



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 \_\_\_\_\_ СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ \_\_\_\_\_

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ДОМАШНЕМУ ЗАДАНИЮ

НА ТЕМУ:

Система заявок на производстве

Студент ИУ5-51Б  
(Группа)  
(И.О.Фамилия)

\_\_\_\_\_  
(Подпись, дата) Бирюкова Е.И.

Руководитель курсовой работы

\_\_\_\_\_  
(Подпись, дата) А.И. Канев  
(И.О.Фамилия)

2024 г.

## АННОТАЦИЯ

Расчётно-пояснительная записка содержит 30 страниц. С приложениями объем составляет 42 страниц. Работа включает в себя 8 диаграмм и 25 изображение системы. В процессе выполнения было использовано 8 источников.

Объектом разработки является система заявок на производстве. Данная система позволяет автоматизировать процесс подачи и обработки заявок на производстве различного рода деталей, а также улучшить взаимодействие между заказчиками и работниками.

Цель работы заключается в разработке эффективной системы обработки заявок на производстве, которая включает функциональные возможности для добавления и редактирования данных о программах, формирования заказов, их подтверждения или отклонения, а также отслеживания их статуса.

В ходе работы была спроектирована архитектура системы, разработан интерфейс для взаимодействия посетителей и заказчиков с системой, развернут веб-сервер и приложения, предоставляющие доступ к функционалу системы через нативное и прогрессивное веб-приложения.

Пояснительная записка содержит 2 приложения.

## СОДЕРЖАНИЕ

|  |           |
|--|-----------|
| <b>АННОТАЦИЯ .....</b>   | <b>2</b>  |
| <b>СОДЕРЖАНИЕ .....</b>  | <b>3</b>  |
| <b>ВВЕДЕНИЕ .....</b>  | <b>4</b>  |
| <b>1 ПРЕДМЕТНАЯ ОБЛАСТЬ .....</b>                                | <b>6</b>  |
| <b>2 АРХИТЕКТУРА.....</b>  | <b>10</b> |
| <b>3 АЛГОРИТМЫ .....</b>   | <b>14</b> |
| <b>4 ОПИСАНИЕ ИНТЕРФЕЙСА .....</b>                               | <b>16</b> |
| <b>ЗАКЛЮЧЕНИЕ .....</b>  | <b>24</b> |
| <b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....</b>                    | <b>25</b> |
| 1. Введение .....  | 27        |
| 2. Назначение разработки.....                                    | 27        |
| 3. Стадии и этапы разработки.....                                | 27        |
| 4. Требования к функциональным характеристикам .....             | 28        |
| 5. Требования к составу и параметрам технических средств.....    | 32        |
| 6. Требования к информационной и программной совместимости ..... | 32        |
| <b>ПРИЛОЖЕНИЕ Б СПИСОК HTTP МЕТОДОВ .....</b>                    | <b>39</b> |

## ВВЕДЕНИЕ

В настоящее время заводы получают все больше заказов на производство тех или иных изделий. В связи с этим, работники, которые обрабатывают эти заказы, испытывают большие трудности. Для того, чтобы решить эту проблему, была разработана система заявок на производстве.

Целью работы является реализация системы для автоматизации обработки заявок на производство деталей, которая будет состоять из веб-приложения, веб-сервиса и десктопного приложения.

Основное назначение разработанной системы заключается в автоматизации и оптимизации процесса обработки заявок на производстве. Система позволит заказчикам просматривать всю необходимую информацию о программах производства. Также она позволит заказчикам – быстро и легко заказывать все необходимые работы, создавая на них заявки, а работникам – принимать / отклонять их, тем самым сделав процесс обработки заявок на производстве удобнее.

Нефункциональные требования к разрабатываемой системе:

1. Должна поддерживаться кроссплатформенность.
2. Интерфейс системы и текст ошибок должны быть на русском языке.

В ходе работы необходимо выполнить следующие задачи:

1. Создать MVP и базовый дизайн на основе stankonsalt.ru
2. Создать базу данных для хранения информации о заказчиках, программах, заказах и заказчиках
3. Создать веб-сервис на языке Python с использованием Django Rest Framework.
4. Реализовать авторизацию и хранение сессий в Redis
5. Разработать базовый SPA на React для гостя
6. Внедрить адаптивность, менеджер состояний Redux Toolkit, PWA, разработать Taui приложение

7. Завершить разработку интерфейса заказчика на React, использовать для обращений к методам веб-сервиса Axios
8. Реализовать интерфейс получателя в React
9. Разработать десктопное приложение Tauri
10. Развернуть приложение при помощи GitHub Pages
11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм
12. Оформить git-репозиторий на сервисе GitHub, содержащий исходный код работы.

# 1 ПРЕДМЕТНАЯ ОБЛАСТЬ

Процесс обработки заявок на производстве технически довольно сложен [1]. Разработанная система позволяет заказчикам быстро и удобно формировать заявки на производство. Каждый заказ вначале создается как черновик, в который можно добавлять и из которого можно удалять программы. Один заказ может содержать сразу несколько программ.

Каждая программа имеет название, описание, длительность и фото. Работники производства имеют возможность отредактировать любую программу. Также система предусматривает возможность добавление новых программ, а также удаление и редактирование уже существующих программ администрацией.

Когда заказчик определится с программами для заказа, он может либо удалить свой черновой заказ, либо сформировать его.

Сформированные заказы отправляются на рассмотрение администрации. Администраторы, в свою очередь, принимают решение об одобрении или отклонении заказов.

Заказчики могут следить за статусами своих сформированных заказов, в режиме реального времени, на отдельной странице.

Функции пользователей с разными ролями описаны на диаграмме прецедентов (рис. 1).

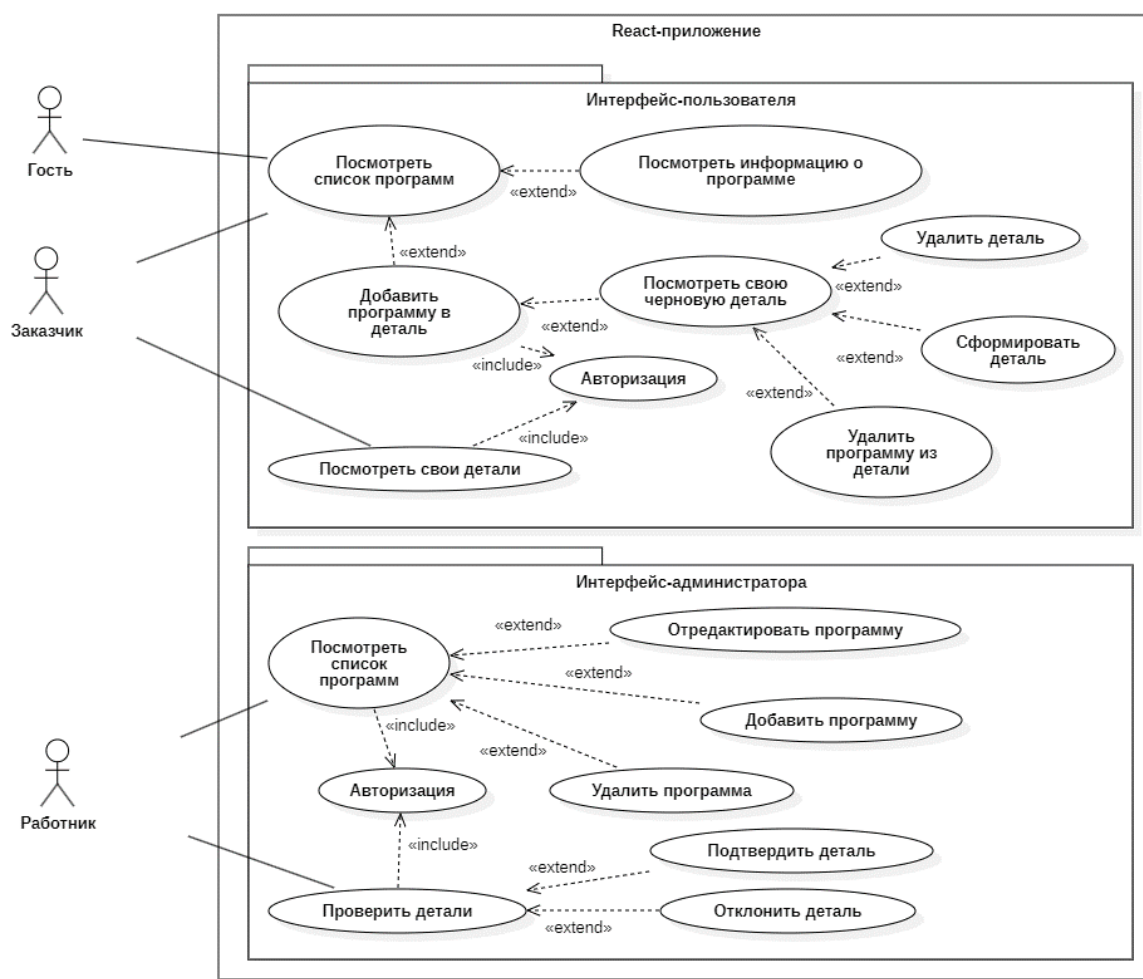


Рисунок 1 – Диаграмма прецедентов

Гостям доступен просмотр списка программ. Гости, прошедшие этап регистрации, являются заказчиками. Заказчики могут добавлять программы в заказ, формировать его, а также просматривать список своих заказов.

