|  |  |
| --- | --- |
|  | **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: «Приказы ректора»

Студент: Журмилов В.Д.

Группа ИУ5-53Б

Преподаватель: Канев А.И.

2024г.

# **ВВЕДЕНИЕ**

В эпоху цифровых технологий, когда электронные системы документа оборота становятся неотъемлемой частью крупных организаций, традиционные методы пересылки документов, такие как почта, уступают свои позиции эффективным и мгновенным средствам обмена данными в виртуальном пространстве. В связи с этим, сервис для создания и отправки приказов через электронные каналы может значительно повысить скорость актуализации приказов. В данном контексте представляется востребованной идея разработки сервиса, объединяющего веб-сервис, веб-приложение, мобильное приложение и выделенный сервис для подписания приказов.

Цель данного проекта заключается в создании сервиса, способствующего оперативной и эффективной коммуникации между сотрудниками в рамках учебного учреждения.

Система предназначена для составителей приказов и модераторов приказов. Предусмотрен доступ к списку подразделений с их подробной информацией. Составители приказов могут создавать приказы, выбирая подразделения и указывая текст приказа. Система предоставляет автоматизированный способ создания, учета и ведения приказов, обеспечивая эффективное управление электронными приказами. Модераторы имеют возможность подтверждать или отклонять приказы, а также редактировать существующие подразделения или создавать новые.

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 языке C# .NET Core 7.0.
  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 составителя приказов в React.
  8. Реализовать интерфейс модератора приказов React.
  9. Создать мобильное приложение на React-Native.
  10. Создать асинхронный сервис для отслеживания отправки уведомления на Golang.
  11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

# **БИЗНЕС-ПРОЦЕСС**

Администрации университета может быть необходимо срочно составить приказ для большого количества подразделений [1]. Для того, чтобы не отправлять приказ отдельно каждому и тратить на это много времени, составители приказа могут создавать приказ для рассылки сразу всем. Каждый приказ вначале создается как черновик, в который можно добавлять и из которого можно удалять подразделения. В одном приказе можно указать сразу несколько подразделений, и тогда приказ отправится сразу всем подразделениям.

Когда составитель удостовериться, что выбраны все нужные подразделения, он формирует приказ и он отправляется на модерирование. После этого его уже нельзя редактировать. Можно также посмотреть историю своих приказов [2]. К выбору подразделений также стоить отнестись с внимательностью. Для этого модераторы отклоняют и подтверждают приказы таким образом, чтобы приказы получили нужные подразделения, которые и должные получить приказ. Приказы требуют подпись ректора. За это отвечает внешний сервис контроля отправки, он оповещает о результате отправления.

Подразделение – это структурное подразделение МГТУ им. Баумана. У каждого из них разное название, описание. Также у подразделений есть изображение, по которому сразу можно узнать необходимое подразделение. В описании подразделения могут присутсвовать какие-то ошибки, и в этом случае модератор может либо изменить его данные, либо полностью удалить подразделение из списка. Если у подразделения поменялись данные, то модератор может внести соответствующие изменения в системе. Он может отредактировать какое-либо подразделение, обновить фото, или исправить какие-то ошибки. Также модератор может создать новое подразделение. Функции пользователей с разными ролями описаны на диаграммах прецедентов (рис. 1).

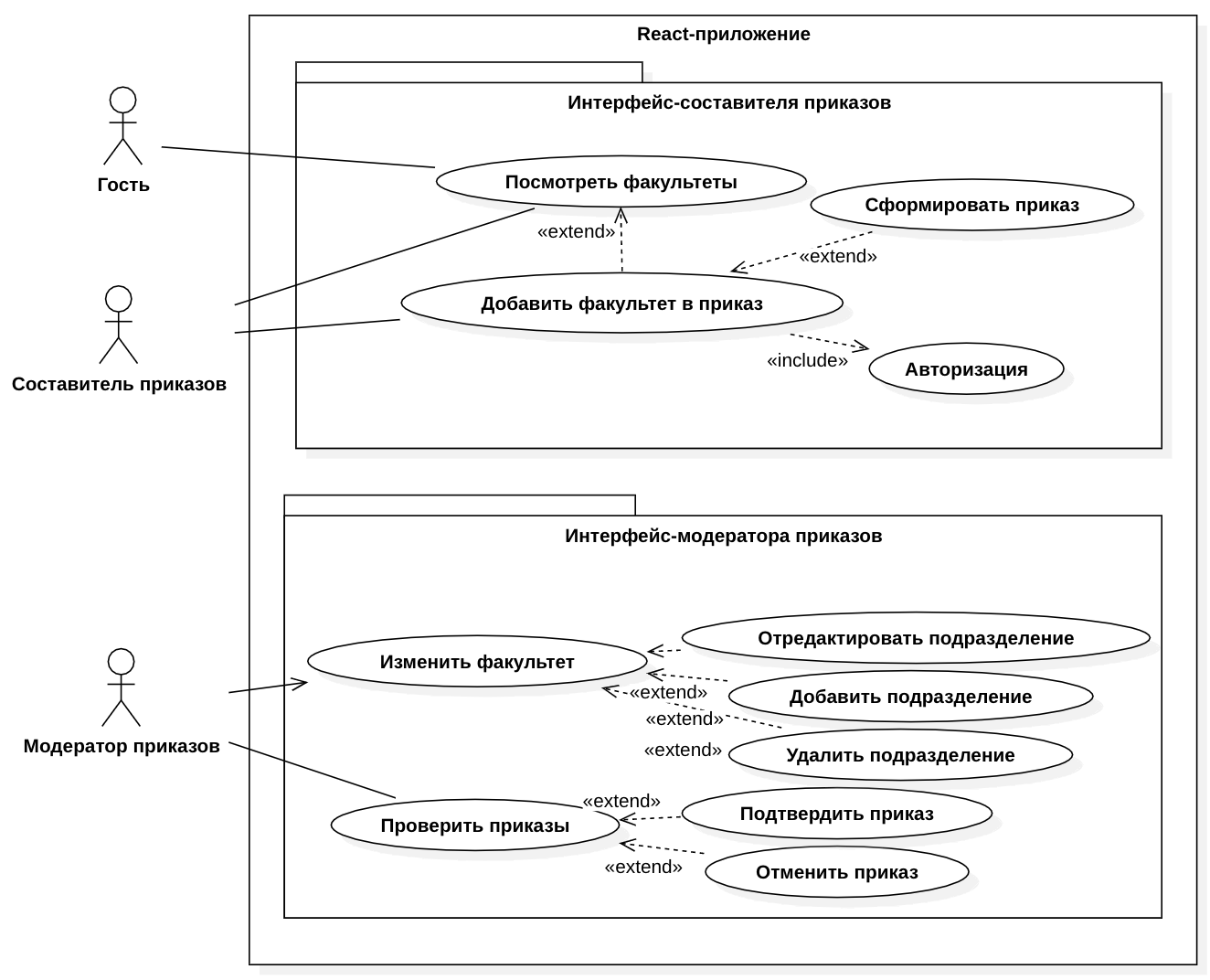


Рисунок 1 - Диаграмма прецедентов

Гостям доступен просмотр подразделений. Зарегистрированные гости – составители приказов. Они могут добавлять подразделения в приказ, просматривать список своих приказов и сформировать текущий приказ. Приказы обрабатываются модераторами. В результате обработки приказа, его либо подтверждают, либо отклоняют. Также модератору приказов доступны уникальные функции для работы с подразделениями, а именно: просмотр всех подразделений, создание и удаление подразделений, а также просмотр всех подразделений в табличном виде. Процесс формирования приказа отражен на диаграмме деятельности (рис. 2).

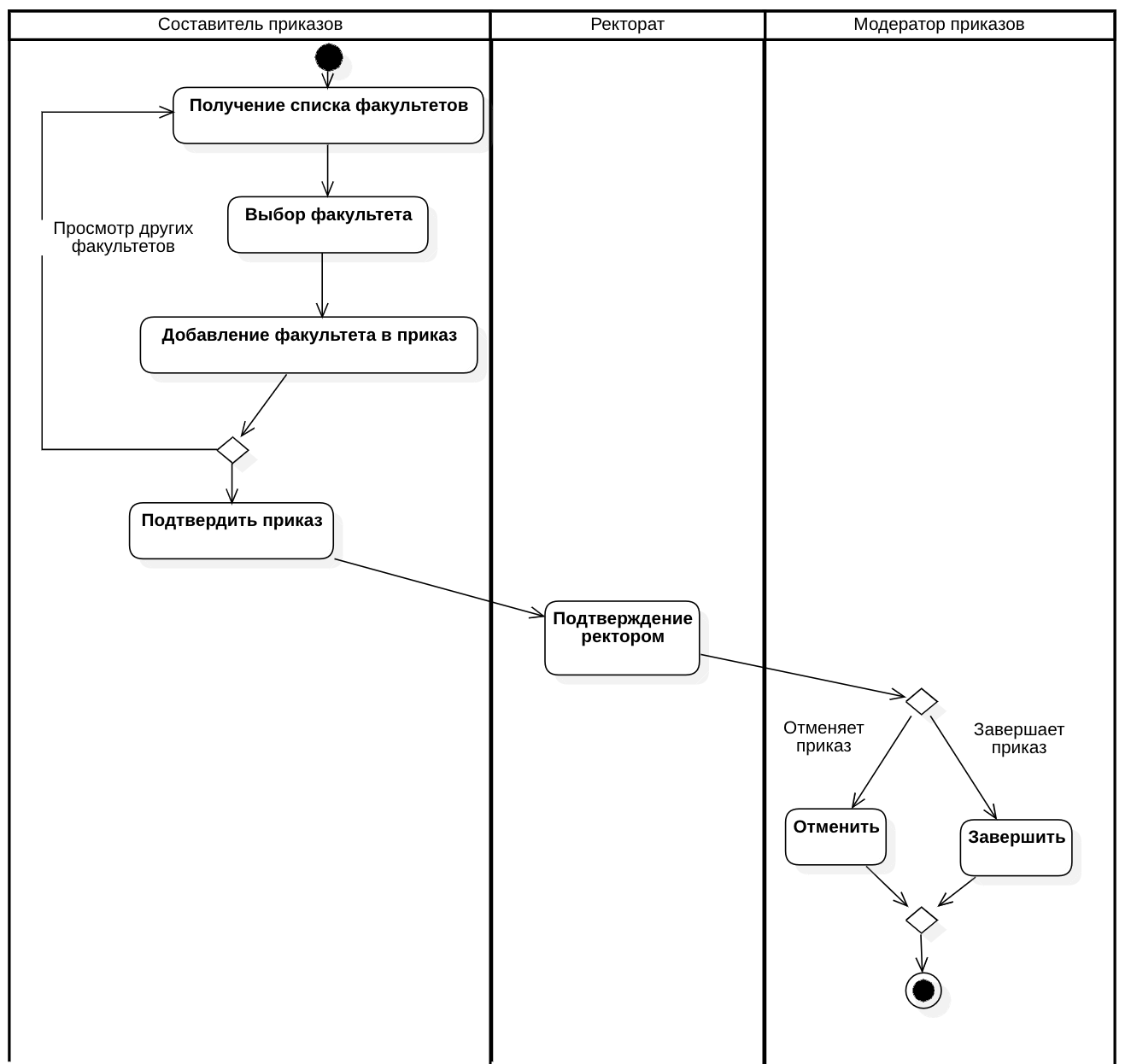


Рисунок 2 - Диаграмма деятельности

Составитель выбирает подразделения, затем формирует на основе выбранных подразделений приказ. Эту заявку обрабатывает асинхронный сервис, а затем и модератор. Возможные состояния уведомления отражены на диаграмме состояний (рис. 3).

