МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность профиля / специализации)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Разработка и администрирование REST веб-приложения службы логистики»

Студент К.Д. Мамонов
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н., Н.В. Хрипунов
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент С.А. Гудкова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2024

Аннотация

Тема выпускной квалификационной работы – «Разработка и администрирование REST веб-приложения службы логистики».

Актуальность темы ВКР обусловлена развитием технологий и их интеграцией в управление бизнес-процессами в компаниях и на предприятиях различных сфер деятельности.

Объектом исследования является деятельность отдела логистики предприятия ИП «Стеблев Д.В».

Предмет исследования – автоматизация процесса формирования и обработки заявок на поставку комплектующих.

Целью работы является разработка веб-приложения, позволяющего автоматизировать процессы формирования и обработки заявок на поставку комплектующих отделом логистики предприятия.

В первой главе работы описана организационная структура предприятия ИП «Стеблев Д.В», выполнен анализ деятельности отдела логистики с использованием методологий IDEF0, описана сущность задачи на разработку приложения и произведен анализ уже существующих аналогов подобных систем.

Вторая глава содержит логическое моделирование разрабатываемого веб-приложения, проектируется хранилище данных для веб-приложения, и формулируются требования к аппаратному обеспечения для развёртывания приложена на серверах предприятия.

В третьей главе описываются выбранные при разработке технологии, описывается разработанное веб-приложение. Так же, глава содержит демонстрацию его работа: функции, заполняемые формы, передаваемые и получаемые данные, принцип регистрации и авторизации.

Работа содержит: 58 страниц, 31 рисунок, 6 таблиц, 3 приложения и список из 20 используемых источников.

Abstract

The topic of the final qualification work is "Development and administration of the REST web application of the logistics service".

The relevance of the topic of the WRC is due to the development of technologies and their integration into business process management in companies and enterprises of various fields of activity.

The object of the study is the activity of the logistics department of the enterprise IP "Steblev D.V.".

The subject of the study is automation of the process of forming and processing applications for the supply of components.

The purpose of the work is to develop a web application that allows you to automate the processes of forming and processing applications for the supply of components by the logistics department of the enterprise.

Structurally, the work is presented with an introduction, three chapters, a conclusion, a list of references and sources, as well as four appendices.

The first chapter of the work describes the organizational structure of the enterprise IP "Steblev D.V.", analyzes the activities of the logistics department using IDEF0 methodologies, describes the essence of the task of developing an application and analyzes existing analogues of similar systems.

The second chapter contains a logical modeling of the web application being developed, a data warehouse for the web application is designed, and hardware requirements for deploying the application on enterprise servers are formulated.

The third chapter describes the technologies selected during development, describes the developed web application. Also, the chapter contains a demonstration of its work: functions, forms to be filled out, data transmitted and received, the principle of registration and authorization.

The work contains: 68 pages, 31 figures, 6 tables, 3 appendices and a list of 20 sources used.

Оглавление

Введение

В современном мире трудно представить, что компания или предприятие не использует программное обеспечение для оптимизации своих бизнес-процессов, поскольку программное обеспечение позволяет улучшить определенные рабочие процессы внутри компании или предприятия.

Темой данной выпускной квалификационной работы является: «Разработка и администрирование REST веб-приложения службы логистики».

Объектом же исследования является деятельность отдела логистики предприятия ИП «Стеблев Д.В».

Предметом исследования является автоматизация процессов оформления и обработки заявок на поставку комплектующих отделом логистики предприятия.

Отдел логистики является важным структурным подразделением, от продуктивности его деятельности может зависеть деятельность всего предприятия, поэтому разработка такого веб-приложения является актуальной и приведёт к усовершенствованию производственных процессов всего предприятия.

Внедрение веб-приложения в систему отдела логистики позволит ускорить процессы формирования и обработки заявок на поставку комплектующих, улучшить и упростить процесс подготовки отчетов, а также сведёт к минимуму возможные в них ошибки.

Чтобы достичь цели, необходимо рассмотреть и выполнить следующие задачи:

- изучить деятельность предприятия ИП «Стеблев Д.В», его организационную структуру и деятельность объекта автоматизации;
- проанализировать существующие программные решения, которые потенциально могли бы быть внедрены в систему предприятия;
- обосновать постановку задачи на разработку и администрирование REST веб-приложения службы логистики;

- смоделировать концептуальную модель бизнес-процесс «как есть» и «как будет» для деятельности предприятия;
- спроектировать разрабатываемое приложение;
- спроектировать модель данных;
- спроектировать базу данных приложения;
- произвести демонстративное тестирование разработанного веб-приложения.

Первая глава содержит описание предприятия ИП «Стеблев Д.В», характеристику объекта автоматизации, в соответствии с темой ВКР, отдела логистики. Так же производится концептуальное проектирование бизнес-процессов предприятия, сравниваются аналоги среду уже реализованных программных решений. Результатом сравнения является основание для разработки собственного веб-приложения для службы логистики предприятия.

Втора глава несёт в себе логическое проектирование разрабатываемого веб-приложения. В главе производится проектирование архитектуры веб-приложения, модели данных, описываются требования относительно целостности данных и к серверному обеспечению для развертывания веб-приложения на серверах предприятия.

Третья и заключительная глава описывает выбранные технологии, используемые при разработке. Демонстрируется работа приложения, его функции, передаваемые и получаемые данные.

Глава 1 Технико-экономическая характеристика предприятия

1.1 Описание деятельности компании ИП «Стеблев Д.В»

ИП «Стеблев Д.В» - «Индивидуальный предприниматель Стеблев Дмитрий Викторович». Предприятие осуществляет свою деятельность на территории Самарской области и имеет офис по адресу – г. Тольятти, ул. Офицерская, дом 14, офис 9.

Основными видами деятельности предприятия являются:

- оптовая торговля автомобильными деталями, узлами и принадлежностями;
- розничная торговля автомобильными деталями, узлами и принадлежностями;
- комплектация автомобильных деталей, узлов и принадлежностей.

Предприятие специализируется на комплектации и продаже автомобильных деталей, узлов и автомобильных принадлежностей. Комплектующие хранятся на территории склада запчастей предприятия, после чего они отправляются в пункт комплектации, где проходят процесс комплектации для дальнейшей продажи, затем готовая продукция попадает на товарный склад.

Предприятие может предложить своим покупателям более 30 позиций различных товаров для автомобилей, так же имеется возможность покупки путем оформления заказа, в случае отсутствия того или иного товара на складе. Помимо этого, предприятие обладает собственным автопарком, ввиду чего, покупатель может приобрести большие объемы товара с доставкой в любое удобное ему место.

1.2 Организационная структура предприятия ИП «Стеблев Д.В» и ее характеристика

Организационная структура предприятия ИП «Стеблев Д.В» представлена на рисунке 1.

Структурно предприятие разделено на следующие функциональные отделы:

- отдел информационных технологий,
- отдел кадров,
- юридический отдел,
- отдел логистики,
- отдел финансов.

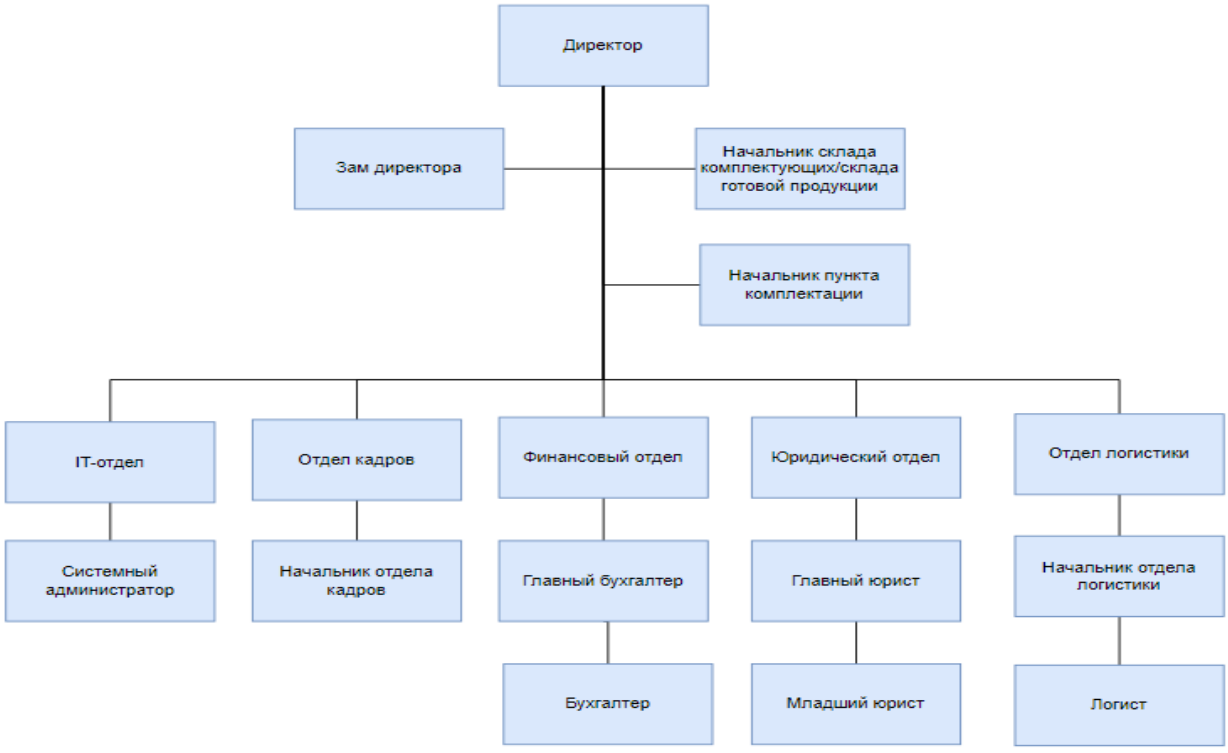

Рисунок 1 - Организационная структура предприятия ИП «Стеблев Д.В»

Начальник склада комплектующих/склада готовой продукции – отвечает за деятельность складов, управляет персоналом склада.

Начальник пункта комплектации – занимается управлением персоналом пункта комплектации, отслеживает процессы комплектации продукции.

Отдел информационных технологий - этот отдел отвечает за установку корпоративного оборудования и ведение корпоративных баз данных.

Отдел кадров - отдел занимается подбором персонала, планированием рабочего графика, а также за соблюдением трудового кодекса РФ.

Финансовый отдел - это отдел, который отвечает за финансовую часть компании.

Юридический отдел - занимается правовой защитой интересов предприятия, соблюдением законодательства РФ в процессе деятельности предприятия, а также консультирует руководителей подразделений по юридическим вопросам.

Отдел логистики - отвечает за оптимизацию и управление потоками товаров, поддерживает связь между поставщиками, складами, пунктом комплектации и точками сбыта. Так же отдел должен организовывать эффективные маршруты доставки, хранения и распределения продукции.

1.3 Краткая характеристика объекта автоматизации

В соответствии с темой ВКР, объектом автоматизации выступает отдел логистики предприятия.

На текущий момент отдел основными видами деятельности отдела логистики являются: обеспечение поставок комплектующих на склад запчастей и в пункт комплектации предприятия, поиск поставщиков и организация маршрутов поставок.

Обеспечение поставок на склад запчастей и в пункт комплектации предприятия происходит путем формирования заявок. На данный момент, все процессы, связанные с поставкой комплектующих, а также оформление и

обработка заявок на поставку, возможна лишь при помощи мобильной связь: заказчик (начальник склада комплектующих/склада готовой продукции или же начальник пункта комплектации) делает звонок в отдел логистики предприятия и тем самым начинает процесс оформления заявки на поставку. Помимо этого, по мобильной связи так же осуществляется связь с поставщиками, пунктами сбыта и перевозчиками.

1.4 Концептуальное проектирование предметной области

Чтобы проанализировать бизнес-процессы предприятия, необходимо построить диаграммы, соответствующих бизнес-процессов.

Нотациями, позволяющими построить модели бизнес-процессов являются:

- business process model and notation (BPMN);
- event-driven process chain (EPC);
- integrated computer aided manufacturing definition (IDEF0).

«Нотация BPMN (Business Process Management Notation, нотация моделирования бизнес-процессов) — нотация класса WorkFlow, один из наиболее распространенных методов описания бизнес-процессов на сегодня. В нотации BPMN используется базовый набор интуитивно понятных элементов, которые позволяют определять сложные семантические конструкции.» [8] Данная нотация не подходит для моделирования бизнес-процессов предприятия ИП «Стеблев Д.В».

«EPC (event-driven process chain, событийная цепочка процессов) – графический стандарт моделирования бизнес-процессов, нотация класса WorkFlow. EPC-метод был разработан Августом-Вильгельмом Шеером в начале 1990-х годов при создании ARIS. Отличительной особенностью является обязательное чередование событий и функций. На диаграммах EPC можно увидеть последовательность решений, функции, события, документы и статусы и другие элементы бизнес-процесса.» [8] Данная нотация также не

является наилучшим выбором для моделирования бизнес-процессов предприятия ИП «Стеблев Д.В».

IDEF0 – «это методология графического моделирования процессов, используемая для реализации систем и разработки программного обеспечения. Эти методы используются в функциональном моделировании данных, симуляции, объектно-ориентированном анализе и приобретении знаний.» [19].

Проанализировав существующие нотация, было принято решение использовать нотацию IDEF0, при помощи которой будет строиться модель бизнес-процессов предприятия ИП «Стеблев Д.В».

В процессе моделирования необходимо реализовать следующие виды моделей:

- как есть – «это описание процессов в том состоянии, в котором они находятся в данный момент.» [4];
- как будет – «это представление целевого состояния.» [4].

В качестве инструмента для моделирования схем бизнес-процессов предприятия, была выбрана программа «Ramus Educational».

Схема бизнес-процессов «как есть» предприятия ИП «Стеблев Д.В» изображена на рисунке 2.

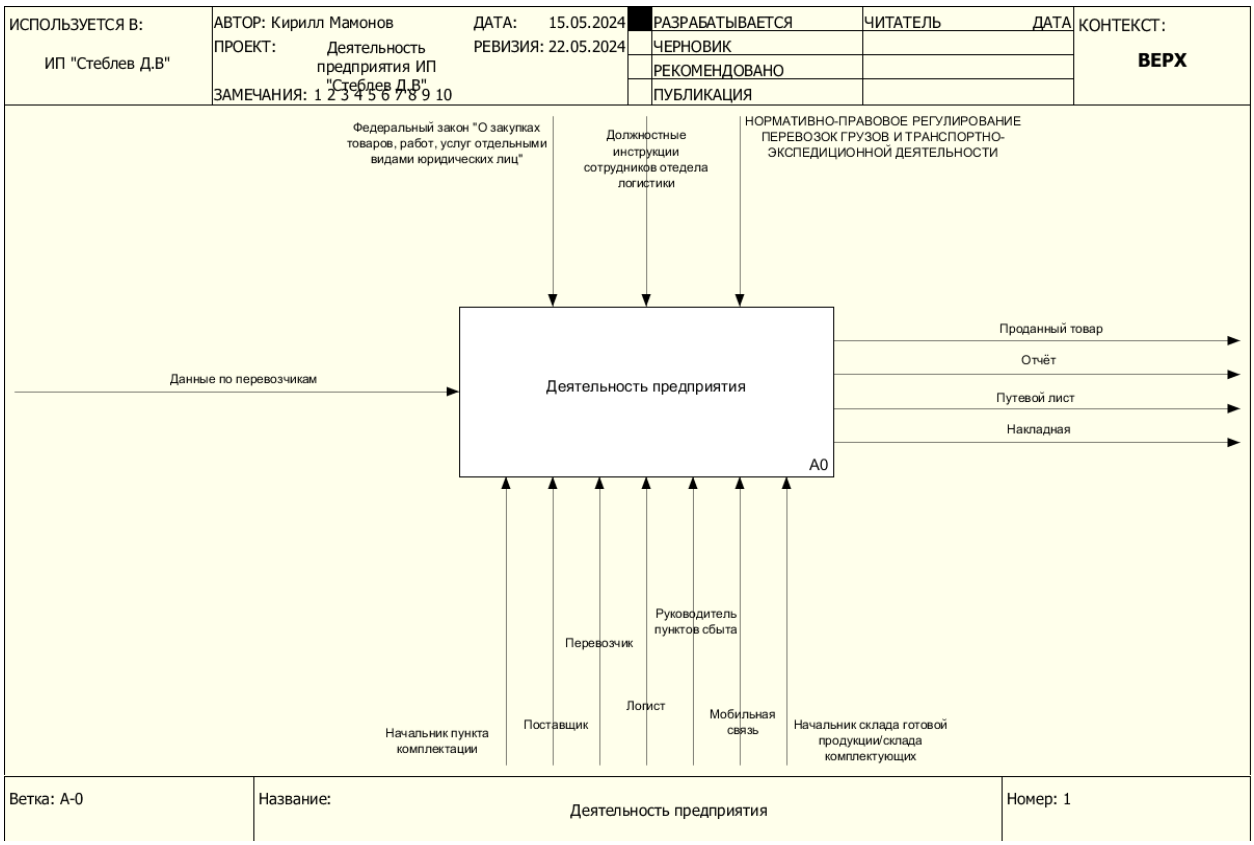

Рисунок 2 - Диаграмма верхнего уровня «как есть» процессов деятельности предприятия

Входным потоком являются данные о перевозчиках.

Потоками управления являются:

– федеральный закон "О закупках товаров, работ, услуг отдельными видами юридических лиц";
– должностные инструкции сотрудников отдела логистики;
– нормативно-правовое регулирование перевозок грузов и транспортно-экспедиционной деятельности.

Механизмами являются:

– начальник склада готовой продукции/склада комплектующих – занимается организацией складской работы, контролирует логистику товаров на складе, а также может отправлять запросы на пополнение склада различными комплектующими;

- начальник пунктом комплектации – занимается организацией работы пунктом комплектации предприятия, может отправлять запросы на поставку комплектующих со склада;
- поставщик – поставляет необходимые комплектующие;
- логист – сотрудник отдела логистики;
- руководитель пунктов сбыта – руководит пунктами сбыта продукции предприятия;
- перевозчик – водитель предприятия или сторонняя организация, занимается перевозкой груза из пункта из одной точки в другую;
- мобильная связь – для связи с сотрудниками предприятия, поставщиками, пунктами сбыта и перевозчиками.

Выходными потоками являются:

- проданный товар – товар, переданный на продажу пунам сбыта;
- отчёт – отчёт о проделанной роботе;
- путевой лист – документ, необходимый для транспортировки товара;
- накладная – документ, содержащих информацию о передаче товара или любых иных материальных ценностей.

На рисунке 3 представлена декомпозиция верхнего уровня А0 деятельности предприятия.

