



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ
Преддипломная практика

приказ Университета о направлении на практику от «21» апреля 2025 г. №3669-С

Отчет представлен к
рассмотрению:

Студент группы ИКБО-20-21

«17» мая 2025

Мухаметшин А.Р.

(подпись и расшифровка подписи)

Отчет утвержден.
Допущен к защите:

Руководитель практики
от кафедры

«17» мая 2025

Аждер Т.Б.

(подпись и расшифровка подписи)

Москва 2025 г.



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ
Преддипломная практика

Студенту 4 курса учебной группы ИКБО-20-21

Мухаметшину Александру Ринатовичу

Место и время практики: РТУ МИРЭА кафедра ИиППО, с 21 апреля 2025 г. по 17 мая 2025 г.

Должность на практике: студент

1. СОДЕРЖАНИЕ ПРАКТИКИ:

1.1. Изучить: научную и техническую литературу, электронные информационно-образовательные ресурсы, применяемые для профессиональной деятельности по теме практики

1.2. Практически выполнить:

1.2.1 Спроектировать бизнес-процессы исследуемой предметной области, для их автоматизации в информационной системе ИС;

1.2.2 Сформировать требования к ИС;

1.2.3 Осуществить проектирование ИС;

1.2.4 Описать реализацию клиентской и серверной части ИС;

1.2.5 Описать обеспечение информационной безопасности при эксплуатации ИС;

1.2.6 Описать работу пользователя с ИС

1.2.7. Оценить технические характеристики практической разработки;

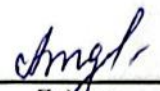
1.2.8. Выполнить тестирование разрабатываемого проекта, собрать метрики и оценить результат проведенного тестирования.

1.3. Ознакомиться: с актуальными нормативно-правовыми документами, международными и отечественными стандартами, с СУБД и инструментальными средствами разработки ИС


2. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ: подготовить презентацию и доклад по результатам практики.

3. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ: в процессе практики рекомендуется использовать периодические издания и отраслевую литературу годом издания не старше 5 лет от даты начала прохождения практики

Руководитель практики от кафедры
«21» апреля 2025 г.



Подпись (Аждер Т.Б.)

Задание получил
«21» апреля 2025 г.


Подпись (Мухаметшин А.Р.)

СОГЛАСОВАНО:
Заведующий кафедрой:

«21» апреля 2025 г.


Подпись (Болбаков Р.Г.)

Проведенные инструктажи:

Охрана труда:

«21» апреля 2025 г.

Инструктирующий


Подпись

Болбаков Р.Г., зав. кафедрой
ИиППО

Инструктируемый


Подпись

Мухаметшин А.Р.

Техника безопасности:

«21» апреля 2025 г.

Инструктирующий


Подпись

Болбаков Р.Г., зав. кафедрой
ИиППО

Инструктируемый


Подпись

Мухаметшин А.Р.

Пожарная безопасность:

«21» апреля 2025 г.

Инструктирующий


Подпись

Болбаков Р.Г., зав. кафедрой
ИиППО


Инструктируемый


Подпись

Мухаметшин А.Р.

С правилами внутреннего распорядка ознакомлен:

«21» апреля 2025 г.


Подпись

Мухаметшин А.Р.



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

**РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ
ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ**

студента Мухаметшина А.Р. 4 курса группы ИКБО-20-21 очной формы обучения, обучающегося по направлению подготовки 09.03.04 Программная инженерия, профиль «Разработка программных продуктов и проектирование информационных систем».

Неделя	Сроки выполнения	Этап	Отметка о выполнении
1	21.04.2025	Подготовительный этап, включающий в себя организационное собрание (Вводная лекция о порядке организации и прохождения производственной практики, инструктаж по технике безопасности, получение задания на практику)	Выполнено 21.04.25 <i>Amgl</i>
1	22.04.2025- 25.04.2025	Аналитический этап (Построение графических моделей БП "as is", "to be"; разработка логической модели базы данных; формирование таблиц сущностей с атрибутами и типами данных; описание входной информации, применяемых классификаторов и справочников; построение макетов выходной информации; описание математического обеспечения)	Выполнено 25.04.25 <i>Amgl</i>
1	26.04.2025	Представление руководителю структурированного материала согласно содержанию выше указанного этапа	Выполнено 26.04.25 <i>Amgl</i>
2	28.04.2025- 02.05.2025	Технологический этап (Обоснование выбора средств разработки; описание реализации базы данных (логическая и физическая модели базы данных); построение экранных форм)	Выполнено 02.05.25 <i>Amgl</i>
2	03.05.2025	Представление руководителю материала согласно содержанию выше указанного этапа, реализуемого на 2 неделе; а также иного ранее непредставленного материала предыдущих этапов в случае наличия отметок о невыполнении	Выполнено 03.05.25 <i>Amgl</i>
3	05.05.2025- 10.05.2025	Технологический этап (Описание реализации клиентской части ИС) описание обеспечения информационной безопасности эксплуатации ИС)	Выполнено 10.05.25 <i>Amgl</i>

4	12.05.2025- 13.05.2025	Представление руководителю материала согласно содержанию выше указанного этапа, реализуемого на 3 неделе; а также иного ранее непредставленного материала предыдущих этапов в случае наличия отметок о невыполнении	Выполнено 13.05.25 Стмл.
4	14.05.2025	Представление руководителю предварительной версии отчета с обеспечением согласованности материала по всем его частям, полученных на	Выполнено 14.05.25 Стмл.
4	15.05.2025- 16.05.2025	Подготовка окончательной версии отчета по практике (Оформление материалов отчета в полном соответствии с требованиями на оформление письменных учебных работ студентов)	Выполнено 16.05.25 Стмл.
4	17.05.2025	Представление окончательной версии отчета по практике руководителю	Выполнено 17.05.25 Стмл.

Руководитель практики от
кафедры

Стмл. /Аждер Т.Б., к.т.н., доцент/

Обучающийся

Мухаметшин А.Р. /Мухаметшин А.Р./

Согласовано:

Заведующий кафедрой

Болбаков Р.Г. /Болбаков Р.Г., к.т.н., доцент/

УДК 004.4

Руководитель практики: к.т.н., доцент Т.Б. Аждер

Консультант по экономическому разделу: к.э.н., доцент И.В. Чижанькова

Мухаметшин А.Р., Преддипломная практика по образовательной подготовке бакалавров 09.03.04 «Программная инженерия» на тему «Интеллектуальная система оценки спроса на продукт»: М. 2025 г., МИРЭА – Российский технологический университет (РТУ МИРЭА), Институт информационных технологий (ИТ), кафедра инструментального и прикладного программного обеспечения (ИиППО) – 41 стр., 10 илл., 5 табл., 5 листинг., 0 формул, 15 ист. лит (в т.ч. 10 на английском яз.).

Ключевые слова: прогнозирование спроса, машинное обучение, микросервисная архитектура, Python, Golang, анализ данных, REST API, Docker, база данных.

Целью работы является разработка интеллектуальной системы для автоматизированной оценки спроса на продукт, основанной на анализе данных из разнородных источников с применением алгоритмов машинного обучения.

Mukhametshin A.R., Pre-graduate practice on educational training of bachelors 09.03.04 “Software Engineering” on the topic “Intelligent system of product demand estimation”: M. 2025, MIREA - Russian Technological University (RTU MIREA), Institute of Information Technologies (IT), Department of Instrumental and Applied Software (IiPPO) - 41 p., 10 illustrations, 5 tables, 5 listings, 0 formulas, 15 references (including 10 in English).

Keywords: demand forecasting, machine learning, microservice architecture, Python, Golang, data analysis, REST API, Docker, database.

The goal of the work is to develop an intelligent system for automated product demand estimation based on analyzing data from heterogeneous sources using machine learning algorithms.

РТУ МИРЭА: 119454, Москва, пр-т Вернадского, д. 78

Кафедра инструментального и прикладного программного обеспечения (ИиППО)

Тираж: 1 экз. (на правах рукописи)

Файл: «ПП_ИППО_МухаметшинАР_ИКБО-20-21.pdf», исполнитель Мухаметшин А.Р.

© Мухаметшин А. Р.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ	6
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	7
ВВЕДЕНИЕ	8
1 ИССЛЕДОВАНИЕ И ПРОЕКТИРОВАНИЕ СИСТЕМЫ.....	10
1.1 Анализ существующих решений	10
1.2 Определение требований к решению	11
1.3 Выбор инструментов и методов создания информационной системы	12
1.4 Проектирование архитектуры информационной системы.....	13
1.5 Проектирование клиентской части информационной системы.....	13
1.6 Проектирование серверной части информационной системы.....	14
1.7 Проектирование схемы базы данных	15
Вывод по разделу 1	16
2 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ	17
2.1 Разработка серверной части системы.....	17
2.2 Разработка клиентской части системы.....	24
2.3 Тестирование системы	28
2.4 Расчёт вычислительной и емкостной сложности	32
2.5 Расчет экономической стоимости	34
Вывод по разделу 2	35
ЗАКЛЮЧЕНИЕ	36
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	37
Приложение А	39

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчёте применяются следующие термины и определения.