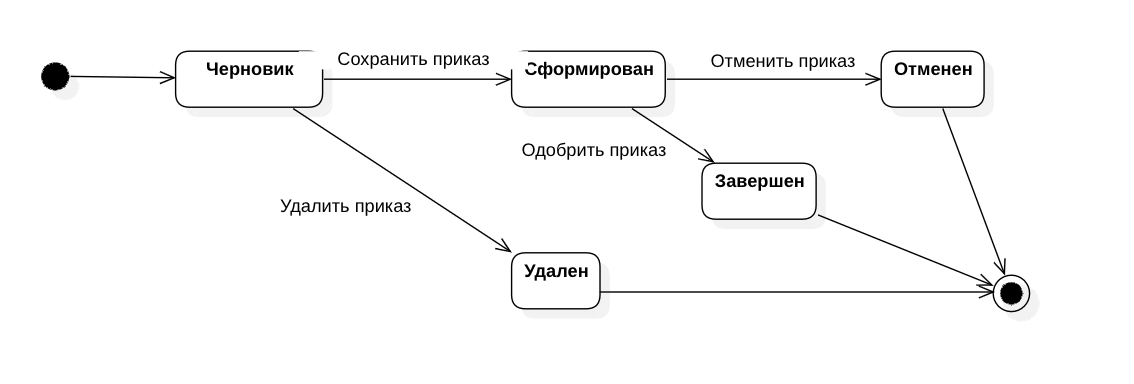


Рисунок 3 - Диаграмма состояний уведомления

При выборе первого подразделения формируется черновик. Последующие выбранные подразделения добавляются в этот черновик. Уведомитель затем формирует заявку, удаляет её или выходит из приложения. Сформированную заявку обрабатывает модератор. Он может одобрить или отклонить её, в зависимости от правильности составления приказа.

# **АРХИТЕКТУРА**

Архитектура системы отображена на диаграмме развертывания (рис. 4). Бэкенд разворачивается на самом устройстве, а базы данных и s3 хранилище изображений в докер контейнерах [6]. Такое решение было принято в связи с тем, что технология докер контейнеров позволяет быстро и удобно разворачивать целые системы на любых устройствах, поддерживающих докер. Еще одним неоспоримым преимуществом докер контейнеров является то, что их контейнеры представляют собой виртуальные машины, а значит их работа не зависит от платформы устройствах. Асинхронный сервис [3] разворачиваются отдельно и необязательно на том же компьютере, что и основной сервис.

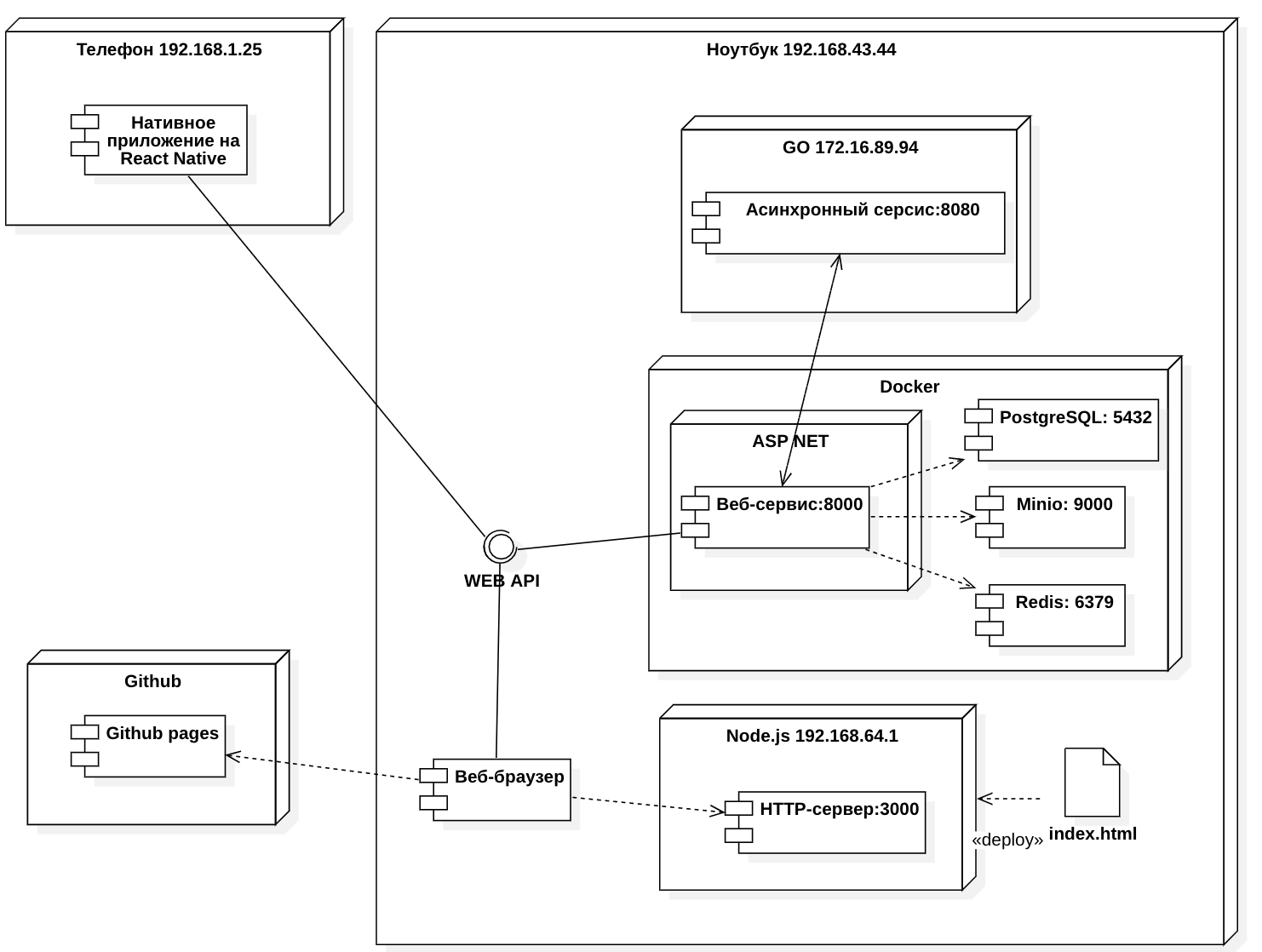


Рис. 4 - Диаграмма развертывания

Мобильное [10] и браузерное приложения [7, 8, 9] обращаются к веб-сервису на базе технологии ASP .NET [3, 4, 5] через REST Web-API. Использование ASP .NET обосновано его проверенной эффективностью, признанием в индустрии и современностью языка. Современность используемого в ней языка C# также склоняет к выбору этой технологии.

Данные хранятся в СУБД PostgreSQL [11], 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, приказ также имеет поле Тело приказа, отражающее текст приказа и подпись, полученную от асинхронного сервиса. Модель получателей представляет собой набор полей, необходимых исключительно для бизнес-логики. Для хранения в одной заявке нескольких карт используется промежуточная таблица, реализующая связь М-М.

