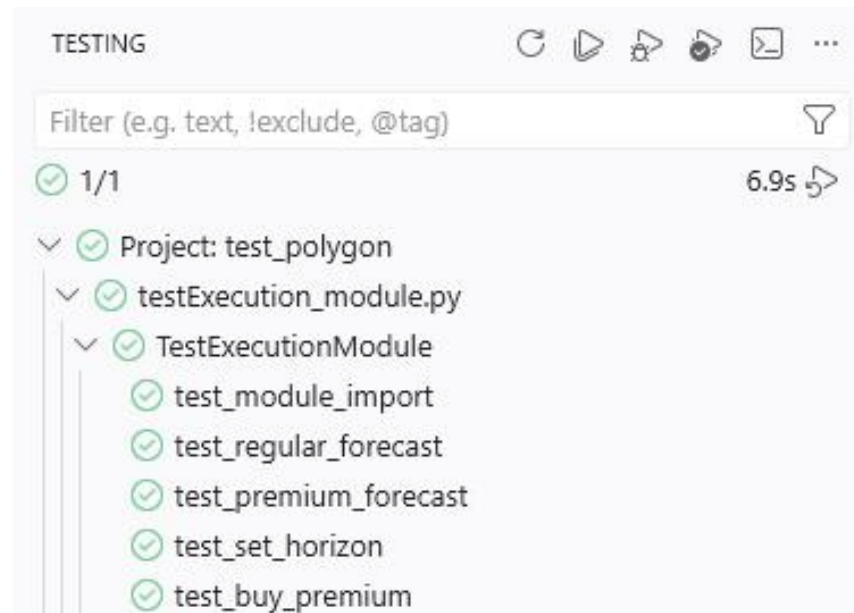


Рисунок 17 – Результаты модульного тестирования клиентской части

Все тесты были успешно пройдены, что свидетельствует о корректной работе тестируемых методов.

4.3 Метрики кода

Метриками кода называются некоторые количественные величины, позволяющие подсчитать, насколько код программы обладает тем или иным свойством. Существует огромное количество самых разнообразных метрик, имеющих различное назначение. Наиболее известными являются следующие группы метрик:

- метрики размера программы (SLOC-метрики),
- метрики сложности потока управления программ (метрики Холстеда и цикломатическая сложность).

Рассмотрим метод `find_trade_points` из модуля `strategy_module.py`, отвечающий за поиск потенциальных точек для входа и выхода из торговой позиции. Его код представлен в листинге 6.

Листинг 6 – Метод поиска торговых позиций

```
def find_trade_points(prices, threshold=0.01):
    # Инициализация списков для хранения локальных минимумов и
    максимумов
    minima = []
    maxima = []
    n = len(prices)

    # Поиск локальных экстремумов во временном ряде цен
    for i in range(1, n-1):
        if prices[i] < prices[i-1] and prices[i] < prices[i+1]:
            minima.append((i, prices[i]))
        if prices[i] > prices[i-1] and prices[i] > prices[i+1]:
            maxima.append((i, prices[i]))

    # Формирование списка сделок на основе найденных экстремумов
    trades = []
    min_idx = 0
    max_idx = 0

    # Итеративный поиск пар: минимум -> максимум с учетом
    хронологии
    while min_idx < len(minima) and max_idx < len(maxima):
        if minima[min_idx][0] < maxima[max_idx][0]:
            buy_idx, buy_price = minima[min_idx]
            # Пропускаем максимумы, которые идут раньше
            найденного минимума
            while max_idx < len(maxima) and maxima[max_idx][0] <
            buy_idx:
                max_idx += 1
            if max_idx < len(maxima):
                sell_idx, sell_price = maxima[max_idx]
                # Расчет потенциальной прибыли сделки
                profit = (sell_price - buy_price) / buy_price
                if profit >= threshold:
                    trades.append({'buy_idx': buy_idx,
                    'buy_price': buy_price, 'sell_idx': sell_idx, 'sell_price':
                    sell_price, 'profit': profit})
                    min_idx += 1
                    max_idx += 1
                else:
                    break
            else:
                max_idx += 1

    # Преобразование списка сделок в отдельные кортежи точек
    входа и выхода
```

```

buy_points = [(t['buy_idx'], t['buy_price']) for t in trades]
sell_points = [(t['sell_idx'], t['sell_price']) for t in trades]

```

```

return buy_points, sell_points, trades

```

Данный метод находится в клиентской части приложения и запускается при нажатии пользователем на кнопку «Отправить ответ». Он проверяет на корректность ответ пользователя, и сохраняет результат в базе данных.

Для данного метода будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

4.3.1 Метрики размера программы

Данные метрики представляют собой подсчет количества строк программного кода, а также числа строк с комментариями. Вычислим метрики размера программы для приведенного выше программного кода. Результаты представлены в таблице 5.

Таблица 5 – Метрики размера программы

Метрика	Количество
Количество строк кода	42
Строки-комментарии	7
Логические строки кода (SLOC)	35

Также вычислим процент комментариев. Он равен $(7/42) \cdot 100 = 16,67$.

Метрика «Количество строк кода» отражает физический объём реализации функции. В инженерии надёжного ПО рекомендуется придерживаться принципа единой ответственности: функция должна решать одну задачу. Согласно рекомендациям, оптимальный размер функции составляет 10–50 строк [17]. Значение 42 строки для метода `find_trade_points` укладывается в этот диапазон, что свидетельствует о фокусировке метода на одной задаче – поиске торговых сигналов.

Метрика «Строки с комментариями» показывает уровень документирования кода. Согласно стандартам критических систем, рекомендуемая доля комментариев составляет 15–25% от общего объёма кода [18]. Значение 16,7% указывает на сбалансированную документацию: код содержит достаточные пояснения для понимания логики новым

разработчиком, но не перегружен избыточными описаниями очевидных конструкций.

Метрика «Логические строки кода» (SLOC – Source Lines of Code) исключает пустые строки и комментарии, отражая реальную плотность алгоритма. Для атомарных функций бизнес-логики рекомендуемое значение – менее 50 строк. Значение 35 подтверждает, что метод не перегружен и легко поддаётся модульному тестированию.

Всё вышеперечисленное свидетельствует о достаточном количестве комментариев в коде для его хорошего понимания при отсутствии избыточных комментариев, переполняющих программу.

4.3.2 Метрики Холстеда

Для вычисления метрик Холстеда необходимо измерить четыре характеристики программы [19 – 21]:

- число уникальных операторов,
- число уникальных операндов,
- общее число операторов,
- общее число операндов.

Словарь программы характеризует общее количество уникальных лексем, используемых в коде. Эта метрика отражает разнообразие синтаксических конструкций и сущностей данных. Для функций бизнес-логики оптимальным считается диапазон 20–50.

Длина программы отражает суммарное количество всех лексем в коде. Эта метрика коррелирует с физическим объёмом реализации. Рекомендуемое значение для атомарной функции – менее 150 лексем.

Объём программы измеряет информационное содержание кода в битах. Метрика оценивает количество информации, которое должен обработать разработчик для понимания реализации. Согласно оригинальной работе Холстеда, значение объёма менее 1000 считается признаком функции, которую один разработчик может полностью удержать в рабочей памяти.

Сложность программы оценивает трудность понимания или модификации кода. Метрика учитывает как разнообразие операторов, так и частоту использования операндов. Пороговое значение для поддерживаемого кода – менее 50.

Интеллектуальные усилия представляют оценку умственных затрат на написание или понимание кода. Практический эмпирический порог: функция с $E < 25\,000$ может быть полностью понята одним разработчиком в рамках одного рабочего сеанса.

С целью вычисления этих характеристик был составлен словарь операторов (таблица 6).

Таблица 6 – Словарь операторов

Операторы	Число вхождений
=	8
[]	7
()	6
<	5
.	5
:	5
and	4
if	4
len	4
+=	3
while	2
append	2
>	2
for	1
>=	1
return	1
n1 = 16	N1 = 60

После составления словаря операторов, необходимо также составить словарь операндов.

С целью вычисления вышеописанных характеристик был составлен словарь операндов (таблица 7)

Таблица 7 – Словарь операндов

Операнды	Число вхождений
prices	6
maxima	6
minima	5
Max_idx	5
i	5
Min_idx	4
Buy_price	3
Buy_idx	3
Seel_price	3
Sell_idx	3
trades	3
profit	2
threshold	2
n2 = 14	N2 = 56

Таким образом, число уникальных операторов программы $n1 = 14$, число уникальных операндов программы $n2 = 14$, общее число операторов в программе $N1 = 76$, общее число операндов в программе $N2 = 47$.

На основе метрик Холстеда можно вычислить целый ряд характеристик программы. Рассмотрим некоторые из них.