Заказчик выбирает программы, которым намеревается заказать, затем на основе выбранного списка формирует заказ. После чего заказчик может просмотреть её, а также согласовать или отклонить. Возможные состояния статуса заказа отображены на диаграмме состояний (рисунок 2)

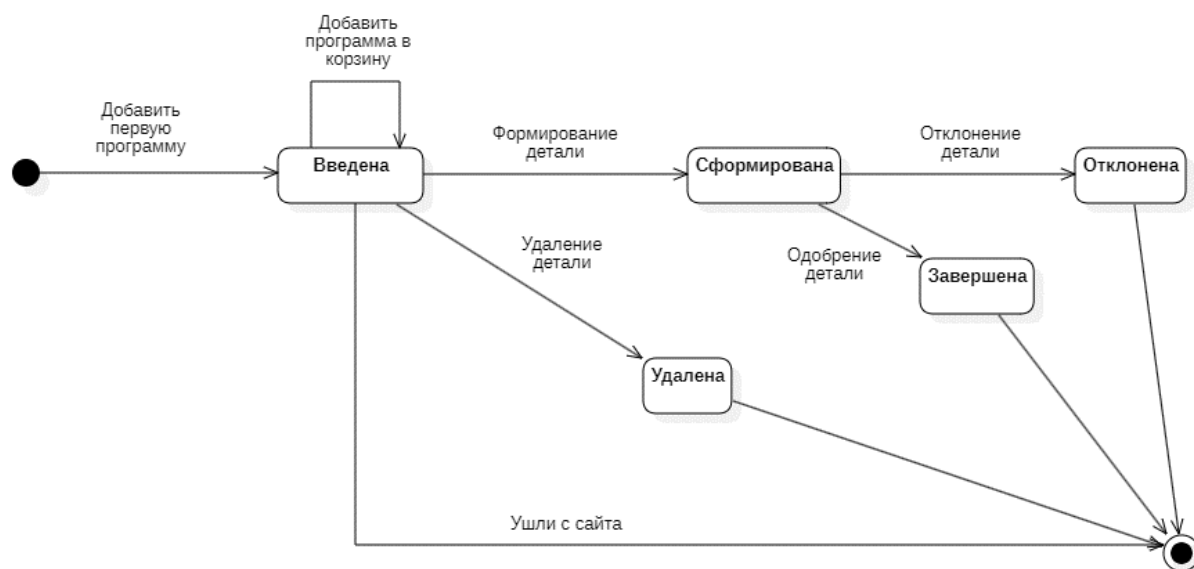


Рисунок 2 – Диаграмма состояний заявок

Заказы обрабатываются работниками производства. В результате обработки заказа его либо одобряют, либо отклоняют. Заказчику также доступны операции для работы со списком заказов: редактирование, добавление и удаление программ, а также просмотр списка всех заказов в табличном виде.

Процесс оформления заказа отражен на диаграмме деятельности (рисунок 3).

В начале взаимодействия с системой оформления заказов заказчик запрашивает список доступных программ. При необходимости заказчик может добавить программы в текущий черновой заказ. После чего может продолжить выбор программ из каталога или перейти к оформлению заказа. Затем заказчик должен указать свое имя. По нажатии кнопки «Оформить» отправка заказа формируется и становится доступна в интерфейсе заказчика.

Работник имеет возможность просматривать заказы посетителей, а также принимать решение об их одобрении или отклонении.



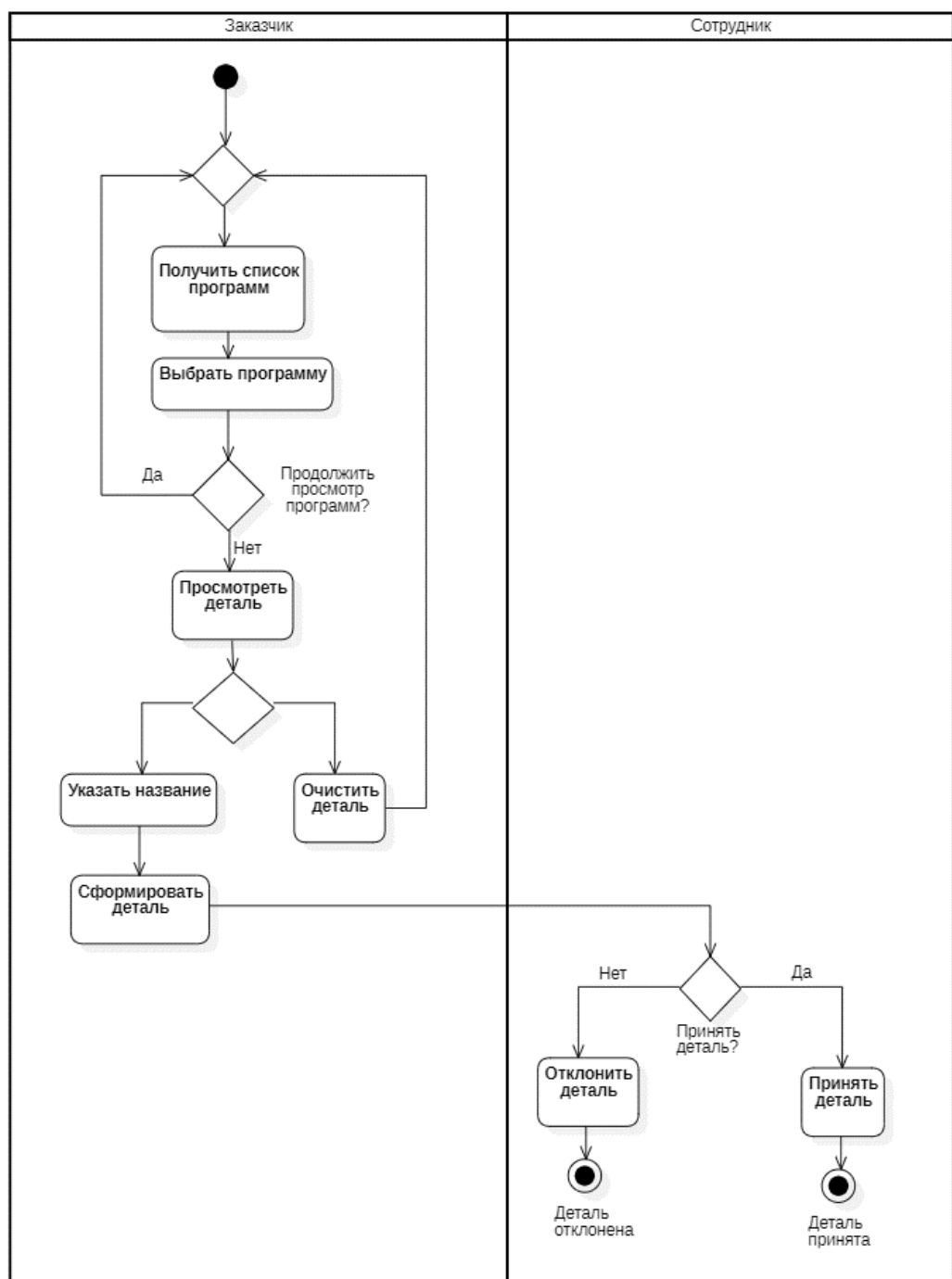


Рисунок 3 – Диаграмма деятельности

## 2 АРХИТЕКТУРА

Архитектура системы отображена на диаграмме развертывания (рисунок 4).

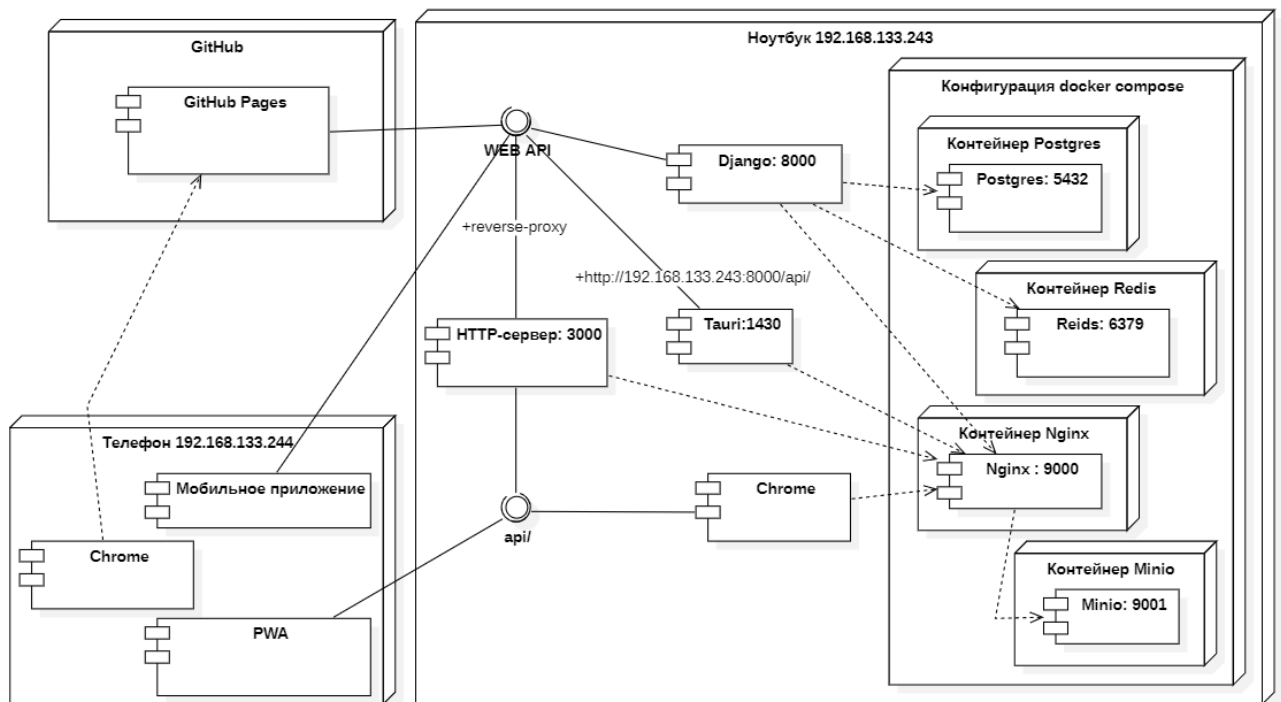


Рисунок 4 – Диаграмма развертывания

Веб-сервис, реализованный на языке Python [2] с использованием фреймворка Django Rest Framework [3] связан с серверами Redis [4], Minio [5], и СУБД PostgreSQL [6]. В Redis хранятся идентификаторы сессий пользователей, добавление которых происходит при входе пользователя в систему.

