

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., Проектная 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) – 38 стр., 27 илл., 1 табл., 0 листинг., 0 формул, 18 ист. лит (в т.ч. 11 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., Project 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) - 38 p., 27 illustrations, 1 tables, 0 listings, 0 formulas, 18 references (including 11 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.pdf», исполнитель Мухаметшин А.Р.  
© Мухаметшин А. Р.

**СОДЕРЖАНИЕ**

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 6](#_Toc1)

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 7](#_Toc2)

[ВВЕДЕНИЕ 8](#_Toc3)

[1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ 10](#_Toc4)

[1.1 Анализ существующих решений 10](#_Toc5)

[1.2 Определение требований к решению 13](#_Toc11)

[1.3 Выбор инструментов и методов создания информационной системы 16](#_Toc14)

[1.4 Постановка задачи к проектированию и разработке информационной системы 18](#_Toc21)

[Вывод по разделу 1 19](#_Toc22)

[2 ПРОЕКТНЫЙ РАЗДЕЛ 20](#_Toc23)

[2.1 Проектирование функциональной схемы 20](#_Toc24)

[2.2 Проектирование архитектуры информационной системы 21](#_Toc25)

[2.3 Проектирование клиентской части информационной системы 22](#_Toc26)

[2.4 Проектирование серверной части информационной системы 24](#_Toc27)

[2.5 Разработка диаграмм логической модели системы 25](#_Toc28)

[2.6 Проектирование жизненного цикла информационной системы 28](#_Toc29)

[2.7 Проектирование схемы базы данных 29](#_Toc30)

[Вывод по разделу 2 32](#_Toc31)

[ЗАКЛЮЧЕНИЕ 33](#_Toc32)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 34](#_Toc33)

# 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. |
| 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) |
| CRM | – | Customer Relationship Management (Управление взаимоотношениями с клиентами) |
| ERP | – | Enterprise Resource Planning (Планирование ресурсов предприятия) |
| ML | – | Machine Learning (Машинное обучение) |
| ИС | – | Информационная система |
| ИТ | – | Информационные технологии |
| ПО | – | Программное обеспечение |

**ВВЕДЕНИЕ**

Целью проектной практики является разработка интеллектуальной системы (ИС) для автоматизированной оценки спроса на продукт, которая позволит бизнесу прогнозировать рыночные потребности на основе анализа данных из разнородных источников с использованием технологий машинного обучения (ML). Система направлена на обеспечение компаний инструментом для повышения конкурентоспособности за счет точного предсказания спроса и оптимизации бизнес-процессов.

Актуальность разработки обусловлена стремительными изменениями рыночных условий, в которых традиционные методы оценки спроса, основанные на экспертных суждениях или ручном анализе данных, становятся недостаточно эффективными. Современные компании сталкиваются с необходимостью оперативной адаптации к динамике рынка, что требует автоматизации процессов сбора, обработки и анализа данных. 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 демонстрирует рост точности предсказаний до 85% в сравнении с традиционными подходами [2]. В условиях цифровизации бизнеса разработка таких систем приобретает стратегическое значение.

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. Это обеспечивает высокую адаптивность системы 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 выносится 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. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ
   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.

* + 1. Ozon Seller

Ozon Seller [8] – это инструмент аналитики, разработанный для продавцов на платформе Ozon, одной из крупнейших российских торговых онлайн-площадок. Платформа предоставляет пользователям доступ к статистике продаж, прогнозам спроса и аналитике конкурентной среды. Веб-интерфейс приложения позволяет визуализировать ключевые показатели, такие как объемы продаж по категориям товаров и динамика спроса в определенные периоды. Пример веб-интерфейса представлен на рисунке 1.1.

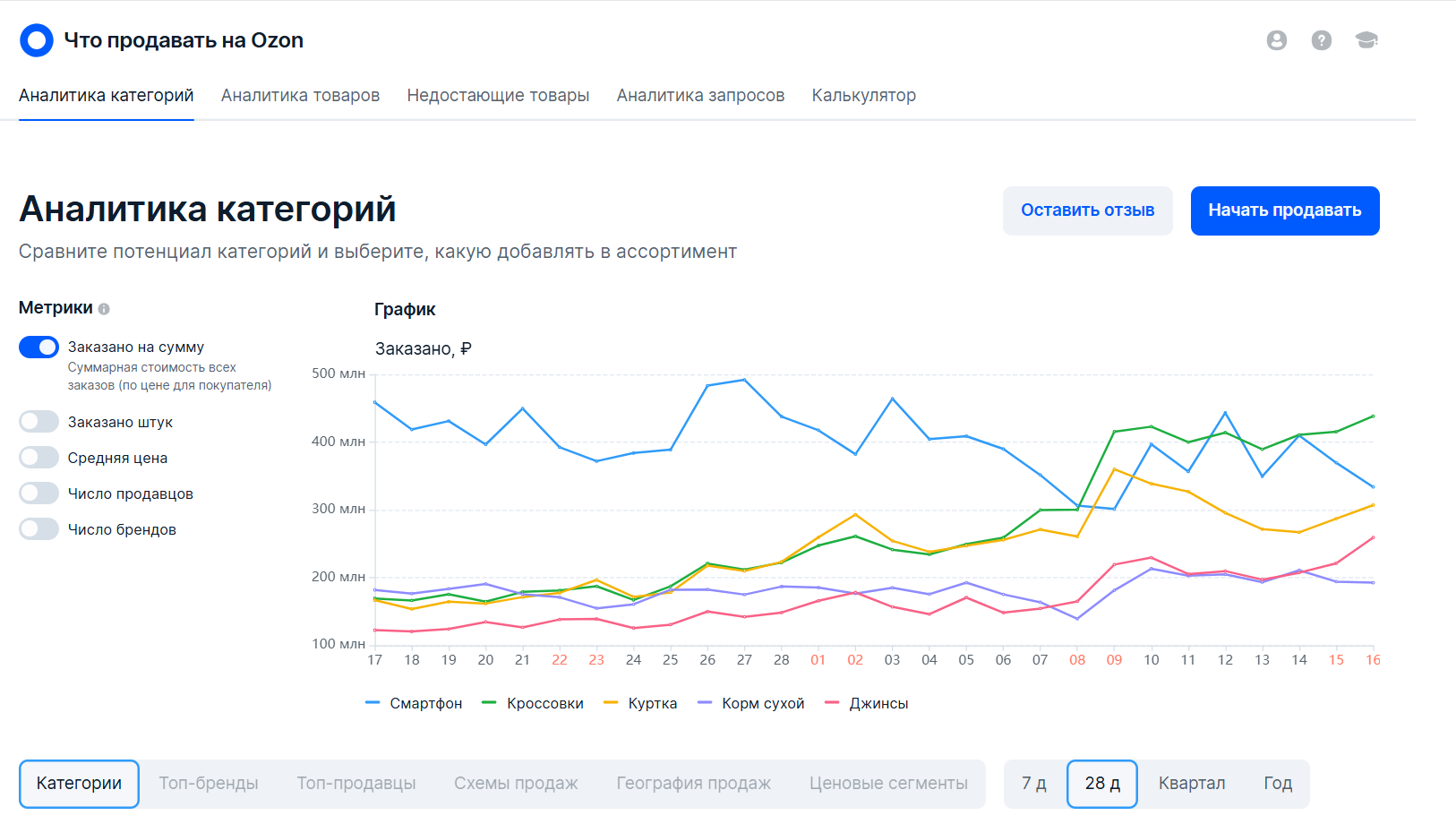


Рисунок 1.1 – Веб-интерфейс платформы Ozon Seller

* + 1. Moneyplace

Moneyplace [9] – это российская аналитическая платформа, предназначенная для продавцов маркетплейсов, включая Ozon, Wildberries и Яндекс.Маркет. Сервис предоставляет инструменты для анализа спроса, конкурентной среды и управления ассортиментом. Веб-интерфейс платформы ориентирован на визуализацию данных о продажах и прогнозах, что помогает пользователям принимать обоснованные решения о закупках и ценообразовании. Пример интерфейса показан на рисунке 1.2.