Расчёт словаря программы производится по формуле:

$$n = n1 + n2 = 16 + 14 = 30,$$

где $n1$ – количество операторов в программе;

$n2$ – количество операндов программе.

Расчёт длины программы производится по формуле:

$$N = N1 + N2 = 60 + 56 = 116,$$

где $N1$ – число вхождений всех операторов в программе;

$N2$ – число вхождений всех операндов программе.

Расчёт объёма программы производится по формуле:

$$V = N * \log_2 n = 116 * \log_2 30 = 569,$$

где N – длина программы;

n – словарь программы.

Расчёт сложности программы производится по формуле:

$$D = (n1/2) * (N2/n2) = (16 / 2) * (56 / 14) = 32,$$

где $n1$ – количество операторов в программе;

$n2$ – количество операндов программе;

$N2$ – число вхождений всех операндов программе.

Расчёт оценки интеллектуальных усилий производится по формуле:

$$E = D * V = 32 * 569 = 18208,$$

где D – сложность программы;

V – объем программы.

Полученные характеристики подтверждают высокое качество реализации метода `find_trade_points`:

Низкий объём ($V = 569$) указывает на минимально необходимое количество информации для решения задачи, без избыточных абстракций.

Умеренная сложность ($D = 32$) отражает интуитивную структуру алгоритма, упрощающую тестирование и отладку.

Приемлемые интеллектуальные усилия ($E \approx 18K$) гарантируют, что метод может быть понят и модифицирован в рамках одного сеанса работы.

Сбалансированное соотношение операторов и операндов, доминирование простых арифметических и логических конструкций свидетельствуют о хорошей читаемости кода.

Исходя из всех рассчитанных метрик можно сделать заключение о том, что метод соответствует лучшим практикам инженерии надёжного ПО – он атомарен, документирован, имеет предсказуемую сложность и легко покрывается юнит-тестами. Рекомендуется сохранить текущую структуру при дальнейшей эволюции стратегии.

ЗАКЛЮЧЕНИЕ

В рамках выполнения магистерской диссертации на тему «Программное обеспечение анализа и обработки данных» была успешно достигнута поставленная цель. Разработано комплексное программное обеспечение, предназначенное для анализа и прогнозирования цены криптовалюты на основе рекуррентной нейронной сети архитектуры LSTM с последующей интеграцией результатов работы модели в удобный и функциональный Telegram-бот. Созданная система обеспечивает обработку больших объёмов временных рядов, генерацию прогнозов различной глубины и предоставление персонализированных торговых сигналов пользователям с разным уровнем доступа.

В процессе исследования был проведён всесторонний анализ предметной области, связанной с прогнозированием финансовых временных рядов, особенно в условиях высокой волатильности криптовалютного рынка. Изучены существующие подходы к решению подобных задач, включая статистические методы, классические машинные алгоритмы и современные нейронные архитектуры. В результате проведённого сравнительного анализа обоснован выбор LSTM-модели как наиболее подходящего инструмента, способного эффективно учитывать долгосрочные зависимости в данных, справляться с проблемой исчезающего градиента и моделировать сложные нелинейные взаимосвязи, характерные для динамики цены криптовалюта. Особое внимание было уделено реализации multi-output прогнозирования, позволяющего получать не только точечные прогнозы, но и целые ценовые траектории на расширенном горизонте.

На основе результатов анализа были сформулированы подробные функциональные и нефункциональные требования к разрабатываемой системе. Функциональные требования учитывали необходимость предоставления ежедневных торговых сигналов (buy/sell/hold) для обычных пользователей и расширенного прогноза на горизонте до трёх месяцев с указанием потенциальных точек входа и выхода из позиций для премиум-

пользователей. Нефункциональные требования включали в себя высокую точность и стабильность прогнозов, масштабируемость системы, безопасность хранения и передачи данных, а также удобство использования клиентской части.

Важным этапом работы стало проектирование модульной архитектуры программного обеспечения. Серверная часть системы была спроектирована с учётом принципов контейнеризации и включает в себя специализированные модули, отвечающие за получение и предобработку рыночных данных, периодическое обучение и обновление LSTM-модели, генерацию прогнозов, формирование торговых рекомендаций с элементами риск-менеджмента, управление пользователями и подписками, а также взаимодействие с внешними сервисами. Клиентская часть реализована в виде Telegram-бота, обеспечивающего удобный и оперативный доступ к результатам работы системы. Особое внимание было уделено вопросам развертывания – использована Docker-контейнеризация для изоляции компонентов, что повышает отказоустойчивость и упрощает дальнейшее масштабирование решения.

Результаты диссертационной работы обладают существенной научной и практической значимостью. С научной точки зрения работа подтверждает эффективность применения современных методов глубокого обучения для решения задач анализа сложных финансовых временных рядов. Полученные результаты дополняют существующие исследования в области прогнозирования криптовалют и демонстрируют перспективность использования многошагового прогнозирования на основе LSTM-архитектуры. С практической стороны разработанное программное обеспечение представляет собой готовый инструмент, который может быть использован розничными инвесторами и трейдерами для поддержки принятия обоснованных инвестиционных решений в условиях нестабильного рынка.

Разработанная система соответствует актуальным стратегическим приоритетам цифровой трансформации экономики Российской Федерации, обозначенным в национальных программах и стратегиях развития. Внедрение технологий искусственного интеллекта и анализа больших данных в финансовую сферу способствует повышению эффективности инвестиционной деятельности, развитию fintech-решений и укреплению конкурентоспособности отечественных разработок на международном уровне.

В дальнейшем планируется продолжить работу над системой. Предстоит провести полномасштабное интеграционное и нагрузочное тестирование в реальных условиях эксплуатации, расширить охват прогнозирования на другие популярные криптовалюты, существенно улучшить качество модели за счёт включения дополнительных групп признаков, таких как on-chain метрики, данные рыночного сентимента и макроэкономические индикаторы. Также значительное внимание будет уделено разработке и внедрению полноценного модуля backtesting торговых стратегий, что позволит провести детальную оценку экономической эффективности предлагаемых рекомендаций с использованием широкого спектра финансовых метрик.

Таким образом, в ходе выполнения магистерской диссертации были успешно достигнуты все поставленные цели и решены задачи. Разработанное программное обеспечение анализа и обработки данных на основе LSTM-модели с интеграцией в Telegram-бот представляет собой полноценный, конкурентоспособный и перспективный продукт, обладающий значительным потенциалом для практического применения в сфере криптовалютных инвестиций и дальнейшего научного развития.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Указ Президента Российской Федерации от 9 мая 2017 г. № 203 «О Стратегии развития информационного общества в Российской Федерации на 2017–2030 годы». [Электронный ресурс]. URL: <http://kremlin.ru/acts/bank/41919> (дата обращения: 01.05.2026).
2. Национальный проект «Экономика данных и цифровая трансформация государства». Паспорт федерального проекта «Инфраструктура доступа к информационно-телекоммуникационной сети “Интернет”» (утв. Министерством цифрового развития, связи и массовых коммуникаций Российской Федерации, 2025 г.). [Электронный ресурс]. URL: <https://base.garant.ru/412776143/> (дата обращения: 01.05.2026).
3. Федеральный закон от 27 июля 2006 г. № 152-ФЗ «О персональных данных». [Электронный ресурс]. URL: <https://base.garant.ru/12148567/> (дата обращения: 01.05.2026).
4. Ark Invest. Big Ideas 2026: Bitcoin Market Cap Forecast. [Электронный ресурс]. URL: <https://crypto.news/ark-invest-sees-криптовалюта-hitting-16t-and-crypto-28t-by-2030/> (дата обращения: 01.05.2026).
5. Банк России. Пресс-релиз от 11 декабря 2024 г. «Установлены факты манипулирования рынком при совершении биржевых сделок с отдельными ценными бумагами». [Электронный ресурс]. URL: http://bulldog.bankodrom.ru/eng/press/pr/?file=638696996906528008ENG_PNP.htm (дата обращения: 01.05.2026).
6. Nazareth, N., Ramana Reddy, Y. V. Financial applications of machine learning: A literature review // Expert Systems with Applications. 2023. Vol. 119640. DOI: 10.1016/j.eswa.2023.119640.
7. ГОСТ Р 71476-2024 (ИСО/МЭК 22989:2022) «Искусственный интеллект. Концепции и терминология искусственного интеллекта» (утв. и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 28 октября 2024 г. № 1550-ст).

[Электронный ресурс]. URL: <https://base.garant.ru/411037370/> (дата обращения: 11.05.2026).

