|  |  |
| --- | --- |
|  | **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: «Электронные уведомление»

Студент: Балюк А.В.

Группа ИУ5-53Б

Преподаватель: Канев А.И.

2023г.

# **ВВЕДЕНИЕ**

В век электронных уведомлений уже мало кто пользуется записками. Также люди не хотят тратить свое время на личные встречи для передачи небольшого объема информации. Все действия происходят в сети, и иногда другому пользователю (друг, подчинённый и т.д.) необходимо быстро уведомить о чем-то. В связи с этим было решено разработать систему для быстрой и удобной отправки электронных уведомлений нужным людям, которая облегчит процесс взаимодействия людей внутри сети.

Целью работы является реализация системы для отправки электронных уведомлений выбранным пользователям, включающую в себя веб-сервис, веб-приложение, мобильное приложение и выделенный сервис для отслеживания статуса отправки уведомления.

Система предназначена для простых пользователей и модераторов сервиса. Для отправки уведомления пользователю необходимо создать новое черновое уведомление. Система предоставляет автоматизированный способ создания учета и ведения уведомлений. Модераторы имеют возможность подтверждать или отклонять уведомления, а также редактировать и создавать новых получателей.

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. Реализовать REST-API на GO.
  4. Реализовать интерфейс гостя на технологии React.
  5. Развернуть веб-приложение React на Github Pages.
  6. Добавить авторизацию, аутентификацию и Swagger в веб-сервис.
  7. Реализовать интерфейс пользователя в React.
  8. Реализовать интерфейс модератора React.
  9. Создать мобильное приложение на React-Native.
  10. Создать асинхронный сервис для отслеживания статуса отправки уведомления на Django.
  11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

# **БИЗНЕС-ПРОЦЕСС**

Человеку может быть необходимо срочное о чем-то уведомить большое количество людей. Для того, чтобы не искать этих людей в живую и тратить на это много времени, пользователи могут создавать уведомления для рассылки сразу всем. Каждое уведомление вначале создается как черновик, в который можно добавлять и из которого можно удалять получателей. В одном уведомлении можно указать сразу несколько получателей, и тогда уведомление отправится сразу всем людям.

Когда пользователь удостовериться, что выбраны все нужные получатели, он формирует уведомление и оно отправляется на модерирование. После этого его уже нельзя редактировать. Можно также посмотреть историю своих уведомлений. В уведомлении необходимо четко и правильно указывать его тип, так как это может привести к дезинформации и, как следствие, время и ресурсы людей могут быть потрачены впустую. К выбору получателей также стоить отнестись с внимательностью. Для этого модераторы отклоняют и подтверждают уведомления таким образом, чтобы уведомление с четко прописанным типом получили нужные люди, которые и должные взаимодействовать между собой. На случай, если пользователь передумал отправлять уведомление или допустил ошибку присутствует возможность отменить отправку.

Получатель – это конкретный человек со своими личными данными. У каждого разная почта, ФИО, адрес и т.д. Также у получателей есть личное изображение, по которому сразу можно узнать необходимого человека. В описании получателей могут присутсвовать какие-то ошибки, и в этом случае администратор может либо на время, либо навсегда скрыть этого получателя. Если у человека поменялись личные данные, то модератор может внести соответствующие изменения в системе. Он может отредактировать какого-либо получателя, обновить фото, или исправить какие-то ошибки. Также модератор может создать нового получателя. Функции пользователей с разными полями описаны на диаграммах прецедентов (рис. 1).

Гостям доступен просмотр получателей. Зарегистрированные гости – пользователи. Они могут добавлять получателей в уведомление, просматривать список своих уведомлений и сформировать текущее уведомление. Уведомления обрабатываются модераторами. В результате обработки уведомления, его либо подтверждают, либо отклоняют. Также модератору доступны уникальные функции для работы с получателями, а именно: просмотр всех получателей, создание и удаление получателей, а также просмотр всех получателей в табличном виде. Процесс формирования уведомления отражен на диаграмме деятельности (рис. 2).

Пользователь выбирает получателей, затем формирует на основе выбранных получателей уведомление. Эту заявку обрабатывает асинхронный сервис, а затем и модератор. В случае одобрения для уведомления запускается процесс отслеживания отправки уведомления в асинхронном сервисе. Возможные состояния уведомления отражены на диаграмме состояний (рис. 3).

При выборе первого получателя формируется черновик. Последующие выбранные получатели добавляются в этот черновик. Пользователь затем формирует заявку, удаляет её или выходит из приложения. Сформированную заявку обрабатывает модератор иди администратор. Они могут одобрить или отклонить её.

# **АРХИТЕКТУРА**

Архитектура системы отображена на диаграмме развертывания (рис. 4). Часть бекенда разворачивается кластере docker контейнеров. 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. Кластер описан в docker-compose.yml файле. Асинхронный сервис разворачивается отдельно и необязательно на том же компьютере, что и кластер основных контейнеров.

