|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**НА ТЕМУ:**

**Создание Веб-приложения для создания Дата-центра  
Дисциплина «Разработка интернет-приложений»**

Студент ИУ5-51Б \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.Ю.Ильин

(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.И.Канев

(Подпись, дата) (И.О.Фамилия)

2024г.

# АННОТАЦИЯ

Расчётно-пояснительная содержит 28 страниц. С приложениями объем составляет 48 страницы. Работа включает в себя 8 диаграмм и 16 изображений системы. В процессе выполнения было использовано 9 источников.

Объектом разработки является система создания дата-центра. Данная система предназначена для хранения и управления данными о заказах и товаров дата-центра.

Цель системы - дать возможность пользователям сформировать Дата-центр и редактировать его.

В работе была разработана архитектура веб сервиса, рассчитанного на создание дата-центра, разработан интерфейс для общения с данным веб сервисом, разработан и развернут веб сервер, нативное приложение и прогрессивное веб приложение, способные общаться с данным веб сервисом.

Пояснительная записка содержит 2 приложения.

# СОДЕРЖАНИЕ

[АННОТАЦИЯ 1](#_Toc185419960)

[СОДЕРЖАНИЕ 2](#_Toc185419961)

[ВВЕДЕНИЕ 3](#_Toc185419962)

[1 ПРЕДМЕТНАЯ ОБЛАСТЬ 5](#_Toc185419963)

[2 АРХИТЕКТУРА 10](#_Toc185419964)

[3 АЛГОРИТМЫ 14](#_Toc185419965)

[4 ОПИСАНИЕ ИНТЕРФЕЙСА 16](#_Toc185419966)

[ЗАКЛЮЧЕНИЕ 20](#_Toc185419967)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc185419968)

[ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ 22](#_Toc185419969)

[1 Введение 23](#_Toc185419970)

[2 Назначение разработки 23](#_Toc185419971)

[3 Стадии и этапы разработки 23](#_Toc185419972)

[4 Требования к функциональным характеристикам 24](#_Toc185419973)

[5 Требования к составу и параметрам технических средств 29](#_Toc185419974)

[6 Требования к составу и параметрам технических средств 29](#_Toc185419975)

[ПРИЛОЖЕНИЕ Б СПИСОК HTTP МЕТОДОВ 0](#_Toc185419976)

# ВВЕДЕНИЕ

В эпоху стремительного развития информационных технологий, когда цифровые данные и услуги становятся неотъемлемой частью нашей повседневной жизни, дата-центры приобретают особое значение как ключевые элементы инфраструктуры. Они играют ключевую роль в обеспечении надежности и безопасности данных, позволяя компаниям эффективно управлять своими ресурсами и поддерживать бесперебойную работу. В условиях цифровой экономики дата-центры становятся основным элементом для хранения и обработки данных, предоставляя компаниям конкурентные преимущества и возможность быстро адаптироваться к изменениям на рынке. Поэтому его наличие и эффективное функционирование крайне важны для успешного ведения бизнеса.

Цель системы, включающей в себя веб-сервис, веб-приложение, нативное приложение, дать возможность пользователям сформировать Дата-центр и редактировать его.

Данная система предназначена для хранения и управления данными о заказах и товаров дата-центра. Клиенты могут создавать новые заказы на товары или оборудование, отслеживать статус своих заявок. Операторы выполняют модерацию заказов, подтверждая или отклоняя их в зависимости от их корректности. Администраторы могут редактировать и добавлять новые услуги и товары в систему, а также просматривать и управлять всеми заявками клиентов, обеспечивая их актуальность и полноту.

Нефункциональные требования к разрабатываемой системе:

1. Должна поддерживаться кроссплатформенность.
2. Интерфейс системы и текст ошибок должны быть русифицированы.

В ходе работы необходимо выполнить следующие задачи:

* 1. Создание MVP и базового дизайна на основе datahouse.ru;
  2. Реализовать хранение данных в БД (Postgres);
  3. Создать веб-сервис на бэкенде (DRF);
  4. Реализовать авторизацию и хранение сессий в Redis;
  5. Разработать SPA на React для гостя;
  6. Внедрить адаптивность, менеджер состояний Redux Toolkit, PWA, разработать Tauri приложение;
  7. Завершить разработку интерфейса пользователя на React, использовать для обращений к методам веб-сервиса Axios;
  8. Добавить в приложение React интерфейс администратора;
  9. Реализовать нативное приложение;
  10. Развернуть веб-приложение в GitHub Pages;
  11. Подготовка всей документации (РПЗ, ТЗ и набор диаграмм);
  12. Подготовка репозитория на GitHub.

# ПРЕДМЕТНАЯ ОБЛАСТЬ

Дата-центр — специализированный объект, представляющий собой связанную систему ИТ-инфраструктуры, части которой размещены в здании или помещении, подключенной к внешним сетям, как инженерным, так и телекоммуникационным [1]. Он выступает в роли невидимого столпа, обеспечивающего стабильную и бесперебойную работу интернета, телекоммуникационных сетей и облачных сервисов. Без дата-центров невозможно представить функционирование современных цифровых систем, которые стали основой для множества бизнес-процессов и коммуникаций. Эти сложные и высокотехнологичные комплексы представляют собой настоящие инженерные шедевры, спроектированные и функционирующие с учетом огромного количества факторов. Каждая деталь тщательно продумана, чтобы обеспечить максимальную надежность и эффективность. В условиях возрастающей нагрузки и требований к безопасности, дата-центры продолжают развиваться и адаптироваться, оставаясь незаменимыми для поддержания стабильности и прогресса в цифровом мире.

Важным аспектом работы дата-центров является обеспечение безопасности и конфиденциальности данных. Современные дата-центры используют системы резервного копирования, восстановления данных, защиты каналов передачи информации, включающие шифрование, аутентификацию и контроль доступа, чтобы предотвратить утечки информации [2].

С увеличением числа кибератак дата-центры становятся ключевыми объектами инфраструктуры, поскольку они хранят и обрабатывают значительные объемы данных, которые имеют критическое значение для различных организаций. В условиях возрастающих угроз кибербезопасности, внедрение передовых технологий, таких как искусственный интеллект (ИИ) и машинное обучение (МО), становится необходимостью. Эти технологии позволяют дата-центрам эффективно защищать данные и обеспечивать непрерывность работы критически важных сервисов и систем. Искусственный интеллект и машинное обучение анализируют огромные массивы данных в режиме реального времени, выявляя потенциальные угрозы и аномалии. Они способны предсказывать атаки, что позволяет предпринимать меры по их предотвращению, значительно повышая уровень кибербезопасности. Современные системы мониторинга и реагирования, основанные на ИИ и МО, способны быстро адаптироваться к новым угрозам и методам их обхода, что делает их незаменимыми инструментами в борьбе с киберугрозами. Таким образом, дата-центры не только обеспечивают защиту данных, но и гарантируют бесперебойную работу критически важных сервисов и систем. Инвестиции в искусственный интеллект и машинное обучение становятся неотъемлемой частью стратегии кибербезопасности. Эти технологии помогают дата-центрам оставаться устойчивыми и защищенными в условиях быстро меняющейся киберсреды, где угрозы становятся все более изощренными и сложными.

Каждая отрасль имеет свои уникальные требования к уровню отказоустойчивости, скорости доступа к данным, безопасности и соответствию нормативным актам. Например, в финансовой сфере требуется высокая степень защиты данных и мгновенный доступ к информации для проведения транзакций. В здравоохранении критически важно обеспечить надежность систем, чтобы избежать ошибок в лечении и обеспечить конфиденциальность медицинских данных. В промышленности необходимо учитывать строгие стандарты безопасности и быстро реагировать на аварийные ситуации. В сфере телекоммуникаций требуется высокая пропускная способность и минимальное время отклика для обеспечения стабильного подключения. В розничной торговле важна быстрая обработка заказов и возможность работы с большими объемами данных.

Для достижения этих целей была разработана наша система, предоставляющая интерфейс для гибкого конфигурирования Дата центра.

Пользователь может выбрать оборудование для своего Дата центра, указать адрес и время доставки, сформировать свой заказ и отправить его на подтверждение администратору. Каждый Заказ в начале создается как черновик, в который можно добавлять и из которого можно удалять оборудование. Один заказ может содержать в себе несколько единиц оборудования; тогда, в случае одобрения заказа, можно будет оплатить все товары из списка. Для заказа необходимо указать время и адрес, на который осуществляется доставка.

Пользователь может посмотреть историю своих заказов. В случае, если заказ составлен корректно, администратор одобряет его. В противном случае – отклоняет.

Администратор может редактировать список предоставляемого оборудования: добавлять новые комплектующие, удалять неактуальные, а также редактировать информацию о каждой существующей позиции. Администратор имеет возможность просматривать все заказы всех пользователей, фильтровать их по имени пользователя, дате формирования и статусу.

Функции пользователей с различными ролями описаны на диаграмме прецедентов (рисунок 1).