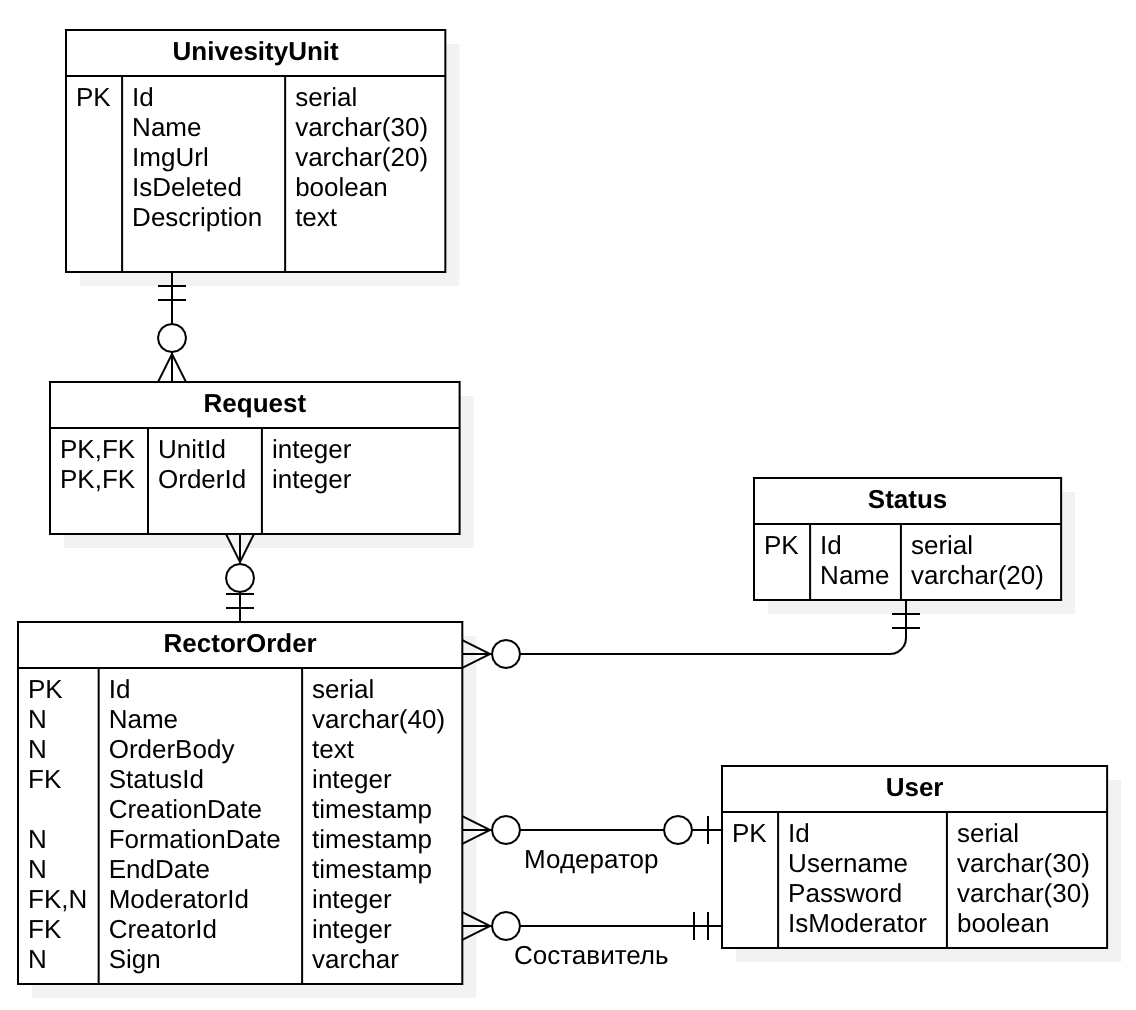


Рисунок 5 - ER диаграмма

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

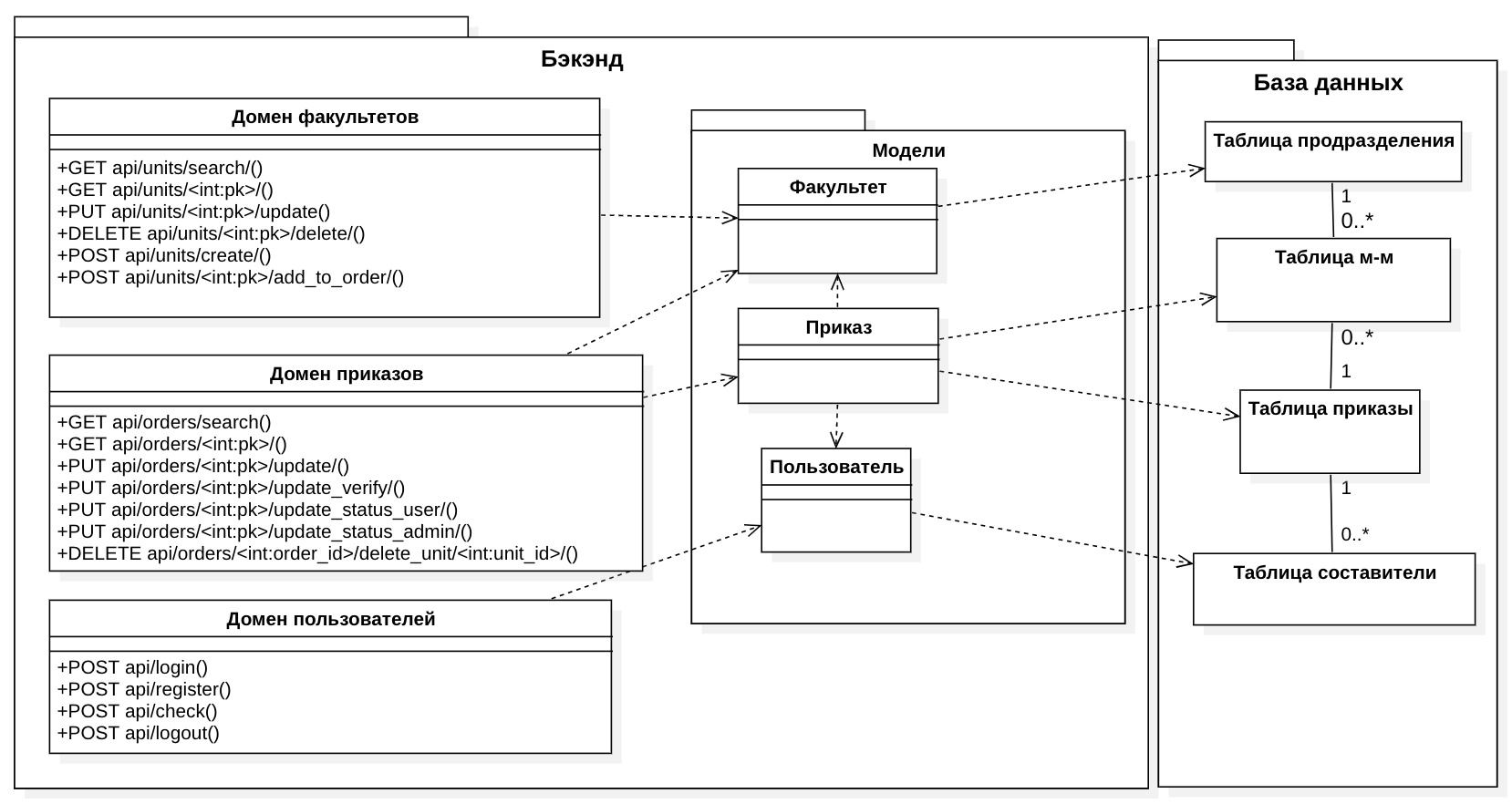


Рисунок 6 *-* Диаграмма классов бекенда

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

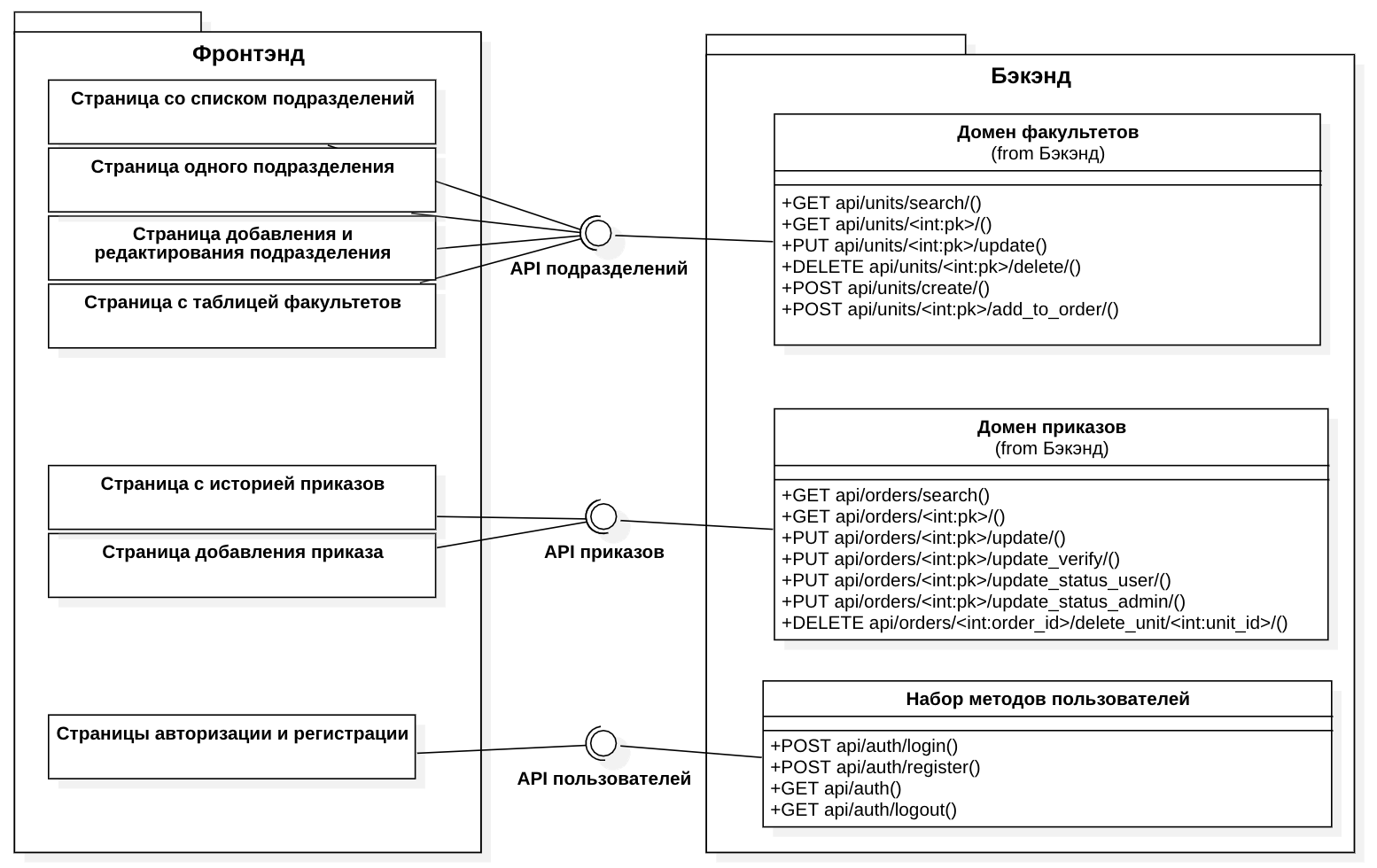


Рисунок 7 *-* Диаграмма классов фронтенда

# **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: получатели, уведомления, и аутентификация. Методы следуют правилам REST API.

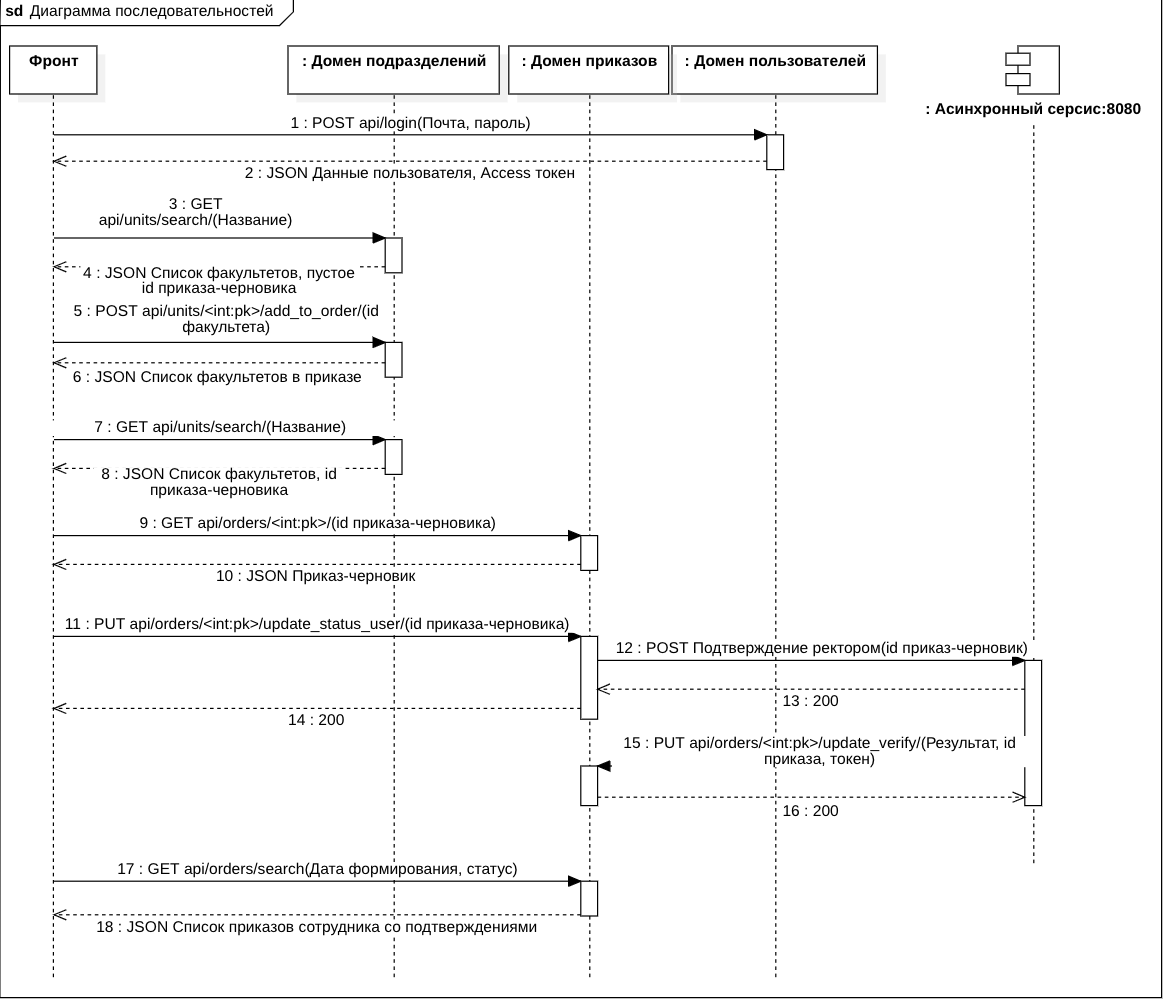


Рисунок 8 - Диаграмма последовательности

В начале бизнес-процесса происходит аутентификация пользователя. Для этого он отправляет через графический интерфейс запрос, передавая в нем логин и пароль. Если аккаунт с такими данными существует, то пользователь получает Cookie в ответном запросе. Если же такого аккаунта не существует, или пароль введен неверно, пользователь получит ошибку. В таком случае ему надо либо пройти регистрацию, либо ввести пароль верно. Затем графический интерфейс пользователя запрашивает у веб-сервиса список подразделений, которые возвращаются в JSON формате. Составитель приказов выбирает подразделение, которому хочет отправить уведомление, и, нажимая на кнопку «Добавить в корзину» в графическом интерфейсе, отправляет запрос на добавление получателя в свое черновое уведомление. Этот процесс может продолжаться несколько раз.

Когда уведомитель определится с выбором, он нажимает на кнопку «сформировать» в графическом интерфейсе. После этого приложение запрашивает id черновой заявки уведомителя и затем отправляет запрос на формирование этой заявки. В этот момент основной веб-сервис выполняет асинхронный запрос к другому сервису, чтобы он отследил статус отправки уведомления и вернул его в основной сервис. Через некоторое количество времени уведомитель может увидеть, что статус его уведомления изменился.

Процесс рассмотрения уведомлений происходит также через графический интерфейс. Модераторы могут просматривать списки всех уведомлений и, нажимая на соответствующие кнопки, отправлять запросы на подтверждение или отклонение уведомлений в основной веб-сервис. В эти запросы также можно включить фильтры по имени создателя уведомления, одному из статусов уведомления и диапазону дат, в которых эти уведомления были сформированы. Также модераторы могут управлять получателями через графический интерфейс. Им доступны такие функции, как создание и редактирование получателей, просмотр списка получателей в виде таблицы и удаление их. Для каждой из этих функций присутствует свой метод, отправляемый на основной веб-сервис.