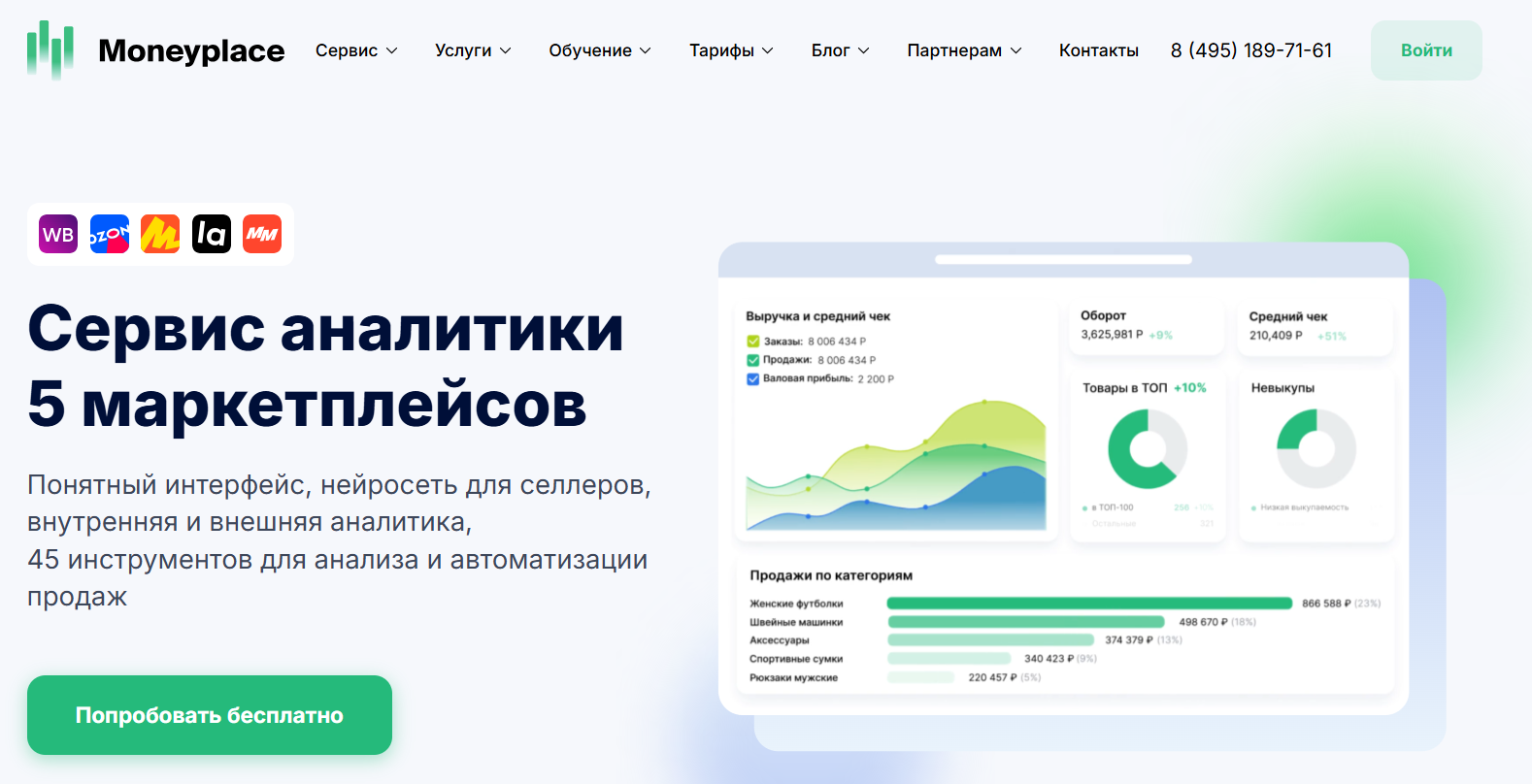


Рисунок 1.2 – Веб-интерфейс платформы 1.1.2 Moneyplace

* + 1. MPStats

MPStats [10] – это аналитическая платформа, предназначенная для продавцов на маркетплейсах Wildberries, Ozon и Яндекс.Маркет. Сервис предоставляет инструменты для анализа продаж, конкурентов, оптимизации рекламных кампаний и исследования товаров. Веб-интерфейс платформы позволяет пользователям получать доступ к детальным отчетам о продажах, выручке, ценах, рейтингах и других ключевых показателях. Кроме того, MPStats предлагает расширение для браузера Chrome, которое упрощает доступ к аналитическим данным непосредственно на страницах маркетплейсов. Пример интерфейса показан на рисунке 1.3.

