|  |  |
| --- | --- |
| Логотип СПбУТУиЭ | **ЧОУ ВО «САНКТ–ПЕТЕРБУРГСКИЙ УНИВЕРСИТЕТ**  **ТЕХНОЛОГИЙ УПРАВЛЕНИЯ И ЭКОНОМИКИ»**  **ПРОГРАММЫ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ** |

**ДОПУСКАЮ К ЗАЩИТЕ**

Директор колледжа

С.С. Леванова

(подпись)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.

**ДИПЛОМНАЯ РАБОТА**

**Тема:** **:** Разработка автоматизированной информационной системы обработки данных (на примере ООО «Север-Рыба»)

**Код, специальность** 09.02.07 Информационные системы и программирование

**Форма обучения** Очная

**Группа** ОДО-ИС11-22-1

**Разработал обучающийся** К.Л.Корнеева

(подпись, дата)

**Руководитель ДР** А.А. Григорян

(должность, подпись, дата)

**Нормоконтролер** Г.В. Ситников

(должность, подпись, дата)

**Санкт–Петербург**

**2025**

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** **3**](#_Toc195271388)

[**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ** **5**](#_Toc195271389)

[**1 Анализ предметной области** **5**](#_Toc195271390)

[1.1 Характеристика предприятия и его деятельности 5](#_Toc195271391)

[1.2 Выявление проблем и обоснование необходимости автоматизации 9](#_Toc191142723)

[1.3 Современные технологии для разработки веб–приложений 9](#_Toc195271393)

[**ПРАКТИЧЕСКАЯ ЧАСТЬ** **13**](#_Toc195271394)

[**2 Техническая реализация системы веб–приложения** **13**](#_Toc195271395)

[2.1 Обоснование выбора предметной области, способов обработки данных и внедряемых технологий 13](#_Toc195271396)

[2.2 Выбор программных инструментов и технологий 20](#_Toc191142728)

[2.3 Экономическое обоснование веб-приложения 22](#_Toc195271398)

[**3 Разработка автоматизированной информационной системы** **25**](#_Toc195271399)

[3.1 Разработка backend части с помощью FastApi 25](#_Toc195271400)

[3.2 Разработка frontend части с помощью React 33](#_Toc195271401)

[3.3 Разработка базы данных с помощью PostgreSQL 38](#_Toc195271402)

[3.4 Интерфейс веб–приложения 41](#_Toc195271403)

[**ЗАКЛЮЧЕНИЕ** **47**](#_Toc195271404)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** **49**](#_Toc195271405)

[**ПРИЛОЖЕНИЯ** **51**](#_Toc195271406)

**ПРИЛОЖЕНИЕ А** [**Код контроллера AdminController** **51**](#_Toc195271407)

**ПРИЛОЖЕНИЕ Б** [**Код из контроллера ChatController** **57**](#_Toc195271408)

# **ВВЕДЕНИЕ**

В современных условиях развития информационных технологий предприятия сталкиваются с необходимостью автоматизации процессов обработки данных. В связи с этим для повышения эффективности работы и оптимизации бизнес-процессов во многих организациях внедряются специализированные информационные системы.

Актуальность автоматизированной информационной системы обусловлена понижением затрат, повышением точности, а также уменьшению вероятности ошибок. Это особенно актуально для предприятий, занимающихся производством и реализацией продукции, где важно учитывать множество параметров, таких как логистика, складские остатки, поставки и продажи.

Одно из главных преимуществ данного проекта, это что он будет разрабатываться с помощью современных веб–технологий, включая фреймворк FastApi для серверной части, React для клиентской стороны и технологию PostgreSQL для реализации базы данных приложения. Такой технологический стек обеспечивает высокую производительность системы, ее масштабируемость и безопасность хранения данных, что особенно важно при работе с конфиденциальной информацией, связанной с производственным процессом.

В данной выпускной квалификационной работе рассматривается процесс разработки автоматизированной информационной системы, предназначенной для эффективной обработки данных на предприятии. Основное внимание уделяется созданию удобного интерфейса для пользователей, разработке надежной базы данных и обеспечению высокой производительности системы.

Объектом исследования является процесс учета и обработки данных в компании ООО «Север-Рыба». Предметом исследования выступает разработка и внедрение автоматизированной информационной системы, обеспечивающей централизованное хранение и обработку информации.

Целью данной работы является создание автоматизированной информационной системы, которая позволит оптимизировать процессы учета, повысить точность и скорость обработки данных, а также упростить контроль над операционной деятельностью предприятия.

Для достижения поставленной цели были сформулированы следующие задачи:

* изучение современных технологий автоматизации и выбор наиболее подходящих инструментов для разработки системы;
* реализация программных модулей и алгоритмов обработки данных;
* обосновать выбор технологического стека.

В первой главе описываются теоретические основы разработки автоматизированной информационной системы, также рассматриваются потребности в автоматизации производственного процесса, функциональные требования к системе и современные технологии для создания веб–приложений.

Во второй главе рассматривается техническая реализация системы и описывается выбор технологий, проектирование архитектуры с помощью Process Modeler, а также будет представлена экономическая выгода автоматизированной информационной системы

В третьей главе показывается основная разработка веб–приложения детализируется процесс создания backend при помощи FastApi, frontend с помощью React и базы данных на PostgreSQL.

Разработчиком дипломной работы является Корнеева Ксения – студентка 3 курса, группы ОДО–ИС11–22–1 отделение сервиса и прикладной информатики СПбУТУиЭ.

# **ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

# **1 Анализ предметной области**

## 1.1 Характеристика предприятия и его деятельности

ООО «Север-Рыба» – предприятие, занимающееся переработкой и реализацией рыбной продукции. Основное направление деятельности компании – закупка, обработка, хранение и поставка рыбной продукции оптовым и розничным покупателям. Предприятие активно сотрудничает с поставщиками сырья, логистическими компаниями и торговыми сетями, обеспечивая широкий ассортимент продукции на рынке.

Компания располагает современной производственной базой, включающей цеха по переработке рыбы, холодильные склады, упаковочные линии и собственную логистическую инфраструктуру. Производственные мощности позволяют осуществлять все этапы переработки, включая очистку, разделку, шоковую заморозку, копчение и вакуумную упаковку продукции. Все процессы соответствуют санитарно-гигиеническим нормам и требованиям к качеству пищевых продуктов.

ООО «Север-Рыба» уделяет особое внимание контролю качества продукции. На каждом этапе производства осуществляется строгий контроль, включая лабораторные исследования образцов продукции, соблюдение температурных режимов хранения и контроль поставок.

На сегодняшний день предприятие сталкивается с рядом сложностей, связанных с обработкой и управлением данными. Учет поставок, контроль складских запасов, документооборот и управление заказами ведутся с использованием устаревших методов, что может приводить к несвоевременной обработке информации и возникновению ошибок.

Для повышения эффективности работы и оптимизации процессов учета данных компания рассматривает возможность внедрения автоматизированной информационной системы. Данная система позволит минимизировать человеческий фактор, ускорить обработку информации, повысить точность учета и улучшить управление ресурсами предприятия. Внедрение автоматизированной системы также обеспечит прозрачность операций и упростит анализ ключевых показателей деятельности компании.

## 1.2 Выявление проблем и обоснование необходимости автоматизации

Современные предприятия, работающие с большими объемами данных, сталкиваются с рядом сложностей, связанных с учетом, обработкой и анализом информации. В условиях растущей конкуренции и высокой динамики рынка эффективное управление данными становится ключевым фактором успешного функционирования организации. При отсутствии современных решений по автоматизации на предприятии могут возникать значительные проблемы, влияющие на скорость и качество выполнения бизнес-процессов.

Одной из главных проблем, с которыми сталкиваются компании, является высокая нагрузка на сотрудников, связанная с необходимостью обработки большого количества информации вручную. При использовании традиционных методов учета, таких как ведение бумажной документации или работа с электронными таблицами, возрастает вероятность ошибок, связанных с человеческим фактором. Даже небольшие погрешности при внесении данных могут приводить к серьезным последствиям, таким как нарушение отчетности, проблемы с инвентаризацией и сбои в логистических процессах.

Еще одной распространенной проблемой является дублирование данных, возникающее при отсутствии единой централизованной базы. В организациях, использующих разрозненные методы хранения информации, нередко встречаются ситуации, когда одни и те же данные фиксируются в разных источниках без синхронизации между ними. Это приводит к несогласованности информации, усложняет процесс анализа и затрудняет принятие управленческих решений.

Кроме того, важным аспектом является сложность оперативного доступа к данным. В условиях стремительного развития технологий компании должны иметь возможность быстро получать необходимую информацию для анализа и принятия решений. Однако при ручных методах обработки поиск нужных данных может занимать значительное время, что замедляет работу сотрудников и снижает эффективность бизнеса.

Также стоит отметить проблемы, связанные с безопасностью данных. Ведение учета вручную или использование локальных файловых хранилищ без должной защиты может привести к утечке конфиденциальной информации или потере важных данных. Современные предприятия нуждаются в надежных механизмах защиты, обеспечивающих ограниченный доступ к информации, резервное копирование данных и контроль за их изменениями.

Все вышеперечисленные проблемы подтверждают необходимость внедрения автоматизированной информационной системы, способной централизовать учет данных, минимизировать влияние человеческого фактора и ускорить процесс обработки информации. Использование автоматизированных решений позволяет значительно повысить точность данных, обеспечить их доступность в режиме реального времени, а также улучшить аналитические возможности компании.

Автоматизация процессов обработки данных дает ряд ключевых преимуществ:

* уменьшение вероятности ошибок за счет исключения ручного ввода информации;
* централизованное хранение данных с возможностью быстрого поиска и анализа;
* автоматическое обновление информации и синхронизация между различными подразделениями;
* улучшение безопасности и защиты данных за счет контроля доступа и резервного копирования;
* оптимизация рабочего времени сотрудников, сокращение затрат на ведение учета;
* возможность интеграции с другими системами для комплексного управления бизнес-процессами.

Таким образом, внедрение автоматизированной информационной системы является необходимым шагом для повышения эффективности работы предприятия.

## 1.3 Современные технологии для разработки веб–приложений

При разработке автоматизированной информационной системы необходимо выбрать методы и технологии, которые обеспечат надежную, безопасную и эффективную работу системы. Выбор подходящих решений зависит от множества факторов, включая объем обрабатываемых данных, требования к быстродействию, удобство использования, интеграцию с другими системами, а также надежность и безопасность. Каждый из этих факторов играет важную роль в проектировании системы и должен быть учтен на этапе разработки.

Одним из ключевых элементов автоматизированной информационной системы является база данных, так как именно она отвечает за хранение, обработку и доступ к данным. На данный момент существует множество решений для управления базами данных, однако наибольшей популярностью пользуются реляционные базы данных, такие как MySQL, PostgreSQL и Microsoft SQL Server. Они обеспечивают структурированное хранение данных, возможность выполнения сложных запросов и высокий уровень безопасности. Среди всех доступных вариантов в данном проекте было принято решение использовать PostgreSQL, так как эта система обладает высокой производительностью, поддерживает транзакции, репликацию и масштабирование. Кроме того, PostgreSQL является свободно распространяемой системой, что делает ее доступной для использования без необходимости приобретения лицензий.

Для серверной части системы необходимо выбрать технологию, которая обеспечит быструю обработку запросов, поддержку асинхронных операций и удобную архитектуру. Среди возможных решений можно выделить такие технологии, как PHP, Node.js, Django и FastAPI. PHP является широко распространенным языком программирования для веб-разработки, однако он не обладает достаточной производительностью при обработке больших объемов данных. Node.js предлагает асинхронную обработку запросов и высокую скорость работы, но требует специфического подхода к написанию серверной логики. Django представляет собой мощный веб-фреймворк, однако его монолитная структура делает систему менее гибкой. В данном проекте выбрана технология FastAPI, так как она сочетает в себе высокую скорость работы, поддержку асинхронных операций и удобный механизм валидации данных. FastAPI также обеспечивает удобную интеграцию с базами данных и поддерживает OpenAPI, что облегчает документирование API.

Пользовательский интерфейс играет важную роль в автоматизированной системе, так как именно через него пользователи взаимодействуют с системой. Существует множество технологий для разработки клиентской части, среди которых можно выделить HTML + CSS + JavaScript, Angular, Vue.js и React.js. HTML, CSS и JavaScript являются основными инструментами для создания веб-страниц, однако их использование без дополнительных библиотек требует значительных усилий при реализации сложного функционала. Angular представляет собой мощный фреймворк, но он требует высокой квалификации разработчиков и достаточно сложен в освоении. Vue.js является более легковесным решением, но не предоставляет такой же гибкости, как React.js. В данном проекте выбрана технология React.js, так как она позволяет создавать динамические пользовательские интерфейсы, обеспечивает высокую скорость работы и предоставляет удобные инструменты для управления состоянием приложения.

Для передачи данных между клиентской и серверной частями системы используется REST API. REST API является стандартом для веб-разработки и обеспечивает простой, удобный и надежный способ передачи информации. Он позволяет клиенту запрашивать данные с сервера в формате JSON и получать ответы в удобном для обработки виде. В отличие от других технологий, таких как GraphQL, REST API не требует сложной настройки и легко интегрируется с любыми клиентскими приложениями. Использование REST API также обеспечивает защиту передаваемых данных, так как можно применять различные механизмы аутентификации и авторизации.

Безопасность данных является важнейшим аспектом при разработке автоматизированной информационной системы.

Для защиты информации необходимо использовать современные технологии, позволяющие предотвратить несанкционированный доступ, кражу данных и утечку конфиденциальной информации. В данном проекте используются такие методы, как аутентификация с помощью JWT (JSON Web Token), шифрование паролей с использованием алгоритма bcrypt, а также передача данных по защищенному протоколу HTTPS. JWT позволяет безопасно управлять сеансами работы пользователей, исключая возможность подделки или кражи токенов. Алгоритм bcrypt обеспечивает надежное хеширование паролей, что делает невозможным их восстановление злоумышленниками. Использование HTTPS предотвращает возможность перехвата передаваемых данных, что особенно важно при работе с конфиденциальной информацией.

Развертывание системы осуществляется с использованием технологии контейнеризации Docker. Docker позволяет создавать изолированные контейнеры для различных компонентов системы, таких как серверная часть, база данных и фронтенд-приложение. Это упрощает процесс развертывания, обеспечивает удобное управление зависимостями и повышает надежность работы системы. Использование контейнеризации также позволяет легко переносить систему между различными средами без необходимости настройки окружения с нуля.

Для управления исходным кодом используется система контроля версий Git, а код хранится в репозитории на платформе GitHub. Git позволяет отслеживать изменения в коде, управлять версиями проекта и организовывать совместную работу разработчиков. Использование Git также дает возможность откатываться к предыдущим версиям кода в случае возникновения ошибок, что значительно облегчает процесс отладки и тестирования.

Выбор технологий играет ключевую роль в разработке автоматизированной информационной системы, так как от него зависит производительность, удобство использования, безопасность и масштабируемость системы. PostgreSQL обеспечивает надежное хранение данных, FastAPI – быструю обработку запросов, а React.js – удобный и динамичный пользовательский интерфейс. REST API гарантирует эффективную передачу данных между клиентом и сервером, а современные методы безопасности защищают информацию от угроз. Контейнеризация с помощью Docker и управление кодом через Git обеспечивают надежную эксплуатацию и поддержку системы.

Таким образом, при разработке автоматизированной информационной системы были выбраны наиболее оптимальные технологии, соответствующие современным требованиям. Использование современных решений позволит создать эффективную, удобную и безопасную систему, способную автоматизировать бизнес-процессы и повысить их эффективность. Благодаря тщательно подобранному технологическому стеку система сможет легко адаптироваться к изменяющимся требованиям бизнеса и масштабироваться при необходимости

# **ПРАКТИЧЕСКАЯ ЧАСТЬ**

# **2 Техническая реализация системы веб–приложения**

## 2.1 Обоснование выбора предметной области, способов обработки данных и внедряемых технологий

Чтобы описать бизнес–процесс веб–приложения, потребуется использовать различные ПО, такие как Visio, Process Modeler (ERwin/BPwin), StarUML и другие. Для этой реализации я буду использовать программу Process Modeler (ERwin).

ERwin – это программное обеспечение для моделирования данных, которое используется для создания и управления логическими и физическими моделями баз данных. Оно помогает организациям проектировать и документировать структуры данных, а также обеспечивает интеграцию с различными системами управления базами данных (СУБД). ERwin позволяет автоматизировать процесс создания схем баз данных, что упрощает разработку и поддержку сложных информационных систем. Основные функции данной программы являются [21]:

* моделирование данныx, создание логических и физических моделей данных;
* генерация схем баз данных, автоматическая генерация SQL–скриптов для создания баз данных;
* реверс–инжиниринг, импорт существующих баз данных для создания их моделей;
* интеграция с СУБД, поддержка различных баз данных, таких как Oracle, MySQL, PostgreSQL, SQL Server и других;
* документирование, генерация отчетов и документации по моделям данных;
* коллаборация, возможность совместной работы над моделями в команде.