8. Bybit. ИИ Аврора от Bybit: новый подход к криптотрейдингу с ИИ-ботами. [Электронный ресурс]. URL: <https://www.bybitx.global/ru-RU/promo/events/AuroraAITradingBot/> (дата обращения: 11.05.2026).

9. 3Commas. Торговые боты и платформа автоматизации на крипторынке. [Электронный ресурс]. URL: <https://3commas.io/ru> (дата обращения: 12.05.2026).

10. GNY.io. GNY aiBOT – AI-powered cryptocurrency trading bot for Telegram. [Электронный ресурс]. URL: <https://gny.io/> (дата обращения: 12.05.2026).

11. Kaur R., Uppal M., Gupta D., Juneja S., Arafat S.Y., Rashid J., Kim J., Alrooba R. Development of a cryptocurrency price prediction model: leveraging GRU and LSTM for криптовалюта, Litecoin and Ethereum // PeerJ Computer Science. 2025. Vol. 11. P. e2675. DOI: 10.7717/peerj-cs.2675.

12. Li Z., Mao S. Implementations of Deep Learning in Cryptocurrency Based on Comparisons of Various Scenarios // Proceedings of the 2025 2nd International Conference on Mechanics, Electronics Engineering and Automation (ICMEEA 2025). Atlantis Press, 2025. P. 721–729. DOI: 10.2991/978-94-6463-821-9_77.

13. Mubarak A.I. Enhancing cryptocurrency price forecasting: Performance evaluation of baseline versus Bayesian-optimized LSTM models // Investment Management and Financial Innovations. 2026. Vol. 23, No. 2. P. 52–66. DOI: 10.21511/imfi.23(2).2026.05.

14. Čatloch D., Chovancová E. A Comparative Study of Machine Learning Models on Cryptocurrency Prices // Acta Electrotechnica et Informatica. 2026. Vol. 26, No. 1. P. 27–35. DOI: 10.2478/aei-2026-0004.

15. Optimized Stacked-LSTM Model for Cryptocurrency Price Prediction // Scientific Reports. 2025. Vol. 15, No. 8119. DOI: 10.1038/s41598-025-92563-y.

16. Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Computation. 1997. Vol. 9, No. 8. P. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.
17. Rational Unified Process (RUP). Унифицированный процесс разработки программного обеспечения. [Электронный ресурс]. URL: https://e-learning.bmstu.ru/iu5/pluginfile.php/6538/mod_folder/content/0/lec_03-04-RUP.pdf (дата обращения: 11.05.2026).
18. TensorFlow. An end-to-end open source machine learning platform. [Электронный ресурс]. URL: <https://www.tensorflow.org/> (дата обращения: 16.05.2026).
19. Keras: Deep Learning for humans. [Электронный ресурс]. URL: <https://keras.io/> (дата обращения: 11.05.2026).
20. Python Telegram Bot. A pure Python asynchronous interface for the Telegram Bot API. [Электронный ресурс]. URL: <https://github.com/python-telegram-bot/python-telegram-bot> (дата обращения: 11.05.2026).
21. Docker. Empowering App Development for Developers. [Электронный ресурс]. URL: <https://www.docker.com/> (дата обращения: 16.05.2026).
22. Optuna: A hyperparameter optimization framework. [Электронный ресурс]. URL: <https://optuna.org/> (дата обращения: 01.05.2026).
23. Pandas: Python Data Analysis Library. [Электронный ресурс]. URL: <https://pandas.pydata.org/> (дата обращения: 07.05.2026).
24. NumPy. [Электронный ресурс]. URL: <https://numpy.org/> (дата обращения: 12.05.2026).
25. Scikit-learn: Machine Learning in Python. [Электронный ресурс]. URL: <https://scikit-learn.org/> (дата обращения: 11.05.2026).

ПРИЛОЖЕНИЕ А

(обязательное)

Программное обеспечение анализа и обработки данных

Листинг программы

Листинг А.1 – Файл data_module.py

```
import pandas as pd
import logging
import sys

START_DATE = '2013-12-28'

logging.basicConfig(level=logging.INFO, format='%(asctime)s -
%(levelname)s - %(message)s')

def load_and_preprocess_data(file_path: str = 'data.csv') ->
pd.DataFrame:
    try:
        df = pd.read_csv(file_path)
        required_columns = ['snapped_at', 'price', 'market_cap',
'total_volume']
        if not all(col in df.columns for col in
required_columns):
            raise ValueError("Отсутствуют обязательные колонки:
{required_columns}")
        df['snapped_at'] = pd.to_datetime(df['snapped_at'],
utc=True)

        if df.isnull().any().any():
            logging.warning("Обнаружены пропущенные значения.
Удаление строк с NaN.")
            df = df.dropna()

        df = df.sort_values('snapped_at')
        if df['snapped_at'].duplicated().any():
            logging.warning("Обнаружены дубликаты по дате.
Удаление дубликатов.")
            df = df.drop_duplicates(subset='snapped_at',
keep='last')

        start_date =
pd.to_datetime(START_DATE).tz_localize('UTC')
        df = df[df['snapped_at'] >= start_date]
        df = df.reset_index(drop=True)
        logging.info(f"Данные успешно загружены и обработаны.
Форма DataFrame: {df.shape}")
        logging.info(f"Диапазон дат: с {df['snapped_at'].min()}
по {df['snapped_at'].max()}")
        return df

    except FileNotFoundError:
        logging.error(f"Файл не найден: {file_path}")
        sys.exit(1)
    except Exception as e:
        logging.error(f"Ошибка при загрузке или обработке данных:
{str(e)}")
        sys.exit(1)
```

```

if __name__ == "__main__":
    df = load_and_preprocess_data('data.csv')
    print(df.head())
    print(df.tail())
    print(df.info())

```

Листинг А.2 – Файл model_making_module.py

```

import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import EarlyStopping,
ModelCheckpoint
import joblib

SEQ_LENGTH = 120
FORECAST_HORIZON = 30
MODEL_PATH = 'lstm_price_multioutput.h5'
SCALER_PATH = 'scaler_price_multioutput.save'

def load_price_data(csv_path):
    df = pd.read_csv(csv_path)
    if 'price' not in df.columns:
        raise ValueError('В файле нет колонки price!')
    return df['price'].values.reshape(-1, 1)

def create_sequences(data, seq_length, horizon):
    X, y = [], []
    for i in range(len(data) - seq_length - horizon + 1):
        X.append(data[i:i+seq_length])

    y.append(data[i+seq_length:i+seq_length+horizon].flatten())
    return np.array(X), np.array(y)

def train_lstm_multioutput(csv_path='data.csv',
                           seq_length=SEQ_LENGTH,
                           horizon=FORECAST_HORIZON,
                           model_path=MODEL_PATH,
                           scaler_path=SCALER_PATH):
    data = load_price_data(csv_path)
    scaler = MinMaxScaler()
    data_scaled = scaler.fit_transform(data)
    joblib.dump(scaler, scaler_path)
    X, y = create_sequences(data_scaled, seq_length, horizon)
    n = len(X)
    n_train = int(n * 0.8)
    n_val = int(n * 0.1)
    X_train, y_train = X[:n_train], y[:n_train]
    X_val, y_val = X[n_train:n_train+n_val],
y[n_train:n_train+n_val]

```

```

X_test, y_test = X[n_train+n_val:], y[n_train+n_val:]
model = Sequential([
    LSTM(units, return_sequences=True,
input_shape=(sequence_length, n_features)),
    Dropout(dropout_rate),
    LSTM(units, return_sequences=False),
    Dropout(dropout_rate),
    Dense(units // 3),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')
callbacks = [
    EarlyStopping(patience=10, restore_best_weights=True),
    ModelCheckpoint(model_path, save_best_only=True)
]
model.fit(X_train, y_train, validation_data=(X_val, y_val),
        epochs=100, batch_size=32, callbacks=callbacks)
val_loss = model.evaluate(X_val, y_val)
test_loss = model.evaluate(X_test, y_test)
print(f'Validation    loss:    {val_loss:.6f},    Test    loss:
{test_loss:.6f}')
print(f'Модель    и    scaler    сохранены:    {model_path},
{scaler_path}')
```

```

if __name__ == '__main__':
    train_lstm_multioutput()
```