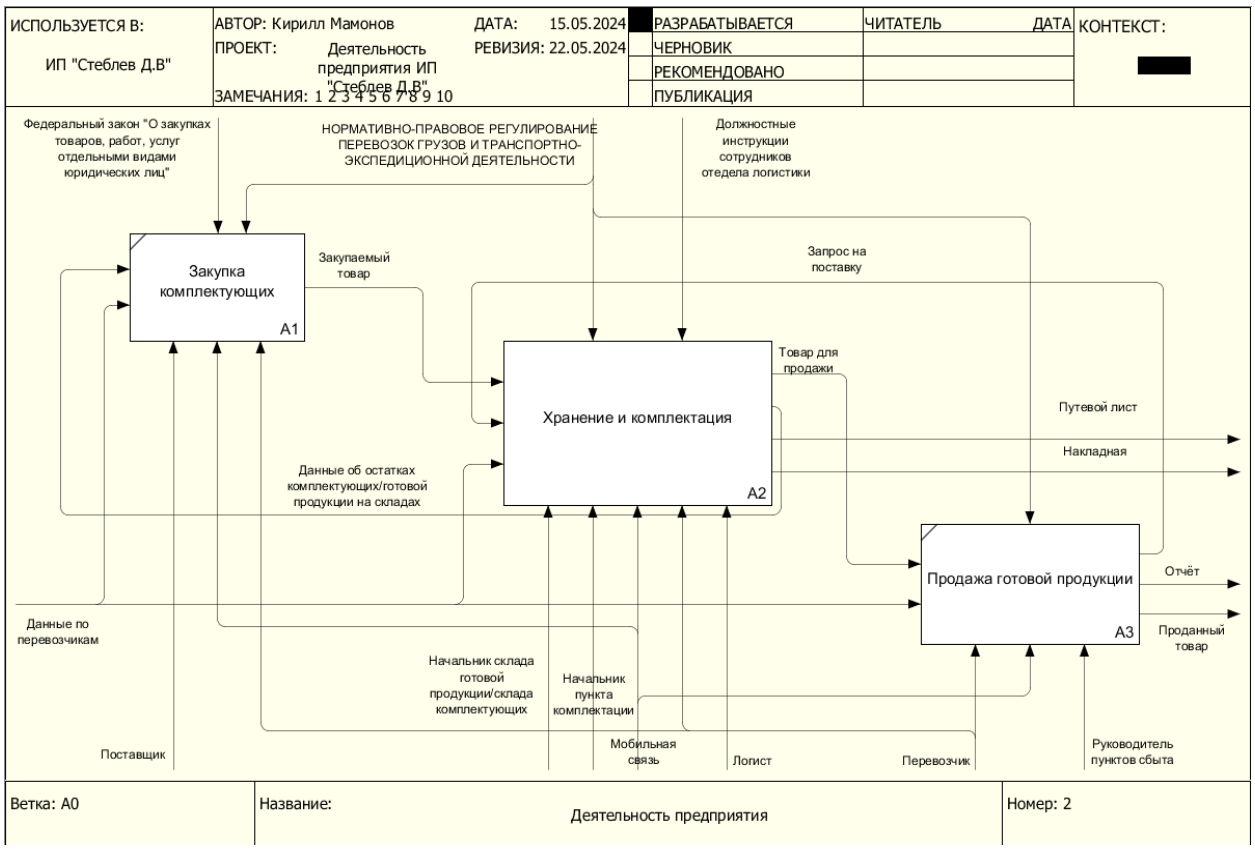

Рисунок 3 - Декомпозиция диаграммы A0 «как есть» процесса деятельности предприятия

Декомпозиция состоит их трёх бизнес-процессов:

– закупка комплектующих,

– хранение и комплектация,

– продажа готовой продукции.

Для построенной декомпозиции A0, на рисунке 4 изображена декомпозиция блока хранение и комплектация.

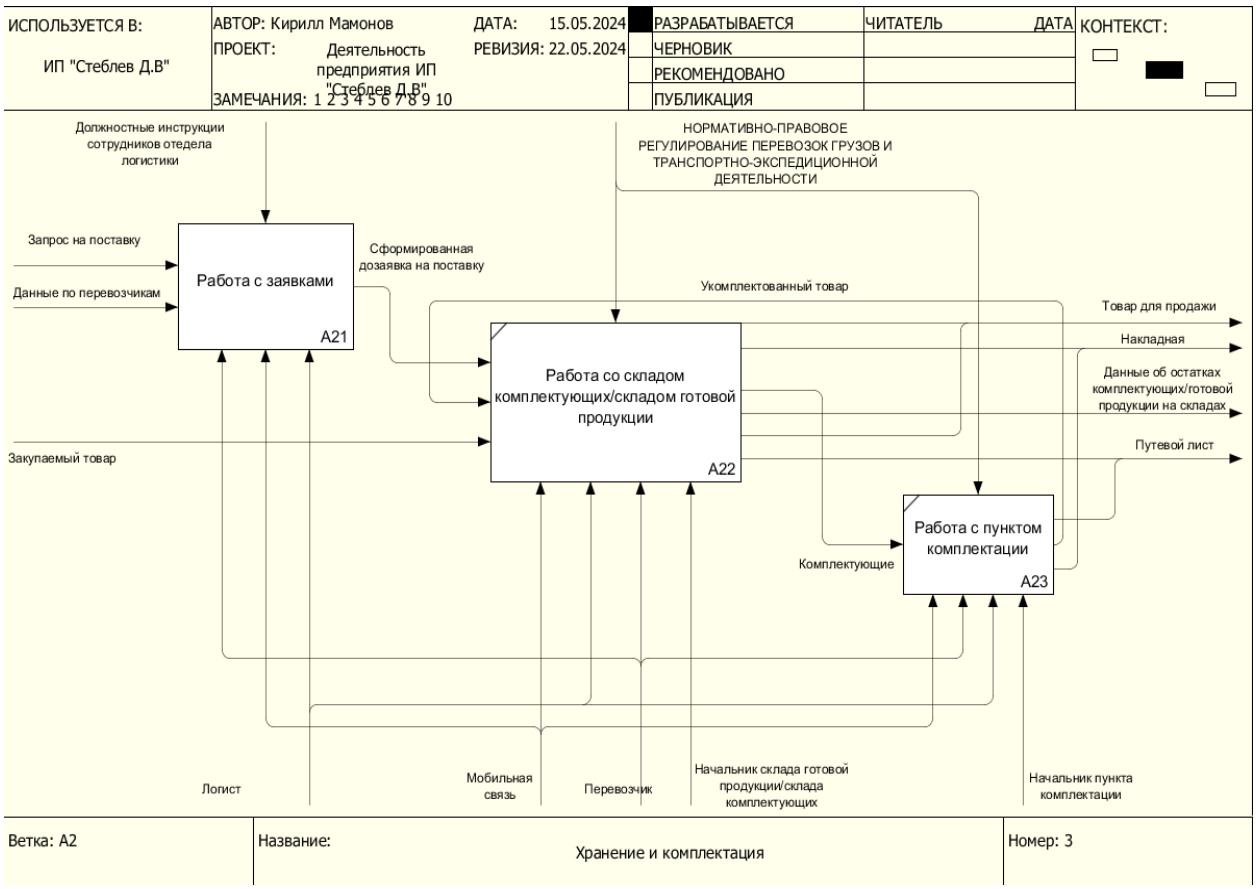

Рисунок 4 – Декомпозиция «как есть» бизнес-процесса хранение и комплектация

Декомпозиция бизнес-процесса «хранение и комплектация» состоит из трех блоков:

– работа с заявками;

– работа со складом комплектующих/складом готовой продукции;

– работа с пунктом комплектации.

Функция "работа с заявками" является одной из важнейших задач отдела логистики предприятия, поскольку от скорости и эффективности выполнения этой задачи зависят производственные возможности предприятия. В связи с этим, для блока «работа с заявками» также построена диаграмма декомпозиции. На рисунке 5 изображена декомпозиция блока «работа с заявками».

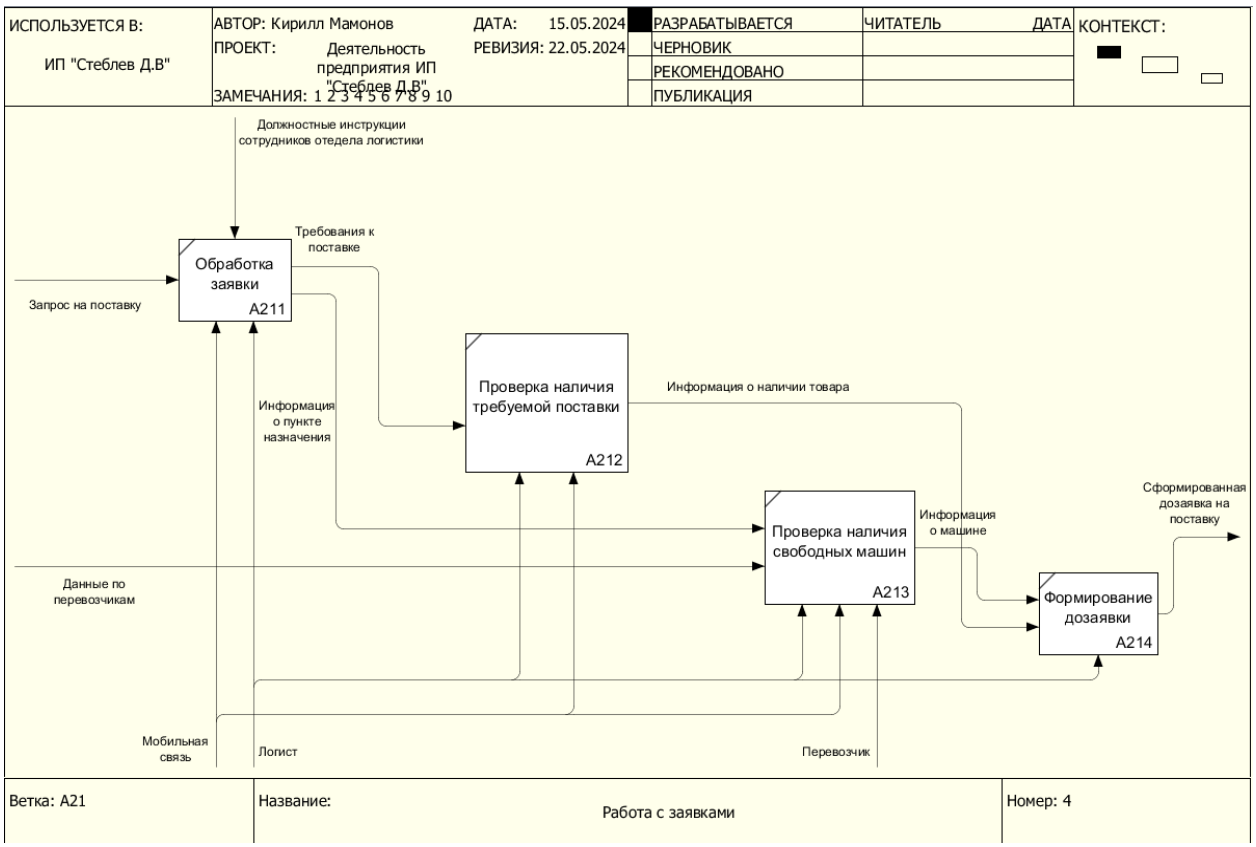

Рисунок 5 - Декомпозиция «как есть» бизнес-процесса работа с заявками

Составными бизнес-процесса «работа с заявками» являются подпроцессы:

- обработка заявки,
- проверка наличия требуемой поставки;
- проверка наличия свободных машин;
- формирование дозаявки.

Анализ модели «как есть» показал, что существующие бизнес процессы имеют ряд недостатков. Один из них является мобильная связь, ввиду того, что все процессы связанные, с деятельностью предприятия и отдела логистики в частности, происходят по мобильной связи, скорость работы отдела логистики и всего предприятия снижается, что плохо сказывается на продажах.

Так же недостатком является необходимость хранения всей информации в таблицах в программе MS Excel. Данный недостаток можно ликвидировать, используя, в качестве хранилища данных, стороннюю базу данных.

Ещё одним недостатком в существующей модели бизнес-процессов являются высокие затраты времени на оформление сотрудником отдела логистики нескольких заявок. Если сотрудники смогут за короткий промежуток времени оформить разные заявки на поставку, это поможет улучшить быстродействие отдела логистики и улучшить производственные возможности предприятия.

Модель «как есть» служит основой для построения модели бизнес процессов предприятия «как будет», чтобы способствовать совершенствованию бизнес-процессов предприятия в дальнейшем.

На рисунке 6 приведена контекстная диаграмма «как будет» деятельности предприятия.

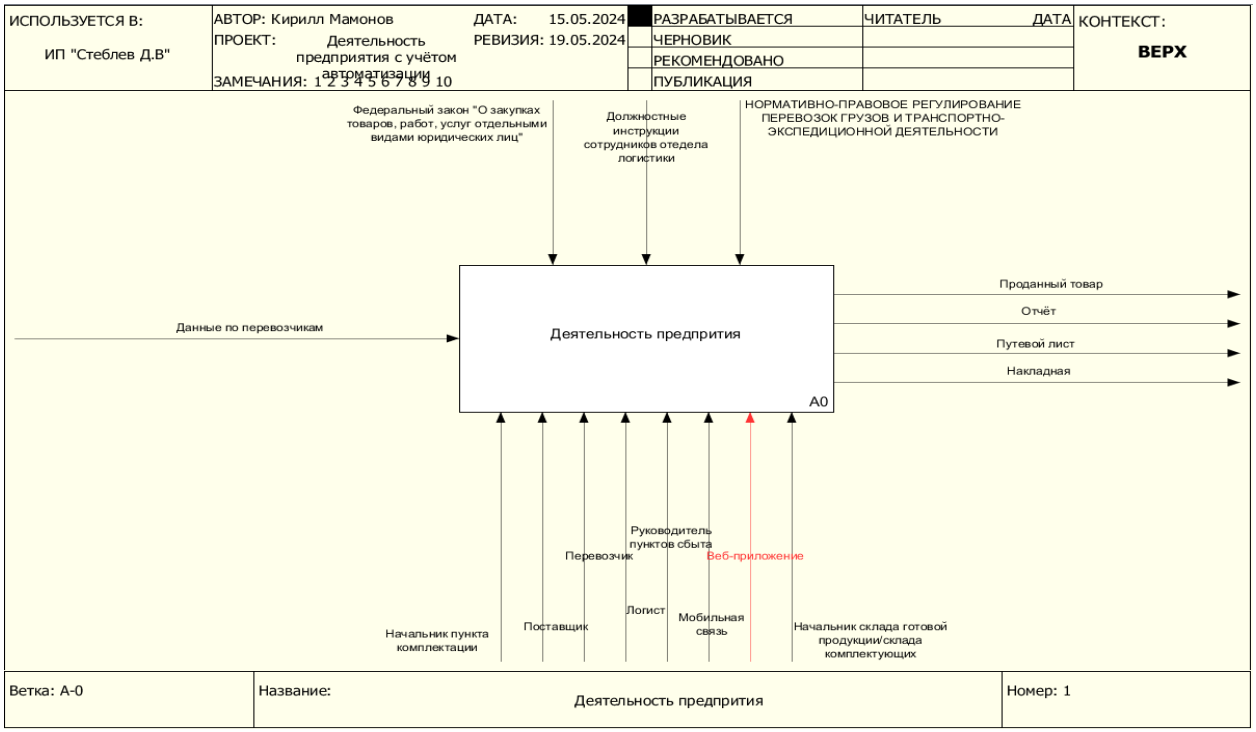

Рисунок 6 - Контекстная диаграмма A-0 «как будет» «деятельность отдела логистики»

Автоматизация деятельности предприятия заключается во внедрении в систему предприятия веб-приложения для формирования и обработки заявок. На рисунке 7 изображена декомпозиция диаграммы А0 «как будет» для деятельности предприятия.

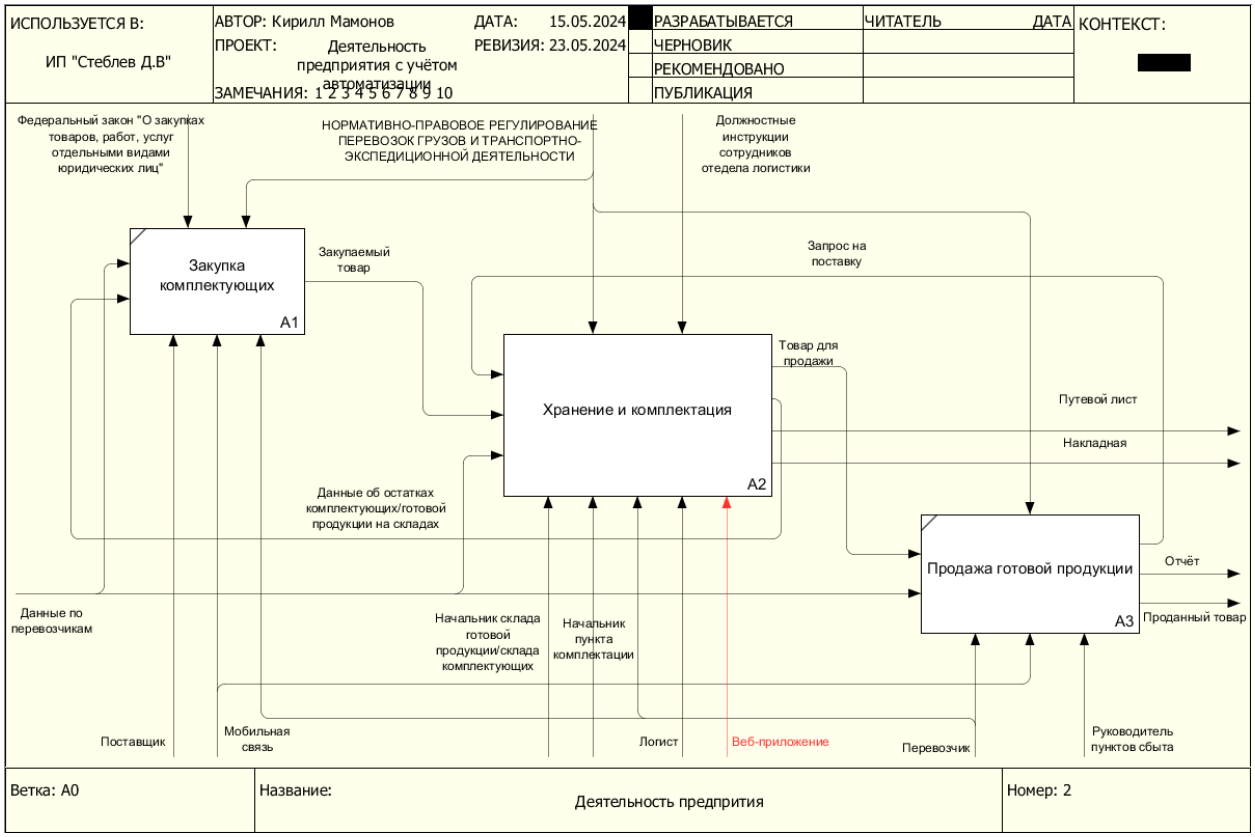

Рисунок 7 – Декомпозиция «как будет» контекстной диаграммы А-0

Декомпозиция «как будет» контекстной диаграммы А0 включает три процесса:

– закупка комплектующих,

– хранение и комплектация,

– продажа готовой продукции.