IDEF0 – это методология и нотация для моделирования бизнес–процессов и функциональных зависимостей. Она используется для описания системы в терминах её функций, входов, выходов, механизмов и управления. IDEF0 фокусируется на том, что делает система, а не на том, как она это делает.

Основные элементы диаграммы IDEF0:

* функция (Activity), прямоугольник, представляющий действие или процесс;
* входы (Inputs), данные или материалы, которые используются в процессе (стрелки слева);
* выходы (Outputs), результаты процесса (стрелки справа);
* механизмы (Mechanisms), ресурсы, которые выполняют процесс (стрелки снизу);
* управление (Controls), правила, стандарты или ограничения, влияющие на процесс (стрелки сверху).

На рисунке 1 указан проект «Электронный журнал», и то какие основные функции он будет выполнять, это также и будет примером, того, как должна выглядеть IDEF0.

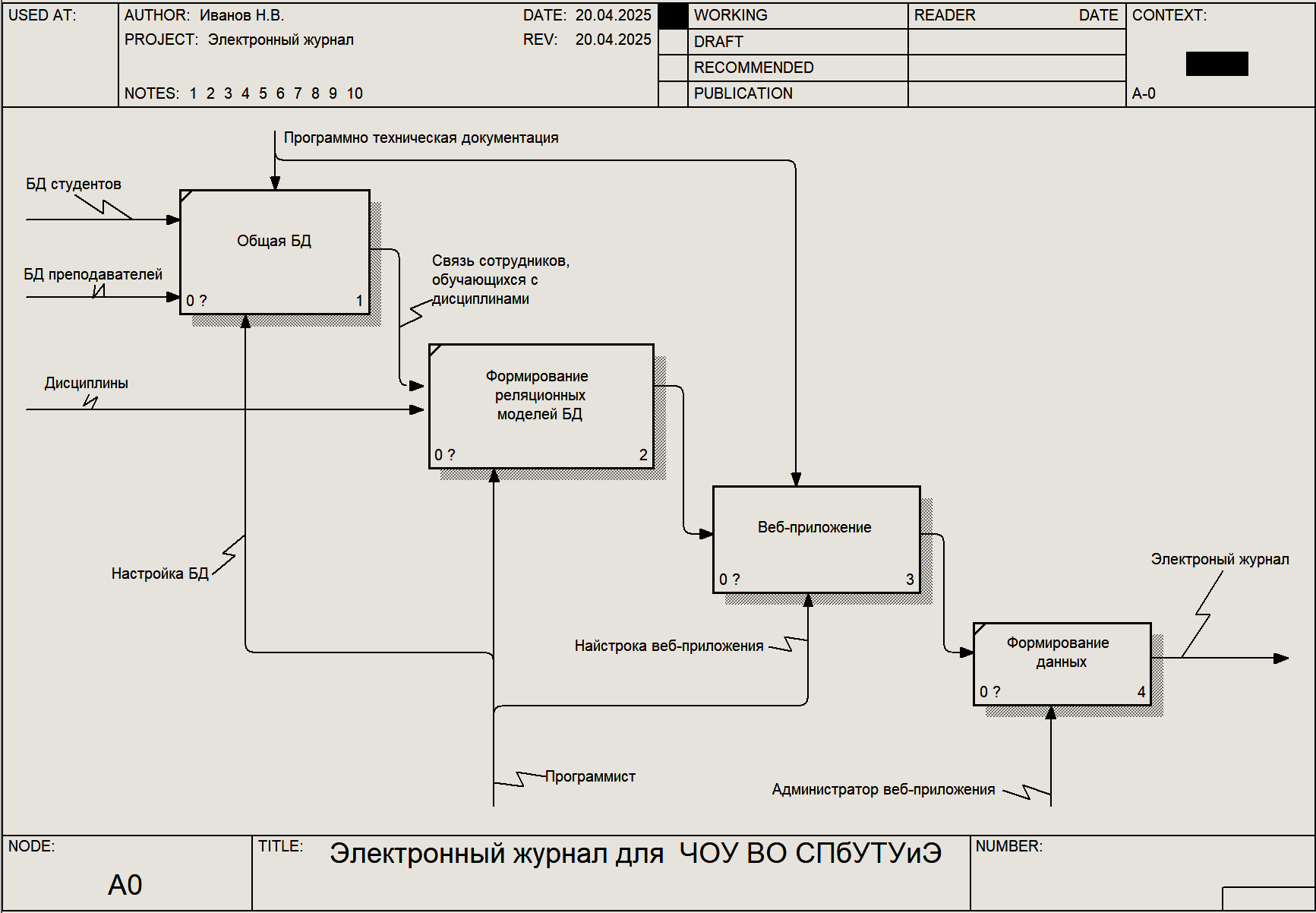


Рисунок 1 – Разработка «Электронный журнал»

Как видно на рисунке выше, после декомпозиции блока «Электронный журнал» мы создаем 4 блока IDEF0, где каждый блок играет важную роль для проектирования веб-приложения. В конечном итоге мы получим электронный журнал. Первый блок «Общая БД» имеет самую большую важность, для создания проекта, так как важно знать, каждого обучающегося и преподавателя, чтобы в дальнейшем конкатенировать их в таблицы и тем самым образуется второй блок «Формирование реляционных моделей БД».

Далее у нас образуется блок «Веб-приложение», в котором будет продемонстрирована структура в IDEF3, в ней будет логистика выбора роли пользователя, где по окончанию будет понятно, какие исходные данные будет получать пользователь. И последний блок «Формирование данных», который уже отвечает за итоговый результат, по настройкам и фильтрам.

IDEF3 – это методология для описания последовательностей действий и взаимосвязей между процессами. В отличие от IDEF0, которая фокусируется на функциях, IDEF3 больше ориентирована на как выполняются процессы и на их временные зависимости.

Основные элементы диаграммы IDEF3:

* единицы работы (Units of Behavior, UOB), блоки, представляющие конкретные действия или события;
* связи (Links), линии, показывающие последовательность выполнения действий;
* точки ветвления (junctions), элементы, описывающие параллельные или альтернативные пути выполнения процессов;
* объекты (Objects), данные или ресурсы, которые участвуют в процессах.

На рисунке 2 указана иерархия блоков, то есть как проходит авторизация пользователя, где после чего проходит определение его роли.

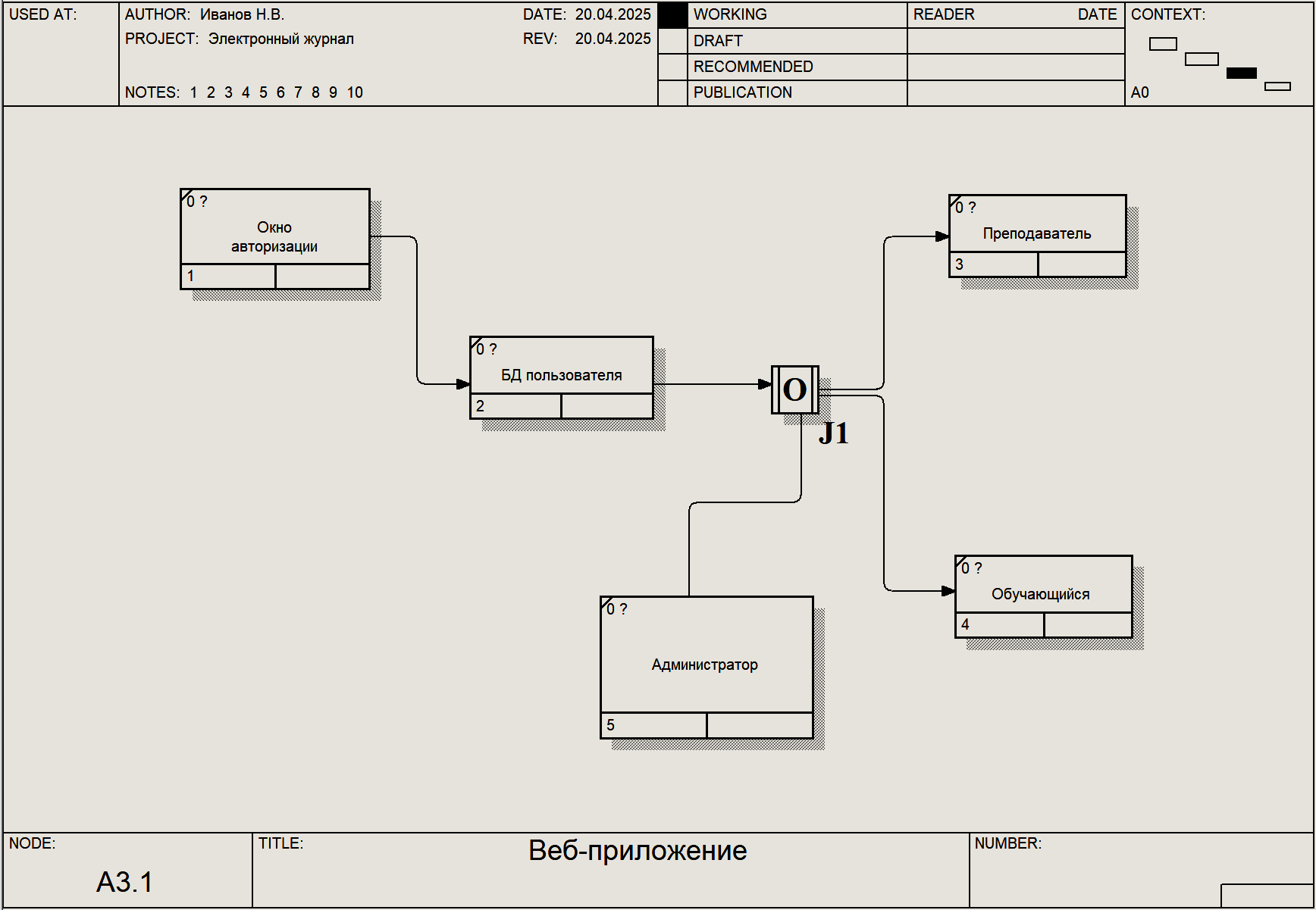


Рисунок 2 – Структура работы веб-приложения

Исходя из рисунка выше, можно подчеркнуть, что пользователь под «Администратор», имеет прямую и обратную связь со всеми пользователями, другими словами имеет полный доступ.

## 2.2 Выбор программных инструментов и технологий

Разработка автоматизированной информационной системы требует использования современных программных инструментов и технологий, которые обеспечат надежную, безопасную и эффективную работу системы. Выбор технологий основан на анализе требований к системе, ее функциональности, производительности, безопасности и удобству использования.

Автоматизированная система должна обеспечивать хранение, обработку и передачу данных, а также предоставлять пользователям удобный интерфейс для работы. Для этого необходимо выбрать подходящие технологии для серверной части, базы данных, клиентского интерфейса, безопасности и развертывания.

Серверная часть отвечает за обработку запросов пользователей, управление данными и взаимодействие с базой данных. Возможные варианты для реализации серверной части включают PHP, Node.js, Django и FastAPI. В данном проекте выбран FastAPI, так как он обеспечивает высокую производительность, поддержку асинхронных запросов, удобную работу с базами данных и интеграцию с API-документацией.

База данных является ключевым компонентом системы, обеспечивая хранение и управление информацией. Среди возможных решений были рассмотрены MySQL, PostgreSQL и MongoDB. В данном проекте используется PostgreSQL, так как она поддерживает сложные транзакции, обеспечивает надежность хранения данных и хорошо масштабируется.

Клиентская часть системы должна быть удобной, интуитивно понятной и динамичной. Рассматриваемые варианты включают использование HTML + CSS + JavaScript, Vue.js, Angular и React.js. В данном проекте выбран React.js, так как он позволяет создавать адаптивные и производительные интерфейсы с возможностью повторного использования компонентов.

Для обмена данными между клиентской и серверной частями используется REST API, так как он является стандартным и хорошо поддерживается всеми современными технологиями. REST API позволяет передавать данные в формате JSON, что упрощает взаимодействие между различными компонентами системы.

Защита данных пользователей является важным аспектом разработки системы. В проекте используются следующие методы:

* + JWT (JSON Web Token) – для аутентификации пользователей и контроля доступа к данным.
  + HTTPS – для безопасного обмена информацией между клиентом и сервером.
  + bcrypt – для безопасного хранения паролей пользователей.

Для удобства развертывания системы используются следующие инструменты:

* Docker – для контейнеризации всех компонентов системы, упрощающий переносимость и масштабируемость.
* GitHub – для управления версиями кода и совместной работы над проектом.
* Nginx – для балансировки нагрузки и обработки входящих запросов.

Выбранные технологии обеспечивают надежность, безопасность и производительность системы. Использование FastAPI для серверной части, PostgreSQL для базы данных и React.js для клиентской части делает систему гибкой и удобной.

REST API обеспечивает передачу данных, а современные методы безопасности защищают информацию пользователей. Развертывание через Docker и Nginx делает систему масштабируемой и легко управляемой.

Таким образом, выбранные программные инструменты и технологии позволяют создать надежную и эффективную автоматизированную информационную систему, полностью соответствующую требованиям предприятия.

## 2.3 Экономическое обоснование веб–приложения

Одним из главных аспектов проекта автоматизированной информационной системы для ООО "Север-Рыба" является «Модуль управления запасами», если администратор занимается настройкой системы, редактированием данных, созданием пользователей и другими важными функциями, то «Модуль управления запасами» должен полностью сам автоматически создаваться и обновляться.

В самом модуле управления запасами ничего сложного не должно быть, есть какая-то категория продукции, у неё на каждый склад должен быть свой учетный журнал, где будет продукт, дата и количество. Самое интересное и сложное, это должно быть всё организовано автоматически, например администратор создал новую категорию продукции, добавил туда товары, и создал связь между категорией и складом, то есть у одной категории может быть несколько складов, на которых она должна храниться, так же эту категорию он закрепил за каким-то менеджером или за группой менеджеров.

После всех действий, последний шаг остаётся за складским работником, он открывает раздел «Модуль управления запасами», выбирает какую-то категорию продукции, которая за ним закреплена и открывает журнал, важный момент, что склад определяется автоматически, так как за каждым складским работником помимо категории продукции закреплён ещё и склад, на котором он работает. Система должна определить какой склад обслуживает данный работник, существуют ли в принципе какие-то записи по этой категории продукции и по этому складу, если их нет она должна создать на каждый продукт блок (запись), где будет продукт, категория, количество, которое по умолчанию нулевое и сегодняшняя дата. В случае, если какие-то записи были найдены, система должна проверить, а есть ли блок (запись) на сегодняшнюю дату, если нет она должна её создать, но если записи какие-то есть и есть также записи на сегодня, то система должна просто всех их вернуть, и на стороне клиента просто отрисовать в виде таблицы. В итоге, получается полностью автоматизированный раздел «Модуль управления запасами» и администратору для его работы ничего делать не нужно.

Исходя из выше сказанного, можно прийти к решению, что при использовании исключительно данного стека технологий наше веб-приложение будет куда быстрее и производительнее. Следовательно, с использованием FastAPI для бэкенда, React для фронтенда и PostgreSQL для базы данных, потребуется куда меньше времени, как для изучения этих технологий, так и для разработки веб-приложения, нежели чем использовать альтернативные технологические решения.

Теперь перейдём к одной из главных уникальностей проекта «АИС для ООО "СЕВЕР-РЫБА"», это асинхронная обработка запросов и эффективное взаимодействие между клиентом и сервером. FastAPI, построенный на Starlette и Pydantic, обеспечивает высокую производительность благодаря встроенной поддержке асинхронных операций, что особенно важно при работе с большими объемами данных в рыбной промышленности. В отличие от традиционных REST API, где каждый запрос блокирует выполнение до получения ответа, асинхронный подход позволяет параллельно обрабатывать множество запросов, значительно увеличивая пропускную способность системы. Это работает следующим образом:

1. клиент отправляет запрос к API, который обрабатывается асинхронно, не блокируя другие операции;
2. сервер может одновременно обрабатывать множество запросов, оптимально используя ресурсы системы;
3. результаты запросов возвращаются клиенту по мере их готовности, без ненужных задержек.

Благодаря этой архитектуре, если администратор решит создать или отредактировать категорию продукции, система быстро обработает изменения и сразу же отобразит их пользователям. React с его виртуальным DOM обеспечивает эффективное обновление только тех частей интерфейса, которые действительно изменились. PostgreSQL, как надежная и производительная реляционная база данных, гарантирует целостность и быстрый доступ к данным. Полный список преимуществ выбранного технологического стека:

* высокая производительность FastAPI для обработки запросов;
* эффективная отрисовка интерфейса с помощью React;
* надежное хранение данных в PostgreSQL.

Как итог, выбранный стек технологий обеспечивает оптимальное сочетание производительности, масштабируемости и удобства разработки. FastAPI предоставляет автоматическую валидацию данных и интерактивную документацию API, React упрощает создание динамических пользовательских интерфейсов, а PostgreSQL обеспечивает надежное хранение данных с поддержкой сложных запросов и транзакций.