Анализ данных	– процесс обработки и интерпретации данных для выявления закономерностей и поддержки принятия решений.
Контейнеризация	– Технология, позволяющая упаковывать приложения и их зависимости в изолированные контейнеры для упрощения развертывания и масштабирования (например, Docker).
Машинное обучение	– раздел искусственного интеллекта, использующий алгоритмы для обучения моделей на основе данных.
Микросервисная архитектура	– подход к разработке, при котором приложение разбивается на независимые, автономные сервисы, каждый из которых отвечает за конкретную функцию системы.
Прогнозирование спроса	– метод оценки будущих потребностей рынка на основе анализа исторических данных и текущих трендов.
Масштабируемость	– возможность увеличивать ресурсы или добавлять дополнительные экземпляры сервисов для поддержания производительности системы при росте нагрузки.
REST API	– архитектурный стиль для создания программных интерфейсов, использующий HTTP-запросы для обмена данными между клиентом и сервером.
Веб-скрапинг	– технология автоматизированного извлечения данных с веб-сайтов для последующей обработки и анализа.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчёте применяются следующие сокращения и обозначения.

API	–	Application Programming Interface (Программный интерфейс приложения)
CSS	–	Cascading Style Sheets (каскадные таблицы стилей)
CSV	–	Comma-Separated Values (формат файла для хранения табличных данных)
ERP	–	Enterprise Resource Planning (Планирование ресурсов предприятия)
ML	–	Machine Learning (Машинное обучение)
ИС	–	Информационная система
ИТ	–	Информационные технологии
ПО	–	Программное обеспечение

ВВЕДЕНИЕ

В условиях стремительной цифровизации и динамично меняющихся рыночных реалий компании сталкиваются с необходимостью оперативного реагирования на изменения спроса. Традиционные методы прогнозирования, основанные на экспертных оценках или ручном анализе данных, теряют свою эффективность, не успевая за скоростью рыночных трансформаций. В этой связи разработка интеллектуальных систем, способных автоматизировать процессы сбора, обработки и анализа данных с использованием технологий машинного обучения, становится стратегически важной задачей для повышения конкурентоспособности бизнеса.

Целью преддипломной практики является создание интеллектуальной системы (ИС) для автоматизированной оценки спроса на продукт. Система позволит бизнесу прогнозировать рыночные потребности на основе анализа данных из разнородных источников с применением технологий машинного обучения. Она направлена на предоставление компаниям инструмента для точного предсказания спроса и оптимизации бизнес-процессов, что способствует снижению издержек и повышению эффективности.

Актуальность разработки обусловлена тем, что в современных условиях традиционные подходы к оценке спроса становятся недостаточно точными и оперативными. Исследования показывают, что внедрение интеллектуальных систем анализа данных позволяет сократить издержки на 15–20% за счет минимизации избыточных запасов и упущенных возможностей [1]. Кроме того, использование технологий машинного обучения для прогнозирования спроса обеспечивает рост точности предсказаний до 85% по сравнению с традиционными методами [2]. Таким образом, разработка подобных систем приобретает ключевое значение в условиях цифровизации бизнеса.

Новизна работы заключается в создании универсальной информационной системы, которая интегрирует сбор данных из внутренних источников (CRM, ERP), внешних API (например, Google Trends) и веб-

скрапинга (данные маркетплейсов), их обработку с применением ML-моделей и предоставление результатов через удобный веб-интерфейс и API. В отличие от существующих решений, предлагаемая система обладает гибкостью настройки источников данных и прогнозных моделей, что обеспечивает высокую адаптивность к различным бизнес-сценариям и требованиям пользователей.

В ходе работы требуется выполнить следующие задачи: – провести анализ предметной области и существующих решений; – спроектировать архитектуру системы; – выбрать средства и методы для разработки; – разработать информационную систему и провести тестирование; – рассчитать стоимость проведения работ.

Объектом исследования является интеллектуальная система оценки спроса на продукт, включающая модули сбора, предобработки и анализа данных с использованием технологий машинного обучения для прогнозирования рыночных потребностей.

Предметом исследования являются процессы автоматизированного сбора, предобработки и анализа данных из разнородных источников с использованием технологий машинного обучения для прогнозирования спроса на продукт в интеллектуальной системе.

На защиту выносятся информационная система, обеспечивающая автоматизированный сбор данных из различных источников, их анализ с помощью ML-моделей и предоставление прогнозов спроса через API и веб-интерфейс.

Стандарты, используемые в работе: СМКО МИРЭА 7.5.1/03.П.30-19 [3], ГОСТ 7.32-2017 [4], ГОСТ Р 7.0.100-2018 [5], СМКО МИРЭА 7.5.1/03.П.67-19 [6], ФГОС ВО 3++ по направлению подготовки 09.03.04 Программная инженерия [7].

1 ИССЛЕДОВАНИЕ И ПРОЕКТИРОВАНИЕ СИСТЕМЫ

1.1 Анализ существующих решений

В условиях цифровизации бизнеса и роста объемов данных системы автоматической оценки спроса становятся важным инструментом для компаний, стремящихся оптимизировать управление запасами и повысить эффективность маркетинговых стратегий. Для определения требований к разрабатываемой информационной системе проведен сравнительный анализ существующих решений. Были исследованы следующие системы:

- Ozon Seller [8]
- Moneyplace [9]
- MPStats [10]

По итогу сравнительного анализа существующих аналогов разрабатываемой ИС составлена таблица 1.1.

Таблица 1.1 – Сравнительный анализ аналогов ИС

Критерий	Ozon Seller	Moneyplace	MPStats
Функциональность	Отчеты о продажах, анализ конкурентов, рекомендации по закупкам	Анализ спроса, сравнение цен, прогноз остатков	Аналитика продаж, анализ конкурентов, оптимизация рекламы, исследование товаров
Интеграция	REST API для доступа к данным	API для получения аналитических данных	API для получения аналитических данных
Гибкость	Ограничена платформой Ozon	Поддержка нескольких маркетплейсов	Поддержка Wildberries, Ozon, Яндекс.Маркет
Аналитика и статистика	Статистика продаж и конкурентов	Детализированная аналитика по товарам	Подробные отчеты по продажам, выручке, ценам, рейтингам
Дополнительные возможности	Простота интерфейса	Анализ данных с разных платформ	Расширение для браузеров, инструменты для продавцов

Проведенный анализ показывает, что существующие решения, такие как Ozon Seller, Moneyplace и MPStats, ориентированы преимущественно на анализ данных внутри конкретных маркетплейсов и не обладают достаточной гибкостью для интеграции внешних источников или применения продвинутых методов прогнозирования. Разрабатываемая система должна преодолеть эти ограничения.

Таким образом, разрабатываемая система должна сочетать универсальность, высокую точность прогнозов и удобство использования, что обеспечит ее конкурентоспособность на рынке аналитических решений.

1.2 Определение требований к решению

Функциональные требования

Система должна автоматизировать сбор данных об использовании продуктов и прогнозировать их спрос. Целевой аудиторией являются владельцы бизнесов.

Необходимо обеспечить:

- сбор данных из подготовленных пакетов данных, внешних API и веб-скрапинг;
- обработка и нормализация данных для ML;
- прогнозирование спроса с использованием ML-моделей;
- предоставление API для интеграции с внешними системами;
- визуализация прогнозов и аналитики через веб-интерфейс.

Нефункциональные требования

Основными метриками, которые необходимо обеспечить является:

- время отклика API – не более 1 секунды при нагрузке до 100 пользователей;
- время генерации прогноза – не более 5 минут;
- безопасность: авторизация через JWT [13], шифрование данных.

Система должна запускаться в Docker'е для обеспечения кроссплатформенности.

Требования пользователей

Для удобства использования система должна иметь:

- удобный интерфейс для настройки источников данных и просмотра прогнозов;
- экспорт отчетов в PDF/Excel.

1.3 Выбор инструментов и методов создания информационной системы

Для серверной части системы выбраны следующие технологии:

- Go (Gin [11]): Высокопроизводительный язык программирования с фреймворком Gin для создания микросервисов;
- RabbitMQ: Брокер сообщений для асинхронной коммуникации между микросервисами.

Преимущества данного подхода включают высокую скорость выполнения запросов, легкость масштабирования и надежность передачи данных, что соответствует требованиям микросервисной архитектуры.