Для дальнейшего рассмотрения необходимо рассмотреть процесс, который будет автоматизирован в дальнейшем. На рисунке 8 представлена декомпозиция процесса «хранение и комплектация».

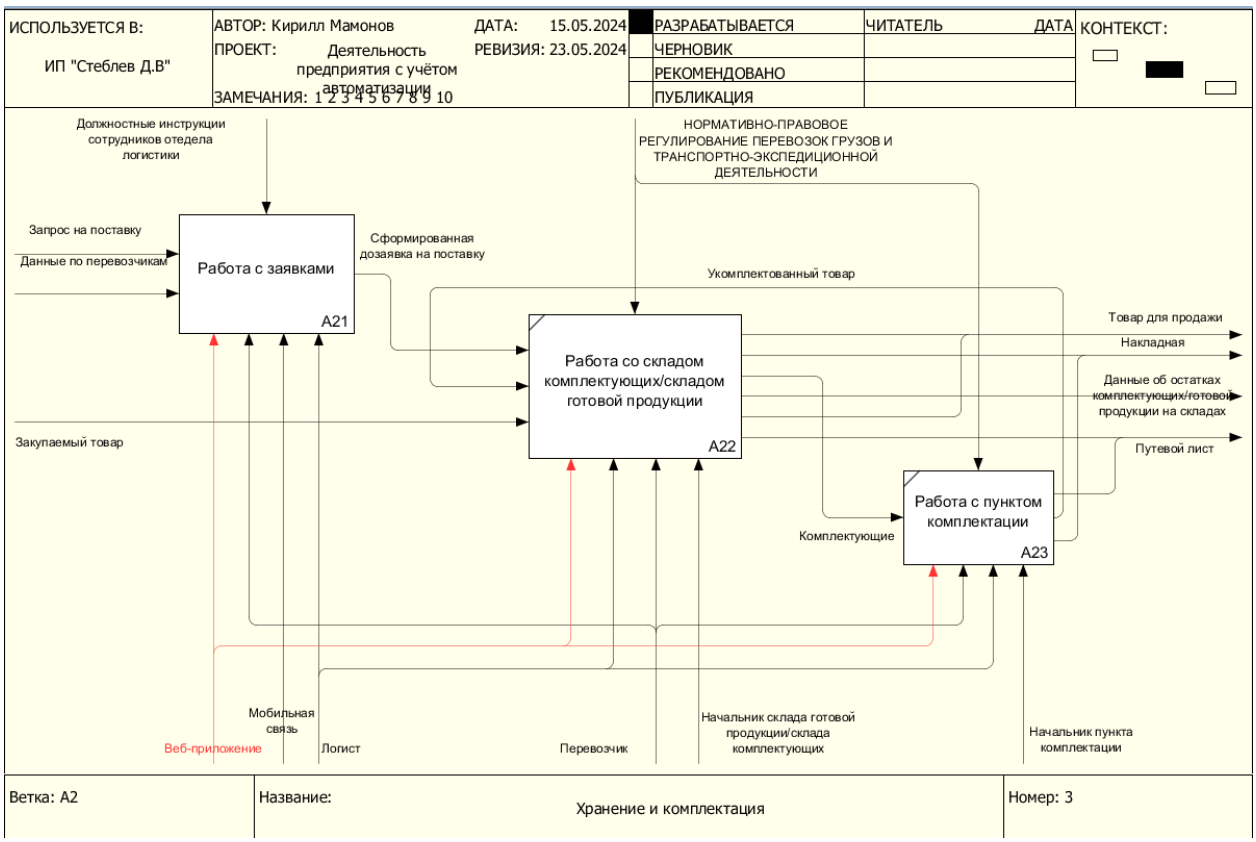

Рисунок 8 – Декомпозиция «как будет» процесса хранение и комплектация

Как видно из рисунка 8, вся работа, связанная с хранением и комплектацией будет автоматизирована посредством внедрения веб-приложения.

Чтобы более точно сформулировать необходимость внедрения веб-приложения с систему предприятия, необходимо рассмотреть декомпозицию бизнес-процесса работа с заявками для отдела логистики предприятия. На рисунке 9 изображена декомпозиция диаграммы «как будет»» бизнес-процесса «работа с заявками».

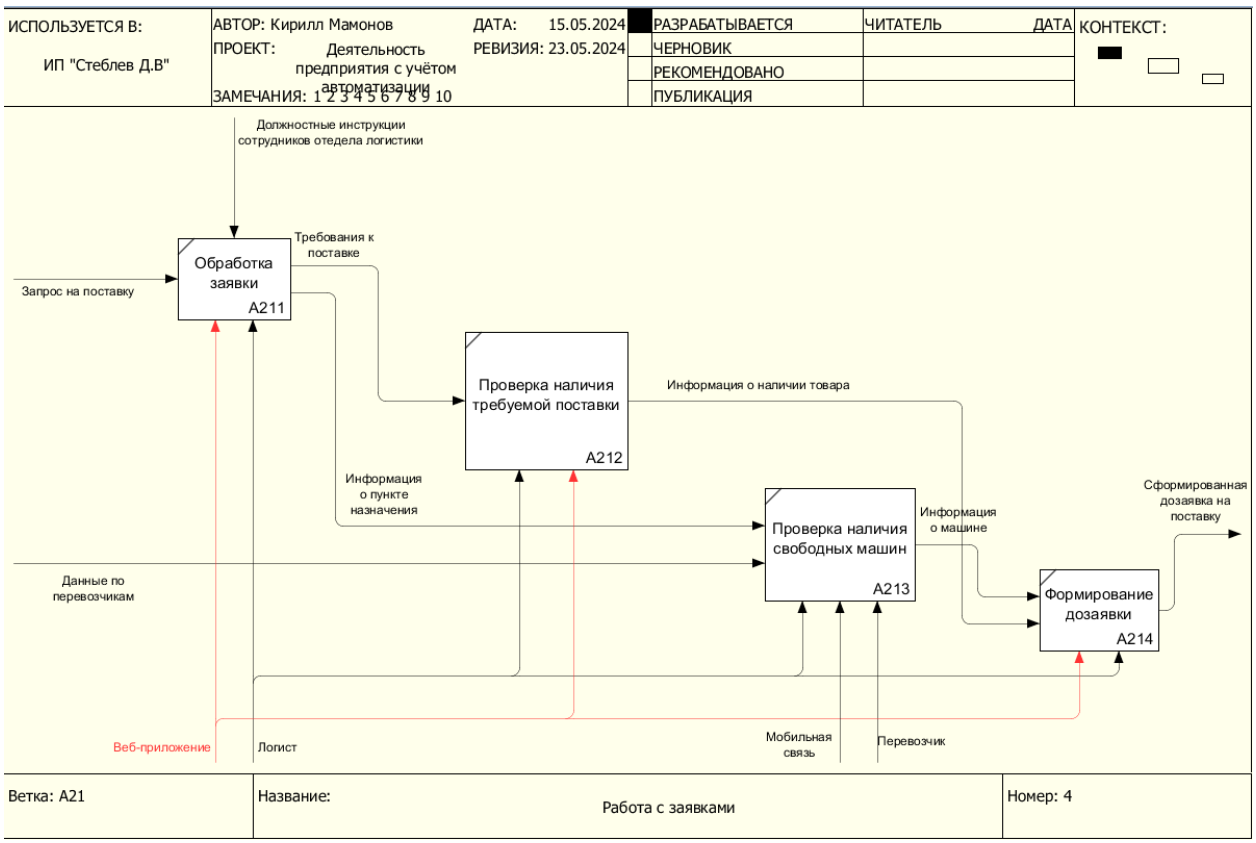

Рисунок 9 - Декомпозиция процесса «как будет» бизнес-процесса работа с заявками

На рисунке 9 можно увидеть, что весь процессы работы с заявками, будет автоматизирован благодаря внедрению веб-приложения для работы с заявками.

Разрабатываемое приложение позволит сократить время, затрачиваемое отделом логистики на формирование и обработку заявок на поставку, позволит быстро проверить наличие товара или комплектующих на складах предприятия, и позволит быстро формировать дозаявки на необходимую поставку.

Построенная диаграмма бизнес-процессов деятельности предприятия «как будет», является обоснованием для технического задания на разработку и администрирования веб-приложения для отдела логистики предприятия ИП «Стеблев Д.В».

1.5 Анализ существующих разработок для автоматизации комплекса задач

На данный момент существует множество уже разработанных систем, позволяющих автоматизировать деятельность различных предприятий, вне зависимости от их вида деятельности.

Одной из самых распространённых является система «1С: Предприятие». Система создавалась для повышения эффективности работы компании. Функциональность распределенной системы обусловлена ее масштабируемостью, так как она подходит для различных сфер продаж.

Немало известным является «SAP. ERP система» эта система помогает внедрить планирование ресурсов предприятия, которое позволяет реализовать основные процессы в единой системе для финансов, производства, управления персоналом, логистической цепочки, сервисных служб, закупок и других подразделений.

Так же, к не менее востребованным системам относится SAP Extended Warehouse Management (SAP EWM). Система помогает предприятиям эффективно управлять складскими операциями и контролировать их. Она предоставляет инструменты и функции для оптимизации управления запасами, упрощения складских процессов и повышения прозрачности цепочки поставок.

Преимущества таких систем заключаются в масштабируемости и обширном функционале, но их сложно реализовать и могут потребоваться значительные затраты на реализацию и поддержку.

Чтобы понять, подходят ли данные программные решения для внедрения в систему предприятия, необходимо их проанализировать. В процессе анализа будут сформулированы достоинства и недостатки каждой из представленных систем. В таблице 1 отображён анализ существующих программных решений.

Таблица 1 – Анализ существующих программных решений

Критерий сравнения	Программное решение		
	1С:Предприятие	SAP.ERP система	SAP EWM
отслеживание остатков на складах	+	+	+
формирование заявок на поставку	-	+	-
отдельная база данных	+	+	-
редактирование заявок	-	-	+
администрирование	-	+	+
поиск заявок по фильтрам	-	-	+
техническая поддержка	+	-	-
мобильное приложение	-	+	-
стоимость	от 25 000 за один компьютер	от 500 000 рублей	от 1 5000 000 рублей

Проанализировав существующие программные решения, был сделан вывод, что ни одно из них не может полностью реализовать все потребности предприятия. Кроме того, основным недостатком подобных систем является их стоимость, которая может варьироваться от реализации к реализации.

Результатом анализа является основание для начала разработки собственного веб-приложения REST службы логистики, в котором будут реализованы все необходимые для предприятия функции.

1.6 Постановка задачи на разработку и администрирование веб - приложения REST службы логистики

В результате моделирование бизнес-процессов отдела логистики предприятия, а также сравнении аналогов, среди уже существующих систем, выявлена необходимость в разработке веб-приложения REST для формирования и обработки заявок отдела логистики. Приобретение одного из рассматриваемых в пункте 1.5, готовых решений является недопустимым, ввиду финансовых потерь в связи с приобретением лицензии на использование

веб-приложения и сложностью дальнейшей поддержки системы без обращения к компании поставщику.

Система работы с заявками должна быть помощником в организации деятельности предприятия, путем упрощения работы отдела логистики предприятия.

Веб-приложение должно служить для автоматизации процессов формирования и обработки заявок на поставку отдела логистики. Приложение должно ориентироваться на архитектурный стиль REST. «REST, или Representational State Transfer, — это архитектурный стиль, обеспечивающий стандарты между компьютерными системами в сети, упрощающий взаимодействие систем друг с другом. REST-совместимые системы, часто называемые системами RESTful, характеризуются тем, что они не сохраняют состояние и разделяют задачи клиента и сервера.» [20]

Шаблон архитектуры разрабатываемой информационной системы REST изображён на рисунке 8.

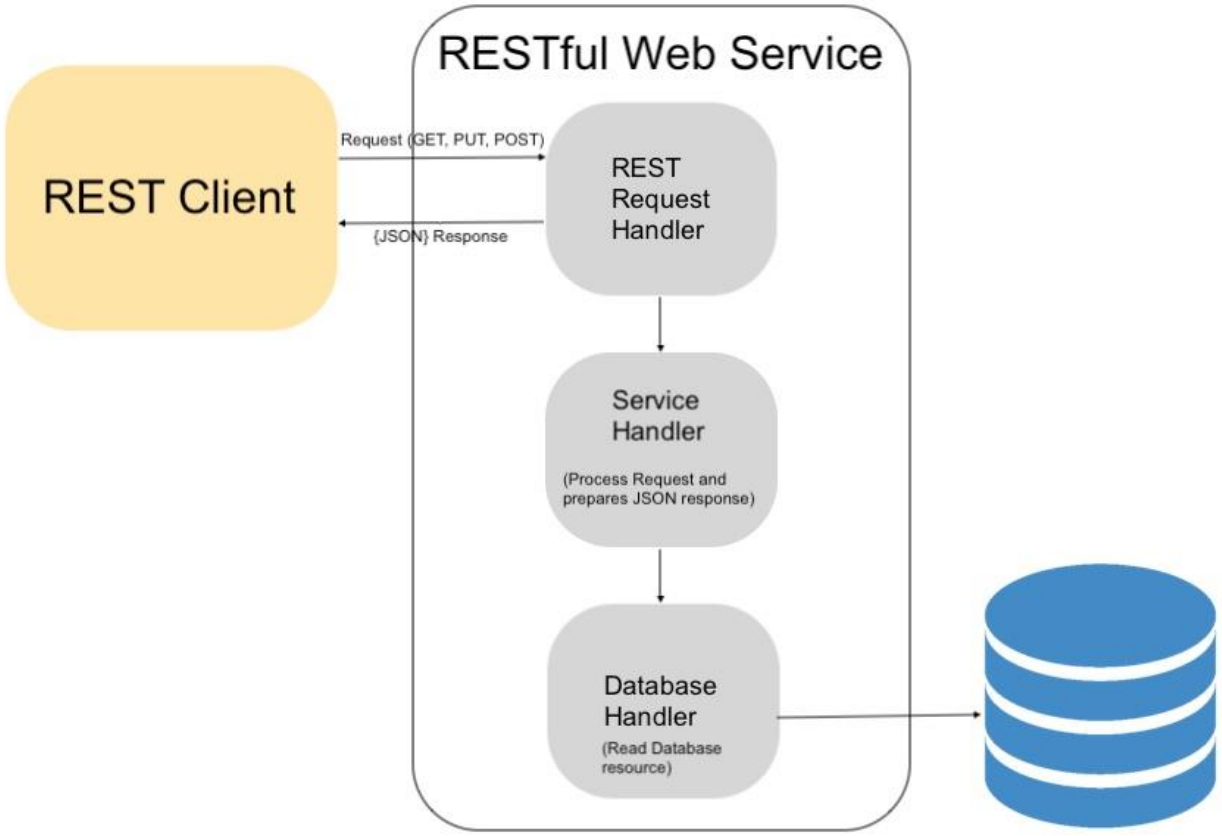

Рисунок 8 – шаблон архитектуры REST веб-приложения

«REST API не сильно отличаются от веб-сайтов. Оба они отвечают на HTTP-запросы. Но ключевое отличие заключается в том, что вместо того, чтобы отвечать на запросы с помощью HTML, как это делают веб-сайты, REST API обычно отвечают с помощью формата, ориентированного на данные, такого как JSON или XML. [18]

Ввиду следования архитектуре REST, архитектурно, приложение должно ориентироваться на микросервисную архитектуру, для более удобной масштабируемости в дальнейшем.

Приложение должно иметь реализацию следующих требований:

– администрирование веб-приложения (регистрация, авторизация);
– реализация CRUD операций внутри веб-приложения (получение, добавление, изменение, удаление данных) для работы с заявками;

– предоставление всей информации о собственных заявках авторизованным пользователям;
– отслеживание остатков комплектующих/деталей на складах предприятия.

Приложение должно отправлять и получать данные, в формате json, xml или другие. Помимо этого, должна быть реализована валидация данных, чтобы, любой запрос пользователя сопровождался откликом, в случае передачи невалидных данных, пользователь получал ошибку, с указанием проблемы.

Выводы по главе 1.

В процессе работы над главной был выполнен ряд задач: изучена деятельность предприятий ИП «Стеблев Д.В», его организационная структура, были разработаны концептуальные модели деятельности предприятия «как есть» и «как будет», произведен анализ существующих программных решений, на основании произведенного анализа, была сформулирована и обоснована задача на разработку и администрирование REST веб-приложения службы логистики.

Глава 2. Логическое проектирование веб-приложения REST службы логистики

2.1 Логическое моделирование веб -приложения

Для разрабатываемого приложения была построена диаграмма прецедентов (рисунок 9). На диаграмме задействуются следующие актеры:

– начальник склада комплектующих/склада готовой продукции (пользователь);

– логист.

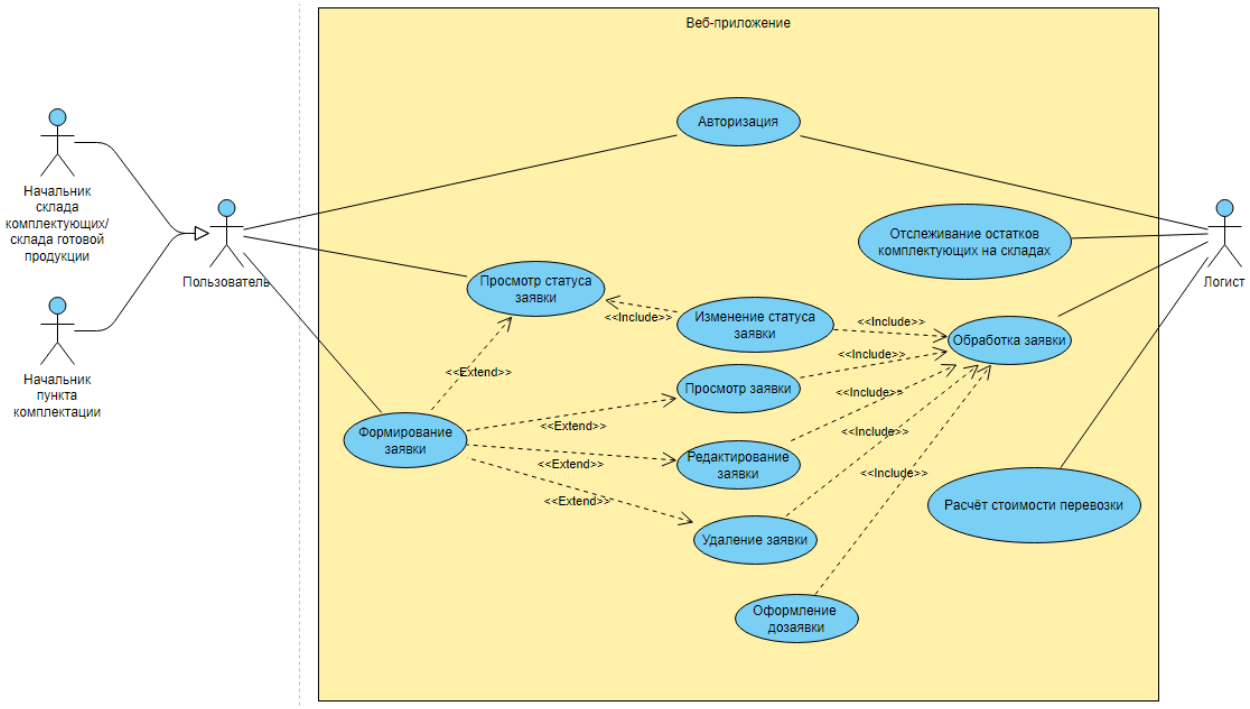

Рисунок 9 – Диаграмма прецедентов

Пользователю доступны следующие прецеденты:

– авторизация,

– формирование заявки (указываются атрибуты требуемой поставки);

– просмотр статуса выполнения заявки;

- базовые CRUD операции: просмотр заявки, редактирование заявки, удаление заявки).