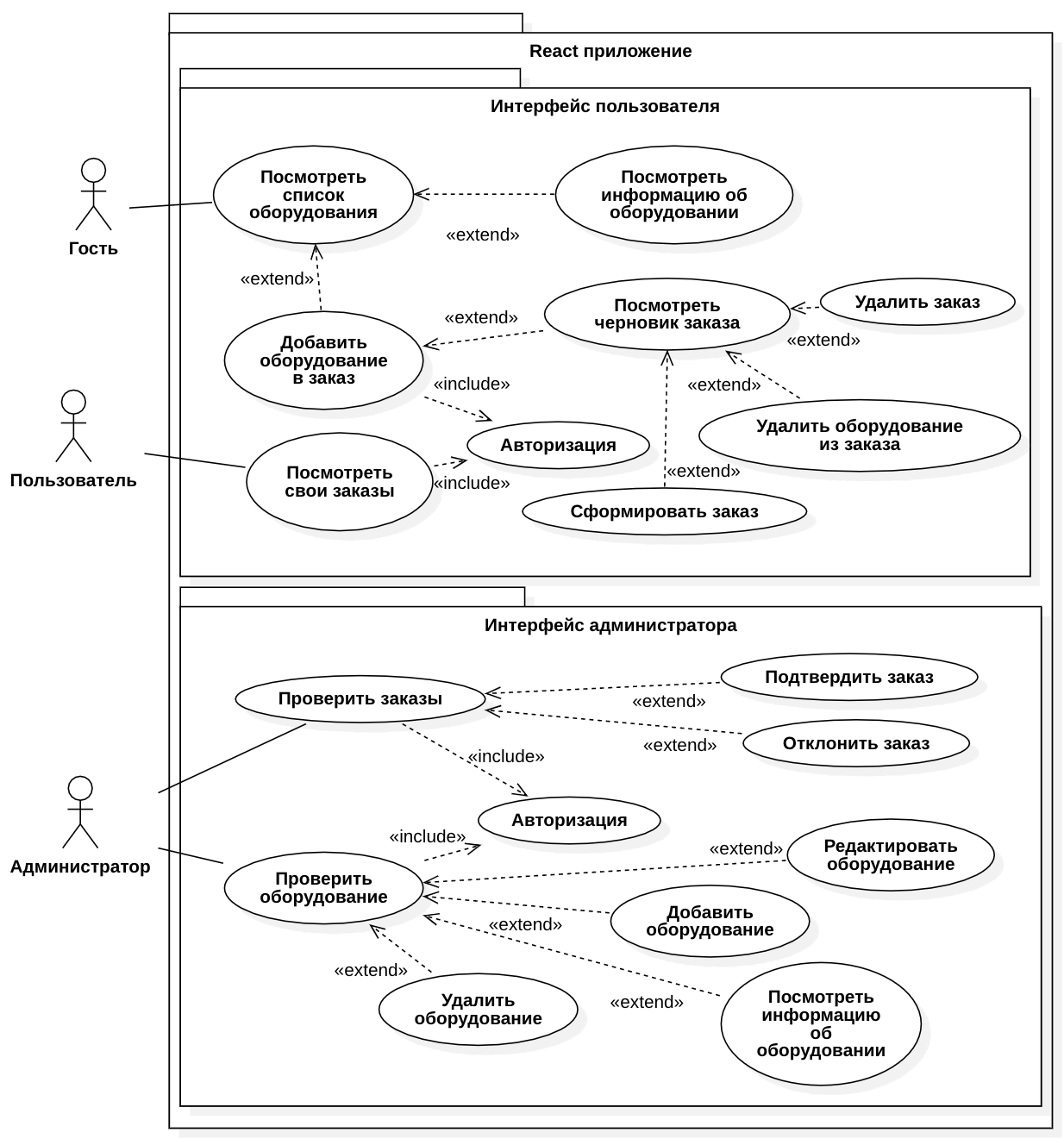


Рисунок 1 – Диаграмма прецедентов

Гости могут ознакомиться с полным списком доступного оборудования. Гости, прошедшие этап регистрации, являются пользователями. Пользователи могут добавлять оборудование в заказ, формировать заказ из черновика, а также просматривать список своих заказов.

Пользователь выбирает оборудование, затем на основе выбранного списка товаров формирует заказ. После чего администратор может посмотреть его, согласовать или отклонить. В случае некорректного составления заказа пользователь имеет возможность удалить заказ. Возможные состояния статуса заказа отображены на диаграмме состояний (рисунок 2).

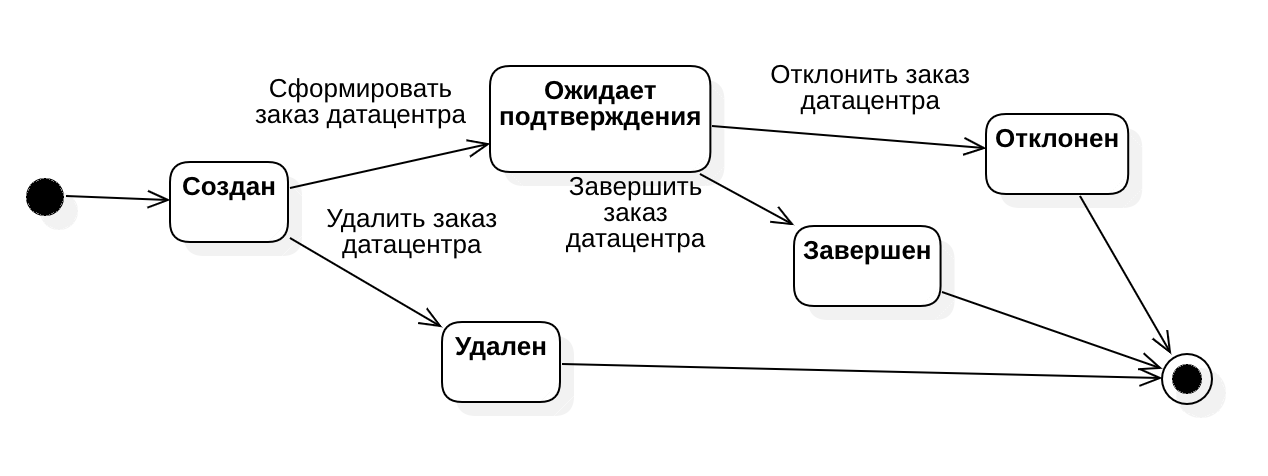


Рисунок 2 – Диаграмма состояний счета

Заказы проходят через администратора, который тщательно их обрабатывает. В зависимости от результата, заказ либо одобряют, либо отклоняют. Администратор имеет полный контроль над оборудованием: он может просматривать все комплектующие, редактировать их характеристики, создавать новые позиции и удалять устаревшие. Также он может формировать список всех комплектующих в виде таблицы для удобства учета и анализа.

Процесс оформления заказа подробно описан на диаграмме деятельности (рисунок 3). Эта диаграмма охватывает все этапы, через которые проходит заказ, обеспечивая полное представление о последовательности действий.

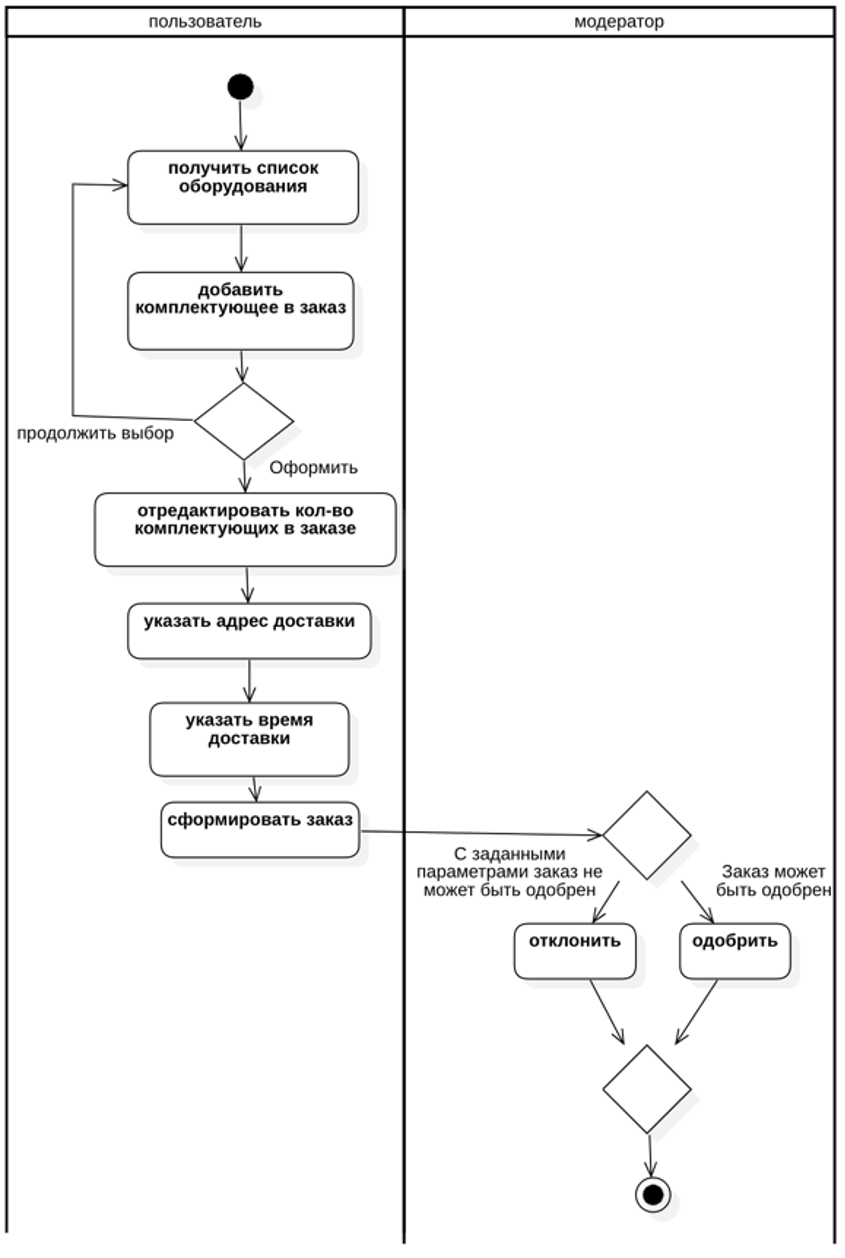


Рисунок 3 – Диаграмма деятельности

В начале взаимодействия с системой оформления заказа пользователь запрашивает список доступных комплектующих. При необходимости пользователь может добавить конкретное оборудование в текущий черновой заказ. После чего может продолжить выбор комплектующих из каталога или перейти к оформлению заказа. В случае, если пользователь выбрал оформление заказа, ему необходимо указать время и адрес доставки. По нажатии кнопки «Подтвердить заказ» заказ формируется и становится доступен в интерфейсе администратора.

Администратор имеет возможность просматривать заказы пользователей, а также принимать решение об одобрении заказа, если он удовлетворяет условиям доставки, или отклонении счета, если он не удовлетворяет требованиям.

# АРХИТЕКТУРА

Архитектура системы отображена на диаграмме развертывания (рисунок 4).