Язык программирования Python был выбран благодаря его хорошей производительности и простоте синтаксиса. Использование фреймворков Django и Django Rest Framework позволяет создавать надежные и эффективные веб-сервисы. Веб-сервис на Django является общим для веб-сервера и десктопного приложения Tauri [7].

Данные хранятся в СУБД PostgreSQL. Она была выбрана, является стандартом современной индустрии разработки.

Структура данных отражена на ER диаграмме (рисунок 5). Модель Программы представляет собой набор полей, отражающих параметры

программы. Данные о заказах хранятся в таблице Заказы. Для хранения в одном заказе нескольких программ используется промежуточная таблица связи М-М. Данные о пользователях системы (заказчиках и заказчиках) хранятся в таблице «Пользователь».

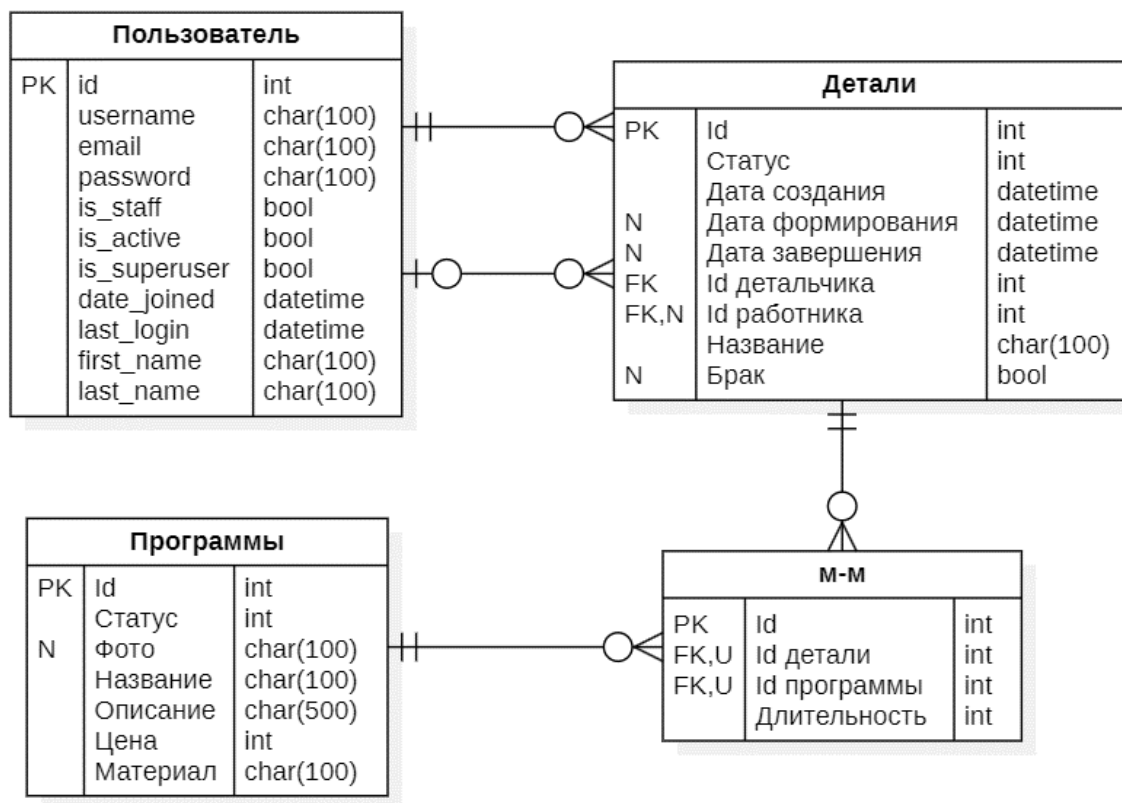


Рисунок 5 – ER диаграмма.

HTTP-сервер реализован с использованием языка TypeScript и фреймворка React [8]. Устройство бэкенд-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.

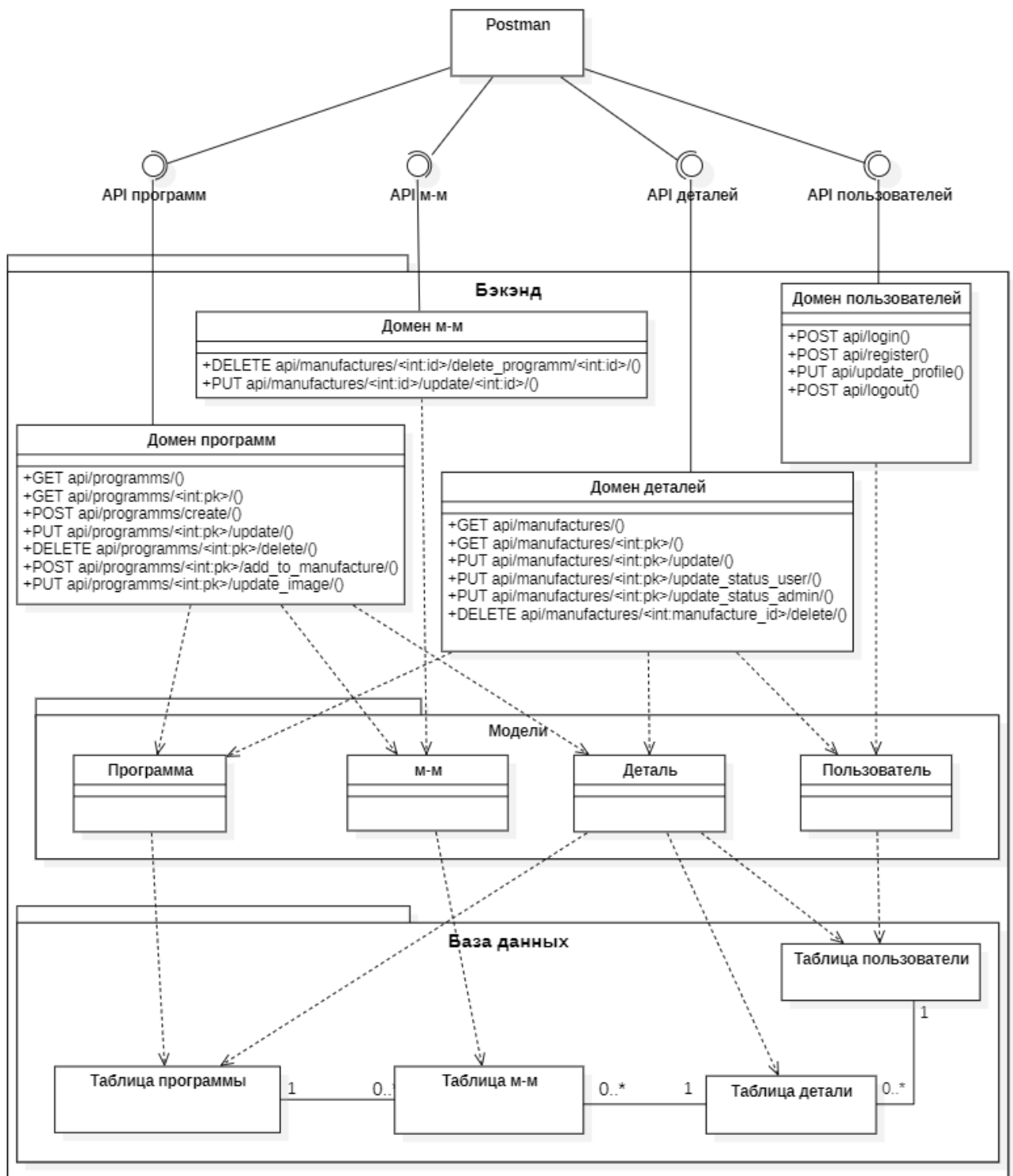


Рисунок 6 – Диаграмма классов бэкенда

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.