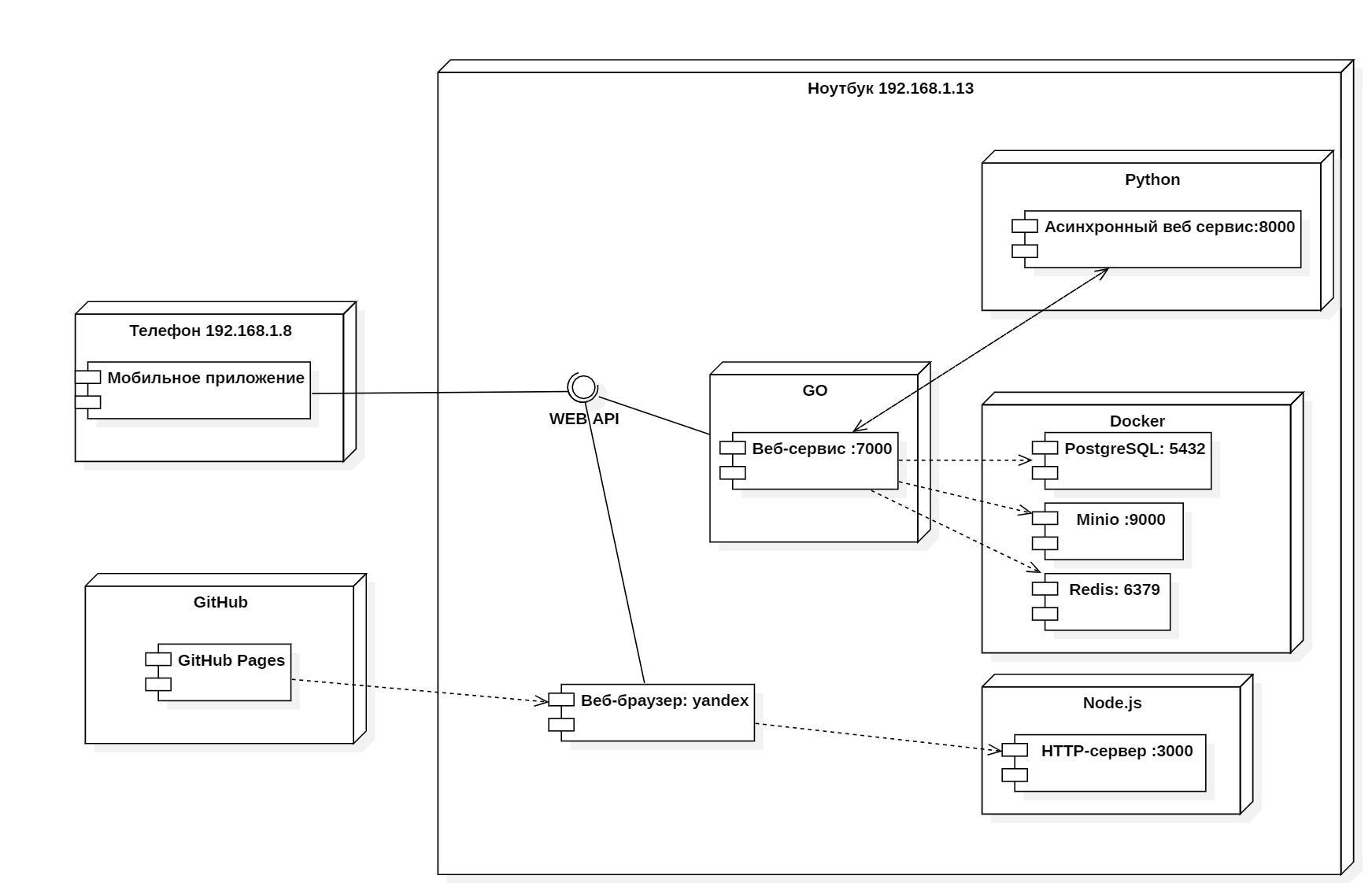


Рис. 4 - Диаграмма развертывания

Нативное [5] и браузерное приложения [6, 7] обращаются к веб-сервису на базе технологии GO 7 [8, 9] через REST Web-API. Использование GO обусловлено тем, что эта технология является проверенным и быстродействующим решением, стандартом индустрии. Современность используемого в ней языка C# также склоняет к выбору этой технологии.

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. Модель получателей представляет собой набор полей, необходимых исключительно для бизнес-логики. Для хранения в одной заявке нескольких карт используется промежуточная таблица «Получатели уведомлений» , реализующая связь М-М.

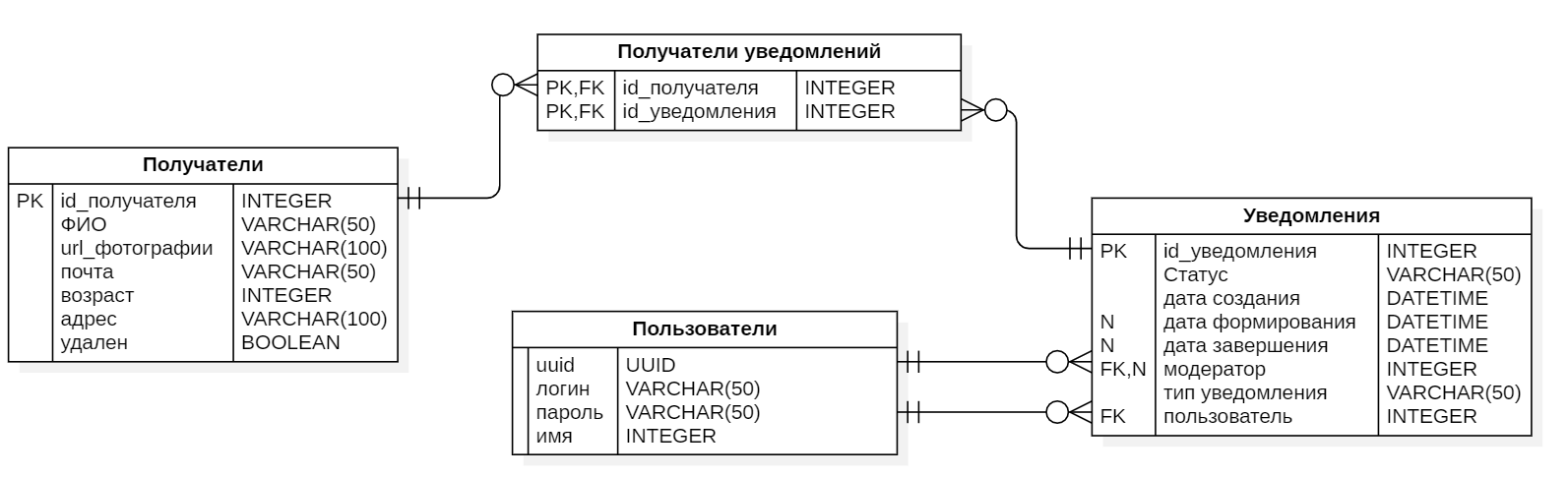


Рисунок 5 - ER диаграмма

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.