Экономический эффект заключается в простоте изучения и использования современных технологий FastAPI, React и PostgreSQL, то есть наше веб-приложение разработается как быстрее, так и выгоднее в ценовой политике при заказе разработки веб-приложения.

# **3 Разработка веб–приложения «Электронный журнал»**

## 3.1 Разработка backend части с помощью FastApi

Для разработки backend части проекта использовался фреймворк FastAPI, Python 3.11 и pip для управления зависимостями. FastAPI был выбран благодаря высокой производительности и простоте создания API-интерфейсов с автоматической генерацией документации. Для установки необходимых компонентов использовались следующие команды, представленные на рисунке 3. Установка фреймворка FastAPI и его компонентов представлена на рисунке 4.

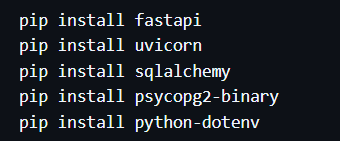


Рисунок 3 – Установка необходимых компонентов

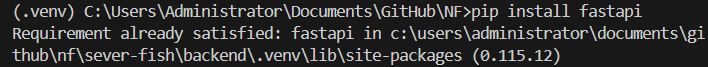


Рисунок 4 – Установка библиотеки FastAPI

Для работы с базой данных используется PostgreSQL, настройки подключения к которой хранятся в файле .env, как показано на рисунке 5. Это позволяет обеспечить безопасность доступа и гибкость при переносе проекта между различными средами разработки.

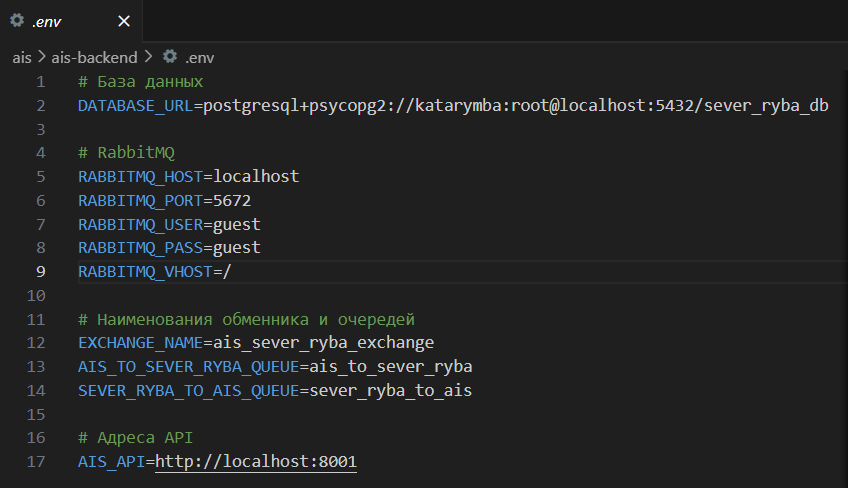


Рисунок 5 – Конфигурационный файл .env с настройками подключения к базе данных

Структура проекта организована в соответствии с принципами модульности и разделения ответственности. Основные компоненты backend-части приложения включают в себя файлы, представленные на рисунке 6.

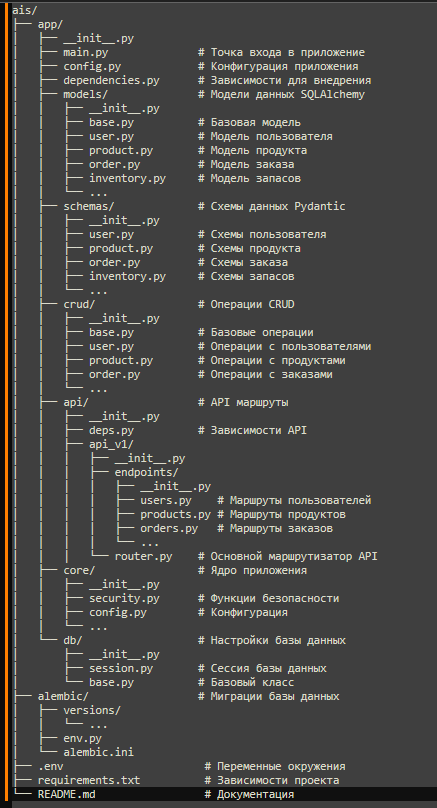


Рисунок 6 – Список моделей

Модели данных в проекте определены с использованием SQLAlchemy ORM, что позволяет абстрагироваться от особенностей конкретной СУБД и работать с объектами Python. На рисунке 7 представлен пример модели пользователя, которая включает в себя основные атрибуты и отношения с другими моделями.

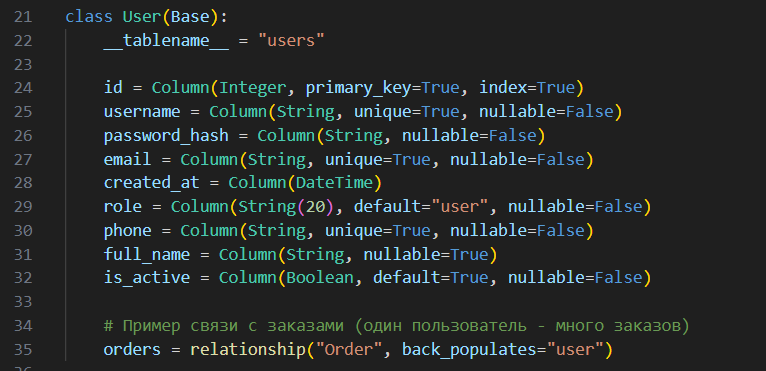


Рисунок 7 – Модель пользователя системы

Для валидации входящих данных и структурирования исходящих ответов API используются схемы Pydantic. Они обеспечивают строгую типизацию данных, автоматическую валидацию и преобразование типов. На рисунке 8 представлен пример схемы для создания и отображения администратора.

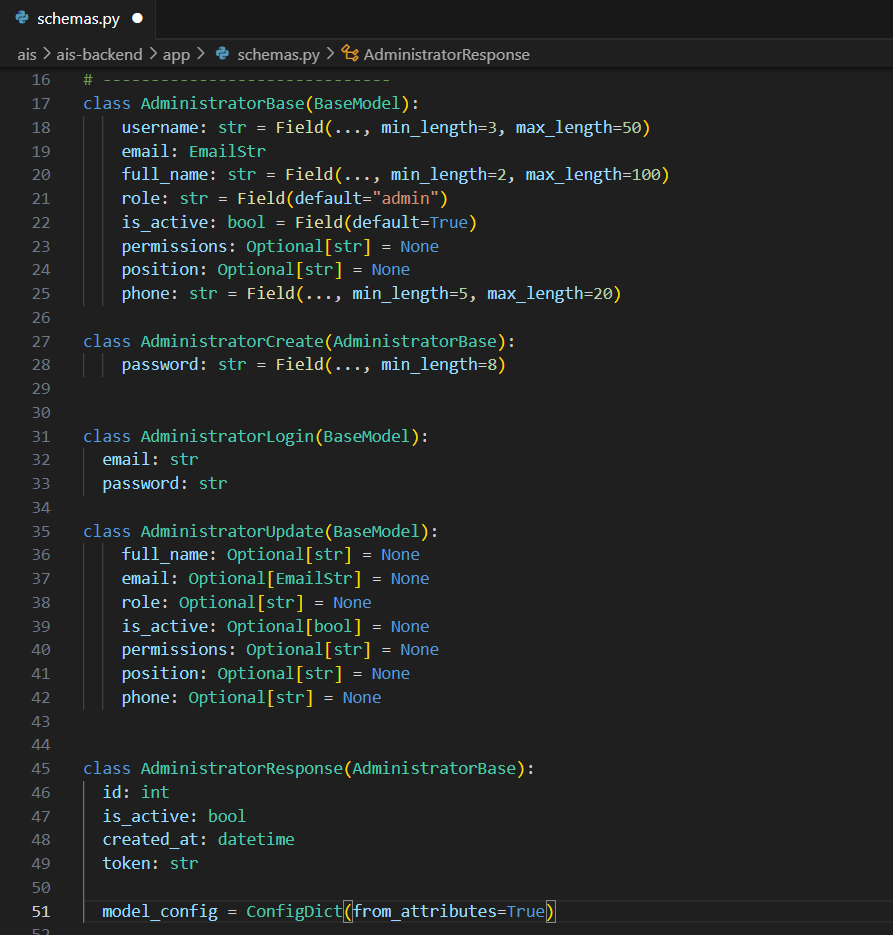


Рисунок 8 – Схемы Pydantic для работы с администраторами

Маршрутизация запросов в FastAPI организована с использованием модуля APIRouter, который позволяет группировать связанные маршруты и применять общие зависимости. На рисунке 9 представлен пример маршрутов для работы с администраторами.

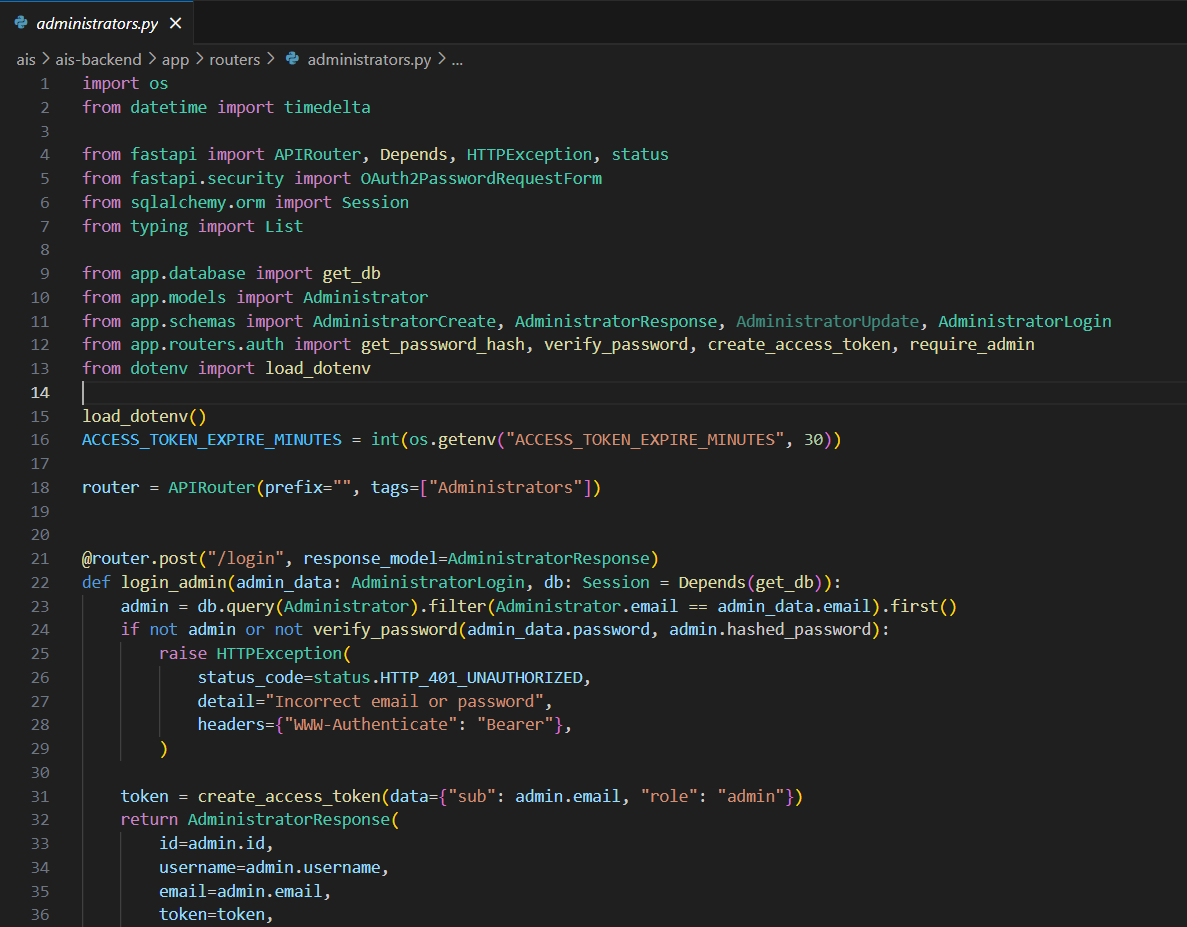


Рисунок 9 – Маршруты API для работы с администраторами

Для обеспечения безопасности API реализована аутентификация на основе JSON Web Token (JWT). На рисунке 10 представлен фрагмент кода, отвечающий за создание и проверку токенов.

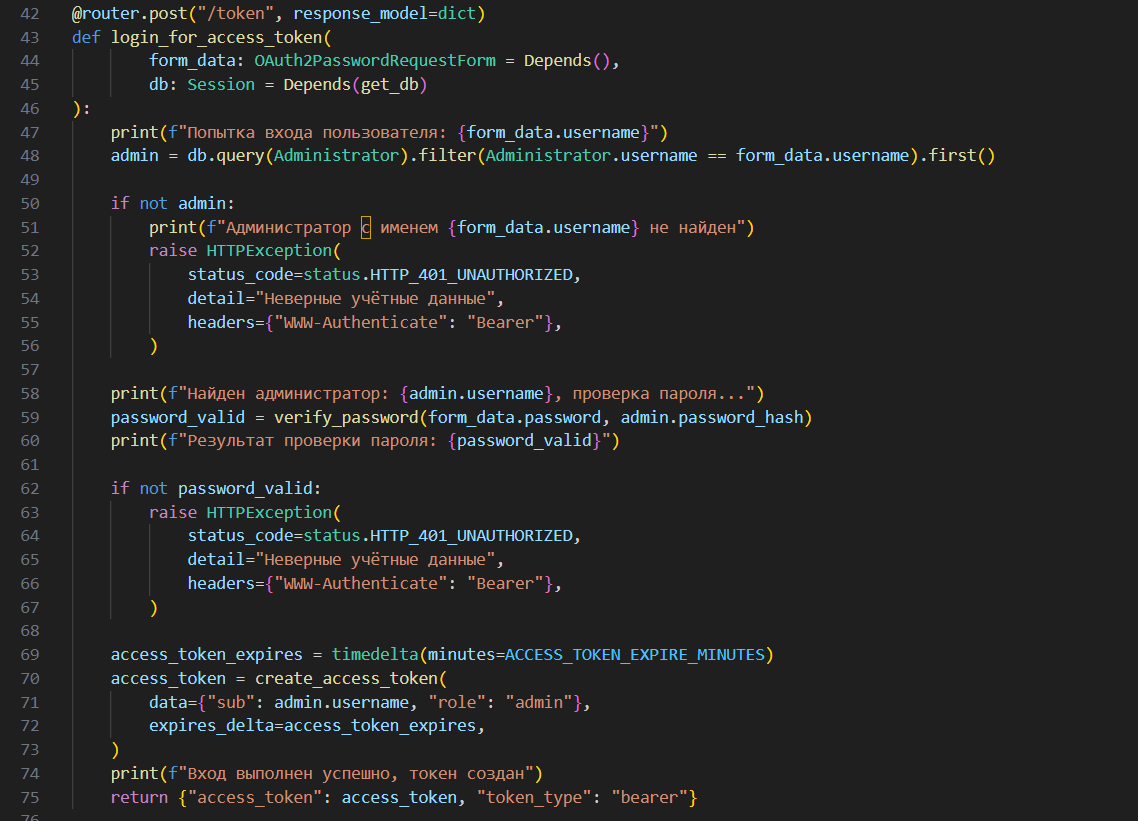


Рисунок 10 – Модуль безопасности для работы с JWT-токенами

Для управления миграциями базы данных используется Alembic, который позволяет отслеживать изменения в структуре БД и применять их в упорядоченном виде. На рисунке 11 представлен пример миграции, создающей таблицу пользователей.