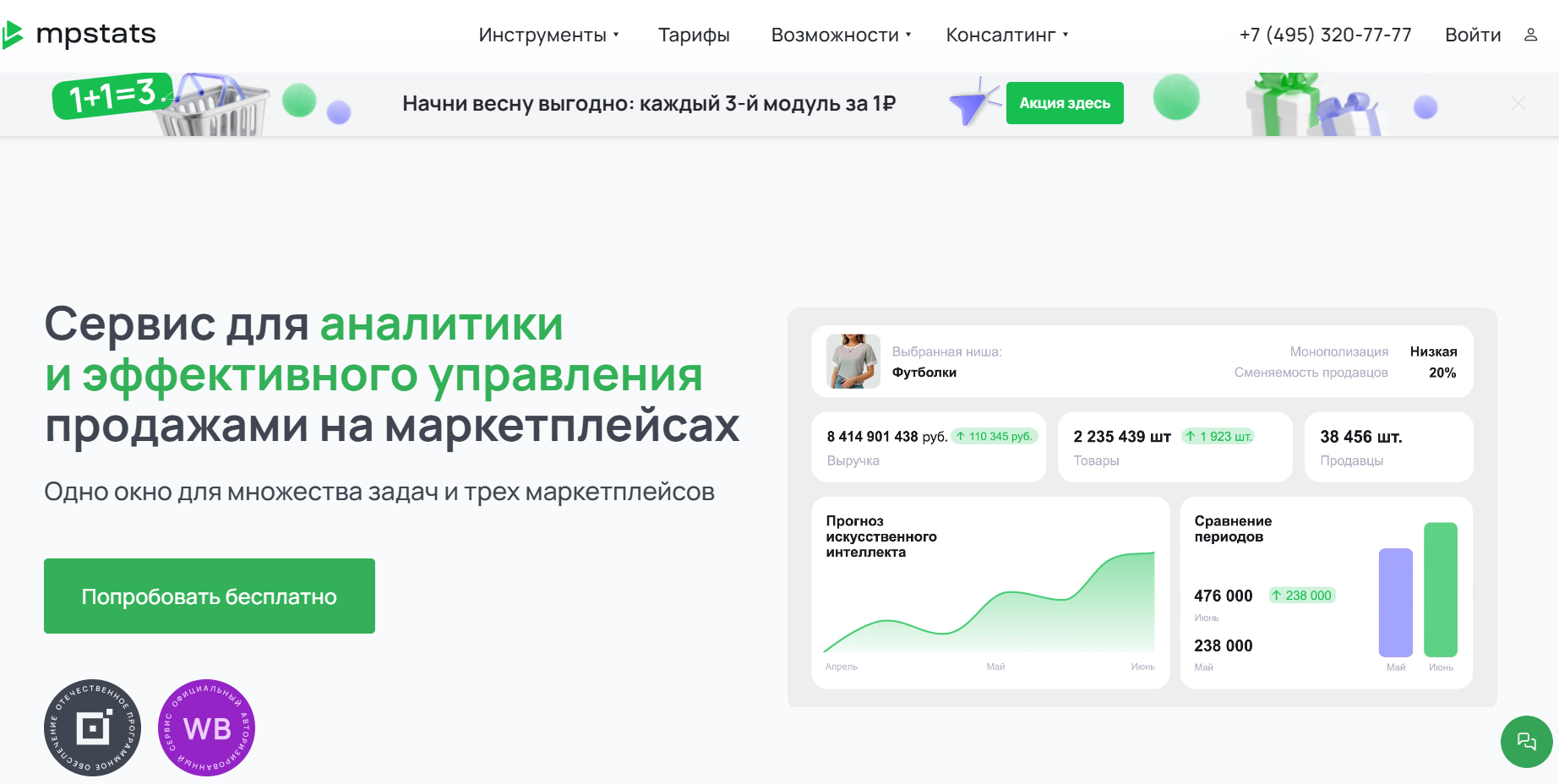


Рисунок 1.3 – Веб-интерфейс платформы MPStats

* + 1. Результаты сравнительного анализа

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 |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| 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 и соответствовать следующим требованиям:

* гибкость настройки: Возможность подключения различных источников данных (внутренние системы, API, веб-скрапинг) и адаптации под нужды конкретного бизнеса;
* интеграция: Предоставление REST API для легкой интеграции с существующими системами компаний и экспорта данных в удобных форматах (JSON, PDF, Excel);
* аналитика: Использование ML-моделей для глубокого анализа и точного прогнозирования спроса, а также визуализация результатов через веб-интерфейс.

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

* 1. Определение требований к решению
     1. Формулирование требований к программной части системы

**Функциональные требования**

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

Необходимо обеспечить:

* сбор данных из внутренних систем (CRM, ERP), 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 [16], 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. Формулирование требований к аппаратной части системы

**Требования к серверной части системы**

Для серверной части, включающей микросервисы для сбора данных, обработки запросов и маршрутизации через API Gateway, выдвигаются следующие минимальные требования к аппаратному обеспечению:

* процессор: 64-битный, с минимальной частотой 2.0 ГГц, 2 ядра (например, Intel Core i3 или аналогичный);
* оперативная память: 4 ГБ для обеспечения одновременной работы нескольких микросервисов и брокера сообщений;
* пропускная способность сетевого соединения: 100 Мбит/с для быстрой передачи данных между сервисами и внешними источниками;
* хранилище: 50 ГБ SSD для хранения временных данных и логов.

Требования к оперативной памяти обусловлены необходимостью параллельной обработки запросов от множества пользователей и асинхронного взаимодействия между микросервисами через брокер сообщений. Высокая пропускная способность сети необходима для оперативного получения данных из внешних API и передачи их в систему для последующей обработки.

**Требования к модулям обработки данных и машинного обучения**

Для микросервисов, отвечающих за предобработку данных и выполнение прогнозов спроса с использованием моделей машинного обучения (ML Service), выдвигаются более строгие требования:

* процессор: 64-битный с поддержкой аппаратной виртуализации (Intel VT-x или AMD-V), минимум 4 ядра (например, Intel Core i5 или аналогичный);
* оперативная память: 8 ГБ (рекомендуется 16 ГБ для работы с большими наборами данных и сложными моделями);
* пропускная способность сетевого соединения: 100 Мбит/с для передачи данных между сервисами и базой данных;
* графический процессор (GPU): 2 ГБ видеопамяти (например, NVIDIA GTX 1050 или аналогичный) или 8 CPU ядер при отсутствии GPU;
* хранилище: 100 ГБ SSD для хранения обработанных данных, моделей и временных файлов.

Необходимость в увеличенной оперативной памяти и GPU обусловлена высокими вычислительными нагрузками при обучении и использовании ML-моделей, таких как регрессии или нейронные сети. GPU значительно ускоряет выполнение операций с тензорами, что критично для работы с большими объемами данных в реальном времени. При отсутствии GPU нагрузка перекладывается на CPU, что требует дополнительных ядер для поддержания производительности.

* 1. Выбор инструментов и методов создания информационной системы
     1. Выбор технологий для серверной части

Для серверной части системы выбраны следующие технологии:

* Go (Gin [13]): Высокопроизводительный язык программирования с фреймворком Gin для создания микросервисов. Go обеспечивает быструю обработку запросов, низкое потребление ресурсов и простоту работы с конкурентными задачами благодаря встроенным горутинам. Это особенно важно для микросервисов, таких как Data Collector и API Gateway, которые обрабатывают большие объемы входящих данных;
* gRPC и Protobuf: Используются для быстрого и эффективного взаимодействия между микросервисами. gRPC, основанный на Protocol Buffers, обеспечивает низкую задержку и высокую пропускную способность по сравнению с REST, что критично для передачи данных между сервисами в реальном времени;
* RabbitMQ: Брокер сообщений для асинхронной коммуникации между микросервисами. RabbitMQ позволяет декомпозировать процессы сбора, обработки и анализа данных, обеспечивая устойчивость системы при высоких нагрузках.

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

* + 1. Выбор технологий для клиентской части

Клиентская часть системы реализована с использованием следующих технологий:

* React [15] и TypeScript: React.js – библиотека JavaScript для построения динамических и адаптивных пользовательских интерфейсов с применением компонентного подхода;
* Tailwind CSS: Фреймворк для быстрого создания стилизованных интерфейсов. Tailwind предоставляет набор готовых классов, что ускоряет разработку и обеспечивает гибкость настройки дизайна под нужды пользователей.

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

* + 1. Выбор СУБД

Для управления данными выбраны две системы:

* PostgreSQL: Основная реляционная база данных для хранения структурированных данных (пользователи, обработанные данные, прогнозы). PostgreSQL обеспечивает высокую производительность, надежность и поддержку сложных запросов, что делает её подходящей для аналитических задач;
* Redis: Используется как in-memory кэш для ускорения доступа к часто запрашиваемым данным, таким как последние прогнозы или аналитические сводки. Redis снижает нагрузку на PostgreSQL и улучшает время отклика серверной части.

Комбинация PostgreSQL и Redis обеспечивает баланс между долговременным хранением данных и быстрым доступом к ним, что критично для системы с высокой частотой запросов.