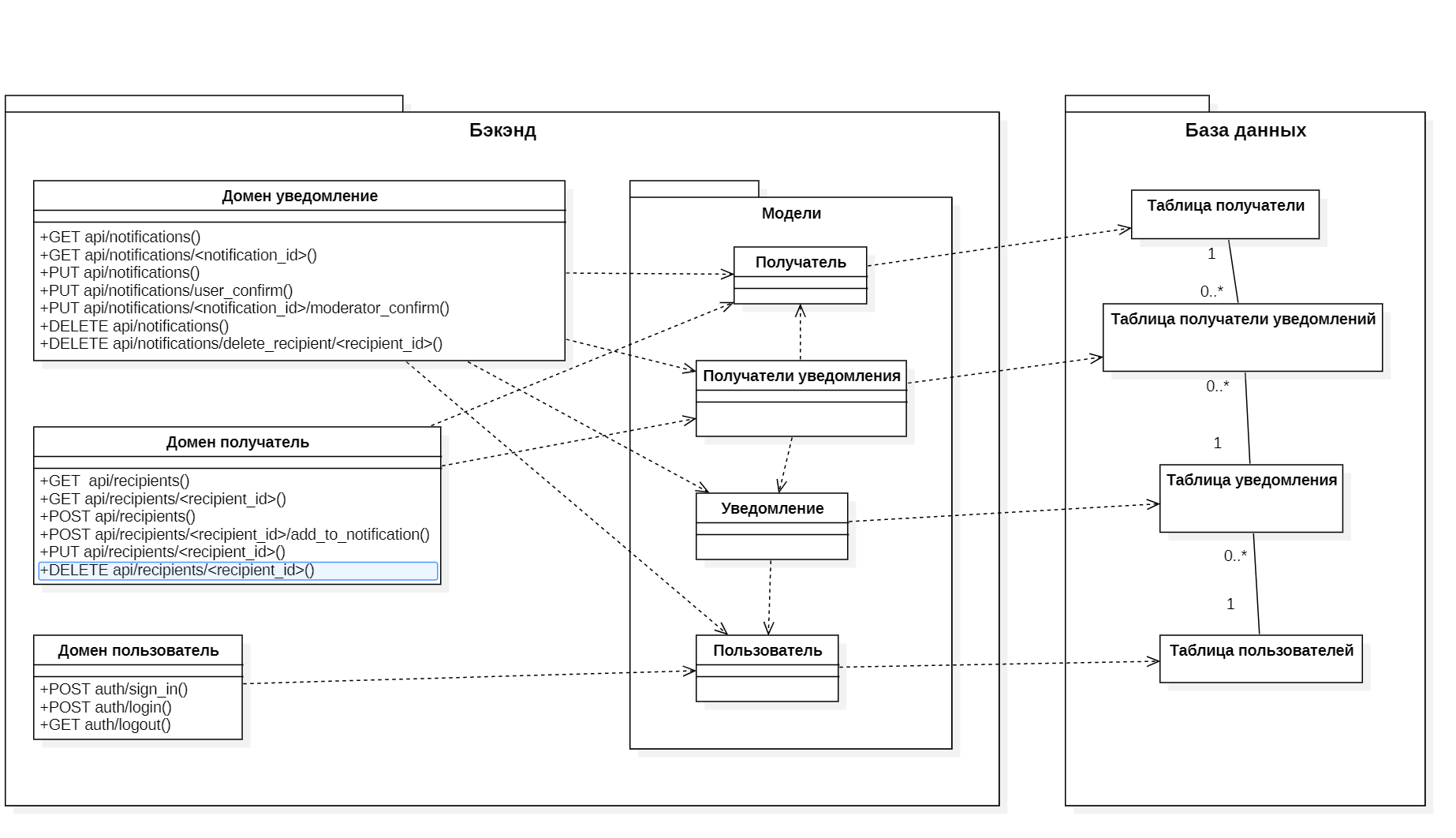


Рисунок 6 *-* Диаграмма классов бекенда

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).