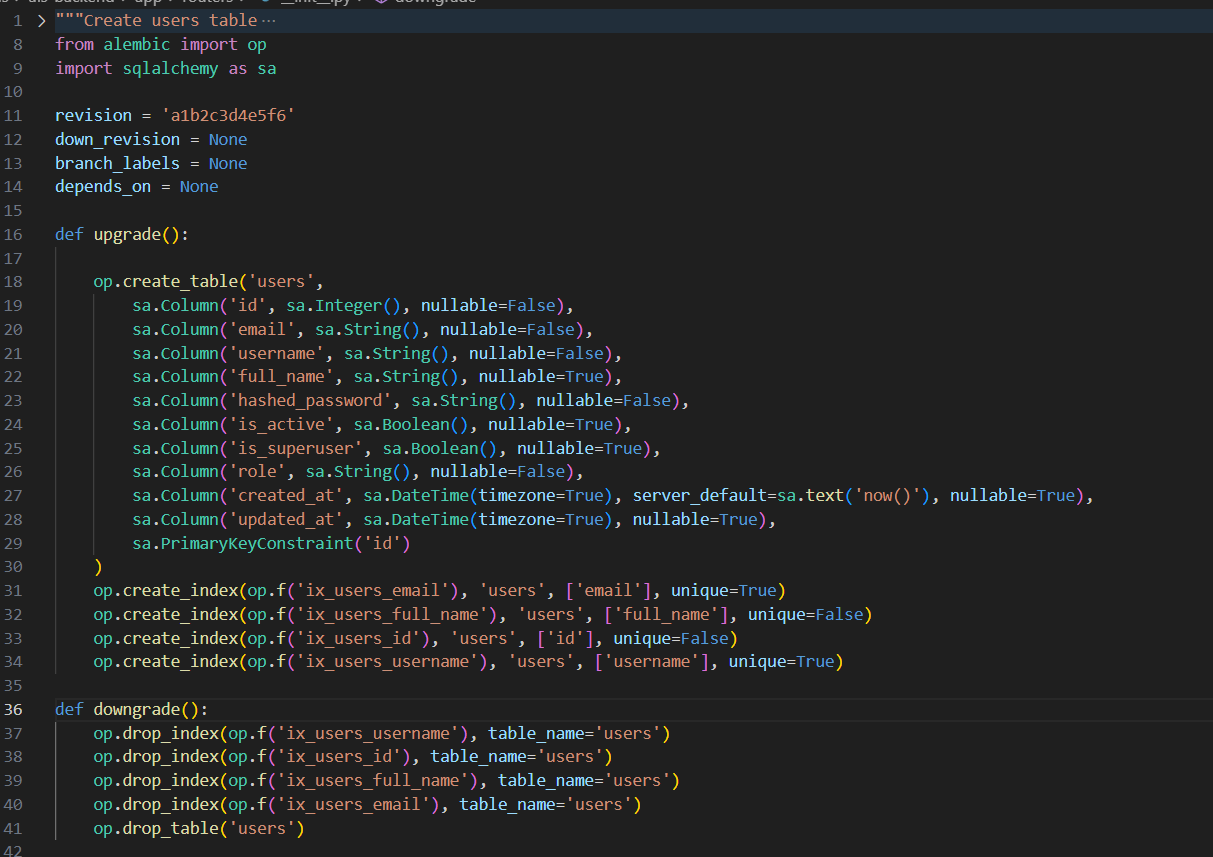


Рисунок 11 – Пример миграции Alembic для создания таблицы пользователей

Запуск приложения осуществляется с помощью Uvicorn. Конфигурация запуска представлена на рисунке 12. Польный код можно увидеть в ПРИЛОЖЕНИЕ А.

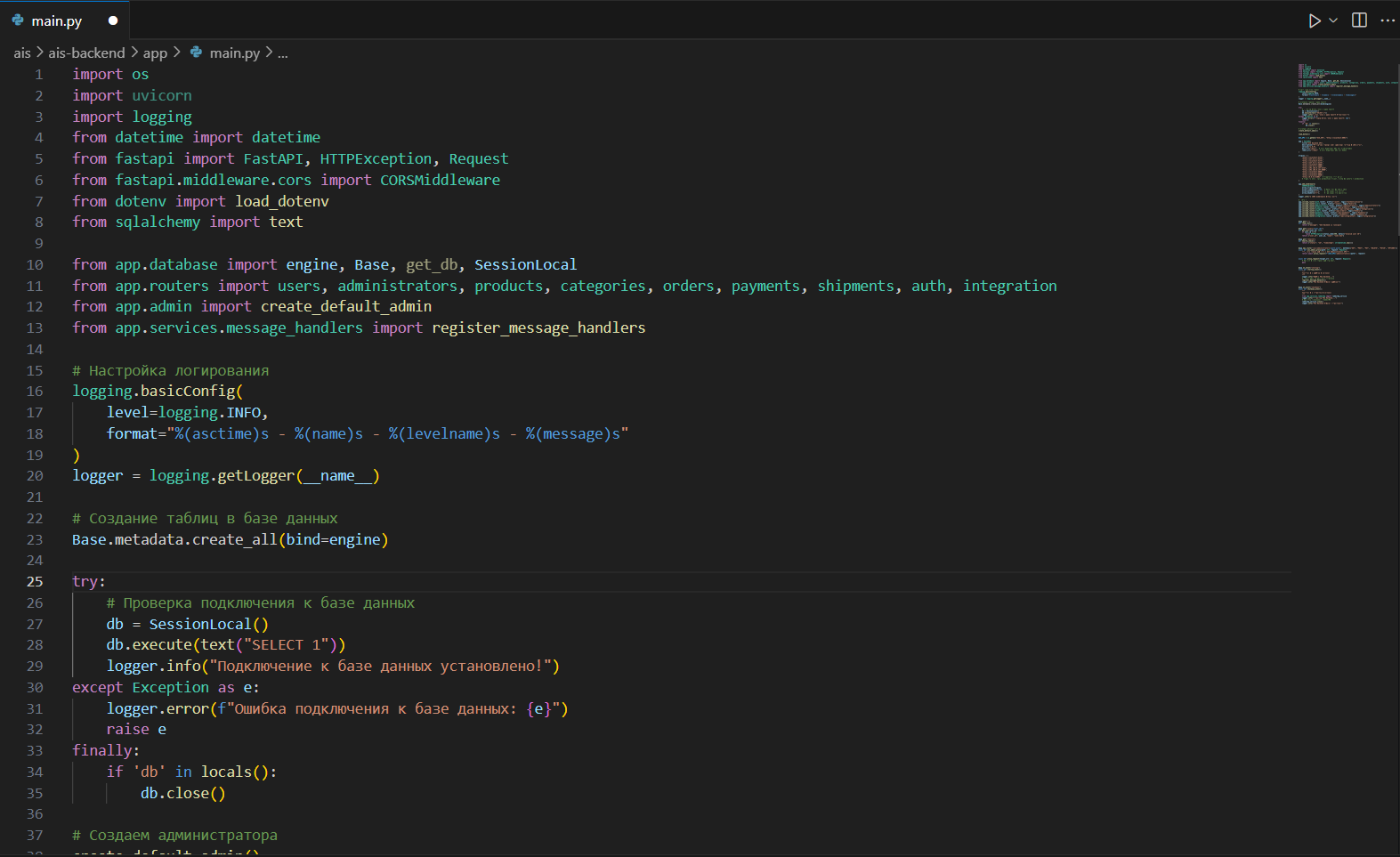


Рисунок 12 – Основной файл приложения с настройкой FastAPI

Одним из ключевых преимуществ FastAPI является простота создания и документирования API-интерфейсов. В рамках разработки автоматизированной информационной системы для ООО "Север-Рыба" была реализована система маршрутов с четким разделением по функциональным областям. Основные маршруты API представлены на рисунке 13.

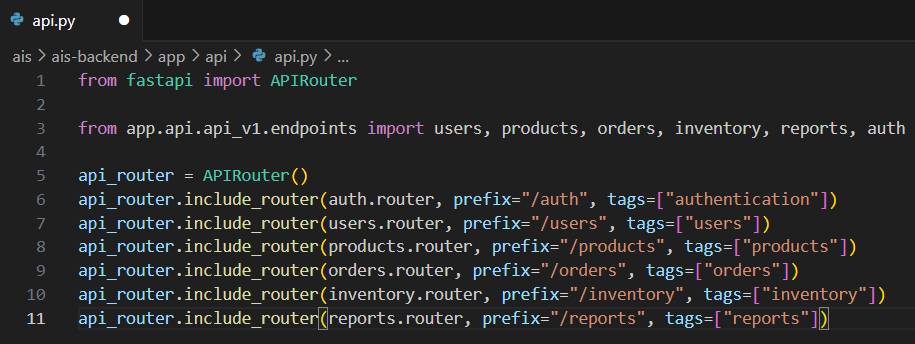


Рисунок 13 – Основные маршруты API системы

Для каждой функциональной области были разработаны отдельные эндпоинты, обеспечивающие выполнение соответствующих операций. Например, для управления продукцией был реализован набор маршрутов, позволяющих создавать, получать, обновлять и удалять информацию о продуктах, как показано на рисунке 14. Полный код страницы можно увидеть в ПРИЛОЖЕНИЕ Б.

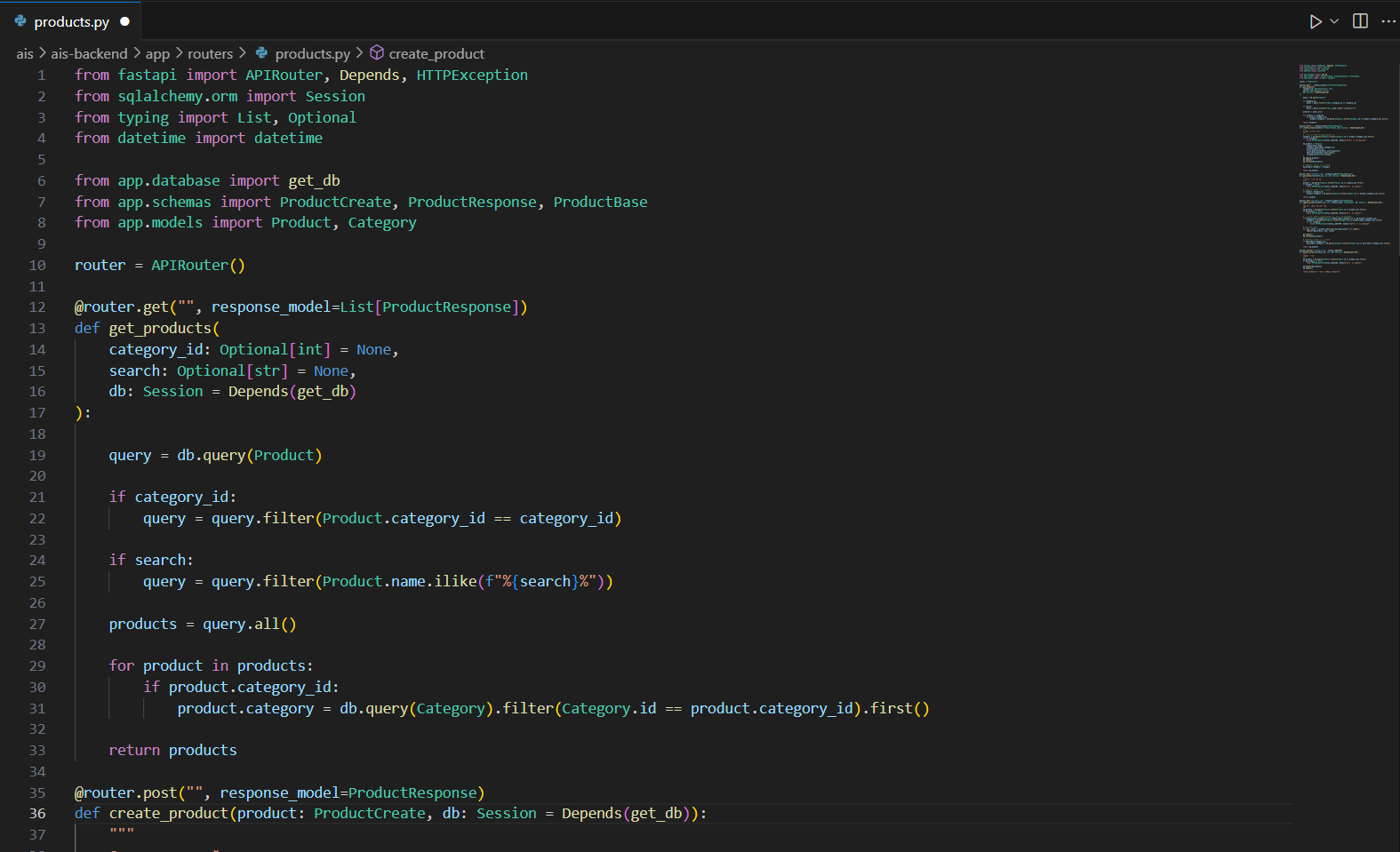


Рисунок 15 – Маршруты для управления продукцией

Для обеспечения безопасности доступа к системе была разработана комплексная система аутентификации и авторизации на основе JWT-токенов. Процесс аутентификации включает в себя проверку учетных данных пользователя и выдачу токена доступа с ограниченным сроком действия. Основные компоненты системы аутентификации представлены на рисунке 15.

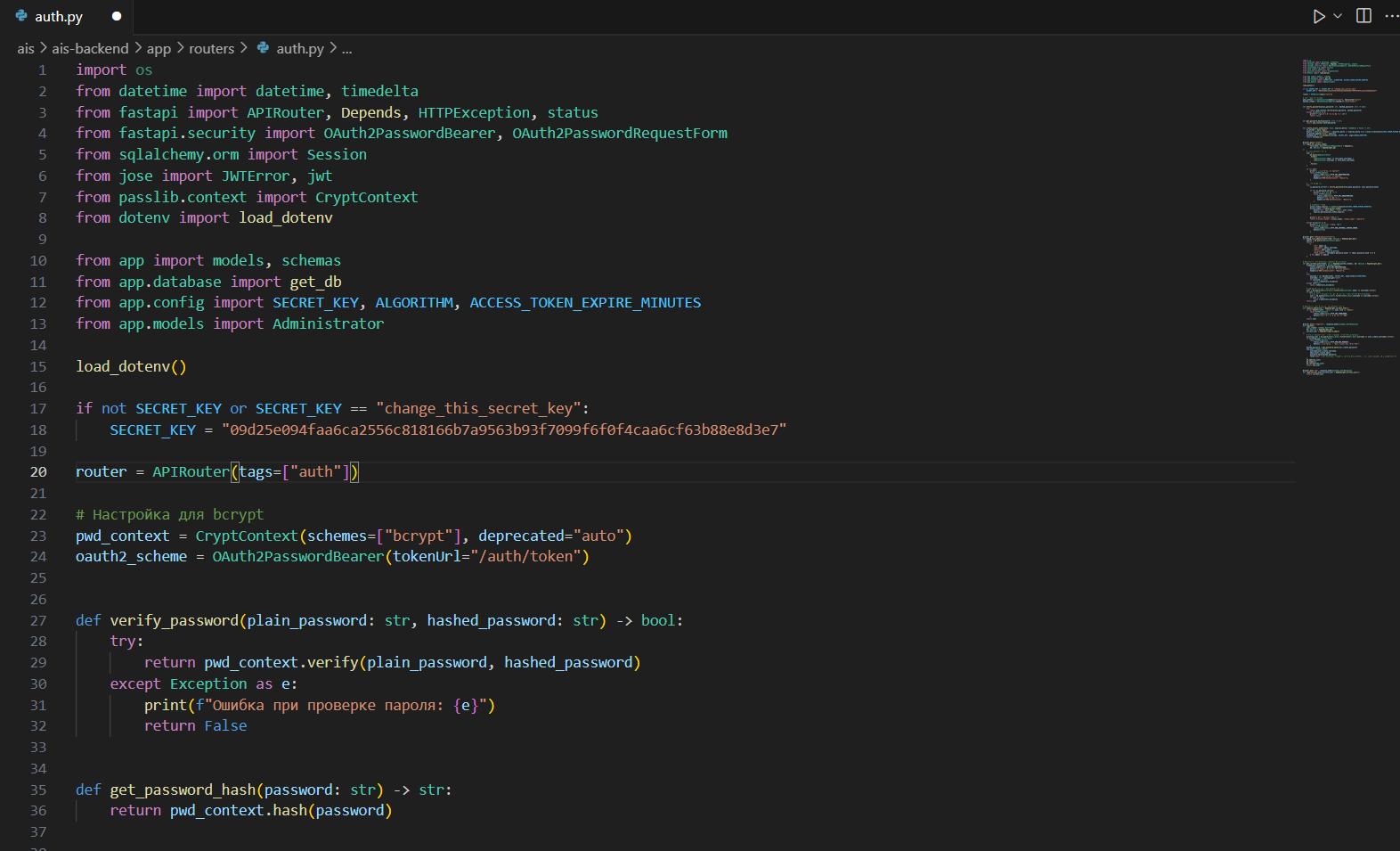


Рисунок 15 – Маршруты аутентификации

Одним из ключевых преимуществ FastAPI является автоматическая генерация документации API. Документация создается на основе типов аннотаций Python и комментариев к функциям обработчиков запросов. Для доступа к интерактивной документации Swagger UI используется URL /docs, а для документации в формате ReDoc – URL /redoc. Пример автоматически сгенерированной документации Swagger UI представлен на рисунке 20.