Для сотрудника отдела логистики (логиста), доступны следующие прецеденты:

- авторизация,
- отслеживание остатков комплектующих на складах;
- расчёт стоимости перевозки;
- обработка заявки: просмотр пришедшей заявки, редактирование заявки, удаление заявки, изменение статуса заявки, оформление дозаявки;
- просмотр статуса заявки.

Далее рассмотрим диаграмму последовательности для процесса «формирование заявки». Диаграмма последовательности «используется для точного определения логики сценария выполнения прецедента.» [7]

Наиболее оптимальным выбором для создания диаграммы последовательности, является методология UML.

«Отличительная особенность UML — это возможность напрямую связать модели с языками программирования, благодаря чему нотацию можно рассматривать в качестве верхнеуровневого инструмента разработки.» [6]

«UML можно рассматривать как наследницу идей объектно-ориентированного анализа и проектирования. Для лучшего понимания, чем же является UML, разберем ключевые понятия объектно-ориентированного подхода к проектированию» [6]:

- «Объект - простейшая сущность, базовый строительный блок»; [6]
- «Класс - чертеж объекта, его условное описание». [6]

На рисунке 10 представлена UML диаграмма последовательности, разрабатываемого веб-приложения.

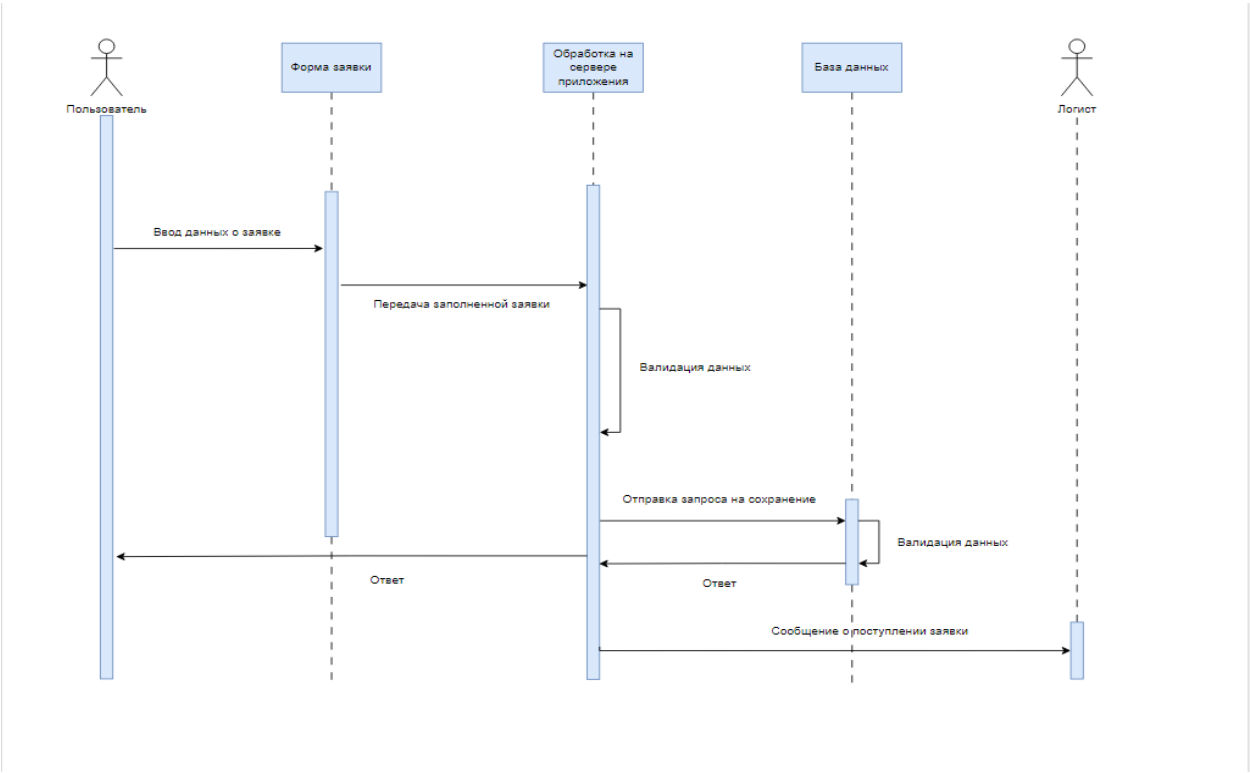

Рисунок 10 – UML диаграмма последовательности

На рисунке 10 можно увидеть, что пользователю доступен процесс оформления заявки. Каждый этап обработки заявки обладает своим откликом на разные события (проблемы валидации данных на сервере приложения, на сервере БД и другие). Вторым же пользователем является логист, который может посмотреть поступившие заявки.

2.2 Информационное обеспечение веб -приложения формирования и обработки заявок

Чтобы обеспечить целостность и уникальность данных и избежать в дальнейшем коллизии сущностей, для всех записей в БД и объектов в программе, будут задействованы идентификаторы. В таблице 2 представлена информация о идентификаторах, сохраняемых сущностей.

Таблица 2 – Идентификаторы сущностей

Наименование идентификатора сущности	Система кодирования
id заявки	порядковая (инкремент)
id груза	порядковая (инкремент)
id пользователя	псевдослучайное число
id роли	порядковая (инкремент)

Таким образов, идентификаторы позволят указать на уникальность сохраняемых в БД сущностей.

Входными данными при оформлении заявки являются: пункт назначения, груз: наименование ТМЦ, количество. На рисунке 11 продемонстрирован пример входных данных.

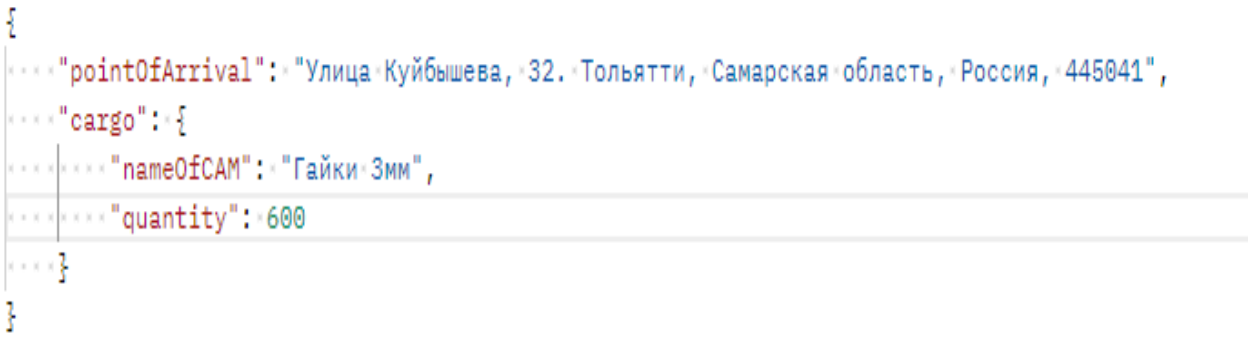

Рисунок 11 – Пример входных данных

Для наиболее эффективной обработки заявки, пользователь должен точно указать входные данные, опираясь на правила заполнения формы. Структура формы, заполняемой пользователем для оформления заявки приведена в таблице 3.

Таблица 3 – Структура входных данных

Описание поля	Наименование поля	Размеры поля (количество знаков)
пункт назначения	pointOfArrival	от 1 до 100
наименование ТМЦ	nameOfCAM	от 1 до 100
количество(штук)	quantity	от 1 до 100000

Как видно из таблицы 3, все поля имеют фиксированную длину и максимальное и минимальное значение и в случае несоблюдения поставленных требований, пользователь получит ошибку валидации с просьбой соблюдения поставленных норм.

В след за формированием и сохранением дозаявки, формируются данные, которые будут отправлены пользователю в качестве ответа.

Выходной информацией является оформленная заявка с дополнительными атрибутами заявки. Пример выходных данный продемонстрирован на рисунке 12.


```
{
    "orderCode": 10,
    "cargo": {
        "cargoId": 10,
        "cargoWeight": 4.0,
        "cargoLength": 1000.0,
        "cargoWidth": 215.0,
        "cargoHeight": 198.0,
        "nameOfCAM": "Гайки 3мм",
        "quantity": 600
    },
    "typeOfTruck": "FiveTons",
    "resultPrice": 49142.0,
    "distance": 700.0,
    "dateOfSubmission": "2024-04-20T17:51:34.007206",
    "responsible": {
        "username": "mamonovkirill",
        "nameOfThePosition": "Начальник Склада",
        "workplace": "улица Василисы Кожиной, 13, Москва, 121096"
    },
    "pointOfDeparture": "Улица Куйбышева, 32. Тольятти, Самарская область, Россия, 445041",
    "pointOfArrival": "улица Василисы Кожиной, 13, Москва, 121096",
    "departureDate": "2024-04-20T17:51:34.007206",
    "arrivalDate": "2024-04-21T17:51:34.007206",
    "orderStatus": "Delivered"
}
```


Рисунок 12 – Пример выходных данных

Выходные данные содержат полный набор характеристик заявки, основанных, на введённых пользователем данных. Структура выходной информации сущностей представлена в таблицах 4, 5, 6.

Таблица 4 – Структура выходной информации сущности «заявка»

Описание поля	Наименование поля
код заявки	orderCode
тип грузовика	typeOfTruck
итоговая стоимость	resultPrice
дистанция	distance
дата оформления заявки	dateOfSubmission
пункт отправления	pointOfDeparture
пункт назначения	pointOfArrival
дата отправки	departureDate
дата прибытия	arrivalDate
статус заявки	orderStatus

Таблица 5 - Структура выходной информации сущности «груз»

Описание поля	Наименование поля
id груза	cargoId
вес груза	cargoWeight
длина груза	cargoLength
ширина груза	cargoWidth
высота груза	cargoHeight
наименование ТМЦ	nameOfCAM
количество	quantity

Таблица 6 - Структура выходной информации сущности «пользователь»

Описание поля	Наименование поля
имя пользователя	username
должность	nameOfThePosition
место работы	workplace

Выходная информация содержит всю необходимую пользователю информацию по его заявке, включая дату прибытия груза, а также статус выполнения заявки.

2.3 Проектирование базы данных веб-приложения

2.3.1 Технологии проектирования базы данных веб-приложения службы логистики

В своей книге Дубейковский пишет: «проектирование базы данных включает в себя следующие этапы: построение концептуальной схемы по объектам предметной области, построение логической модели базы данных. Концептуальную схему целесообразно строить с использованием нотации Чена» [3].

В процессе проектирования логической модели данных использовался нотация Мартина.

Нотация содержит:

- сущность – «индивидуально идентифицируемый объект реальности, такой как человек, концепция или событие, о котором могут храниться данные.» [15];
- «атрибуты - это характеристики сущности, отношения «многие ко многим» или отношения «один к одному». [14];
- отношение – указывает связи между сущностями;

Выделяет три типа отношений между сущностями:

- один к одному - «один экземпляр сущности связан только с одним экземпляром другой сущности.»; [10]
- один ко многим – «один экземпляр сущности связан со множеством экземпляров другой сущности.»; [10]
- многие ко многим - «множество экземпляров одной сущности связаны со множеством экземпляров другой сущности.» [10]

Таким образом, при помощи нотации Мартина, можно изобразить на схеме, предметную область, её сущности, их атрибуты и связи между ними.

2.3.2 Разработка концептуальной модели данных веб-приложения

Концептуальная схема предназначена для определения основных сущностей, необходимых для решения задач. На рисунке 13 представлена концептуальная модель для базы данных проектируемой системы.

Для соответствия предметной области, в приложении были выделены следующие сущности:

– users (пользователи) – пользователь приложения,
– orders (заявки) – оформленные пользователями заявки,
– cargo (груз) – информация о характеристиках груза, дополняющая заявку;
– roles (роли) – роли, присеваемые пользователям при регистрации.

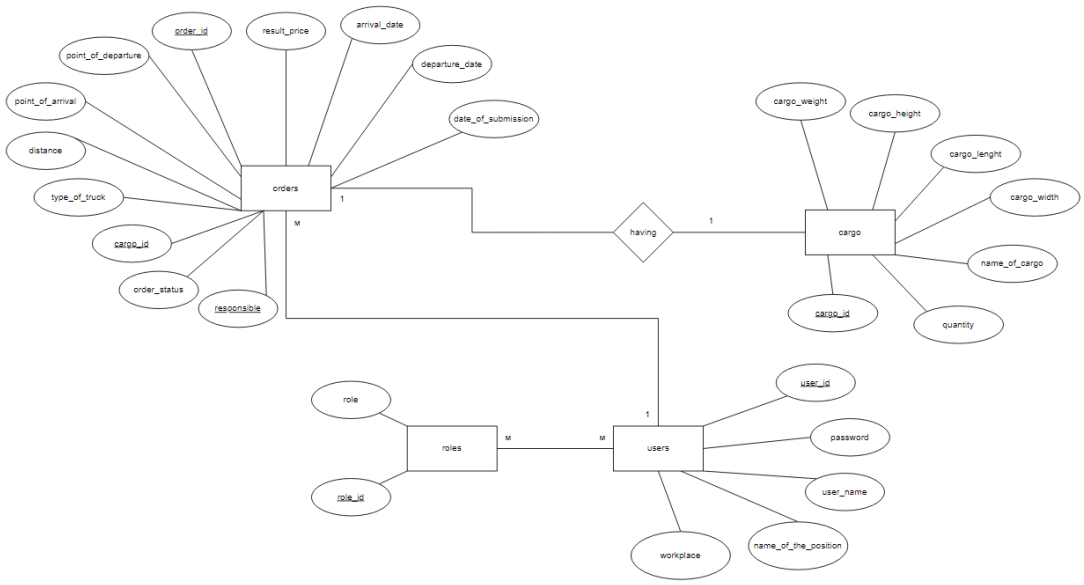

Рисунок 13 – Концептуальная модель

Концептуальная схема является основой для построения логической модели данных информационной системы отдела логистики предприятия.

2.3.3 Обоснование вида логической модели

Перед тем, как начать разрабатывать логическую модель данных, необходимо проанализировать виды баз данных. На данный момент существует два вида баз данных: реляционные и нереляционные. Нереляционные базы данных имею два подвида: сетевые и иерархические. Рассмотри особенности каждой из них:

«Реляционная модель данных была представлена доктором Эдгаром Ф. Коддом в 1970 году как способ упростить представление отношений данных. Реляционная модель представляет данные в виде отношений, которые по сути представляют собой таблицы со строками и столбцами. Каждая строка, также известная как кортеж, представляет одну запись данных, а каждый столбец соответствует атрибуту типа данных.» [5].

«Иерархическая модель данных — одна из первых моделей баз данных, разработанная в 1960-х годах. Он представляет данные с использованием древовидных структур, где каждый узел содержит один родительский и несколько дочерних узлов. Эта модель хорошо подходит для отношений «один ко многим» (1:N), когда родительский объект связан с несколькими дочерними объектами.» [5]

«Сетевая модель данных была разработана в конце 1960-х годов как развитие иерархической модели. Он расширяет иерархическую модель, позволяя узлу иметь несколько родительских и дочерних узлов. Эта гибкость позволяет сетевой модели данных представлять отношения «многие ко многим» (M:N), что делает ее подходящей для более сложных структур данных.» [5]

Проанализировав самые распространённые виды баз данных, было принято решение выбрать реляционную модели.

Так же, в процессе проектирования базы данных для приложения, необходимо выполнить нормализацию. «Нормализация базы данных (БД) - это метод проектирования реляционных БД, который помогает правильно структурировать таблицы данных. Процесс направлен на создание системы с

четким представлением информации и взаимосвязей, без избыточности и потери данных.» [12] Существует пять нормальных форм (НФ), но чаще всего применяют нормализацию к третьей нормальной форме (3НФ). Нормализация может быть проведена на любом этапе проектирования базы данных.

2.3.4 Выбор СУБД для базы данных веб-приложения

Любая проектируемая мая база данных будет взаимодействовать с СУБД. И ввиду того, что в ходе анализа, в качестве модели для базы данных приложения, была выбрана реляционная модель. В качестве СУБД для базы данных разрабатываемого веб-приложения, была выбрана PostgreSQL.

«PostgreSQL, или просто postgres, - объектно-реляционная система управления базами данных с открытым исходным кодом. На первом месте в ней стоит расширяемость, техническое совершенство и совместимость. Она конкурирует с основными базами данными: Oracle, MySQL, SQL Server и другими. Используется в самых разных секторах, включая государственные учреждения, открытые и коммерческие продукты. Это кросс-платформенная СУБД, работающая в большинстве современных операционных систем, в т. ч. Windows, macOS и различных дистрибутивах Linux. Совместима со стандартами SQL и обладает всеми свойствами ACID.» [2]

2.3.5 Разработка логической модели данных информационной системы сопровождения заказов

Разработанная логическая модель для базы данных приложения приведена на рисунке 14. Данная модель пока что не была приведена к 3НФ.