# **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, пользователи и авторизация. Методы следуют правилам REST API.

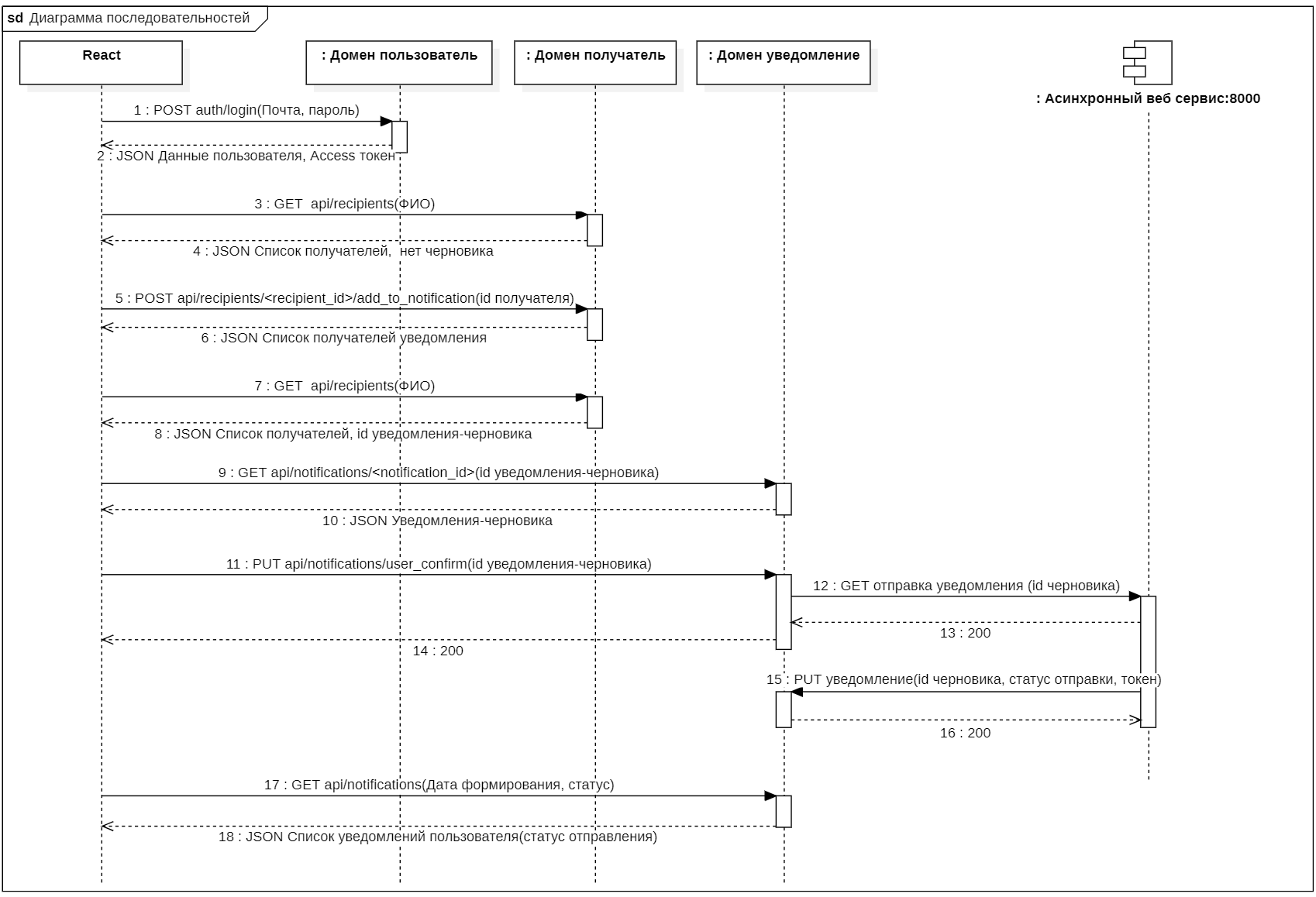


Рисунок 8 - Диаграмма последовательности

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 JWT 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 получателей, которые возвращаются в JSON формате. Пользователь выбирает получателя, 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 пользователь определится с выбором, он нажимает на кнопку «сформировать» в графическом интерфейсе. После этого приложение запрашивает id черновой заявки пользователя 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. Позже пользователь может увидеть, подтверждено ли его уведомление.

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 и диапазону дат, в котором 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 функция 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).

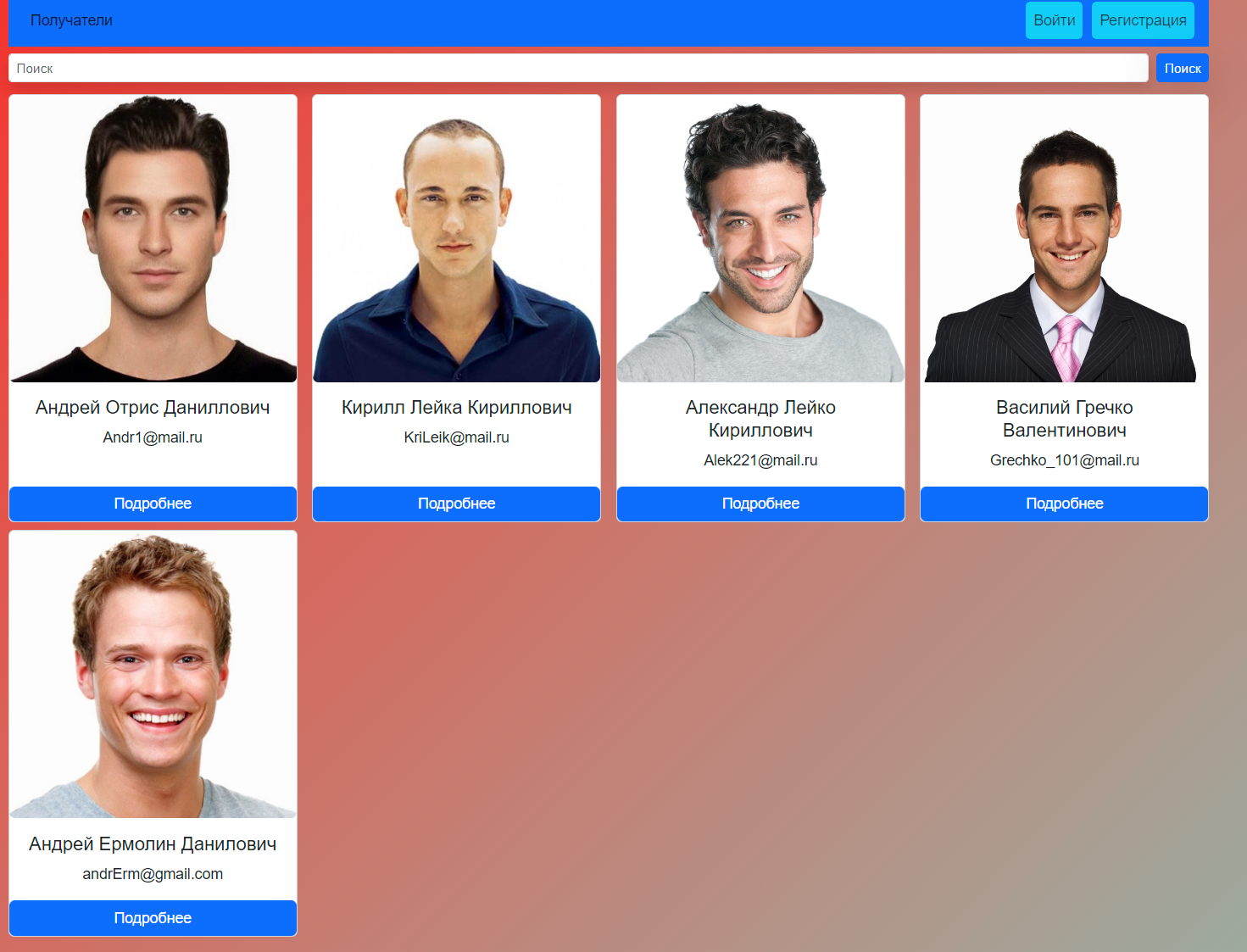


Рисунок 9 - Главное меню (неавторизированный пользователь)