Листинг A.3 – Файл execution_module.py

```

import logging
from datetime import datetime
from typing import List, Tuple, Dict, Any

from telegram import Update, InlineKeyboardButton,
InlineKeyboardMarkup
from telegram.ext import (
    Application,
    CommandHandler,
    CallbackQueryHandler,
    MessageHandler,
    filters,
    ConversationHandler,
    ContextTypes,
)
import aiohttp
import json
from strategy_module import find_trade_points
TELEGRAM_BOT_TOKEN = "BOT_TOKEN"
API_BASE_URL = "http://localhost:8000"
WAITING_FOR_HORIZON = 1
logging.basicConfig(
    level=logging.INFO,
```

```

        format="% (asctime)s      -      %(name)s      -      %(levelname)s      -
%(message)s",
    )
    logger = logging.getLogger(__name__)

    async def fetch_forecast(user_id: int, horizon: int) -> Dict[str,
Any]:
        async with aiohttp.ClientSession() as session:
            try:
                params = {"user_id": user_id, "horizon": horizon}
                async with session.get(f"{API_BASE_URL}/predict",
params=params) as resp:
                    if resp.status == 200:
                        data = await resp.json()
                        return {"predictions":
data.get("predictions", [])}
                    elif resp.status == 403:
                        return {"error": "Подписка не активна.
Оплатите доступ."}
                    else:
                        text = await resp.text()
                        return {"error": f"Ошибка сервера
{resp.status}: {text}"}
            except Exception as e:
                return {"error": f"Не удалось соединиться с сервером:
{str(e)}"}

    async def set_user_horizon(user_id: int, horizon: int) -> bool:
        """Отправляет новый горизонт прогнозирования на сервер."""
        async with aiohttp.ClientSession() as session:
            try:
                payload = {"user_id": user_id, "horizon": horizon}
                async with
session.post(f"{API_BASE_URL}/user/horizon", json=payload) as
resp:
                    return resp.status == 200
            except Exception:
                return False

    async def check_subscription_status(user_id: int) -> bool:
        async with aiohttp.ClientSession() as session:
            try:
                async with
session.get(f"{API_BASE_URL}/user/{user_id}/status") as resp:
                    if resp.status == 200:
                        data = await resp.json()
                        return data.get("is_premium", False)
                    return False
            except Exception:
                return False

```

```

async def start(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> None:
    keyboard = [
        [InlineKeyboardButton("Обычный прогноз",
callback_data="basic")],
        [InlineKeyboardButton(Премииум прогноз",
callback_data="premium")],
        [InlineKeyboardButton("Оплатить подписку",
callback_data="subscribe")],
        [InlineKeyboardButton("Изменить горизонт",
callback_data="change_horizon")],
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await update.message.reply_text(
        " *Бот прогнозирования цены криптовалюта*\n\n"
        "Выберите действие:",
        reply_markup=reply_markup,
        parse_mode="Markdown",
    )

```

```

async def button_callback(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> None:
    query = update.callback_query
    await query.answer()
    user_id = query.from_user.id
    data = query.data

    if data == "basic":
        await query.edit_message_text(" Запрашиваю обычный
прогноз")
        result = await fetch_forecast(user_id, horizon=5)
        if "error" in result:
            await query.edit_message_text(f" {result['error']}")
            return

        predictions = result["predictions"]
        if len(predictions) < 5:
            await query.edit_message_text(" Недостаточно данных
для формирования сигнала.")
            return
        buy_points, sell_points, trades =
find_trade_points(predictions[:5], threshold=0.01)
        if trades:
            first_trade = trades[0]
            if first_trade['buy_idx'] < first_trade['sell_idx']:
                signal = "BUY "
            else:
                signal = "SELL "
        else:
            signal = "HOLD "

```

```

        current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S UTC")
        message = (
            f" *Новый торговый сигнал (криптовалюта)*\n"
            f" Дата и время: {current_time}\n"
            f" Рекомендация: *{signal}*\n\n"
            f"Система торговли на базе LSTM-модели."
        )
        await query.edit_message_text(message,
            parse_mode="Markdown")

        elif data == "premium":
            await query.edit_message_text(" Запрашиваю премиум-прогноз...")
            /predict/premium
            async with aiohttp.ClientSession() as session:
                try:
                    async with session.get(f"{API_BASE_URL}/predict/premium", params={"user_id": user_id}) as resp:
                        if resp.status == 200:
                            data = await resp.json()
                            predictions = data.get("predictions", [])
                        elif resp.status == 403:
                            await query.edit_message_text(" Подписка не активна. Оплатите доступ.")
                            return
                        else:
                            await query.edit_message_text(f"Ошибка сервера: {resp.status}")
                            return
                    except Exception as e:
                        await query.edit_message_text(f" Ошибка соединения: {str(e)}")
                        return

            if not predictions:
                await query.edit_message_text(" Прогноз не получен.")
                return

            buy_points, sell_points, trades = find_trade_points(predictions, threshold=0.01)
            forecast_lines = [f"День {i+1}: ${p:,.2f}" for i, p in enumerate(predictions)]
            forecast_str = "\n".join(forecast_lines)

            if trades:
                trades_str = "\n".join([
                    f"Сделка {i+1}: Покупка (день {t['buy_idx']+1}, цена ${t['buy_price']:,.2f}) → "
                    f"Продажа (день {t['sell_idx']+1}, цена ${t['sell_price']:,.2f}), Профит: {t['profit']*100:.2f}%"
                    for i, t in enumerate(trades)
                ])

```

```

else:
    trades_str = "Торговых возможностей не найдено."

    current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S UTC")
    message = (
        f"*Премиум-прогноз (криптовалюта)*\n"
        f"Дата и время: {current_time}\n"
        f"Прогноз на период (горизонт: {len(predictions)} дней):\n"
        f"{forecast_str}\n\n"
        f"*Найдено сделок:* {len(trades)}\n"
        f"{trades_str}\n\n"
        f"Система торговли на базе LSTM multioutput-модели."
    )
    await query.edit_message_text(message,
    parse_mode="Markdown")

    elif data == "subscribe":
        payment_link = await get_payment_link(user_id)
        await query.edit_message_text(
            f"*Оплата подписки*\n\n"
            f"Для доступа к премиум-функциям перейдите по ссылке:\n{payment_link}\n\n"
            f"После оплаты статус обновится автоматически.",
            parse_mode="Markdown",
        )

    elif data == "change_horizon":
        is_premium = await check_subscription_status(user_id)
        if not is_premium:
            await query.edit_message_text(
                "Изменение горизонта доступно только премиум-пользователям. Оплатите подписку."
            )
            return
        await query.edit_message_text(
            f"*Настройка горизонта прогнозирования*\n\n"
            f"Введите число от 1 до 30 (количество дней для премиум-прогноза):",
            parse_mode="Markdown",
        )
        return ConversationHandler.ENTER_STATE

async def ask_horizon(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int:
    user_id = update.effective_user.id
    text = update.message.text.strip()
    try:
        horizon = int(text)
        if not (1 <= horizon <= 30):
            raise ValueError
    except ValueError:

```

```

        await update.message.reply_text(
            " Ошибка: введите целое число от 1 до
30.\nПопробуйте снова:"
        )
        return WAITING_FOR_HORIZON
    success = await set_user_horizon(user_id, horizon)
    if success:
        await update.message.reply_text(
            f" Горизонт прогнозирования установлен на {horizon}
дней.\n"
            f"Теперь премиум-прогноз будет рассчитываться на
этот период."
        )
        keyboard = [
            [InlineKeyboardButton(" Обычный прогноз",
callback_data="basic")],
            [InlineKeyboardButton(" Премиум прогноз",
callback_data="premium")],
            [InlineKeyboardButton(" Оплатить подписку",
callback_data="subscribe")],
            [InlineKeyboardButton(" Изменить горизонт",
callback_data="change_horizon")],
        ]
        reply_markup = InlineKeyboardMarkup(keyboard)
        await update.message.reply_text("Выберите действие:",
reply_markup=reply_markup)
    else:
        await update.message.reply_text(
            " Не удалось сохранить настройки. Проверьте статус
подписки и попробуйте позже."
        )
    return ConversationHandler.END

async def cancel(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
    await update.message.reply_text("Отменено.")
    return ConversationHandler.END

def main():
    app = Application.builder().token(TELEGRAM_BOT_TOKEN).build()
    app.add_handler(CommandHandler("start", start))
    app.add_handler(CallbackQueryHandler(button_callback))
    conv_handler = ConversationHandler(
        entry_points=[CallbackQueryHandler(button_callback,
pattern="^change_horizon$")],
        states={
            WAITING_FOR_HORIZON: [MessageHandler(filters.TEXT &
~filters.COMMAND, ask_horizon)],
        },
        fallbacks=[CommandHandler("cancel", cancel)],
    )
    app.add_handler(conv_handler)
    logger.info("Бот запущен")

```

```

        app.run_polling(allowed_updates=Update.ALL_TYPES)
if __name__ == "__main__":
    main()

