|  |  |
| --- | --- |
|  | **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»**

Дисциплина «Разработка интернет-приложений»

Расчетно-пояснительная записка

Тема: «Фаблаб»

Студент: Щепетов Д.О.

Группа ИУ5-52Б

Преподаватель: Канев А.И.

2023г.

# **ВВЕДЕНИЕ**

В современном мире концепция "Фаблаб" становится все более важной и актуальной, предоставляя уникальные возможности для инноваций и творчества. В 2024 году мы вступаем в эру, где доступ к передовым технологиям и оборудованию становится ключевым моментом для реализации креативных идей.

Система предназначена для использования в лабораториях Фаблаба, где работают участники и сотрудники. Она предоставляет им автоматизированные инструменты для создания, управления и мониторинга заказов на печать микроэлектроники. Для того чтобы начать работу над проектом, участникам Фаблаба нужно будет выбрать необходимые виды работ. Система также предоставит возможность сотрудникам редактировать и управлять текущими заказами, обеспечивая эффективную координацию усилий внутри Фаблаба. Внедрение этой системы планируется в ближайшие месяцы, чтобы обеспечить максимальную поддержку творческому потенциалу участников Фаблаба и создать условия для успешной реализации их идей.

Целью работы является реализация системы для создания заявок на печать микроэлектроники, включающую в себя веб-сервис, веб-приложение, десктопное приложение и выделенный сервис расчёта времени синтеза.

Нефункциональные требования к разрабатываемой системе:

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

В ходе работы необходимо выполнить следующие задачи:

* 1. Разработать дизайн приложения.
  2. Создать базу данных в PostgreSQL.
  3. Создать веб-сервис на технологии Django REST Framework.
  4. Реализовать интерфейс гостя на технологии React.
  5. Задеплоить на Github Pages.
  6. Добавить авторизацию и аутентификацию в веб-сервис.
  7. Реализовать интерфейс участника в React.
  8. Реализовать интерфейс сотрудника в React.
  9. Создать десктопное приложение на React Native.
  10. Создать выделенный сервис для результатов завершения работ на Go.
  11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

# **БИЗНЕС-ПРОЦЕСС**

В окружении Фаблаба, где участники хотят проводить различные этапы печати микроэлектроники, существует определенное ограничений, которые нужны для обеспечения безопасности процесса. Для избегания необходимости согласования каждого этапа печати через лабораторное управление [1], участники могут подавать заявки на проведение печати, состоящего из конкретных видов работ. Заказ может включать в себя несколько работ, что позволяет отследить сразу весь процесс печати.

Когда участник определился с перечнем нужных для его печати работ, он формирует заказ, и он отправляется на модерирование. После этого он уже не может её редактировать. Можно также посмотреть историю своих печатей. Проверив печать, сотрудники отклоняют и принимают заказы таким образом, чтобы проходили только безопасные и правильно составленные печати. На случай если у участника изменятся планы, есть опция отмены заказа.

Работы представляют собой реальные процессы производства и обработки микроэлектроники, такие как литография, травление, прессование. Они различаются по характеристикам, таким как классам и формулам. Все характеристики работы можно посмотреть по её текстовому описанию. На какое-то время печать определённых этапов работы может быть запрещена (например, когда материала не осталось на складе) [1], и на это случай сотрудник может скрыть этап работы (поменять его статус на «Удалён»). В течении времени в лаборатории могут поставлять новые материалы и микроэлектронику [2], информация о работах тоже может изменяться [2]. Поэтому у сотрудников есть опция изменения существующих работ и добавления новых. Функции пользователей с разными ролями представлены на диаграммах прецендентов (рис. 1).

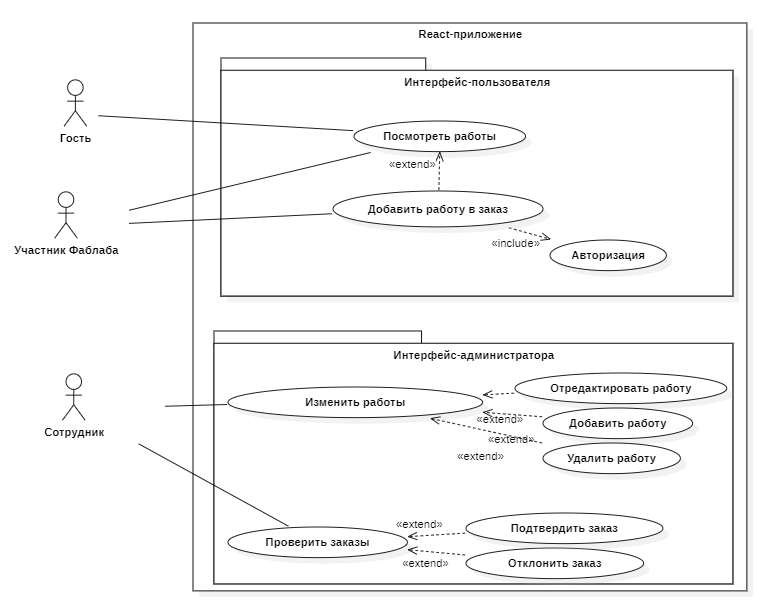


Рисунок 1 - Диаграмма прецедентов

Гостям доступен просмотр работ. Зарегистрированные гости – участники фаблаба. Они могут добавлять работы в заказ, просматривать список своих заказов и сформировывать текущий заказ. Заказы обрабатываются сотрудниками. В результате обработки заказа его либо одобряют, либо отклоняют. Сотрудникам также доступны уникальные функции для проведения работ, а именно: просмотр всех работ, редактирование, создание и удаление работ, а также просмотр списка всех работ в табличном виде. Процесс оформления заказа отражен на диаграмме деятельности (рис. 2).

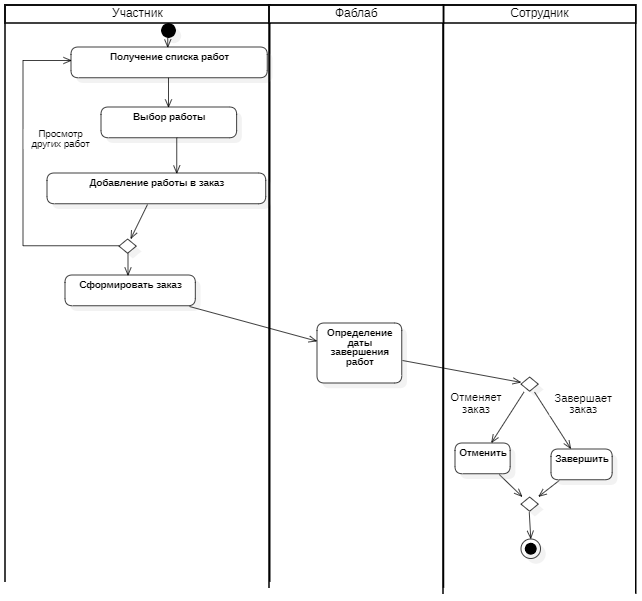


Рисунок 2 - Диаграмма деятельности

Участник выбирает работы, затем формирует на основе выбранных видов работ заказ. Этот заказ затем обрабатывает асинхронный сервис, а затем и сотрудник. Для заказа запускается запрос результата печати микроэлектроники на асинхронном сервисе. Возможные состояния заявки отражены на диаграмме состояний (рис. 3).

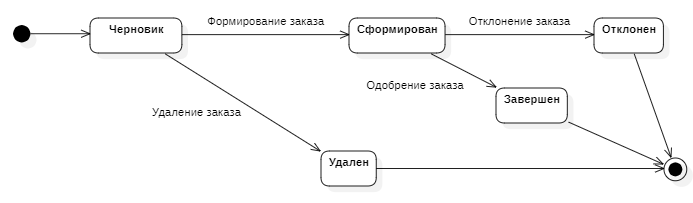


Рисунок 3 - Диаграмма состояний заявки

При выборе первого вида работы формируется черновик. Последующие выбранные работы добавляются в этот черновик. Участник затем формирует заказ, удаляет его или выходит из приложения. Сформированный заказ обрабатывает сотрудник. Он может одобрить или отклонить её.

# **АРХИТЕКТУРА**

Архитектура системы отображена на диаграмме развертывания (рис. 4). Бекенд разворачивается на веб сервисе на порту 8000. Сервис calc\_date\_end [3] разворачивается отдельно и необязательно на том же компьютере, что и основной сервис. Также веб сервис обращается к S3 хранилищу статических файлов Minio и к резидентной базе данных Redis.[4]. Нативное [5] и браузерное приложения [6, 7] обращаются к веб-сервису на базе технологии фреймворка Django [8, 9] через REST Web-API.

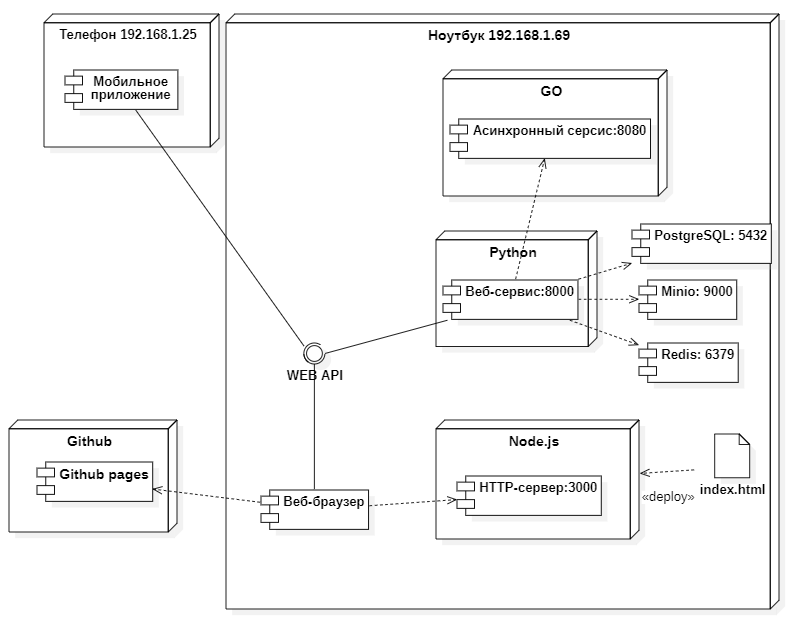


Рисунок 4 - Диаграмма развертывания

Данные хранятся в СУБД PostgreSQL [10], их структура отражена на ER диаграмме (рис. 5). СУБД PostgreSQL является одним из стандартов индустрии, поэтому было решено использовать её. Структура данных довольна проста. Помимо базовых полей, пользователь также имеет поле calc\_date\_end, которое отражает даты формирования заказов. Модель работ представляет собой набор полей, необходимых исключительно для бизнес-логики. Для хранения в одном заказе нескольких работ используется промежуточная таблица Работы-Заказы, реализующая связь М-М.