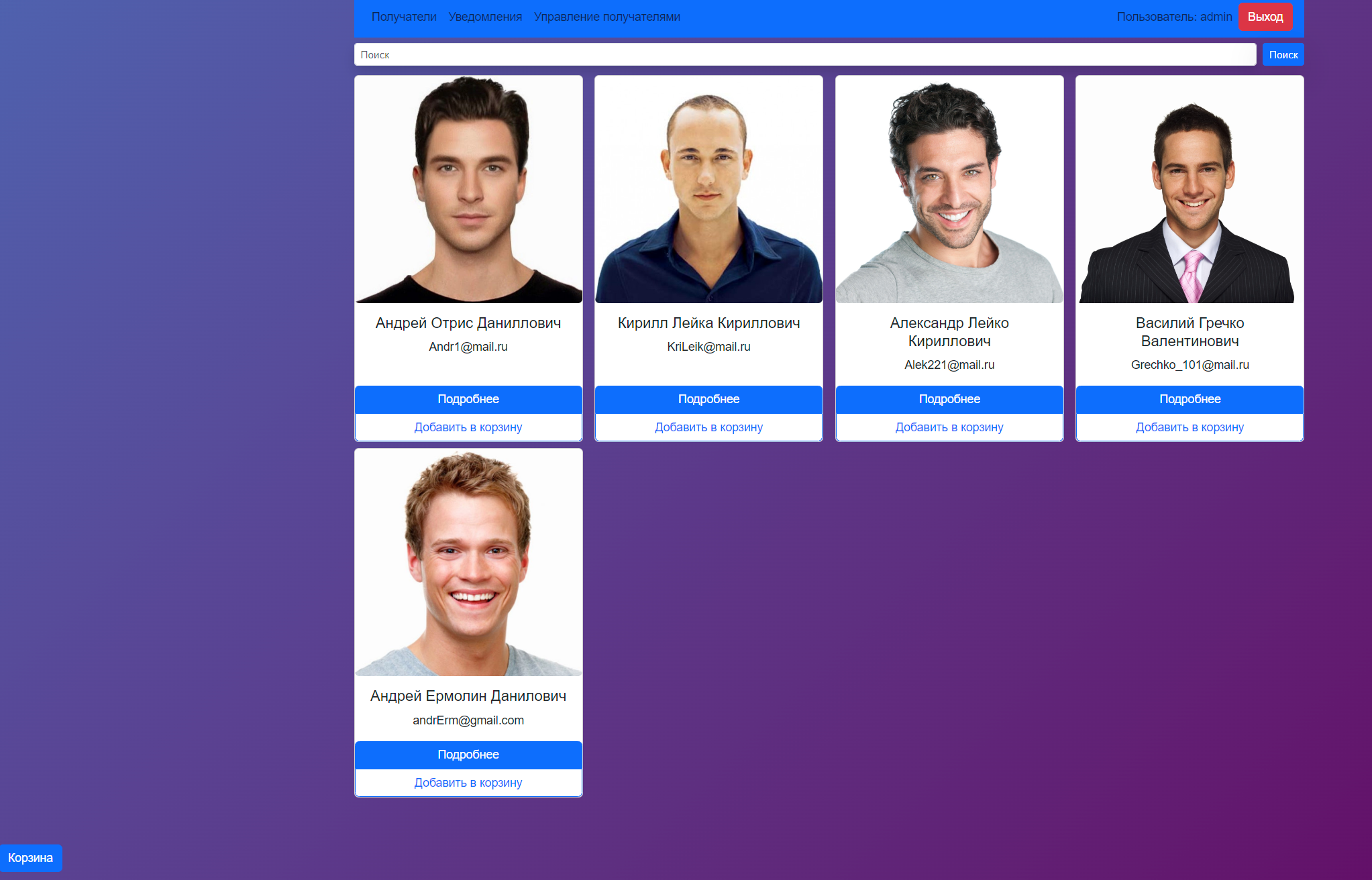


Рисунок 10 - Главное меню (модератор)

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 JWT, который сохраняется в local storage и используется при отправлении запросов.