Клиентская часть системы реализована с использованием следующих технологий:

- React [12] и TypeScript: React.js – библиотека JavaScript для построения динамических и адаптивных пользовательских интерфейсов с применением компонентного подхода.

Использование React с TypeScript и Tailwind CSS позволяет создать удобный, быстрый и адаптивный веб-интерфейс, обеспечивающий интерактивное взаимодействие с системой и визуализацию результатов анализа.

Для управления данными выбраны две системы:

- PostgreSQL: Основная реляционная база данных для хранения структурированных данных (пользователи, обработанные данные, прогнозы).

Модуль машинного обучения реализован с использованием:

– Python (scikit-learn, LightGBM): Python выбран как основной язык для разработки ML-моделей благодаря богатому набору библиотек. Scikit-learn используется для обработки данных, а LightGBM – для построения высокоэффективных градиентных бустинговых моделей, которые отлично подходят для прогнозирования временных рядов спроса и цен;

– Go с интеграцией Python: Для эффективного использования моделей применяется подход с вызовом Python-скриптов из Go-сервиса, что обеспечивает гибкость при обучении и использовании моделей при сохранении высокой производительности API.

Такой подход обеспечивает гибкость в обучении моделей и их эффективное использование в реальном времени.

Для удобного развертывания и масштабирования системы используется Docker и Docker Compose. Каждый микросервис работает в своём контейнере, что обеспечивает изоляцию и упрощает управление зависимостями.

1.4 Проектирование архитектуры информационной системы

Система основана на микросервисной архитектуре в рамках трёхуровневой модели (презентация, логика, данные), что обеспечивает гибкость, масштабируемость и высокую отказоустойчивость. API Gateway выступает единой точкой входа, принимая запросы от клиентов, выполняя проверку аутентификации с использованием JWT-токенов и маршрутизацию к нужным сервисам. Связь между микросервисами реализована как через синхронные HTTP-запросы для быстрых операций, так и асинхронно через брокер сообщений RabbitMQ, что позволяет обрабатывать задачи в фоновом режиме и повышает устойчивость системы при пиковых нагрузках. Для хранения данных используется реляционная СУБД PostgreSQL, выбранная за её надёжность, производительность и поддержку сложных аналитических запросов, таких как агрегация временных рядов.

1.5 Проектирование клиентской части информационной системы

Клиентская часть представляет собой веб-приложение, разработанное на

React с использованием TypeScript, что обеспечивает удобный и современный интерфейс для управления данными, просмотра прогнозов и аналитических отчётов. React позволяет создавать модульные и переиспользуемые компоненты интерфейса, такие как таблицы или графики, а TypeScript добавляет статическую типизацию, что минимизирует ошибки и улучшает читаемость кода. Взаимодействие с серверной частью осуществляется через REST API, предоставляемый API Gateway, что гарантирует безопасную и стандартизированную передачу данных. Компонентный подход React упрощает поддержку кода и добавление новых функций, таких как интерактивные визуализации аналитических данных, что критично для бизнес-аналитиков.

1.6 Проектирование серверной части информационной системы

Серверная часть состоит из набора микросервисов, каждый из которых выполняет строго определённую задачу:

- Data Collector Service: отвечает за сбор данных о товарах из различных источников (например, CSV-файлы, API поставщиков) и передачу их в RabbitMQ для асинхронной обработки.

- Data Processor Service: получает данные из RabbitMQ, выполняет очистку (удаление дубликатов, обработка пропусков), нормализацию и сохраняет их в PostgreSQL, подготавливая для аналитики.

- ML Service: обучает модели машинного обучения (например, LightGBM) на основе исторических данных для прогнозирования спроса и цен, сохраняет обученные модели и предоставляет API для генерации прогнозов в реальном времени.

- Auth Service: управляет аутентификацией и авторизацией пользователей, генерирует и проверяет JWT-токены для обеспечения безопасности.

- API Gateway: централизованно маршрутизирует запросы от клиента к соответствующим сервисам, проверяет токены и балансирует нагрузку.

Связь между сервисами осуществляется через HTTP для быстрых операций и RabbitMQ для длительных задач. Каждый сервис следует паттерну Controller-Service-Repository, что обеспечивает чёткое разделение бизнес-логики, обработки данных и доступа к базе.

1.7 Проектирование схемы базы данных

Схема базы данных на PostgreSQL разработана для поддержки всех ключевых функций системы и оптимизирована для аналитики. Основные сущности включают:

- Пользователи (users): хранит данные для аутентификации и ролей (Auth Service);
- Товары (products): содержит информацию о товарах, такую как идентификаторы и категории (Data Processor Service);
- Исторические данные (product_historical_data): временные ряды продаж и спроса для обучения моделей;
- Прогнозы (predictions): результаты работы ML Service, включая значения и метаданные прогнозов;
- Сессии (sessions): данные о пользовательских сессиях для повышения безопасности (Auth Service).

База данных контейнеризирована с использованием Docker, что обеспечивает изоляцию, удобство развёртывания и дополнительную защиту данных.

Разработанная схема базы данных представлена на рисунке 1.1.

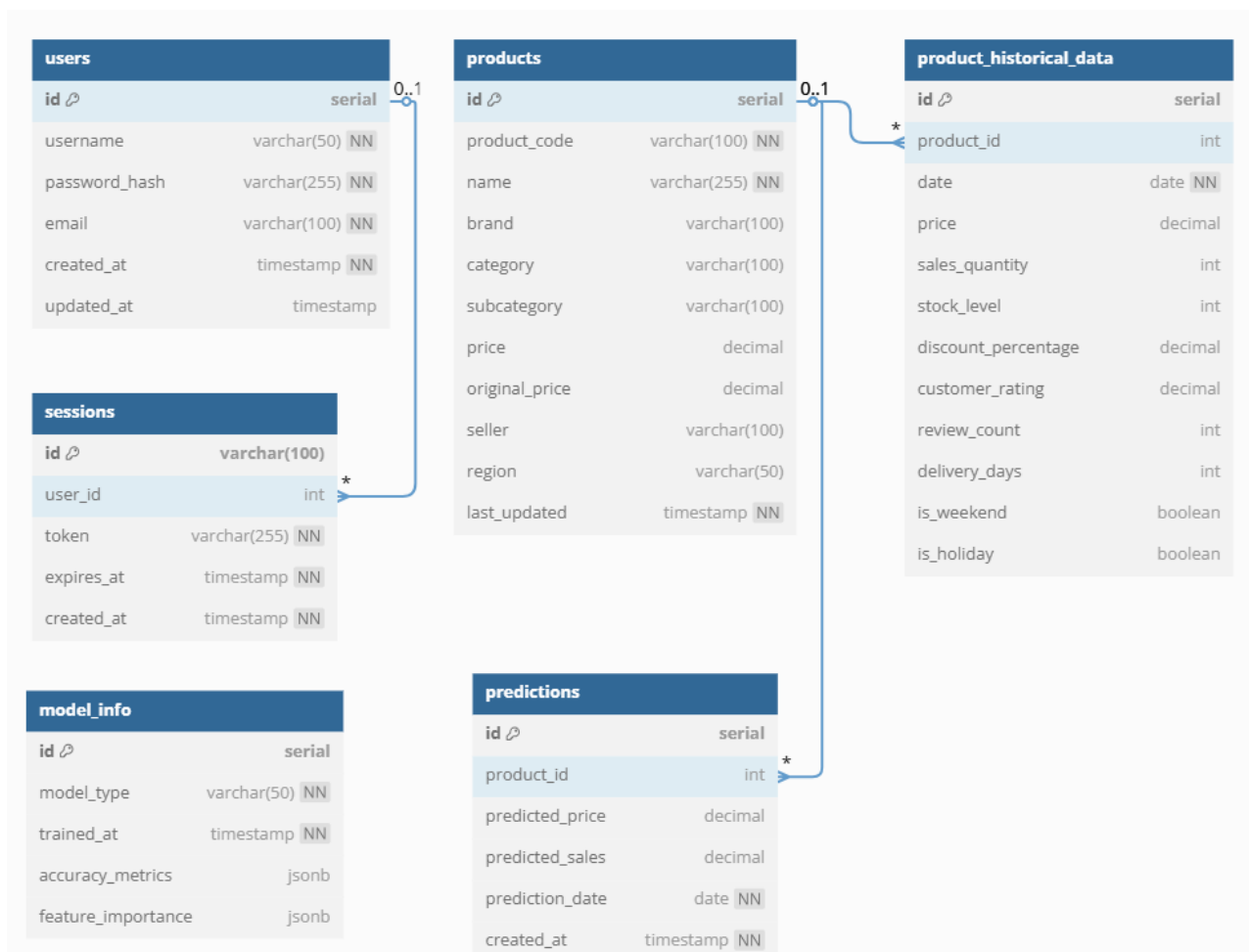


Рисунок 1.1 – Схема базы данных системы

Вывод по разделу 1

В данном разделе проведен сравнительный анализ существующих решений для оценки спроса, что позволило выявить ключевые требования и конкурентные преимущества разрабатываемой системы. На основе анализа определены функциональные и нефункциональные требования, а также выбраны оптимальные инструменты и методы разработки. Созданы компоненты клиентской части на базе React с TypeScript и серверной части, состоящей из специализированных микросервисов и спроектирована схема базы данных PostgreSQL. Предложенное решение обеспечивает гибкость, масштабируемость и отказоустойчивость системы в соответствии с современными стандартами программной инженерии [15].