```

Листинг А.4 – Файл lstm_module.py

```

import numpy as np
import pandas as pd
from tensorflow.keras.models import load_model
import joblib
import logging
import os

SEQ_LENGTH = 120
FORECAST_HORIZON = 30
MODEL_PATH = 'lstm_price_multioutput.h5'
SCALER_PATH = 'scaler_price_multioutput.save'

logging.basicConfig(level=logging.INFO, format='%(asctime)s -
%(levelname)s - %(message)s')

def load_lstm_model(model_path: str = MODEL_PATH):
    if not os.path.exists(model_path):
        logging.error(f"Файл модели не найден: {model_path}")
        raise FileNotFoundError(f"Модель не найдена по пути:
{model_path}")
    model = load_model(model_path)
    logging.info(f"Модель успешно загружена из {model_path}")
    return model

def load_scaler(scaler_path: str = SCALER_PATH):
    if not os.path.exists(scaler_path):
        logging.error(f"Файл scaler не найден: {scaler_path}")
        raise FileNotFoundError(f"Scaler не найден по пути:
{scaler_path}")
    scaler = joblib.load(scaler_path)
    logging.info(f"Scaler успешно загружен из {scaler_path}")
    return scaler

def prepare_sequence(df: pd.DataFrame, sequence_length: int =
SEQ_LENGTH):
    if 'price' not in df.columns:
        raise ValueError('В DataFrame нет колонки price!')
    data = df['price'].values.reshape(-1, 1)
    scaler = load_scaler()
    scaled_data = scaler.transform(data)
    if len(scaled_data) < sequence_length:
        raise ValueError(f"Недостаточно данных для
последовательности длиной {sequence_length}")
    last_sequence = scaled_data[-sequence_length:]
    last_sequence = last_sequence.reshape((1, sequence_length,
1))
    return last_sequence, scaler

```

```

def generate_multioutput_predictions(df: pd.DataFrame,
model=None, horizon: int = FORECAST_HORIZON) -> np.ndarray:
    if model is None:
        model = load_lstm_model()
    last_sequence, scaler = prepare_sequence(df)
    pred_scaled = model.predict(last_sequence, verbose=0)
    pred_scaled = pred_scaled.reshape(-1, 1)
    pred_real = scaler.inverse_transform(pred_scaled)[: , 0]
    logging.info(f"Сгенерировано {horizon} предсказаний цен
(multi-output)")
    return pred_real

if __name__ == "__main__":
    from data_module import load_and_preprocess_data
    df = load_and_preprocess_data('data.csv')
    try:
        preds = generate_multioutput_predictions(df)
        print(f"Предсказанные цены на следующие
{FORECAST_HORIZON} периода:")
        print(preds)
    except Exception as e:
        print(f"Ошибка: {e}")

```

Листинг А.5 – Файл strategy_module.py

```

import matplotlib.pyplot as plt
from lstm_module import generate_multioutput_predictions

def find_trade_points(prices, threshold=0.01):
    minima = []
    maxima = []
    n = len(prices)
    for i in range(1, n-1):
        if prices[i] < prices[i-1] and prices[i] < prices[i+1]:
            minima.append((i, prices[i]))
        if prices[i] > prices[i-1] and prices[i] > prices[i+1]:
            maxima.append((i, prices[i]))
    trades = []
    min_idx = 0
    max_idx = 0
    while min_idx < len(minima) and max_idx < len(maxima):
        if minima[min_idx][0] < maxima[max_idx][0]:
            buy_idx, buy_price = minima[min_idx]
            while max_idx < len(maxima) and maxima[max_idx][0] <
buy_idx:
                max_idx += 1
            if max_idx < len(maxima):
                sell_idx, sell_price = maxima[max_idx]
                profit = (sell_price - buy_price) / buy_price
                if profit >= threshold:
                    trades.append({'buy_idx': buy_idx,
'buy_price': buy_price,

```

```

        'sell_idx': sell_idx,
'sell_price': sell_price,
        'profit': profit})
        min_idx += 1
        max_idx += 1
    else:
        break
    else:
        max_idx += 1
    buy_points = [(t['buy_idx'], t['buy_price']) for t in trades]
    sell_points = [(t['sell_idx'], t['sell_price']) for t in
trades]
    return buy_points, sell_points, trades

def plot_trades(prices, buy_points, sell_points):
    plt.figure(figsize=(12, 6))
    plt.plot(prices, label='Predicted Price', color='blue')
    if buy_points:
        buy_idx, buy_price = zip(*buy_points)
        plt.scatter(buy_idx, buy_price, color='green',
label='Buy (Entry)', marker='^', s=100)
    if sell_points:
        sell_idx, sell_price = zip(*sell_points)
        plt.scatter(sell_idx, sell_price, color='red',
label='Sell (Exit)', marker='v', s=100)
    plt.title('Trade Points on Predicted Prices (MultiOutput)')
    plt.xlabel('Index')
    plt.ylabel('Price')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    from data_module import load_and_preprocess_data
    df = load_and_preprocess_data('data.csv')
    preds = generate_multioutput_predictions(df)
    print("Предсказанные цены:")
    print(preds)

    threshold = 0.0011
    buy_points, sell_points, trades = find_trade_points(preds,
threshold=threshold)

    print(f"\nНайдено {len(trades)} сделок (порог
{threshold*100:.2f}%):")
    for i, t in enumerate(trades, 1):
        print(f"Сделка {i}: Покупка (индекс {t['buy_idx']}, цена
{t['buy_price']:.2f}) -> Продажа (индекс {t['sell_idx']}, цена
{t['sell_price']:.2f}), Профит: {t['profit']*100:.2f}%")

    plot_trades(preds, buy_points, sell_points)

```

Листинг А.6 – Файл `monitoring_module.py`

```
import logging
import pandas as pd
from datetime import datetime
import os

LOG_FILE_CSV = 'signals_log.csv'

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler('system_monitoring.log',
encoding='utf-8'),
        logging.StreamHandler()
    ]
)

def initialize_signals_log():
    if not os.path.exists(LOG_FILE_CSV):
        columns = [
            'timestamp_utc', 'market_phase', 'phase_description',
            'strategy_name',
            'strategy_params', 'signal', 'notes'
        ]
        pd.DataFrame(columns=columns).to_csv(LOG_FILE_CSV,
index=False)
        logging.info(f"Создан новый файл лога сигналов:
{LOG_FILE_CSV}")

def log_signal(signal: str, phase: int, strategy_name: str,
params: dict, notes: str = '') -> None:
    phase_descriptions = {0: 'Bear', 1: 'Bull', 2: 'Sideways'}
    phase_desc = phase_descriptions.get(phase, 'Unknown')
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S UTC')
    log_message = (
        f"Сигнал сгенерирован | "
        f"Фаза: {phase_desc} | "
        f"Стратегия: {strategy_name} | "
        f"Параметры: {params} | "
        f"Сигнал: {signal.upper()} | "
        f"Заметки: {notes}"
    )
    logging.info(log_message)
    new_record = {
        'timestamp_utc': timestamp,
        'market_phase': phase,
        'phase_description': phase_desc,
        'strategy_name': strategy_name,
        'strategy_params': str(params),
        'signal': signal,
        'notes': notes
    }
```

```

    }
    try:
        df_new = pd.DataFrame([new_record])
        df_new.to_csv(LOG_FILE_CSV, mode='a', header=False,
index=False)
    except Exception as e:
        logging.error(f"Ошибка при записи в CSV-лог сигналов:
{str(e)}")

def log_system_event(event_type: str, message: str) -> None:
    full_message = f"Системное событие [{event_type}]:
{message}"
    if event_type == 'ERROR':
        logging.error(full_message)
    else:
        logging.info(full_message)

def calculate_simple_performance_metrics(df_historical:
pd.DataFrame, last_signal: str) -> dict:
    recent = df_historical['price'].tail(30)
    if len(recent) > 1:
        cumulative_return = (recent.iloc[-1] / recent.iloc[0]) -
1
        return {'last_30d_return': f'{cumulative_return:.2%}'}
    return {}

initialize_signals_log()

if __name__ == "__main__":
    log_system_event('START', 'Тестирование модуля мониторинга')
    test_params = {'rsi_threshold_low': 28, 'rsi_threshold_high':
72}
    log_signal(
        signal='buy',
        phase=1,
        strategy_name='momentum_strategy',
        params=test_params,
        notes='Backtest profit: +5.2%'
    )
    log_system_event('COMPLETED', 'Тест завершен успешно')

```

Листинг А.7 – Файл integrationTest_module.py

```

from data_module import load_and_preprocess_data
from lstm_module import generate_multioutput_predictions
from strategy_module import find_trade_points
from execution_module import send_telegram_signal,
send_telegram_forecast
from monitoring_module import log_signal, log_system_event
from datetime import datetime, timedelta

def generate_forecast_dates(last_date_str, periods):
    last_date = datetime.strptime(last_date_str, "%Y-%m-%d")