# **ОПИСАНИЕ ИНТЕРФЕЙСА**

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

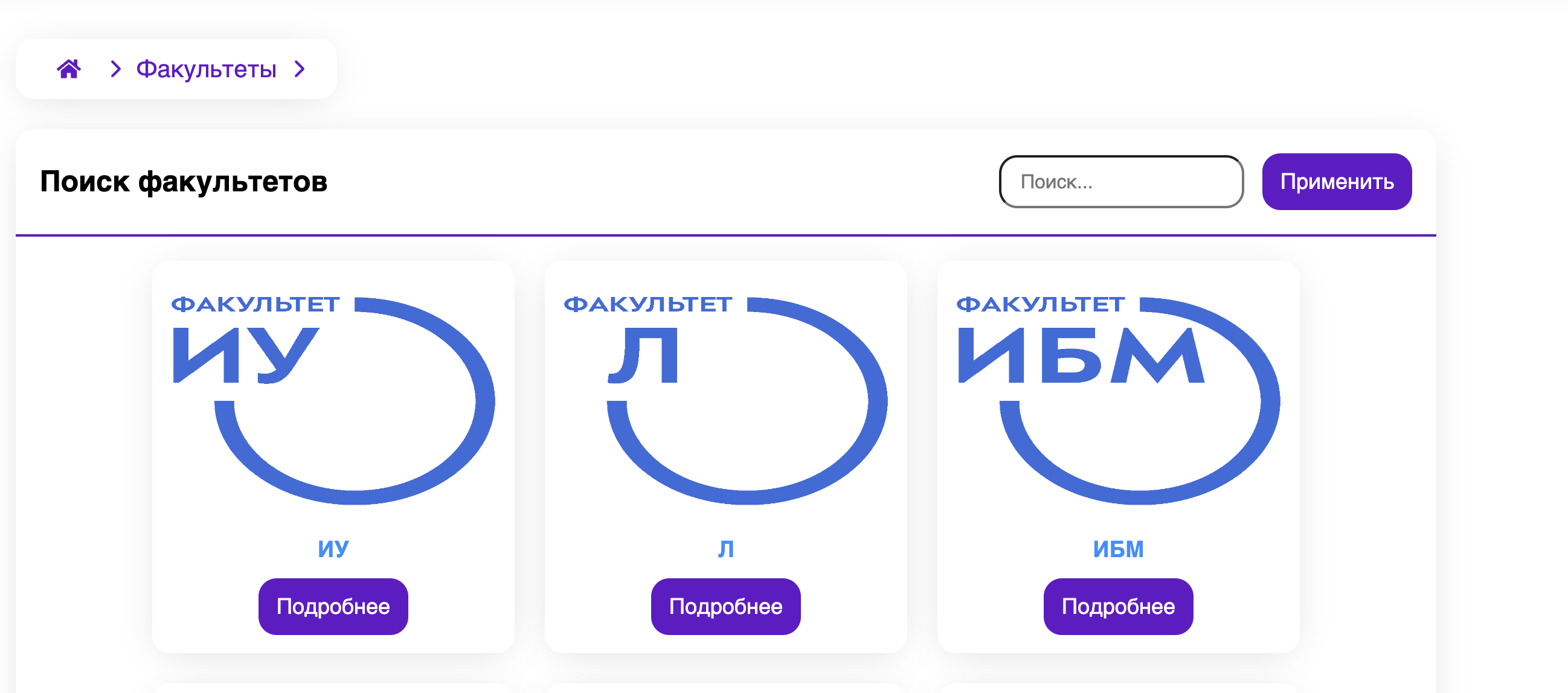


Рисунок 9 - Главное меню (неавторизированный пользователь)

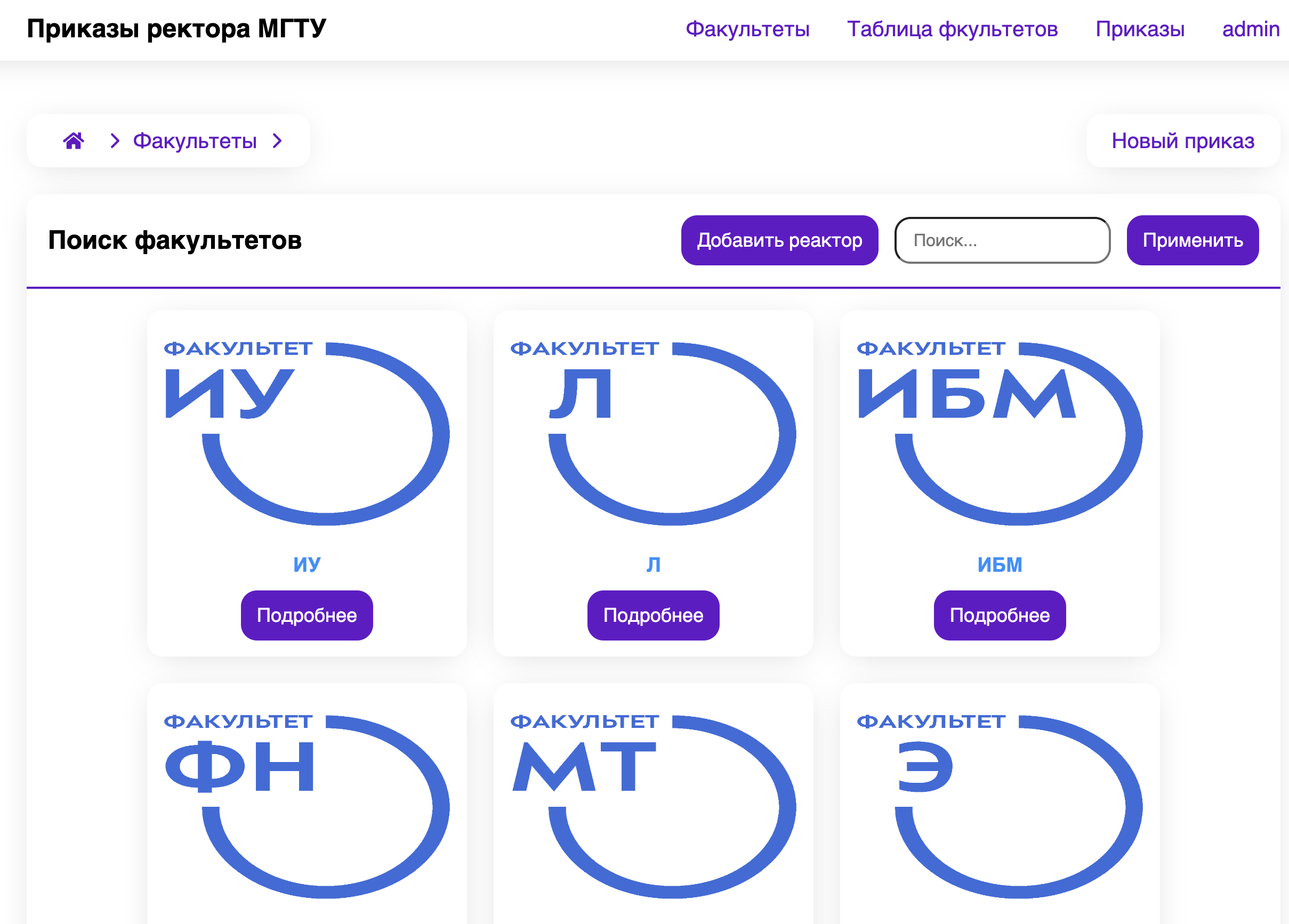


Рисунок 10 - Главное меню (модератор)

Первоначальная страница для всех пользователей и гостей. В зависимости от типа пользователя её содержимое меняется. Гости, например, могут посмотреть список получателей и подробную информацию о них, а также нажать на кнопку «Войти», либо «Регистрация». А у пользователя уже есть кнопки «Получатели», «Уведомления», «Корзина» и «Выход».

На странице с формой авторизации (рис. 11) отображается форма, через которую гость входит в свой аккаунт. При успешном вводе логина и пароля гость получает Cookie, который используется при отправлении запросов.

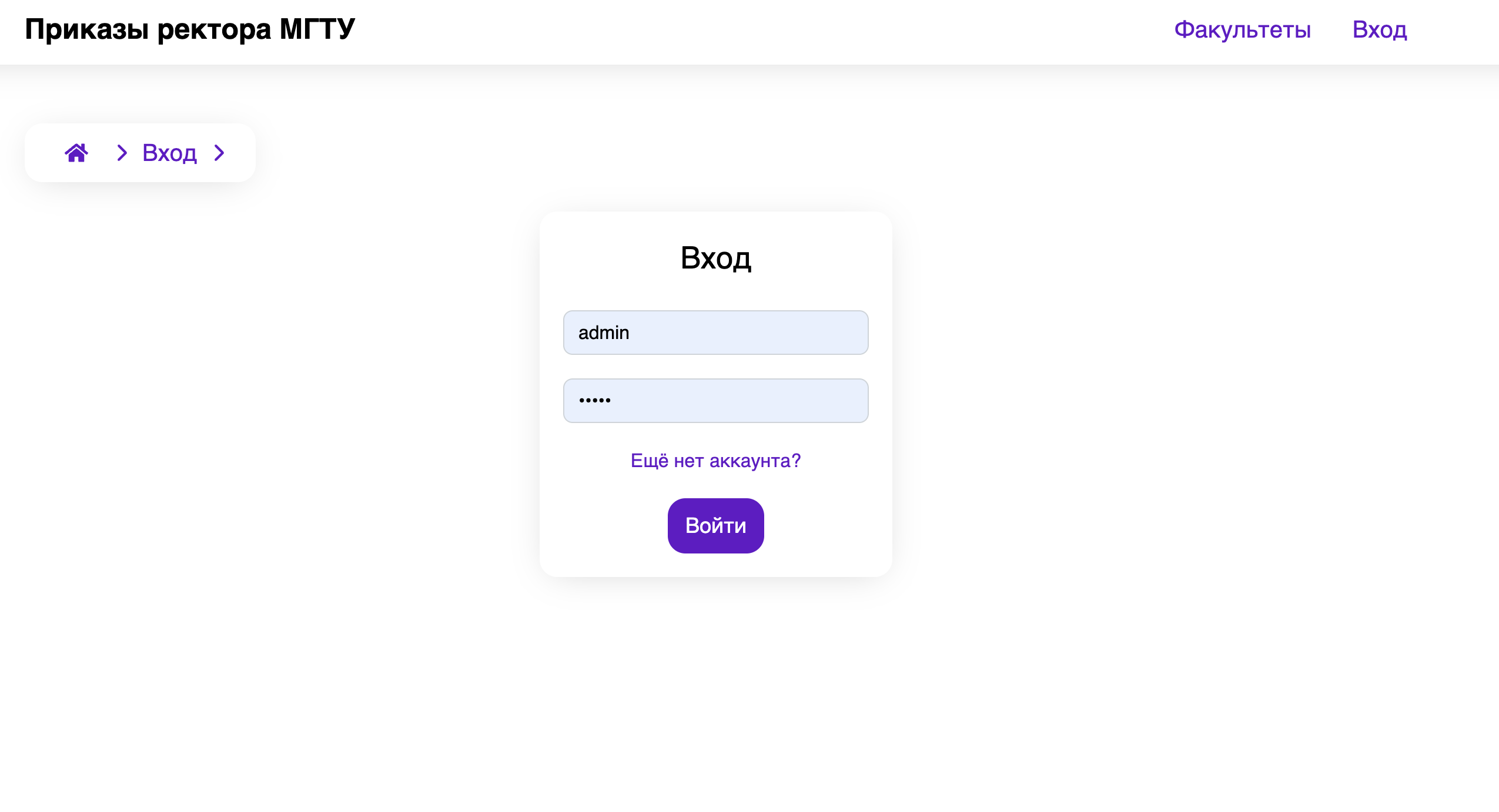


Рисунок 11 - Страница авторизации

На странице с формой регистрации (рис. 12) гости могут завести аккаунт. Для этого нужно указать логин, и пароль. Если введенный логин уже занят система попросит пользователя сменить его.