2 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

2.1 Разработка серверной части системы

При реализации каждого модуля было принято решение следовать принципам Чистой архитектуры – архитектурного подхода к проектированию программного обеспечения, цель которого – сделать код легко читаемым, тестируемым, расширяемым и независимым от внешних деталей, таких как базы данных, брокеры сообщений или интерфейс пользователя. Основные идеи чистой архитектуры:

- разделение ответственности – код делится на слои, каждый из которых отвечает за свою функциональность;
- зависимости направлены внутрь – внешние компоненты (БД, брокеры сообщений) зависят от бизнес-логики, а не наоборот;
- интерфейсы и инверсии зависимостей – конкретные реализации внедряются через интерфейсы, что облегчает подмену и тестирование.

Следуя этим принципам была реализована структура, представленная на рисунке 2.1.

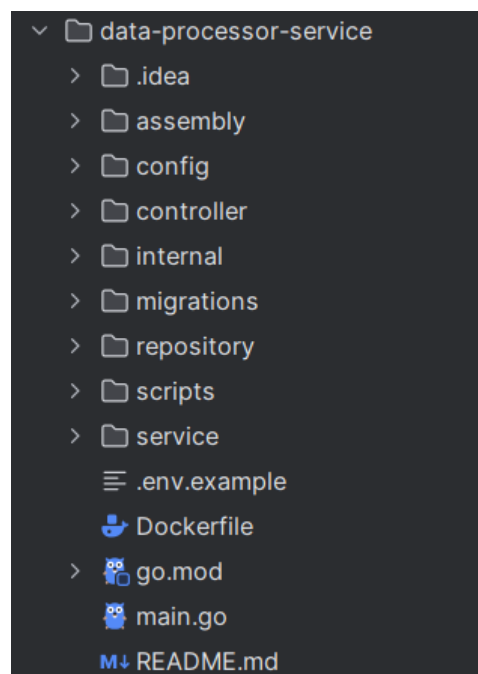


Рисунок 2.1 - структура проекта

Такая структура позволяет достичь высокой модульности кода, упрощает тестирование и обеспечивает четкое разделение ответственности между компонентами сервиса обработки данных.

Оркестрация микросервисов осуществляется с помощью `docker-compose.yml`, который определяет взаимосвязи между контейнерами, настраивает сетевые соединения через единую сеть `app-network` и конфигурирует переменные окружения для каждого сервиса. Файл также описывает тома для постоянного хранения данных (PostgreSQL, RabbitMQ) и устанавливает зависимости между сервисами, гарантируя правильный порядок запуска. Благодаря такой организации, вся система запускается единой командой `docker-compose up`, что существенно упрощает процесс развертывания как в среде разработки, так и на `production`-серверах.

2.1.1 Разработка модуля авторизации

Сервис авторизации (`auth-service`) представляет собой ключевой компонент системы, отвечающий за аутентификацию пользователей, управление сессиями и обеспечение безопасного доступа к ресурсам приложения. Данный микросервис был выделен в отдельный модуль, что позволило централизовать логику безопасности и разделить ответственность между компонентами системы.

Сервис авторизации реализует два интерфейса взаимодействия: REST API для операций регистрации, входа и выхода пользователей, а также внутренний API для проверки токенов и получения информации о пользователях. Такой подход обеспечивает гибкость интеграции и четкое разделение публичного и внутреннего API.

Одна из ключевых функций `auth-service` — регистрация новых пользователей и их аутентификация. При регистрации сервис проверяет уникальность имени пользователя и `email`, после чего сохраняет информацию в базе данных PostgreSQL. Важной особенностью реализации является безопасное хранение паролей — они не хранятся в открытом виде, а

хешируются с помощью алгоритма bcrypt [14], что обеспечивает защиту даже в случае компрометации базы данных.

Для обеспечения безопасности системы auth-service использует JWT (JSON Web Tokens) — компактный и самодостаточный способ безопасной передачи информации между сторонами в виде JSON-объекта. Это позволяет реализовать механизм авторизации без необходимости хранения состояния сессии на сервере.

Процесс авторизации запроса включает несколько шагов: клиент отправляет запрос с JWT-токеном в заголовке Authorization, API Gateway извлекает токен и отправляет запрос на проверку в auth-service, который проверяет валидность токена и возвращает информацию о пользователе. API Gateway, получив подтверждение, направляет запрос в соответствующий сервис.

2.1.2 Разработка модуля сбора данных

Сервис сбора данных (data-collector-service) представляет собой ключевой компонент разработанной системы, ответственный за получение, первичную обработку и передачу данных о товарах для последующего анализа и прогнозирования спроса. Выделение этой функциональности в отдельный микросервис обусловлено необходимостью изолировать процесс получения данных от их обработки, что обеспечивает более эффективное масштабирование и устойчивость системы в целом.

Основная задача data-collector-service – загрузка данных из внешних источников и их трансформация в унифицированный формат для последующей передачи в сервис обработки данных. Сервис работает как с периодическими запланированными задачами, так и с ручными запросами на получение данных, что обеспечивает гибкость в обновлении информации о товарах.

Взаимодействие с другими компонентами системы реализовано через асинхронный обмен сообщениями с использованием брокера RabbitMQ. Это

позволяет разделить процессы сбора и обработки данных, обеспечивая их независимость и отказоустойчивость.

Так же в данном сервисе реализован Планировщик задач (Scheduler) – он управляет периодическим запуском сбора данных согласно настраиваемому расписанию. Это обеспечивает регулярное обновление информации без необходимости ручного вмешательства. Запуск планировщика показан на листинге 2.1.

Листинг 2.1 – Запуск планировщика сбора данных

```
func (p *DataProcessor) StartScheduler(ctx context.Context, interval
time.Duration) {
    p.logger.Infof("Starting scheduler with interval: %v", interval)

    if err := p.ProcessData(ctx); err != nil {
        p.logger.Errorf("Initial data processing failed: %v", err)
    }
    ticker := time.NewTicker(interval)
    defer ticker.Stop()
    for {
        select {
        case <-ticker.C:
            p.logger.Info("Scheduler triggered data processing")
            if err := p.ProcessData(ctx); err != nil {
                p.logger.Errorf("Scheduled data processing failed:
%v", err)}
        case <-ctx.Done():
            p.logger.Info("Scheduler stopped")
            return
        }
    }
}
```

2.1.3 Разработка модуля подготовки данных

Сервис подготовки данных (data-processor-service) выполняет критически важную роль в процессе анализа и прогнозирования спроса на товары. Его основная задача – преобразование собранных сырых данных в

структурированный и очищенный формат, пригодный для машинного обучения и аналитики.

Сервис получает данные из очереди RabbitMQ, куда их отправляет data-collector-service, и выполняет многоэтапную обработку. Этот процесс включает в себя нормализацию значений, удаление дубликатов, обработку пропущенных значений и вычисление дополнительных признаков, которые будут использоваться для прогнозирования.

Важной особенностью сервиса является создание и расчет временных характеристик, таких как скользящие средние, лаги продаж и цен, а также сезонные признаки (день недели, месяц, праздники). Эти признаки существенно повышают точность прогнозирования временных рядов. На листинге A.1 представлена функция подготовки временных признаков.

Ключевым компонентом сервиса является Python-скрипт, который использует библиотеки pandas для обработки данных и подготовки их для машинного обучения. Скрипт выполняет несколько важных функций:

- преобразование типов данных и интерполяцию пропущенных значений в числовых полях;
- создание временных признаков (день недели, месяц, квартал);
- расчет лаговых значений продаж и цен (за 1, 3 и 7 дней);
- вычисление скользящих средних для цен и продаж (за периоды 3 и 7 дней);
- формирование целевых переменных для прогнозирования: цена через 7 дней и суммарные продажи за следующие 7 дней.

На листинге A.2 представлена функция создания признаков из скрипта.

Сервис использует систему миграций для управления схемой базы данных PostgreSQL, обеспечивая автоматическое создание необходимых таблиц и индексов при первом запуске или обновлении приложения. Это гарантирует согласованность схемы данных при развертывании сервиса в различных средах.