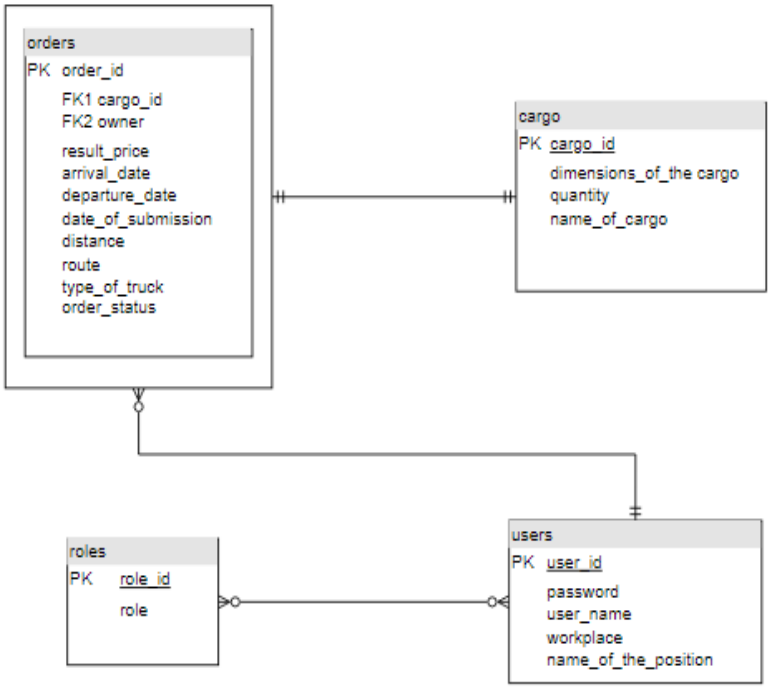

Рисунок 14 – Логическая модель (без нормализации)

Якобсон А. отмечает, что «Реляционные отношения соответствуют первой нормальной форме (1НФ), так как все атрибуты имеют атомарные значения. Реляционные отношения находятся во второй нормальной форме (2НФ), так как они находятся в 1НФ и все неключевые атрибуты неприводимо зависят от ключевых. Реляционные отношения находятся в третьей нормальной форме (3НФ), так как они находятся в 2НФ и отсутствуют транзитивные зависимости между неключевыми атрибутами» [13].

Для приведения модели базы данных к 3НФ, необходимо внести ряд изменений в модель:

– разделить атрибут “route” (маршрут) на “point_of_arrival” (место отправки) и “point_of_departure” (место прибытия);

– разделить атрибут “dimensions_of_the cargo” (габариты груза) на “cargo_weight” (вес груза), “cargo_height” (высота груза), “cargo_lenght” (длина груза), “cargo_width” (ширина груза).

Приведённая к 3НФ логическая модель базы данных изображена на рисунке 15.

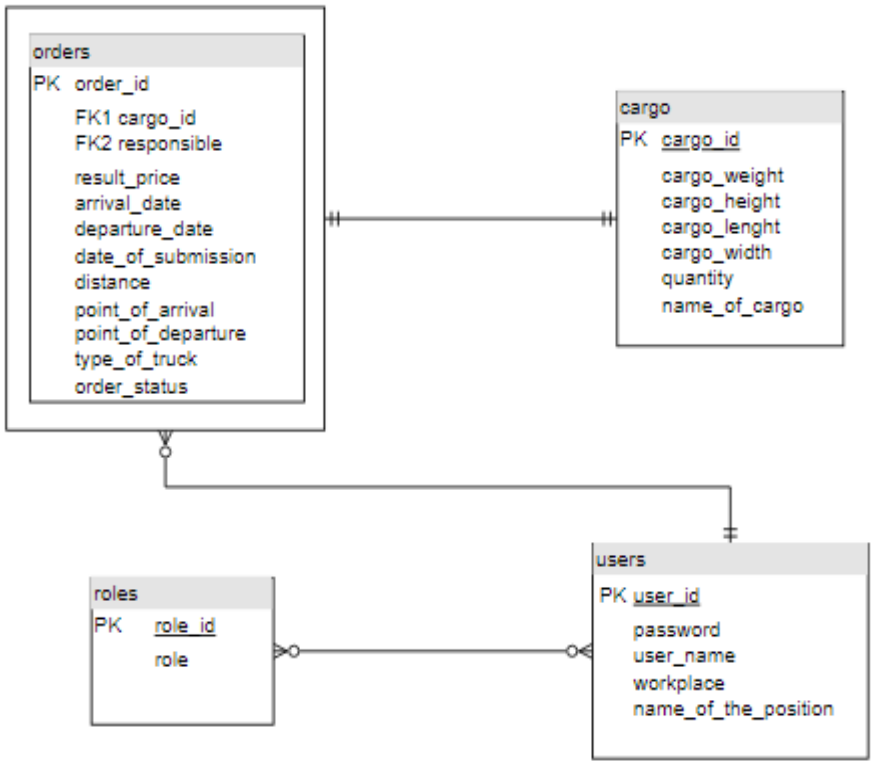

Рисунок 15 – Логическая модель, приведённая к 3НФ

Ориентируясь на разработанную логическую модель, была простроена физическая модель. Так как в качестве СУБД для базы данных выбрана PostreSQL, структура физической модели будет изменена, чтобы соответствовать требованиям. На рисунке 16 изображена физическая модели базы данных.

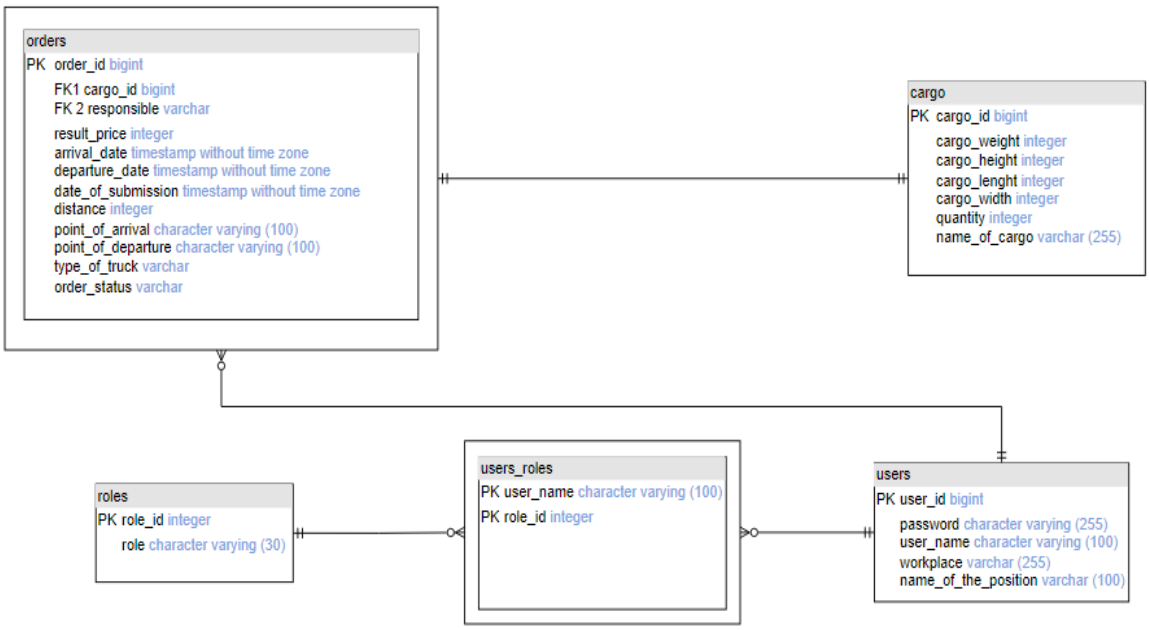

Рисунок 16 – Физическая модель

Как видно из схемы, для соответствия требованиям PostgreSQL, таблицы users и roles были разделены, и связаны через дочернюю таблицу user_roles с отношением один ко многом относительно таблиц users и roles.

2.4 Требования к аппаратному обеспечению веб-приложения службы логистики

Чтобы «развернуть» разрабатываемое веб-приложение на серверах предприятия, сервер должен обладать достаточными техническими характеристиками. «Диаграммы развертывания UML представляют физическое размещение артефактов системы на ее узлах.» [9] На рисунке 17 приведена диаграмма развёртывания приложения, на ее основе, можно будет сделать вывод, сколько и какие сервера должны иметься у предприятия, чтобы развернуть приложение.

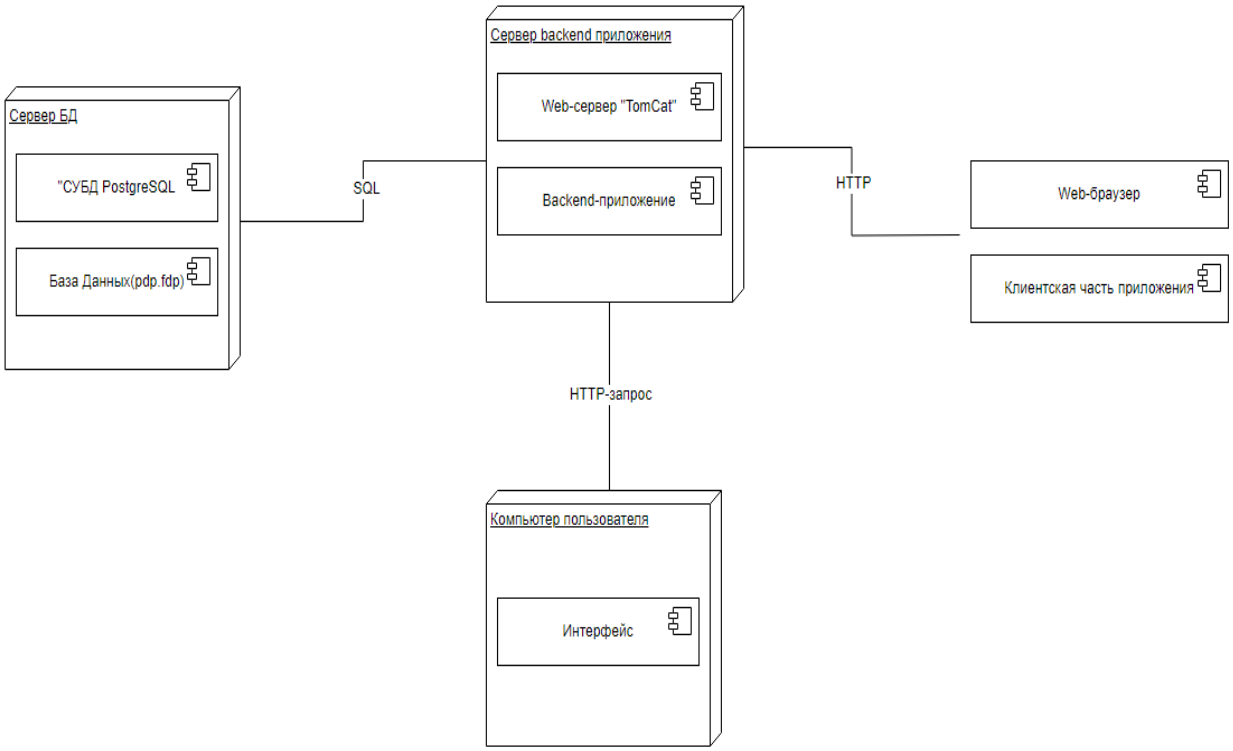

Рисунок 17 – Диаграмма развертывания

Как видно из рисунка 17, предприятие должно обладать по меньшей мере двумя серверами, а именно: сервер backend приложения, чтобы развернуть само приложение, и сервер базы данных, на котором будут храниться данные, необходимые для корректной работы веб-приложения.

Вывод по главе 2

В процессе работы над второй главой было произведено логическое моделирование разрабатываемого веб-приложения службы логистики, также были спроектированы модели данных, но основании которых будет спроектирована база данных приложения и сформулированы требования к аппаратному обеспечению.

Глава 3 Реализация разработанного веб-приложения

3.1 Описание разработанного приложения

Для разработки веб-приложения, был выбран объектом-ориентированный язык программирование Java, а также в качестве фрейморка, позволяющего вести веб разработку выбран Spring Framework.

Java – «это объектно-ориентированный язык, основанный на классах, который разработан для переносимости, что означает, что Java код может работать на различных аппаратных средствах и операционных системах. Java широко используется для разработки приложений корпоративного уровня, мобильных приложений, видеоигр и других типов программного обеспечения. Он известен своей философией "напиши один раз, запусти где угодно", поскольку код Java может быть скомпилирован для запуска на любой платформе, поддерживающей виртуальную машину Java (JVM).» [11] Этот язык программирования идеально подходит для написания веб-приложения.

Spring Framework – «фреймворк с открытым исходным кодом, написанный на Java. Его можно использовать для разработки на всех этих языках. Spring предоставляет огромный набор инструментов и библиотек, которые упрощают и ускоряют процесс разработки, позволяя сосредоточиться на бизнес-логике приложения.» [1]

Spring Framework позволяет разрабатывать серверную часть веб-приложения. Сам Spring также называют «фреймворком фреймврков» из-за объединения внутри единой системы, множество различных компонентов.

Для написания полноценного веб-приложения понадобятся следующие компоненты Spring Framework: Spring Core, Spring Data JPA, Spring Security, Spring Web, Spring Boot версии 3.0 и выше, Spring Validator, а также сторонние библиотеки: PostgreSQL Driver, Hibernate, Lombok, JWT.

По принципу “code first” сперва были разработаны сущности и их зависимости в программном коде, на рисунке 18 и в приложении А демонстрируется пример класса-сущности «заявка».

```
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "order_id")
    private Long orderId; //код заявки

    @OneToOne
    @JoinColumn(name = "cargo_id", referencedColumnName = "cargo_id")
    @Cascade(org.hibernate.annotations.CascadeType.ALL)
    private Cargo cargo;

    @Column(name = "type_of_truck")
    @Enumerated(EnumType.STRING)
    private TypeOfTruck typeOfTruck;//тип перевозки

    @Column(name = "result_price")
    private Double resultPrice;

    @Column(name = "distance")
    private Double distance;//дистанция (в км)

    @Column(name = "date_of_submission")
    private LocalDateTime dateOfSubmission;//дата оформления заявки(берется время и дата сервера)

    @ManyToOne
    @JoinColumn(name = "responsible", referencedColumnName = "user_name")
    private User responsible;//связь с юзером(ответственный по заявке)

    @Column(name = "point_of_departure")
    private String pointOfDeparture;//точка отправления

    @Column(name = "point_of_arrival")
    private String pointOfArrival;//точка прибытия

    @Column(name = "departure_date")
    private LocalDateTime departureDate;//дата отправки

    @Column(name = "arrival_date")
    private LocalDateTime arrivalDate;//дата прибытия (если нужна конкретная дата прибытия)
```

Рисунок 18 – Пример класса-сущности

Далее, на основе сущностей и их зависимостей, была спроектирована база данных, в качестве СУБД был выбран PostgreSQL. Даграмма базы данных информационной системы изображена на рисунке 19.

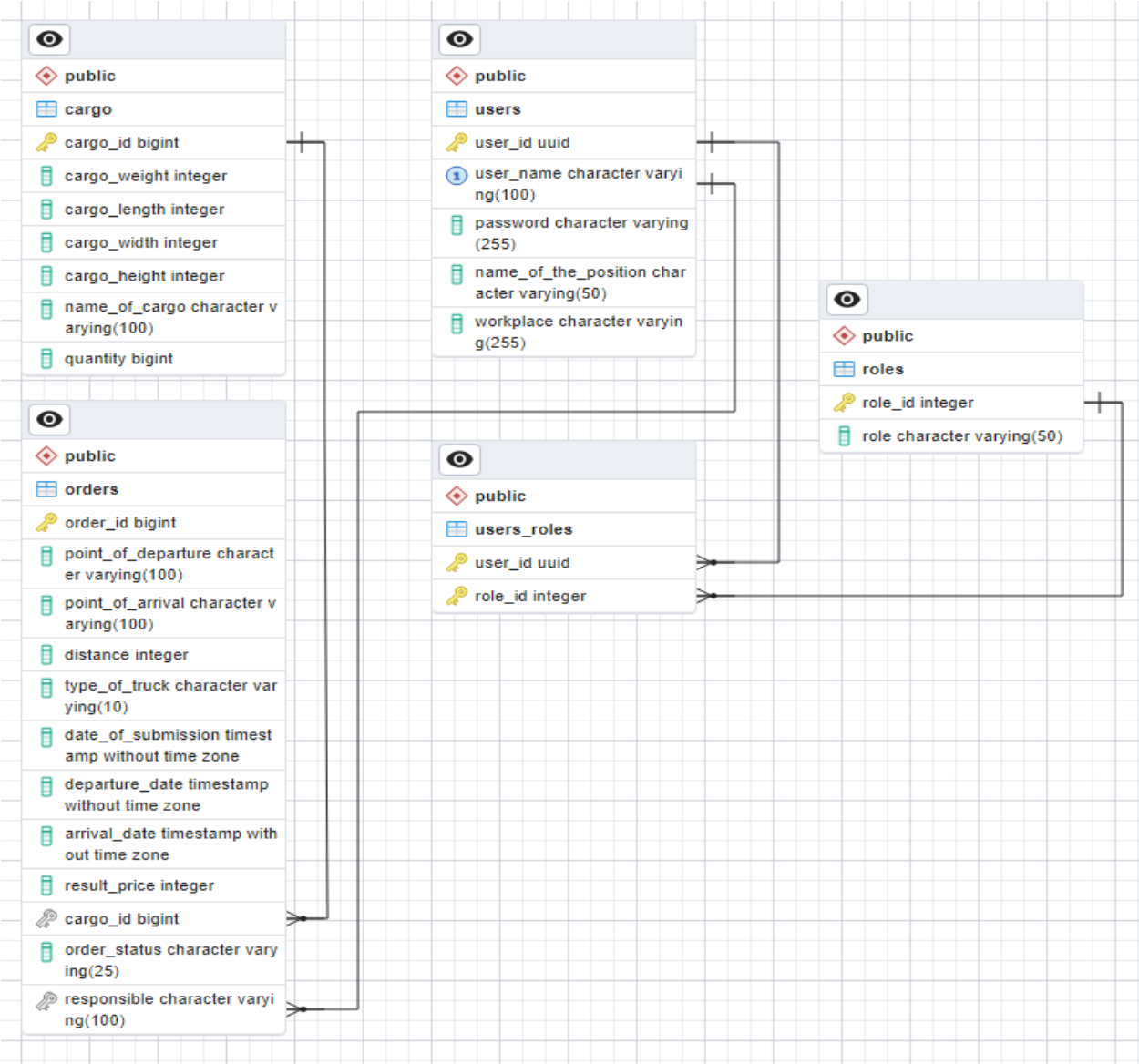

Рисунок 19 – Диаграмма базы данных

По спецификации JPA, «JPA означает Java Persistence API. Это платформа объектно-реляционного сопоставления (ORM), которая позволяет нам сопоставлять объекты Java с таблицами в реляционной базе данных. Другими словами, JPA предоставляет способ сохранения объектов Java в базе данных с помощью набора аннотаций, определяющих сопоставление между классами Java и таблицами базы данных.» [17], необходимо инкапсулировать слой доступа к хранилищу данных, для этого необходимо использовать

паттерн «Репозиторий», код класса, реализующего паттерн «Репозиторий», приведен в приложении Б.

Чтобы четко разделить зоны ответственности классов, все классы были разделены по разным пакетам и выделены в «слои». Диаграмма пакетов изображена на рисунке 20.