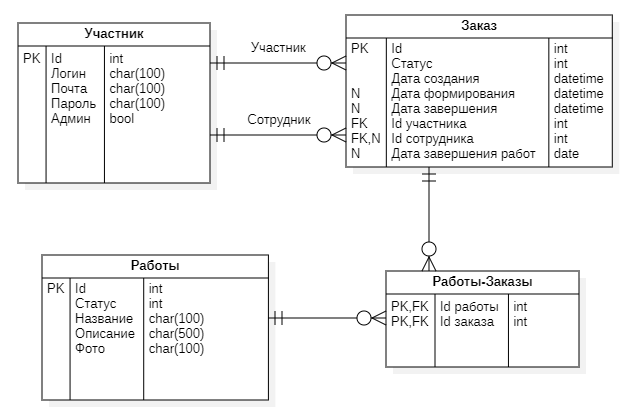


Рисунок 5 - ER диаграмма

Устройство бекенда приложения изображено на диаграмме классов бекенда (рис 6.). Модели имеют связи с соответствующими таблицами в базе данных. Также некоторые модели имеют связи с внешними сервисами. В частности, работы имеют связь с сервером статических файлов, так как в работах хранится ссылка на их изображение, хранимое на сервере статических файлов.

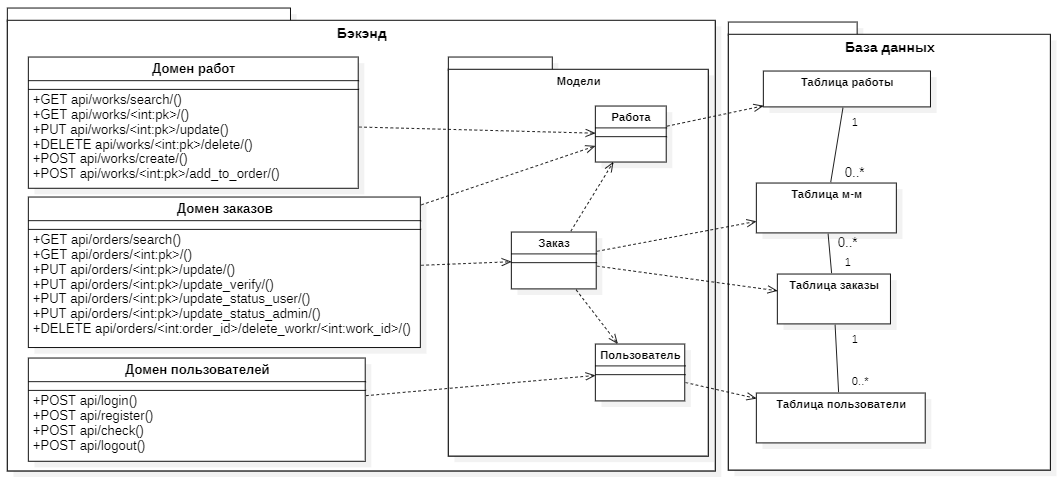


Рисунок 6 *-* Диаграмма классов бекенда

Связь фронтенда и бекенда отражена на диаграмме классов фронтенда (рис. 7). Ключевые страницы имеют связь с API аутентификации, так как доступ к ним осуществляется только для авторизированных пользователей с определенными правами (ролями).

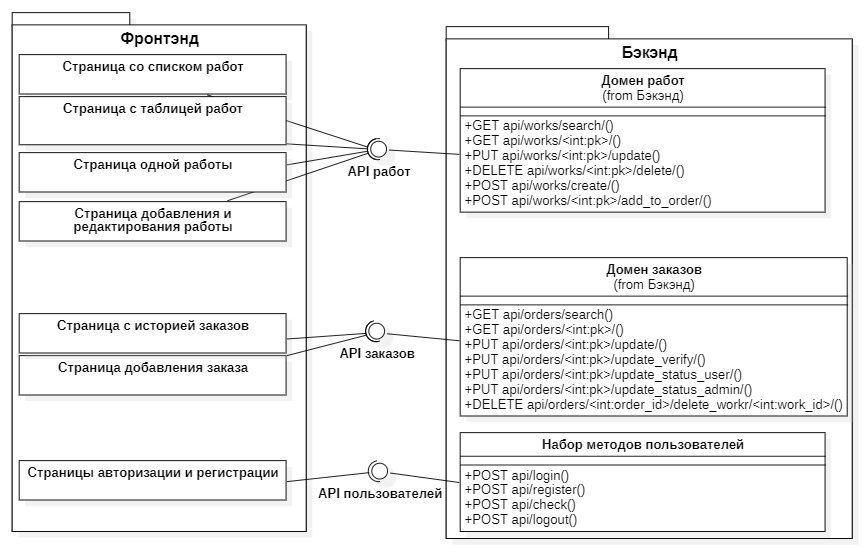


Рисунок 7 *-* Диаграмма классов фронтенда

# **АЛГОРИТМЫ**

Алгоритм работы системы отображен на диаграмме последовательности (рис. 8). В основе системы лежит веб-сервис, реализующий внутри себя всю бизнес-логику. Он предоставляет доступ к методам из следующих доменов: карты, заявки, пользователи и аутентификация. Методы следуют правилам REST API.

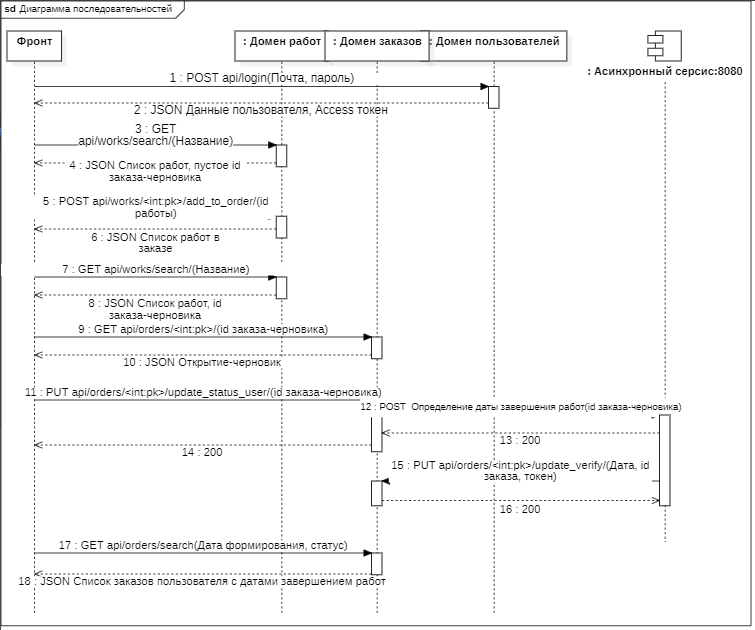


Рисунок 8 - Диаграмма последовательности

В начале бизнес-процесса происходит аутентификация участника. Для этого он отправляет через графический интерфейс запрос, передавая в нем логин и пароль. Если аккаунт с такими данными существует, то участник получает JWT в ответном запросе. Если же такого аккаунта не существует, или пароль введен неверно, участник получит ошибку. В таком случае ему надо либо пройти регистрацию, либо ввести пароль верно. Затем графический интерфейс участника запрашивает у веб-сервиса список работ, которые возвращаются в JSON формате. Участник выбирает работу, которую хочет использовать в печати, и, нажимая на кнопку «добавить» в графическом интерфейсе, отправляет запрос на добавление работы в свой черновой заказ. Этот процесс может продолжаться несколько раз. На странице чернового заказа участник может менять работы в графическом интерфейсе.

Когда участник определится с выбором, он нажимает на кнопку «сформировать» в графическом интерфейсе. После этого приложение запрашивает id чернового заказа участника и затем отправляет запрос на формирование этого заказа. После этого основной веб-сервис выполняет асинхронный запрос к сервису calc\_date\_end на то, чтобы он запросил результат печати микроэлектроники для данного заказа. Участник фаблаба сможет увидеть через некоторое время, что его заказ прошел или не прошел печать.

Процесс рассмотрения заказов происходит также через графический интерфейс. Сотрудники могут просматривать списки всех заказов и, нажимая на соответствующие кнопки, отправлять запросы на одобрение или отклонение заказов в основной веб-сервис. В эти запросы также можно включить фильтры по имени создателя заявки и диапазону дат, в котором должны были быть созданы заявки. Также через графический интерфейс они могут управлять непосредственно работами. Им доступны такие функции, как создание и редактирование работ, просмотр списка работ и удаление их. Для каждой из этих функция присутствует свой метод, отправляемый на основной веб-сервис.

# **ОПИСАНИЕ ИНТЕРФЕЙСА**

При открытии приложения мы оказываемся на странице с карточками или списком работ. Это первоначальная страница для всех пользователей и гостей. Внешний вид этой страницы зависит от роли пользователя. Для гостей там отображаются карточки работ с возможностью только посмотреть подробную информацию, а в верхнем меню есть возможность войти в аккаунт. (рис. 9).