* + 1. Выбор технологий для создания ML-модуля

Модуль машинного обучения реализован с использованием:

* Python (scikit-learn, TensorFlow [11,12]): Python выбран как основной язык для разработки ML-моделей благодаря богатому набору библиотек. Scikit-learn используется для базовых алгоритмов (например, линейной регрессии), а TensorFlow – для более сложных моделей, таких как нейронные сети для анализа временных рядов;
* ONNX [17]: Формат для сохранения обученных моделей и их интеграции с Go через библиотеку onnxruntime-go. Это позволяет выполнять прогнозы непосредственно в микросервисе ML Service, минимизируя задержки.

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

* + 1. Выбор средств развертывания веб-приложения

Для удобного развертывания и масштабирования системы используется Docker. Этот инструмент позволяет запускать приложение в изолированных контейнерах, не требуя сложных настроек окружения, что значительно упрощает развертывание и масштабирование.

* + 1. Выбор среды разработки и вспомогательные инструменты

Для разработки системы выбраны следующие инструменты:

* GoLand: Интегрированная среда разработки (IDE) для Go, предоставляющая удобный интерфейс, автодополнение кода и встроенные плагины для работы с микросервисами и gRPC. Это ускоряет процесс написания серверной части;
* Visual Studio Code (VS Code): Легковесная IDE для разработки клиентской части на React и TypeScript. Поддержка расширений (например, для Tailwind CSS) делает её универсальным инструментом для фронтенда;
* PgAdmin: Графический интерфейс для администрирования PostgreSQL, упрощающий управление базой данных и выполнение запросов.

Эти инструменты выбраны за их популярность, функциональность и совместимость с технологическим стеком проекта.

* 1. Постановка задачи к проектированию и разработке информационной системы

Целью проектной практики является создание интеллектуальной системы, позволяющей пользователям анализировать рынок и эффективно прогнозировать спрос на товары.

Для реализации поставленной цели выделены следующие задачи к проектированию и разработке:

* спроектировать информационную систему (архитектуру и взаимодействие элементов системы);
* определить и обосновать информационные, технические, программные средства для разработки веб-приложения;
* разработать серверную и клиентскую части информационной системы, предоставляющей пользователям заявленный функционал;
* произвести тестирование разработанной информационной системы.

## **Вывод по разделу 1**

В данном разделе проведен детальный 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. Поставлены задачи к проектированию и разработке, которые охватывают создание архитектуры, реализацию системы и её тестирование. Этот подход обеспечивает прочную основу для дальнейшей работы над проектом, гарантируя соответствие системы современным стандартам программной инженерии.

1. ПРОЕКТНЫЙ РАЗДЕЛ
   1. Проектирование функциональной схемы

Для разработки информационной системы, предназначенной для автоматизированной оценки спроса на продукт, необходимо было определить и формализовать ключевые бизнес-процессы, обеспечивающие её функционирование. С этой целью была спроектирована функциональная схема в методологии IDEF0 [14], которая позволяет структурировать процессы создания системы, отражая их взаимосвязи, входные данные, управляющие факторы и выходные результаты.

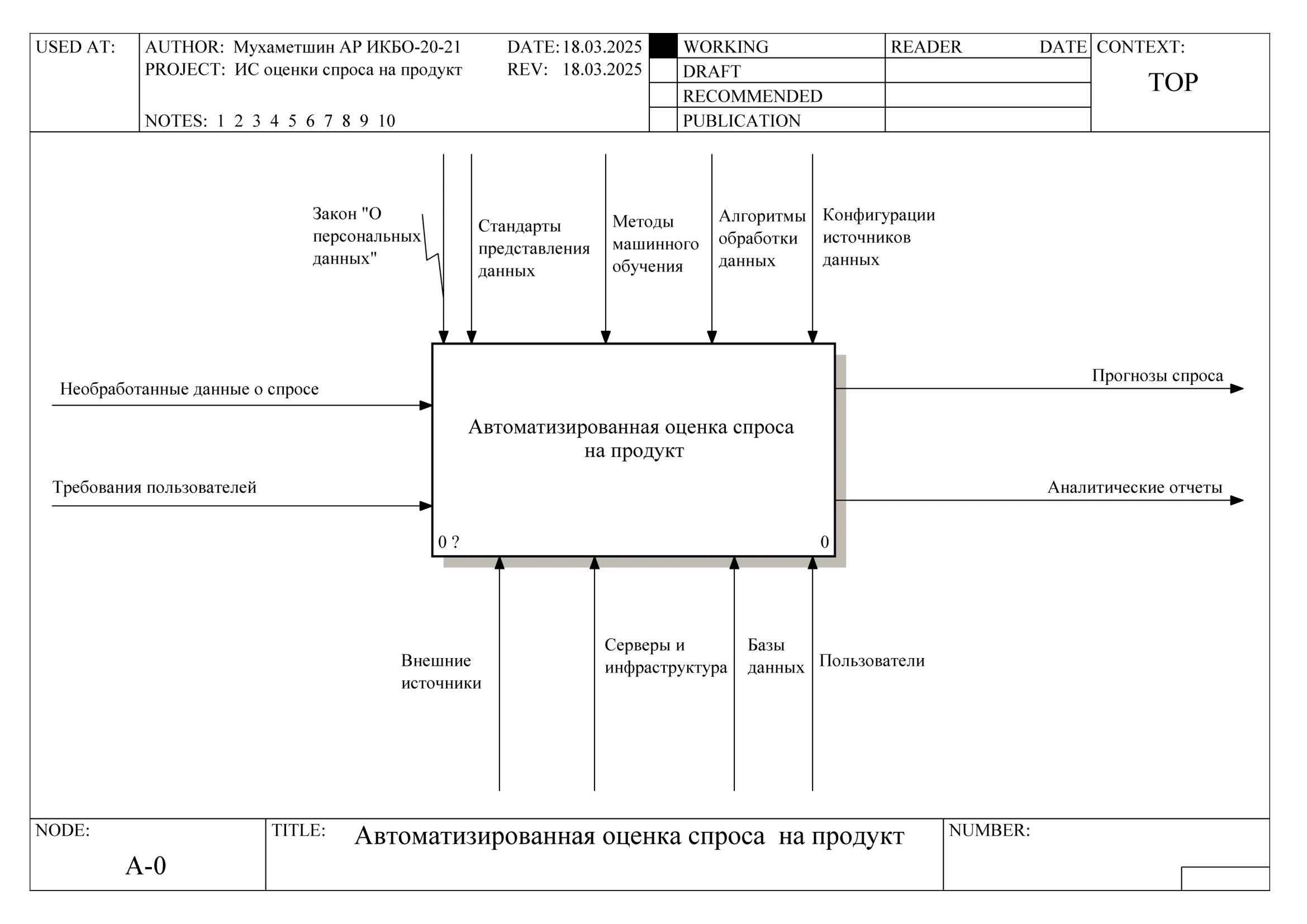


Рисунок 2.1 - Контекстная диаграмма функциональной схемы

Контекстная диаграмма, представленная на рисунке 2.1, демонстрирует взаимодействие системы с внешней средой: источниками данных (CRM, ERP, внешние API, веб-скрапинг), пользователями (бизнес-аналитиками) и инфраструктурой (серверы, базы данных). Входными данными выступают необработанные данные о спросе и требования пользователей, управляющими факторами – стандарты разработки и конфигурации источников, а выходными результатами – прогнозы спроса и аналитические отчёты.