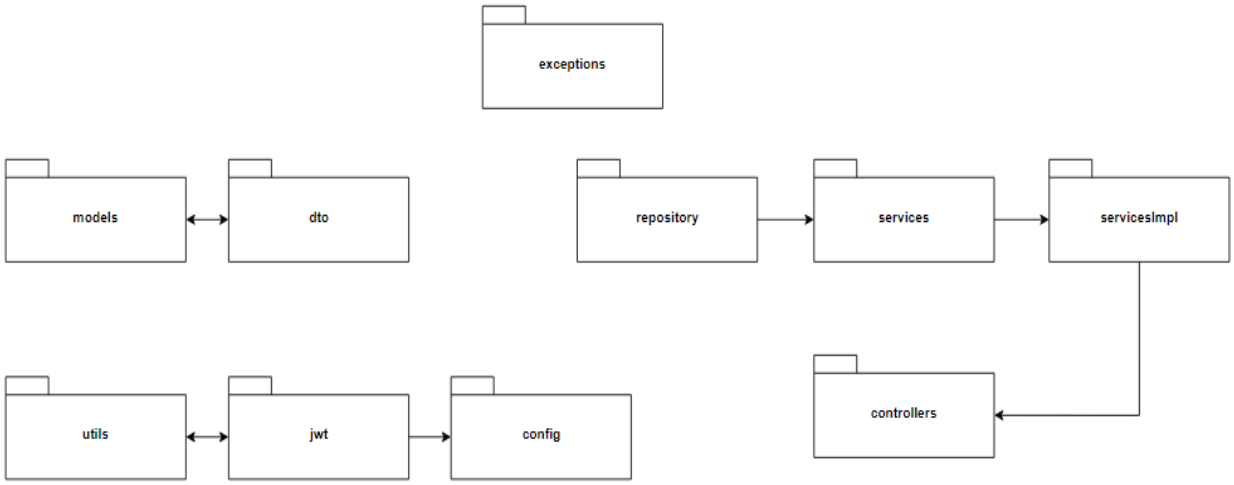

Рисунок 20 – Диаграмма пакетов приложения

Как видно из рисунка 20, разрабатываемое приложение включает в себя 10 пакетов. Ниже представлено описание каждого из них:

- services – содержит классы (интерфейсы) с сигнатурами методом для работы с данными;
- servicesImpl – содержит классы, реализующие классы пакета services, в класса реализуется логика работы с данными;
- models – пакет, содержащий классы-сущности;
- dto – слой представления данных;
- repository – пакет с классами для доступа к хранилищу данных, реализующими паттерн «репозиторий»;
- controllers – в пакете содержатся классы, отвечающие за обработку HTTP запросов, приходящих от «клиента»;

– config – пакет, содержащий всю конфигурацию приложения.
– exceptions – классы в этом пакете отвечают за глобальную обработку исключений приложения на всех «слоях»;
– jwt – содержит классы конфигурации авторизации/регистрации;

Программный код некоторых классов приведен в приложении.

3.2 Демонстрация работы приложения

Первое, что встречает клиент, это страница с доступными ему функциями. На рисунке 21 изображен перечень функций, доступный клиенту.

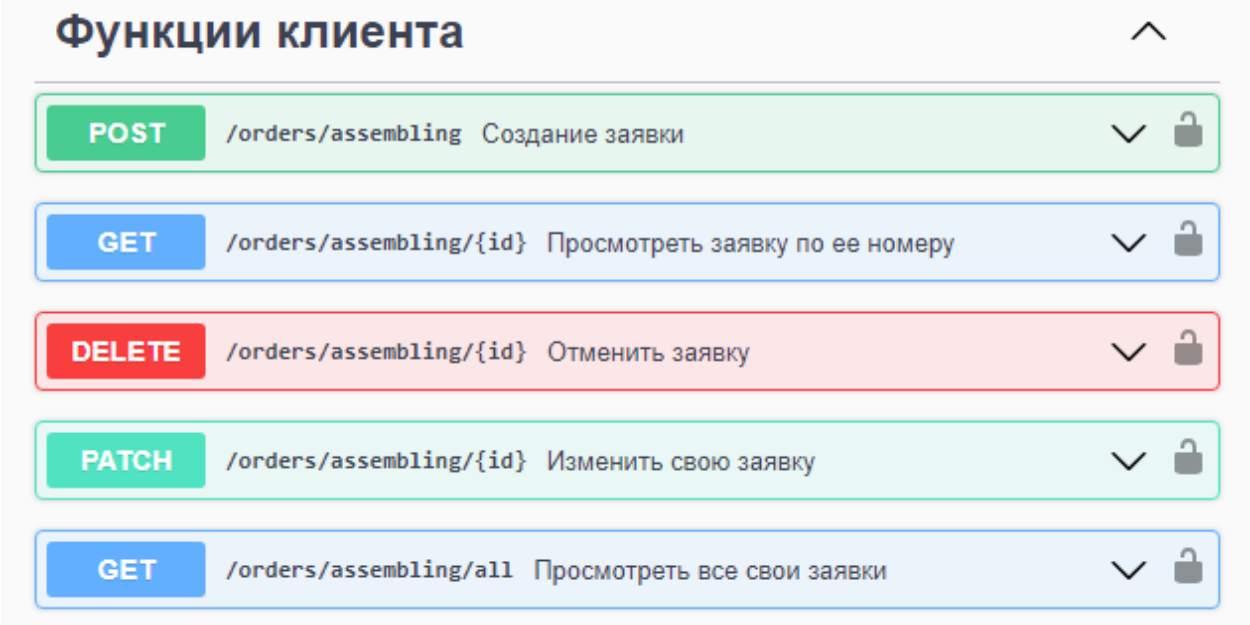

Рисунок 21 – Функции клиента

Прежде чем получить возможность взаимодействовать с системой, клиенту сперва необходимо зарегистрироваться. Изображенная на рисунке 22 форма, принимает данный в виде пары ключ значение, значениями же являются: логин, пароль и подтверждение пароля. После успешной регистрации, клиент получает статус код ОК и уведомление об успешной регистрации.

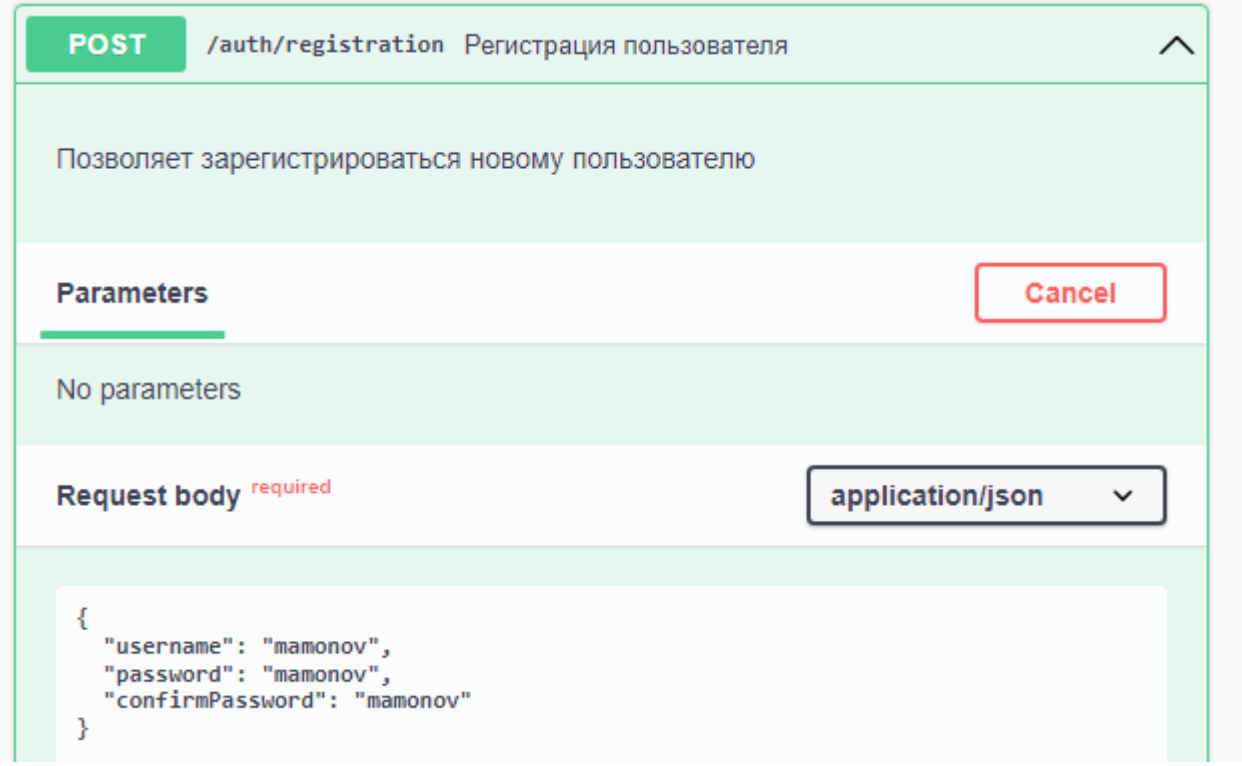

Рисунок 22 – Форма регистрации

После успешной регистрации, клиент может перейти к форме авторизации, на рисунке 23 приведена форма авторизации зарегистрированного пользователя. Форма принимает следующие данные: логин, пароль. После успешной авторизации, пользователю присваивается JWT-токен, этот токен временный и является токеном доступа к функциям приложения. Программный код класса, отвечающего за регистрацию, авторизацию и генерацию JWT-токена приведен в приложении В. «JSON Web Token (JWT) — это открытый стандарт (RFC 7519), определяющий компактный и автономный способ безопасной передачи информации между сторонами в виде объекта JSON. Эту информацию можно проверить и ей можно доверять, поскольку она имеет цифровую подпись. JWT можно подписать с использованием секрета (с помощью алгоритма HMAC) или пары открытого/закрытого ключей с использованием RSA или ECDSA.»[16]

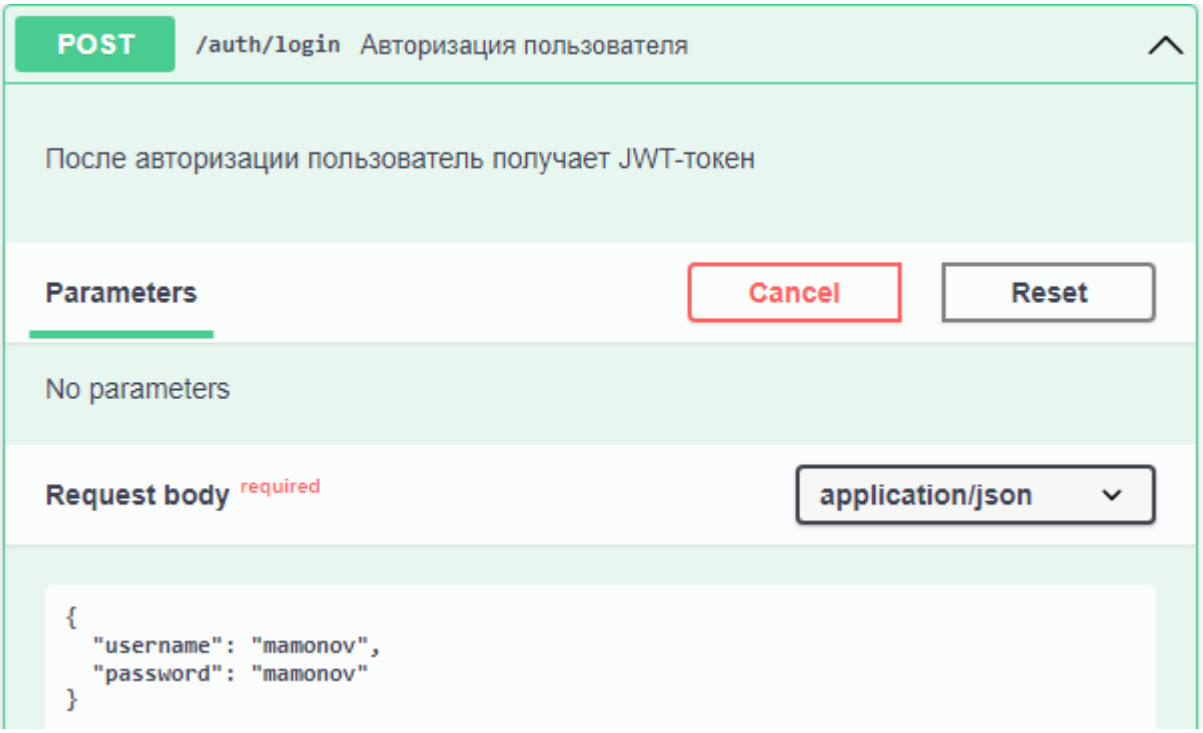

Рисунок 23 – Форма авторизации

Как только пользователь авторизуется, ему становятся доступны функции приложения, в соответствии с его правами доступа. По умолчанию всем пользователями доступна функция просмотра информации об учётной записи. На рисунке 24 приведена страница с информацией о зарегистрированном пользователе.

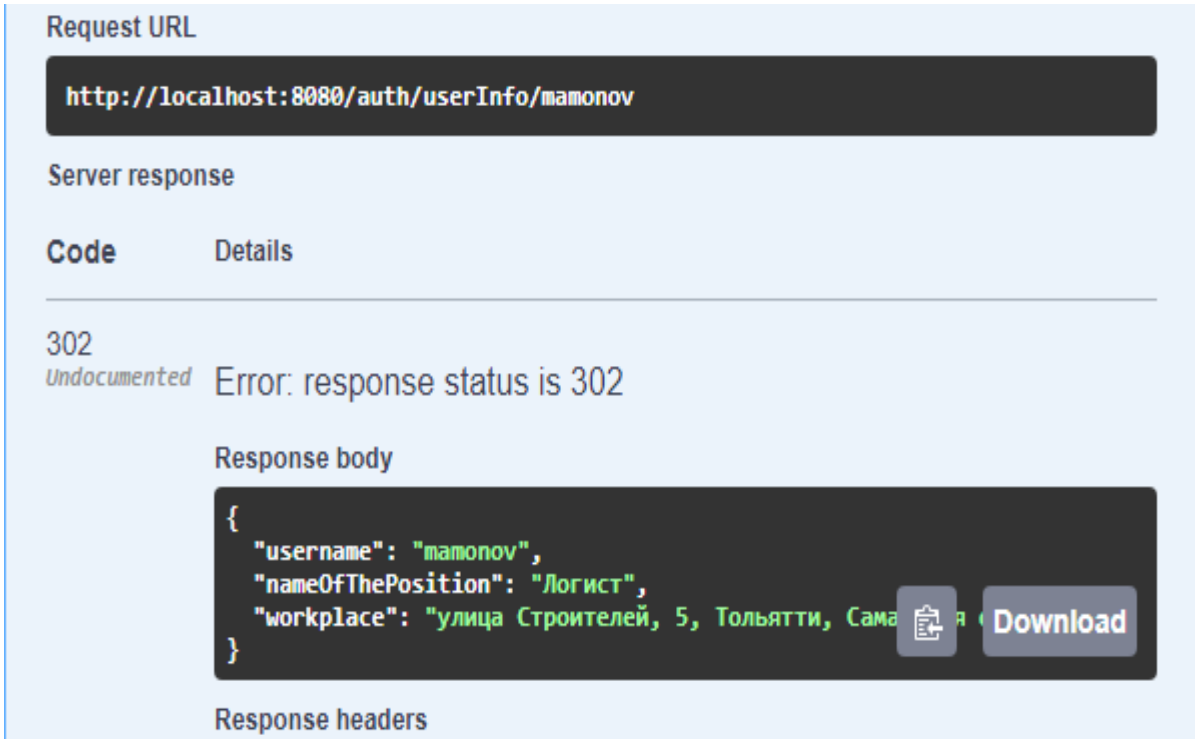

Рисунок 24 – Страница с информацией о пользователе

Помимо этого, авторизованному пользователю, доступны основные функции работы с заявками, в числе которых оформление новой заявки на поставку. На рисунках 25 изображена форма оформления заявки на поставку.

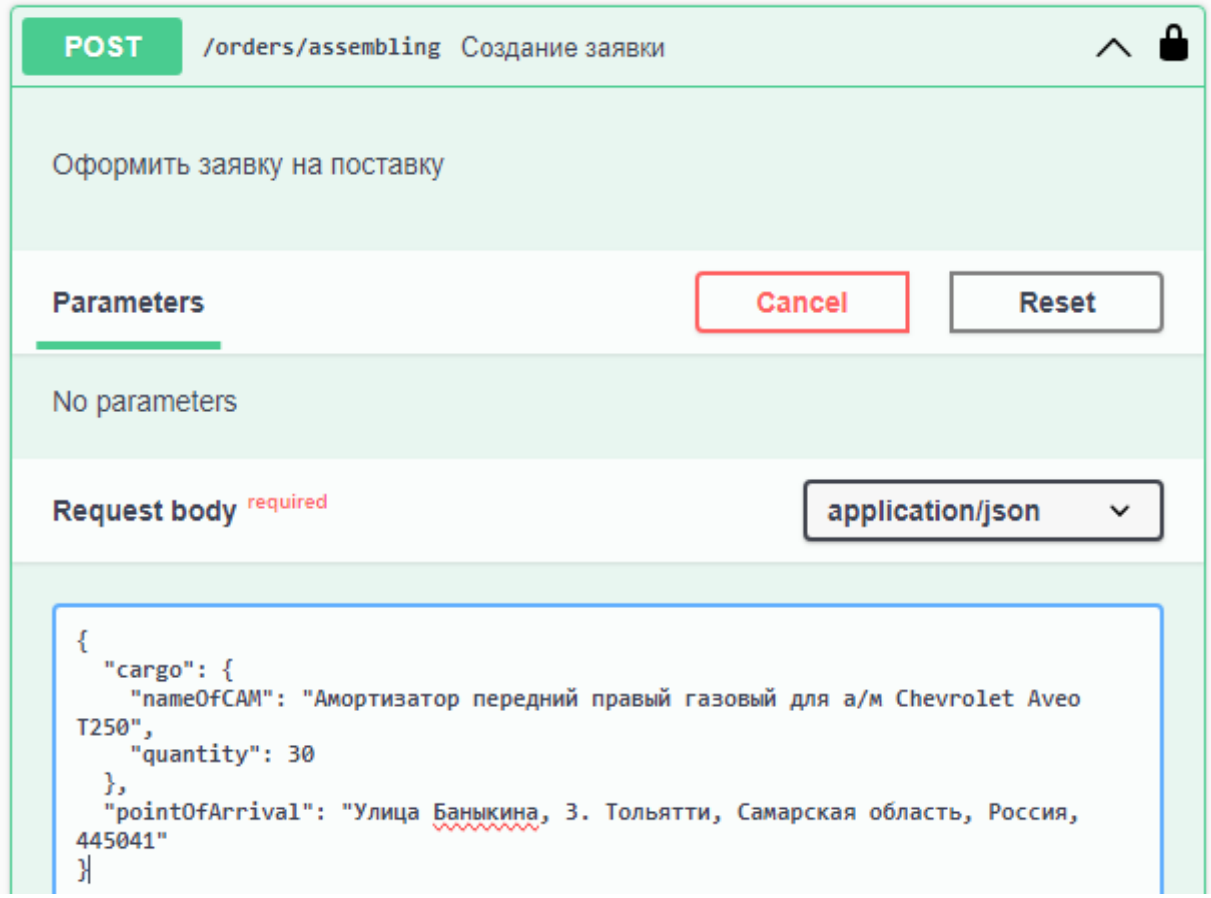

Рисунок 25 – Оформление заявки пользователем

Форма принимает данные, о необходимом товаре, количество товара, и место куда необходима доставить товар. В качестве ответа пользователь получает уведомление об спешном создании заявки и ее номер заявки.