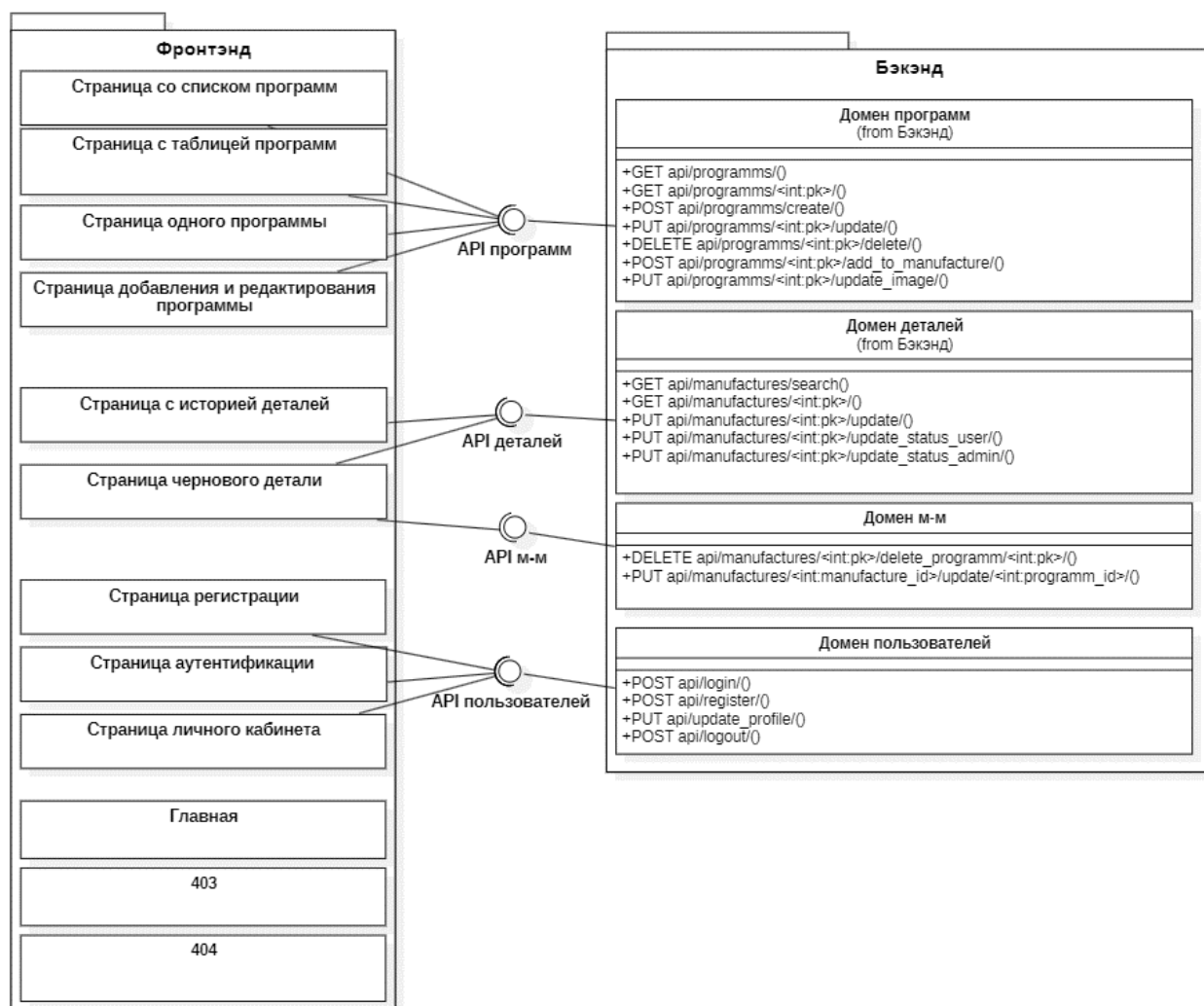


Рисунок 7 – Диаграмма классов фронтенда

### 3 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: пользователи, программы, заказы, м-м. Методы следуют правилам REST API.

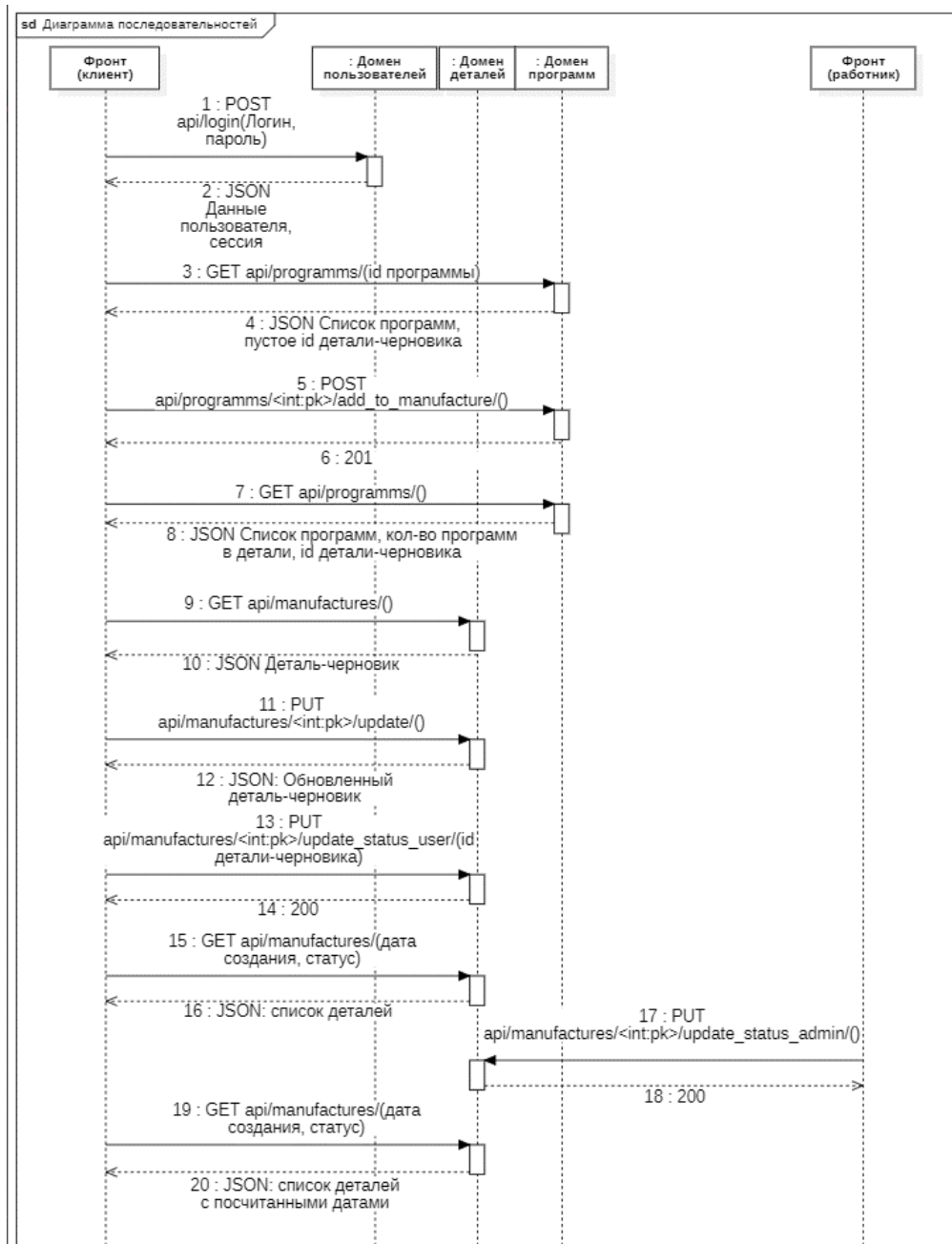


Рисунок 8 – Диаграмма последовательности

В начале бизнес-процесса происходит аутентификация пользователя. Через графический интерфейс гость вводит логин и пароль для доступа к системе. Если учетная запись с указанными данными существует в базе, сервис возвращает информацию о пользователе и идентификатор сессии в ответе. Если же учетной записи с введенным логином не существует или пароль неверен, пользователь получает сообщение об ошибке. В таком случае пользователю предлагается пройти регистрацию или повторно ввести корректные данные. На этом этапе система также определяет роль пользователя — заказчик или заказчик.

После успешной аутентификации заказчик через графический интерфейс запрашивает у веб-сервиса список доступных программ, который возвращается в формате JSON. Заказчик может выбрать нужную программу и добавить его в заказ. Для этого он нажимает кнопку «Добавить» в графическом интерфейсе. Процесс выбора программ и их добавления в черновой заказ может повторяться несколько раз.

Когда заказчик завершает выбор программ и указывает необходимую информацию о заказе, он нажимает кнопку «Сформировать». Приложение отправляет на веб-сервис запрос на формирование заказа. После этого заказчик может отслеживать статус своих заказов на специальной странице интерфейса, где отображаются все его созданные заказы с указанием текущего статуса.

## 4 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 посетителя)

На странице с формой регистрации (рисунок 11) гость может создать новый аккаунт.

The screenshot shows the registration page. At the top is the same header as in previous images. Below the header, there is a link 'Регистрация /'. The main content is a form titled 'Форма регистрации'. It contains three input fields: 'Введите имя' with the value 'Антон', 'Введите почту' with the value 'anton@mail.ru', and 'Введите пароль' with masked characters '\*\*\*\*'. Below these fields is a blue button labeled 'Зарегистрироваться'.

Рисунок 11 – Страница регистрации

На странице с формой входа (рисунок 12) заказчик может войти в свой аккаунт. При успешном вводе данных аккаунта на клиент приходит идентификатор сессии, который хранится в cookie браузера.



[Вход /](#)

**Форма входа**

Введите логин

anton

Введите пароль

....

Войти

Рисунок 12 – Страница входа

После входа в аккаунт заказчик попадает на главную страницу (рисунок 13). На этой странице располагается описание разработанной системы.