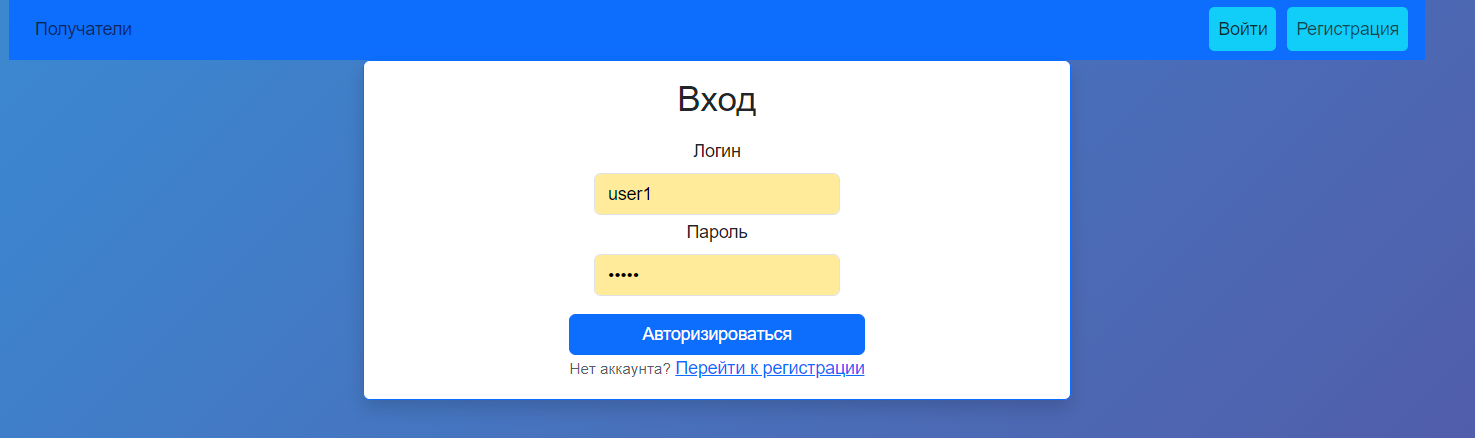


Рисунок 11 - Страница авторизации

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.

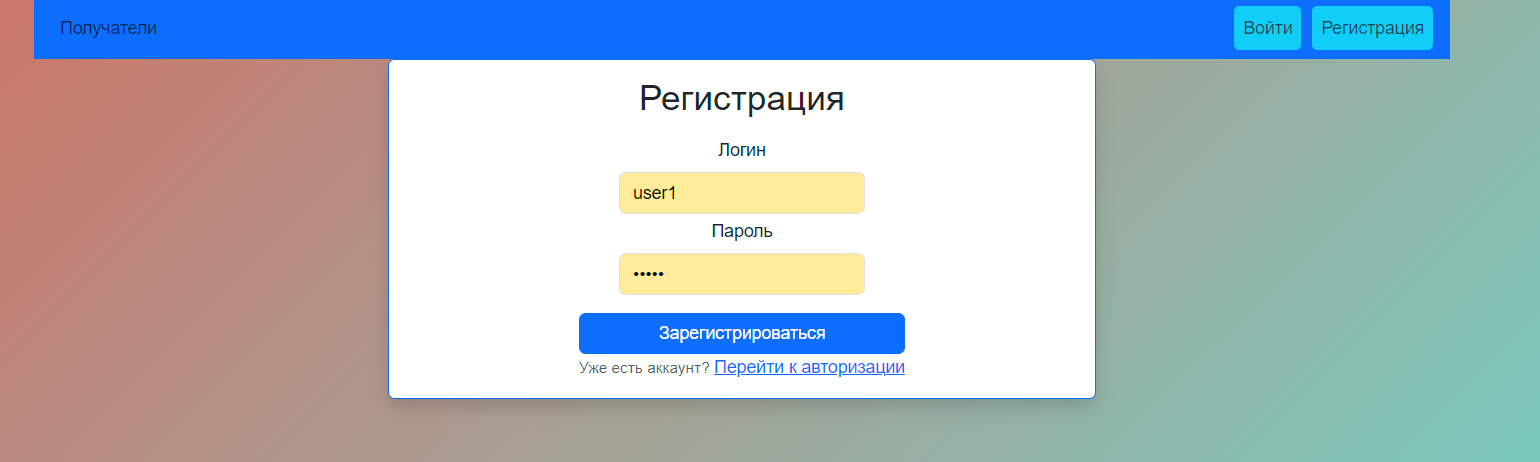


Рисунок 12 - Страница регистрации

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.