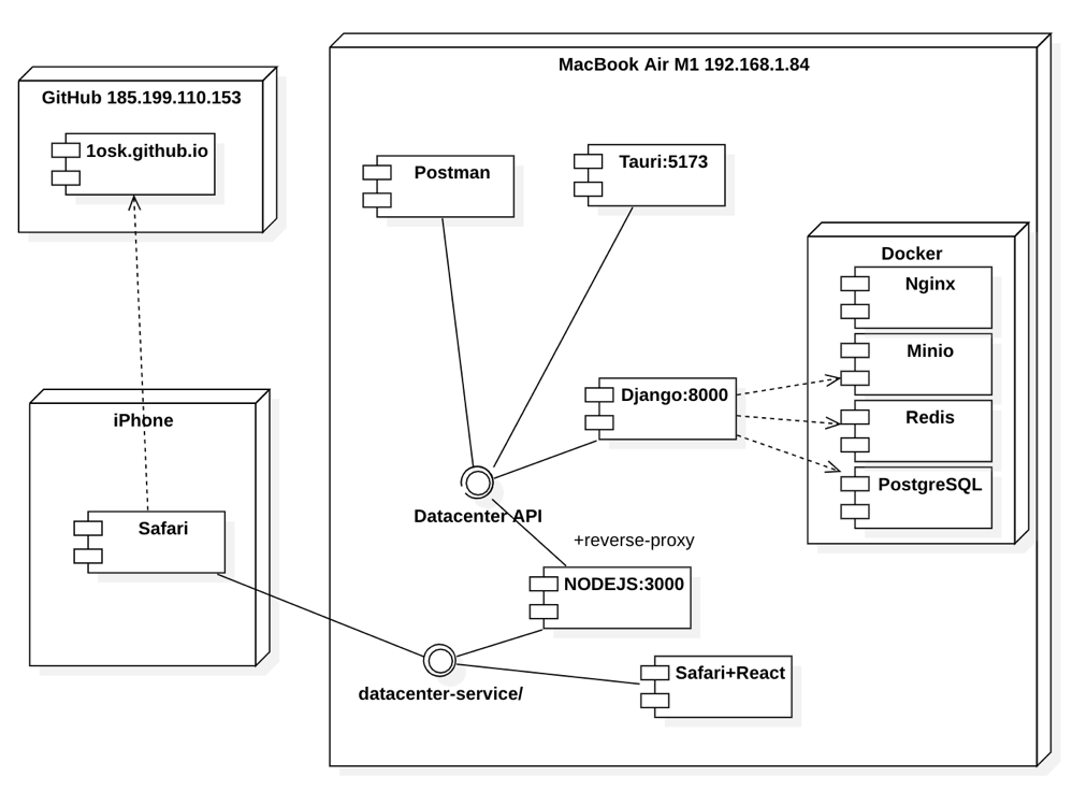


Рисунок 4 – Диаграмма развертывания

Веб-сервис, реализованный на фреймворке Django [3] с использованием DRF [4] связан с серверами Minio [5], Redis [6] и СУБД PostgreSQL [7], которые находятся в Docker. В Redis хранятся активные сессии пользователей, при выходе пользователя из системы – удаляются.

Язык программирования Python [8] был выбран в силу его простоты, а также популярности. Использование Django обусловлено тем, что этот фреймворк имеет достаточно широкие возможности и является стандартом индустрии. Веб-сервис на Django является общим для веб-сервера и десктопного приложения Tauri [9].

Данные хранятся в СУБД PostgreSQL. Она была выбрана, поскольку, наравне с Python и Django она является стандартом современной индустрии разработки.

Структура данных отражена на ER-диаграмме (рисунок 5). Модель оборудования представляет собой набор полей, отражающих свойства оборудования. Данные об Оборудовании хранятся в таблице datacenter\_app\_datacenterservice. Для хранения в одном заказе нескольких комплектующих используется промежуточная таблица связи datacenter\_app\_datacenterorderservice, которая реализует связь М-М. Таблица datacenter\_app\_datacenterorder представляет собой список заказов. Данные о пользователях система и администраторах – хранятся в таблице auth\_user.

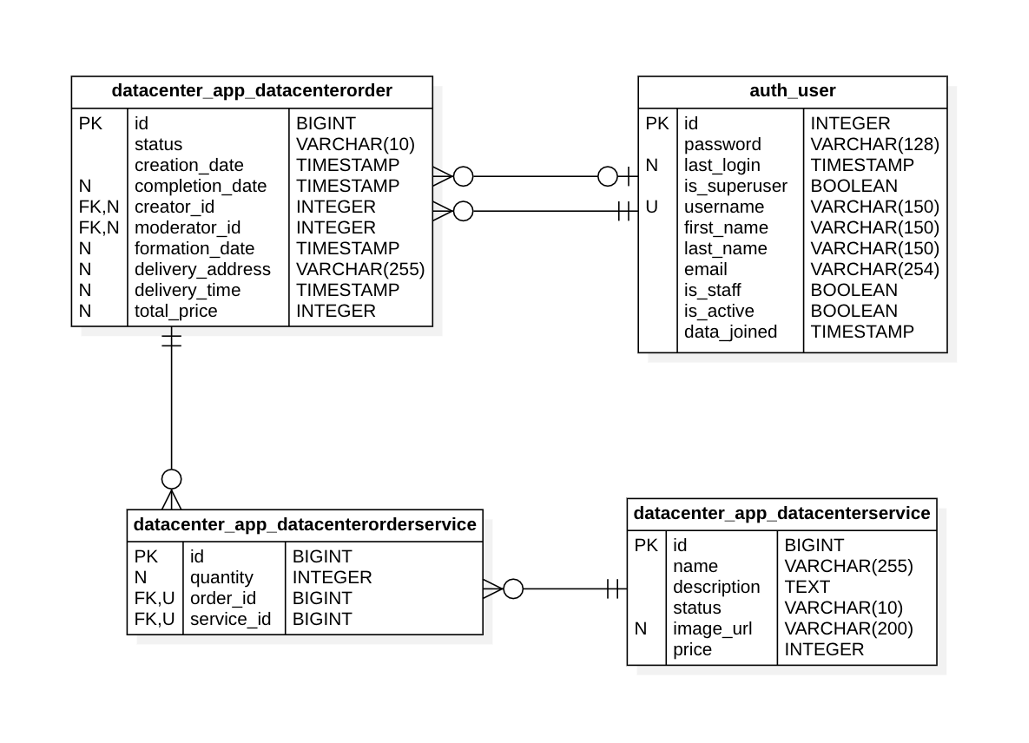


Рисунок 5 – ER-диаграмма

Веб-сервер реализован с использованием языка TypeScript и фреймворка React [10]. Устройство бэкенда приложения разработанной системы приведено на диаграмме классов бэкенда (рисунок 6). Пользователи взаимодействуют с доменами. Домены связаны с моделями. Модели имеют связи с таблицами в базе данных.

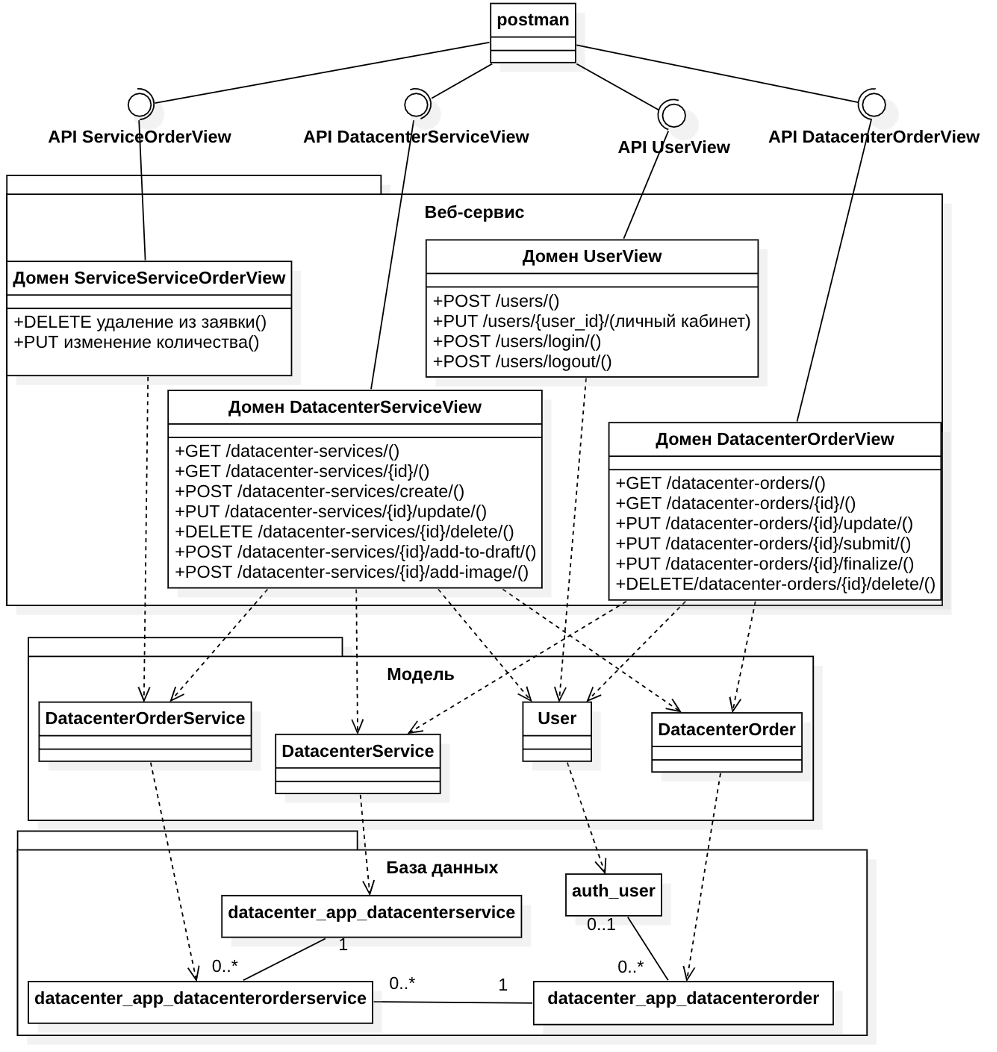


Рисунок 6 – Диаграмма классов бэкенда

Связь фронтенда и бэкенда отражена на диаграмме классов фронтенда (рисунок 7). Каждая страница связана с API, которое используется для взаимодействия с данными на соответствующей странице.