На декомпозиции контекстной диаграммы, изображённой на рисунке 2.2, раскрываются основные функции системы: сбор данных, их обработку, обучение моделей машинного обучения, генерацию прогнозов и предоставление результатов через веб-интерфейс и API.

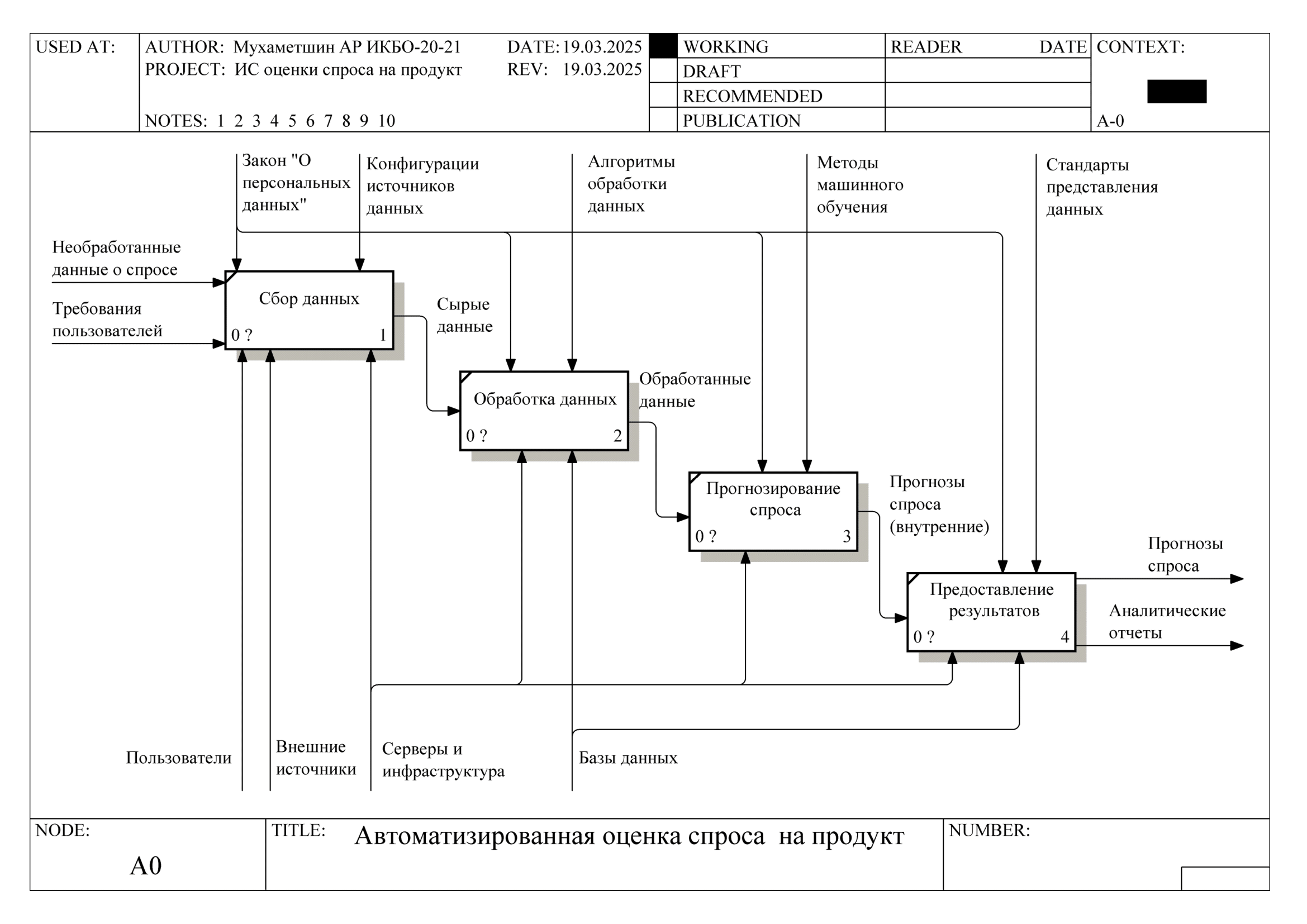


Рисунок 2.2 – Декомпозиция функциональной схемы

* 1. Проектирование архитектуры информационной системы

Для разработки интеллектуальной системы автоматизированной оценки спроса на продукт была выбрана микросервисная архитектура, реализованная в рамках трехуровневой модели. Такой подход обусловлен необходимостью разделения функциональных компонентов системы – сбора данных, их обработки, прогнозирования спроса с использованием машинного обучения, аутентификации и аналитики – на независимые модули, что обеспечивает гибкость, масштабируемость, отказоустойчивость и эффективность работы.

Архитектура включает три уровня: клиентский, серверный и уровень данных. Используется паттерн API Gateway, реализованный в виде отдельного сервиса, который принимает входящие запросы от клиентского приложения, выполняет их аутентификацию и маршрутизацию к соответствующим микросервисам. Этот подход упрощает взаимодействие между клиентской частью и серверными компонентами, а также повышает безопасность системы за счёт централизованной обработки запросов.

Для хранения данных используется PostgreSQL как основная реляционная база данных, обеспечивающая надежность и поддержку сложных запросов. Redis применяется как кэш для ускорения доступа к часто запрашиваемым данным, таким как последние прогнозы или аналитические сводки. Каждый микросервис имеет собственные таблицы или схемы в PostgreSQL, что минимизирует зависимости между компонентами и упрощает масштабирование.

* 1. Проектирование клиентской части информационной системы

Клиентский уровень реализован через веб-приложение на React с TypeScript и Tailwind CSS, предоставляющее интерфейс для управления источниками данных, просмотра прогнозов и аналитики, а также экспорта результатов в PDF и Excel. Взаимодействие с сервером осуществляется через REST API, предоставляемый API Gateway, который аутентифицирует запросы и направляет их к нужным микросервисам.

React представляет собой библиотеку JavaScript для создания пользовательских интерфейсов. Он реализует компонентный подход, позволяя использовать шаблонные компоненты и обновлять интерфейс без необходимости полной перезагрузки страницы. Это повышает производительность, удобство работы и снижает время отклика и затраты на разработку. TypeScript добавляет статическую типизацию, улучшая качество кода и упрощая его поддержку в процессе разработки системы прогнозирования спроса.

Tailwind CSS – это CSS-фреймворк, который позволяет быстро и гибко создавать интерфейсы. Благодаря утилитарному подходу разработка интерфейсов становится удобной и легко адаптируется под нужды проекта, обеспечивая визуально привлекательный и функциональный дизайн для отображения аналитических данных и прогнозов.

Использование этих технологий обеспечивает высокую производительность клиентской части, удобный и адаптивный интерфейс, а также эффективное взаимодействие с серверной частью системы, что критично для оперативного управления данными и получения результатов прогнозирования.

Схема связи страниц клиентской стороны ИС представлена на рисунке 2.3.