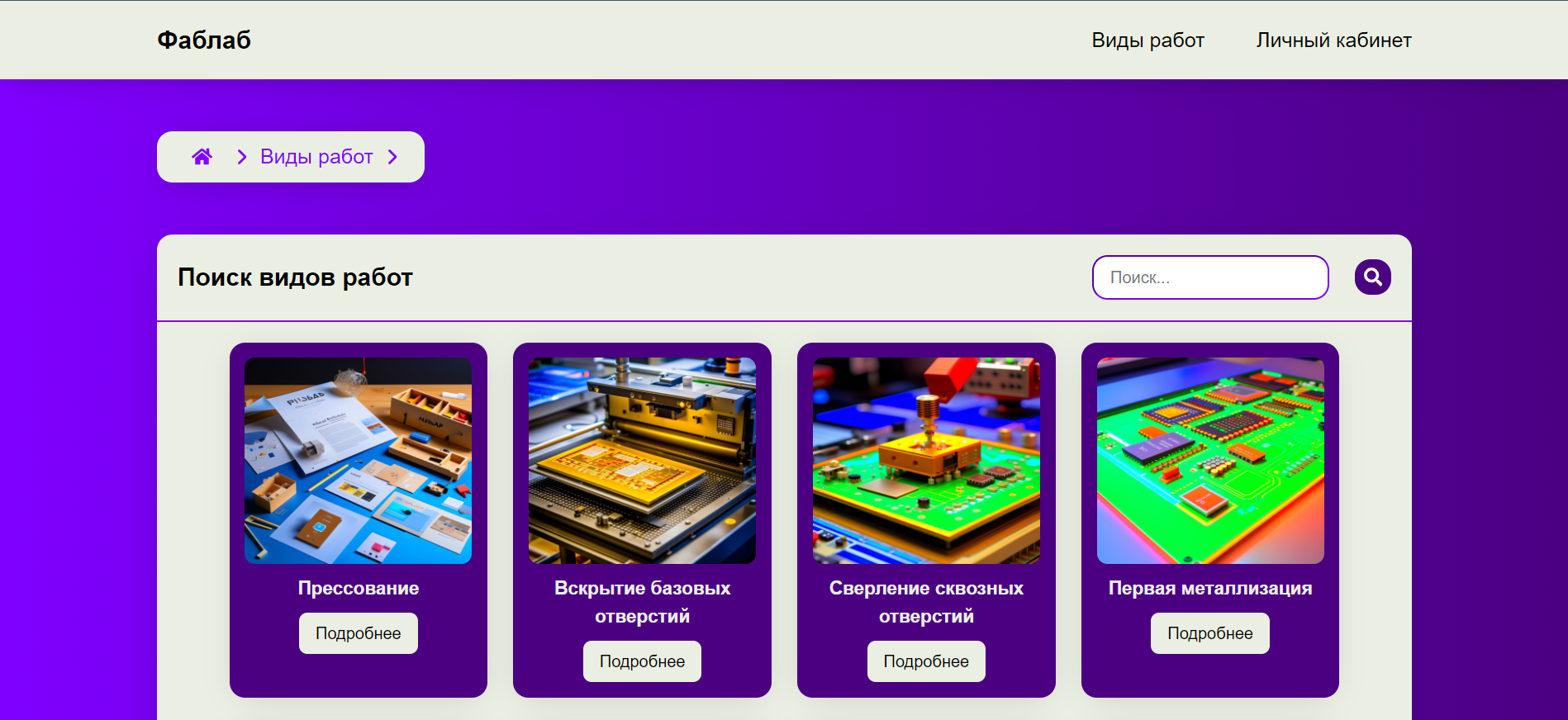
****

Рисунок 9 – Страница работ (неавторизированный пользователь)

У каждой карточки есть кнопка «Подробнее», переносящая пользователя на страницу с подробной информацией о работе (рис. 10). Кнопка «Назад» переносит пользователя обратно на страницу с карточками работ.

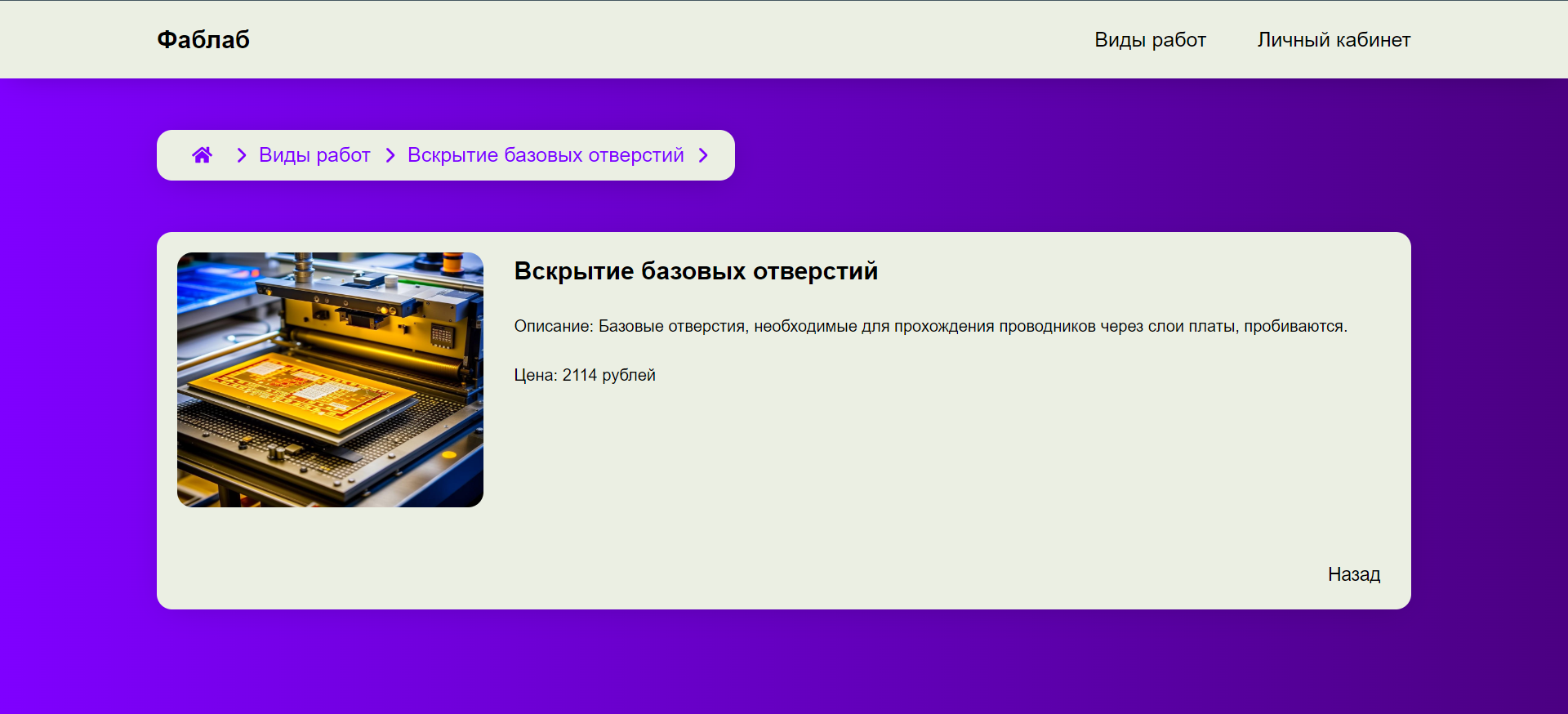
****

Рисунок 10 – Страница работы

На странице с формой авторизации (рис. 11) отображается форма, через которую гость входит в свой аккаунт. При успешном вводе логина и пароля гость получает JWT, который сохраняется в cookies и используется при отправлении запросов. С этой страницы можно перейти на страницу создания аккаунта.

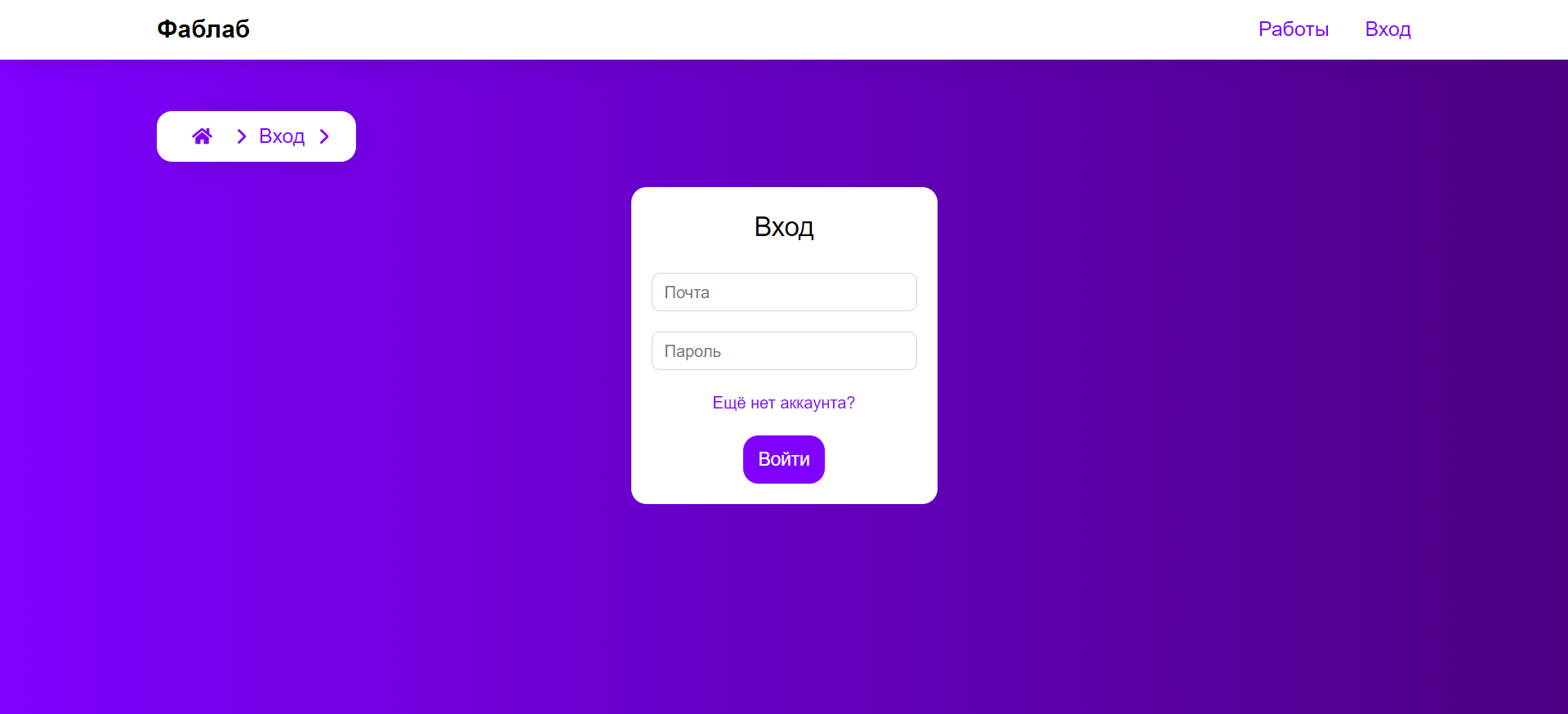
****

Рисунок 11 - Страница авторизации

На странице с формой регистрации (рис. 12) гости могут завести аккаунт. Для этого нужно указать почту, имя и пароль.