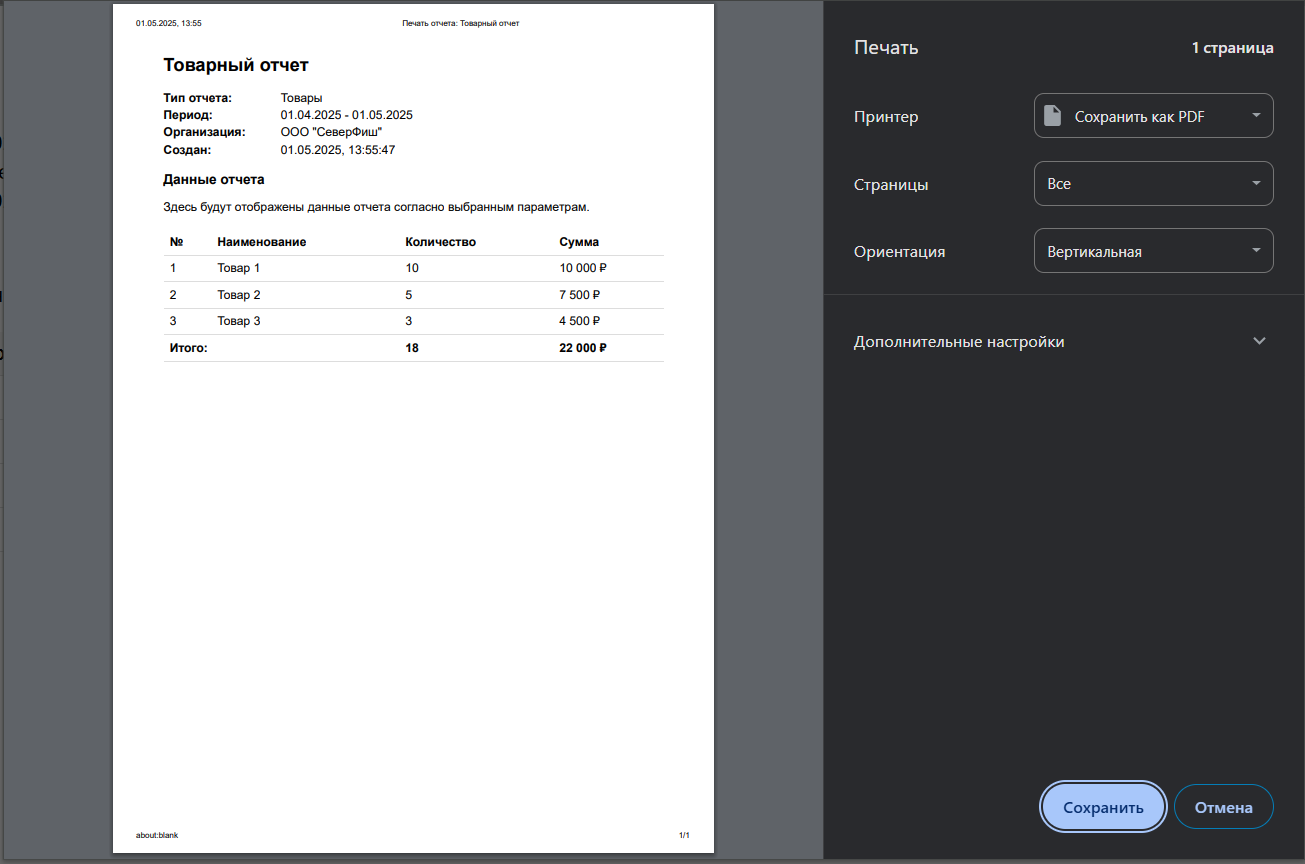


Рисунок 16 – Автоматически сгенерированная документация Swagger UI

С основной backend частью, мы закончили теперь можно переходить к клиентской.

## 3.2 Разработка frontend части с помощью React

Создание проекта началось с установки и настройки необходимых инструментов разработки. После создания базовой структуры проекта были установлены дополнительные библиотеки, необходимые для разработки полнофункционального приложения:

* npm install react-router-dom, устанавливает библиотеки маршрутизации
* npm install axios, устанавливает библиотеки для работы с API;
* npm install @mui/material @emotion/react @emotion/styled, установка UI-библиотеки для создания интерфейса;
* npm install date-fns, установка библиотек для работы с датами.

Для организации навигации в приложении использовалась библиотека React Router, которая позволяет определить маршруты и соответствующие им компоненты. В файле src/App.tsx была реализована основная структура маршрутизации, показанная на рисунке 17.

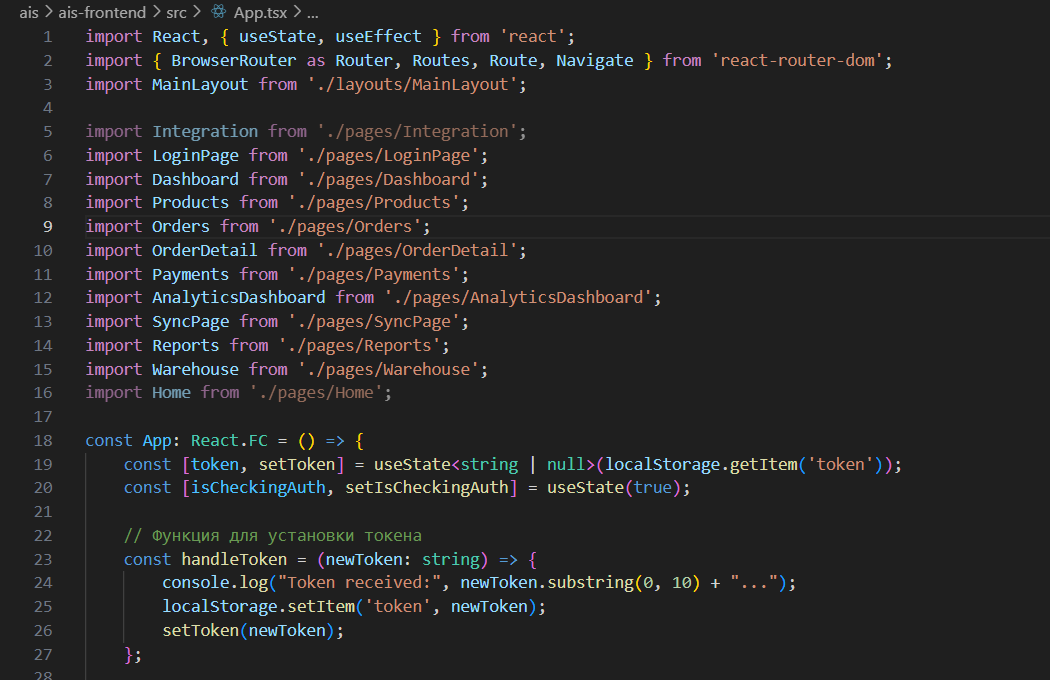


Рисунок 17 – Настройка маршрутизации в приложении

Для взаимодействия с backend-частью системы был разработан API-клиент на основе библиотеки Axios, который обеспечивает единый интерфейс для выполнения HTTP-запросов. Базовая настройка Axios представлена на рисунке 18.

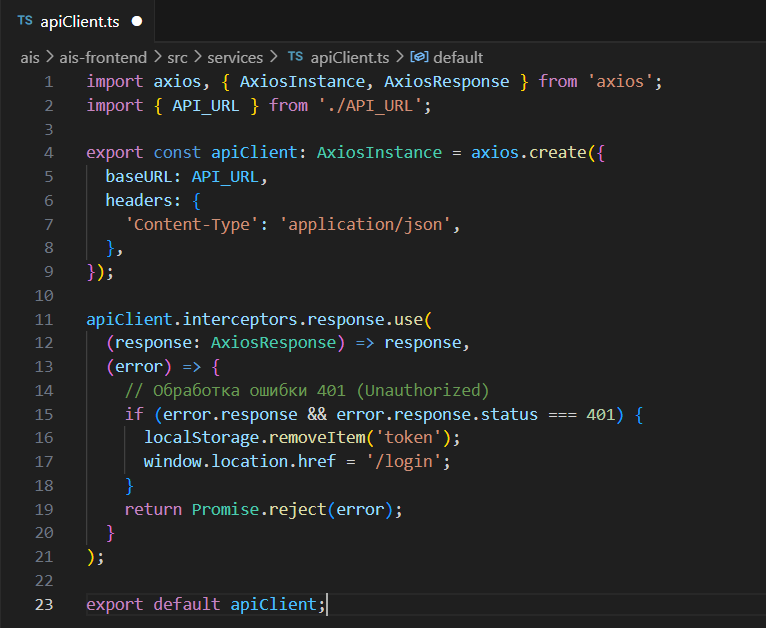


Рисунок 18 – Настройка Axios для взаимодействия с API

Пример сервиса для работы с товарами показан на рисунке 19.

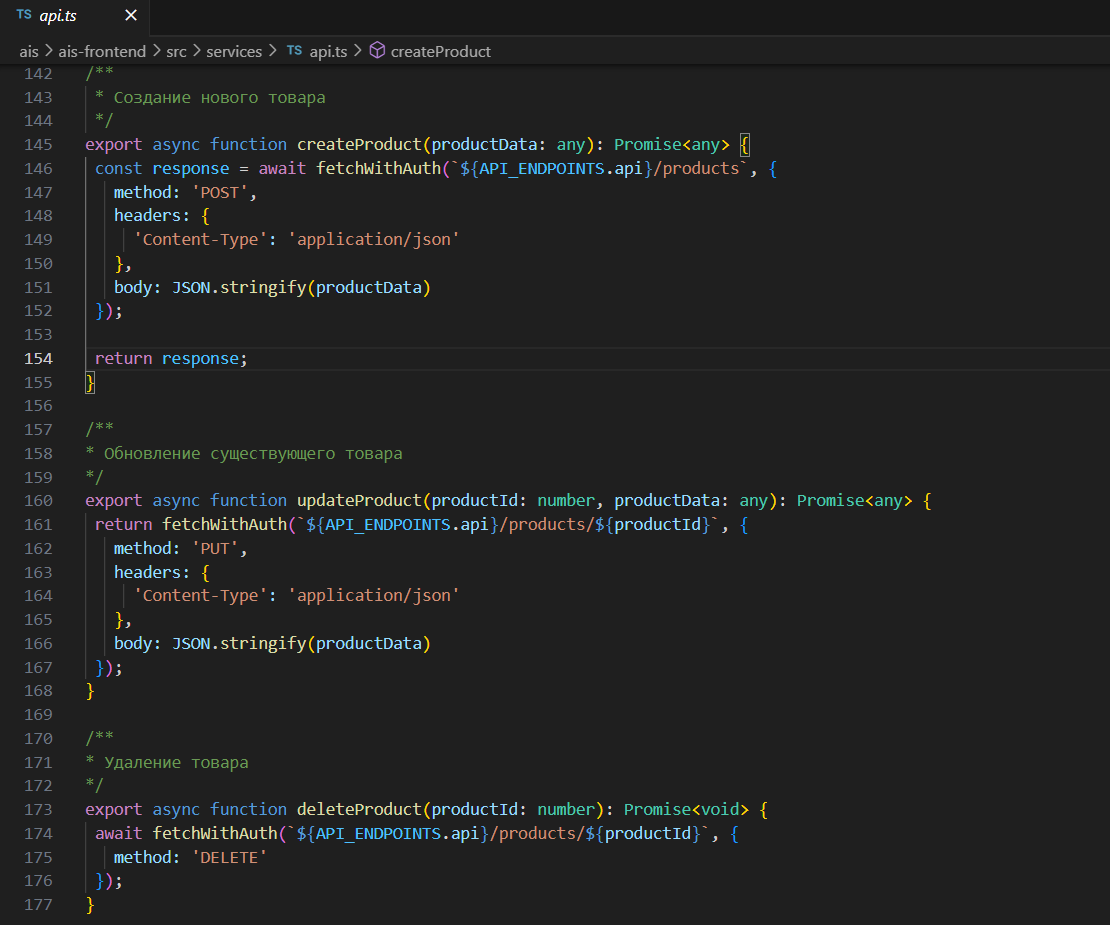


Рисунок 19 – Сервис для работы с продуктами

Интерфейс приложения построен на основе компонентного подхода, что обеспечивает модульность и повторное использование кода. Пример компонента формы для создания продукта показан на рисунке 20. Для реализации макета приложения был создан компонент MainLayout, который обеспечивает общую структуру всех страниц, включая навигационную панель и боковое меню. Этот компонент показан на рисунке 21.

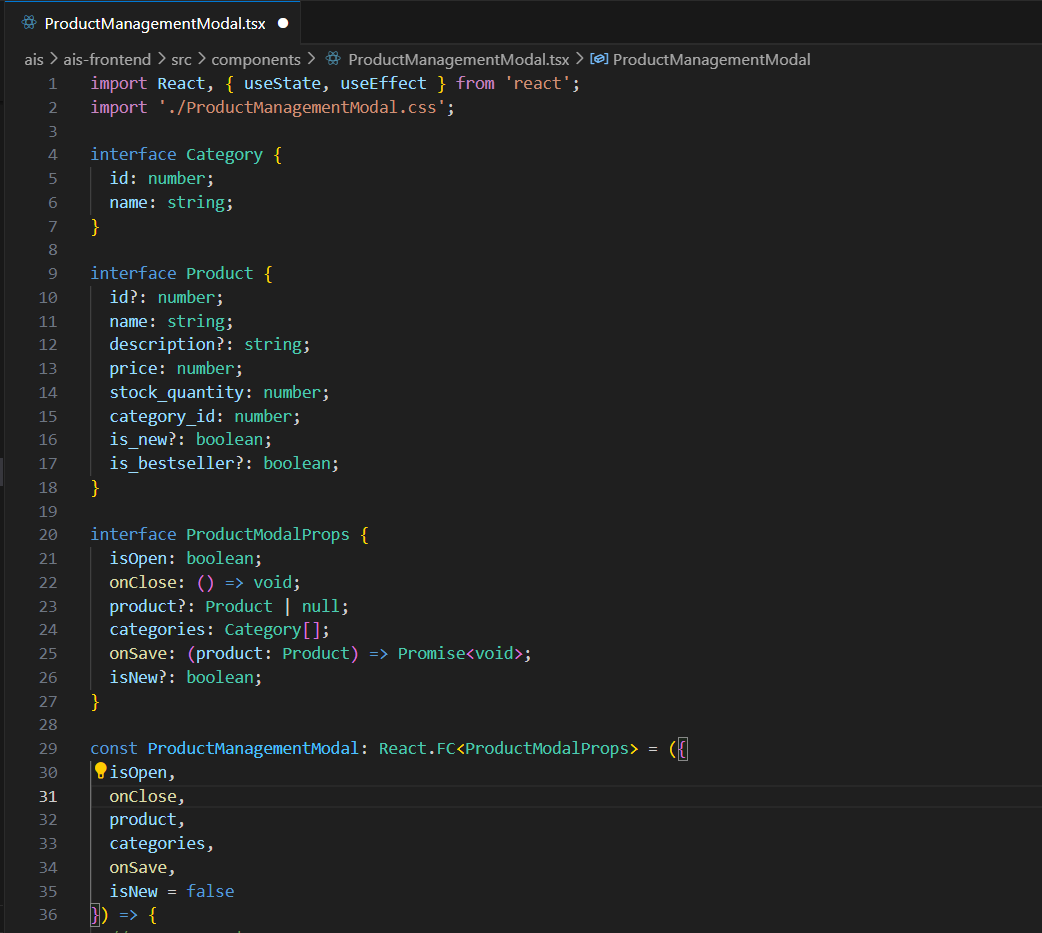


Рисунок 20 – Компонент формы для создания и редактирования продукта

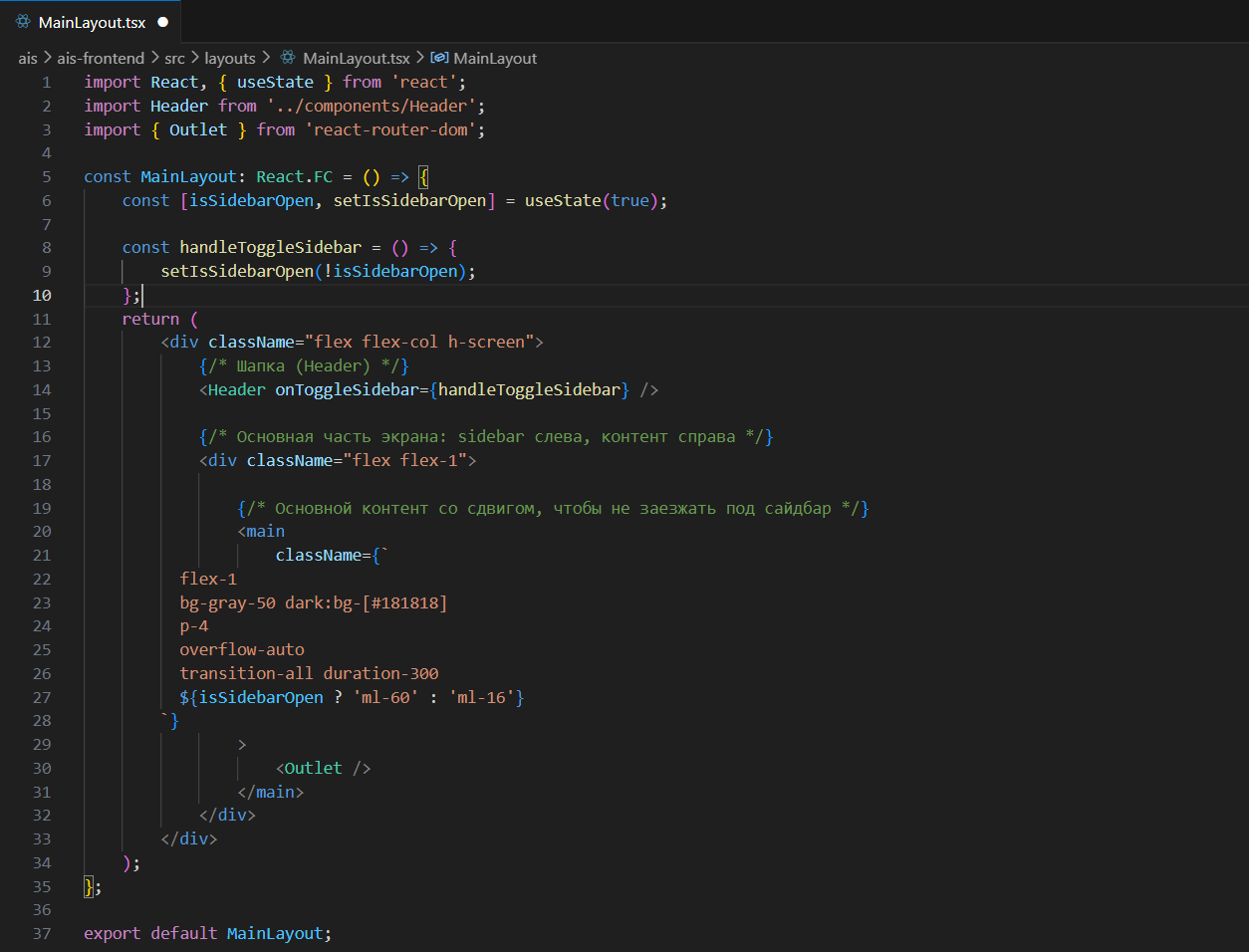


Рисунок 21 – Компонент макета приложения

## 3.3 Разработка базы данных с помощью PostgreSQL

Для разработки информационной системы ООО "Север-Рыба" была спроектирована и реализована база данных с использованием PostgreSQL. Выбор данной СУБД обусловлен её надёжностью, производительностью, соответствием стандартам SQL, а также поддержкой современных возможностей для разработки корпоративных приложений.