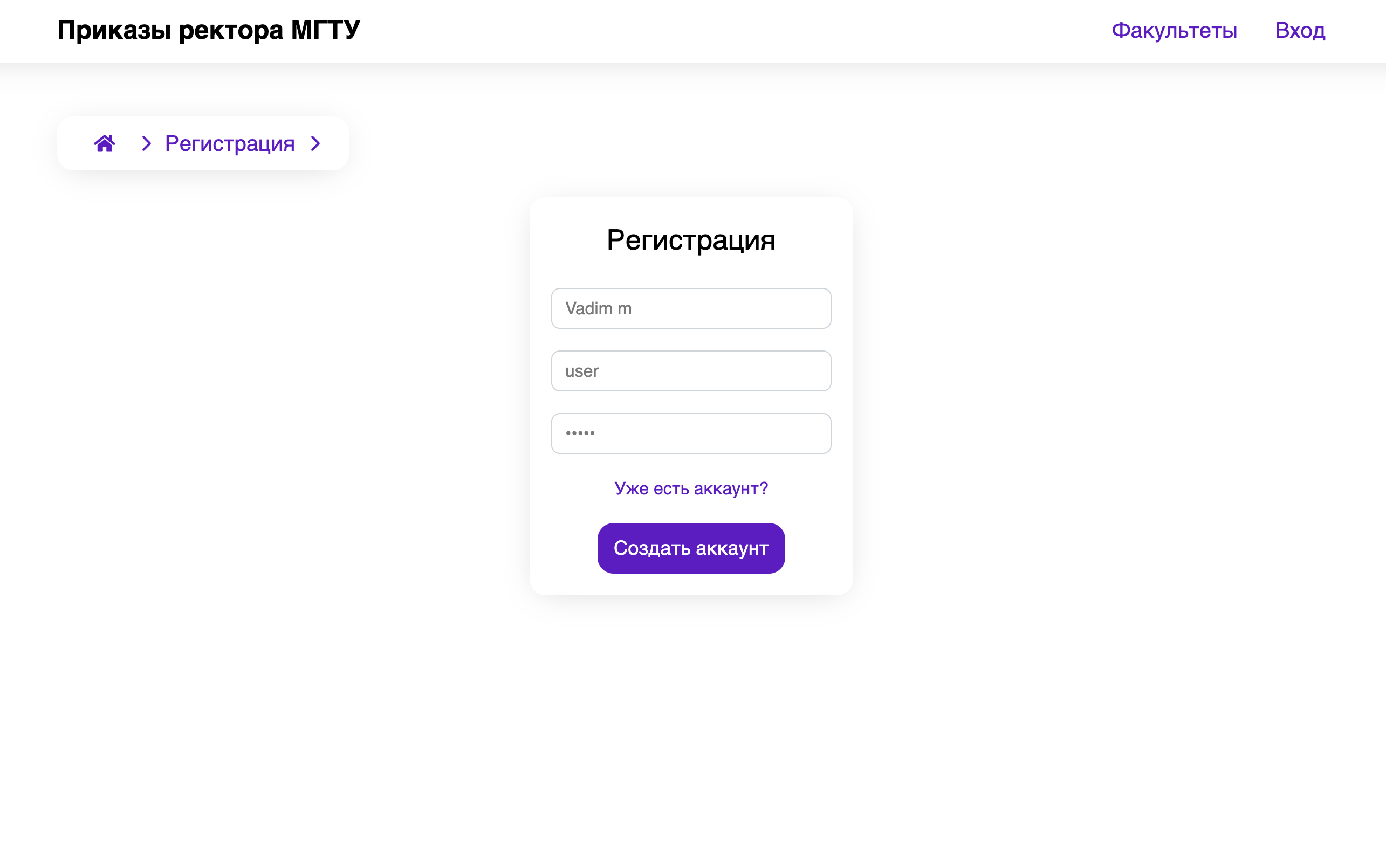


Рисунок 12 - Страница регистрации

На странице со списком получателей (рис. 13) отображается список получателей в виде карточек. У каждой карточки есть кнопка «Подробнее», переносящая пользователя на страницу с подробной информацией о получателе, и кнопка «добавить в корзину» для добавления получателя в черновое уведомление. Сверху находится фильтр получателей, там можно ввести ФИО получателя для отображения нужных пользователю.

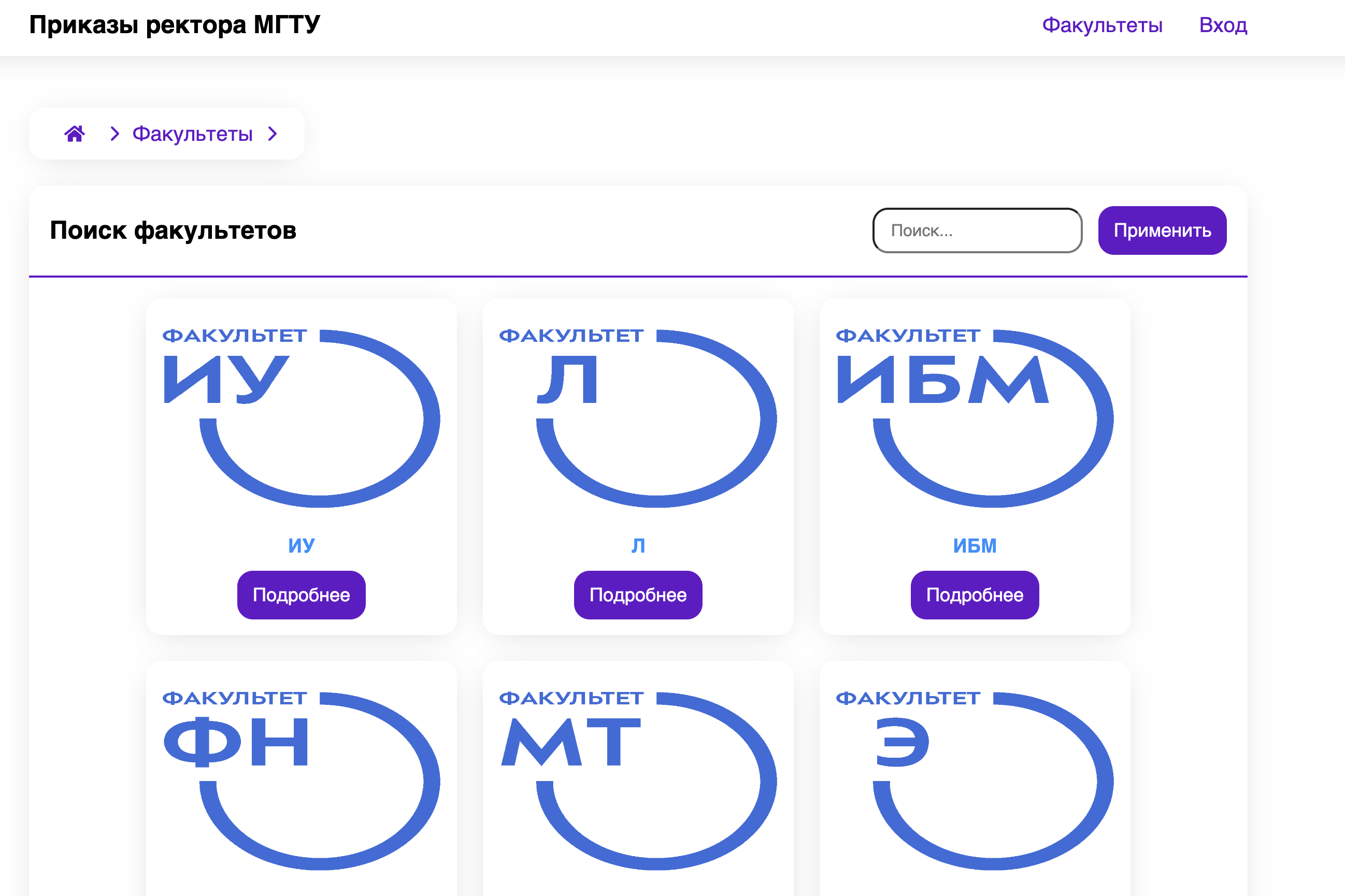


Рисунок 13 - Страница со списком получателей

На странице с подробным описанием получателя (рис. 14) отображается подробная информация о получателе: ФИО, почта, возраст и адрес.

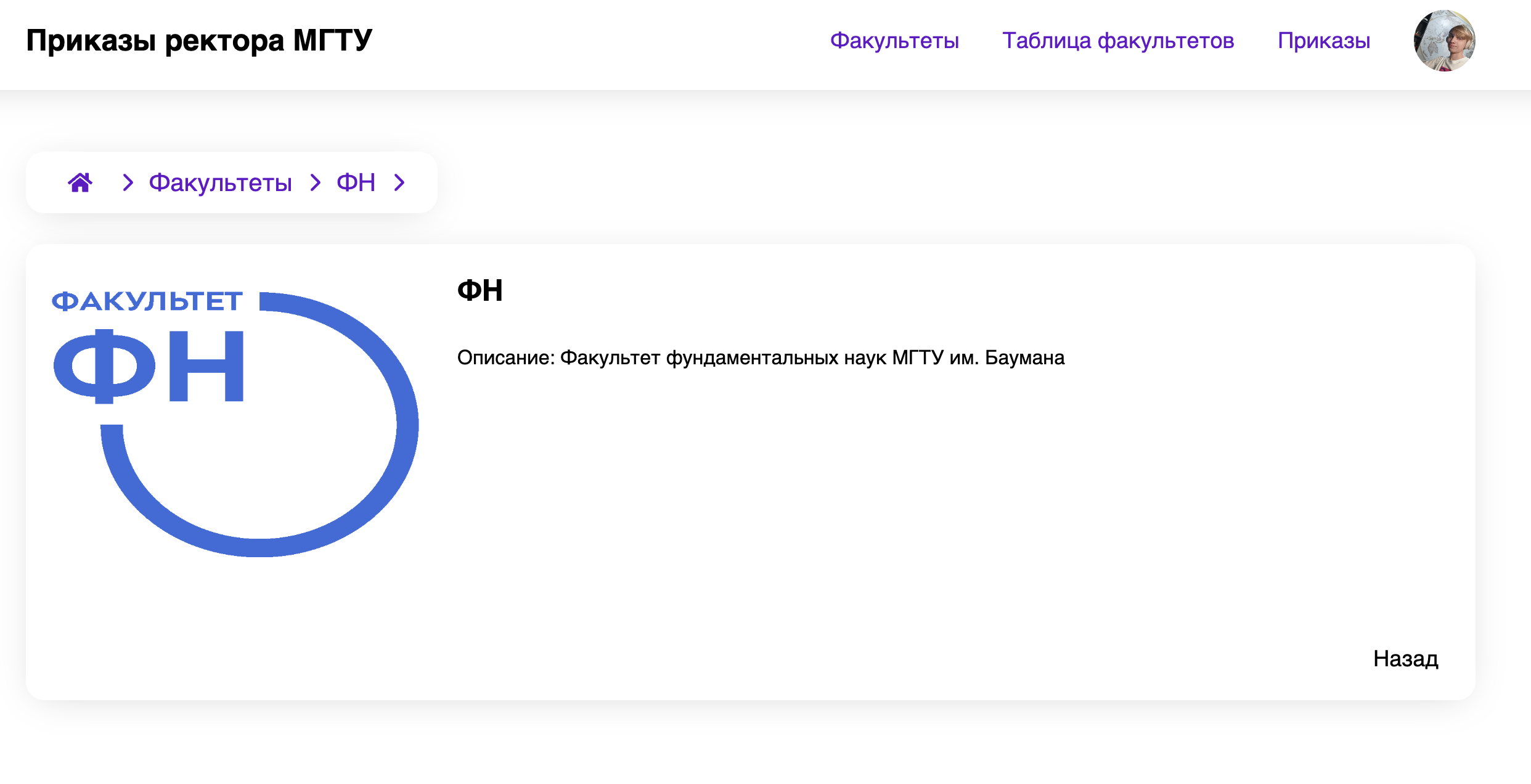


Рисунок 14 - Страница с подробным описанием получателя

На странице со списком уведомлений (рис. 15, 16) отображается список уведомлений. В зависимости от типа пользователя этот список будет функционально отличаться. Так, для уведомителей отображается список созданных ими заявок: статус, статус отправки, дата создания, дата формирования, дата завершения, тип уведомления и кнопка «Подробнее» для открытия заявки. Также можно отфильтровать уведомления по статусу, либо по дате формирования уведомления, выбрав либо одну дату, либо промежуток.