Рисунок 2.3 – Схема связи страниц клиентской части ИС

* 1. Проектирование серверной части информационной системы

Серверный уровень состоит из набора микросервисов, каждый из которых выполняет специализированную задачу и взаимодействует с другими через асинхронные сообщения (с использованием RabbitMQ) или синхронные вызовы (через gRPC). Это обеспечивает устойчивость системы к сбоям: если один сервис временно недоступен, остальные продолжают функционировать, обрабатывая накопленные сообщения из очереди.

Хотя традиционный архитектурный паттерн MVC (Model-View-Controller) часто используется для структурирования серверной логики, в контексте микросервисной архитектуры он адаптирован под модульный подход. Каждый микросервис самостоятельно управляет своей бизнес-логикой и данными, что заменяет классическое разделение на модели, представления и контроллеры. Вместо этого используется связка «Controller-Service-Repository», где:

* Controller отвечает за обработку входящих запросов через API-эндпоинты, предоставляемые каждым микросервисом;
* Service реализует бизнес-логику, например, обработку данных или вызов моделей машинного обучения;
* Repository обеспечивает взаимодействие с базой данных (PostgreSQL) через объектно-ориентированный доступ, минимизируя написание сложных SQL-запросов.

Основные микросервисы системы включают:

* Data Collector Service: Отвечает за сбор данных из различных источников – внутренних систем (CRM, ERP), внешних API (например, Google Trends) и веб-скрапинга маркетплейсов. Сервис извлекает данные периодически или по запросу, обрабатывает ошибки (например, недоступность источников) и отправляет сырые данные в очередь RabbitMQ для дальнейшей обработки.
* Data Processor Service: Занимается очисткой, нормализацией и обогащением данных, полученных через RabbitMQ. Устраняет дубликаты, приводит данные к единому формату и сохраняет их в PostgreSQL, подготавливая для аналитики и машинного обучения.
* ML Service: Реализует логику прогнозирования спроса. Извлекает данные из PostgreSQL, обучает модели машинного обучения, сохраняет их в формате ONNX и генерирует прогнозы по запросам от API Gateway. Результаты возвращаются в JSON и сохраняются в базе данных.
* Auth Service: Управляет аутентификацией и авторизацией. Генерирует JWT-токены на основе учетных данных из PostgreSQL и поддерживает интеграцию с SSO через OAuth2. Токены проверяются через API Gateway для доступа к другим сервисам.
* Analytics Service: Отвечает за агрегацию данных и подготовку отчетов. Выполняет SQL-запросы к PostgreSQL, рассчитывает метрики (например, средний спрос по категориям) и генерирует визуализации или файлы (PDF/Excel) для передачи на фронтенд через API Gateway.
* API Gateway: Выступает единой точкой входа, проверяет JWT-токены через Auth Service, маршрутизирует запросы к микросервисам и кэширует ответы в Redis для повышения производительности.

Этот подход обеспечивает разделение ответственности, упрощает тестирование и масштабирование, что идеально подходит для микросервисной архитектуры системы прогнозирования спроса.

* 1. Разработка диаграмм логической модели системы

Для лучшего понимания процессов необходимых для функционирования системы и её структуры были созданы несколько диаграмм.

Диаграмма последовательности, показана на рисунке 2.4. Она демонстрирует процесс обработки запроса на прогноз спроса – от отправки клиентом запроса через API Gateway до получения результата. Диаграмма детализирует последовательность вызовов между компонентами, включая аутентификацию, обработку данных и генерацию прогноза.

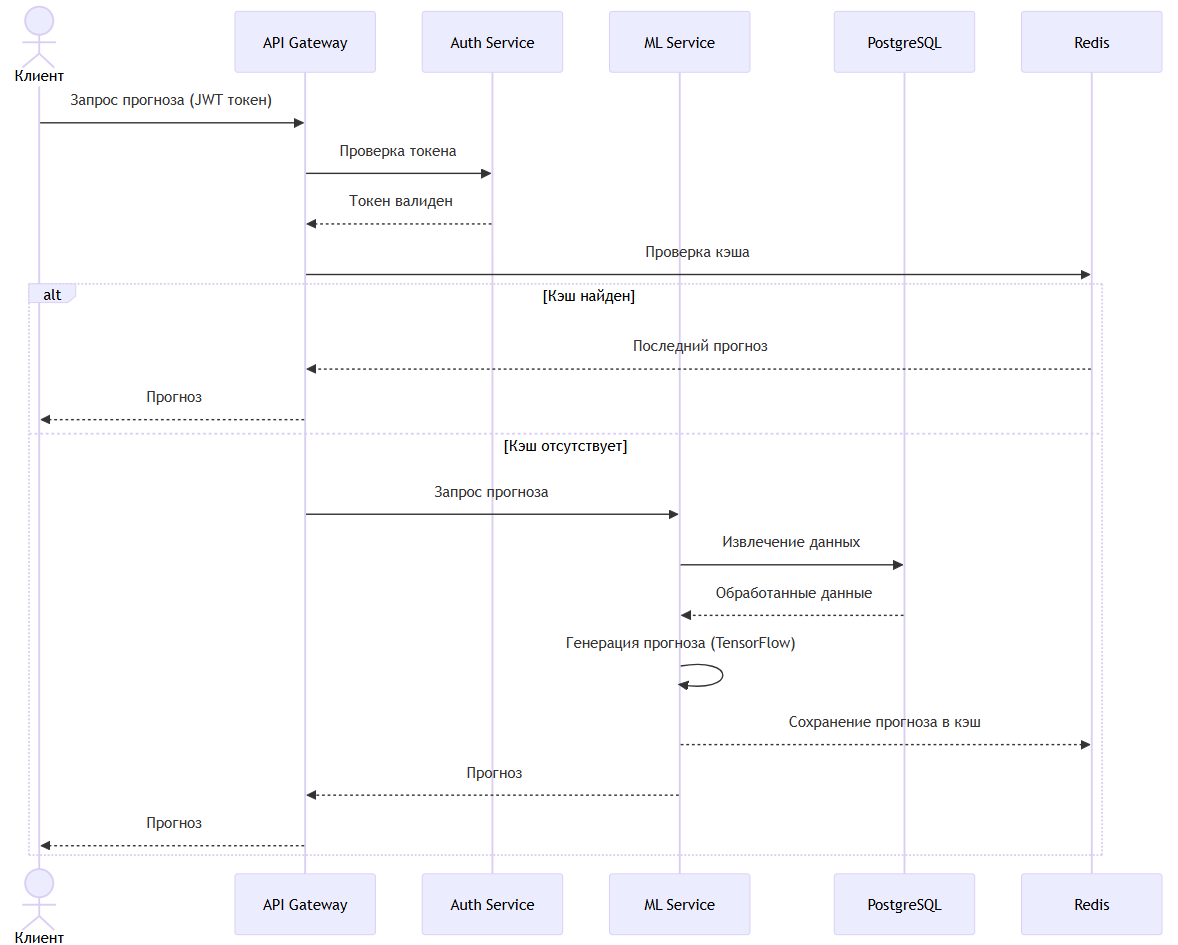


Рисунок 2.4 – Диаграмма последовательности

Диаграмма компонентов представлена на рисунке 2.5 с использованием нотации Mermaid. Она иллюстрирует общую архитектуру, показывая связи между клиентским приложением, API Gateway, микросервисами и хранилищами данных (PostgreSQL и Redis). Диаграмма подчеркивает модульность системы и каналы взаимодействия.