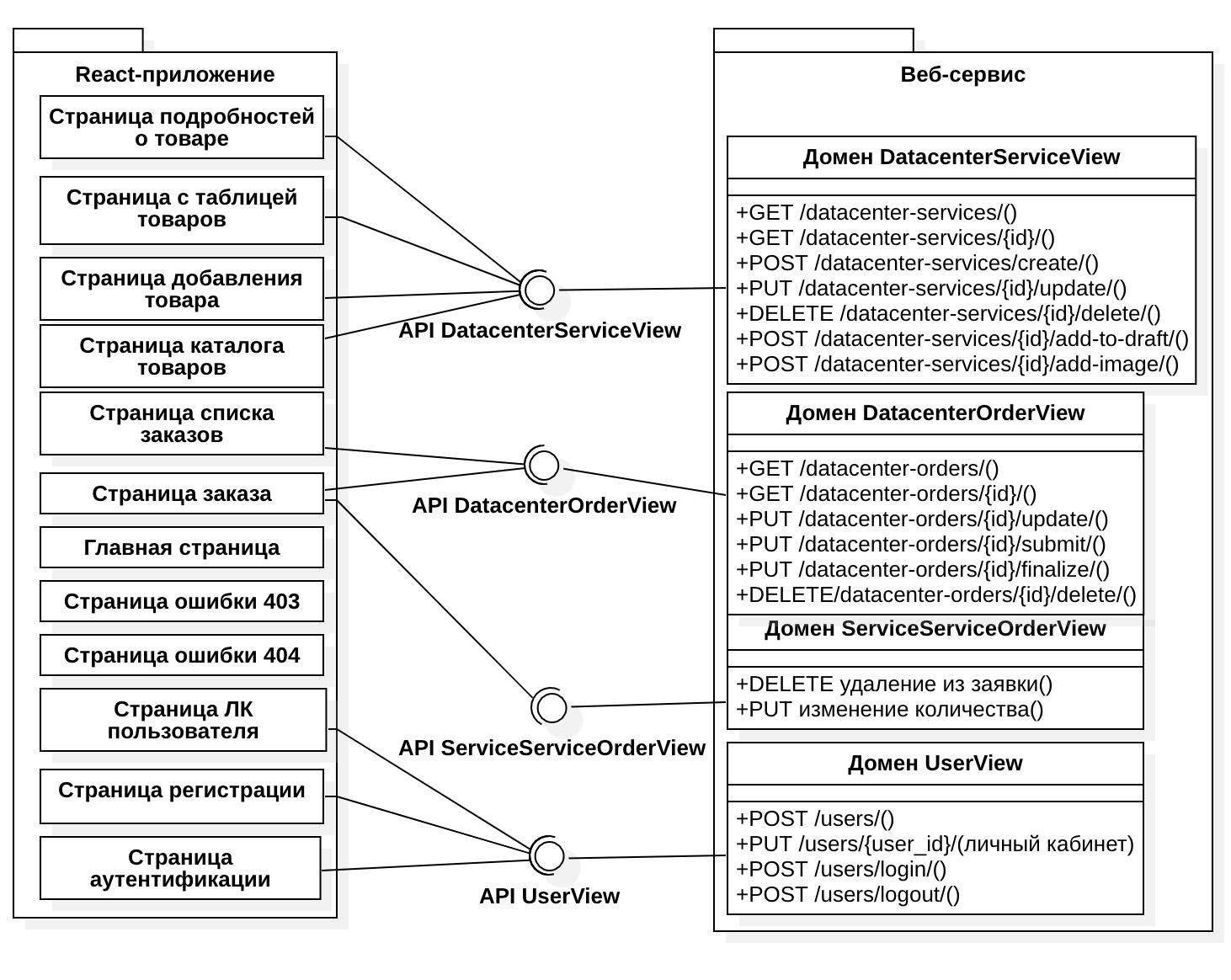


Рисунок 7 – Диаграмма классов фронтенда

# АЛГОРИТМЫ

Алгоритм работы разработанной системы отображен на диаграмме последовательности (рисунок 8). В основе системы лежит веб-сервис, реализующий внутри себя всю бизнес-логику. Он предоставляет доступ к методам из следующих доменов: услуги, заказы, услуги в заказе, пользователи. Методы следуют правилам REST API.

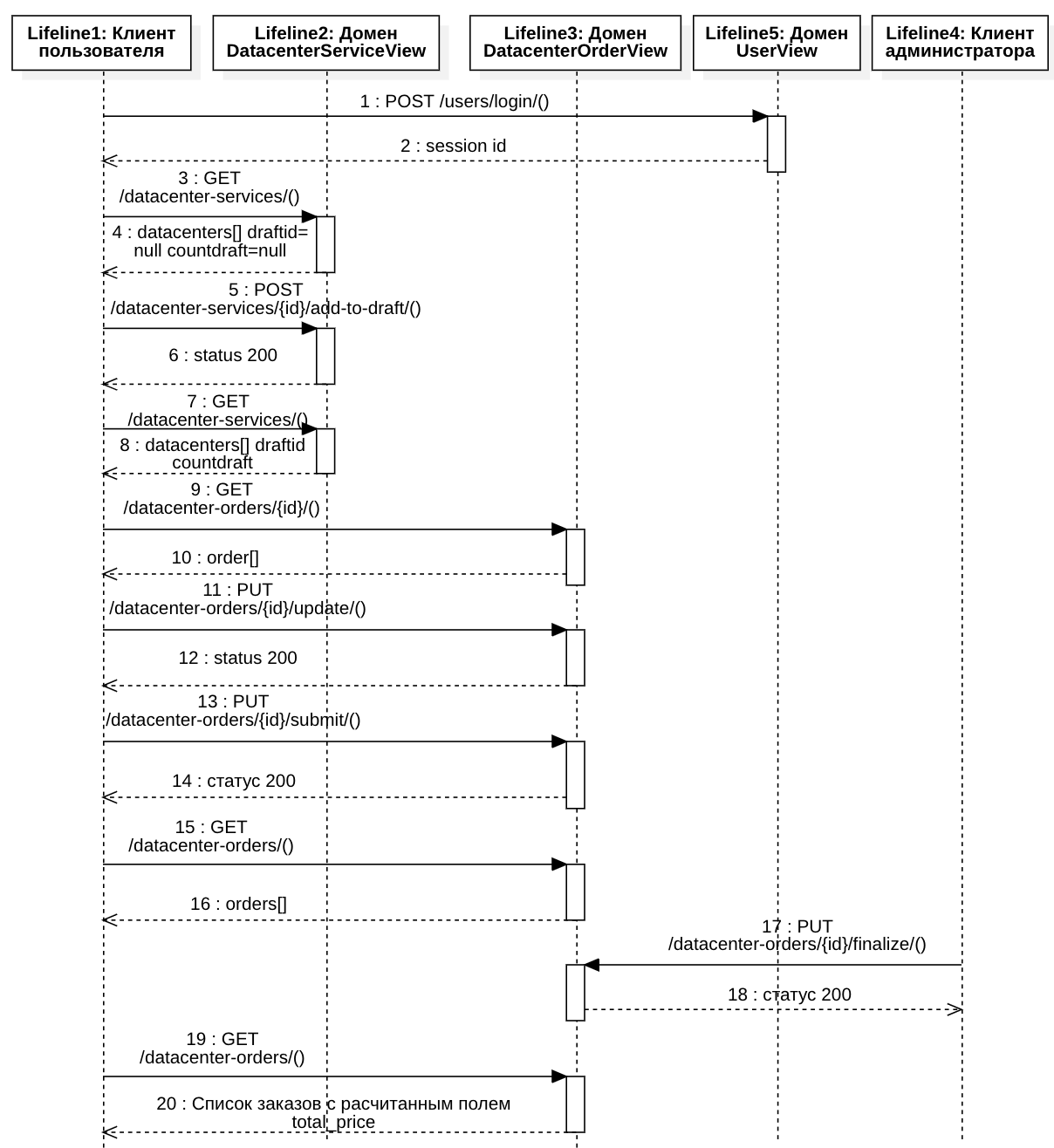


Рисунок 8 – Диаграмма последовательностей

В начале бизнес-процесса происходит аутентификация пользователя. При помощи графического интерфейса пользователь отправляет запрос, передавая в нем логин и пароль. Если аккаунт с указанными данными существует в базе, на клиент возвращается информация о пользователе и устанавливаются куки с полем session\_id (идентификатор текущей сессии) в ответном запросе. Если же аккаунта с введённым логином не существует, или пароль введен неверно, на клиент возвращается ошибка. В таком случае пользователю надо пройти регистрацию, либо ввести верный пароль. На этом же этапе происходит проверка: является пользователь пользователем или администратором. Графический интерфейс пользователя запрашивает у веб-сервиса список комплектующих, которые возвращаются в JSON формате. Пользователь выбирает комплектующее, которое хочет добавить в заказ, и, нажимая на кнопку «добавить» в графическом интерфейсе, отправляет запрос на добавление комплектующего в свой черновой заказ. Этот процесс может продолжаться несколько раз.

Когда пользователь определится со списком комплектующи, а также укажет время и адрес доставки, он нажимает кнопку «Подтвердить Заказ» в графическом интерфейсе. После чего приложение отправляет на веб-сервис запрос на формирование заказа. Пользователь может отслеживать статус сформированных заказов на соответствующей странице в графическом интерфейсе.

Процесс рассмотрения заказов администратором так же происходит при помощи графического интерфейса. Администратор может просматривать список всех созданных пользователями заказов; согласовывать или отклонять их при помощи соответствующих кнопок. Доступна фильтрация по логинам пользователей, а также статусам и дате формирования заказов. Администратор имеет возможность редактировать список комплектующих, а также изменять данные о каждом оборудовании. Для каждой из этих возможностей существует соответствующий метод, отправляемый на веб-сервис.

# ОПИСАНИЕ ИНТЕРФЕЙСА

Главное меню приложения включает пункты, которые доступны в зависимости от роли пользователя (рисунки 9, 10, 11).



Рисунок 9 – Меню приложения (для гостя)

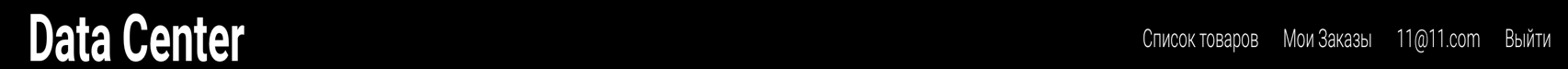


Рисунок 10 – Меню приложения (для пользователя)

На странице с формой регистрации (рисунок 12) отображается форма, при помощи которой гость может создать новый аккаунт.