```

```

        return [(last_date + timedelta(days=i+1)).strftime("%Y-%m-%d") for i in range(periods)]

def main(user_type='regular'):
    log_system_event('START', 'Запуск ежедневного цикла торговой системы')
    try:
        df = load_and_preprocess_data('data.csv')
        preds = generate_multioutput_predictions(df)
        last_date_str = str(df['date'].iloc[-1]) if 'date' in df.columns else datetime.now().strftime("%Y-%m-%d")
        forecast_dates = generate_forecast_dates(last_date_str, len(preds))

        buy_points, sell_points, trades = find_trade_points(preds[:5], threshold=0.01)
        send_telegram_signal(buy_points, sell_points, trades, preds[:5], forecast_dates[:5], dry_run=False)

        if user_type == 'premium':
            buy_points_full, sell_points_full, trades_full = find_trade_points(preds, threshold=0.01)
            send_telegram_forecast(buy_points_full, sell_points_full, trades_full, preds, forecast_dates, dry_run=False)

        log_system_event('COMPLETED', 'Цикл завершён успешно')
    except Exception as e:
        log_system_event('ERROR', f'Критическая ошибка в основном цикле: {str(e)}')

if __name__ == "__main__":
    # Для обычного пользователя: main('regular')
    # Для премиум-пользователя: main('premium')
    main('regular')

```

ПРИЛОЖЕНИЕ Б

(рекомендуемое)

Программное обеспечение анализа и обработки данных

Результаты тестирования серверной части

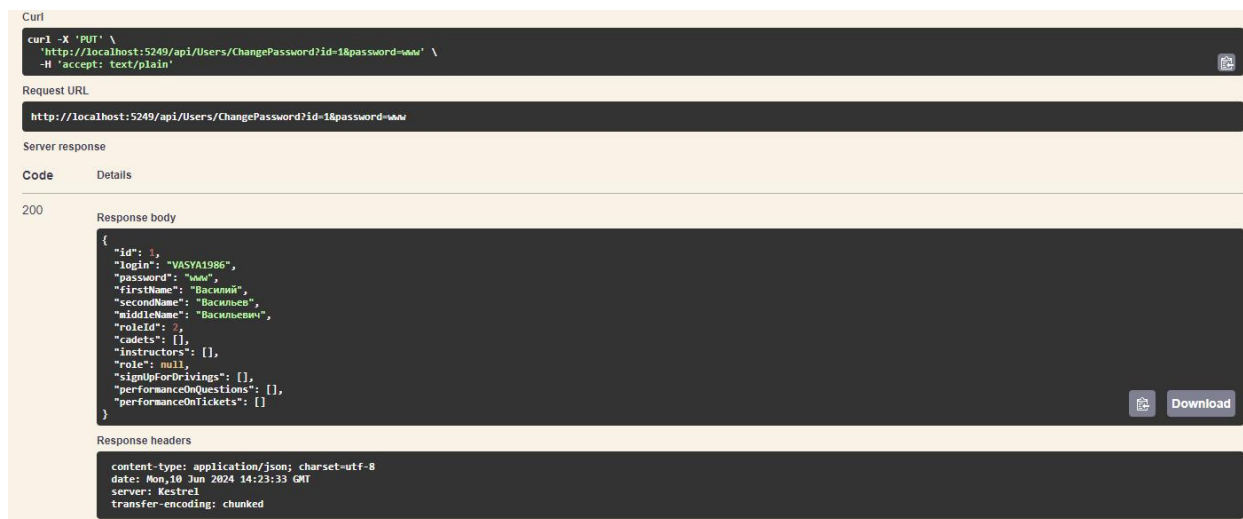


Рисунок Б.4 – Изменение горизонта прогнозирования

ПРИЛОЖЕНИЕ В

(рекомендуемое)