В результате анализа бизнес-процессов ООО "Север-Рыба" была спроектирована структура базы данных, включающая 14 основных таблиц. Структура базы данных представлена на рисунке 22.

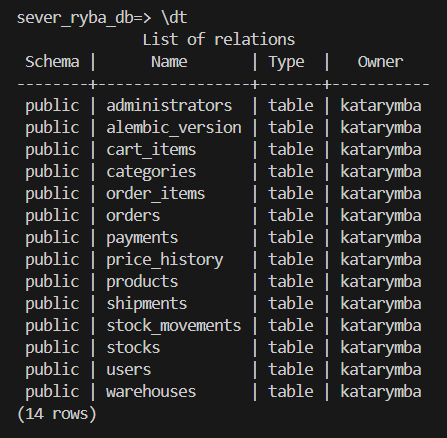


Рисунок 22 – Список таблиц в базе данных ООО "Север-Рыба"

Каждая таблица в структуре базы данных выполняет определённую функцию и отражает конкретный аспект деятельности предприятия:

* administrators – таблица для хранения данных администраторов системы с расширенными правами доступа.
* alembic\_version – служебная таблица для управления версиями схемы базы данных при миграциях.
* cart\_items – таблица для хранения товаров в корзине пользователей в процессе оформления заказа.
* categories – таблица категорий рыбной продукции с описанием и иерархией.
* order\_items – детализация заказов, включающая информацию о заказанных товарах, их количестве и цене.
* orders – основная таблица заказов клиентов с информацией о статусе, дате и общей стоимости.
* payments – таблица платежей с информацией о методе оплаты, статусе и связи с заказами.
* price\_history – история изменения цен на продукцию для аналитики и прогнозирования.
* products – каталог продукции компании с детальным описанием товаров.
* shipments – информация о доставках заказов клиентам.
* stock\_movements – учёт движения товаров на складе (поступления, списания).
* stocks – текущие остатки товаров на складах компании.
* users – данные о пользователях системы (клиенты, сотрудники).
* warehouses – информация о складах компании, их местоположении и характеристиках.

Для более глубокого понимания структуры и связей в базе данных рассмотрим детальное описание ключевых таблиц. На рисунке 23 представлена таблица products.

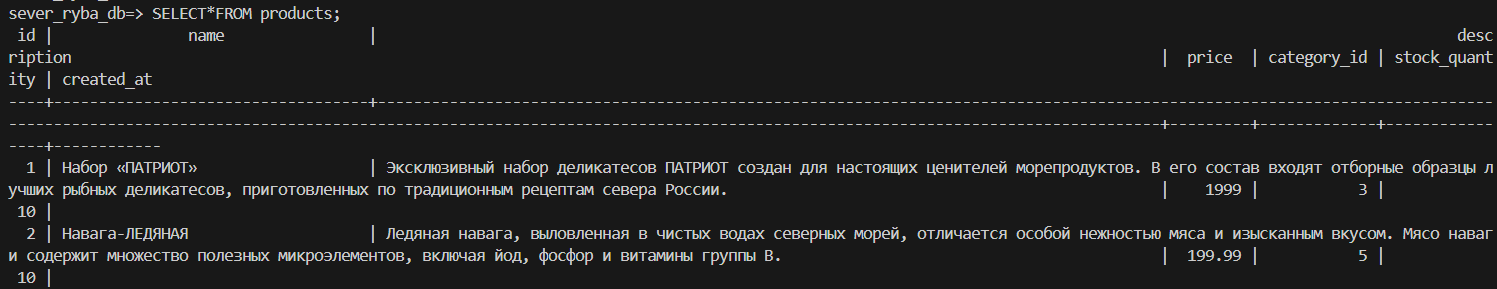


Рисунок 23 – Таблицы products

Таблица products является центральной в системе и хранит всю необходимую информацию о продукции ООО "Север-Рыба". Поля таблицы включают:

* основную информацию о товаре (название, описание, SKU-код);
* ценовые характеристики;
* весовые параметры продукции;
* статус доступности товара для заказа;
* метаданные (дата создания и обновления записи).

На рисунке 24 представлена таблица orders, которая хранит информацию о заказах клиентов

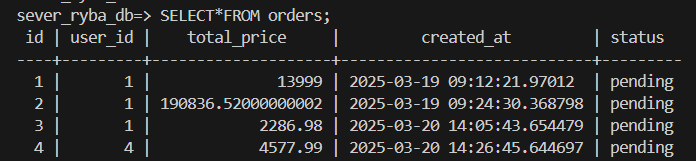


Рисунок 24 – Таблица orders

Для обеспечения целостности данных и отражения бизнес-логики в структуре базы данных были реализованы связи между таблицами. На рисунке 25 показана схема основных связей в базе данных.

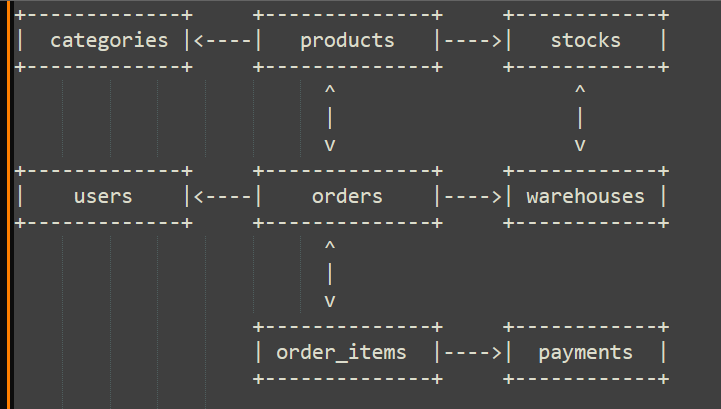


Рисунок 25 – Схема основных связей в базе данных

Таким образом, спроектированная и реализованная база данных PostgreSQL для ООО "Север-Рыба" обеспечивает эффективное хранение и обработку данных, необходимых для автоматизации бизнес-процессов предприятия. База данных учитывает все основные аспекты деятельности компании и предоставляет гибкие возможности для анализа данных и формирования отчетности.

## 3.4 Интерфейс веб–приложения

Доступ к системе осуществляется через экран авторизации, который требует ввода логина и пароля. После успешной авторизации пользователь попадает на главную страницу, которая отображает основную информацию в зависимости от роли пользователя. На рисунке 26 представлен экран авторизации системы.

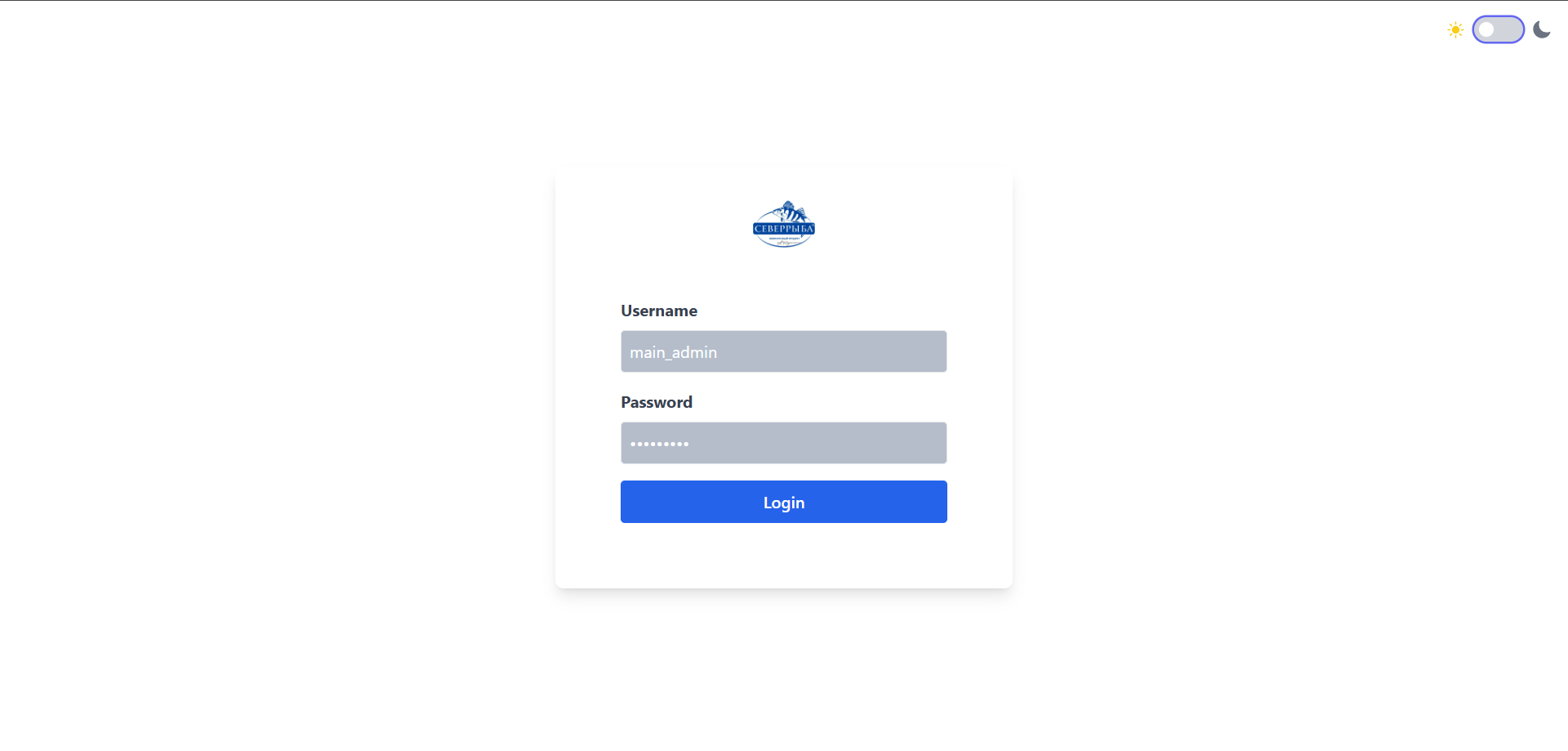


Рисунок 26 – Экран авторизации в информационной системе

После успешного входа в систему пользователь попадает на главную страницу с дашбордом, отображающим ключевую информацию и статистику, актуальную для соответствующей роли. На рисунке 27 представлен дашборд менеджера по продажам.

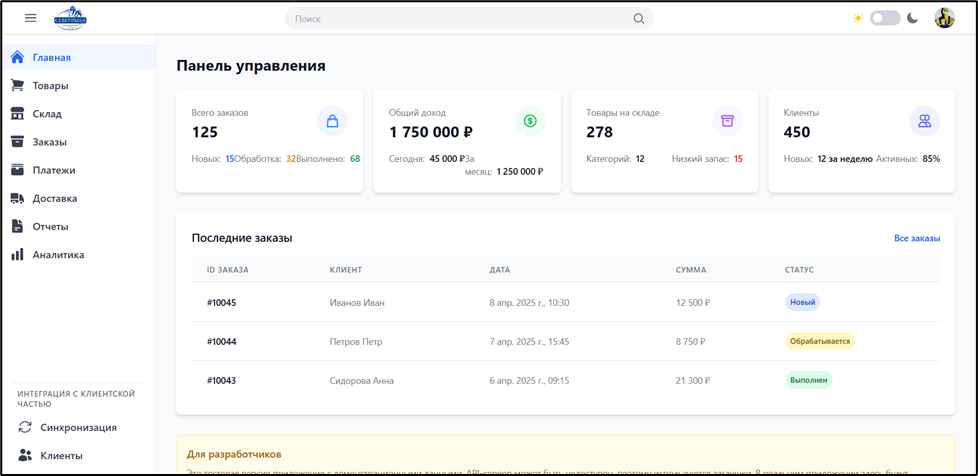


Рисунок 27 – Дашборд менеджера по продажам

Для эффективного управления запасами в системе предусмотрен интерфейс складского учета. На рисунке 28 представлен экран управления остатками на складе.

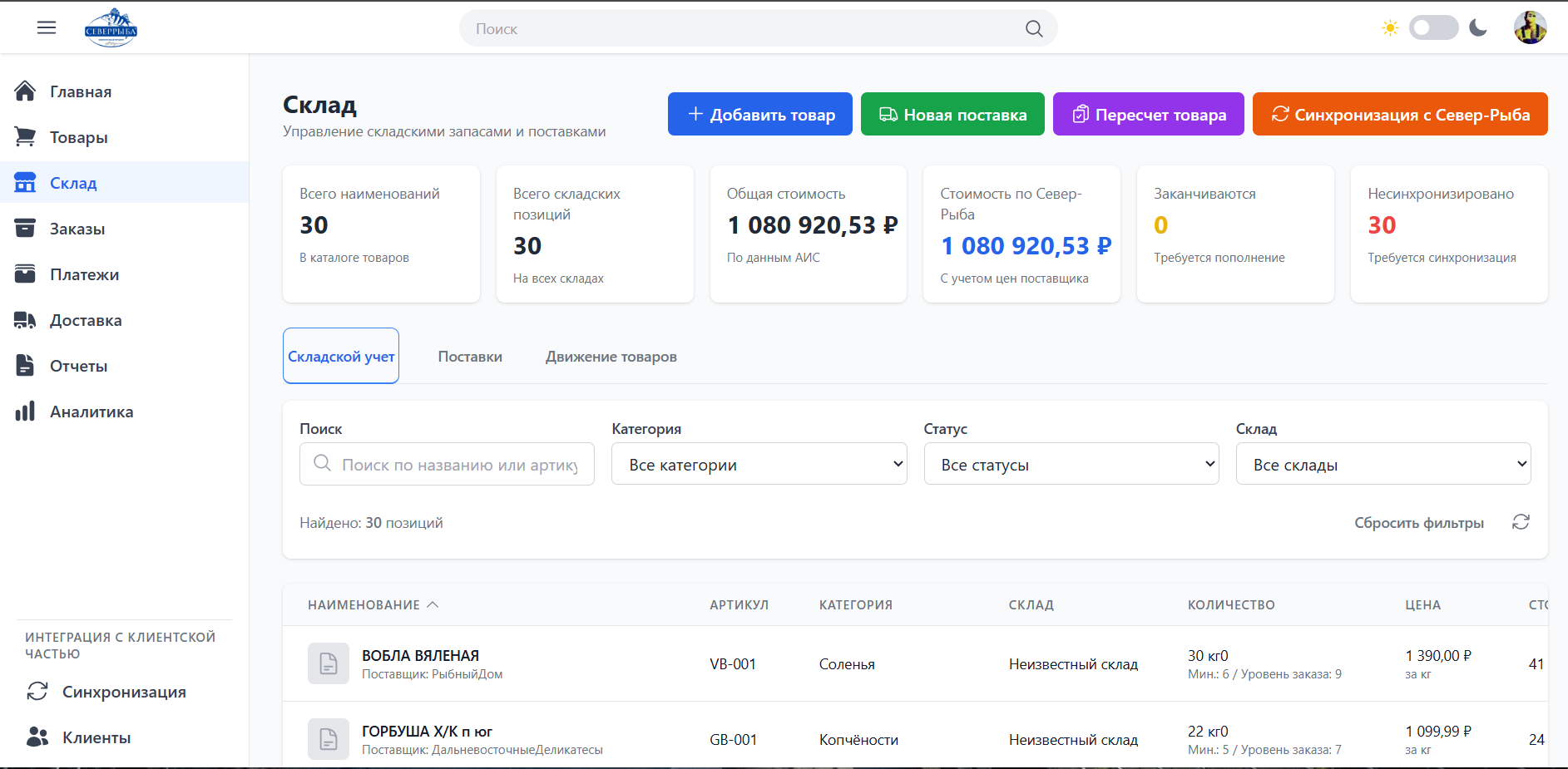


Рисунок 28 – Экран управления остатками на складе

При добавлении нового товара на склад используется форма, представленная на рисунке 29. Также есть форма для создания новой поставки, показанная на рисунке 30.