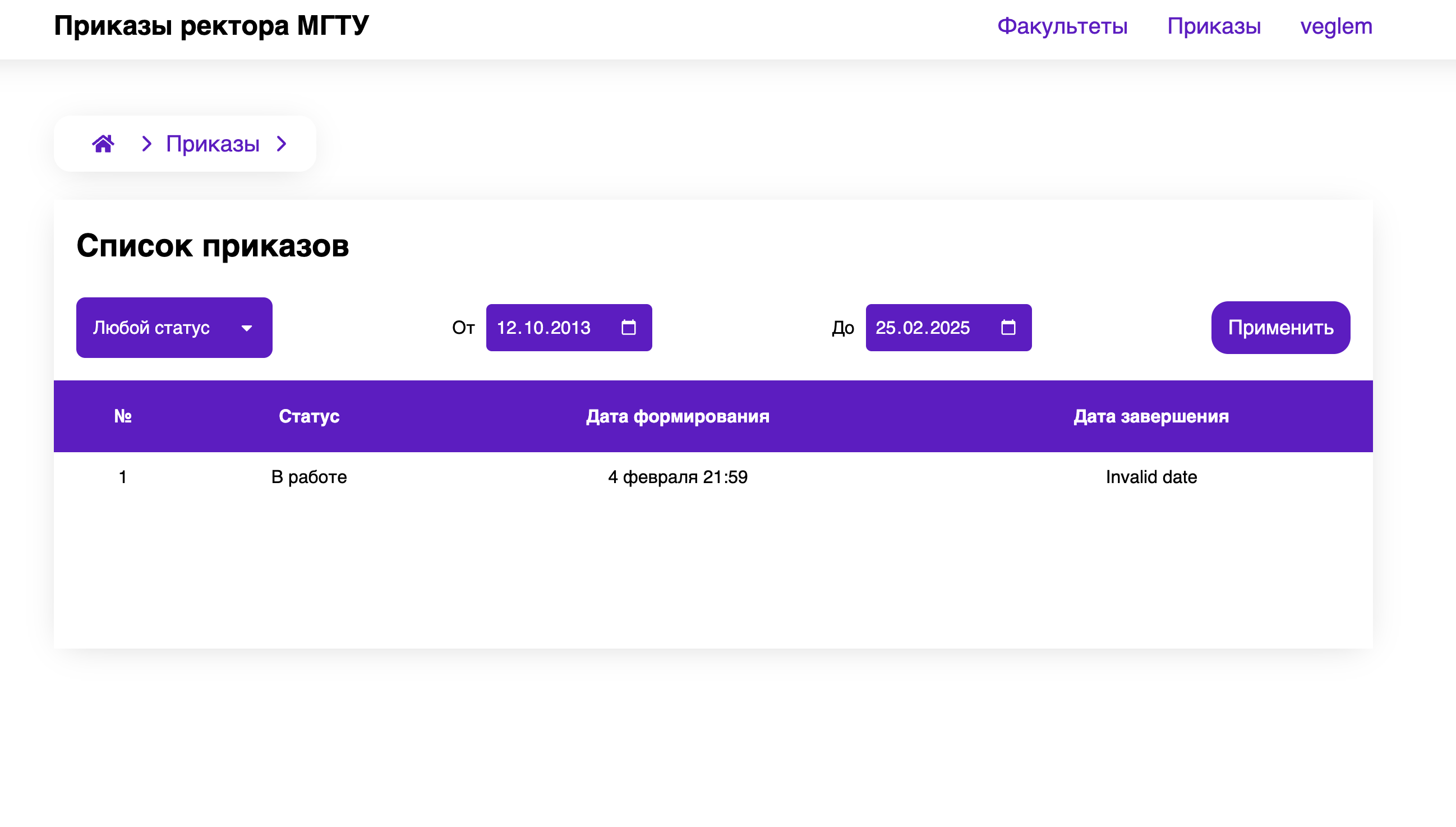


Рисунок 15 - Страница со списком приказов (пользователь)

Для модератора функция этой страницы шире. Для них отображается список всех заявок всех пользователей с более подробной информацией: создатель уведомления, статус, дата создания, дата формирования, дата завершения, тип уведомления, кнопка «Подробнее» для открытия заявки, и две кнопки «Подтвердить» и «Отменить» для подтверждения или отмены уведомления. Также присутствует фильтр по создателю уведомления.

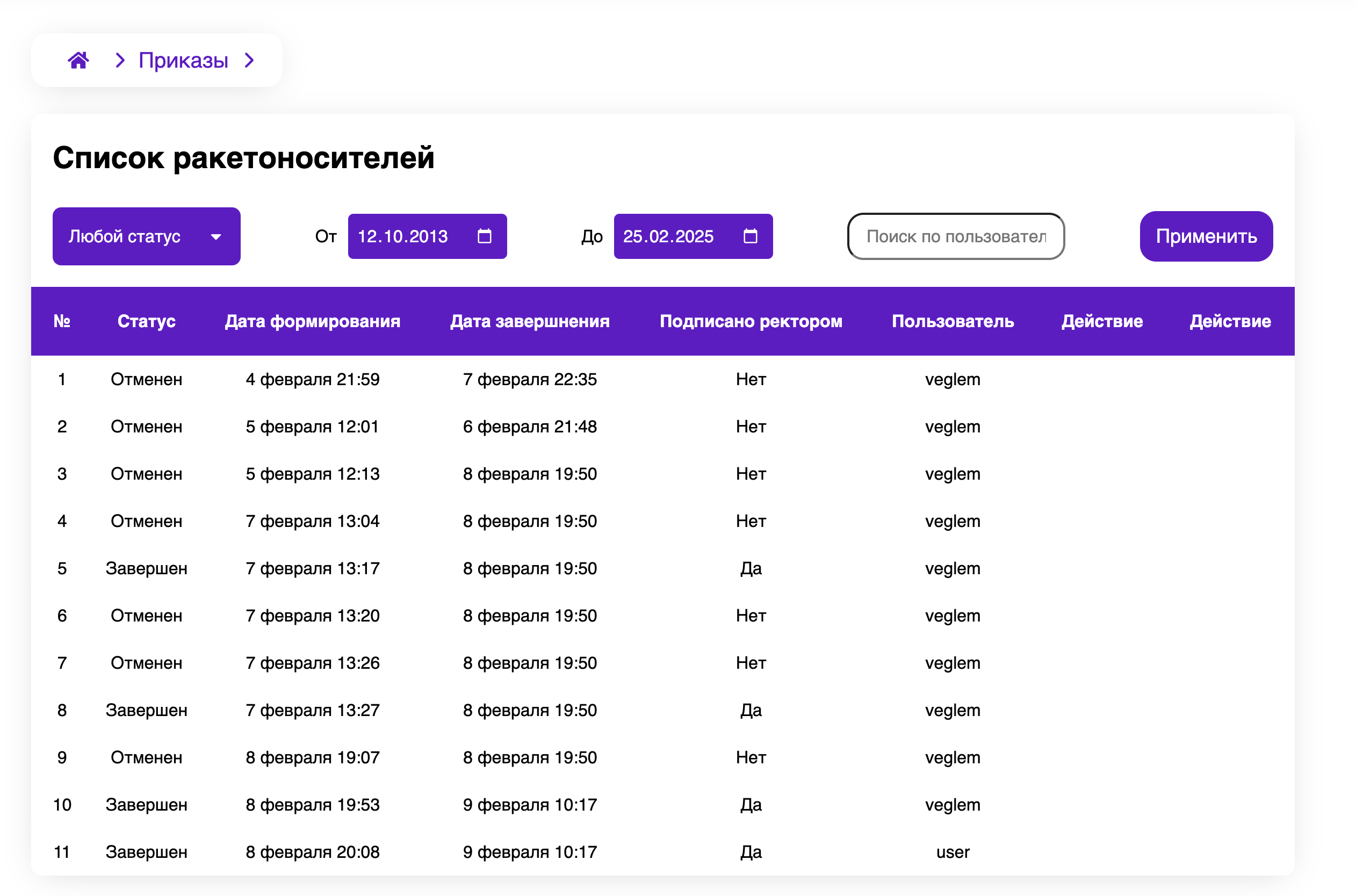


Рисунок 16 - Страница со списком уведомлений (модератор)

На странице с подробным описанием уведомления (рис. 17) отображается подробная информация об уведомлении. Список выбранных получателей в виде карточек, а также все поля уведомления.

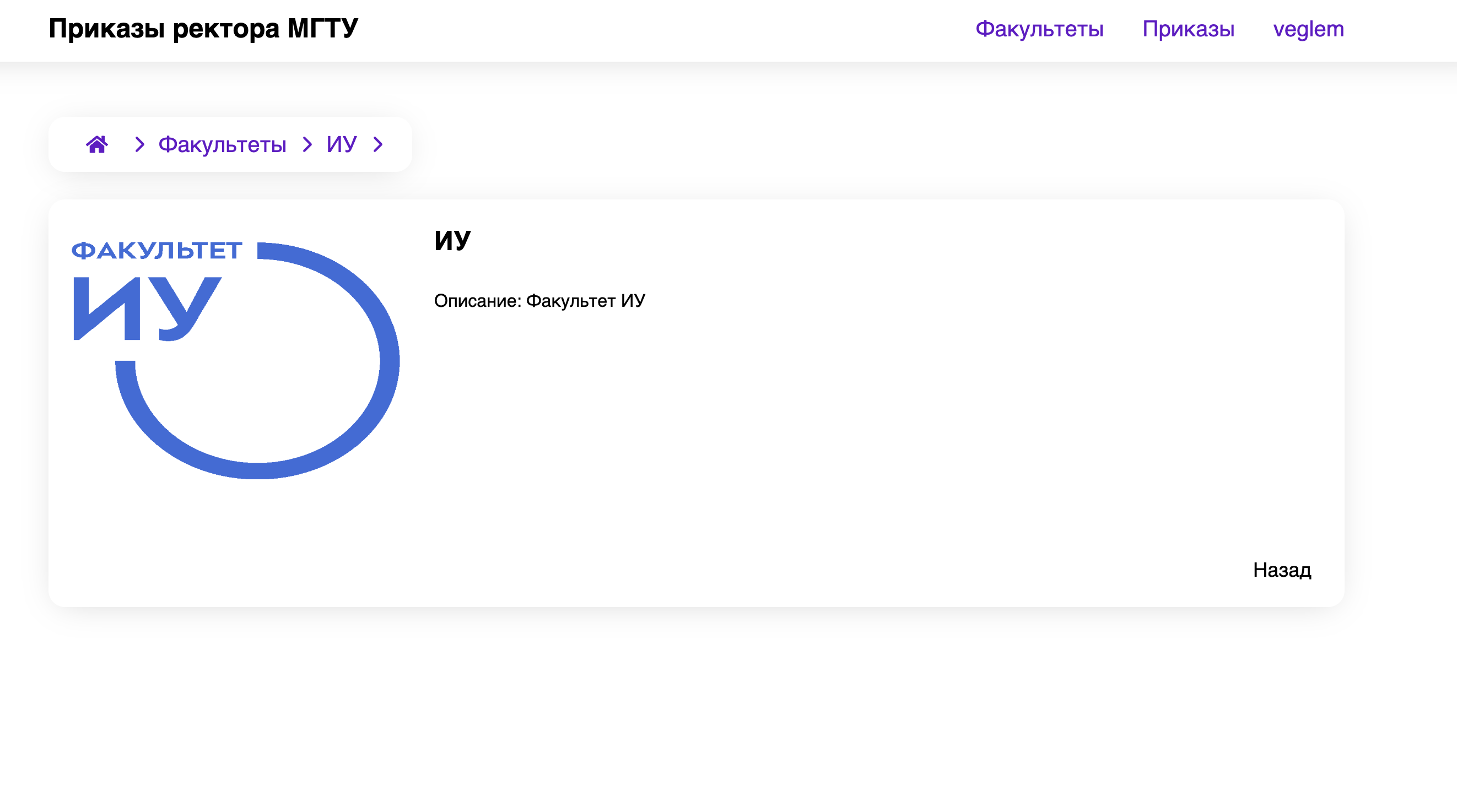


Рисунок 17 - Страница с подробным описанием уведомления

На странице с таблицей получателей (рис. 18) модератор может в компактном и удобном формате просмотреть список всех получателей, существующих в системе. Отображаются следующие поля: изображение получателя, ФИО, адрес, почта, возраст. Также можно открыть существующего получателя или перейти на страницу создания нового (рис. 19). Присутствует фильтрация по ФИО.