Программное обеспечение анализа и обработки данных

Результаты тестирования клиентской части

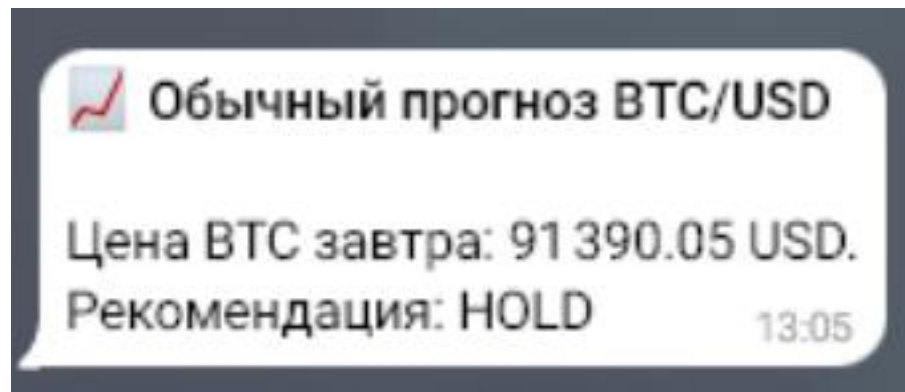


Рисунок В.1 – Обычный прогноз для обычного пользователя

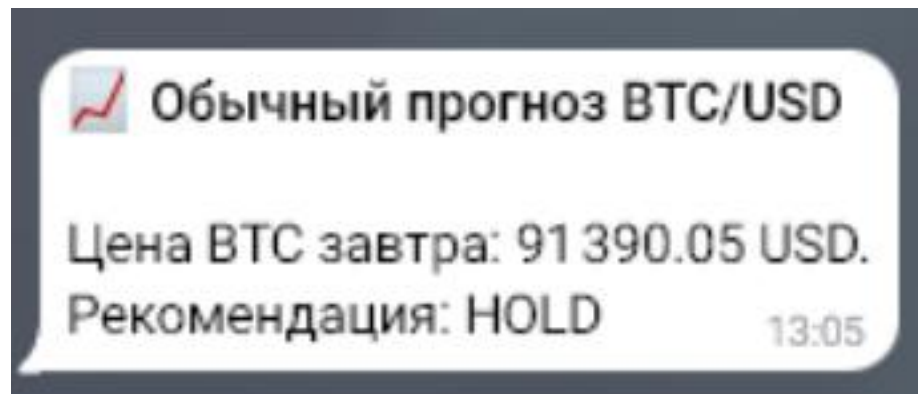


Рисунок В.2 – Обычный прогноз для премиум пользователя

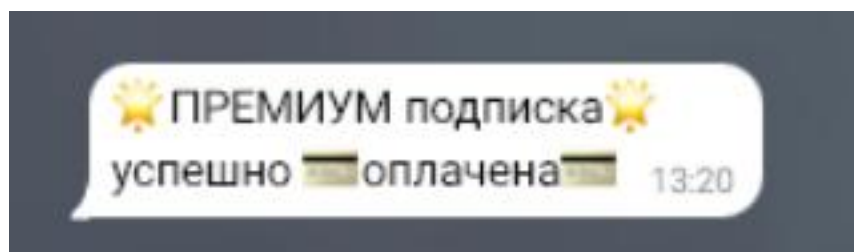


Рисунок В.3 – Оплата подписки

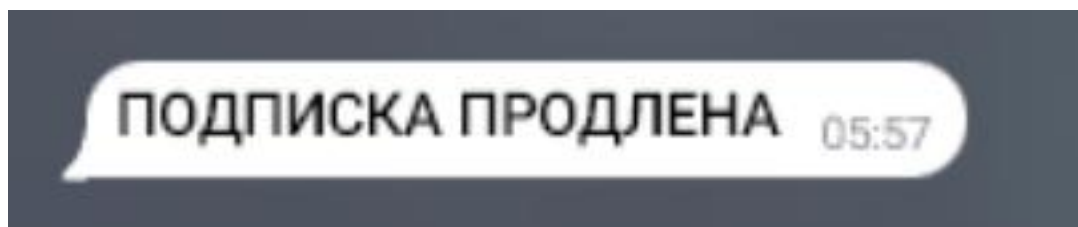


Рисунок В.4 – Продление подписки

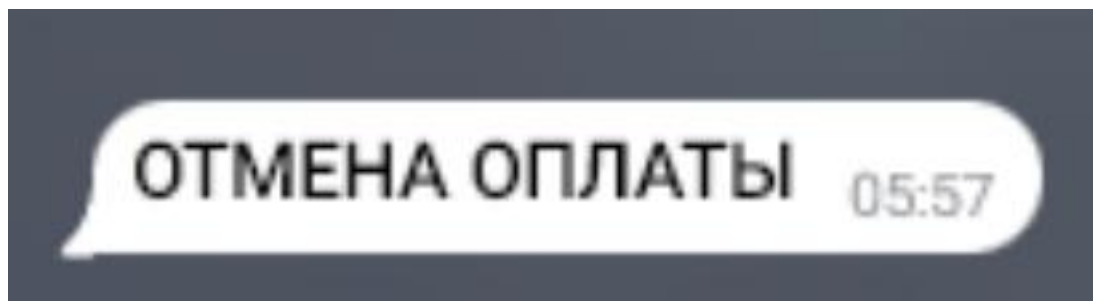


Рисунок В.5 – Прерванная оплата

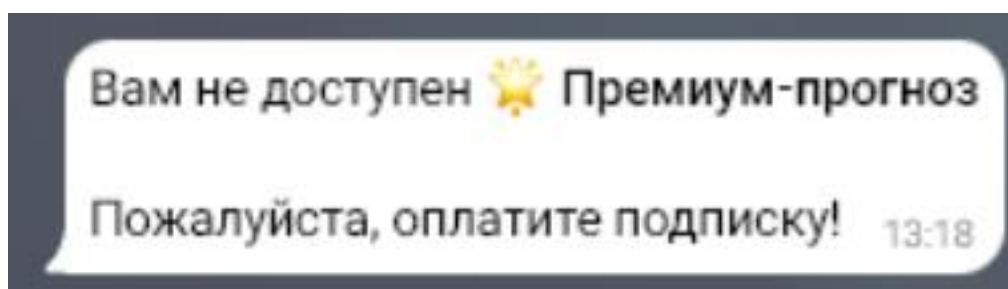


Рисунок В.6 – Премиум прогноз для обычного пользователя



Рисунок В.7 – Премиум прогноз для премиум пользователя

ГОРИЗОНТ ПРОГНОЗИРОВАНИЯ УСПЕШНО ИЗМЕНЁН 13:08

Рисунок В.8 – Смена горизонта для премиум пользователя

ОПЛАТИТЕ ПОДПИСКУ ДЛЯ ИЗМЕНЕНИЯ ГОРИЗОНТА ПРОГНОЗИРОВАНИЯ 13:00

Рисунок В.9 – Смена горизонта для обычного пользователя

ПРИЛОЖЕНИЕ Г

(рекомендуемое)

Программное обеспечение анализа и обработки данных

Презентация



Выпускная квалификационная работа на тему «Программное обеспечение анализа и обработки данных»

Направление подготовки: 09.04.01 – Информатика и вычислительная техника

Студент: группа 24ВВИм1

Ульченко Фёдор Сергеевич

Руководитель: к.т.н., доцент кафедры «ВТ»

Федюнин Роман Николаевич

Пенза, 2026

Объект и предмет разработки

Объект разработки: процесс прогнозирования цены биткоина.

Предмет разработки: программная система прогнозирования цены биткоина на основе LSTM-модели с доставкой прогнозов через Telegram-бота.

Цель и задачи разработки

Целью выпускной квалификационной работы является разработка программного обеспечения прогнозирующего цену биткоина, формирующего торговые сигналы, для повышения эффективности торговли частных трейдеров.

Основные **задачи** разработки:

1. Анализ предметной области и методов прогнозирования криптовалют.
2. Сравнительный анализ нейросетевых архитектур .
3. Проектирование и обучение LSTM-модели.
4. Разработка модуля торговой стратегии.
5. Реализация серверной части.
6. Разработка Telegram-бота.
7. Контейнеризация и развертывание на VPS.
8. Тестирование и анализ результатов.

Актуальность работы

«Стратегия развития информационного общества в Российской Федерации на 2017–2030 годы» (Указ Президента РФ № 203) определяет формирование цифровой экономики как национальный приоритет. Национальный проект «Экономика данных» направлен на внедрение ИИ и анализа больших данных в финансовый сектор.

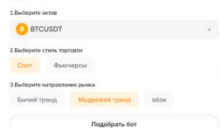
Глобальная капитализация криптовалют превышает 2,6 трлн долл., биткоина — 1,5 трлн долл. (2026).

Высокая волатильность биткоина требует применения методов глубокого обучения.

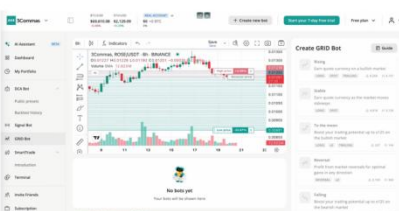
LSTM-модели статистически значимо превосходят традиционные эконометрические модели (Expert Systems with Applications, 2023).

Анализ предметной области Продукты аналогичного назначения

Подберите идеальный торговый бот



Bybit Aurora AI



3 Commas

GNyaiBOT

GNyaiBOT Readout For: BTC

- Current Trend: Neutral to slightly bullish, price oscillating around the 20-day EMA.
- Buy/Sell Signals: No strong signals; MACD histogram negative but line above signal line.
- Support Levels: Lower Bollinger Band at 78,951.036, recent low at 78,914.024.
- Resistance Levels: Upper Bollinger Band at 82,995.017, recent high at 82,049.766.
- Trend Analysis: Neutral with slight bullish bias, weak trend strength indicated by ADX at 31.12.
- Market Conditions: Neither overbought nor oversold, RSI at 59.945 leaning towards overbought.
- Price Action: Stable with minor fluctuations, within Bollinger Bands indicating consolidation.
- Momentum: Weak momentum; negative MACD histogram and declining McClellan Oscillator.
- Volume Activity: OBV negative at -1,149,487,953, Chaikin Oscillator at 4,078.857 showing reduced buying pressure.
- Notable Patterns: No significant chart patterns identified.
- Market Change Since Yesterday: Slight upward movement, closing price increased from 79,314.792 to 81,090.461.

Conclusion: Neutral with slight bullish bias. Caution advised; watch for breakout from consolidation range.

Report generated on May 15, 2026 - 01:21 using data from May 08, 2026 - 00:00 to May 14, 2026 - 00:00

GNyaiBOT

Сравнение аналогов

Рассматриваемая система	Прогноз	Горизонт прогноза	Горизонт анализа	Бесплатная версия	Стоимость премиум версии
Bybit Aurora AI	Нет	Нет	4 дня	Да	Нет
3Commas	Нет	Нет	30 дней	Нет	20-200\$/30дней
GNyaiBOT	Простой ТА	1 день	7 дней	Да	Нет
Разрабатываемое приложение	Полный прогноз	До 30 дней	13 лет	Да	20\$/30дней

Анализ целевой аудитории

Количество владельцев криптовалют в мире — более 700 млн человек (2025–2026).

Наиболее активная возрастная группа — 25–34 лет (31%).

Доля институциональных инвесторов, выделяющих диверсификацию как мотив — 63% (CoinShares, 2026).

Вывод: целевая аудитория — частные трейдеры, заинтересованные в оперативных и точных прогнозах.

Анализ функциональных требований Диаграмма вариантов использования

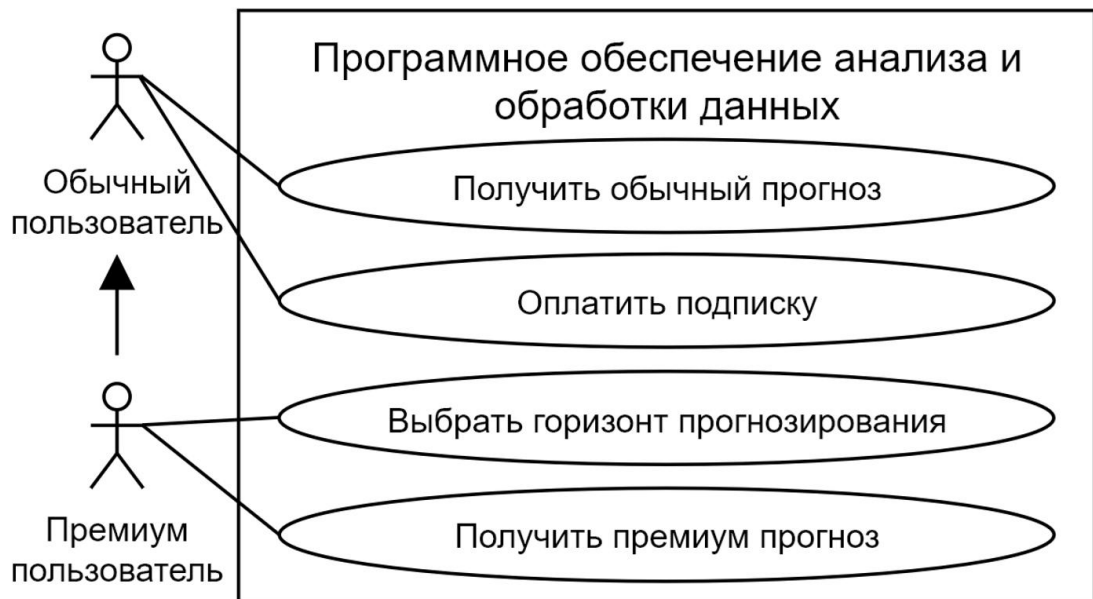


Диаграмма деятельности варианта использования «получить премиум прогноз»