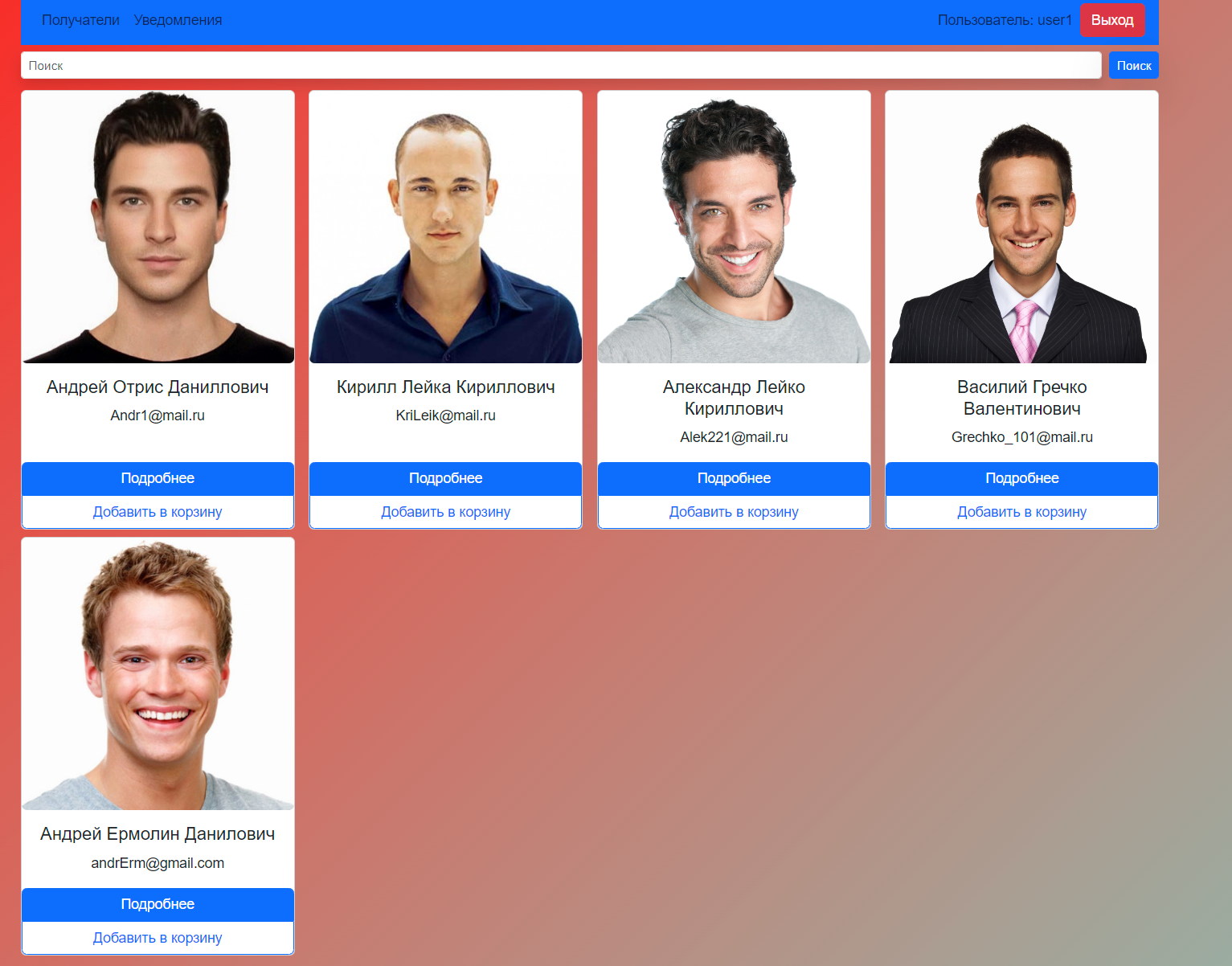


Рисунок 13 - Страница со списком получателей

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

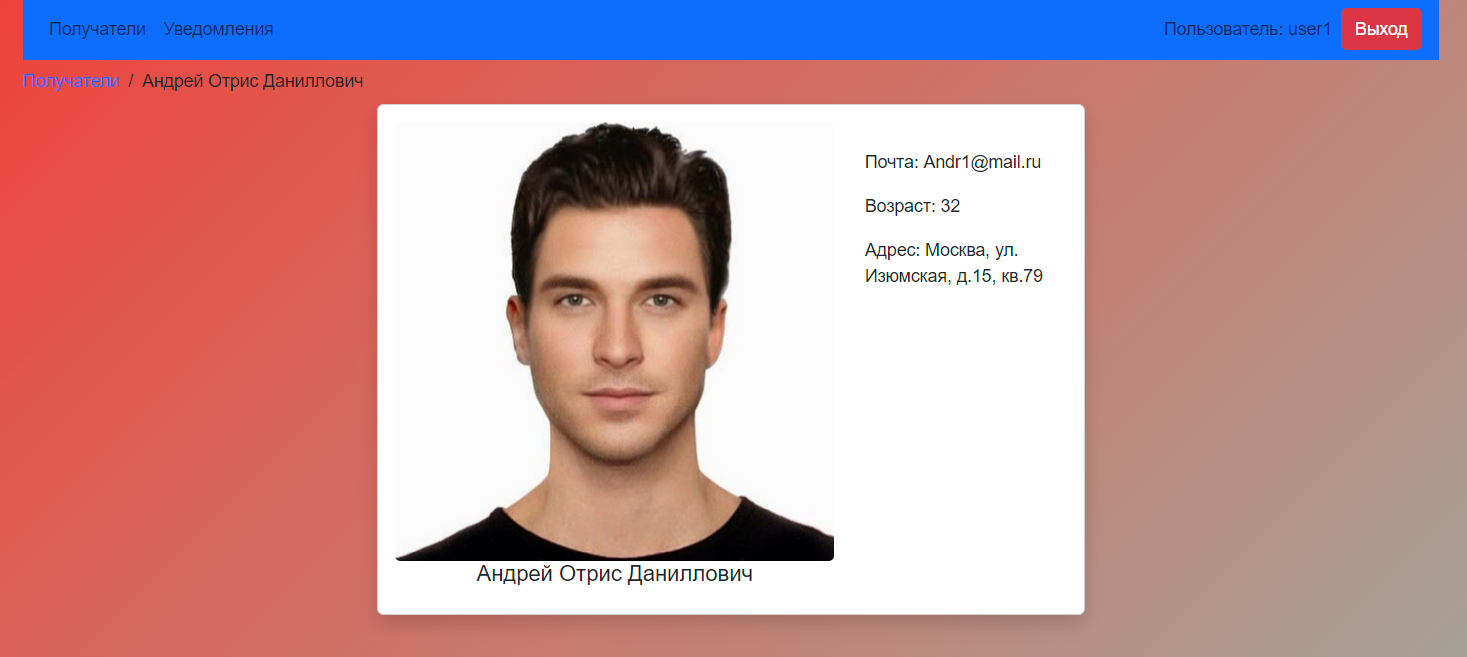


Рисунок 14 - Страница с подробным описанием получателя

На странице со списком уведомлений (рис. 15, 16) отображается список уведомлений. В зависимости от типа пользователя этот список будет функционально отличаться. Так, для пользователей 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.