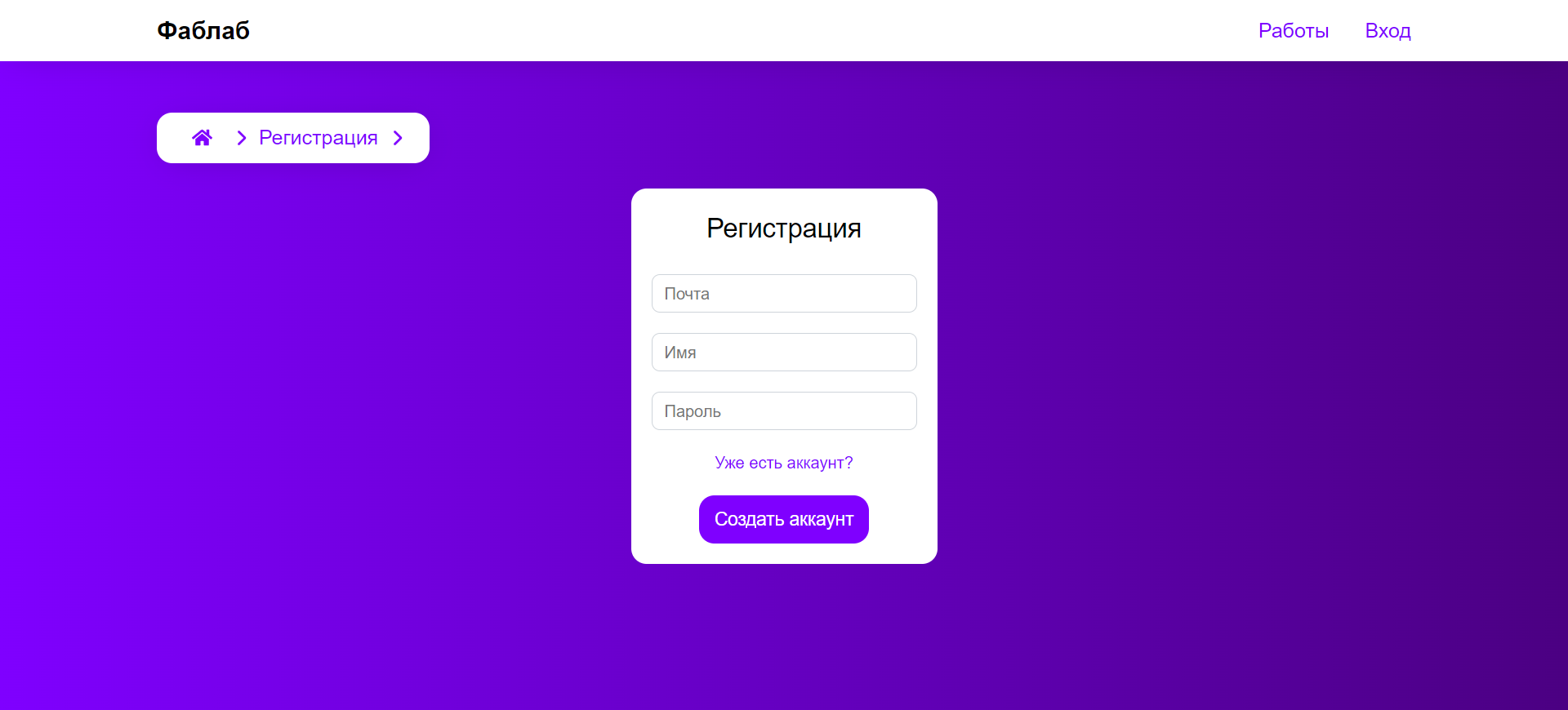
****

Рисунок 12 - Страница регистрации

Для участника фаблаба есть возможность добавить работы в новый заказ, посмотреть черновик заказа, посмотреть список своих заказов или перейти на страницу аккаунта. У каждой карточки появляется кнопка «Добавить»/«Удалить» в зависимости от того выбрал ли участник фаблаба уже эту работу или нет (рис. 13). Сверху есть поле поиска работы, в котором можно ввести часть названия работы.

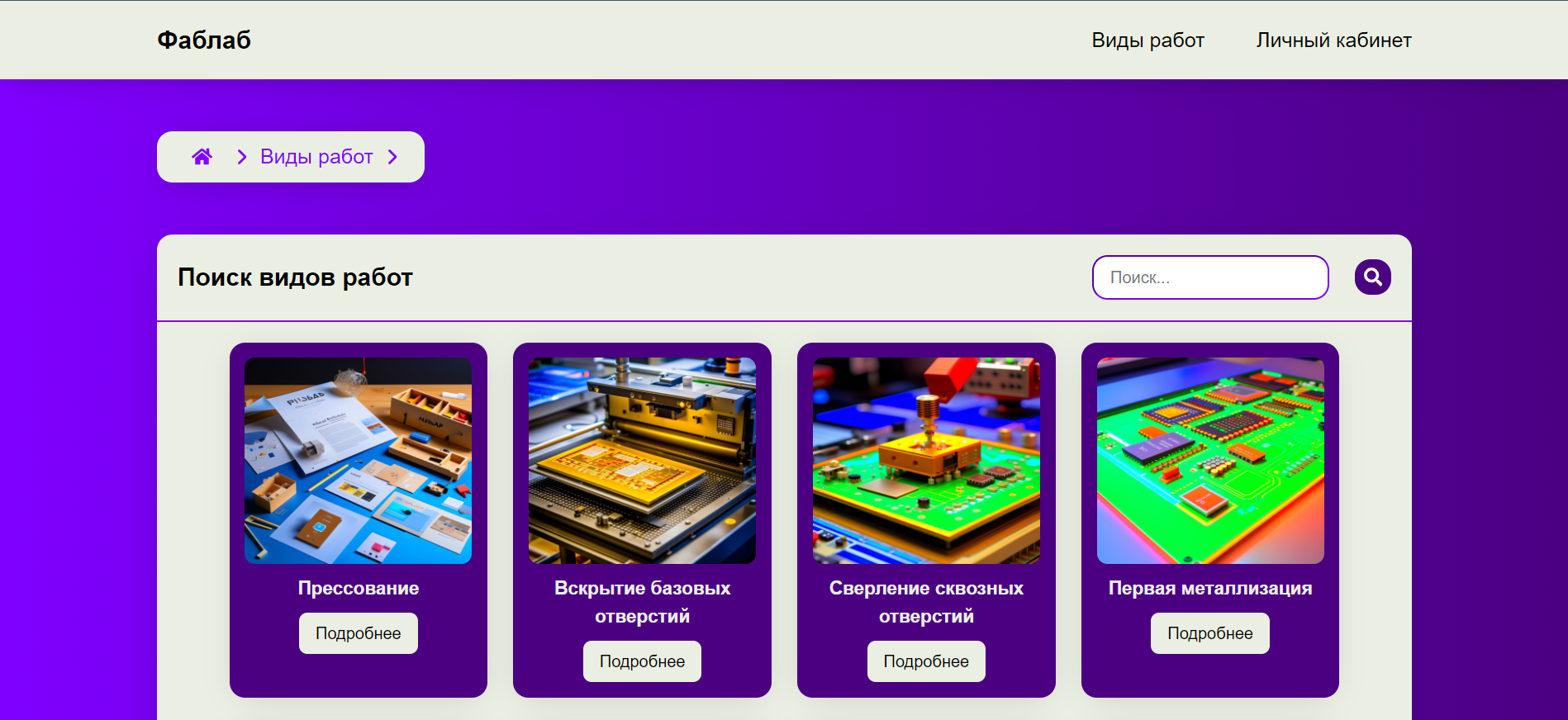
****

Рисунок 13 – Страница работ (участник фаблаба)

Также пользователь (участник фаблаба или сотрудник) может зайти на страницу своего аккаунта (рис. 14). На этой странице отражается информация о пользователе, а также кнопка «Выйти», при нажатии на которую пользователь выйдет из аккаунта и его перенаправит на гостевую страницу со списком работ.



Рисунок 14 – Страница профиля пользователя

Если участник фаблаба на странице со списком работ добавит какую-то работу в заказ, то по кнопке «Новый заказ» он перейдет на страницу с черновым заказом (рис. 15). Там участник может отредактировать название, описание и стоимость работ в составе данного заказа, а также удалить работу из заказа или перейти на страницу с подробной информацией о работе. Если удалить черновик заявки, участника фаблаба вернет на страницу со списком работ.