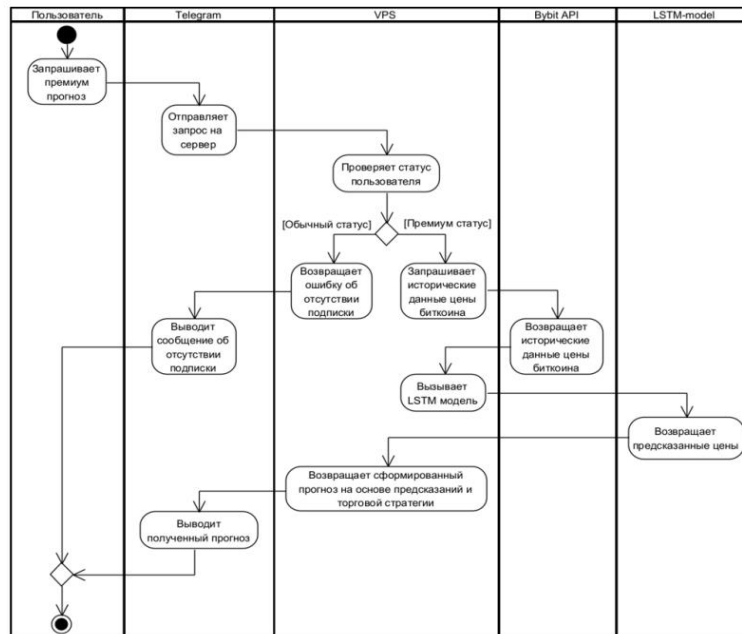
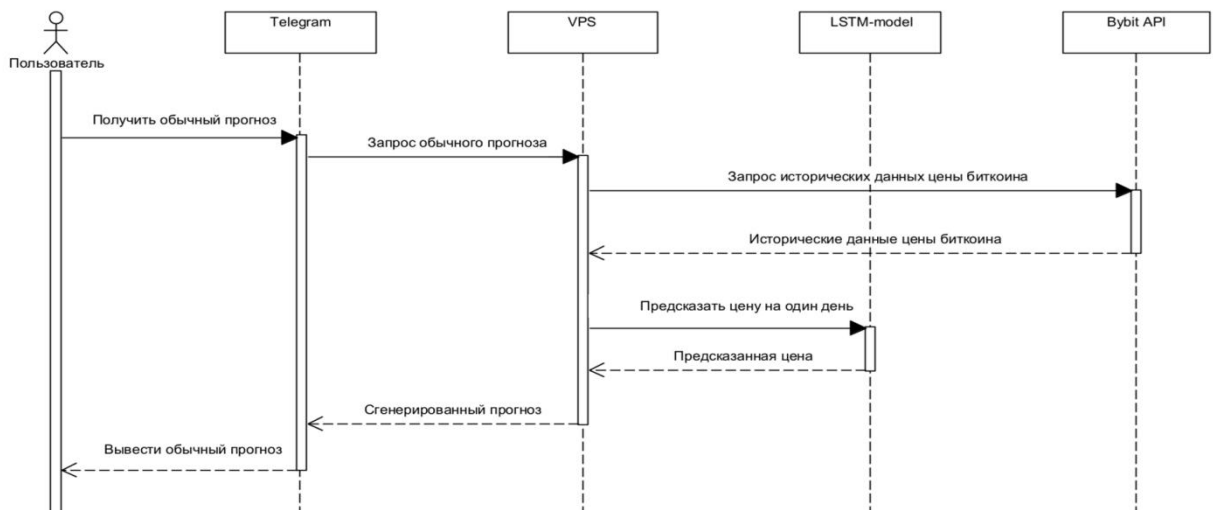
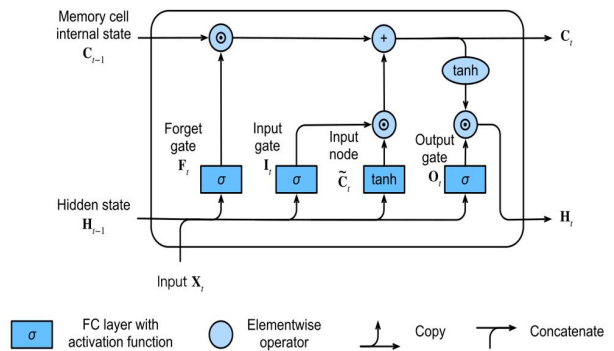


Диаграмма последовательности варианта использования «получить обычный прогноз»



Анализ нефункциональных требований Используемые технологии разработки

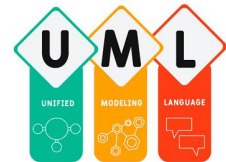
- RUP
- LSTM
- TensorFlow
- Keras
- Optuna
- Telegram API



Используемые средства разработки

Языковые средства:

- Python;
- UML.

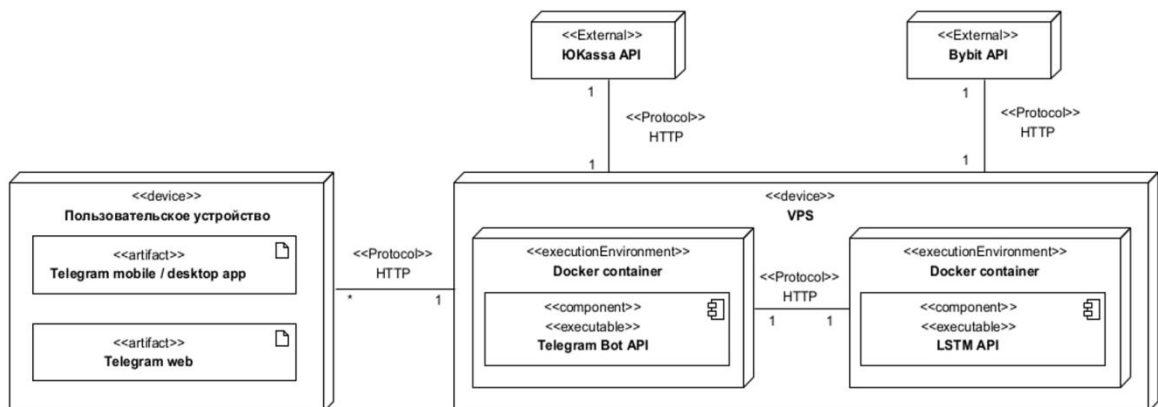


Инструментальные средства:

- Microsoft Visual Studio Code;
- Visual Paradigm;
- Microsoft Project.

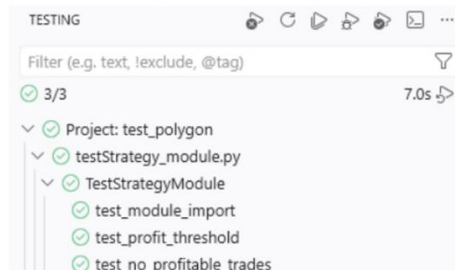


Диаграмма развертывания программного обеспечения

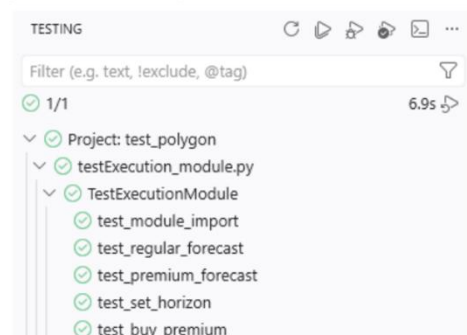


Контроль качества

Модульное тестирование серверной части:



Модульное тестирование клиентской части:



Основные результаты работы

- Выполнен анализ предметной области и требований;
- выполнен анализ программного обеспечения аналогичного назначения;
- произведен анализ целевой аудитории;
- спроектирована архитектура программного обеспечения;
- спроектирована, реализована и обучена LSTM-модель;
- спроектирован и реализован торговый модуль;
- спроектирован и реализован Telegram-бот;
- спроектирована и реализована серверная часть приложения;
- проведено функциональное и модульное тестирование приложения;
- рассчитаны некоторые метрики программного кода;
- в перспективе планируется расширение списка прогнозируемых активов;
- по теме данной выпускной квалификационной работы опубликована научная статья.

Спасибо за внимание