В соответствии с правилами валидации, в случае, если будут указаны некорректные данные, пользователь получит соответствующий ответ, где будет указано: причина ошибки и статус-код.

После того, как пользователь отправил заявку, логист может просмотреть ее. На рисунке 26 изображена страница просмотра последних пришедших заявок.

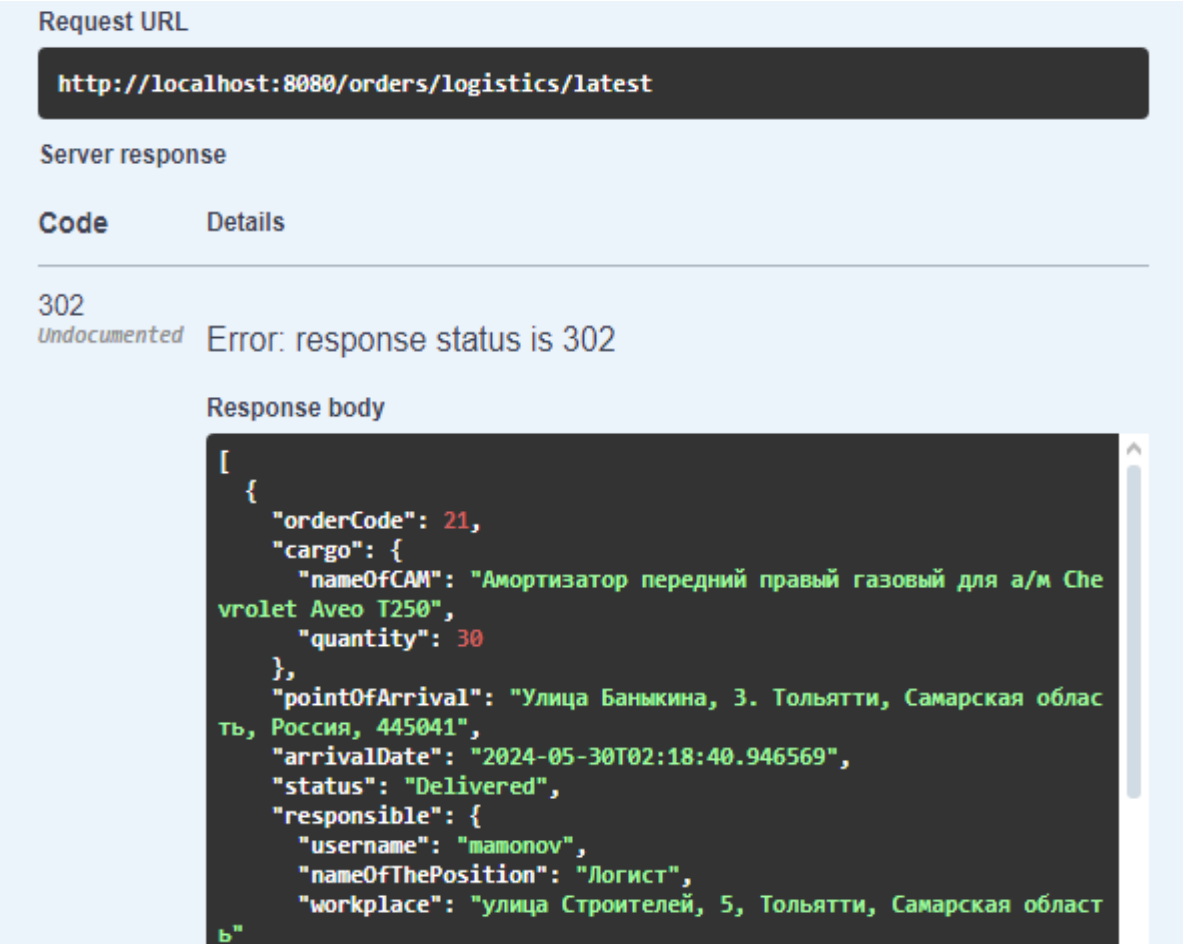

Рисунок 26 – Просмотр последних пришедших заявок логистом

Получаемые логистом данные содержат данные о поставке, статус заявки, ответственное лицо, место отбытия. Все данные, за исключением данных о поставку логист должен указать самостоятельно в процессе обработки заявки.

После получения логистом заявки, он может изменить ее статус. На рисунке 27 изображена страница для изменения статуса заявки. Логист должен передать номер заявки и ее статус. В качестве ответа, логист получит сообщение об успешном изменении статуса заявки.

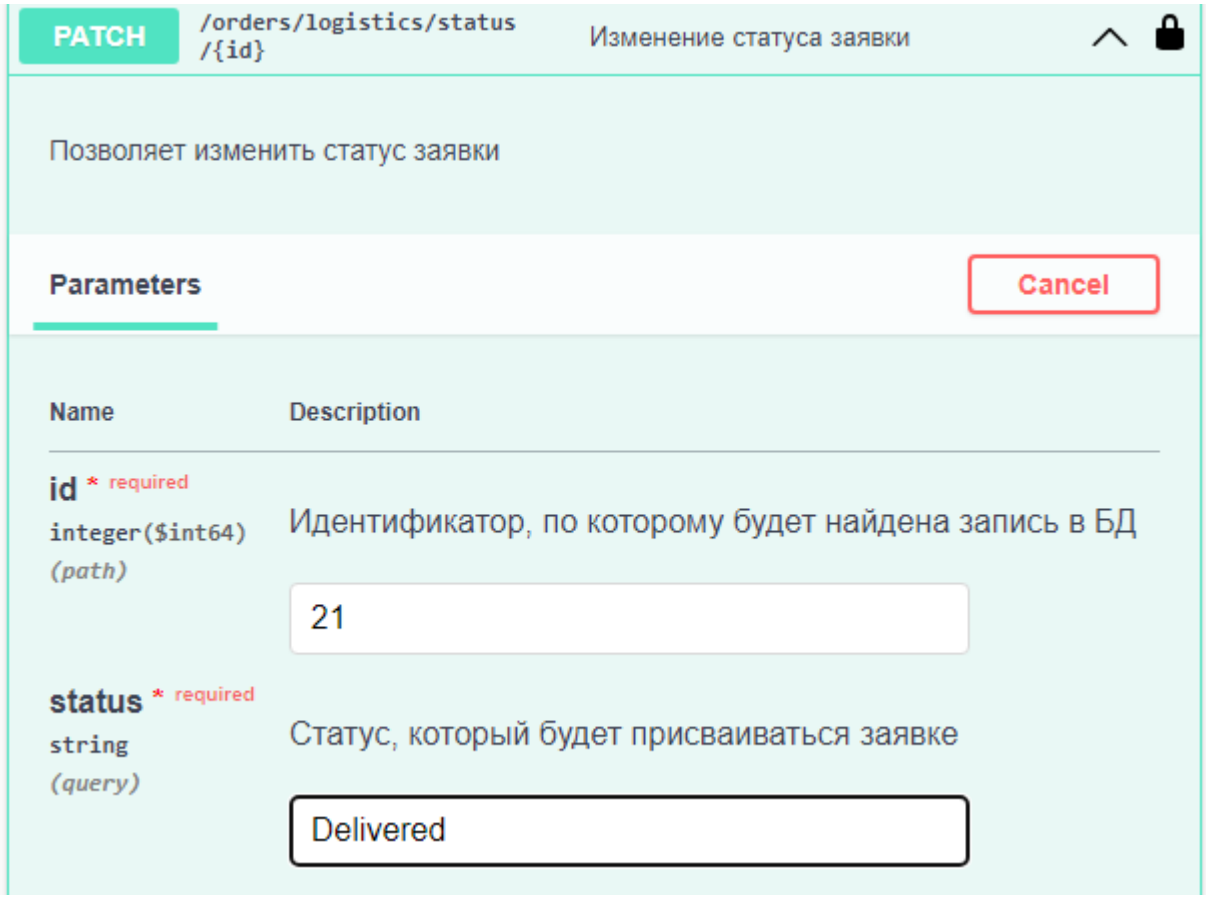

Рисунок 27 – Изменение статуса заявки логистом

Пользователь также сможет просмотреть отправленную им заявку и проверить ее статус. Чтобы просмотреть заявку, пользователь должен указать ее номер. На рисунке 28 изображена страница с просмотром отправленной пользователем заявки.

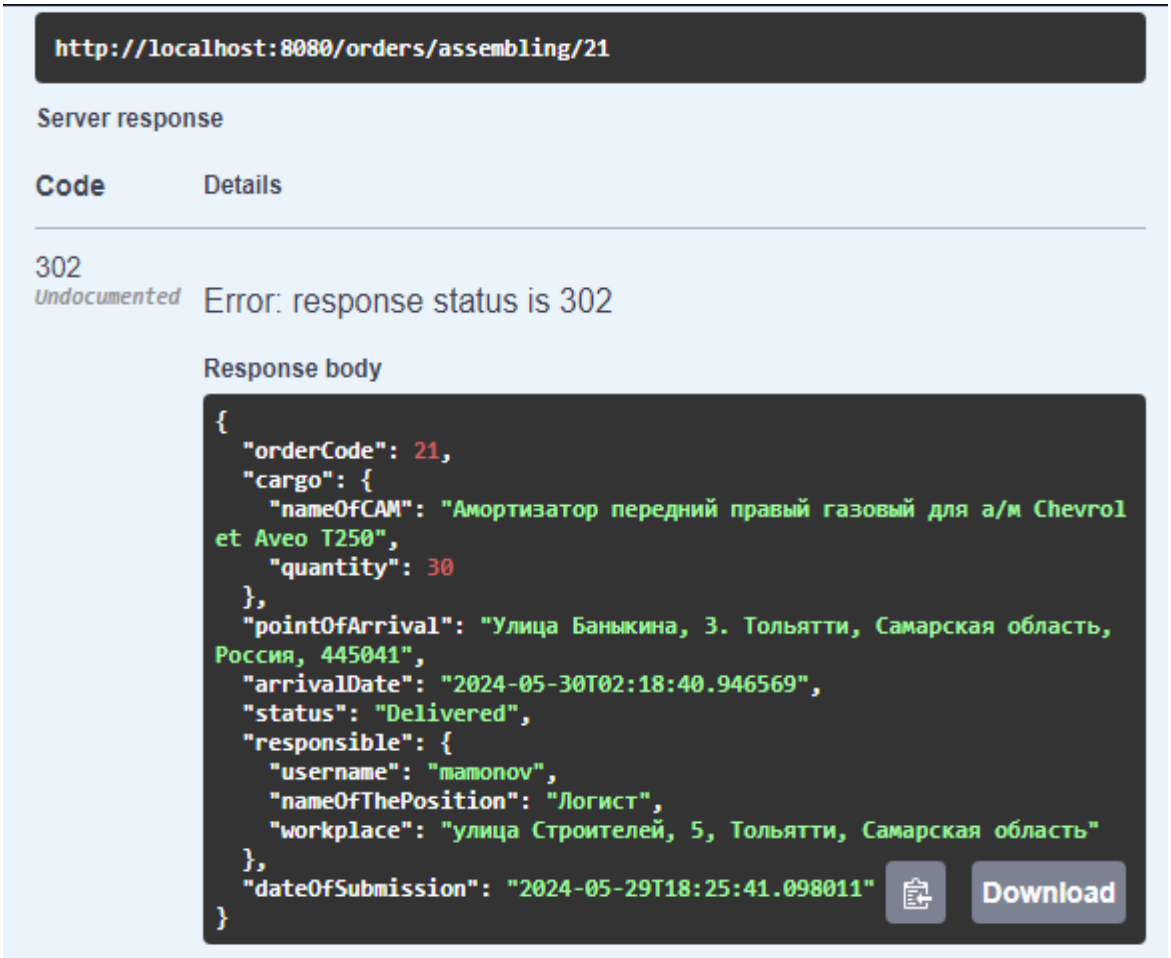

Рисунок 28 – Просмотр отправленной заявки и проверка статуса заявки клиентом

Страница содержит информацию по поставке, которую указал сам клиент. Так же в заявке будут указаны: статус заявки, ответственный за ее выполнение логист, место отправки, дата прибытия поставки в пункт назначения.

В процессе обработки заявки, логист должен проверить наличие необходимого товара на складе готовой продукции. На рисунке 29 изображена страница для просмотра остатков товара на складе предприятия.

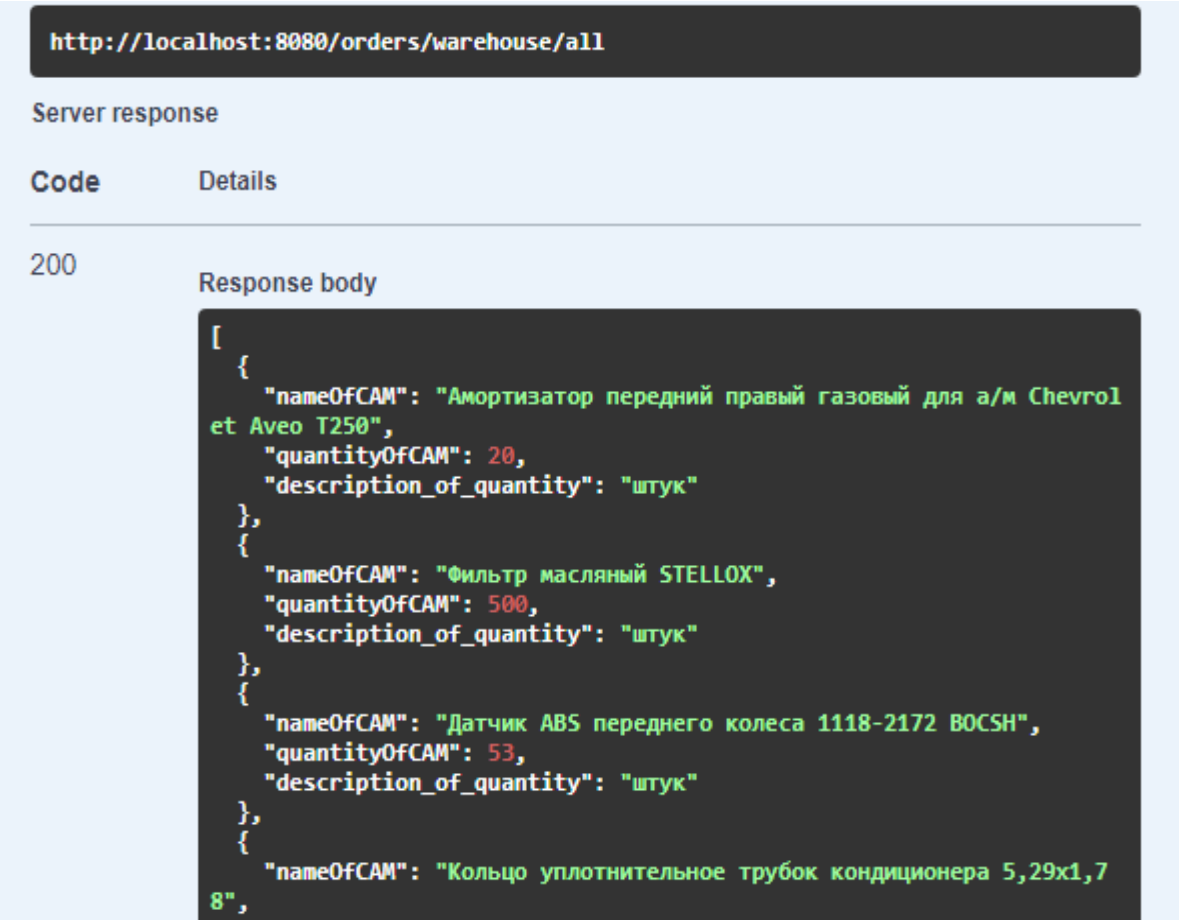

Рисунок 29 – Страница для проверки наличия товара на складе

Страница выводит список из товаров, находящихся на складе. Информация содержит: наименование товара, количество товара и количественная единица.

После обработки заявки, логист должен сформировать дозаявку на поставку товара. На рисунке 30 демонстрируется форма для формирования логистом дозаявки на поставку.

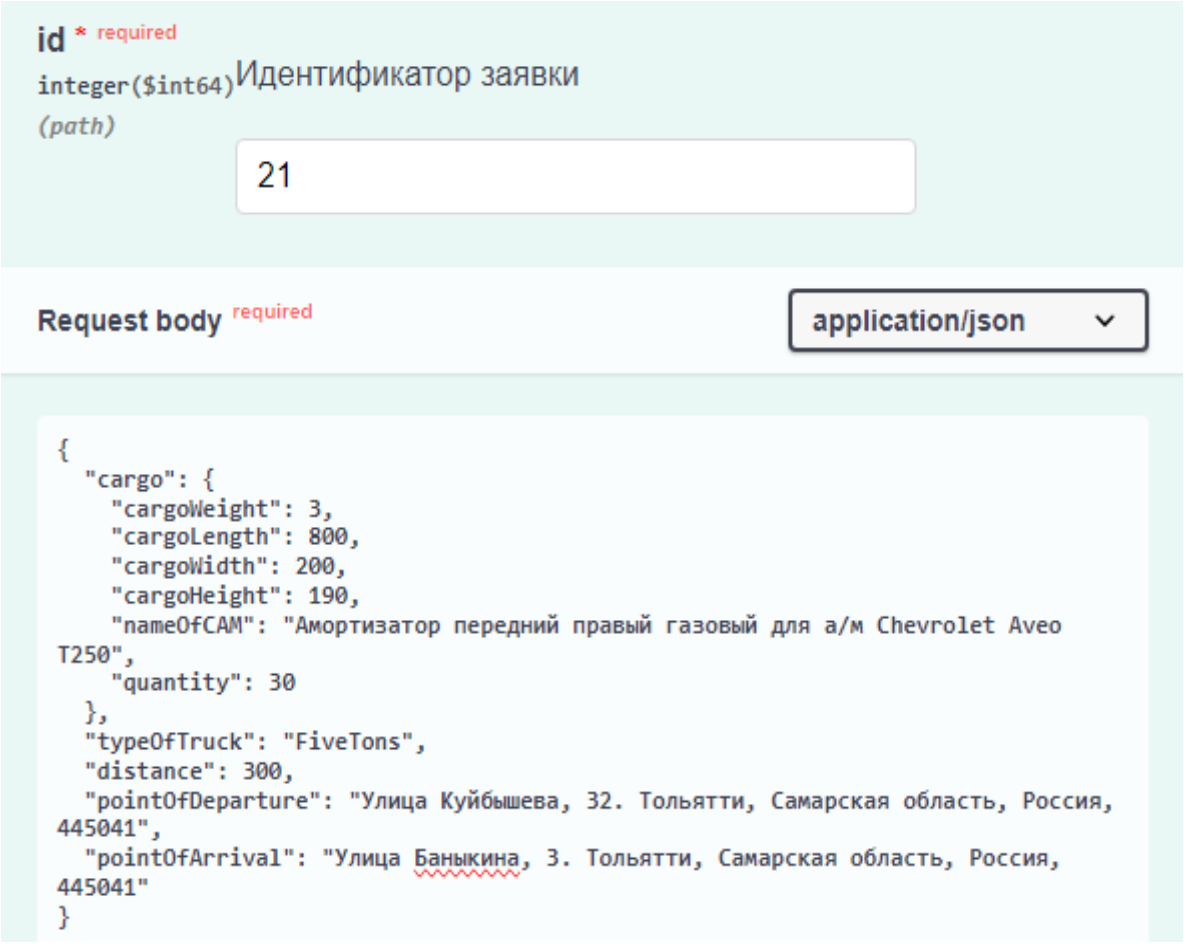

Рисунок 30 – Форма для формирования дозаявки логистом

Форма принимает следующие данные: наименование поставки, количество, адрес, куда будет доставляться товар, габариты груза, расстояние от пункта отправки до пункта назначения, адрес пункта отправки. Если товара присутствует на складе, то дозаявка будет перенаправлена на склад готовой продукции.