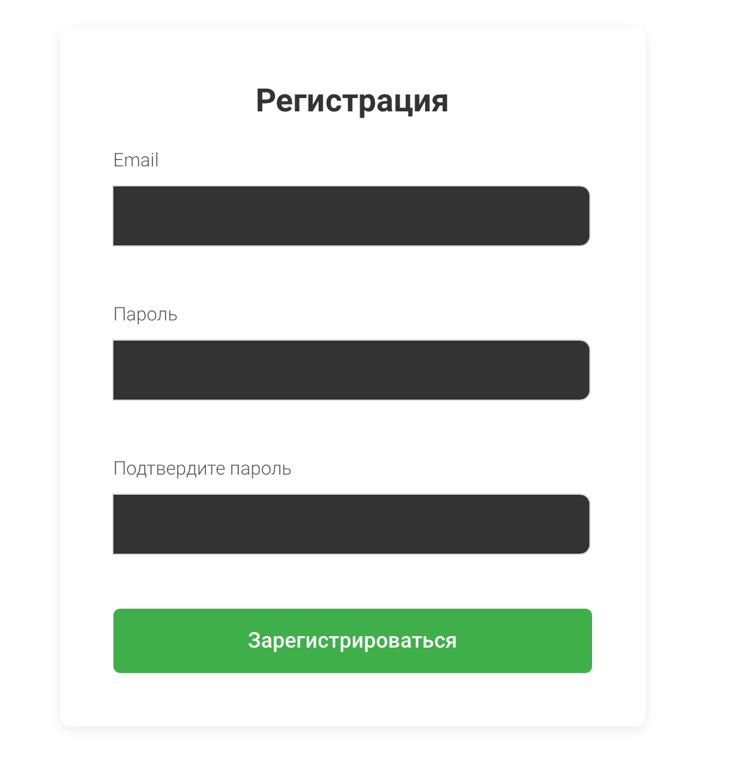


Рисунок 12 – Страница регистрации

На странице с формой входа (рисунок 13) отображается форма, через которую пользователь может войти в свой аккаунт. При успешном вводе данных аккаунта на клиент приходят куки с идентификатором текущей сессии.

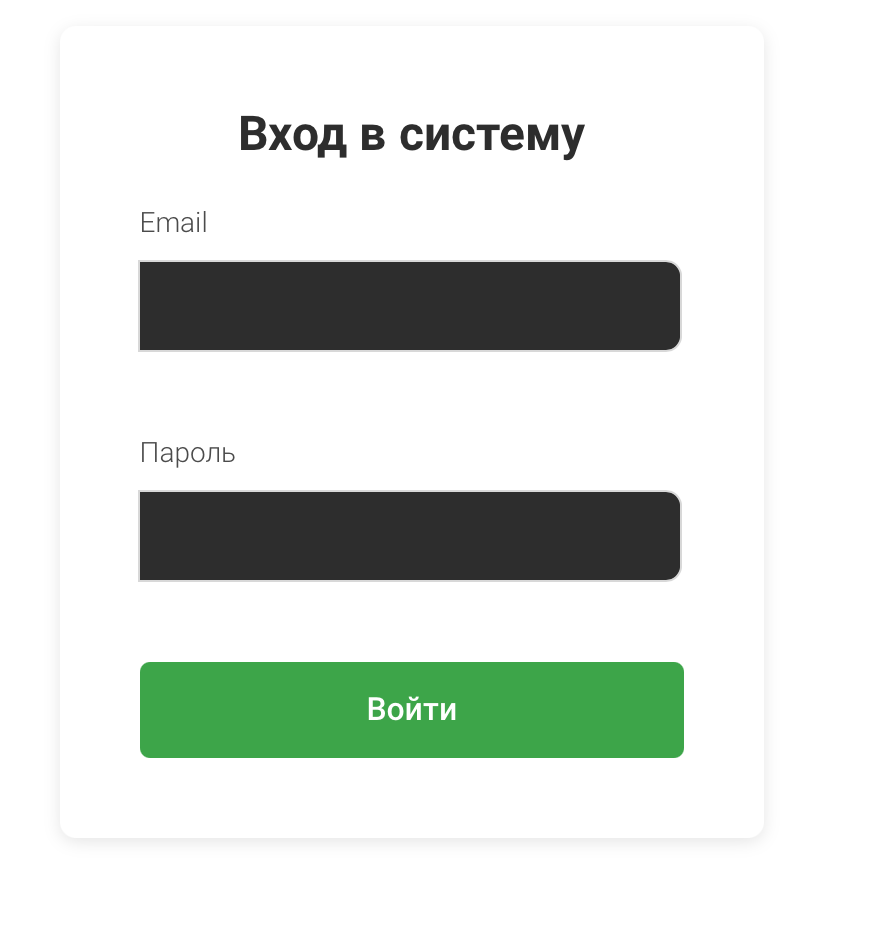


Рисунок 13 – Страница входа

После входа в аккаунт пользователь попадает на главную страницу (рисунок 14). На этой странице располагается приветствие и карусель из фотографий оборудования.



Рисунок 14 – Главная страница

Страница каталога товаров (рисунок 15) содержит список комплектующих в виде карточек. При нажатии на кнопку «Подробнее о комплектующем» на любой карточке, открывается страница с подробным описанием выбранного товара. Также с помощью нажатия на кнопку «Добавить в заказ» пользователь может добавлять комплектующие в черновик заказа. В верху страницы находится слайдер для фильтрации комплектующих по цене.

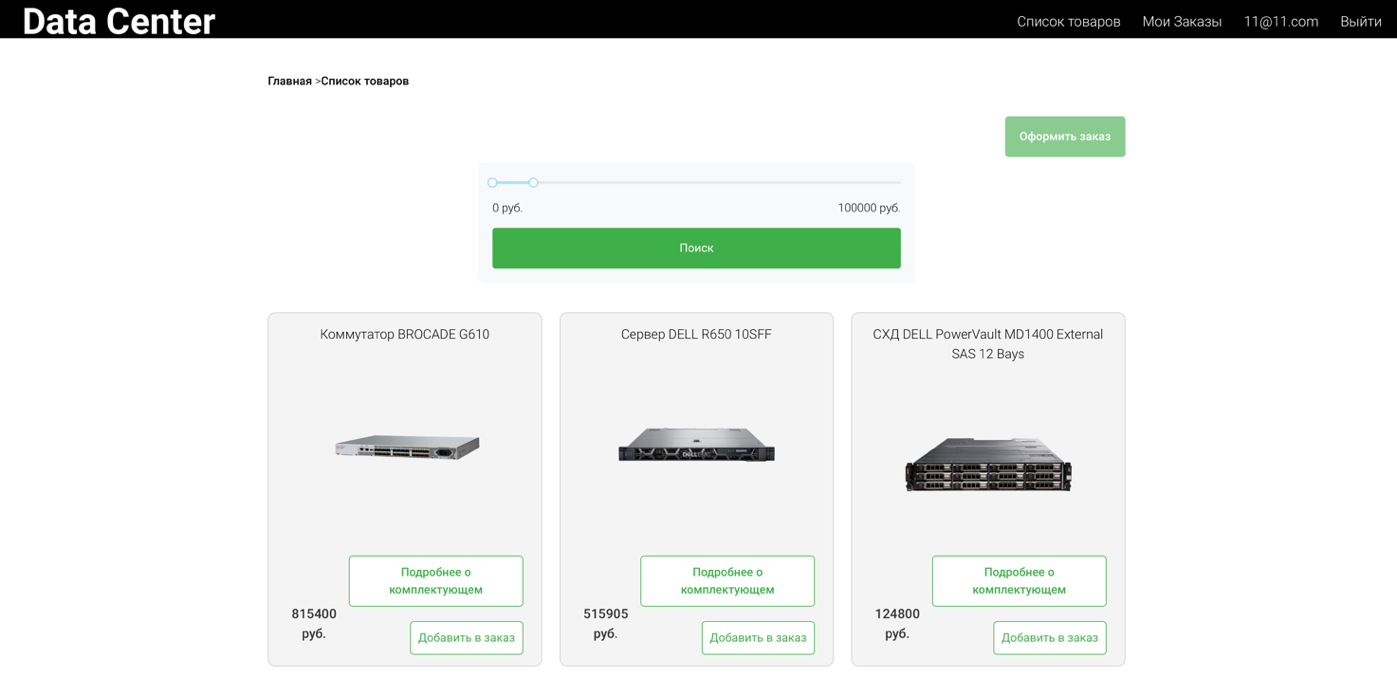


Рисунок 15 – Страница каталога товаров

На странице с подробным описанием товара выводится основная информация об комплектующем (рисунок 16).



Рисунок 16 – Страница с описанием комплектующего

На странице заказа (рисунок 17) отображается текущий заказ пользователя. У пользователя есть возможность удалить товары из счета, изменить количество, указать время и адрес доставки, а также отправить заказ на проверку. Редактирование одобренных заказов недоступно.

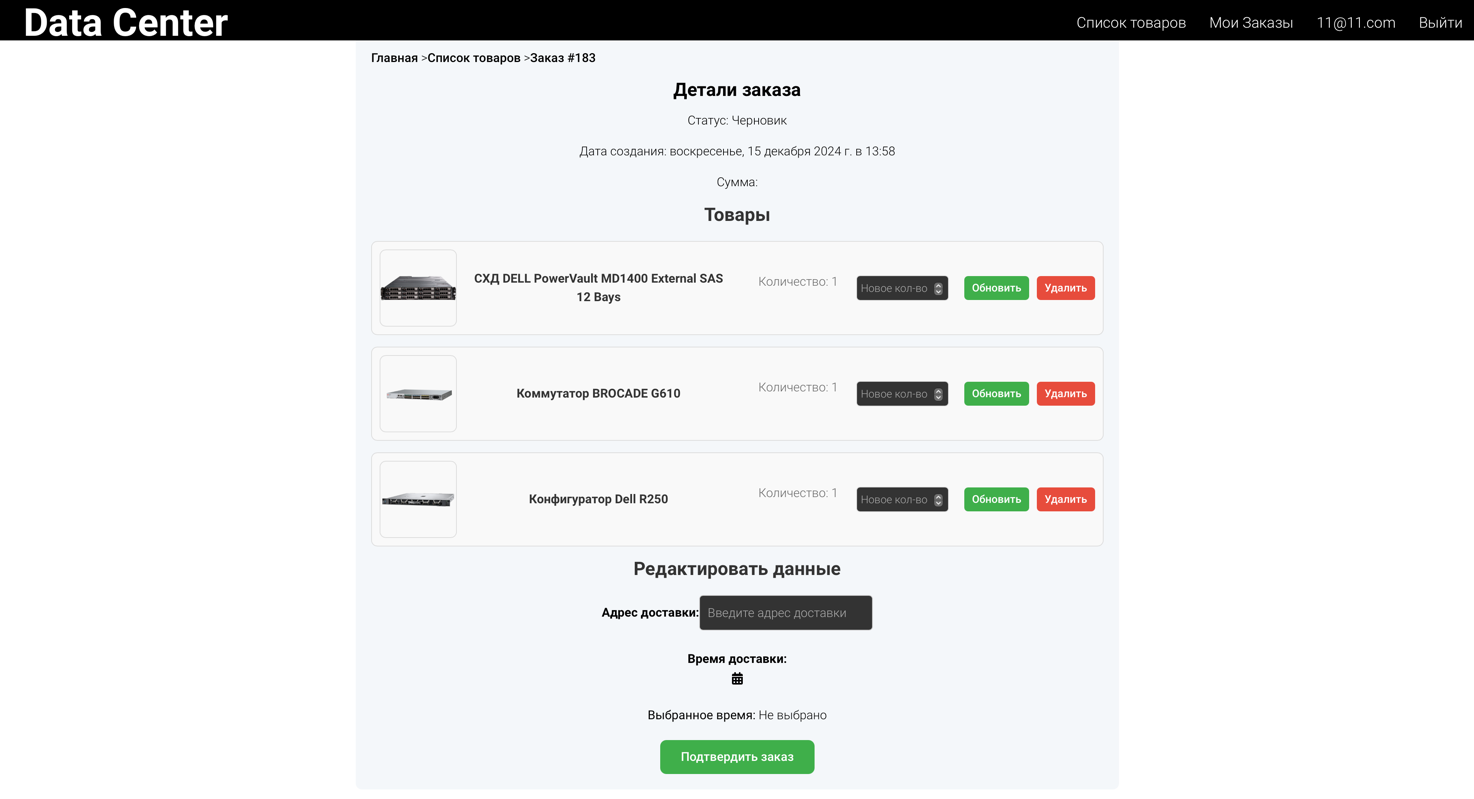


Рисунок 17 – Страница заказа

На странице списка заказов (рисунок 18) пользователи могут просматривать оформленные ими заказы в виде таблицы. На этой странице можно посмотреть подробную информацию о заказе.