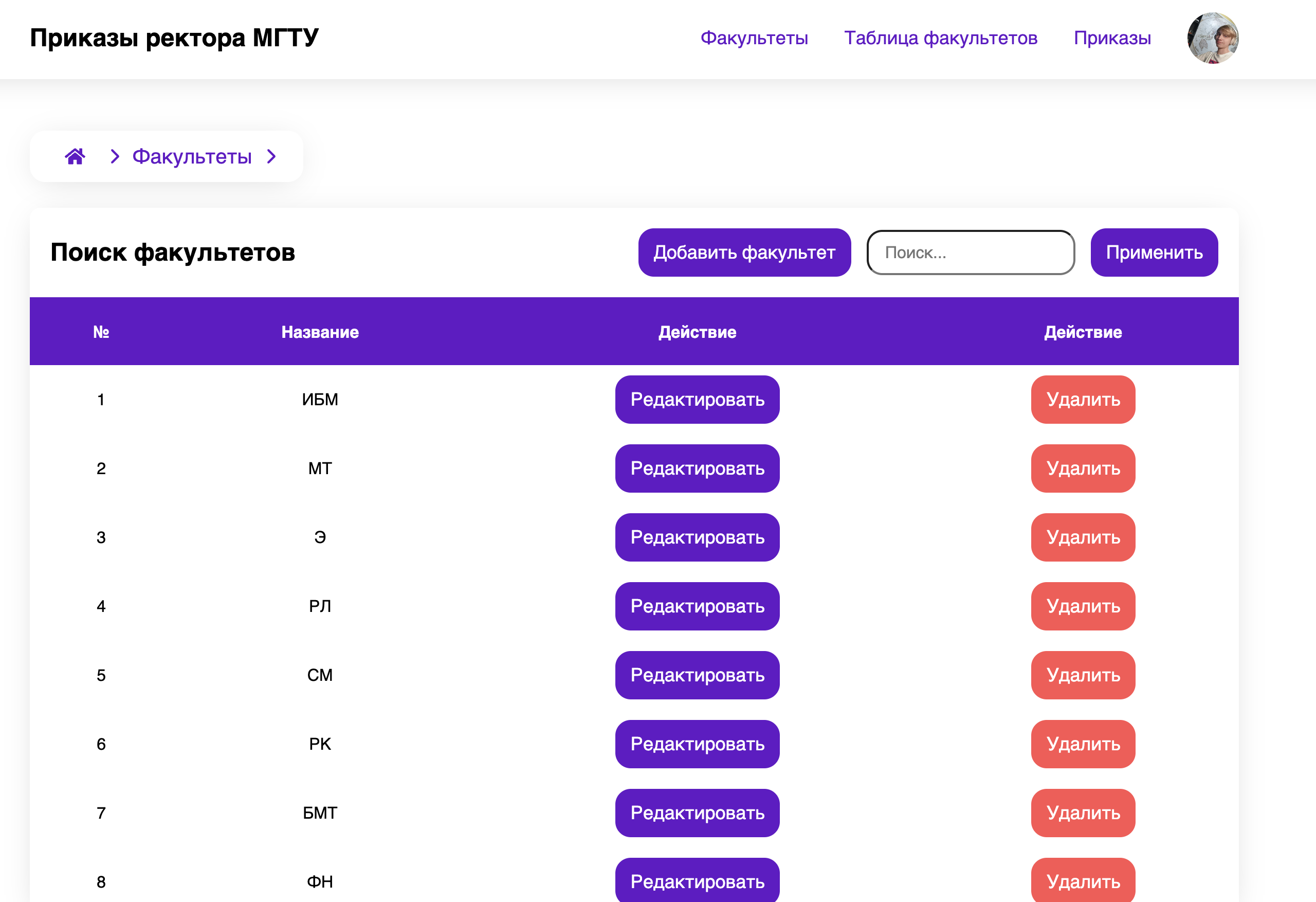


Рисунок 18 - Страница с таблицей получателей

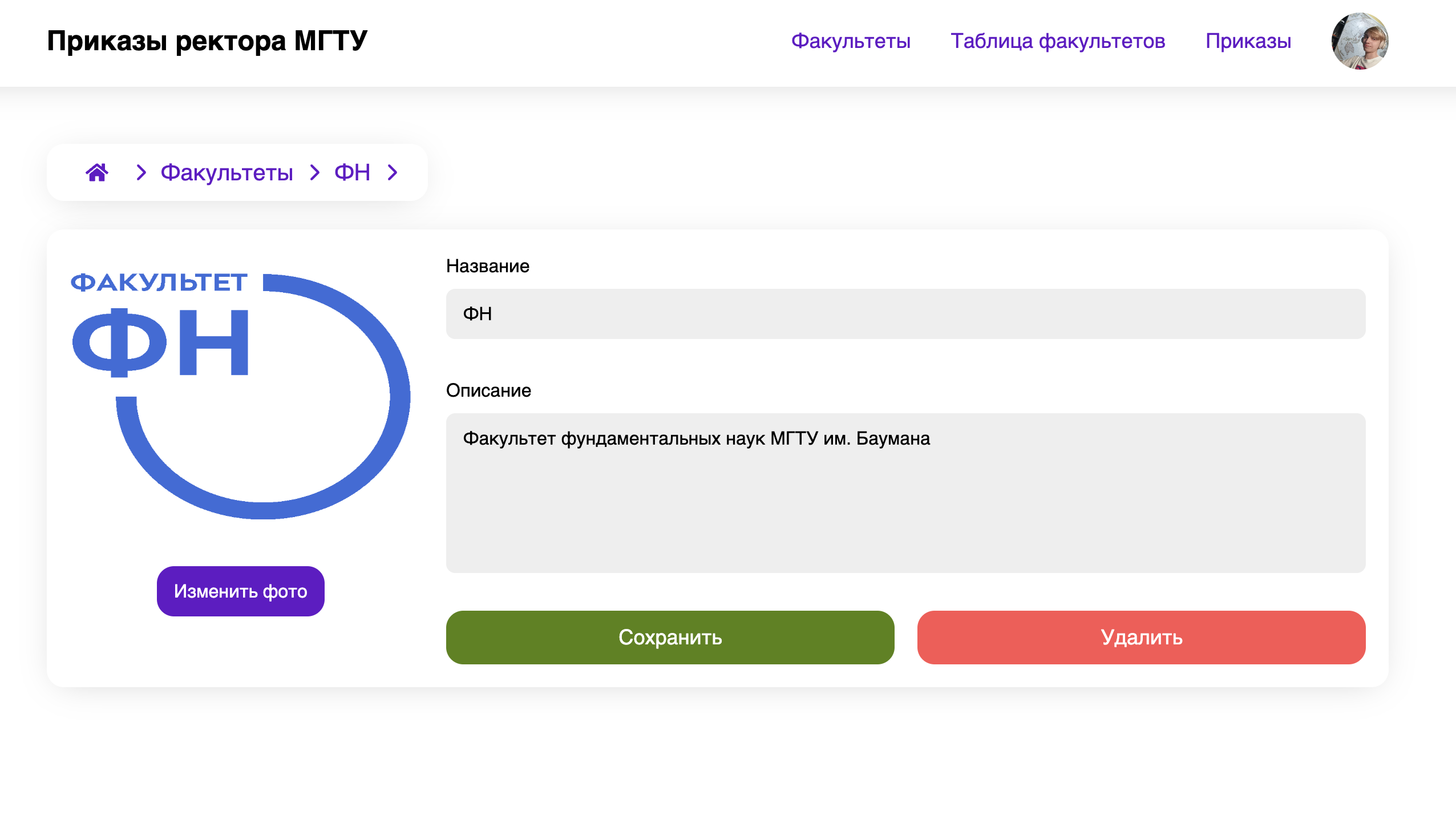


Рисунок 19 - Страница редактирования получателя

На странице редактирования получателя (рис. 19) модератор может отредактировать существующего получателя или создать нового. Для редактирования доступны все поля: ФИО, адрес, возраст, почта, и изображение получателя.

# **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 C#, с использованием веб-фреймворка ASP .NET Core.
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://veglem.github.io/rip-frontend-repository/orders.
6. В веб-сервис добавлена авторизация через сессии, а методы задокументированы через Swagger.
7. Реализован интерфейс пользователя. Доступ к нему имеют только авторизированные пользователи.
8. Выделенный сервис был разработан и развернут в отдельном виртуальном окружении Python.
9. Реализован интерфейс модератора для подтверждения новых уведомлений и редактирования/добавления получателей.
10. Было реализовано мобильное 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/veglem/rip-backend-2023.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

* 1. Проблемы и возможности автоматизации документооборота в современных ВУЗах [Электронный ресурс] // na-journal. URL: https://na-journal.ru/6-2023-informacionnye-tekhnologii/5837-problemy-i-vozmojnosti-avtomatizacii-dokumentooborota-v-sovremennyh-vuzah
  2. Как создать приложение экстренного оповещения для безопасной жизни, используя No-Code [Электронный ресурс] // AppMaster. URL: https://appmaster.io/ru/blog/kak-sozdat-prilozhenie-ekstrennogo-opoveshcheniia
  3. Руководство по C# [Электронный ресурс] // Microsoft. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/
  4. Документация по ASP.NET [Электронный ресурс] // Microsoft. URL: https://learn.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-8.0
  5. Руководство по WebApi [Электронный ресурс] // Microsoft. URL: https://learn.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio (дата обращения: 14.10.2023)
  6. Полное практическое руководство по Docker [Электронный ресурс] // Habr. URL: https://habr.com/ru/articles/310460/ (дата обращения: 10.10.2023).
  7. Руководство по React [Электронный ресурс] // Metanit. URL: https://metanit.com/web/react/ (дата обращения: 11.11.2023).
  8. Quick Start – React [Электронный ресурс] // React. URL: https://react.dev/learn (дата обращения: 11.11.2023).
  9. Руководство по React Bootstrap [Электронный ресурс] // React Bootstrap. URL: <https://react-bootstrap.netlify.app/docs/getting-started/introduction>
  10. Руководство по React Native [Электронный ресурс] // React Nativ. URL: <https://reactnative.dev/docs/getting-started>
  11. Руководство по PostgreSQL [Электронный ресурс] // Metanit. URL: https://metanit.com/sql/postgresql/ (дата обращения: 05.10.2023).

**Приложение. Техническое задание**

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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: «Приказы ректора»

Студент: Журмилов В.Д.

Группа ИУ5-53Б

Преподаватель: Канев А.И.