[Главная /](#)

### Система заявок на производстве

Наша компания предоставляет услуги по изготовлению деталей на деталь по чертежам, эскизам и образцам детальчика. Мы производим детали с высокой точностью, с чётким соблюдением сроков - используя высокоточные обрабатывающие центры с ЧПУ токарной, токарно-фрезерной группы. Принимаем в работу серийные и мелкосерийные детали, а также индивидуальные поручения сложных исполнений. Имеем парк современного, собственного оборудования - в чём не сложно убедиться посетив наше производство по адресу указанному в шапке сайта.

Рисунок 13 – Главная страница

Страница списка программ (рисунок 14) содержит список программ доступных к добавлению в заказ. При нажатии на кнопку “Открыть” на любой карточке, открывается страница с подробной информацией о программе. Также с помощью нажатия на кнопку «Добавить» пользователи могут добавлять программы в черновой заказ. Вверху страницы находится строка для поиска программ по имени.

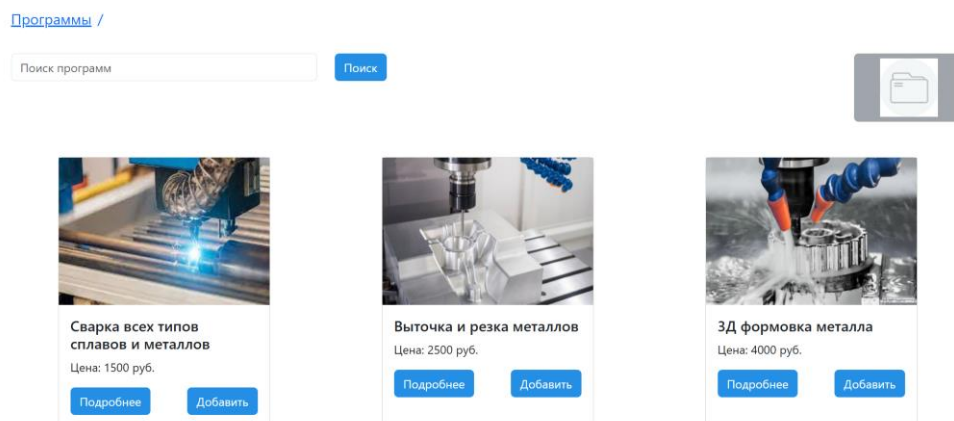


Рисунок 14 – Страница списка программ

Страница с подробной информацией о программе выглядит следующим образом (рисунок 15).

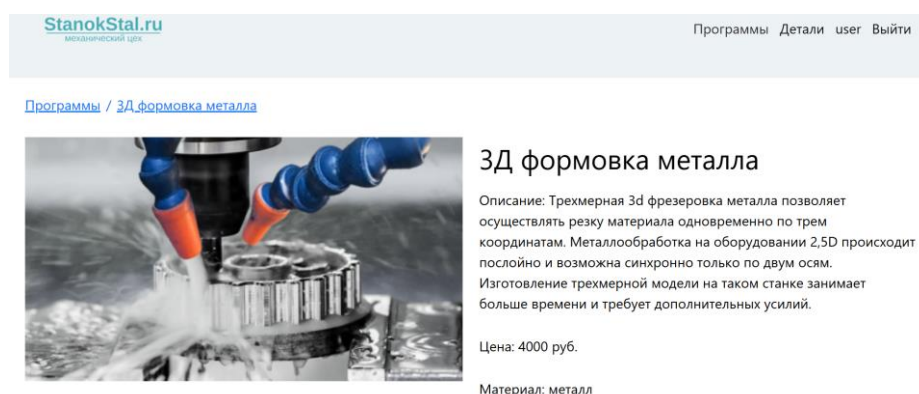



Рисунок 15 – Страница с подробной информацией о программе

На странице одного заказа (рисунок 16) отображается полная информация о заказе, есть возможность удалить программу из заказа, указать название, длительность в минутах каждой входящей в заказ программы, а также сформировать черновой заказ или удалить его. Редактирование и удаление оформленных заказов запрещено.

## Черновая деталь

Название

Введите название



ЗД формовка металла

Цена: 4000 руб.

Длительность (мин)

[Подробнее](#) [Удалить](#)

[Сохранить](#)

[Отправить](#)

[Удалить](#)

Рисунок 16 – Страница черного заказа

На странице списка заказов (рисунок 17) заказчики могут просматривать созданные ими заказы. На этой странице можно посмотреть подробную информацию о заказах, нажав на интересующую строку таблицы.

От

30.12.2024

До

02.03.2025

Статус

Любой

Применить

| № | Статус   | Брак | Дата создания      | Дата формирования  | Дата завершения    | Действие | QR-код |
|---|----------|------|--------------------|--------------------|--------------------|----------|--------|
| 1 | Завершен | Нет  | 17 января 2025 г.  | 29 января 2025 г.  | 30 января 2025 г.  | Открыть  |        |
| 2 | В работе | Нет  | 17 ноября 2024 г.  | 26 января 2025 г.  |                    | Открыть  |        |
| 3 | В работе | Нет  | 4 января 2025 г.   | 13 января 2025 г.  |                    | Открыть  |        |
| 4 | В работе | Нет  | 26 декабря 2024 г. | 30 декабря 2024 г. |                    | Открыть  |        |
| 5 | Завершен | Да   | 29 ноября 2024 г.  | 30 декабря 2024 г. | 31 декабря 2024 г. | Открыть  |        |
| 6 | Завершен | Да   | 20 ноября 2024 г.  | 29 декабря 2024 г. | 30 декабря 2024 г. | Открыть  |        |

Рисунок 17 – Страница списка заказов

Пользователь может изменить свой пароль на странице редактирования данных пользователя (рисунок 18). Она открывается при нажатии на кнопку «Изменить профиль», которая находится в выпадающем меню, которое активируется нажатием на логин пользователя в меню.

[Личный кабинет /](#)

Логин: user

Сбросить пароль

Введите новый пароль

Сохранить

Рисунок 18 – Страница редактирования информации о пользователе

В случае, если пользователь запрашивает несуществующую страницу, он перенаправляется на страницу ошибки 404 (рисунок 19).

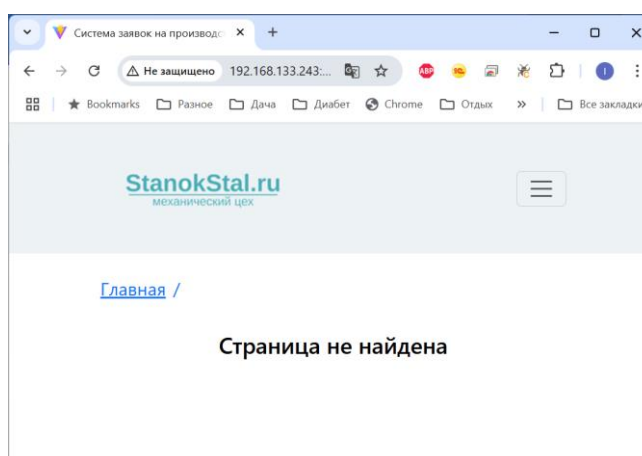


Рисунок 19 – Страница 404

В случае, если пользователь пытается получить доступ к страницам, для просмотра которых ему не хватает прав, он перенаправляется на страницу ошибки 403 (рисунок 20).



[Главная /](#)

Нет доступа

Рисунок 20 – Страница 403

Приложение поддерживает возможность работы как прогрессивное веб-приложение (PWA), что позволяет пользователю устанавливать его на устройство как нативное приложение. Интерфейс адаптирован для мобильных

устройств, сохраняя функциональность основной версии, и состоит из трех страниц: главная страница PWA (рисунок 21, а), страница списка программ (рисунок 21, б) и страница с подробной информацией о программе (рисунок 21, в).

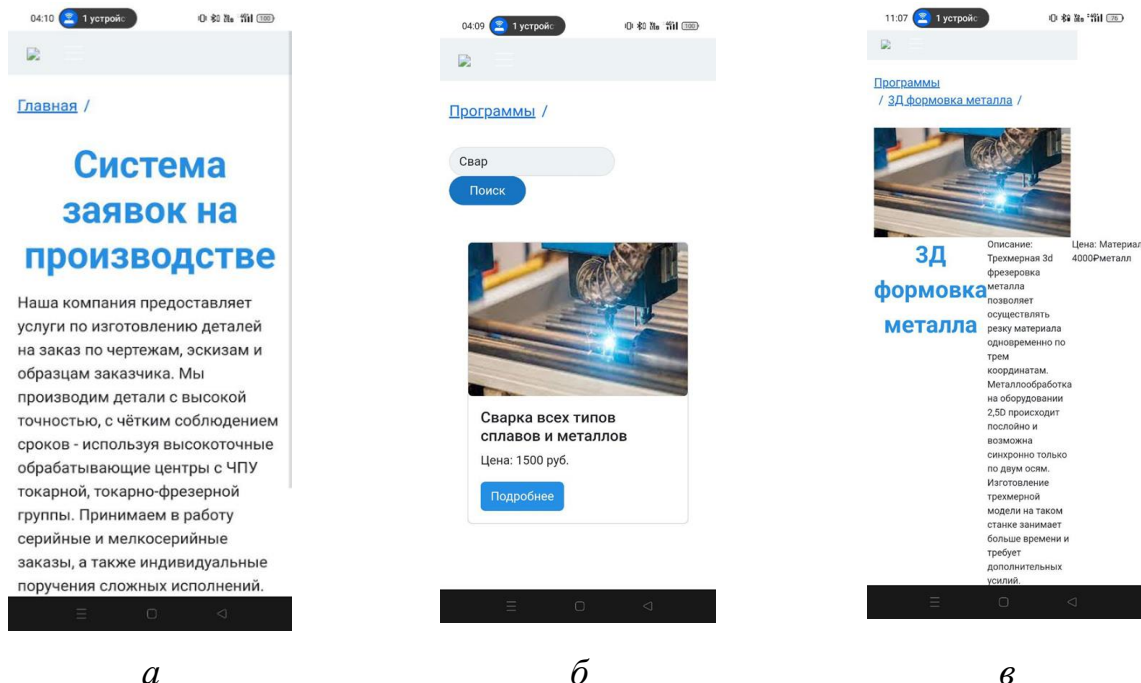


Рисунок 21 – Интерфейс PWA

а – главная страница; б – страница списка программ; в – страница с подробной информацией о программе

Была разработана десктопная версия приложения на основе Tauri. Интерфейс состоит из трех страниц: главная страница (рисунок 22), страница списка программ (рисунок 23) и страница с подробной информацией о программе (рисунок 24).