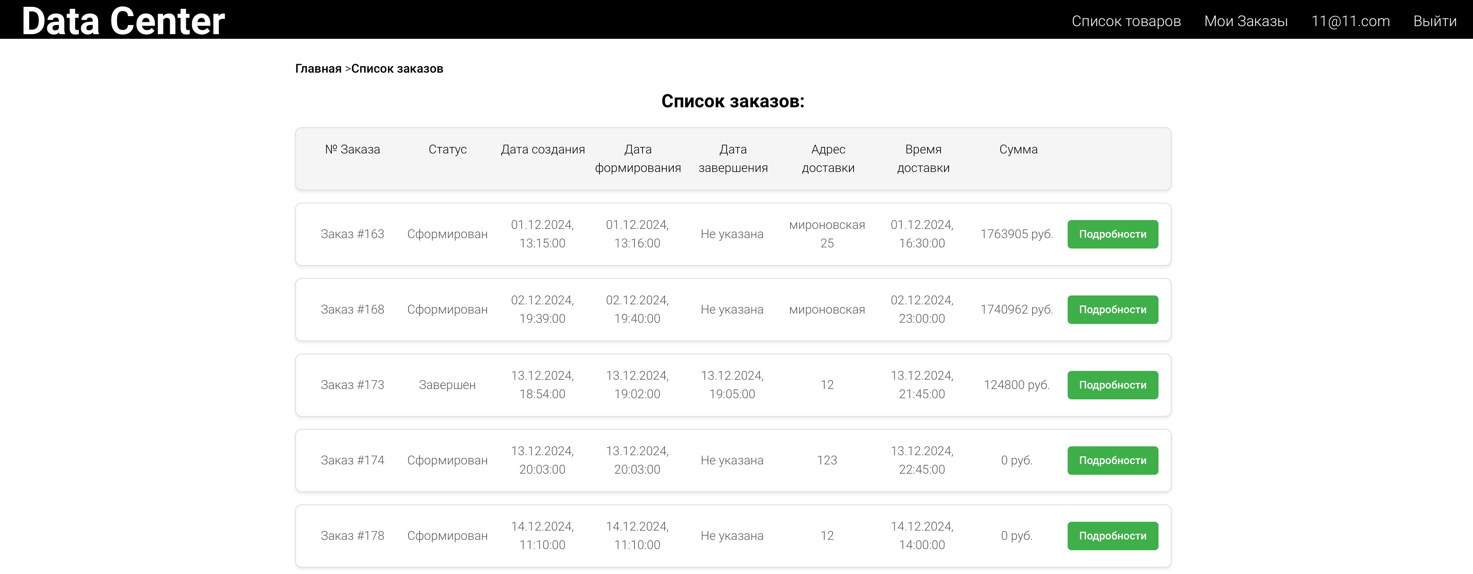


Рисунок 18 – Страница списка заказов

Потребитель может изменить свои данные (пароль) на странице редактирования данных пользователя (рисунок 19). Она открывается при нажатии на логин пользователя в меню.

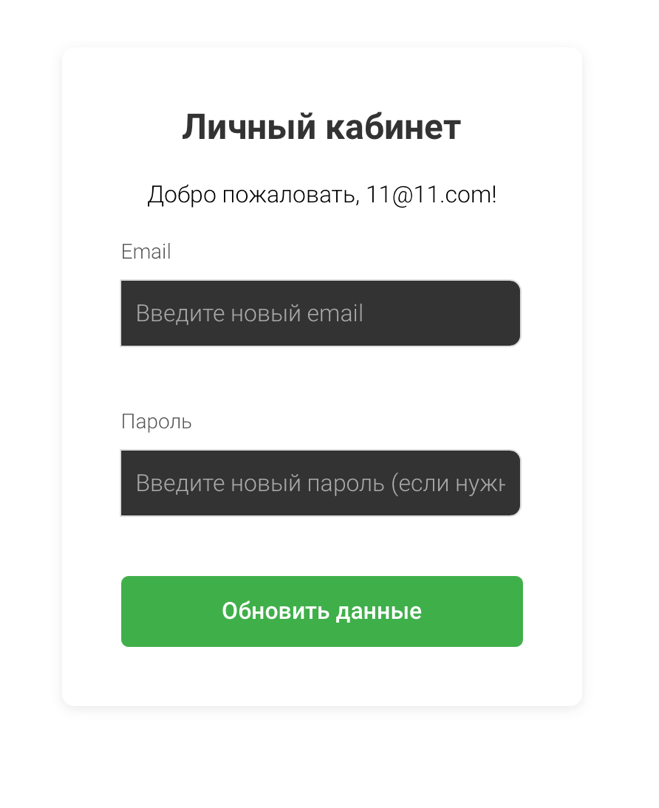


Рисунок 19 – Страница редактирования информации о пользователе

Администратор может просматривать информацию обо всех товарах, занесённых в систему на странице списка товаров. Исполнитель имеет возможность добавить новый товар, отредактировать или удалить существующий.

На странице списка всех заказов пользователей, администратор может согласовывать или отклонять заказы пользователей путём нажатия соответствующих кнопок в интерфейсе. Администратор может просматривать подробную информацию о заказе путём нажатия на его. Доступны поля фильтрации по логину пользователя, статусу и датам формирования заказа.

С помощью формы редактирования и создания товаров администратор может добавлять новые товары или изменять существующие.

В случае, если пользователь запрашивает несуществующую станицу, он перенаправляется на страницу ошибки 404.

В случае, если пользователь пытается получить доступ к страницам, для просмотра которых ему не хватает прав, он перенаправляется на страницу ошибки 403.

Перейдем к интерфейсу приложения, развернутого на Github Pages.

На навигационной панели содержится ссылка на страницу каталога комплектующих (рисунок 20).



Рисунок 20 – Главная страница Github Pages

На странице каталога товаров изображены фиксированные 6 товаров с моковыми изображениями (рисунок 21). Реализована возможность фильтрации по цене, возврата на главную страницу с помощью хлебной крошки, перехода на страницу информации о товаре

На странице информации о товаре отображена общая информация о характеристиках и цене (рисунок 22). Так же реализована возможность возврата с помощью хлебной крошки.

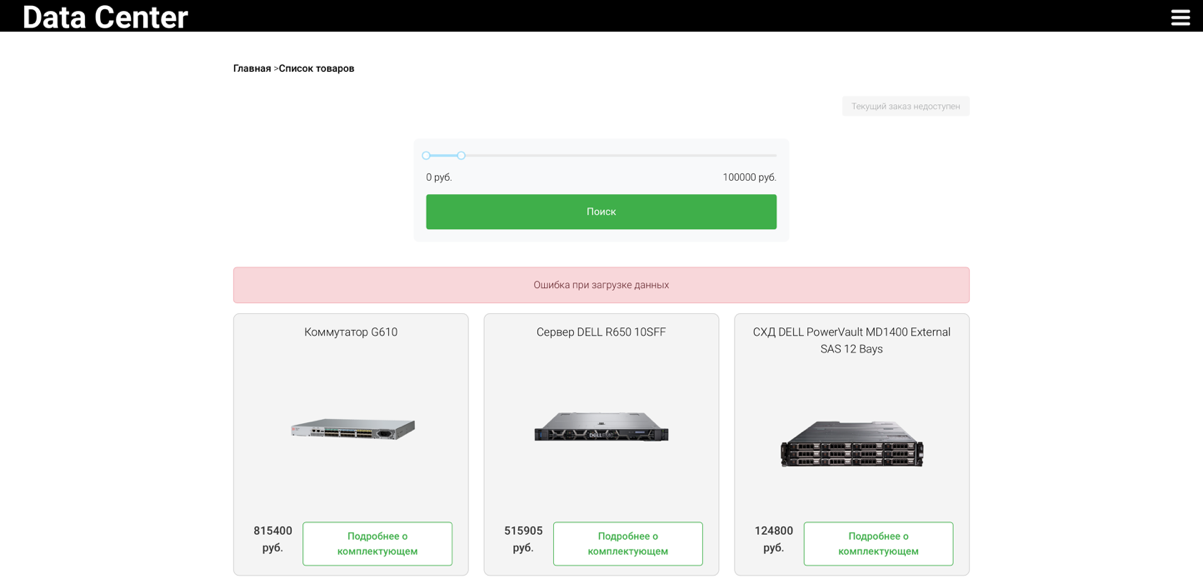


Рисунок 21 – Страница списка товаров Github Pages



Рисунок 22 – Страница с информацией о товаре Github Pages

Перейдем к интерфейсу десктопного приложения Tauri.

Все страницы: главная (рисунок 23), страница списка товаров (рисунок 24), страница с информацией об услуге (рисунок 25) – имеют интерфейс, аналогичный приложению на Github Pages.