Москва, 2024

# **Цель**

Разработка системы электронных уведомлений, состоящей из веб-сервиса, веб-приложения, мобильного приложения и асинхронного сервиса для обеспечения эффективной работы, и координации системы рассылок уведомлений.

# **Назначение**

Система предназначена для пользователей и модераторов. Система предоставляет пользователю интерфейс для просмотра и управления отправкой электронных уведомлений получателям. Администраторы могут взаимодействовать с базой данных и выполнять модерацию уведомлений и получателей. Главная задача системы – автоматизация процесса создания уведомлений и обеспечение бесперебойной рассылки получателям.

# **Задачи:**

* 1. Разработать дизайн приложения.
  2. Создать базу данных в PostgreSQL.
  3. Создать веб-сервис на языке C#.
  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 пользователя в React.
  8. Реализовать интерфейс модератора в React.
  9. Создать мобильное приложение на React Native.
  10. Создать асинхронный сервис для отслеживания отправки уведомления на Golang.
  11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

# **Методы веб-сервиса**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Метод | URL | Описание | Входные данные | Выходные данные |
| Методы подразделений | | | | | |
| 4.1.1 | GET | api/units/<id>/image | Возвращает изображение из файлового хранилища сервиса. id – идентификатор подразделения. | id: int | Файл |
| 4.1.2 | GET | api/units | Возвращает список подразделений, удовлетворяющих переданному фильтру и черновое уведомление пользователя. | filter: string | {  units: [  {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  }  ],  draft: int  } |
| 4.1.3 | POST | api/units/create | Добавление пустого подразделения.  Возвращает созданное подразделение.  Доступно только модератору |  | [  {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  }  ] |
| 4.1.4 | GET | api/units/<id> | Возвращает подразделение по уникальному id | id:int | {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  } |
| 4.1.5 | PUT | api/units/<id>/update | Обновляет информацию о подразделении.  Доступно только модератору. Возвращает обновленное подразделение. | {  name: string,  description: string  } | {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  } |
| 4.1.6 | DELETE | api/units/<id>/delete | Меняет статус подразделения на «Удален».  Возвращает список доступных подразделений.  Доступно только модератору | id: int | [  {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  }  ] |
| 4.1.7 | POST | api/units/<id>/add\_to\_order | Добавляет в черновой приказ пользователя подразделение {id}.  Возвращает список подразделений в черновом приказе.  Доступно только авторизированным пользователям | id: int | [  {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  }  ] |
| 4.1.8 | PUT | api/units/<id>/image | Устанавливает новое изображение.  Доступно только модератору | Файл |  |
| Методы приказов | | | | | |
| 4.2.1 | GET | api/orders | Возвращает список приказов. Возможна фильтрация по дате и статусу.  Доступно только авторизованным пользователям | {  status: int, startDate: datetime,  endDate: datetime  } | [  {  id: int,  name: string,  orderBody: string,  creationDate: datetime  formationDate: datetime,  endDate: datetime,  sign: string,  creator: string,  moderator: string,  units: [  {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  }  ],  status: int  }  ] |
| 4.2.2 | GET | api/orders/<id>/ | Возвращает информацию о приказе и его подразделениях. Доступно авторизованным пользователям | id:int | {  id: int,  name: string,  orderBody: string,  creationDate: datetime  formationDate: datetime,  endDate: datetime,  sign: string,  creator: string,  moderator: string,  units: [  {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  }  ],  status: int  } |
| 4.2.3 | PUT | api/orders/update | Обновляет приказ.  Возвращает обновленный приказ.  Доступно авторизованным пользователям | name: string,  description: string | {  id: int,  name: string,  orderBody: string,  creationDate: datetime  formationDate: datetime,  endDate: datetime,  sign: string,  creator: string,  moderator: string,  units: [  {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  }  ],  status: int  } |
| 4.2.4 | DELETE | api/orders/<id>/delete | Меняет статус заявки на «Удален».  Возвращает только что удаленный приказ.  Доступно авторизованным пользователям | id: int | {  id: int,  name: string,  orderBody: string,  creationDate: datetime  formationDate: datetime,  endDate: datetime,  sign: string,  creator: string,  moderator: string,  units: [  {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  }  ],  status: int  } |
| 4.2.5 | DELETE | api/orders/<id>/delete\_unit/<id> | Удаление подразделения из приказа. Доступно авторизованным пользователям.  Возвращает обновлённый приказ. | id: int | {  id: int,  name: string,  orderBody: string,  creationDate: datetime  formationDate: datetime,  endDate: datetime,  sign: string,  creator: string,  moderator: string,  units: [  {  id: int,  name: string,  imgUrl: string,  status: int,  description: string  }  ],  status: int  } |
| 4.2.6 | PUT | api/orders/<id>/update\_status\_moderator | Меняет статус с «Сформирован» на «Отклонен» или «Завершен».  Возвращает обновленный приказ.  Доступно модератору | id: string  status: int | {  id: int,  name: string,  orderBody: string,  creationDate: datetime  formationDate: datetime,  endDate: datetime,  sign: string,  creator: string,  moderator: string,  units: [  {  id: 0,  name: string,  imgUrl: string,  status: int,  description: string  }  ],  status: int  } |
| 4.2.7 | PUT | api/orders/<id>/update\_status\_user | Меняет статус с «Черновик» на «Сформирован».  Доступно авторизованным пользователям | id: int | {  id: int,  name: string,  orderBody: string,  creationDate: datetime  formationDate: datetime,  endDate: datetime,  sign: string,  creator: string,  moderator: string,  units: [  {  id: 0,  name: string,  imgUrl: string,  status: int,  description: string  }  ],  status: int  } |
| 4.2.8 | PUT | api/orders/<id>/update\_signature | Устанавливает подпись ректора в приказ. Вызывается внешним сервисом. | id: int,  signature: bool,  access\_token: string, |  |
| Методы внешнего сервиса | | | | | |
| 4.3.1 | POST | calc\_sig | Подписывает приказы.  Вызывается основным сервисом | id: int |  |
| Методы пользователя | | | | | |
| 4.4.1 | POST | api/auth/signin | Регистрация | username: string  password: string |  |
| 4.4.2 | POST | api/auth/login | Авторизация | username: string  password: string |  |
| 4.4.3 | POST | api/auth/logout | Выход из аккаунта. Доступно только авторизированным пользователям |  |  |
| 4.4.4 | GET | api/auth | Возвращает информацию о пользователе.  Доступно только авторизированным пользователям |  | {  username: string,  isModerator: bool,  fio: string,  imageUrl: string  } |
| 4.4.5 | PUT | api/auth/update | Меняет информацию о пользователе.  Доступно только авторизированным пользователям | {  fio: string,  username: string,  newPassword: string,  oldPassword: string  } |  |
| 4.4.6 | GET | api/auth/image | Возвращает изображение пользователя из файлового хранилища сервиса.  Доступно только авторизированным пользователям |  | Файл |
| 4.4.7 | PUT | api/auth/image | Устанавливает новое изображение пользователя.  Возвращает ссылку на новое изображение.  Доступно только авторизированным пользователям | Файл | {  imageUrl: string  } |

1. **Функциональные требования:**
   1. Главное меню. Здесь находятся кнопки для навигации и авторизации или информация о текущем пользователе, и меню со списком доступных подразделений.
      1. Доступна всем пользователям.
      2. Действия
         1. Войти – переход на страницу 5.2. Только для гостей.
         2. Открыть список подразделений – переход на страницу 5.4. Для всех пользователей.
         3. Открыть список приказов – переход на страницу 5.6. Только для авторизированных пользователей.
         4. Управление подразделениями – переход на страницу 5.9. Только для модераторов.
   2. Страница с формой авторизации. На этой странице находится форма авторизации.
      1. Доступна гостям
      2. Действия
         1. Войти – производит запрос (метод 4.4.2).
         2. Зарегистрироваться – перенаправляет на страницу 5.3.
   3. Страница с формой регистрации. На этой странице находится форма авторизации.
      1. Доступна гостям
      2. Действия
         1. Зарегистрироваться – производит запрос (метод 4.4.1.), в котором передаются введенные данные формы.
         2. Войти - перенаправляет пользователя на страницу 5.2.
   4. Страница со списком подразделений. На этой странице располагается список всех подразделений и панель фильтрации.
      1. Доступна всем пользователям.
      2. Выводится информация о получателях в виде карточек (метод 4.1.2.).
         1. Изображение подразделения.
         2. Название подразделения.
      3. Действия
         1. Отфильтровать подразделения по названию (метод 4.1.2.).
         2. «Корзина» - перенаправляет на страницу приказа-черновика 5.7.
         3. Добавить подразделение в приказ (метод 4.1.7.). Только для авторизированных пользователей.
         4. Подробнее – перенаправляет на страницу 5.5.
   5. Страница с подробным описанием подразделения
      1. Доступна всем пользователям.
      2. Выводится подробная информация о подразделении.
         1. Изображение подразделения.
         2. Название подразделения.
         3. Описание подразделения.
   6. Страница со списком приказов
      1. Доступна только авторизированным пользователям.
      2. Выводится информация о приказах в табличном формате (метод 4.2.1).
         1. Создатель. Только для модераторов.
         2. Статус приказа.
         3. Статус подписания ректором.
         4. Дата формирования.
         5. Дата завершения.
      3. Действия
         1. Открыть подробную информацию об приказе – перенаправляет на страницу 5.7.
         2. Установить приказу статус «отклонено» или «завершено» (метод 4.2.6.). Только для модераторов.
         3. Отфильтровать список приказов по диапазону дат формирования и/или статусу (метод 4.2.1) и/или создателю (только для модератора).
   7. Страница с подробным описанием приказа. Пользователь может изменить набор подразделений в приказе. Модератор может просматривать приказы всех пользователей.
      1. Доступна только авторизированным пользователям.
      2. Выводится информация о приказе в карточной форме. Список подразделений, в виде карточек (метод 4.2.2.).
         1. Изображение подразделения.
         2. Описание подразделения.
         3. Название подразделения.
      3. Действия
         1. Удалить подразделение из приказа (метод 4.2.5.). Если приказ находится в статусе черновика. Только для создателя приказа.
         2. Сформировать приказ (метод 4.2.7.). Если приказ находится в статусе черновика. Только для создателя приказа.
         3. Удалить приказ (метод 4.2.4.). Если приказ находится в статусе черновика. Только для создателя приказа.
         4. Сохранить название и описание приказа (метод 4.2.3). Доступно в режиме редактирования.
         5. Открыть подразделение – перенаправляет на страницу 5.5.
   8. Страница редактирования/создания подразделения. Модератор может изменить существующего или создать нового получателя.
      1. Доступна только модераторам.
      2. Выводится информация о получателе в построчном формате.
         1. Изображение подразделения.
         2. Название подразделения.
         3. Описание подразделения.
      3. Изменение полей подразделения. Можно изменять все вышеперечисленные поля подразделения (методы 4.1.3, 4.1.5 и 4.1.8).
   9. Страница с таблицей подразделений. Предоставляет модератору удобный способ отображения всех подразделений.
      1. Доступна только модераторам.
      2. Выводит информацию о подразделениях в табличном виде (метод 4.1.2).
         1. Изображение подразделения.
         2. Название подразделения.
         3. Описание подразделения.
      3. Действия
         1. Удалить подразделение (метод 4.1.6)
         2. Редактировать подразделение – перенаправляет на страницу 5.8
         3. Добавить подразделение – перенаправляет на страницу 5.8
2. **Требования к программному обеспечению:**
   1. Серверная часть
      1. ОС: Linux 6.5 / Windows 10
      2. Docker 24.0.6
      3. Python 3.11.5
      4. Golang 1.21.1
      5. Докер образы
      6. redis 6.2-alpine
      7. nginx 1.19.2-alpine
      8. minio RELEASE.2022-10-15T19-57-03Z
      9. postgres 16.1
   2. Клиентская часть
      1. ОС: Windows/MacOS/Linux
      2. Веб-браузер: Safari 11.1+/Chrome 40+/Opera 27+/Firefox 44+/Yandex 20+
3. **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.