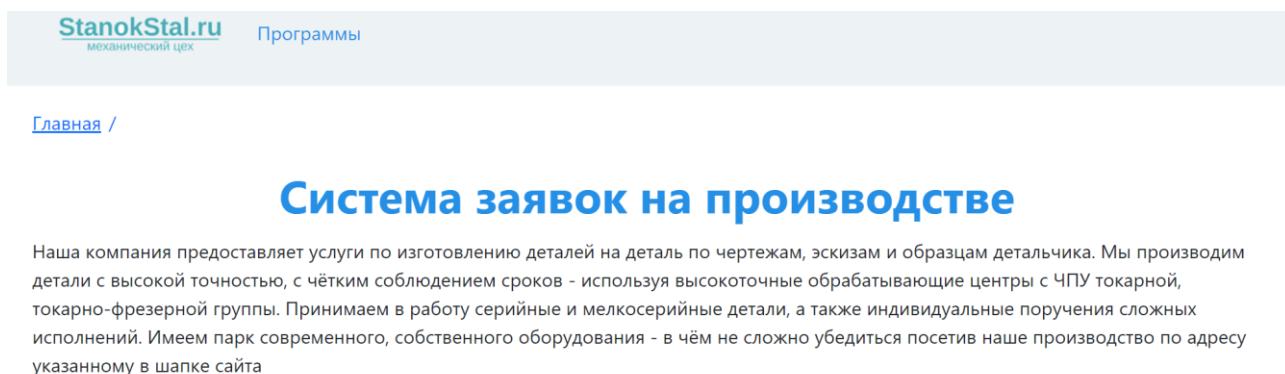


Рисунок 22 – Главная страница Tauri

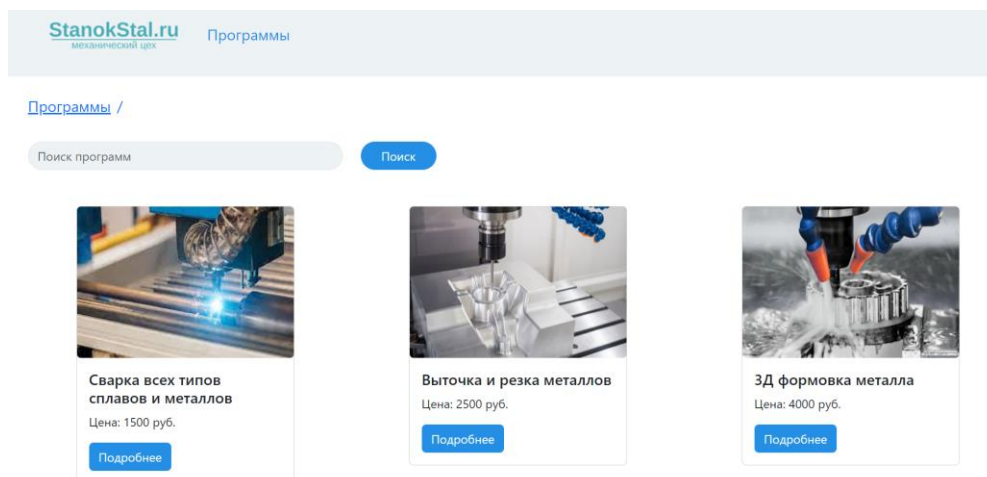


Рисунок 23– Страница списка программ Tauri

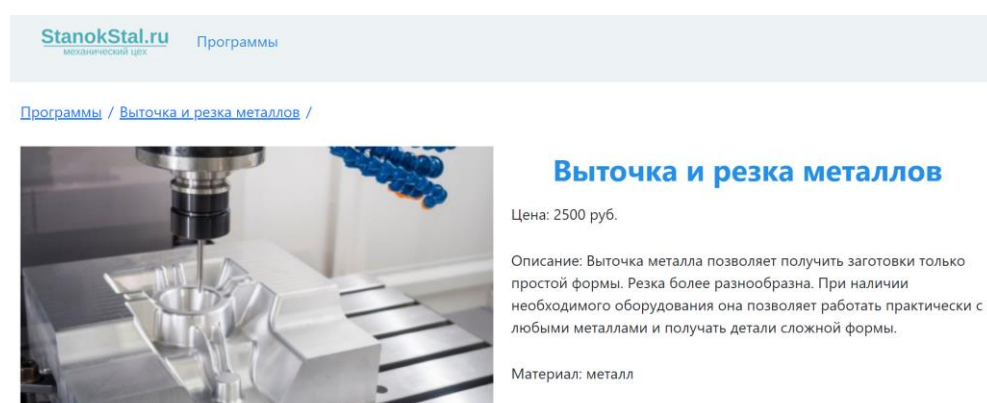
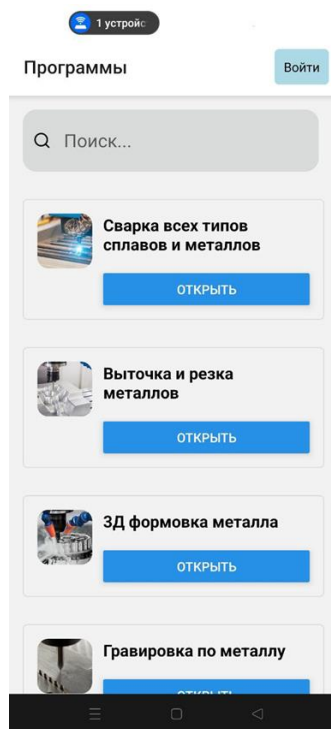
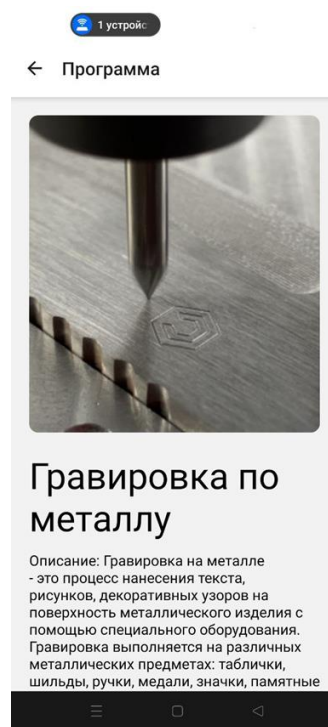


Рисунок 24 – Страница с подробной информацией о программе Tauri

Также было разработано приложения для гостя на Android и подключение к веб-сервису. Простое нативное приложение для интерфейса гостя (без авторизации и редактирования) состоит из 4 страниц с фильтрацией и картинками. Приложение подключается к разработанному API через IP адрес в локальной сети. Интерфейс включает в себя страницу списка программ (рисунок 25, а), страницу с подробной информацией о программе (рисунок 25, б), страницу аутентификации (рисунок 26, а) и страницу со списком заявок на производство (рисунок 26, б).



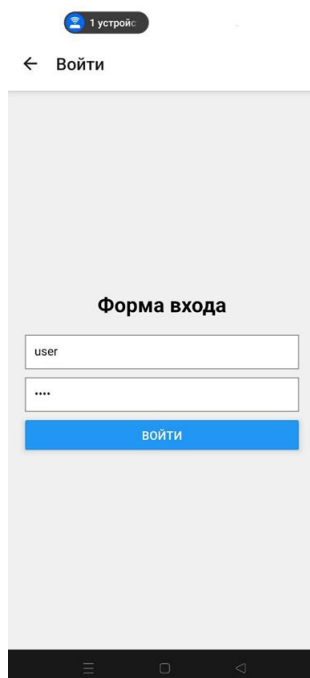
*a*



*б*

Рисунок 25 – Интерфейс Exro Go

*a* – страница списка программ; *б* – страница с подробной информацией о программе



*a*



*б*

Рисунок 26 – Интерфейс Exro Go

*a* – страница аутентификации; *б* – страницу со списком заявок на производство

## ЗАКЛЮЧЕНИЕ

В ходе работы были достигнуты следующие результаты:

1. Разработан дизайн приложения в Figma на основе stankonsalt.ru. После ознакомления с разработкой бэкенда с использованием фреймворка Django создан MVP.
2. Спроектирована и создана база данных PostgreSQL для хранения информации о заказчиках, отправках файлов и пользователей, а затем подключена к бэкенду.
3. Создан веб-сервис на Django.
4. Реализована авторизация и хранение сессий в Redis.
5. Разработан базовый интерфейс приложения для гостя на React.
6. Внедрена адаптивность, менеджер состояний Redux Toolkit, PWA, разработано Tauri приложение.
7. Завершена разработка интерфейса пользователя в React, для обращений к методам веб-сервиса использован Axios.
8. Реализован React интерфейс получателя, внедрен Real-time web.
9. Реализовано десктопное приложение Tauri.
10. Приложение развернуто при помощи GitHub Pages и доступно по ссылке: ...
11. Подготовлен набор документации, включающий РПЗ, ТЗ и набор диаграмм.
12. Оформлен git-репозиторий на сервисе GitHub, содержащий исходный код работы: ...