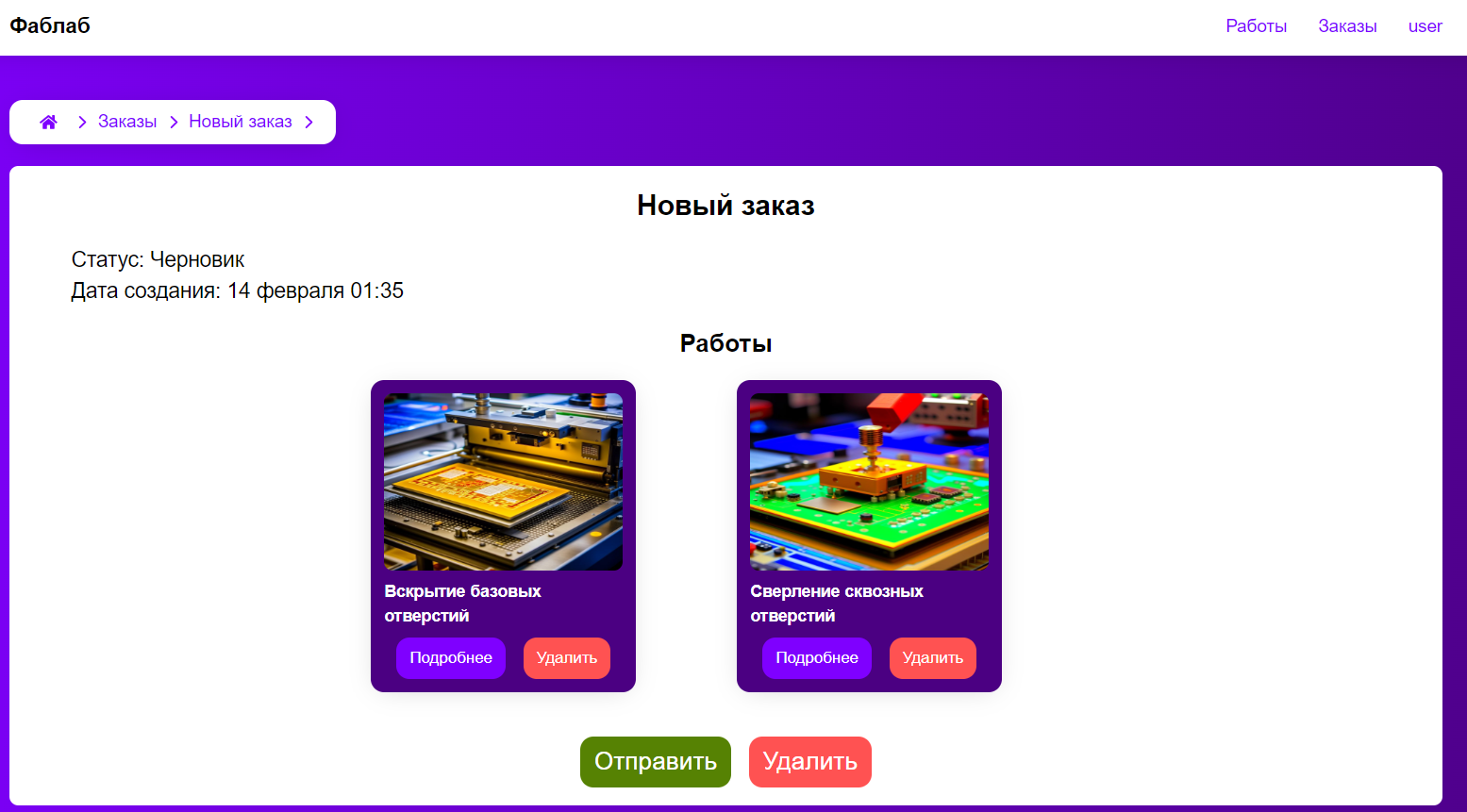


Рисунок 15 - Страница с черновиком заказа

При нажатии на кнопку «Отправить» заказ сформируется и отправится на проверку сотрудникам, а участника фаблаба перенаправит на страницу со списком заказов (рис. 16). Эту страницу также можно открыть, нажав на «Фаблаб» в меню. В зависимости от типа участника этот список будет функционально отличаться. Так, для участников фаблаба отображается список созданных ими заказов: номер, статус, дата формирования, дата завершения и результат печати. Для перехода на страницу печати можно нажать по необходимой строчке. Заказы можно отфильтровать по дате и статусу.

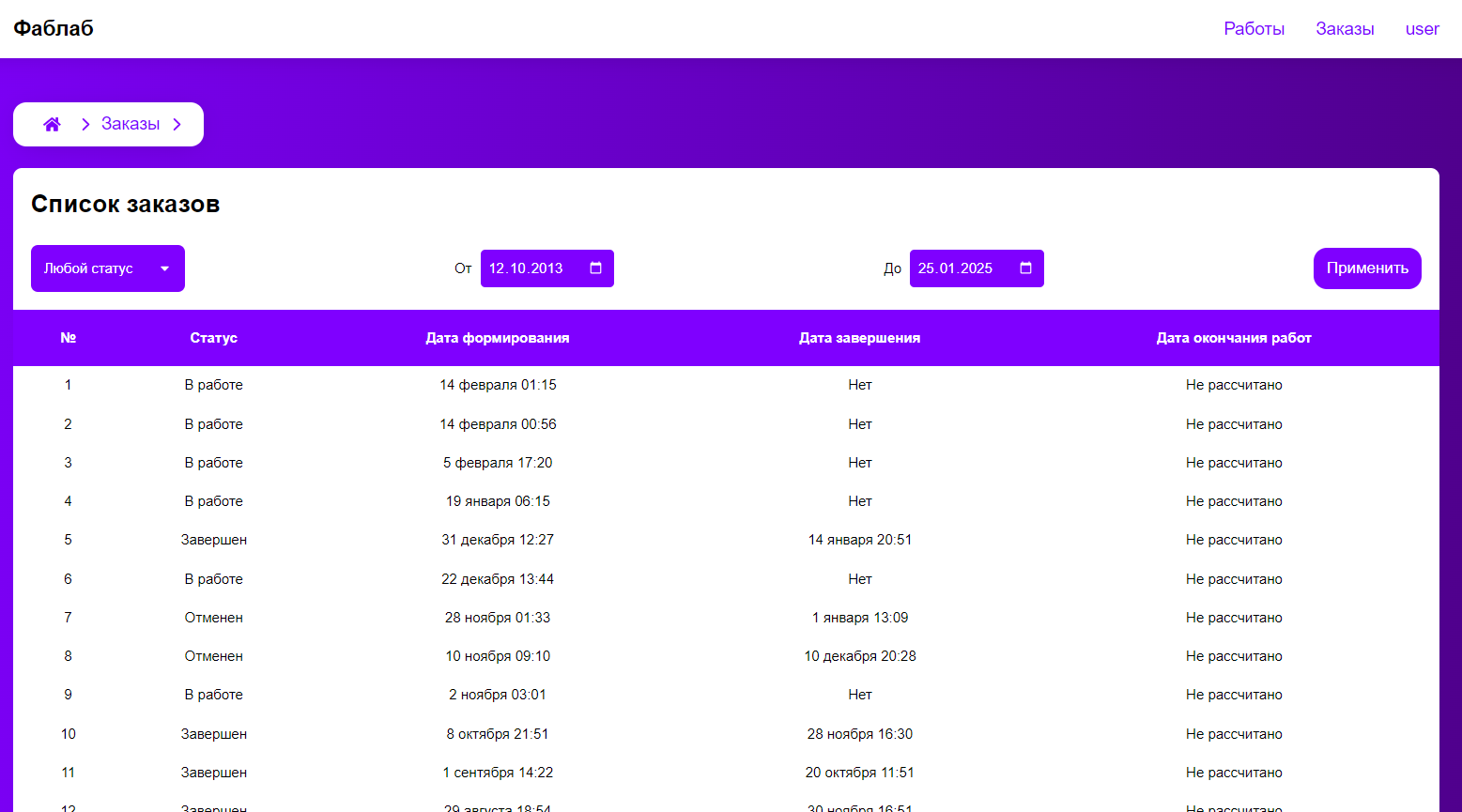


Рисунок 16 - Страница со списком заказов (участник фаблаба)

На странице печати (рис. 17) отображается статус, название, описание, дата создания, дата формирования, дата завершения (при наличии), результат печати, а также карточки входящих в заказ работ.



Рисунок 17 – Страница заказа

Для сотрудников на странице со списком заказов печати отображается информация об участнике, создавшем заказ и кнопки «Принять» и «Отклонить» (рис. 18). Заказы можно отфильтровать по дате, статусу и участникам.

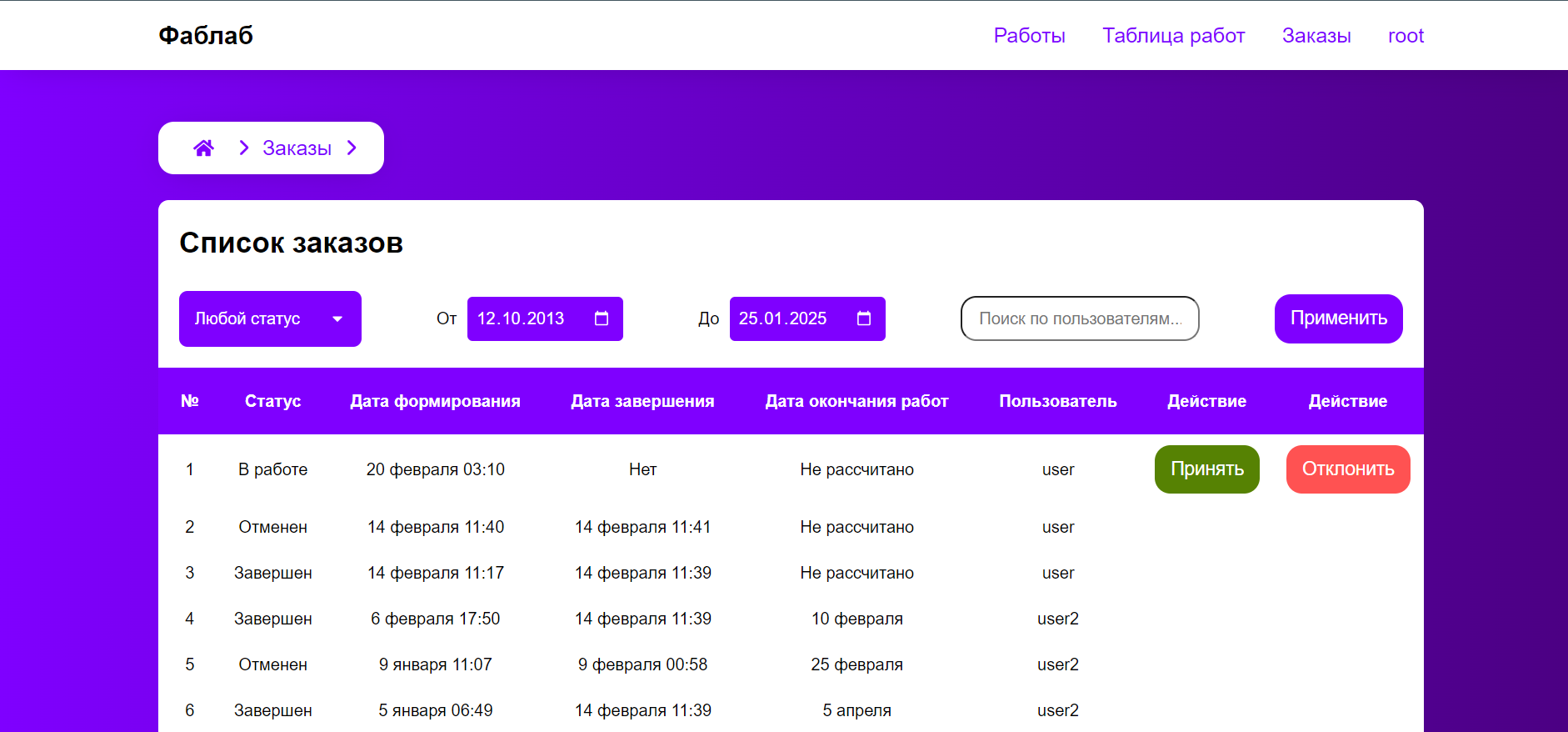


Рисунок 18 - Страница со списком заказов (сотрудник)

На странице с таблицей работ (рис. 19) сотрудник может в компактном и удобном формате просмотреть список всех работ, существующих в системе.