Рисунок 23 – Главная страница Tauri

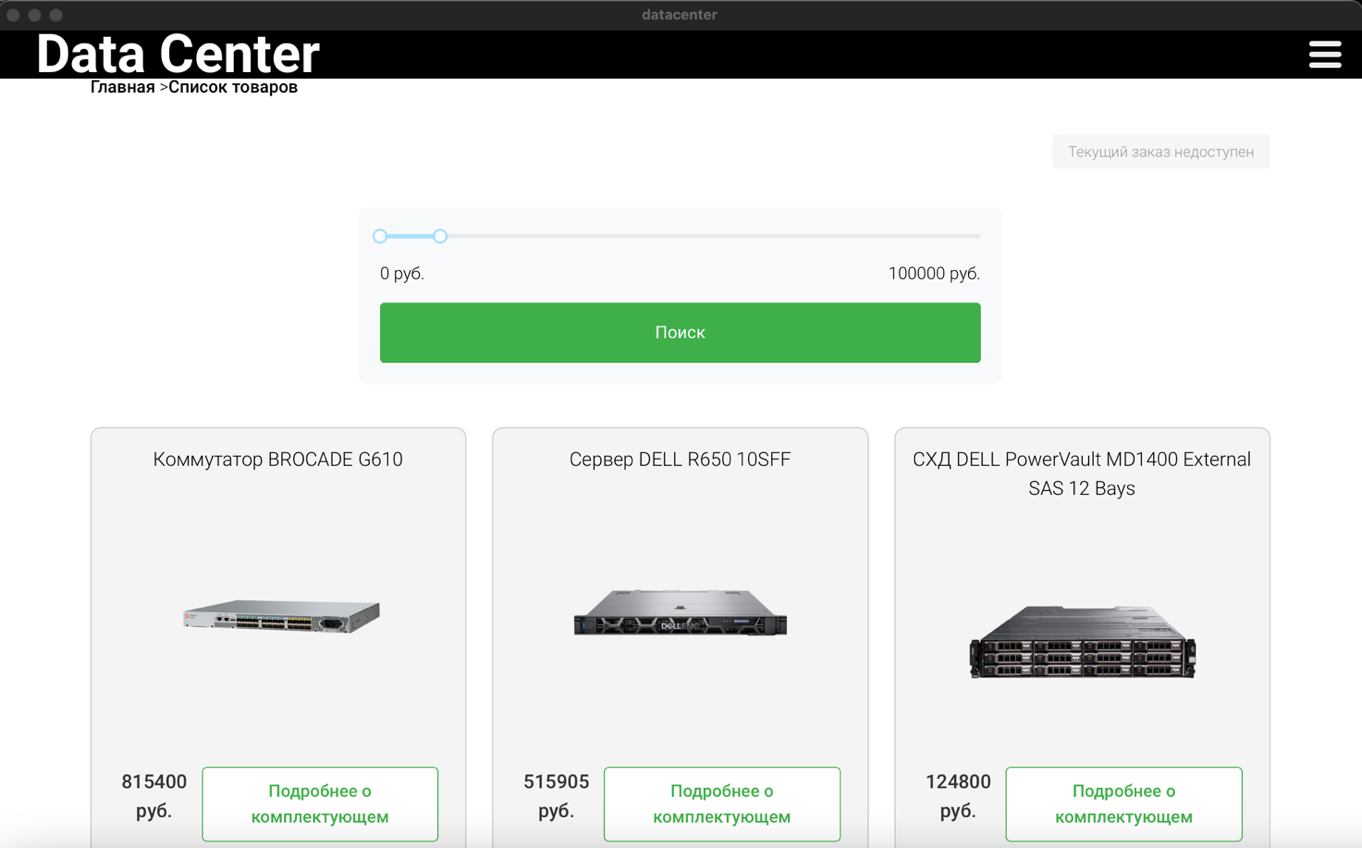


Рисунок 24 – Страница списка товаров Tauri

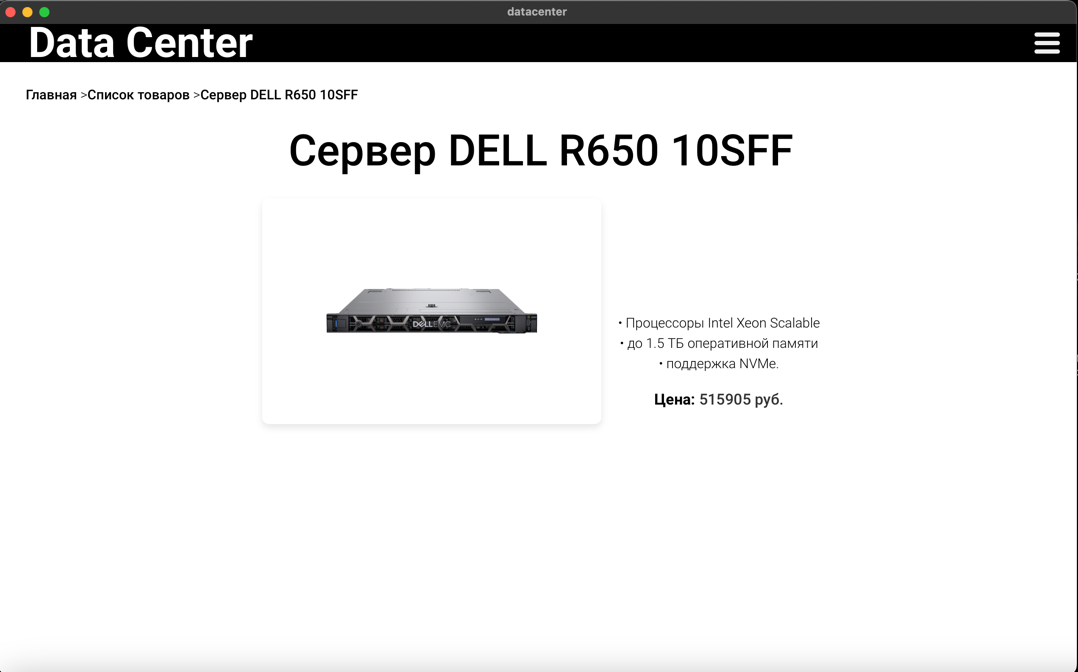


Рисунок 25 – Страница с информацией о товаре Tauri

При запущенном веб-сервере функционал приложения Tauri будет абсолютно аналогичен интерфейсу основного веб-приложения.

# ЗАКЛЮЧЕНИЕ

В ходе работы были достигнуты следующие результаты:

* 1. Создан MVP и базовый дизайн приложения на основе datahouse.ru;
  2. Создана база данных для хранения информации об услугах и заказах;
  3. Создан веб-сервис на Django (DRF);
  4. Реализована авторизация и хранение сессий в Redis;
  5. Разработан базовый SPA на React для гостя;
  6. Внедрена адаптивность, менеджер состояний Redux Toolkit, PWА, разработано Tauri приложение;
  7. Завершена разработка интерфейса потребителя в React;
  8. Реализован интерфейс исполнителя;
  9. Разработано десктопное приложение Tauri;
  10. Приложение развернуто при помощи сервиса GitHub Pages и доступно по ссылке: https://1osk.github.io/Frontend/;
  11. Подготовлен набор документации, включающий РПЗ, ТЗ и набор диаграмм;
  12. Оформлен git-репозиторий на сервисе GitHub, содержащий исходный код проекта: https://github.com/1OSK/Frontend

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Википедия [Электронный ресурс] // URL: https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D1%82%D0%B0-%D1%86%D0%B5%D0%BD%D1%82%D1%80 (дата обращения: 10.09.2024).
2. Habr [Электронный ресурс] // URL: https://habr.com/ru/companies/mws/articles/276103/ (дата обращения: 21.12.2024).
3. Документация по Django [Электронный ресурс] // djangoproject. URL: https://www.djangoproject.com (дата обращения: 20.10.2024)
4. Документация по DRF [Электронный ресурс] // Django-rest-framework. URL: https://www.django-rest-framework.org/topics/documenting-your-api/ (дата обращения: 20.10.2024)
5. Документация по Redis [Электронный ресурс] // Netlify. URL: https://master--redis-doc.netlify.app/docs/ (дата обращения: 10.10.2024)
6. Документация Minio [Электронный ресурс] // Min. URL: https://min.io/docs/minio/kubernetes/upstream/index.html (дата обращения: 09.09.2024)
7. Документация PostgreSQL [Электронный ресурс] // Postgresql. URL: https://www.postgresql.org/docs/ (дата обращения: 15.09.2024)
8. Документация по Python [Электронный ресурс] // Python. URL: https://docs.python.org/3/index.html/ (дата обращения: 20.10.2024)
9. Документация по Tauri [Электронный ресурс] // Tauri. URL: https://v2.tauri.app/develop/ (дата обращения: 20.11.2024)
10. Документация по React [Электронный ресурс] // React. URL: https://react.dev/learn (дата обращения: 01.11.2024)

# ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

|  |  |  |
| --- | --- | --- |
| Утверждаю  Заведующий кафедрой ИУ-5 |  | Согласовано  Научный руководитель |
| \_\_\_\_\_\_\_\_\_\_\_В.И.Терехов  "\_\_"\_\_\_\_\_\_\_\_\_\_2024 г. |  | \_\_\_\_\_\_\_\_\_\_А.И. Канев  "\_\_"\_\_\_\_\_\_\_\_\_\_2024 г. |

**Создание веб-приложения для Дата Центра**

Техническое задание

(вид документа)

писчая бумага

(вид носителя)

20

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Ильин Константин Юрьевич |
| "\_\_"\_\_\_\_\_\_\_\_\_\_\_2024 г. |  |

Москва - 2024

## Введение

Цель системы, включающая в себя веб-сервис, веб-приложение, нативное приложение, которая дает возможность пользователям сформировать Дата-центр и редактировать его.

## Назначение разработки

Данная система предназначена для хранения и управления данными о заказах и товаров дата-центра. Клиенты могут создавать новые заказы на товары или оборудование, отслеживать статус своих заявок. Операторы выполняют модерацию заказов, подтверждая или отклоняя их в зависимости от их корректности. Администраторы могут редактировать и добавлять новые услуги и товары в систему, а также просматривать и управлять всеми заявками клиентов, обеспечивая их актуальность и полноту.