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Изготовление Деталей и Металлообработка на станках с ЧПУ [Электронный ресурс]. // <https://stankonsalt.ru/>;
2. Документация по Python [Электронный ресурс]. // Python URL: <https://docs.python.org/3/> (дата обращения: 28.10.2024);
3. Документация по Django REST Framework [Электронный ресурс] // Django REST Framework. URL: <https://www.django-rest-framework.org/> (дата обращения: 28.10.2024);
4. Документация по Redis [Электронный ресурс] // Netlify. URL: <https://master--redis-doc.netlify.app/docs/> (дата обращения: 10.10.2024);
5. Документация Minio [Электронный ресурс] // Min. URL: <https://min.io/docs/minio/kubernetes/upstream/index.html> (дата обращения: 09.09.2024);
6. Документация PostgreSQL [Электронный ресурс] // Postgresql. URL: <https://www.postgresql.org/docs/> (дата обращения: 15.09.2024);
7. Документация по Tauri [Электронный ресурс] // Tauri. URL: <https://v2.tauri.app/develop/> (дата обращения: 20.11.2024);
8. 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  
\_\_\_\_\_ В.И.Терехов  
" \_\_ " \_\_\_\_\_ 2024 г.

Согласовано  
Научный руководитель  
\_\_\_\_\_ А.И. Канев  
" \_\_ " \_\_\_\_\_ 2024 г.

**Система заявок производство**

Техническое задание

(вид документа)

писчая бумага

(вид носителя)

7

(количество листов)

ИСПОЛНИТЕЛЬ:

\_\_\_\_\_ Бирюкова Екатерина Ильинична  
" \_\_ " \_\_\_\_\_ 2024 г.

Москва - 2024

## **1. Введение**

Реализовать систему для автоматизации обработки заявок на производство деталей, которая будет состоять из веб-приложения, веб-сервиса и десктопного приложения.

## **2. Назначение разработки**

Основное назначение разработанной системы заключается в автоматизации и оптимизации процесса обработки заявок на производстве. Система позволит заказчикам просматривать всю необходимую информацию о программах производства. Также она позволит заказчикам – быстро и легко заказывать все необходимые работы, создавая на них заявки, а работникам – принимать / отклонять их, тем самым сделав процесс обработки заявок на производстве удобнее.

## **3. Стадии и этапы разработки**

- 3.1. Создание MVP и базового дизайна в Figma на основе stankosalt.com
- 3.2. Спроектировать базу данных PostgreSQL
- 3.3. Создать веб-сервис на Django Rest Framework
- 3.4. Реализовать авторизацию и хранение сессий в Redis
- 3.5. Разработать SPA на React для гостя
- 3.6. Внедрить адаптивность, а также возможность работы в режиме PWA
- 3.7. Развернуть веб-приложение в GitHub Pages
- 3.8. Разработать Tauri приложение
- 3.9. Реализовать интерфейс заказчика на React с менеджером состояний Redux Toolkit
- 3.10. Добавить в React приложение интерфейс заказчика
- 3.11. Подготовка всей документации (РПЗ, ТЗ и набор диаграмм)
- 3.12. Подготовка репозитория на GitHub

## **4. Требования к функциональным характеристикам**

### **4.1. Методы HTTP**

- 4.1.1. GET Получение всех программ
- 4.1.2. POST Создание программы
- 4.1.3. GET Получение одного программы
- 4.1.4. PUT Изменение программы
- 4.1.5. DELETE Удаление программы
- 4.1.6. POST Добавление программы в заказ
- 4.1.7. POST Изменение/добавление картинки программы
- 4.1.8. GET Получение всех заказов
- 4.1.9. GET Получение одного заказа
- 4.1.10. PUT Изменение полей заказа
- 4.1.11. DELETE Удаление заказа
- 4.1.12. PUT Сохранение заказа
- 4.1.13. PUT Модерирование заказа
- 4.1.14. DELETE Удаление программы из заказа
- 4.1.15. PUT Изменение значения м-м программы в заказе
- 4.1.16. POST Регистрация
- 4.1.17. PUT Личный кабинет
- 4.1.18. POST Аутентификация
- 4.1.19. POST Деавторизация

### **4.2. Меню**

- 4.2.1. Главная - перенаправляет на страницу 4.5
- 4.2.2. Список программ – перенаправляет на страницу 4.6
- 4.2.3. Список заказов – перенаправляет на страницу 4.9 (доступно только авторизованным пользователям)
- 4.2.4. Редактирование программ – перенаправляет на страницу 4.11 (доступно только заказчику)

- 4.2.5. Зарегистрироваться – перенаправляет на страницу 4.3 (доступно только для гостей)
- 4.2.6. Личный кабинет – перенаправляет на страницу 4.10 (доступно только авторизованным пользователям)
- 4.2.7. Войти – перенаправляет на страницу 4.4 (доступно только для гостей)
- 4.2.8. Выйти – перенаправляет на страницу 4.6 (доступно только авторизованным пользователям) (вызывается метод 4.1.19)
- 4.3. Регистрация
  - 4.3.1. Доступно только гостям
  - 4.3.2. Отображает форму регистрации
    - 4.3.2.1. Поле логина
    - 4.3.2.2. Поле почты
    - 4.3.2.3. Поле пароля
  - 4.3.3. Действия
    - 4.3.3.1. Регистрация пользователя – (вызывается метод 4.1.16)
    - 4.3.3.2. Вернуться к аутентификации – перенаправляет на страницу 4.4
- 4.4. Аутентификация
  - 4.4.1. Доступно только гостям
  - 4.4.2. Отображает форму аутентификации
    - 4.4.2.1. Поле логина
    - 4.4.2.2. Поле пароля
  - 4.4.3. Действия
    - 4.4.3.1. Войти – (вызывается метод 4.1.18)
    - 4.4.3.2. Регистрация – перенаправляет на страницу 4.3
- 4.5. Личный кабинет
  - 4.5.1. Доступно авторизованному Заказчику
  - 4.5.2. Действия

4.5.2.1. Изменить данные пользователя – (вызывается метод 4.1.17)

#### 4.6. Главная

4.6.1. Доступна всем

4.6.2. Отображается статическое описание сервиса

#### 4.7. Список программ

4.7.1. Доступна всем

4.7.2. Отображаются карточки программ (метод 4.1.1)

4.7.2.1. Название

4.7.2.2. Картинка

4.7.2.3. Краткое описание

4.7.3. Действия

4.7.3.1. Поиск – перенаправляет на страницу 4.6, (используется метод 4.1.1), с фильтрующим параметром

4.7.3.2. Подробнее – перенаправляет на страницу 4.7 (используется метод 4.1.3)

4.7.3.3. Добавить в заказ – добавляет программу в заказ-черновик, (вызывается метод 4.1.2), только аутентифицированные Заказчики.

4.7.3.4. Кнопка корзины – перенаправляет на страницу 4.9, только аутентифицированные Заказчики.

#### 4.8. Один программу

4.8.1. Доступна всем

4.8.2. Отображается подробная информация выбранного программы, (вызывается метод 4.1.3)

#### 4.9. Один заказ

4.9.1. Доступно только аутентифицированным заказчикам

4.9.2. Отображает текущий заказ-черновик, (метод 4.1.9)

4.9.2.1. Список добавленных программ

4.9.3. Действия, доступны только в случае, если статус «черновик»

- 4.9.3.1. Убрать программу – удаляет программу из заказа, (вызывается метод 4.1.14)
- 4.9.3.2. Сохранить – сохраняет текущий заказ-черновик, (вызывается метод 4.1.12)
- 4.9.3.3. Очистить – удаляет заказ-черновик, (вызывается метод 4.1.1)
- 4.9.3.4. Сформировать – вносит данные заказа (формирует заказ, вызывается метод 4.1.10)

#### 4.10. Список заказов

- 4.10.1. Доступно авторизированному Заказчику
- 4.10.2. Отображается список заказов (метод 4.1.8)
  - 4.10.2.1. Для заказчика - только заказа заказчика
  - 4.10.2.2. Для заказчика - все заказа
- 4.10.3. Действия
  - 4.10.3.1. Фильтрация – фильтрует заказ по дате создания и статусу (вызывается метод 4.1.8)
  - 4.10.3.2. Сформировать – формирует заказ, выполняется метод 4.1.13, доступно только Заказчику
  - 4.10.3.3. Отклонить – отклоняет заказ, вызывается метод 4.1.13, доступно только Заказчику
  - 4.10.3.4. Посмотреть подробную информацию о программе – перенаправляет на страницу 4.9 (вызывается метод 4.1.9)

#### 4.11. Список программ таблицей

- 4.11.1. Доступно только заказчику
- 4.11.2. Отображаются все программы (вызывается метод 4.1.1)
- 4.11.3. Действия
  - 4.11.3.1. Удалить – удаляет программу (вызывается метод 4.1.5)
  - 4.11.3.2. Редактирование/создание – переход на страницу 4.12

#### 4.12. Редактирование/создание программы

- 4.12.1. Доступно только заказчику

#### 4.12.2. Отображается информация о программе (метод 4.1.3)

4.12.2.1. Название

4.12.2.2. Описание

4.12.2.3. Картинка

#### 4.12.3. Действия

4.12.3.1. Сохранить – обновляет существующий программу  
(вызывается метод 4.1.4)