Важным компонентом сервиса является планировщик, который регулярно запускает процесс обработки данных согласно настраиваемому расписанию. Планировщик обеспечивает актуальность подготовленных данных и автоматизирует весь процесс от получения сырых данных до их преобразования в формат, готовый для машинного обучения.

После обработки данные разделяются на тренировочную и тестовую выборки на основе заданной даты разделения и сохраняются как в файлах CSV для последующего использования в обучении моделей, так и в базе данных PostgreSQL для долговременного хранения и анализа. Сервис обеспечивает целостность данных на всех этапах обработки благодаря тщательно проработанной системе логирования и обработки ошибок.

2.1.4 Разработка модуля машинного обучения

Сервис машинного обучения (ML-service) представляет собой ключевой компонент системы, отвечающий за прогнозирование спроса и цен на товары на основе обработанных данных. Модуль реализован как гибридный сервис, где высокопроизводительный Go-сервер обеспечивает API для взаимодействия с другими компонентами системы, а Python-скрипты выполняют обучение и применение моделей машинного обучения.

Основной функционал сервиса включает две ключевые операции: обучение моделей на исторических данных и генерацию прогнозов для конкретных товаров. Сервис проверяет наличие обученных моделей при запуске и, если они отсутствуют, автоматически запускает процесс обучения, используя подготовленные наборы данных из сервиса обработки.

Go-сервер реализует REST API для взаимодействия с клиентской частью и другими микросервисами. Он обеспечивает валидацию входных данных, маршрутизацию запросов и формирование ответов в формате JSON. Ключевой особенностью реализации является взаимодействие с Python-скриптами через исполнение команд и обработку их вывода, что обеспечивает гибкость при

разработке моделей и использовании популярных библиотек машинного обучения.

Центральным элементом ML-компонента является Python-скрипт, реализующий функциональность для обучения моделей на основе библиотеки LightGBM. Эта библиотека была выбрана благодаря высокой эффективности для задач регрессии с большим количеством категориальных признаков, что характерно для данных о товарах маркетплейса.

Процесс обучения моделей включает несколько ключевых этапов:

- загрузка и валидация тренировочных и тестовых данных;
- подготовка признаков, включая преобразование категориальных переменных;
- обучение двух отдельных моделей: для прогнозирования цен и для прогнозирования продаж;
- сохранение обученных моделей в формате pickle для последующего использования.

Особенностью реализации является оптимизация гиперпараметров моделей LightGBM, включая ограничение глубины деревьев и применение техники ранней остановки (early stopping) для предотвращения переобучения. Модели обучаются независимо друг от друга на одних и тех же признаках, но с разными целевыми переменными.

Модели работают с разнородными данными о товарах, которые включают как числовые, так и категориальные признаки. На листинге А.3 показана структура признаков, используемых для прогнозирования.

Система использует две отдельные модели для прогнозирования: одна предсказывает цены товаров, другая – объемы продаж. На листинге А.4 представлен фрагмент кода, отвечающий за предсказание с использованием обученных моделей.

Сервис обеспечивает возможность как получения прогноза для конкретного товара по запросу, так и пакетного прогнозирования для

множества товаров. Результаты прогнозирования сохраняются в базе данных PostgreSQL для последующего анализа и отображения в пользовательском интерфейсе.

2.1.5 Разработка API-gateway

Сервис API Gateway выступает центральным компонентом системы, обеспечивающим единую точку входа для всех запросов от клиентской части. Основная задача сервиса – маршрутизация запросов к соответствующим микросервисам, аутентификация пользователей и кэширование ответов.

Реализованный на Go с использованием фреймворка Gin, API Gateway перехватывает входящие HTTP-запросы, проверяет JWT-токены через Auth Service и направляет запросы к нужным микросервисам. Важной особенностью реализации является локальное кэширование часто запрашиваемых данных, что значительно снижает нагрузку на другие компоненты системы и улучшает время отклика.

Сервис выполняет несколько ключевых функций:

- проверка и валидация токенов авторизации;
- маршрутизация запросов к соответствующим микросервисам;
- преобразование форматов данных при необходимости;
- локальное кэширование результатов запросов;
- обработка ошибок и формирование унифицированных ответов.

Такой подход централизует логику взаимодействия с клиентским приложением, упрощает разработку и поддержку системы, а также обеспечивает единообразие интерфейсов и повышает безопасность за счет изоляции внутренних сервисов от прямого внешнего доступа.

2.2 Разработка клиентской части системы

Клиентская часть системы (ui-service) разработана как современное одностраничное веб-приложение (SPA) с использованием технологий React и TypeScript. Архитектура клиентской части построена по принципу

компонентного подхода, что обеспечивает переиспользование кода и упрощает поддержку приложения.

Стилизация интерфейса выполнена с использованием подхода utility-first CSS через библиотеку Tailwind CSS, что обеспечивает гибкость оформления компонентов и согласованность дизайна во всем приложении.

При переходе на страницу прогнозирования пользователь имеет возможность получать прогнозы спроса и цен для выбранных товаров. Поддерживается обычный и расширенный режим ввода данных для получения прогнозов. Пример страницы с полученным прогнозом приведен на рисунках 2.2 и 2.3.

Так же для получения истории по уже совершенным прогнозам пользователь может авторизоваться в системе. Пример страницы авторизации приведен на рисунке 2.4. На рисунке 2.5 показана страница истории совершенных запросов.

Система прогнозирования спроса

Прогнозирование | История прогнозов | Выйти

Система прогнозирования спроса

История прогнозов | Обучить модель

Упрощенный режим прогнозирования | Расширенный режим

Название товара * | Регион * | Продавец *

Смартфон Xiaomi 14 Pro | Москва | ИП «Некрасова, Фролов и Кириллова»

Цена | Исходная цена | Уровень запасов

44500

Получить прогноз

Результаты прогнозирования

Основано на данных модели машинного обучения

Прогноз цены 44 943,93 Р Рекомендуемая цена на основе анализа рынка и спроса	Прогноз продаж (неделя) 144 шт. Ожидаемое количество продаж в неделю	Прогноз продаж (день) 20.6 шт. Ожидаемое количество продаж в день
--	--	---

Рисунок 2.2 – Главная страница с получением прогноза в обычном режиме

Система прогнозирования спроса

ПрогнозированиеИстория прогнозовВыйти

Система прогнозирования спроса

История прогнозовОбучить модель

Расширенный режим прогнозированияУпрощенный режим

Название товара *	Бренд *	Категория *
Регион *	Продавец *	Цена *
Исходная цена *	Процент скидки *	Уровень запасов *
Рейтинг покупателя *	Количество отзывов *	Дни доставки *
Выходной день *	Праздничный день *	День недели (0-6) *
Месяц (1-12) *	Квартал (1-4) *	Количество продаж лаг 1 *
Цена лаг 1 *	Количество продаж лаг 3 *	Цена лаг 3 *
Количество продаж лаг 7 *	Цена лаг 7 *	Скользящее среднее продаж 3 *
Скользящее среднее цены 3 *	Скользящее среднее продаж 7 *	Скользящее среднее цены 7 *

Рисунок 2.3 – Главная страница для получения прогнозов в расширенном режиме

Вход в аккаунт

Или зарегистрируйтесь

slashersdcat@gmail.com

.....