Эти визуализации помогают понять архитектурный дизайн, облегчают разработку и обеспечивают единое видение системы для всех участников проекта.

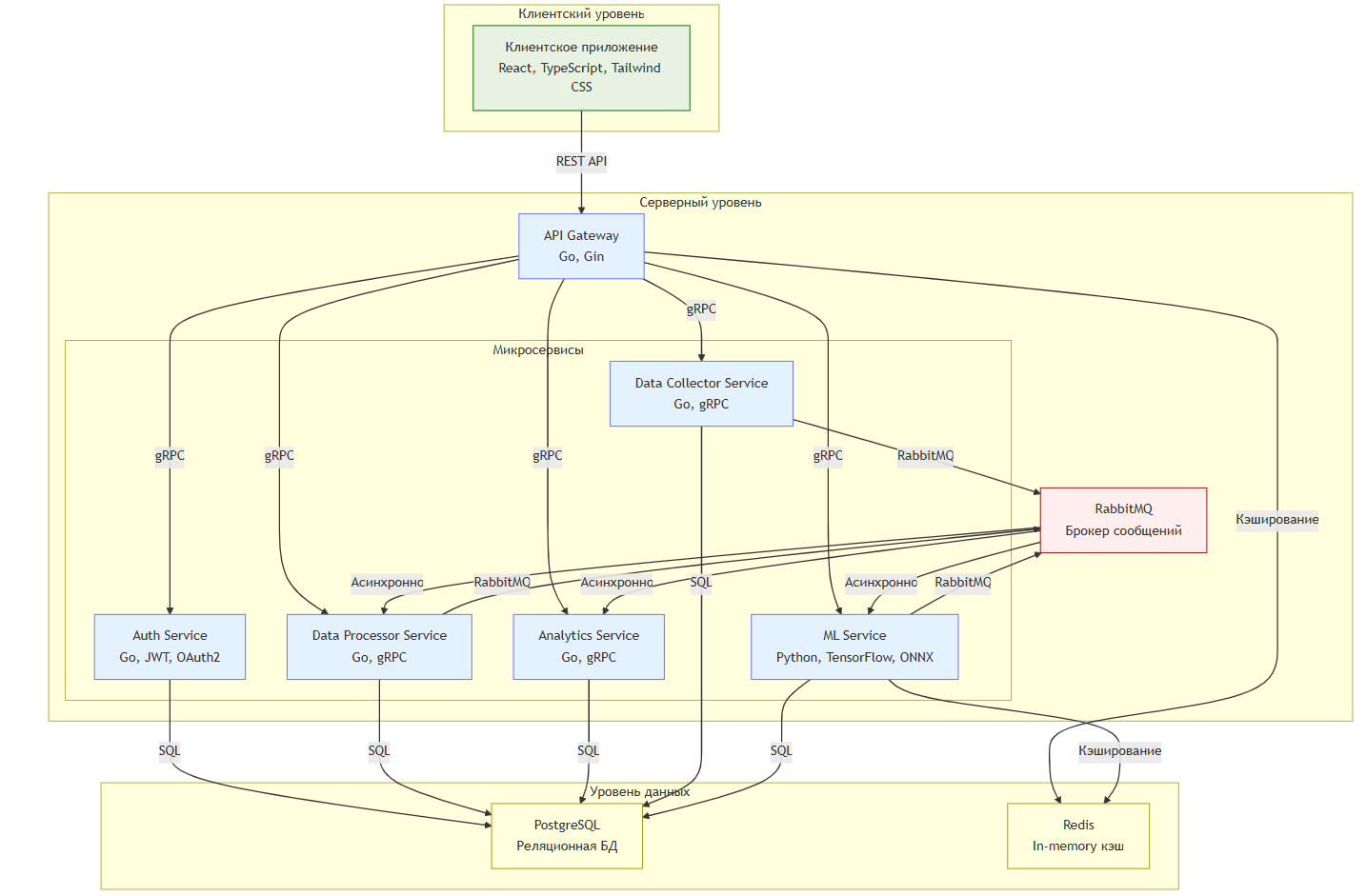


Рисунок 2.5 – Диаграмма компонентов системы

* 1. Проектирование жизненного цикла информационной системы

Для разработки информационной системы, предназначенной для автоматизированной оценки спроса на продукт, была выбрана итеративная модель жизненного цикла. Этот подход предусматривает последовательное выполнение этапов разработки с возможностью возврата к предыдущим стадиям для уточнения требований и исправления выявленных недочётов. Итеративная модель была выбрана благодаря её гибкости, которая позволяет адаптироваться к изменениям в процессе разработки, а также обеспечивать раннее тестирование и постепенное наращивание функциональности системы.

Процесс разработки организован в виде повторяющихся циклов, каждый из которых включает следующие этапы: анализ требований, проектирование архитектуры и компонентов, реализацию программного кода и тестирование промежуточных результатов.

На этапе анализа определяются функциональные и нефункциональные требования, а также уточняются задачи для текущей итерации. Проектирование охватывает разработку архитектуры микросервисов, схемы базы данных и моделей взаимодействия компонентов. Реализация включает написание кода для каждого микросервиса, а тестирование проводится для проверки корректности работы и соответствия требованиям. После завершения каждой итерации результаты оцениваются, что позволяет корректировать план дальнейшей работы.

Преимущества итеративной модели заключаются в следующем:

* гибкость: Возможность вносить изменения в требования на любом этапе разработки, что особенно важно для проекта с использованием машинного обучения, где точность моделей может потребовать доработки;
* раннее выявление ошибок: Тестирование проводится после каждой итерации, что снижает риск накопления критических проблем к финальной стадии;
* постепенное развитие: Функциональность системы наращивается поэтапно, начиная с базовых компонентов (например, сбора данных) и заканчивая сложными функциями (прогнозирование спроса).

Схематическое изображение жизненного цикла представлено на рисунке 2.6.