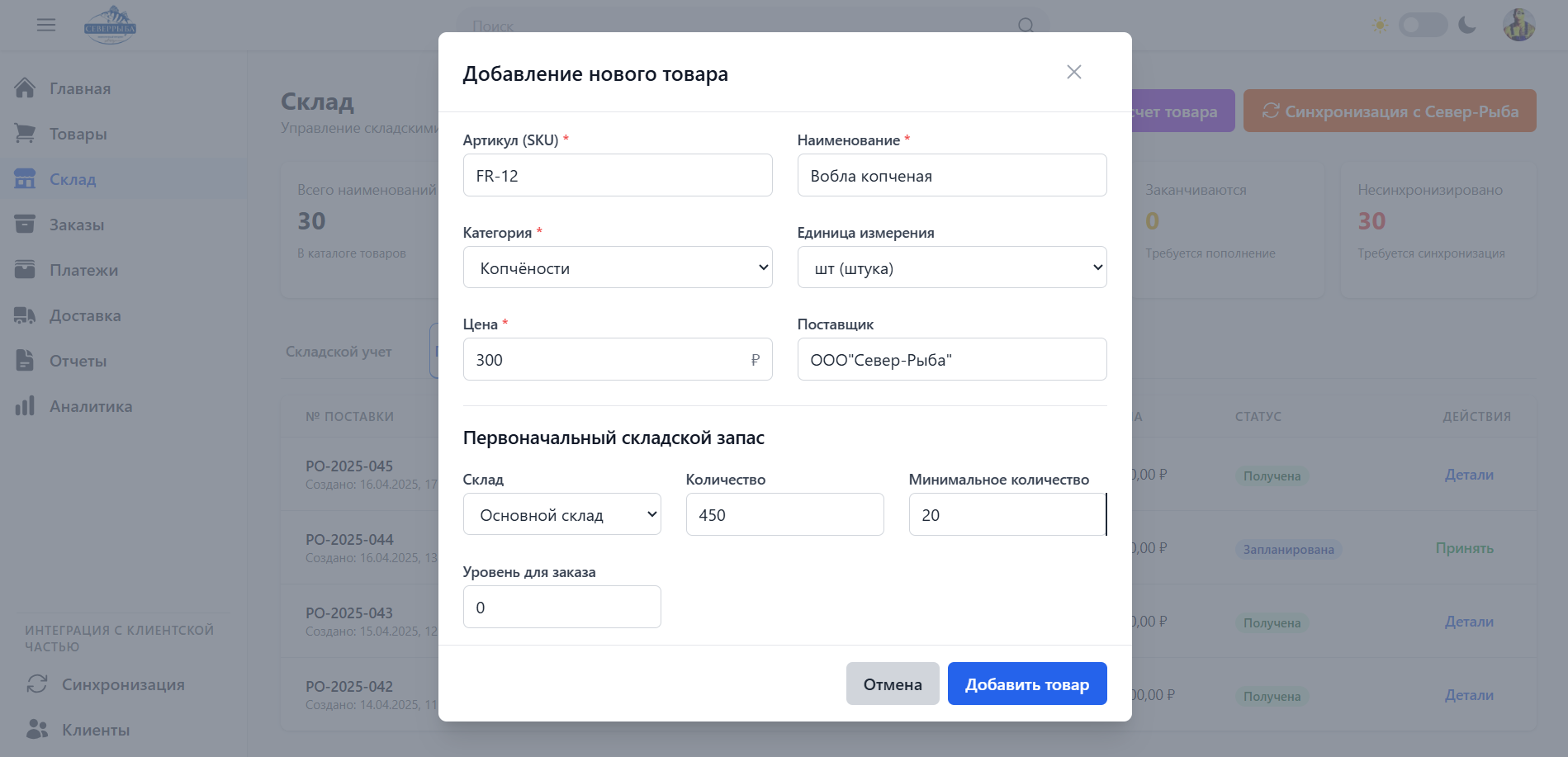


Рисунок 29 – Добавление нового товара на склад

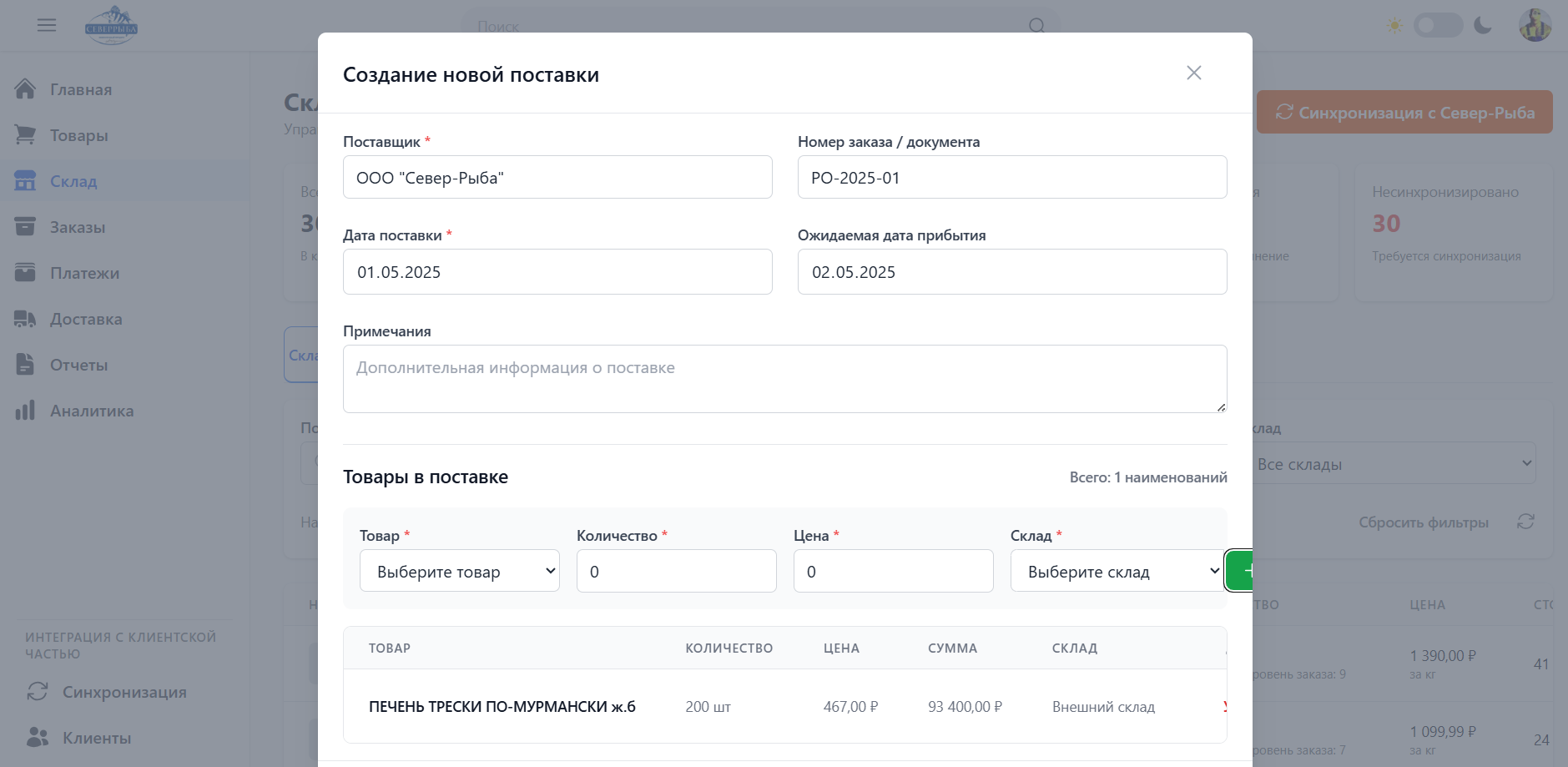


Рисунок 30 – Форма создания новой поставки

Для анализа деятельности предприятия и принятия управленческих решений в системе предусмотрен раздел отчетов. На рисунке 31 представлен экран с основными отчетами. При нажатии на кнопку «Создать отчет» появляется форма корректировки, отчеты в данном приложении можно создать из разных категорий, что показано на рисунке 32. Далее указываются необходимые параметры, изображенные на рисунке 33, и дополнительные настройки отчета такие как выбор ответственного лица и формат файла для будущего отчета. Также на странице реализованы функции печати, отправки и сохранения файлов после создания отчета. При печати автоматически открывается новая вкладка с отчетом и приложение принтера для дальнейшей настройки