Войти

Рисунок 2.4 – страница входа в аккаунт пользователя

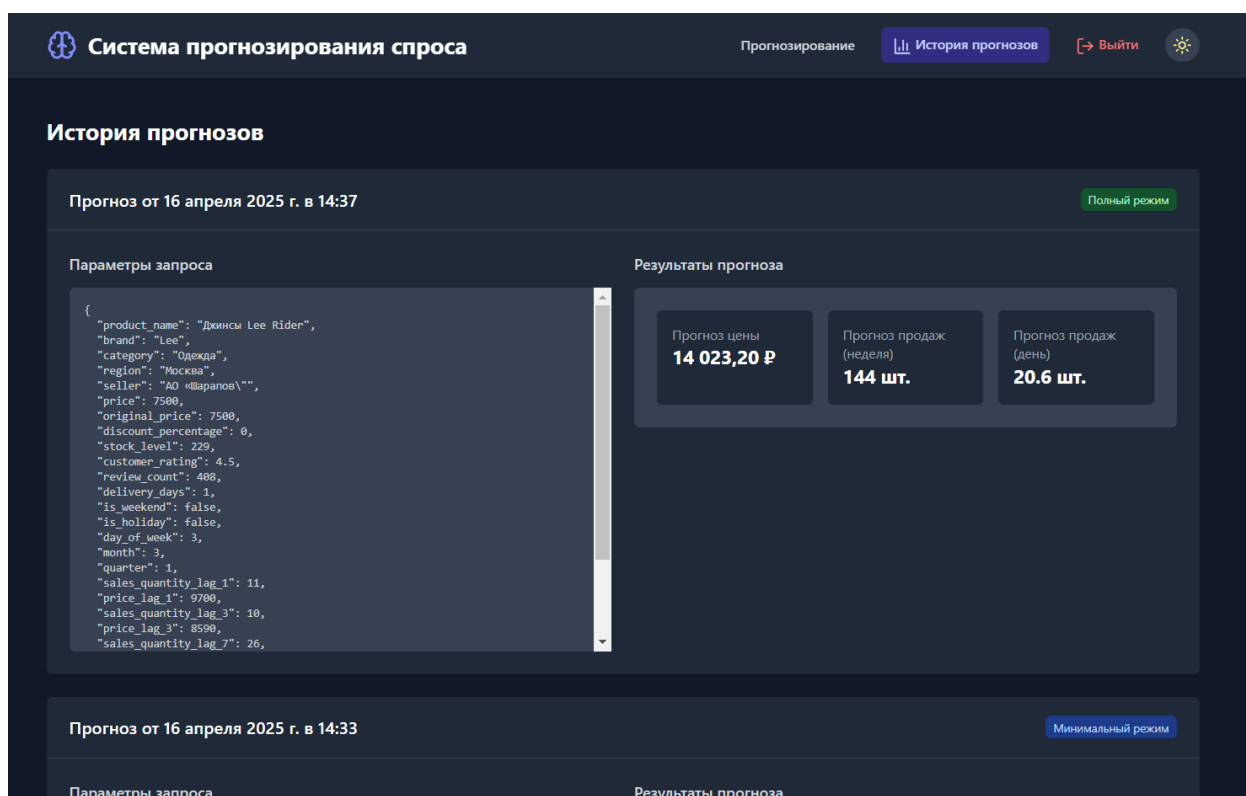


Рисунок 2.5 – Страница истории прогнозов

Приложение реализует адаптивную систему оформления с поддержкой светлой и темной цветовых схем. В навигационном интерфейсе размещен специальный элемент управления, позволяющий вручную переключаться между цветовыми режимами в любой момент использования приложения. Такой подход к организации пользовательского интерфейса не только соответствует современным стандартам веб-разработки, но и значительно повышает комфорт при работе с системой в различных условиях внешнего освещения, снижая нагрузку на зрение пользователя. Пример интерфейса с активированной темной темой представлен на рисунке 2.6.

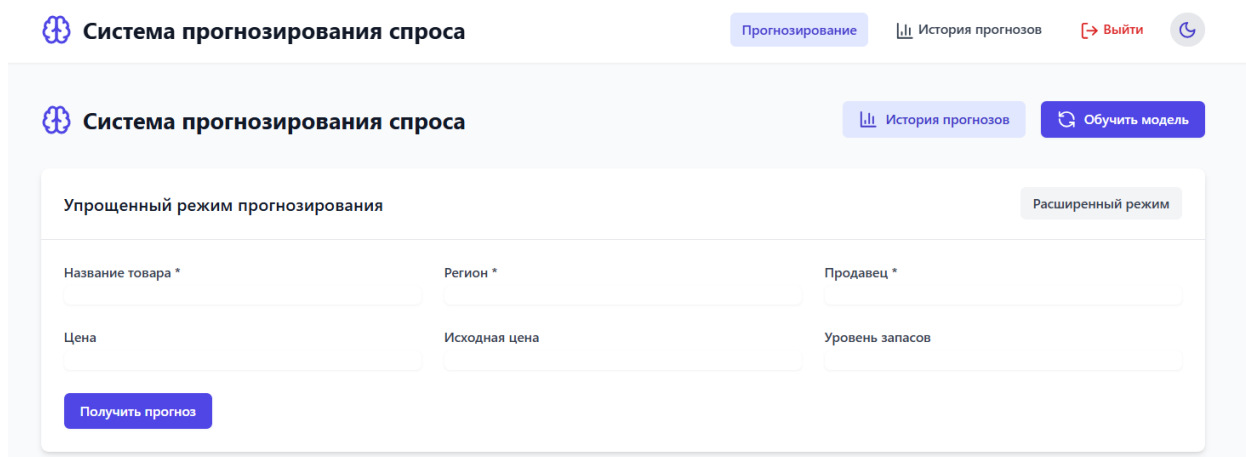


Рисунок 2.6 – Главная страница со светлой темой

2.3 Тестирование системы

Тестирование является критически важным этапом разработки, обеспечивающим корректность работы всех функций системы и соответствие требованиям. Для разработанной интеллектуальной системы оценки спроса было проведено комплексное тестирование как пользовательского интерфейса, так и серверной части с API.

2.3.1 Чек-лист тестирования

Перед началом тестирования был составлен список основных функций веб-приложения, которые необходимо проверить. Для этого был создан чек-лист, в котором перечислены основные тест-кейсы и состояние их выполнения. Тестирование проводилось в браузерах «Google Chrome» и «Mozilla Firefox». Чек-лист представлен в таблице 2.1.

Таблица 2.1 – Чек-лист тестируемого веб-приложения

Тестируемые функции	Google Chrome	Mozilla Firefox
Регистрация и авторизация		
Регистрация пользователя	Выполнено	Выполнено
Авторизация пользователя	Выполнено	Выполнено
Переключение тем оформления	Выполнено	Выполнено
Прогнозирование		
Создание стандартного прогноза	Выполнено	Выполнено
Создание минимального прогноза	Выполнено	Выполнено

Продолжение таблицы 2.1

Визуализация результатов прогноза	Выполнено	Выполнено
Аналитика и отчеты		
Просмотр истории пользователя	Выполнено	Выполнено

2.3.2 Тест-кейсы

Следующим этапом тестирования является создание тест-кейсов. Тест-кейс – это алгоритм действий, по которому предлагается тестировать определенную функцию системы. В тест-кейсах подробно описаны шаги, которые необходимо сделать для подготовки к тесту, сама проверка и ожидаемый результат.

В таблице 2.2 представлены основные тест-кейсы для проверки системы.
Таблица 2.2 Проверка тест-кейсов

Регистрация пользователя		
Шаги выполнения	Ожидаемый результат	Фактический результат
– Вести данные учетной записи – Нажать кнопку «Зарегистрироваться»	– Данные пользователя добавлены в базу данных – Получен доступ к функционалу системы	Фактический результат совпадает с ожидаемым
Авторизация пользователя		
Шаги выполнения	Ожидаемый результат	Фактический результат
– Вести данные учетной записи – Нажать кнопку «Войти»	– Получен доступ к функционалу системы	Фактический результат совпадает с ожидаемым
Переобучение модели прогнозов		
Шаги выполнения	Ожидаемый результат	Фактический результат
– Нажать кнопку «Модель обучена» или «Модель не обучена» (в зависимости от статуса обучения модели) – Дождаться всплывающего уведомления «Модель успешно переобучена»	– Модель переобучена на актуальных данных и сохранена для использования	Фактический результат совпадает с ожидаемым
Шаги выполнения	Ожидаемый результат	Фактический результат
– Нажать кнопку «Модель обучена» – Дождаться всплывающего уведомления «Модель успешно переобучена»	– Модель переобучена на актуальных данных и сохранена для использования	Фактический результат совпадает с ожидаемым

Продолжение таблицы 2.2

Получение прогнозов		
Шаги выполнения	Ожидаемый результат	Фактический результат
<ul style="list-style-type: none"> – На главной странице выбрать режим прогноза – расширенный или обычный – Ввести данные о продукте, на который требуется провести прогнозирования – Нажать кнопку «Получить прогноз» 	<ul style="list-style-type: none"> – Получение прогноза на цену и спрос продукта – Сохранение в базу данных информации о проведенном прогнозе 	Фактический результат совпадает с ожидаемым
Получение истории прогноза		
Шаги выполнения	Ожидаемый результат	Фактический результат
<ul style="list-style-type: none"> – Нажать кнопку «История прогнозов» 	<ul style="list-style-type: none"> – Выгружены и получены все запросы текущего пользователя 	Фактический результат совпадает с ожидаемым

2.3.3 Тестирование API с использованием Swagger

Помимо тестирования пользовательского интерфейса, было проведено тестирование REST API системы с использованием Swagger UI. Этот инструмент позволяет интерактивно взаимодействовать с API без написания кода, что значительно упрощает процесс тестирования.

API Gateway предоставляет документацию Swagger, которая описывает все доступные эндпоинты, методы и параметры. Основные группы API, которые были протестированы:

- Authentication API - регистрация и авторизация пользователей
- Prediction API - создание прогнозов с различными параметрами
- Statistics API - получение статистики по пользовательским прогнозам

На рисунке 2.7 представлен интерфейс Swagger UI для тестирования API.