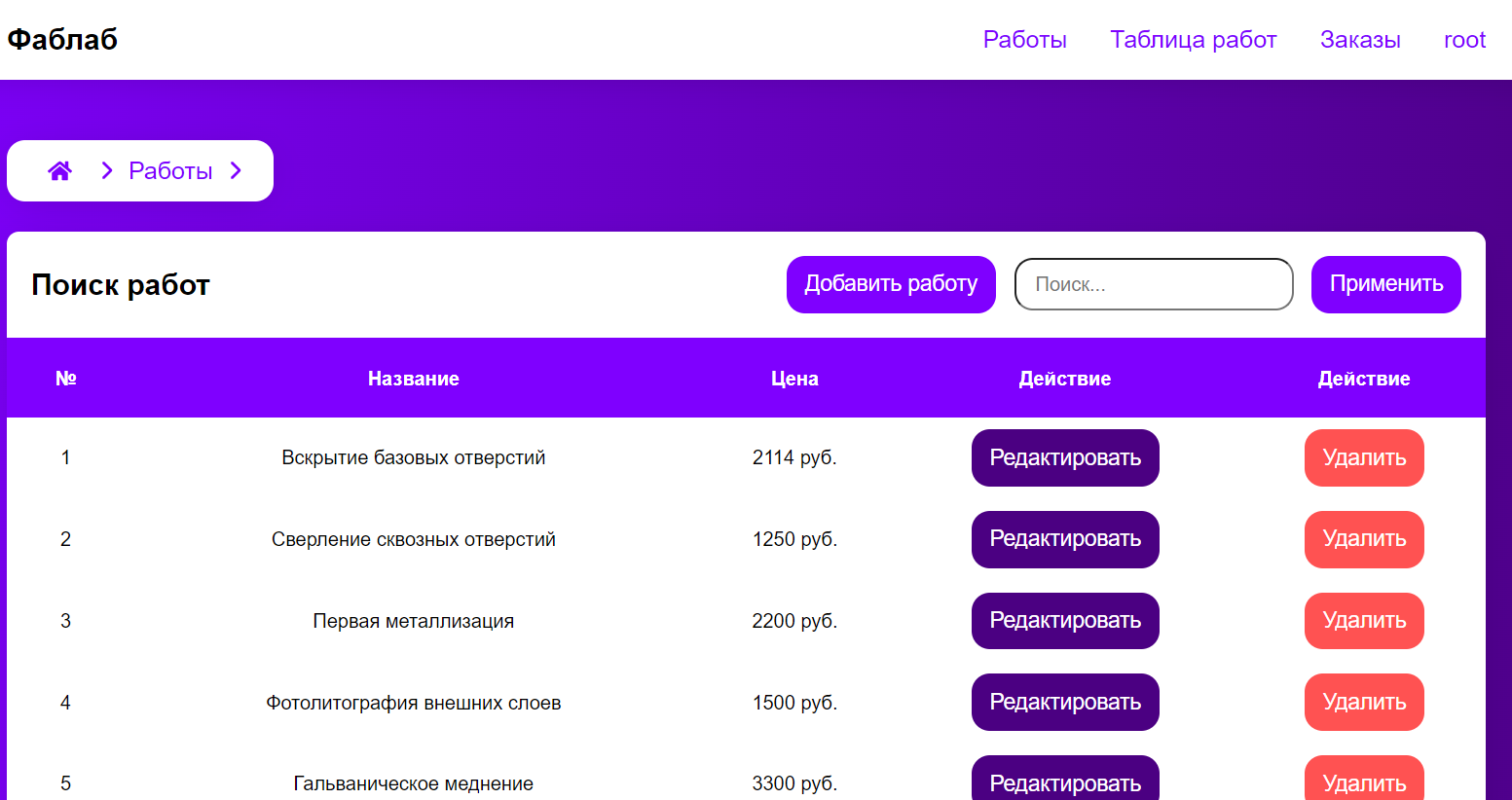


Рисунок 19 – Страница работ (сотрудник)

По нажатию на работу из списка откроется страница редактирования существующей работы, по нажатию на кнопку «Добавить работу» - создание новой (рис. 20, 21). Сотрудник может добавить/изменить название, описание и фотографию работы, а также удалить работу.

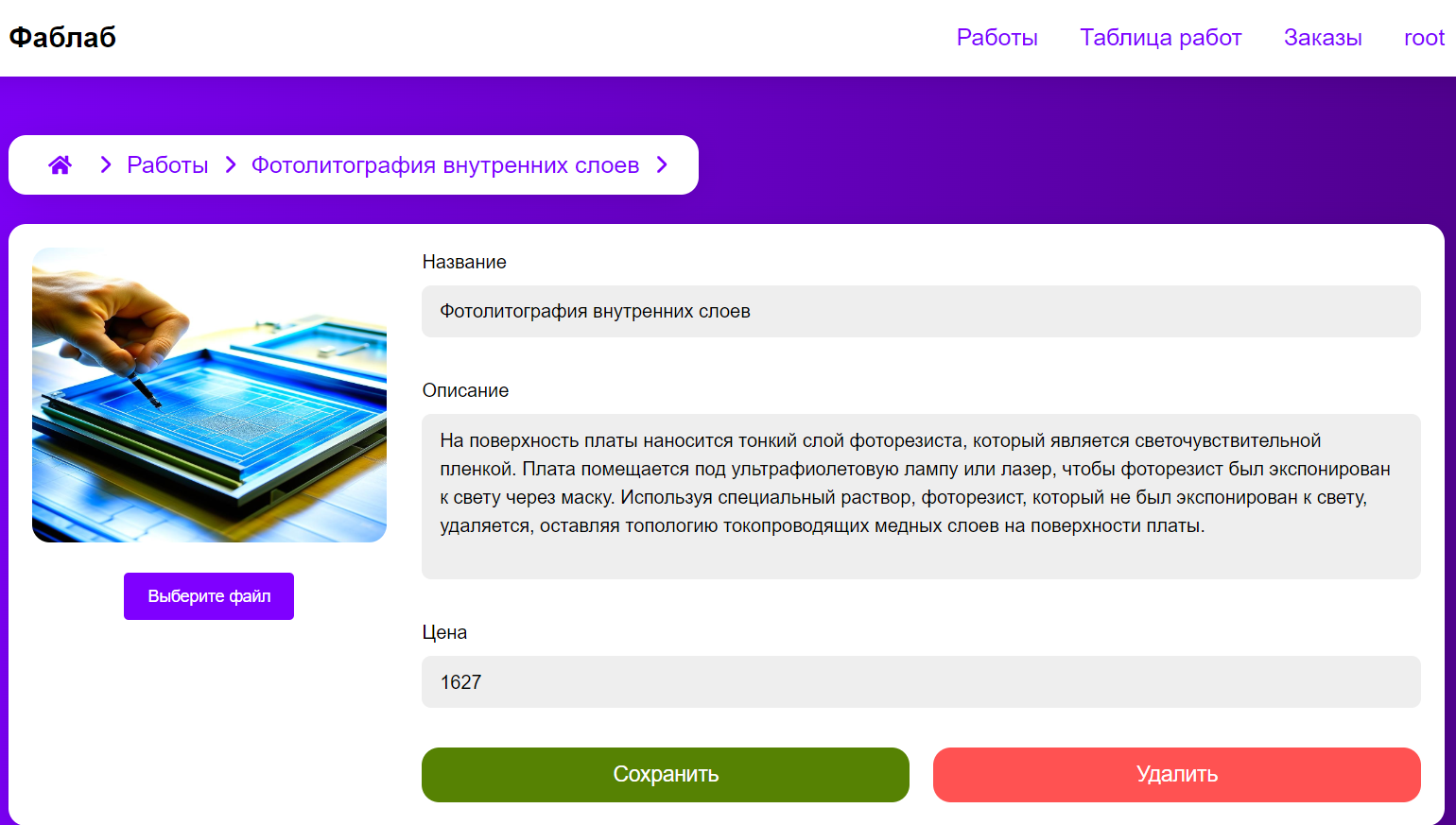


Рисунок 20 - Страница редактирования работы

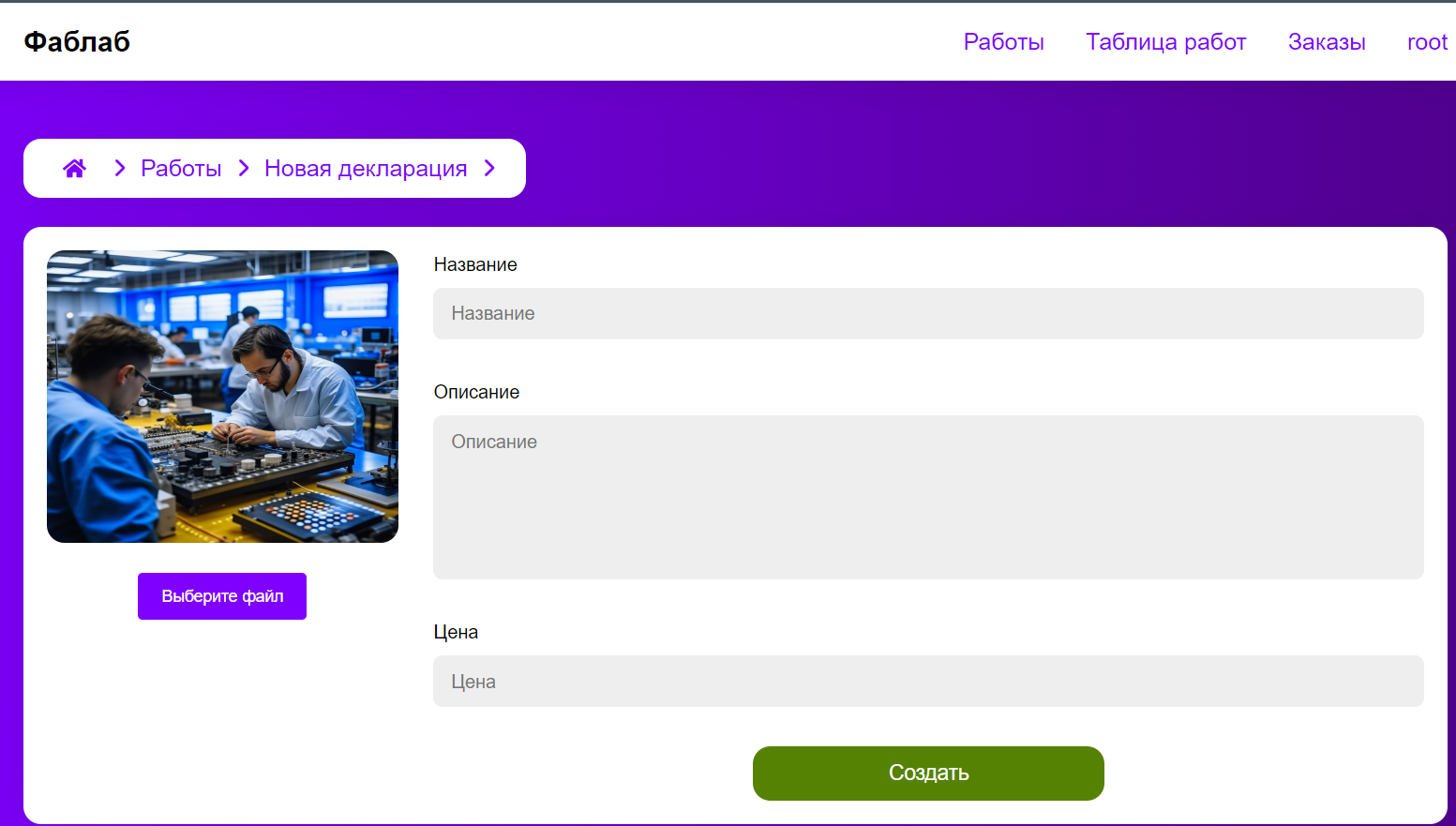


Рисунок 21 - Страница создания новой работы

# **ЗАКЛЮЧЕНИЕ**

В ходе работы были достигнуты следующие результаты:

1. Был разработан дизайн приложения с помощью набора стилей CSS и HTML тегов.
2. База данных была создана и расположена в docker контейнере.
3. Был создан веб-сервис на Django REST Framework, с использованием веб-фреймворка ASP.NET.
4. Разработан интерфейс гостя с использованием технологии React Framework и подключен к веб-сервису.
5. Приложение интерфейса было развернуто на сервисе Github Pages по ссылке [https://sh-dimitrij.github.io/fablab/](https://vk.com/away.php?to=https%3A%2F%2Fsh-dimitrij.github.io%2Ffablab%2F&cc_key=).
6. В веб-сервис добавлена авторизация через JWT, а методы задокументированы через Swagger.
7. Интерфейс участника фаблаба был реализован. Доступ к нему имеют только авторизированные участники.
8. Выделенный сервис был разработан и развернут в отдельном docker кластере.
9. Реализован интерфейс сотрудника для подтверждения новых заказов и редактирования работ.
10. Было реализовано десктопное приложение на React Native, повторяющее интерфейс веб-приложения на React.
11. Подготовлен набор документации, включающий РПЗ, ТЗ и набор UML диаграмм.
12. Исходный код проекта доступен в GitHub https://github.com/sh-dimitrij/fablab-backend.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

* 1. Анализ Фаблаба (Центра технологической поддержки) [Электронный ресурс] // fablab. URL: https://fablab.pro/fab/ (дата обращения 05.01.2024).
  2. Engineering Creativity Studio [Электронный ресурс] // sfvstu. https://sfvstu.ru/fablab (дата обращения 05.01.2024).
  3. Руководство по языку Go [Электронный ресурс] // Metanit. URL: https://metanit.com/go/tutorial/ (дата обращения: 10.12.2023).
  4. Разбираемся с Redis [Электронный ресурс] // Habr. URL: https://habr.com/ru/companies/wunderfund/articles/685894/ (дата обращения: 20.12.2023).
  5. React Native [Электронный ресурс] // React Native. URL: https://reactnative.dev/docs/tutorial (дата обращения: 07.12.2023).
  6. Руководство по React [Электронный ресурс] // Metanit. URL: https://metanit.com/web/react/ (дата обращения: 12.10.2023).
  7. Quick Start – React [Электронный ресурс] // React. URL: https://react.dev/learn (дата обращения: 12.10.2023).
  8. Django введение [Электронный ресурс] // Mozilla. URL: https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction (дата обращения: 04.10.2023).
  9. Руководство по веб-фреймворку Django [Электронный ресурс] // Metanit. URL: https://metanit.com/python/django/?ysclid=lrbtfjzk2n312420810 (дата обращения: 04.10.2023).
  10. Руководство по PostgreSQL [Электронный ресурс] // Metanit. URL: https://metanit.com/sql/postgresql/ (дата обращения: 05.10.2023).

**Приложение. Техническое задание**

|  |  |
| --- | --- |
|  | **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»**

Дисциплина «Разработка интернет-приложений»

Техническое задание

Тема: «Фаблаб»

Студент: Щепетов Д.О.

Группа ИУ5-52Б

Преподаватель: Канев А.И.

2023г.

1. **Цель работы**

Реализовать систему для документирования различных видов проводимых работ, включающую в себя веб-сервис, веб-приложение, десктопное приложение и выделенный сервис для получения результатов печати микроэлектроники.

1. **Назначение**

Система предназначена для участников и работников фаблаба. На сайте предусмотрена проверка заказов. Для регистрации заказа участнику фаблаба необходимо создать заявку с составом выбранных видов проводимых работ. Система предоставляет автоматизированный способ создания, учета и ведения заявок. Также она позволяет сотрудникам принимать или отклонять заявки, а также добавлять новые виды проводимых с микроэлектроникой работ.

1. **Задачи**:
   1. Разработать дизайн приложения.
   2. Создать базу данных в PostgreSQL.
   3. Создать веб-сервис на технологии Django REST Framework.
   4. Реализовать интерфейс гостя на технологии React.
   5. Развернуть веб-приложение React на Github Pages.
   6. Добавить авторизацию и аутентификацию в веб-сервис.
   7. Реализовать интерфейс участника в React.
   8. Реализовать интерфейс сотрудника в React.
   9. Создать десктопное приложение на React Native.
   10. Создать выделенный сервис для результатов клинических испытаний на Go.
   11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.
2. **Методы веб-сервиса:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Метод** | **Описание** | **Url** | **Входные данные** | **Выходные данные** |
| * 1. Методы авторизации и аутентификации | | | | | |
| 4.1.1 | POST | Регистрирует нового участника в базе данных  Если в системе уже существует участник с такой почтой, то возвращает ошибку 409  В случае успеха возвращает статус 201, id и access\_token созданного участника  Доступен всем участникам | api/register | {  “email”: char(100),  “password”: char(100)  } | {  “user\_id”: int,  “access\_token”: char(100)  } |
| 4.1.2 | POST | Выполняет аутентификацию пользователя (участника фаблаба или сотрудника) по логину и паролю  В случае отправки невалидных данных возвращает ошибку 401  В случае успеха возвращает статус 200, а также данные о пользователе | api/login | {  “email”: char(100),  “password”: char(100)  } | {  “user\_id”: int,  “email”: char(100),  “is\_moderator”: bool,  “access\_token”: char(100)  } |
| 4.1.3 | POST | Выполняет аутентификацию пользователя по токену  В случае отправки невалидного токена возвращает ошибку 401  В случае успеха возвращает статус 200, а также данные о пользователе | api/check | {  “access\_token”: char(100)  } | {  “user\_id”: int,  “email”: char(100),  “is\_moderator”: bool,  “access\_token”: char(100)  } |
| 4.1.4 | POST | Добавляет его токен в черный список  Доступно только авторизированным пользователям | api/logout | {  “access\_token”: char(100)  } | Статус 200 / 401 |
| * 1. Методы работ | | | | | |
| 4.2.1. | GET | Возвращает картинку работы. | api/works/<int:work\_id>/image / | {  “work\_id”: int  } | Файл |
| 4.2.2. | PUT | Добавляет в заказ участника работу <work\_id>, возвращает список всех работ в заказе.  Доступно только авторизированным участнкиам | api/works/<int:work\_id>/add\_to\_order/ | {  “work\_id”: int,  “access\_token”: char(100)  } | [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  },  …  ] |
| 4.2.3. | GET | Возвращает список работ, удовлетворяющих переданным критериям и черновой заказ участника.  Будут возвращены работы, отфильтрованные по статусу «действует», а также id чернового заказа участника фаблаба если он есть | api/works/search | {  “query”: char(100),  “access\_token”: char(100)  } | {  “order\_id”: int,  “works”: [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  },  …  ]  } |
| 4.2.4. | GET | Возвращает работу | api/works/<work\_id>/ | {  “work\_id”: int  } | {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  } |
| 4.2.5. | POST | Создает работу  Доступно только сотрудникам | api/works/create/ | {  “access\_token”: char(100)  } | [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  },  …  ] |
| 4.2.6. | PUT | Обновляет информацию о работе  Доступно только сотрудникам | api/works/<work\_id>/update/ | {  “work\_id”: int,  “access\_token”: char(100)  } | {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  } |
| 4.2.7. | DELETE | Помечает работу как удаленное  Доступно только сотруднкиам | api/works/<work\_id> /delete/ | {  “work\_id”: int,  “access\_token”: char(100)  } | [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  },  …  ] |
| 4.2.8. | PUT | Обновляет картинку работы.  Возвращает обновленную картинку работы. | api/works/<work\_id> /update\_image / | {  “work\_id”: int,  “image”: char(100)  } | Файл |
| * 1. Методы заказов | | | | | |
| 4.3.1. | GET | Возвращает список заказов участника  Доступно только авторизированным участникам | api/orders/search/ | {  “status”: int,  “date\_start”: long,  “date\_end”: long  “access\_token”: char(100)  } | [  {  "id": int,  "date": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": " datetime  "owner": {  “name”: char(100),  },  “moderator”: {  “name”: char(100),  },  "works": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  },  …  ]  },  …  ] |
| 4.3.2. | GET | Возвращает подробную информацию о заказе.  Доступно только авторизированным участникам | api/orders/<order\_id>/ | {  “order\_id”: int,  “access\_token”: char(100)  } | {  "id": int,  "date": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": " datetime  "owner": {  “name”: char(100),  },  “moderator”: {  “name”: char(100),  },  "works": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  },  …  ]  } |
| 4.3.3. | PUT | Обновляет информацию о заказе  Возвращает обновленный заказ  Доступно только авторизированным участникам | api/orders/<order\_id>/update/ | {  “order\_id”: int,  “access\_token”: char(100)  } | {  "id": int,  "date": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": " datetime  "owner": {  “name”: char(100),},  “moderator”: {  “name”: char(100),  },  "works": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  },  …  ]  } |
| 4.3.4. | DELETE | Удаляет из заказа участника работу.  Доступно только авторизированным участникам | api/orders/<order\_id>/delete\_work/<work\_id>/ | {  “order\_id”: int,  “work\_id”: int,  “access\_token”: char(100)  } | {  "works": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  },  …  ]  } |
| 4.3.5. | PUT | Устанавливает заказу статус «в работе»  Только если его текущий статус «черновик»  Доступно только авторизированным участникам | api/orders/<order\_id>/update\_status\_user/ | {  “order\_id”: int,  “access\_token”: char(100)  } | {  "id": int,  "date": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": " datetime  "owner": {  “name”: char(100),  },  “moderator”: {  “name”: char(100),  },  "works": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  },  …  ]  } |
| 4.3.6. | DELETE | Устанавливает заказу статус «удален»  Только если его текущий статус «черновик»  Доступно только авторизированным участникам | api/orders/<order\_id>/delete/ | {  “order\_id”: int,  “access\_token”: char(100)  } | Статус 200 / 404 / 405 |
| 4.3.7. | PUT | Устанавливает заказу статус «отклонен» или «завершен»  Только если его текущий статус «в работе»  Доступно только сотрудникам | api/orders/<order\_id>/update\_status\_admin/ | {  “order\_id”: int,  “access\_token”: char(100)  } | {  "id": int,  "date": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": " datetime  "owner": {  “name”: char(100),  },  “moderator”: {  “name”: char(100),  },  "works": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  },  … |
| 4.3.8. | PUT | Обновляет результат печати микроэлектроники заказа  Доступно с ключом доступа | api/orders/<int:order\_id>/update\_date\_end/ | {  “order\_id”: char(100),  “access\_key”: char(100)  } | Статус 200 / 403 |
| * 1. Методы внешнего сервиса clinical\_trial | | | | | |
| 4.4.1. | POST | Определяет результат печати микроэлектроники заказа. Метод внешнего сервиса clinical\_trial | calc\_date\_end/ | {  “order\_id”: int,  } | Статус 200 / 403 |