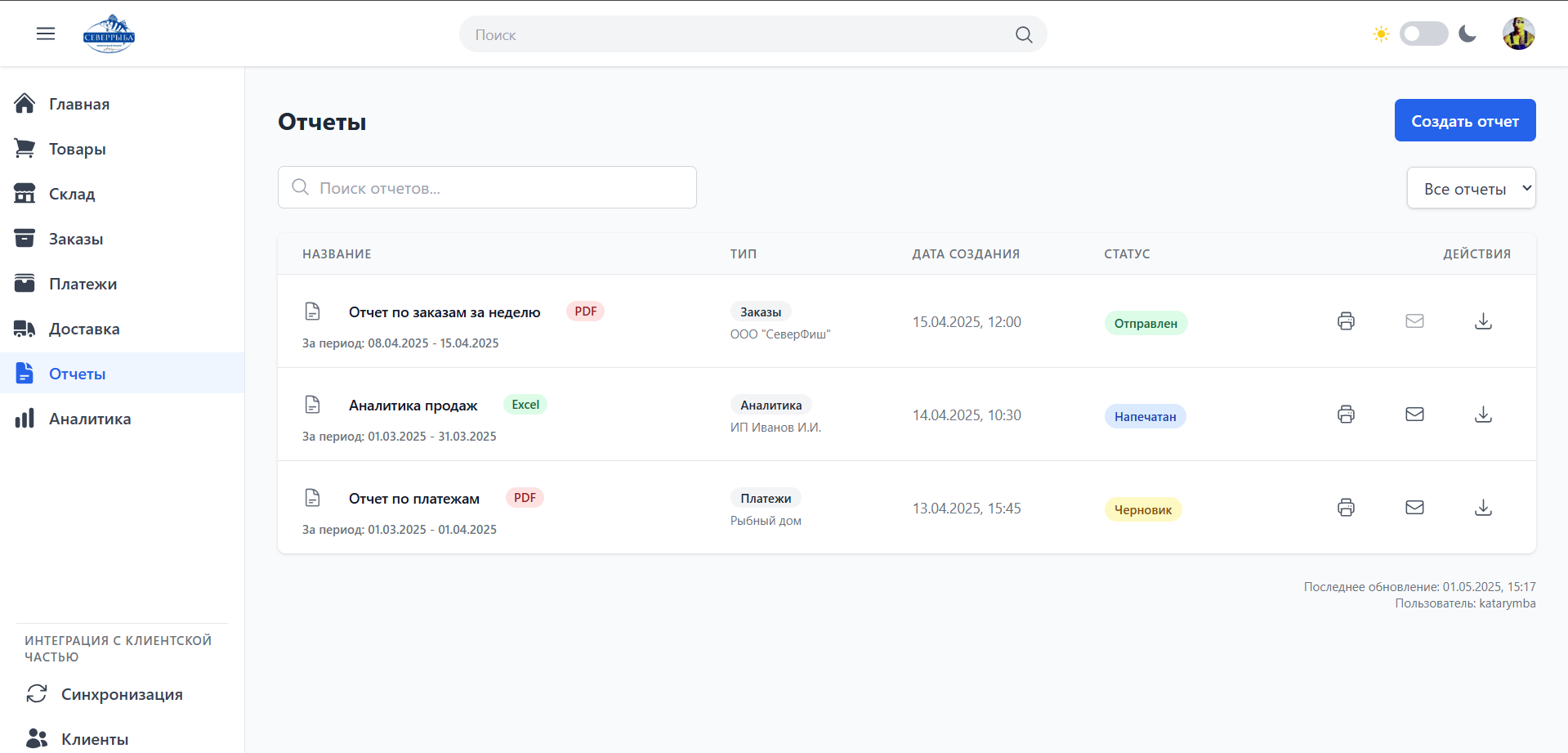


Рисунок 31 – Экран аналитики и отчетов

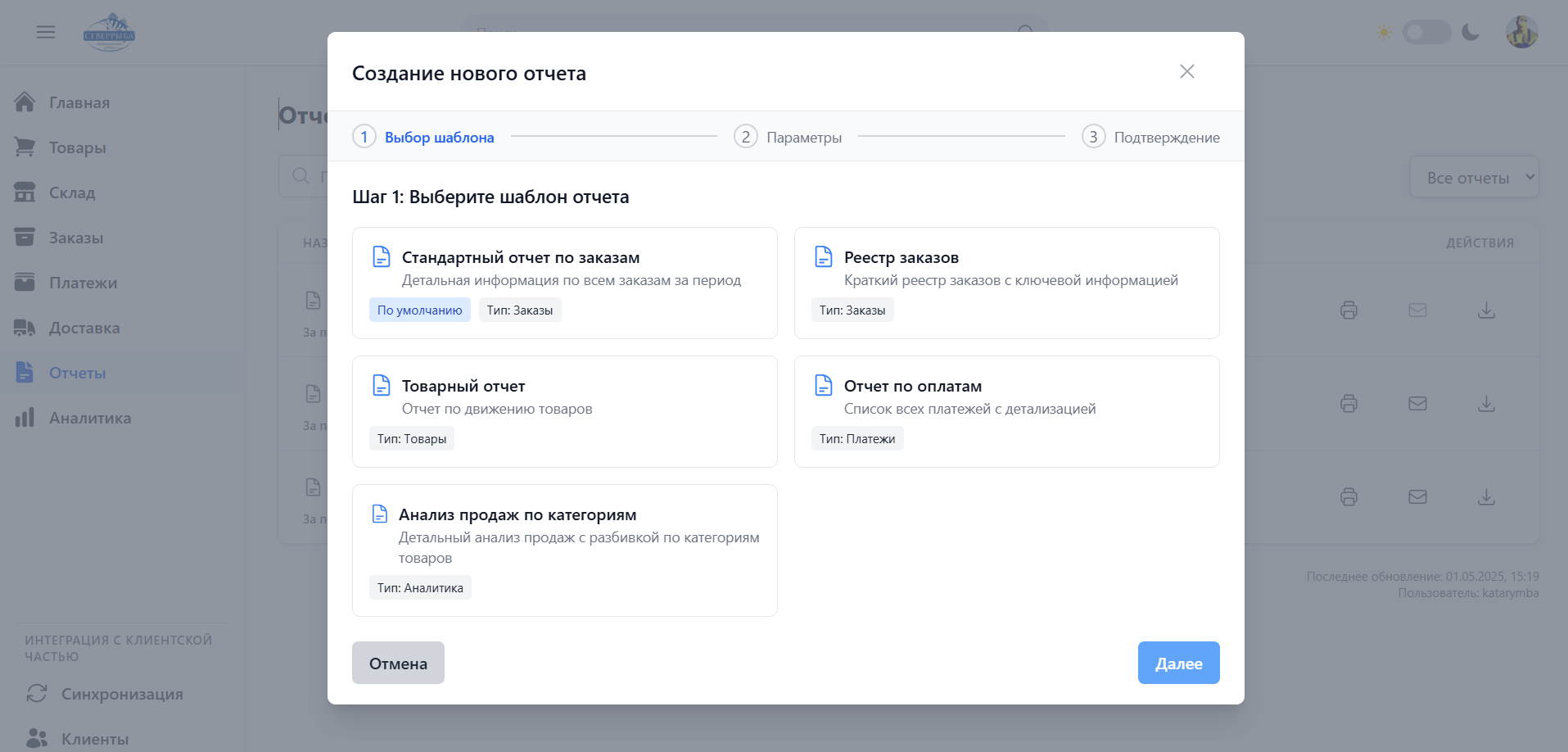


Рисунок 32 – Создание нового отчета

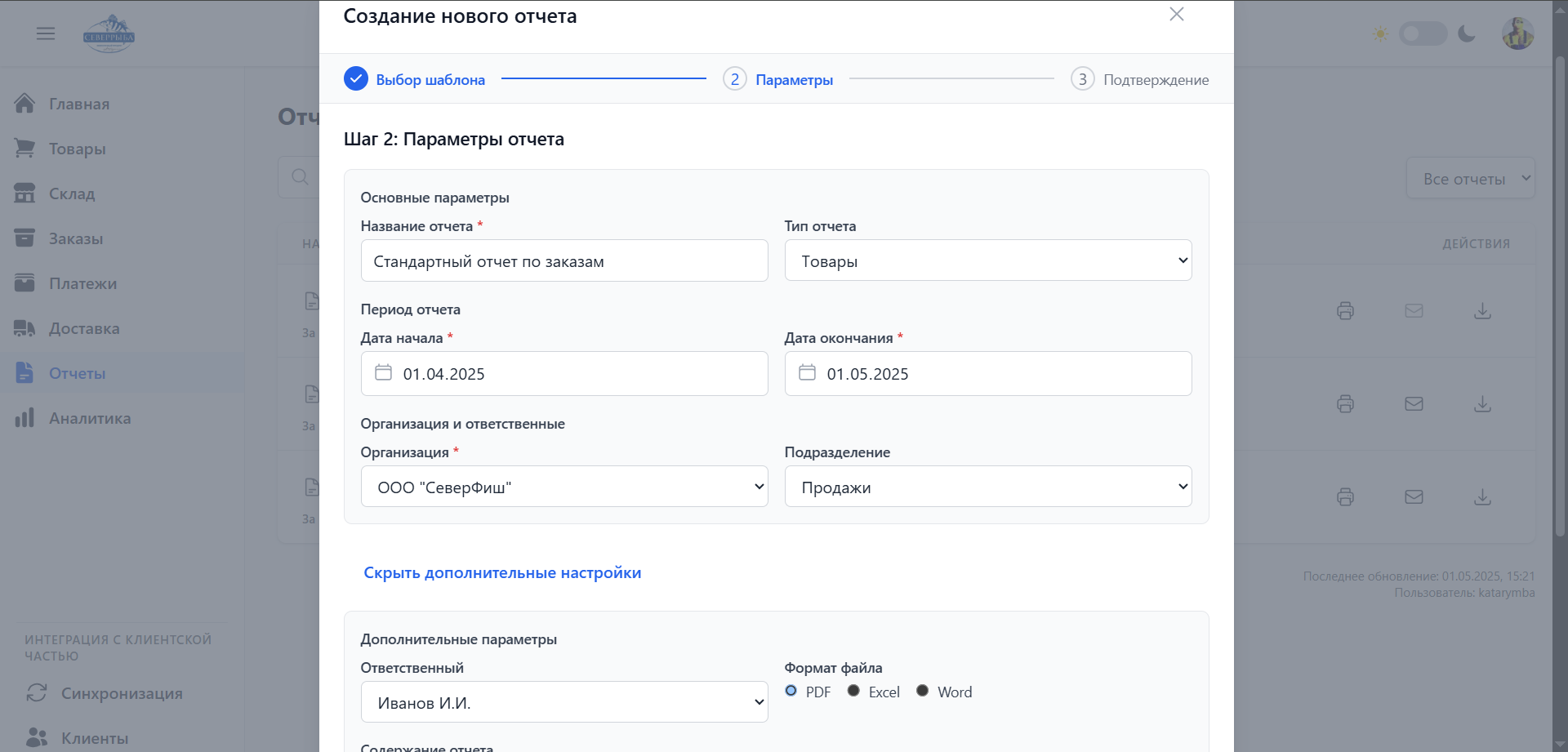


Рисунок 33 – Создание пользователя

На некоторых страницах есть кнопка «Синхронизация», которая сохраняет измененные данные и переносит их в клиентское приложение. Данная функция реализована для удобного онлайн заказа, а также для всегда актуальных данных и у клиента, и у работника.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения дипломной работы были успешно решены поставленные задачи, связанные с автоматизацией бизнес-процессов ООО "Север-Рыба" и созданием современной информационной системы для управления производственной и коммерческой деятельностью предприятия. Основные результаты имеют следующий вид:

* проведен анализ потребностей рыбоперерабатывающего предприятия в цифровизации бизнес-процессов, что подтвердило актуальность разработки системы;
* исследованы существующие аналоги, на основе чего сформулированы функциональные требования к системе, а именно управление каталогом продукции, учет заказов, складской учет, работа с клиентами и формирование аналитической отчетности;
* обоснован выбор технологического стека. Для backend выбран FastAPI благодаря его высокой производительности, асинхронности и современному подходу к разработке API. Для frontend применен React, обеспечивающий компонентный подход, гибкость разработки и эффективное управление состоянием приложения.

В теоретической части была рассмотрена структура организации ООО "Север-Рыба", проанализированы основные бизнес-процессы предприятия, а также исследованы современные технологии, подходящие для реализации информационной системы рыбоперерабатывающего предприятия.

В практической части дипломной работы были определены функции, реализуемые информационной системой ООО "Север-Рыба", разработана структура базы данных PostgreSQL, обеспечивающая эффективное хранение и обработку данных. Также обоснован выбор программных средств и технологий, обеспечивающих надёжную и эффективную работу системы. Все поставленные задачи были выполнены.

Дипломный проект подтвердил целесообразность и эффективность применения современных технологий (FastAPI, React, PostgreSQL) при создании информационных систем для предприятий пищевой промышленности. В результате получено удобное, безопасное и масштабируемое приложение, обеспечивающее цифровизацию бизнес-процессов рыбоперерабатывающего предприятия и повышающее эффективность его работы.

В первой главе была дана общая характеристика организации ООО "Север-Рыба", проанализированы требования к информационной системе и рассмотрены современные технологии для её разработки.

Во второй главе были спроектированы основные модули информационной системы, разработана архитектура приложения и определены основные функциональные возможности. Также был обоснован выбор FastAPI и React в качестве основных технологий для серверной и клиентской частей системы, и проведено экономическое обоснование разработки.

В третьей главе представлено описание разработки backend части на FastAPI, frontend части на React, структуры базы данных PostgreSQL и пользовательского интерфейса информационной системы, включая основные экраны для различных категорий пользователей.

Таким образом, реализованная информационная система может быть успешно внедрена в ООО "Север-Рыба" и адаптирована под различные бизнес-процессы предприятия. Полученные результаты подтверждают актуальность разработанной системы для повышения эффективности управления предприятием и оптимизации его производственной и коммерческой деятельности.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. ГОСТ 19.201–78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению
2. ГОСТ 34.601–90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания
3. Бер, Бибо jQuery. Подробное руководство по продвинутому JavaScript / Бибо Бер. – М.: Символ–плюс, 2023. – 720 c.
4. Волкова, В.Н. Теория информационных процессов и систем. Учебник и практикум (2–е издание) / В.Н. Волкова. – Москва : Издательство Юрайт, 2022. – 433 с.
5. Гарагусо, П. Vue.js 3: шаблоны проектирования и лучшие практики / Пабло Гарагусо. – Москва: Vue FAQ, 2023. – 350 с.
6. Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов / Джон Дакетт. – Москва: Library-IT, 2022. – 512 с.
7. Дакетт, Д. JavaScript и jQuery. Интерактивная веб–разработка / Д. Дакетт. – Москва : Эксмо, 2023. – 640 с.
8. Дронов, В.А. Laravel 9. Быстрая разработка веб-сайтов на PHP / Владимир Александрович Дронов – Москва: БХВ, 2023. – 747с.
9. Дронов, В.А. PHP и MySQL. 25 уроков для начинающих / Владимир Александрович Дронов. – Москва: Питер, 2022. – 384 с.
10. Дэвид, Флэнаган JavaScript. Подробное руководство / Флэнаган Дэвид. – М.: Символ–плюс, 2022. – 423 c.Гуриков, С.Р. Основы алгоритмизации и программирования на Python. Учебное пособие. / С.Р. Гуриков. – Москва : Инфра–М, 2022. – 343 с.
11. Йоретег, Х. Человеческий JavaScript / Хенрик Йоретег. – Москва: Proglib, 2022. – 350 с.
12. Кириченко, А.В., Дубовик, Е.В. Laravel для веб-разработчиков: практическое руководство по созданию профессиональных сайтов / А.В. Кириченко, Е.В. Дубовик. – Москва: Наука и техника, 2022. – 384 с.
13. Никсон, Робин Создаем динамические веб–сайты с помощью PHP, MySQL и JavaScript / Робин Никсон. – М.: Питер, 2022. – 496 c.
14. Павловская, Е.Э. Основы дизайна и композиции: современные концепции. Учебное пособие для среднего профессионального образования / Е.Э. Павловская. – Москва : Издательство Юрайт, 2022. – 119 с.
15. Раушмайер, А. Говорить на JavaScript / Аксель Раушмайер. – Москва: Proglib, 2022. – 400 с.
16. Стаффер, М. Laravel. Полное руководство. 3-е издание / Мэтт Стаффер. – Санкт-Петербург: Питер, 2024. – 512 с.
17. Татро, К. Создаем динамические веб–сайты на PHP (4–е издание) / К. Татро. – Санкт–Петербург : Питер, 2022. – 544 с.
18. Фримен, Э., Фримен, Э. Изучаем HTML, XHTML и CSS / Эрик Фримен, Элизабет Фримен. – Москва: CodeLibrary, 2022. – 480 с.
19. Фрэйн, Б. Отзывчивый дизайн на HTML5 и CSS3 для любых устройств / Б. Фрэйн. – Санкт–Петербург : Питер, 2022. – 336 с.
20. Хэнчетт, Э., Листуон, Б. Vue.js в действии / Эрик Хэнчетт, Бенджамин Листуон. – Москва: Литрес, 2022. – 304 с.
21. AllFussion Process Modeler [Электронный ресурс]. – Режим доступа: http://www.interface.ru/.

# **ПРИЛОЖЕНИЯ**

**ПРИЛОЖЕНИЕ А**

## **Код запуска приложения**

import os

import logging

from datetime import datetime

from fastapi import FastAPI, HTTPException, Request

from fastapi.middleware.cors import CORSMiddleware

from dotenv import load\_dotenv

from sqlalchemy import text

from app.database import engine, Base, get\_db, SessionLocal

from app.routers import users, administrators, products, categories, orders, payments, shipments, auth, integration

from app.admin import create\_default\_admin

from app.services.message\_handlers import register\_message\_handlers

logging.basicConfig(

    level=logging.INFO,

    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s"

)

logger = logging.getLogger(\_\_name\_\_)

Base.metadata.create\_all(bind=engine)

try:

    db = SessionLocal()

    db.execute(text("SELECT 1"))

    logger.info("Подключение к базе данных установлено!")

except Exception as e:

    logger.error(f"Ошибка подключения к базе данных: {e}")

    raise e

finally:

    if 'db' in locals():

        db.close()

create\_default\_admin()

load\_dotenv()

AIS\_API = os.getenv("AIS\_API", "http://localhost:8001")

app = FastAPI(

    title="AIS Backend API",

    description="Автоматизированная информационная система ",

    version="1.0.0",

    docs\_url="/docs",

    redoc\_url="/redoc"

)

origins = [

    "http://localhost:5173",

    "http://127.0.0.1:5173",

    "http://localhost:5174",

    "http://127.0.0.1:5174",

    "http://localhost:3000",

    "http://127.0.0.1:3000",

    "http://192.168.0.157:5173",

    "http://192.168.0.157:8000",

    "http://localhost:8000",

    "http://localhost:8001",

    "http://localhost:8080",

    "http://0.0.0.0:8001" ]

app.add\_middleware(

    CORSMiddleware,

    allow\_origins=origins,

    allow\_credentials=True,

    allow\_methods=["\*"],

    allow\_headers=["\*"],

)

logger.info("CORS middleware подключен!")

app.include\_router(auth.router, prefix="/auth", tags=["Authentication"])

app.include\_router(users.router, prefix="/users", tags=["Users"])

app.include\_router(administrators.router, prefix="/administrators", tags=["Administrators"])

app.include\_router(products.router, prefix="/api/products", tags=["Products"])

app.include\_router(categories.router, prefix="/api/categories", tags=["Categories"])

app.include\_router(orders.router, prefix="/api/orders", tags=["Orders"])

app.include\_router(payments.router, prefix="/api/payments", tags=["Payments"])

app.include\_router(shipments.router, prefix="/shipments", tags=["Shipments"])

app.include\_router(integration.router, prefix="/api/integration", tags=["Integration"])

@app.get("/")

def read\_root():

    return {"message": "AIS Backend is running"}

@app.get("/users/{user\_id}")

def get\_user(user\_id: int):

    if user\_id <= 0:

        raise HTTPException(status\_code=400, detail="Invalid user ID")

    return {"user\_id": user\_id, "name": "John Doe"}

@app.get("/health")

def health\_check():

    return {"status": "ok", "timestamp": str(datetime.now())}

@app.api\_route("/ais/administrators/{path:path}", methods=["GET", "POST", "PUT", "DELETE", "PATCH", "OPTIONS"])

async def ais\_admin\_proxy(path: str, request: Request):

    return await proxy\_request(f"{AIS\_API}/administrators/{path}", request)

async def proxy\_request(target\_url: str, request: Request):

    pass

@app.on\_event("startup")

async def startup\_event():

    logger.info("Запуск АИС Backend...")

    # Регистрация обработчиков сообщений

    register\_message\_handlers()

    logger.info("АИС Backend успешно запущен!")

@app.on\_event("shutdown")

async def shutdown\_event():

    from app.services.rabbitmq import rabbitmq\_service

    logger.info("Остановка АИС Backend...")

    rabbitmq\_service.close()

    logger.info("АИС Backend успешно остановлен!")

**ПРИЛОЖЕНИЕ Б**

## **Код управления маршрутами продукции**

# ais/ais-backend/app/routers/products.py

from fastapi import APIRouter, Depends, HTTPException

from sqlalchemy.orm import Session

from typing import List, Optional

from datetime import datetime

from app.database import get\_db

from app.schemas import ProductCreate, ProductResponse, ProductBase

from app.models import Product, Category

router = APIRouter()

@router.get("", response\_model=List[ProductResponse])

def get\_products(

category\_id: Optional[int] = None,

search: Optional[str] = None,

db: Session = Depends(get\_db)

):

query = db.query(Product)

if category\_id:

query = query.filter(Product.category\_id == category\_id)

if search:

query = query.filter(Product.name.ilike(f"%{search}%"))

products = query.all()

for product in products:

if product.category\_id:

product.category = db.query(Category).filter(Category.id == product.category\_id).first()

return products

@router.post("", response\_model=ProductResponse)

def create\_product(product: ProductCreate, db: Session = Depends(get\_db)):

category = db.query(Category).filter(Category.id == product.category\_id).first()

if not category:

raise HTTPException(status\_code=404, detail="Категория не найдена")

db\_product = Product(

name=product.name,

category\_id=product.category\_id,

price=product.price,

stock\_quantity=product.stock\_quantity,

description=product.description,

created\_at=datetime.utcnow()

)

db.add(db\_product)

db.commit()

db.refresh(db\_product)

db\_product.category = category

return db\_product

@router.get("/{product\_id}", response\_model=ProductResponse)

def get\_product(product\_id: int, db: Session = Depends(get\_db)):

product = db.query(Product).filter(Product.id == product\_id).first()

if product is None:

raise HTTPException(status\_code=404, detail="Товар не найден")

if product.category\_id:

product.category = db.query(Category).filter(Category.id == product.category\_id).first()

return product

@router.put("/{product\_id}", response\_model=ProductResponse)

def update\_product(product\_id: int, product\_data: ProductBase, db: Session = Depends(get\_db)):

db\_product = db.query(Product).filter(Product.id == product\_id).first()

if db\_product is None:

raise HTTPException(status\_code=404, detail="Товар не найден")

if product\_data.category\_id and product\_data.category\_id != db\_product.category\_id:

category = db.query(Category).filter(Category.id == product\_data.category\_id).first()

if not category:

raise HTTPException(status\_code=404, detail="Категория не найдена")

for key, value in product\_data.dict(exclude\_unset=True).items():

setattr(db\_product, key, value)

db.commit()

db.refresh(db\_product)

if db\_product.category\_id:

db\_product.category = db.query(Category).filter(Category.id == db\_product.category\_id).first()

return db\_product

@router.delete("/{product\_id}", status\_code=204)

def delete\_product(product\_id: int, db: Session = Depends(get\_db)):

db\_product = db.query(Product).filter(Product.id == product\_id).first()

if db\_product is None:

raise HTTPException(status\_code=404, detail="Товар не найден")

db.delete(db\_product)

db.commit()

return {"detail": "Товар успешно удален"}