На рисунке 2.8 представлен пример тестирования API авторизации пользователя в системе.

2.4 Расчёт вычислительной и емкостной сложности

В данном разделе высчитывается вычислительная и емкостная сложности основных алгоритмов интеллектуальной системы оценки спроса на продукт. Для оценки алгоритмической сложности используется нотация Big O, которая описывает асимптотическое поведение функций при увеличении размера входных данных. Вычислительная сложность характеризует время выполнения операции, а емкостная - объем памяти, необходимый для хранения данных и промежуточных результатов.

Результаты расчета вычислительной и емкостной сложности представлены в таблице 2.3.

Таблица 2.3 – Результаты расчета вычислительной и емкостной сложности

Функция	Описание функции	Вычислительная сложность	Емкостная сложность
registerUser	Регистрация пользователя	$O(1)$	$O(1)$
loginUser	Авторизация пользователя	$O(1)$	$O(1)$
validateToken	Проверка JWT-токена	$O(1)$	$O(1)$
createPrediction	Создание прогноза (стандартный)	$O(1)$	$O(1)$
createMinimalPrediction	Создание прогноза (минимальный)	$O(\log n)$	$O(1)$
getUserStatistics	Получение статистики пользователя	$O(n)$	$O(n)$
processDataBatch	Обработка партии данных	$O(n \log n)$	$O(n)$
calculateFeatures	Вычисление признаков	$O(n^2)$	$O(n)$
removeDuplicates	Удаление дубликатов	$O(n \log n)$	$O(n)$
trainModels	Обучение моделей	$O(d \times n \log n)$	$O(d \times n)$
predictPrice	Прогнозирование цены	$O(\log n)$	$O(1)$
predictSales	Прогнозирование продаж	$O(\log n)$	$O(1)$
cacheResponse	Кэширование ответа API	$O(1)$	$O(1)$

Где n - количество записей или элементов данных, d - количество деревьев в ансамбле моделей LightGBM.

Операции API Gateway, такие как `registerUser`, `loginUser` и `validateToken`, имеют константную сложность $O(1)$, так как они включают фиксированное количество операций независимо от размера системы. Кэширование ответов также выполняется за константное время, что значительно улучшает производительность при повторных запросах.

Функции получения списков (`getProductList`, `getUserStatistics`) имеют линейную сложность $O(n)$, поскольку необходимо обработать все n элементов.

Наибольшую вычислительную сложность имеют операции обработки данных и обучения моделей. Функция `calculateFeatures` с квадратичной сложностью $O(n^2)$ выполняет вычисление скользящих средних и лаговых значений для продуктов в разных временных окнах. Обучение моделей (`trainModels`) достигает сложности $O(d \times n \log n)$ из-за алгоритмов построения деревьев решений в `LightGBM`.

При этом операции прогнозирования (`predictPrice`, `predictSales`) имеют логарифмическую сложность $O(\log n)$, так как для предсказания требуется только прохождение по уже построенным деревьям. Это обеспечивает высокую скорость генерации прогнозов в режиме реального времени.

Большинство функций системы имеют линейную или константную емкостную сложность, что позволяет эффективно использовать память. Только функция обучения моделей требует существенных объемов памяти, пропорциональных количеству деревьев и размеру обучающей выборки ($O(d \times n)$).

Проведенный анализ показывает, что система обладает хорошей производительностью для пользовательских операций благодаря оптимальной сложности API-функций и кэширования. Основная вычислительная нагрузка сосредоточена в процессах обработки данных и обучения моделей, которые выполняются асинхронно и по расписанию, не влияя на отзывчивость пользовательского интерфейса.

2.5 Расчет экономической стоимости

Была рассчитана стоимость затрат, потраченных на реализацию системы (таблица 2.4).

Таблица 2.4 – Полная себестоимость проекта

№ п/п	Номенклатура статей расходов	Затраты (руб.)	Доля затрат (%)
1	Сырье и материалы	4140,00	1,2
2	Основная заработная плата	121116,00	35,9
3	Дополнительная заработная плата	24223,20	7,2
4	Страховые взносы	43892,40	13,0
5	Амортизация	22688,90	6,7
6	Прочие расходы	121116,00	35,9
Итого		337176,5	100,0

Итоговая стоимость затрат на реализацию системы составила 337176,5 рублей. Для визуализации долевого состава статей затрат в общей себестоимости представим круговую диаграмму на рисунке 2.9. На данной диаграмме будут отражены затраты по каждой из статей, а также их доля затрат в процентах.

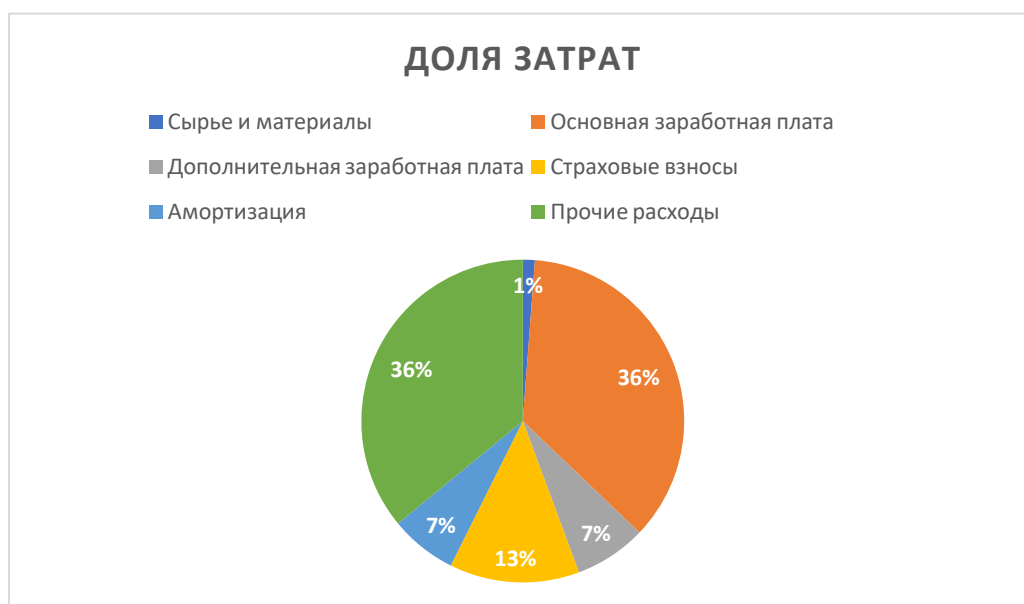


Рисунок 2.9 – Круговая диаграмма долей затрат

Полученные результаты работы будут использоваться внутри университета (организации), поэтому расчет договорной цены не целесообразен.

Вывод по разделу 2

В данном разделе реализована технологическая часть системы оценки спроса, включающая серверную часть на принципах чистой архитектуры и клиентскую часть на React с TypeScript. Разработаны модули авторизации, сбора и подготовки данных, машинного обучения и API Gateway. Проведено тестирование с использованием чек-листов, тест-кейсов и Swagger UI, а анализ сложности подтвердил высокую производительность. Решение соответствует современным стандартам программной инженерии, обеспечивая надежность и масштабируемость.

ЗАКЛЮЧЕНИЕ

В рамках выполнения преддипломной практики проведен детальный анализ предметной области, связанной с разработкой интеллектуальной системы оценки спроса на продукт. Изучены существующие аналоги, такие как Ozon Seller, Moneyplace и MPStats, что позволило выделить их сильные и слабые стороны, а также определить ключевые требования к разрабатываемой системе. На основе проведенного анализа выбраны современные инструменты и технологии разработки: языки программирования Python и Go, фреймворки для реализации микросервисов, базы данных PostgreSQL, а также технология контейнеризации Docker, обеспечивающая изолированную и масштабируемую среду для развертывания приложения.

Спроектирована микросервисная архитектура системы, включающая модули сбора данных, их обработки, прогнозирования спроса с использованием машинного обучения и предоставления аналитики через веб-интерфейс и API. Разработаны функциональная схема в методологии IDEF0 и модель жизненного цикла системы, что обеспечило структурированный подход к проектированию.