## Стадии и этапы разработки

* 1. Создание MVP и базового дизайна на основе datahouse.ru
  2. Реализовать хранение данных в БД (Postgres)
  3. Создать веб-сервис на бэкенде (DRF)
  4. Реализовать авторизацию и хранение сессий в Redis
  5. Разработать SPA на React для гостя
  6. Внедрить адаптивность, разработка Tauri приложения
  7. Интерфейс пользователя на React с менеджером состояний Redux Toolkit
  8. Добавить в приложение React интерфейс администратора
  9. Реализовать нативное приложение
  10. Развернуть веб-приложение в GitHub Pages
  11. Подготовка всей документации (РПЗ, ТЗ и набор диаграмм)
  12. Подготовка репозитория на GitHub

## Требования к функциональным характеристикам

* 1. Методы HTTP
     1. GET Получение всех товаров
     2. POST Добавление товара
     3. GET Получить один товар
     4. PUT Изменить товар
     5. DELETE Удалить товар
     6. POST Добавить товар в черновик заказа
     7. POST Изменение/добавление картинки товара
     8. GET Получить все заказы с фильтрацией
     9. GET Получить один заказ
     10. PUT Изменить поля заказа
     11. DELETE Удалить заказ
     12. PUT Сохранить заказ
     13. PUT Модерировать заказ
     14. DELETE Удалить товар из заказа
     15. PUT Поменять поле количества
     16. POST Регистрировать пользователя
     17. PUT Личный кабинет
     18. POST Аутентификация
     19. POST Деавторизация
  2. Меню:
     1. Главная - перенаправляет на страницу 4.5
     2. Список товаров – перенаправляет на страницу 4.6
     3. Список заказов – перенаправляет на страницу 4.9
     4. Редактирование товаров – перенаправляет на страницу 4.11, доступно только администратору
     5. Зарегистрироваться – перенаправляет на страницу 4.3 появляется только для гостей
     6. Личный кабинет – перенаправляет на страницу 4.10, только аутентифицированным пользователям
     7. Войти – перенаправляет на страницу 4.4, только для гостей
     8. Выйти – доступно только аутентифицированным пользователям (вызывается метод 4.1.19)
  3. Регистрация
     1. Доступно только гостям
     2. Отображает форму регистрации
     3. Поле имени пользователя
     4. Поле пароля
     5. Поле повторить пароль
     6. Действия
     7. Регистрация пользователя
     8. Вернуться к аутентификации – перенаправляет на страницу 4.4
  4. Аутентификация
     1. Доступно только гостям
     2. Отображает форму аутентификации
     3. Поле имени пользователя
     4. Поле пароля
     5. Действия
     6. Войти
     7. Регистрация – перенаправляет на страницу 4.3
  5. Личный кабинет
     1. Доступно аутентифицированному пользователю
     2. Действия
     3. Изменить данные пользователя
  6. Главная
     1. Доступна всем
     2. Отображается статическая информация
     3. Назначение сервиса
     4. Контакты для связи
  7. Список товаров
     1. Доступна всем
     2. Отображаются элементы карточек с товарами, вызывается метод 4.1.1
     3. Название товара
     4. Краткое описание
     5. Действия
     6. Поиск – перенаправляет на страницу 4.6, с фильтрующем параметром
     7. Подробнее – перенаправляет на страницу 4.7
     8. Добавить в заказ – добавляет товар в черновик заказа, только аутентифицированные пользователи.
     9. Кнопка корзины – перенаправляет на страницу 4.9, только аутентифицированные пользователи.
  8. Один товар
     1. Доступна всем
     2. Отображается подробная информация выбранного товара
  9. Один заказ
     1. Доступно только аутентифицированным пользователям
     2. Отображает текущий черновик заказа пользователя
     3. Список выбранных товаров
     4. Поле даты начала формирования товаров в заказ
     5. Поле даты конца формирования товаров в заказ
     6. Поле названия заказа
     7. Действия, доступны только в случае, если статус «черновик»
     8. Убрать товар – удалят товар из заказа
     9. Сохранить – сохраняет текущий черновик заказа
     10. Очистить – удаляет черновик заказа
     11. Заполнить – вносит данные заказа (дата и время доставки)
  10. Список заказов
      1. Доступно аутентифицированному пользователю
      2. Отображается список заказов
      3. Только заказы, созданные данным пользователем, если он не модератор
      4. Все сохраненные заказы в противном случае
      5. Действия
      6. Фильтрация – фильтрует заказы по дате создания или статусу, вызывается
      7. Сформировать – сформирует заказ, доступно только модератору
      8. Отклонить – отклоняет заказ, доступно только модератору
      9. Посмотреть подробную информацию о заказе – перенаправляет на страницу 4.
  11. Список товаров таблицей
      1. Доступно администратору
      2. Отображаются все существующие в системе товары
      3. Действия
      4. Удалить – удаляет товар
      5. Редактирование/создание – переход на страницу 4.12
  12. Редактирование/создание товара
      1. Доступно администратору
      2. Отображается информация об изменяемом/добавляемом товаре
      3. Название
      4. Детальное описание
      5. Подробное описание
      6. Картинка
      7. Действия
      8. Сохранить – добавляет новый товар
      9. Картинка – добавляет/изменяет картинку товара
  13. 404
      1. Доступно всем
      2. Отображается в случае отсутствия ресурса
  14. 403
      1. Доступно всем
      2. Отображается в случае запрета на использование ресурса

## Требования к составу и параметрам технических средств

* 1. Сервер
     1. Процессор Apple M1
     2. Оперативная память 16 Гб
     3. Свободное пространство на диске 10 Гб
  2. Клиент
     1. Процессор Apple M1
     2. Оперативная память 16 Гб
     3. Свободное пространство на диске 10 Гб

## Требования к составу и параметрам технических средств

* 1. Сервер
     1. ОС Sequoia (15.1.1)
     2. Redis (7.2)
     3. Minio (RELEASE 2022-10-15T19-57-03Z)
     4. PostgreSQL (16)
     5. Docker
     6. Node JS
     7. Python3.12
     8. Django5.1
  2. Клиент
  3. Браузер (Safari 16.5.2, Firefox 121.0, Chrome 119.0.6045, Yandex 24.6.3.729, Opera 105.0.4970.16)

# ПРИЛОЖЕНИЕ Б СПИСОК HTTP МЕТОДОВ

Таблица 1 – HTTP методы разрабатываемого веб-сервиса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Тип** | **URL** | **Описание** | **Входные данные** | **Выходные данные** |
| 4.1.1 | GET | /datacenter-services/ | Возвращает все товары  Доступно всем | Datacenter\_max\_price  Datacenter\_min\_price | |  | | --- | | id | | name | | description | | image\_url | | price | |
| 4.1.2 | POST | /datacenter-services/create/ | Добавляет товар,  доступно администратору | {   |  | | --- | | name | | description | | image\_url | | price | |  |  |   } | id   |  | | --- | | name | | description | | image\_url | | price | |
| 4.1.3 | GET | /datacenter-services/{id}/ | Получает данные товара,  Доступно всем | pk: int | |  | | --- | | id | | name | | description | | image\_url | | price | |
| 4.1.4 | PUT | /datacenter-services/{id}/update/ | Изменяет данные товара (кроме картинки), доступно администратору | {   |  |  | | --- | --- | | name | string title: Name maxLength: 255 minLength: 1 | | description | string title: Description x-nullable: true | | image\_url | string($uri) title: Image url maxLength: 200 x-nullable: true | | price | integer title: Price maximum: 2147483647 minimum: 0 x-nullable: true | |  |  |   } | |  | | --- | | id | | name | | description | | image\_url | | price | |
| 4.1.5 | DELETE | /datacenter-services/{id}/delete/ | Удаляет товар, доступно администратору | pk: int |  |
| 4.1.6 | POST | /Datacenter-services/{id}/add-to-draft/ | Добавляет товар в заказ | pk: int | |  | | --- | | id | | status | | creation\_date | | formation\_date | | completion\_date | | creator\_name | | moderator\_name | | delivery\_address | | delivery\_time | | total\_price | | services | |
| 4.1.7 | POST | /datacenter-services/{id}/add-image/ | Добавляет/изменяет картинку товара, доступно администратору | pk: int  {  pic: Image } | image\_url |
| 4.1.8 | GET | /datacenter-orders/ | Получает заказы | |  | | --- | | datacenter\_status  string  (query) | | datacenter\_start\_date  string  (query) | | datacenter\_end\_date  string  (query) | | |  | | --- | | id | | status | | creation\_date | | formation\_date | | completion\_date | | creator\_name | | moderator\_name | | delivery\_address | | delivery\_time | | total\_price | | services | |  | |
| 4.1.9 | GET | /datacenter-orders/{id}/ | Получает заказ | pk: int | |  | | --- | | id | | status | | creation\_date | | formation\_date | | completion\_date | | creator\_name | | moderator\_name | | delivery\_address | | delivery\_time | | total\_price | | services | |
| 4.1.10 | PUT | /datacenter-orders/{id}/finalize/ | Изменяет поля заказа | pk: int  {   |  | | --- | | delivery\_address | | delivery\_time |   }  \* - эти поля проверяются на заполненность. | {  status  } |
| 4.1.11 | DELETE | /datacenter-orders/{id}/delete/ | Удаляет заказ | pk: int | Id\* |
| 4.1.12 | PUT | /datacenter-orders/{id}/submit/ | Сохраняет заказ | pk: int | |  | | --- | | delivery\_address | | delivery\_time | |
| 4.1.13 | PUT | /datacenter-orders/{id}/update/ | Модерирует заказ, доступно модератору | pk: int  {  isAccepted: bool } | |  | | --- | | status | | formation\_date | | completion\_date | | delivery\_address | | delivery\_time | | total\_price | | services | |
| 4.1.14 | DELETE | Datacenter\_service\_order/delete | Удаляет товар в заказе | datacenter\_order\_id | {  status  } |
| 4.1.15 | PUT | /datacenter-orders-services/{datacenter\_order\_id}/datacenter-services/{datacenter\_service\_id}/update/ | Меняет поле количества, доступно пользователю | datacenter\_order\_id \*  string  (path)  datacenter\_service\_id \*  string  (path) | |  | | --- | | status | | formation\_date | | completion\_date | | delivery\_address | | delivery\_time | | total\_price | | services | |
| 4.1.16 | POST | /users/ | Регистрирует пользователя, доступно всем | {  username: string,  password: string,  first\_name: string,  last\_name: string,  is\_staff: bool,  is\_superuser: bool } | {  userID: int,  username: string,  password: string,  first\_name: string,  last\_name: string,  is\_staff: bool,  is\_superuser: bool,  last\_login: date,  is\_active: bool } |
| 4.1.17 | PUT | /users/{user\_id}/ | Личный кабинет | pk: int {  username: string,  password: string,  first\_name: string,  last\_name: string,  is\_staff: bool,  is\_superuser: bool } | {  userID: int,  username: string,  password: string,  first\_name: string,  last\_name: string,  is\_staff: bool,  is\_superuser: bool,  last\_login: date,  is\_active: bool } |
| 4.1.18 | POST | /users/login/ | Аутентификация, доступна всем | {  username: string,  password: string } | {  status: string  } |
| 4.1.19 | POST | /users/logout/ | Деавторизация | {  } | {  status: string  } |