Сформированная дозаявка отправляется в необходимое структурное подразделение предприятия, где будет происходить процесс подготовки товара/комплектующих для отправки. На рисунке 31 изображена страница с сформированной дозаявкой.

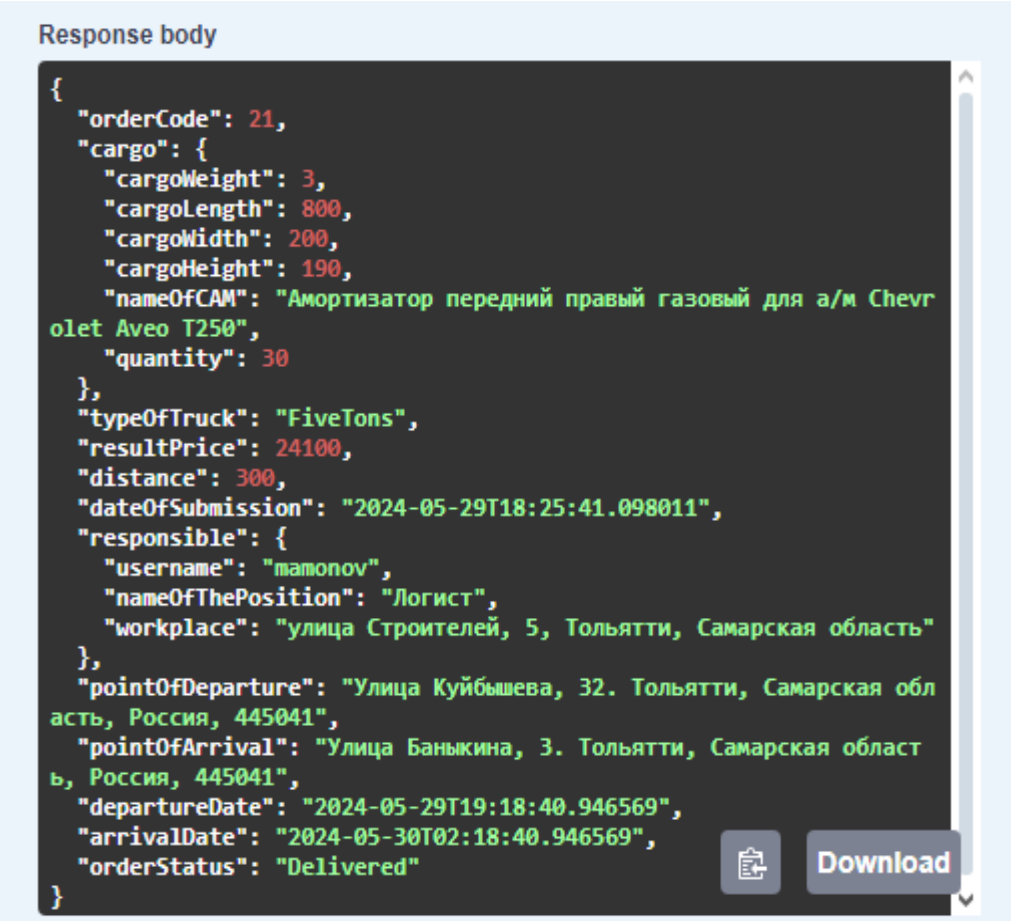

Рисунок 31 - Сформированная дозаявка

После выполнения заявки, логист так же может изменить статус заявки на «выполнена». Информация, содержащиеся в сформированной дозаявке в дальнейшем будет использоваться для составления отчётом отделом логистики.

Вывод по главе 3

По итогам главы, было описано разработанное REST веб-приложение службы логистики, его структура, пакеты, используемые технологии и его база данных. Так же были наглядно продемонстрированы функции приложения, заполняемые формы, передаваемые данные и принцип администрирования приложения посредством JWT.

Заключение

По итогам проделанной работы, был выполнен список следующих задач:

- изучена деятельность предприятия ИП «Стеблев Д.В», его структура;
- дана краткая характеристика отделу логистики предприятия;
- проанализированы бизнес-процессы предприятия и на их основе произведено концептуальное проектирование бизнес-процессов предприятия;
- проанализированы существующие реализации, среди похожих систем, которые можно потенциально внедрить в систему предприятия;
- обоснована и сформулирована задача на разработку и администрирование REST веб-приложения службы логистики;
- построены модели данных, ориентируясь на предметную область;
- спроектирована база данных веб-приложения;
- описаны технологии, используемые при разработке;
- описано разработанное веб-приложение;
- продемонстрирована работа веб-приложение, заполняемые формы, функции.

Веб-приложение для службы логистики, поможет автоматизировать процессы формирования и обработки заявок для отдела логистики предприятия ИП «Стеблев Д.В».

Спроектированное хранилище данных – база данных веб-приложения, позволила упростить подготовку отчетов о проделанной работе и сократить количество возможных ошибок.

После внедрения веб-приложения отдел логистики сможет обрабатывать большие объемы заявок с большей скоростью, что поможет улучшить производительность всего предприятия.

Список используемой литературы и используемых источников

1. А что такое Spring и как он устроен [Электронный ресурс]. URL: https://ru.hexlet.io/blog/posts/spring-framework (дата обращения: 26.04.2024)
2. Джуба С., Волков А.Д40 Изучаем PostgreSQL 10 / пер. с анг. А. А. Слинкина. – М.: ДМК Пресс, 2019. – 400 с.: ил.
3. Дубейковский В. И. Практика функционального моделирования с AllFusion Process Modeler. М.: ДИАЛОГ-МИФИ, 2021. 464 с., с. 382
4. Модели As is & To be [Электронный ресурс]. URL: https://blog.bitobe.ru/article/modeli-as-is-to-be/
5. Модели данных в СУБД [Электронный ресурс]. URL: https://appmaster.io/ru/blog/modeli-dannykh-v-subd (дата обращения: 27.05.2024)
6. Нотация UML: принципы, особенности, примеры диаграмм [Электронный ресурс]. URL: https://www.comindware.ru/blog/uml-notation-overview/ (дата обращения: 27.05.2024)
7. Невзорова О.А., Миннегалиева Ч.Б. Основы UML / О.А. Невзорова, Ч.Б. Миннегалиева. – Казань: Казан.ун-т, 2021. – 43 с.
8. Нотации моделирования бизнес-процессов, Электронный ресурс]. URL: https://leanvector.ru/blog-eksperta/notatsii-modelirovaniya-biznes-protsessov/ (дата обращения: 27.05.2024)
9. Моисеев А.Н., Литовченко М.И. М74 Основы языка UML : учеб. пособие. – Томск : ИздательствоТомского государственного университета, 2023. – 96 с
10. Сущности и связи: как и для чего системные аналитики создают ER-диаграммы [Электронный ресурс]. URL: https://practicum.yandex.ru/blog/chto-takoe-er-diagramma/
(дата обращения: 30.05.2024)

11. Что такое Java? Определение, значение и особенности [Электронный ресурс]. URL: https://appmaster.io/ru/blog/chto-takoe-java-opredelenie-znachenie-osobennosti (дата обращения: 28.05.2024)

12. Что такое нормализация базы данных [Электронный ресурс]. URL: https://wiki.merionet.ru/articles/chto-takoe-normalizaciya-bazy-dannyh (дата обращения: 20.04.2024)

13. Якобсон А. Унифицированный процесс разработки программного обеспечения СПб.: Питер, 2019. 496 с., с. 294

14. Entity-Relationship Diagram Symbols and Notation [Электронный ресурс]. URL: https://www.lucidchart.com/pages/ER-diagram-symbols-and-meaning (дата обращения: 31.05.2024)

15. Entity-Relationship-Diagram [Электронный ресурс]. URL: https://michael-fuchs-sql.netlify.app/2021/03/03/entity-relationship-diagram-erd/#entity (дата обращения: 31.05.2024)

16. Introduction to JSON Web Tokens [Электронный ресурс]. URL: https://jwt.io/introduction (дата обращения: 31.05.2024)

17. Java Persistence API (JPA) For Database Access [Электронный ресурс]. URL: https://www.turing.com/kb/jpa-for-database-access (дата обращения: 31.05.2024)

18. Spring in Action 6th Edition by Craig Walls, с. 164

19. What is IDEF? [Электронный ресурс]. URL: https://www.edrawsoft.com/what-is-idef.html (дата обращения: 31.05.2024)

20. What is REST? [Электронный ресурс]. URL: https://www.codecademy.com/article/what-is-rest (дата обращения: 31.05.2024)

Приложение А

Код сущности «заявка»

```
@Entity
@Table(name = "orders")
@AllArgsConstructor
@RequiredArgsConstructor
@Getter
@Setter
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "order_id")
    private Long orderId; //код заявки

    @OneToOne
    @JoinColumn(name = "cargo_id", referencedColumnName = "cargo_id")
    @Cascade(org.hibernate.annotations.CascadeType.ALL)
    private Cargo cargo;

    @Column(name = "type_of_truck")
    @Enumerated(EnumType.STRING)
    private TypeOfTruck typeOfTruck;//тип перевозки

    @Column(name = "result_price")
    private Double resultPrice;

    @Column(name = "distance")
    private Double distance;//дистанция (в км)
```

```
@Column(name = "date_of_submission")
    private LocalDateTime dateOfSubmission;//дата оформления заявки(берется время и дата сервера)

    @ManyToOne
    @JoinColumn(name = "responsible", referencedColumnName = "user_name")
    private User responsible;//связь с юзером(ответственный по заявке)

    @Column(name = "point_of_departure")
    private String pointOfDeparture;//точка отправления

    @Column(name = "point_of_arrival")
    private String pointOfArrival;//точка прибытия

    @Column(name = "departure_date")
    private LocalDateTime departureDate;//дата отправки

    @Column(name = "arrival_date")
    private LocalDateTime arrivalDate;//дата прибытия

    @Column(name = "order_status")
    @Enumerated(EnumType.STRING)
    private OrderStatus orderStatus;
}
```

Приложение Б

Код сервисного слоя

```
@Service
@Slf4j
public class OrderServiceImpl implements OrderService {
    private final Calculate calculate;
    private final OrderRepository orderRepository;
    private final CargoServiceImpl cargoService;
    private final OrderMapper orderMapper;
    private final CargoMapper cargoMapper;

    @Autowired
    public OrderServiceImplV2(Calculate calculate, OrderRepository orderRepository, CargoServiceImpl cargoService, OrderMapper orderMapper, CargoMapper cargoMapper) {
        this.calculate = calculate;
        this.orderRepository = orderRepository;
        this.cargoService = cargoService;
        this.orderMapper = orderMapper;
        this.cargoMapper = cargoMapper;
    }

    @Override
    public List<ResponseOrderDtoForLogistics> findAllOrdersForLogistic() {
        return orderMapper.mapListOfOrderToListOfResponseOrderDtoForLogistics(orderRepository.findAll());
    }
```

```
 @Override
    public List<ResponseOrderDtoForWarehouse> findAllOrdersForWareHouse() {
        return
orderMapper.mapListOfOrderToListOfResponseOrderDtoForWarehouse(orderRe
pository.findAll());
    }

    @Override
    public                                    List<ResponseOrderDtoForAssembling>
findAllOrdersForWareAssembling() {
        return
orderMapper.mapListOfOrderToListOfResponseOrderDtoForAssembling(orderRe
pository.findAll());
    }

    @Override
    public List<Order> findAndSortAllOrders(Sort sortDir) {
        return orderRepository.findAll(sortDir);
    }

    @Override
    public Optional<Order> findAnyOrderByID(Long id) {
        return orderRepository.findById(id);
    }

    @Override
    @Transactional
    public void saveOrder(RequestOrderForCreate order) {
```

```
 Order orderForCreate =
orderMapper.mapRequestOrderForCreateToEntity(order);
      orderForCreate.setDateOfSubmission(LocalDateTime.now());
      orderRepository.save(orderForCreate);
      log.info("current order - {} has been saved", order);
   }

   @Override
   @Transactional
   public void deleteOrder(Long id) {
      orderRepository.deleteById(id);
   }

   @Override
   public void updateOrder(RequestOrderForUpdate requestOrder, Long id) {
      if (orderRepository.findById(id).isEmpty()) {
         throw new EntityNotFoundException("Order with id: " + id + " not found");
      }

      Order updatedOrder = orderRepository.findById(id).orElse(null);
      Cargo cargo = updatedOrder.getCargo();

      if (requestOrder.getTypeOfTruck() != null) {
         if (requestOrder.getTypeOfTruck() == TypeOfTruck.ThreeTons &&
cargo.getCargoWeight() > 3) {
            throw new IncorrectDataException("Тип транспорта не соответствует
весу груза");
```

```
        } else if (requestOrder.getTypeOfTruck() == TypeOfTruck.FiveTons &&
cargo.getCargoWeight() > 5) {
            throw new IncorrectDataException("Тип транспорта не соответствует
весу груза");
        } else if (requestOrder.getTypeOfTruck() == TypeOfTruck.TenTons &&
cargo.getCargoWeight() > 10) {
            throw new IncorrectDataException("Тип транспорта не соответствует
весу груза");
        } else if (requestOrder.getTypeOfTruck() == TypeOfTruck.TwentyTons
&& cargo.getCargoWeight() > 20) {
            throw new IncorrectDataException("Вес груза превышает норму для
транспорта");
        } else
            updatedOrder.setTypeOfTruck(requestOrder.getTypeOfTruck());
    }
    if (requestOrder.getPointOfArrival() != null) {
        updatedOrder.setPointOfArrival(requestOrder.getPointOfArrival());
    }
    if (requestOrder.getDistance() != null) {
        updatedOrder.setDistance(requestOrder.getDistance());
    }
    if (requestOrder.getCargo() != null) {

updatedOrder.setCargo(cargoService.updateCargo(cargoMapper.mapCargoDtoTo
Entity(requestOrder.getCargo()), cargo.getCargoId()));
    } else throw new IncorrectDataException("incorrect data");
    orderRepository.save(updatedOrder);
```

```
 log.info("requestOrder with id: {} has been updated", updatedOrder.getOrderId());
  }

  @Override
  public List<Order> findByPointOfArrival(String pointOfArrival) {
     return null;
  }

  @Override
  public List<Order> findByResponsible(String responsible) {
     return null;
  }

  @Transactional
  public void changeStatus(Long id, String status) {
     if (orderRepository.findById(id).isEmpty()) {
        throw new EntityNotFoundException("Order with id: " + id + " not found");
     }
     Order orderForChange = orderRepository.findById(id).get();

     if (status.equalsIgnoreCase("Executed"))
orderForChange.setOrderStatus(OrderStatus.Executed);
     if (status.equalsIgnoreCase("Delivered"))
orderForChange.setOrderStatus(OrderStatus.Delivered);
     if (status.equalsIgnoreCase("Completed"))
orderForChange.setOrderStatus(OrderStatus.Completed);
  }
```

```
private void changeOrderForUpdate(Order order) {
        if (order.getDistance() != null) {
            order.setResultPrice(calculate.calculate(order));
        }

        if (order.getOrderStatus().equals(OrderStatus.Delivered)) {
            if (order.getDistance() > 500) {
                order.setArrivalDate(LocalDateTime.now().plusDays((long)
(order.getDistance() / 500)));
            } else {
                order.setArrivalDate(LocalDateTime.now().plusHours((long)
(order.getDistance() / 60)).plusHours(2));
            }
            order.setDepartureDate(LocalDateTime.now());
        } else order.setOrderStatus(OrderStatus.Accepted);

        order.setDateOfSubmission(LocalDateTime.now());

    }
}
```

Приложение В

Код класса авторизации и регистрации

```
@Service
public class AuthServiceImpl implements AuthService {
   private final UserServiceImpl userService;
   private final JwtTokenUtils jwtTokenUtils;
   private final AuthenticationManager authenticationManager;

   @Autowired
   public AuthServiceImpl(UserServiceImpl userService, JwtTokenUtils jwtTokenUtils, AuthenticationManager authenticationManager) {
      this.userService = userService;
      this.jwtTokenUtils = jwtTokenUtils;
      this.authenticationManager = authenticationManager;
   }


   @Override
   public ResponseEntity<?> createAuthToken(@RequestBody JwtRequest authRequest) {
      try {
         authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(authRequest.getUsername(), authRequest.getPassword()));
      } catch (BadCredentialsException e) {
         return new ResponseEntity<>(new JwtException("Невверные логин или пароль"), HttpStatus.UNAUTHORIZED);
      }
```

```
 UserDetails userDetails = userService.loadUserByUsername(authRequest.getUsername());
        String token = jwtTokenUtils.generateToken(userDetails);
        return ResponseEntity.ok(new JwtResponse(token));
    }

    @Override
    public ResponseEntity<?> createNewUser(@RequestBody RegistrationUserDto registrationUserDto) {
        if (!registrationUserDto.getPassword().equals(registrationUserDto.getConfirmPassword())) {
            return new ResponseEntity<>(new IncorrectDataException("Пароли не совпадают"), HttpStatus.BAD_REQUEST);
        }
        if (userService.findByUsername(registrationUserDto.getUsername()).isPresent()) {
            return new ResponseEntity<>(new IncorrectDataException("Пользователь с таким логинов уже зарегистрирован"), HttpStatus.BAD_REQUEST);
        }
        User user = userService.createNewUser(registrationUserDto);
        return ResponseEntity.ok(new UserDto(user.getUsername(), user.getNameOfThePosition(), user.getWorkplace()));
    }
}
```