В ходе преддипломной практики реализованы ключевые компоненты системы: разработаны и интегрированы модули сбора и обработки данных, а также модуль машинного обучения для прогнозирования спроса. Применение современных технологий, таких как микросервисная архитектура, контейнеризация с использованием Docker и интеграция через REST API, обеспечило гибкость и масштабируемость системы. Результаты практики демонстрируют создание высокоточной и адаптивной интеллектуальной системы, способной эффективно прогнозировать спрос на продукт. Разработанное решение выделяется на фоне аналогов благодаря универсальности источников данных, использованию алгоритмов машинного обучения и простоте интеграции, что делает его ценным инструментом для оптимизации бизнес-процессов в условиях цифровизации.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Gartner. Demand Forecasting with Machine Learning: Benefits and Challenges [Электронный ресурс]. – URL: <https://www.gartner.com/en/insights/demand-forecasting> (дата обращения: 05.05.2025). – Текст: электронный.
2. McKinsey & Company. The Future of Demand Prediction with AI [Электронный ресурс]. – URL: <https://www.mckinsey.com/business-functions/operations/our-insights/ai-driven-demand-forecasting> (дата обращения: 05.05.2025). – Текст: электронный.
3. Порядок проведения государственной итоговой аттестации по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры СМКО МИРЭА 7.5.1/03.П.30-19 [Электронный ресурс]. – URL: <https://www.mirea.ru/docs/177338/> (дата обращения: 08.05.2025). – Текст: электронный.
4. Отчет о научно-исследовательской работе. Структура и правила оформления ГОСТ 7.32-2017 [Электронный ресурс]. – URL: https://cs.msu.ru/sites/cmc/files/docs/2021-11gost_7.32-2022.pdf (дата обращения: 06.05.2025). – Текст: электронный.
5. БИБЛИОГРАФИЧЕСКАЯ ЗАПИСЬ. БИБЛИОГРАФИЧЕСКОЕ ОПИСАНИЕ. Общие требования и правила составления ГОСТ Р 7.0.100-2023 [Электронный ресурс]. – URL: https://www.rsl.ru/photo/!_ORS/5PROFESSIONALAM/7_sibid/ГОСТ_P_7_0_100_2018_1204.pdf (дата обращения: 03.05.2025). – Текст: электронный.
6. Положение о выпускной квалификационной работе студентов, обучающихся по образовательным программам подготовки бакалавров СМКО МИРЭА 7.5.1/03.П.67-21 [Электронный ресурс]. – URL: <https://www.mirea.ru/docs/177322/> (дата обращения: 03.05.2025). – Текст: электронный.

7. Федеральный государственный образовательный стандарт высшего образования - бакалавриат по направлению подготовки 09.03.04 Программная инженерия (ФГОС ВО 3++) [Электронный ресурс]. – URL: <https://fgosvo.ru/news/view/1086> (дата обращения: 03.05.2025). – Текст: электронный.
8. Ozon Seller [Электронный ресурс]. – URL: <https://seller.ozon.ru/> (дата обращения: 10.05.2025). – Текст: электронный.
9. Moneyplace [Электронный ресурс]. – URL: <https://moneyplace.io/> (дата обращения: 10.05.2025). – Текст: электронный.
10. MPStats [Электронный ресурс]. – URL: <https://mpstats.io/> (дата обращения: 10.05.2025). – Текст: электронный.
11. Gin Web Framework [Электронный ресурс]. – URL: <https://gin-gonic.com/docs/> (дата обращения: 11.05.2025). – Текст: электронный.
12. React [Электронный ресурс]. – URL: <https://react.dev/> (дата обращения: 11.05.2025). – Текст: электронный.
13. JWT (JSON Web Token) [Электронный ресурс]. – URL: <https://jwt.io/introduction/> (дата обращения: 09.05.2025). – Текст: электронный.
14. Провос Н., Мазьер Д. Схема паролей, адаптируемая к будущему // Материалы технической конференции USENIX 2022. –2022. – С. 1-9.
15. Гвоздева Т.В., Баллод Б.А. Проектирование информационных систем. Стандартизация: учебное пособие. – Санкт-Петербург: Лань, 2023. – 252 с.

Приложение А

Разработка

Листинг А.1 – Расчет временных признаков для данных

```
func (p *Processor) CalculateTimeFeatures(productData []models.ProductData)
[]models.EnrichedProductData {
    // Группировка данных по продукту
    groupedData := make(map[string][]models.ProductData)
    for _, item := range productData {
        groupedData[item.ProductCode] =
append(groupedData[item.ProductCode], item)}
    var enrichedData []models.EnrichedProductData
    // Обработка каждой группы продуктов
    for _, items := range groupedData {
        // Сортировка по времени
        sort.Slice(items, func(i, j int) bool {
            return items[i].Date.Before(items[j].Date)
        })
        // Расчет лагов и скользящих средних
        for i := range items {
            enriched := models.EnrichedProductData{
                ProductData: items[i],
            }
            // Добавление временных признаков
            enriched.DayOfWeek = items[i].Date.Weekday().String()
            enriched.Month = items[i].Date.Month().String()
            enriched.IsWeekend = items[i].Date.Weekday() == time.Saturday ||
items[i].Date.Weekday() == time.Sunday
            // Расчет лагов для не первых элементов
            if i > 0 {
                enriched.PriceLag1 = items[i-1].Price
                enriched.SalesQuantityLag1 = items[i-1].SalesQuantity}
            // Расчет скользящих средних
            if i >= 3 {
                var priceSum, salesSum float64
                for j := i-3; j < i; j++ {
                    priceSum += items[j].Price
                    salesSum += float64(items[j].SalesQuantity)
                }
                enriched.PriceRollingMean3 = priceSum / 3
                enriched.SalesQuantityRollingMean3 = salesSum / 3
            }
        }
    }
}
```

Продолжение листинга А.1

```
        enrichedData = append(enrichedData, enriched)
    }
}
return enrichedData
}
```

Листинг А.2 – Создание временных признаков для данных

```
def create_features(df):
    """Создание признаков для модели."""
    logger.info("Creating features")
    # Добавление временных признаков
    df['day_of_week'] = df['date'].dt.dayofweek
    df['month'] = df['date'].dt.month
    df['quarter'] = df['date'].dt.quarter
    # Сортировка данных
    df = df.sort_values(['product_name', 'date'])
    # Лаги и скользящие средние
    lag_periods = [1, 3, 7]
    rolling_periods = [3, 7]
    for product in df['product_name'].unique():
        product_df = df[df['product_name'] == product]
        if len(product_df) < 7: # Уменьшен порог до 7 записей
            continue
        # Лаги
        for lag in lag_periods:
            df.loc[df['product_name'] == product,
f'sales_quantity_lag_{lag}'] = product_df['sales_quantity'].shift(lag)
            df.loc[df['product_name'] == product, f'price_lag_{lag}'] =
product_df['price'].shift(lag)
        # Скользящие средние
        for window in rolling_periods:
            df.loc[df['product_name'] == product,
f'sales_quantity_rolling_mean_{window}'] = (
product_df['sales_quantity'].rolling(window=window).mean().values
            )
            df.loc[df['product_name'] == product,
f'price_rolling_mean_{window}'] = (
                product_df['price'].rolling(window=window).mean().values
            )
```

Листинг А.3 – Структура признаков, используемых для прогнозирования

```
def _prepare_features(self, df: pd.DataFrame):
    # Категориальные признаки
    categorical_features = ['brand', 'region', 'category', 'seller',
                           'day_of_week', 'month', 'quarter']

    # Числовые признаки
    numerical_features = [
        'price', 'original_price', 'discount_percentage', 'stock_level',
        'customer_rating', 'review_count', 'delivery_days', 'is_weekend',
        'is_holiday', 'sales_quantity_lag_1', 'price_lag_1',
        'sales_quantity_lag_3', 'price_lag_3', 'sales_quantity_lag_7',
        'price_lag_7', 'sales_quantity_rolling_mean_3',
        'price_rolling_mean_3',
        'sales_quantity_rolling_mean_7', 'price_rolling_mean_7'
    ]

    # Объединение признаков
    feature_names = numerical_features + categorical_features
    # Преобразование категориальных признаков
    for cat_feat in categorical_features:
        if cat_feat in df.columns:
            df[cat_feat] = df[cat_feat].astype('category')

    return df[feature_names], feature_names, categorical_features
```

Листинг А.4 – Функция прогнозирования

```
def predict(self, product_data: Dict[str, Any]) -> Dict[str, float]:
    """
    Make predictions for a product
    """
    if self.price_model is None or self.sales_model is None:
        if not self.load_models():
            raise ValueError("Models not trained or loaded properly")

    # Преобразование данных в DataFrame
    df = pd.DataFrame([product_data])

    # Подготовка признаков для прогнозирования
    for cat_feat in self.categorical_features:
```

Продолжение листинга А.4

```
        if cat_feat in df.columns:
            df[cat_feat] = df[cat_feat].astype('category')

    # Прогнозирование
    X = df[self.feature_names]
    price_pred = self.price_model.predict(X)[0]
    sales_pred = self.sales_model.predict(X)[0]

    return {
        "predicted_price": float(price_pred),
        "predicted_sales": float(sales_pred)
    }
```