1. **Функциональные требования:**
   1. Главное меню
      1. Виды работ – переход на страницу со списком / таблицей видов работ
         1. Доступно всем пользователям
         2. Для участников фаблаба выполняется переход на страницу со списком работ (5.4)
         3. Для сотрудников выполняется переход на страницу с таблицей работ (5.5)
      2. Заказы работ – переход на страницу с заказами работ (5.8)
         1. Доступно только авторизированным пользователям.
      3. Вход – переход на страницу с формой авторизации (5.2)
         1. Доступно только неавторизированным пользователям.
      4. Выход
         1. Доступно только авторизированным пользователям.
         2. При нажатии отправляется запрос на API (4.1.4), который добавляет токен пользователя в черный список. Также токен удаляется из cookies, обнуляется состояние пользователя в Redux и происходит переадресация на страницу со списком проводимых работ (5.4)
   2. Страница с формой авторизации
      1. Доступна только неавторизованным пользователям
      2. Содержит
         1. Тестовое поле для ввода почты
         2. Текстовое поле для ввода пароля
         3. Кнопку Войти
         4. Кнопку Уже зарегистрированы?
      3. Действия
         1. При нажатии на кнопку Войти отправляется запрос на API (4.1.2), в котором передаются введенные данные формы. Если введенные данные верны, то в ответе возвращается access токен, который сохраняется в cookies, а также происходит переадресация на страницу со списком проводимых работ (5.4)
         2. При нажатии на кнопку Уже зарегистрированы? происходит перенаправление пользователя на страницу регистрации (5.3)
   3. Страница с формой регистрации
      1. Доступна только неавторизованным пользователям
      2. Содержит
         1. Тестовое поле для ввода почты
         2. Текстовое поле для ввода пароля
         3. Кнопку Зарегистрироваться
         4. Кнопку Уже есть аккаунт?
      3. Действия
         1. При нажатии на кнопку Зарегистрироваться отправляется запрос на API (4.1.1), в котором передаются введенные данные формы. Если введенные данные верны, то в ответе возвращается access токен, который сохраняется в куки, а также происходит переадресация на страницу со списком проводимых работ (5.4)
         2. При нажатии на кнопку Уже есть аккаунт? перенаправляет пользователя на страницу авторизации (5.2)
   4. Страница со списком проводимых работ
      1. Доступна всем пользователям
      2. Содержит
         1. Кнопку «Новый заказ»
            1. Доступна только авторизированным участникам
         2. Текстовое поле для поиска работ по названию
         3. Выводятся работы в виде карточек (метод 4.2.3). Каждая карточка состоит из:
            1. Фото работы
            2. Названии работы
            3. Кнопка «Подробнее»
            4. Кнопка «Добавить»

Доступна только авторизированным участникам

* + 1. Действия
       1. При нажатии на кнопку «Подробнее» (5.4.2.3.3) происходит перенаправление пользователя на страницу с подробным описанием работы (5.6)
       2. При нажатии на кнопку «Добавить» (5.4.2.3.4) отправляется запрос на API (4.1.6), а также меняется состояние кнопки «Новый заказ» (5.4.2.1)
       3. При нажатии на кнопку «Новый заказ» (5.4.2.1) происходит перенаправление пользователя на страницу с черновым заказом работ (5.7), если он существует у пользователя
       4. При каждом наборе символов в поле поиска (5.4.2.2) и нажатия на кнопку поиска отправляется запрос на API (4.2.3), который возвращает список работ, отфильтрованный по названию
  1. Страница с таблицей работ
     1. Доступна только сотрудникам
     2. Содержит
        1. Текстовое поле для поиска работы по названию
        2. Выводит таблицу с работы, содержащую столбцы (метод 4.2.3):
           1. № работы
           2. Название работы
     3. Действия
        1. При нажатии на любую строку таблицы (5.2.2.2) происходит перенаправление пользователя на страницу редактирования работы (5.9)
        2. При наборе символов в поле поиска (5.5.2.1) и нажатия на кнопку поиска отправляется запрос на API (4.2.3), который фильтрует таблицу работ по названию
  2. Страница с подробным описанием работы
     1. Доступна всем пользователям
     2. Выводится информация о работе (метод 4.2.4):
        1. Название работы
        2. Фото работы
        3. Описание работы
  3. Страница чернового заказа работ
     1. Доступна только авторизованным участникам
     2. Выводится информация о заказе работ (метод 4.3.2):
        1. Карточки работ в заказе работ. Каждая карточка содержит:
           1. Название работы
           2. Дата завершения заказа
           3. Кнопку «Удалить»
        2. Текстового поля с названием заказа работ
        3. Текстового поля с описанием заказа работ
        4. Кнопку «Сохранить»
        5. Кнопку «Удалить»
        6. Кнопку «Отправить»
     3. Действия
        1. При нажатии на кнопку «Удалить» (5.7.2.1.3) отправляется запрос на API (4.3.4), который удаляет работу из заказа работ
        2. При нажатии на кнопку «Сохранить» (5.7.2.4) отправляется запрос на API (4.3.3), который сохраняет черновой заказ работ.
        3. При нажатии на кнопку «Удалить» (5.7.2.5) отправляется запрос на API (4.3.6), который удаляет черновой заказ работ и перенаправляет участника на страницу со списком работ (5.4)
        4. При нажатии на кнопку «Отправить» отправляется запрос на API (4.3.5), который формирует новый заказ работ и после чего перенаправляет участника на страницу с историей его заказов работ (5.8)
  4. Страница с историей заказов работ
     1. Доступна только авторизованным участникам
     2. Выводится информация о заказах в табличном виде (метод 4.3.1):
        1. Таблицу с историей заказов работ, содержащую столбцы
           1. № Заказа работ
           2. Статус
           3. Названия работ, которые содержатся в заказе
           4. Дату формирования заказа
           5. Результат печати микроэлектроники
           6. Кнопка подтвердить

Доступен только сотрудникам

* + - * 1. Кнопка отклонить

Доступен только сотрудникам

* + - 1. Фильтр по участнику
         1. Доступен только сотрудникам
         2. Выполняется на стороне клиента
      2. Фильтр по дате формирования заказа в виде диапазона
      3. Фильтр по статусу заказа
         1. Участнику фаблаба доступны статусы

Черновик

В работе

Завершён

Отклонён

Удалён

* + - * 1. Сотруднику доступны статусы

В работе

Завершён

Отклонён

* + 1. Действия
       1. При изменении состояния фильтров (5.8.2.3 и 5.8.2.4) отправляется запрос на API (4.3.1), который обновляет таблицу с заказами работ
       2. При нажатии на любую строчку таблицы происходит перенаправление участника на страницу заказа (5.8.2)
       3. При нажатии на кнопку Подтвердить (5.8.2.1.6) отправляется запрос на API (4.3.7), который меняет статус заказа с «в работе» на «завершён»
       4. При нажатии на кнопку Отклонить (5.8.2.1.7) отправляется запрос на API (4.3.7), который меняет статус заказа с «в работе» на «отклонен»
  1. Страница редактирования / добавления работ
     1. Доступна только сотрудникам
     2. Содержит
        1. Файловое поле, для загрузки / изменения фото работы
        2. Текстовое поле для ввода / изменения названия работы
        3. Текстовое поле для ввода / изменения описания работы
        4. Кнопку «Удалить»
        5. Кнопку «Сохранить»
        6. Кнопку «Добавить»
     3. Действия
        1. При нажатии на кнопку «Удалить» (5.9.2.6) выполняется запрос на API (4.2.7), а также происходит переадресация на страницу со списком работ(5.4)
        2. При нажатии на кнопку «Сохранить» (5.9.2.7) выполняется запрос на API (4.2.6), а также происходит переадресация на страницу со списком работ (5.4)
        3. При нажатии на кнопку «Добавить» (5.9.2.8) выполняются два запроса на API: сначала на добавление нового работы (4.2.5), а затем на обновление работы (4.2.6), также происходит переадресация на страницу со списком работ (5.4)

1. **Требования к программному обеспечению:**
   1. Серверная часть
      1. Операционная система (любая из)
         1. Linux **(**6.4.12 и выше)
         2. Windows (22H2 и выше)
      2. Веб-сервер Nginx (1.25.3 и выше)
      3. Minio (RELEASE.2023-12-14T18-51-57Z и выше)
      4. Redis (7.2 и выше)
      5. Python (3.10 и выше) с фреймворком Django (4.1 и выше)
      6. СУБД PostgreSQL (16.1 и выше)
      7. Доступ к интернету
   2. Клиентская часть
      1. Операционная система (любая из)
         1. Windows (22H2 и выше)
         2. MacOS (11.7.10 и выше)
         3. Linux **(**6.4.12 и выше)
      2. Веб-браузер (любой из)
         1. Safari (16.5.2 и выше)
         2. Chrome (119.0.6045 и выше)
         3. Opera (105.0.4970.16 и выше)
      3. Доступ к интернету
2. **Требования к аппаратному обеспечению:**
   1. Серверная часть
      1. Процессор минимум 2-ядерный с частотой от 2 ГГц.
      2. Оперативная память от 4 Гб.
      3. Место на жестком диске от 2 Гб.
   2. Клиентская часть
      1. Процессор с частотой от 1ГГц.
      2. Оперативная память от 512 Мб.