4.12.3.2. Добавить – добавляет новый программу (метод 4.1.2)

4.12.3.3. Картинка – добавляет/изменяет картинку программы  
(вызывается метод 4.1.7)

#### 4.13. 404

4.13.1. Доступно всем

4.13.2. Отображается в случае отсутствия ресурса

#### 4.14. 403

4.14.1. Доступно всем

4.14.2. Отображается в случае запрета на использование ресурса

### **5. Требования к составу и параметрам технических средств**

#### 5.1. Сервер

5.1.1. Процессор Ryzen 1 7100f

5.1.2. Оперативная память 4 Гб

5.1.3. Свободное пространство на диске 16 Гб

#### 5.2. Клиент

5.2.1. Процессор Apple M1

5.2.2. Оперативная память 16 Гб

5.2.3. Свободное пространство на диске 10 Гб

### **6. Требования к информационной и программной совместимости**

#### 6.1. Сервер



- 6.1.1. OC Linux (6.4.12)
- 6.1.2. Redis (7.2)
- 6.1.3. Minio (RELEASE 2022-10-15T19-57-03Z)
- 6.1.4. PostgreSQL (16)
- 6.1.5. Docker
- 6.1.6. Node JS
- 6.1.7. Python3.12
- 6.1.8. Django5.1
- 6.2. Клиент
  - 6.2.1. 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  | /api/programms/             | Возвращает список программ, а также id чернового заказа<br>Доступно всем пользователям | {<br>“product_name”: char(100)<br>}  | {<br>“draft_manufacture_id”: int,<br>“programms_count”: int,<br>“programms”: [<br>{<br>“id”: int,<br>“name”: char(100),<br>“description”: char(500),<br>“price”: int,<br>“status”: int,<br>“image”: char(100)<br>},<br>...<br>]<br>} |
| 4.1.2 | POST | api/programms/create/       | Добавляет программу в черновой заказ<br>Доступно только авторизованным пользователям   | {<br>“name”: char(100),<br>“description”: char(500),<br>“number”: int<br>} | 201 / 403 / 404  |
| 4.1.3 | GET  | api/programms/<product_id>/ | Получает информацию о программе<br>Доступно всем пользователям                         | {<br>“product_id”: int<br>}  | {<br>“id”: int,<br>“name”: char(100),<br>“description”: char(500),<br>“price”: int,<br>“status”: int,<br>“image”: char(100)<br>}   |

|       |        |   |  |  |  |
|-------|--------|---|--|--|--|
| 4.1.4 | PUT    | api/programs/<product_id>/update/             | Изменяет данные программы (кроме картинки)<br>Доступно только авторизованным пользователям | {<br>"product_id": int<br>}                    | {<br>"id": int,<br>"name": char(100),<br>"description": char(500),<br>"price": int,<br>"status": int,<br>"image": char(100)<br>}                   |
| 4.1.5 | DELETE | api/programs/<product_id>/delete/             | Удаляет программу<br>Доступно только авторизованным пользователям                          | {<br>"product_id": int<br>}                    | [<br>{<br>"id": int,<br>"name": char(100),<br>"description": char(500),<br>"price": int,<br>"status": int,<br>"image": char(100)<br>},<br>...<br>] |
| 4.1.6 | POST   | api/programs/<product_id>/add_to_manufacture/ | Добавляет программу в черновой заказ<br>Доступно только авторизованным пользователям       | {<br>"product_id": int<br>}                    | 201 / 403 / 404  |
| 4.1.7 | POST   | api/programs/<product_id>/update_image /      | Добавляет/изменяет картинку программы<br>Доступно только авторизованным пользователям      | {<br>"product_id": int,<br>"image": bytes<br>} | {<br>"id": int,<br>"name": char(100),<br>"description": char(500),<br>"price": int,<br>"status": int,<br>"image": char(100)<br>}                   |

|       |     |                                    |   |  |  |
|-------|-----|------------------------------------|---|--|--|
| 4.1.8 | GET | api/manufactures/                  | <p>Возвращает список заказов, отфильтрованных по диапазону дат формирования и по статусам</p> <p>Доступно только авторизованным пользователям</p> | <pre>{   "status": int,   "date_start": char(100),   "date_end": char(100) }</pre> | <pre>[   {     "id": int,     "status": int,     "date_created": datetime,     "date_formation": datetime,     "date_complete": datetime,     "owner": char(100),     "moderator": char(100),     "name": char(100),     "date": date   },   ... ]</pre>   |
| 4.1.9 | GET | api/manufactures/<manufacture_id>/ | <p>Возвращает информацию о заказе</p> <p>Доступно только авторизованным пользователям</p>   | <pre>{   "cook_id": int }</pre>  | <pre>{   "id": int,   "status": int,   "date_created": datetime,   "date_formation": datetime,   "date_complete": datetime,   "owner": char(100),   "moderator": char(100),   "name": char(100),   "date": date,   "programms": [     {       "id": int,       "name": char(100),       "description": char(500),       "price": int,       "status": int,       "image": char(100),       "value": int     },     ...   ] }</pre> |

|            |        |   |  |                          |   |
|------------|--------|---|--|--------------------------|---|
|            |        |   |  |                          | ...<br>]<br>}   |
| 4.1.1<br>0 | PUT    | api/manufactures/<br><manufac<br>ture_id><br>/update/                     | Изменяет<br>информацию о<br>заказе<br>Доступно только<br>авторизованным<br>пользователям | {<br>“cook_id”: int<br>} | {<br>"id": int,<br>"status": int,<br>"date_created": datetime,<br>“date_formation”: datetime,<br>"date_complete": datetime,<br>"owner": char(100),<br>“moderator”: char(100),<br>“name”: char(100),<br>“date”: date,<br>"programms": [...]<br>} |
| 4.1.1<br>1 | DELETE | api/manufactures/<br><manufac<br>ture_id><br>/delete/                     | Удаляет заказ<br>Доступно только<br>авторизованным<br>пользователям                      | {<br>“cook_id”: int<br>} | 200 / 404 / 403   |
| 4.1.1<br>2 | PUT    | api/manufactures/<br><manufac<br>ture_id><br>/update_s<br>tatus_use<br>r/ | Формирует заказ<br>Доступно только<br>авторизованным<br>пользователям                    | {<br>“cook_id”: int<br>} | {<br>"id": int,<br>"status": int,<br>"date_created": datetime,<br>“date_formation”: datetime,<br>"date_complete": datetime,<br>"owner": char(100),<br>“moderator”: char(100),<br>“name”: char(100),<br>“date”: date,<br>"programms": [...]<br>} |
| 4.1.1<br>3 | PUT    | api/manufactures/<br><manufa  | Модерирует<br>заказ  | {<br>“cook_id”: int<br>} | {<br>"id": int,<br>"status": int,   |

|            |        |  |  |   |  |
|------------|--------|--|--|---|--|
|            |        | cture_id><br>/update_status_admin/                               | Доступно только<br>работникам  |   | "date_created": datetime,<br>"date_formation": datetime,<br>"date_complete": datetime,<br>"owner": char(100),<br>"moderator": char(100),<br>"name": char(100),<br>"date": date,<br>"programms": [...]<br>} |
| 4.1.1<br>4 | DELETE | api/manufactures/<manufacture_id>/delete_programm/<programm_id>/ | Удаляет<br>программу из<br>заказа<br><br>Доступно только<br>авторизованным<br>пользователям                  | {<br>"cook_id": int,<br>"product_id": int<br>}  | 200 / 404 / 403  |
| 4.1.1<br>5 | PUT    | api/manufactures/<int:manufacture_id>/update/<int:programm_id>/  | Обновляет<br>значение м-м<br>программы в<br>заказе<br><br>Доступно только<br>авторизованным<br>пользователям | {<br>"cook_id": int,<br>"product_id": int,<br>"value": int<br>}                           | {<br>"id": int,<br>"name": char(100),<br>"description": char(500),<br>"price": int,<br>"status": int,<br>"image": char(100),<br>"value": int<br>}  |
| 4.1.1<br>6 | POST   | api/users/register/  | Регистрирует<br>пользователя<br><br>Доступно только<br>гостям  | {<br>"username":<br>char(100),<br>"email":<br>char(100),<br>"password":<br>char(100)<br>} | {<br>"user_id": int,<br>"username": char(100),<br>"password": char(100)<br>}   |

|            |      |                               |   |   |  |
|------------|------|-------------------------------|---|---|--|
| 4.1.1<br>7 | PUT  | api/users/<br>update_profile/ | Обновляет<br>данные<br>пользователя<br>Доступно только<br>авторизованным<br>пользователям | {<br>“user_id”: int,<br>“email”:<br>char(100),<br>“username”:<br>char(100),<br>“password”:<br>char(100),<br>} | {<br>“user_id”: int,<br>“username”: char(100),<br>“username”: char(100),<br>“password”: char(100)<br>} |
| 4.1.1<br>8 | POST | /api/users<br>/login/         | Аутентификация<br>Доступно всем   | {<br>“user_id”: int,<br>“username”:<br>char(100),<br>“password”:<br>char(100)<br>}                            | {<br>“user_id”: int,<br>“username”: char(100),<br>“username”: char(100),<br>“password”: char(100)<br>} |
| 4.1.1<br>9 | POST | api/users/<br>logout/         | Деавторизация<br>Доступно только<br>авторизованным<br>пользователям                       |   | 200 / 403  |