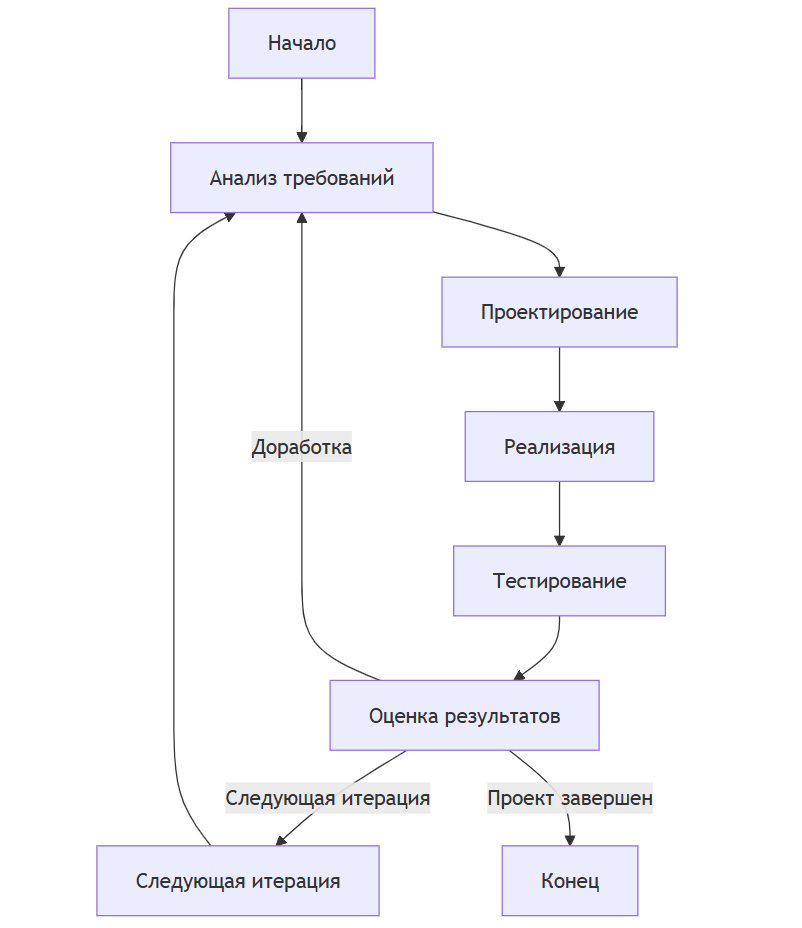


Рисунок 2.6 – Пример работы итерационной модели жизненного цикла

* 1. Проектирование схемы базы данных

Проектирование базы данных является критически важным этапом разработки информационной системы, так как от эффективности структуры хранения данных зависят производительность, масштабируемость и надёжность системы. Разработанная схема базы данных обеспечивает поддержку всех ключевых функций системы – сбора данных, их обработки, прогнозирования спроса и предоставления аналитики, – а также гарантирует гибкость для дальнейшего расширения функциональности.

В рамках микросервисной архитектуры система использует централизованную базу данных PostgreSQL с разделением данных по схемам или префиксам таблиц для каждого сервиса, что упрощает управление и минимизирует зависимости между компонентами.

Схема базы данных включает следующие основные сущности:

* пользователи (Users): Хранит информацию о пользователях системы, включая учетные данные и роли для управления доступом. Используется микросервисом Auth Service;
* источники данных (DataSources): Содержит конфигурацию источников данных (CRM, ERP, API, веб-скрапинг), включая расписание их сбора. Используется микросервисом Data Collector Service;
* сырые данные (RawData): Хранит необработанные данные, собранные из источников, с привязкой к соответствующему источнику. Используется Data Collector Service для записи и Data Processor Service для чтения;
* обработанные данные (ProcessedData): Содержит данные после очистки, нормализации и обогащения, готовые для анализа и прогнозирования. Используется Data Processor Service для записи и ML Service для чтения;
* модели машинного обучения (MLModels): Хранит информацию о моделях ML, включая их параметры, метрики и пути к файлам моделей. Используется ML Service;
* прогнозы (Forecasts): Содержит результаты прогнозирования спроса с привязкой к модели и входным данным. Используется ML Service для записи и Analytics Service для чтения;
* аналитика (Analytics): Хранит агрегированные аналитические данные для отчетов и визуализаций. Используется Analytics Service;
* отчеты (Reports): Содержит информацию о сгенерированных отчетах с привязкой к пользователю, запросившему отчет. Используется Analytics Service.

Разработанная схема базы данных представлена на рисунке 2.7.



Рисунок 2.7 – Схема базы данных системы

Получившаяся база данных будет находиться в контейнере Docker под управлением СУБД PostgreSQL. Это позволит повысить безопасность благодаря дополнительной изоляции.

## **Вывод по разделу 2**

В данном разделе спроектирована функциональная схема в методологии IDEF0 и разработана микросервисная архитектура информационной системы для автоматизированной оценки спроса на продукт. Созданы компоненты клиентской части на базе React с TypeScript и серверной части, состоящей из специализированных микросервисов. Разработаны UML-диаграммы, визуализирующие логическую модель системы, выбрана итеративная модель жизненного цикла и спроектирована схема базы данных PostgreSQL. Предложенное решение обеспечивает гибкость, масштабируемость и отказоустойчивость системы в соответствии с современными стандартами программной инженерии [18].

ЗАКЛЮЧЕНИЕ

В рамках выполнения проектной 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 и Redis, 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 процессе работы сформулированы разделы по проектированию и разработке, включая выбор технологий, проектирование архитектуры и схемы базы данных. Результаты проектной практики демонстрируют создание эффективной и гибкой интеллектуальной системы, способной прогнозировать спрос на продукт с высокой точностью и адаптироваться к различным бизнес-сценариям. Разработанное решение обладает конкурентными преимуществами благодаря универсальности источников данных, применению машинного обучения и удобству интеграции через REST API, что делает его востребованным 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: 20.12.2024). – 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: 20.12.2024). – 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: 20.12.2024). – 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\_7.32-2017.pdf (дата обращения: 20.12.2024). – Текст: электронный.
5. БИБЛИОГРАФИЧЕСКАЯ ЗАПИСЬ. БИБЛИОГРАФИЧЕСКОЕ ОПИСАНИЕ. Общие требования и правила составления ГОСТ Р 7.0.100-2018 [Электронный ресурс]. – URL: https://www.rsl.ru/photo/!\_ORS/5PROFESSIONALAM/7\_sibid/ГОСТ\_P\_7\_0\_100\_2018\_1204.pdf (дата обращения: 20.12.2024). – Текст: электронный.
6. Положение о выпускной квалификационной работе студентов, обучающихся по образовательным программам подготовки бакалавров СМКО МИРЭА 7.5.1/03.П.67-19 [Электронный ресурс]. – URL: https://www.mirea.ru/docs/177322/ (дата обращения: 20.12.2024). – 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: 20.12.2024). – Текст: электронный.
8. Ozon Seller [Электронный ресурс]. – URL: https://seller.ozon.ru/ (дата обращения: 20.12.2024). – Текст: электронный.
9. Moneyplace [Электронный ресурс]. – URL: https://moneyplace.io/ (дата обращения: 20.12.2024). – Текст: электронный.
10. MPStats [Электронный ресурс]. – URL: https://mpstats.io/ (дата обращения: 20.12.2024). – Текст: электронный.
11. TensorFlow [Электронный ресурс]. – URL: https://www.tensorflow.org/ (дата обращения: 20.12.2024). – Текст: электронный.
12. Scikit-learn [Электронный ресурс]. – URL: https://scikit-learn.org/stable/ (дата обращения: 20.12.2024). – Текст: электронный.
13. Gin Web Framework [Электронный ресурс]. – URL: https://gin-gonic.com/docs/ (дата обращения: 20.12.2024). – Текст: электронный.
14. Нотация IDEF0 [Электронный ресурс]. – URL: https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/b\_remodeling/idef\_0 (дата обращения: 20.12.2024). – Текст: электронный.
15. React [Электронный ресурс]. – URL: https://react.dev/ (дата обращения: 20.12.2024). – Текст: электронный.
16. JWT (JSON Web Token) [Электронный ресурс]. – URL: https://jwt.io/introduction/ (дата обращения: 20.12.2024). – Текст: электронный.
17. ONNX [Электронный ресурс]. – URL: https://onnx.ai/ (дата обращения: 20.12.2024). – Текст: электронный.
18. Гвоздева Т.В., Баллод Б.А. Проектирование информационных систем. Стандартизация: учебное пособие. – Санкт-Петербург: Лань, 2019. – 252 с.