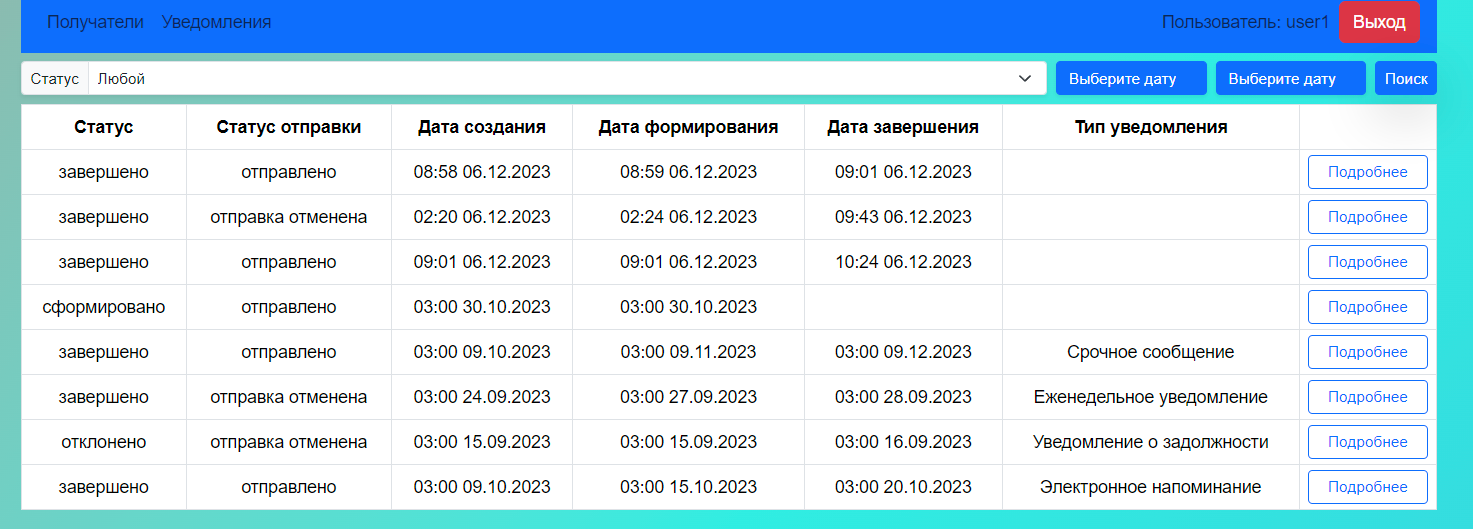


Рисунок 15 - Страница со списком уведомлений (пользователь)

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.

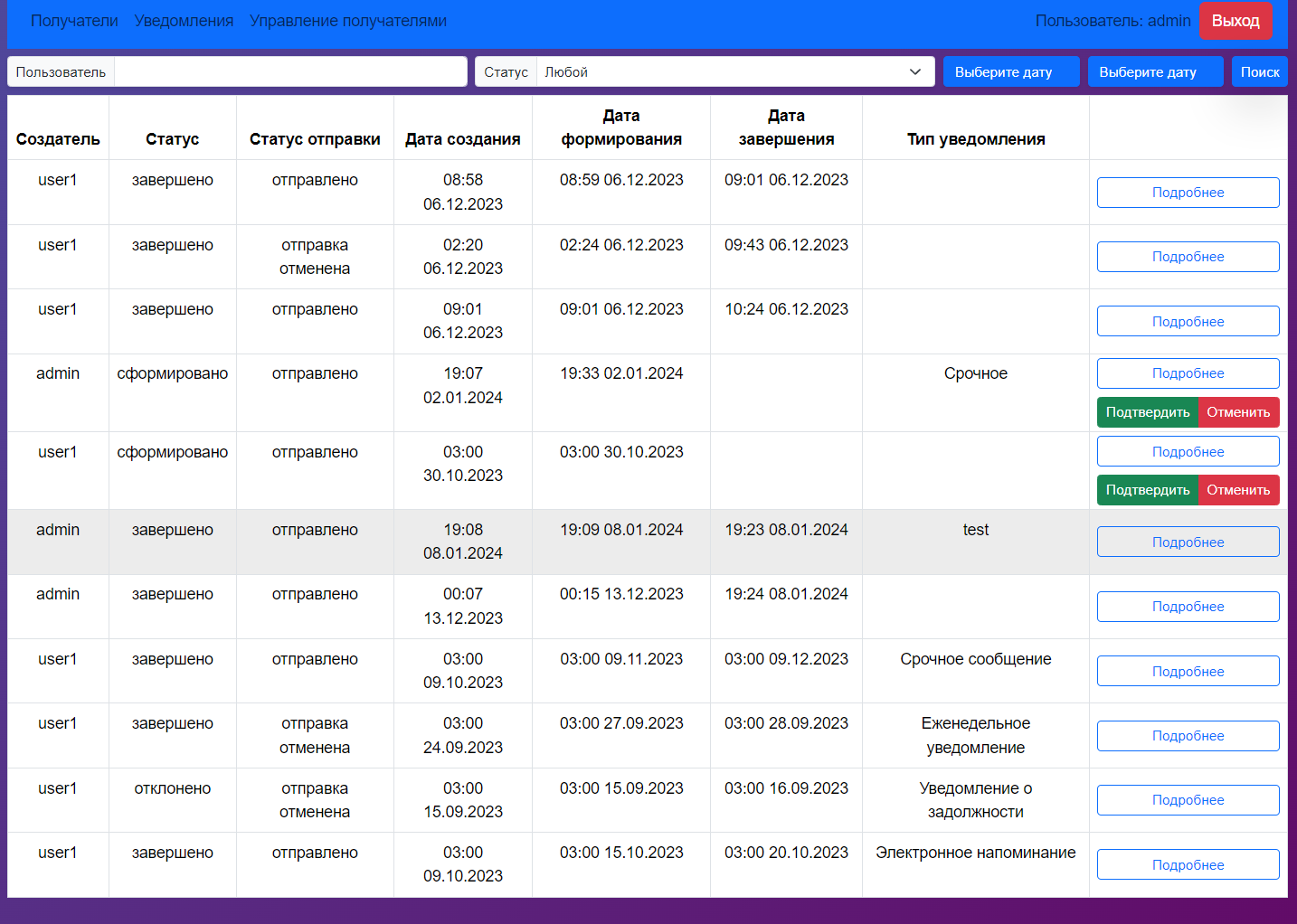


Рисунок 16 - Страница со списком уведомлений (модератор)

На странице с подробным описанием уведомления (рис. 17) отображается подробная информация об уведомлении. Список выбранных получателей в виде карточек, а также все поля уведомления. Для сформированных уведомлений есть кнопки «Подтвердить» и «Отменить» для соответствующих действий с заявкой.

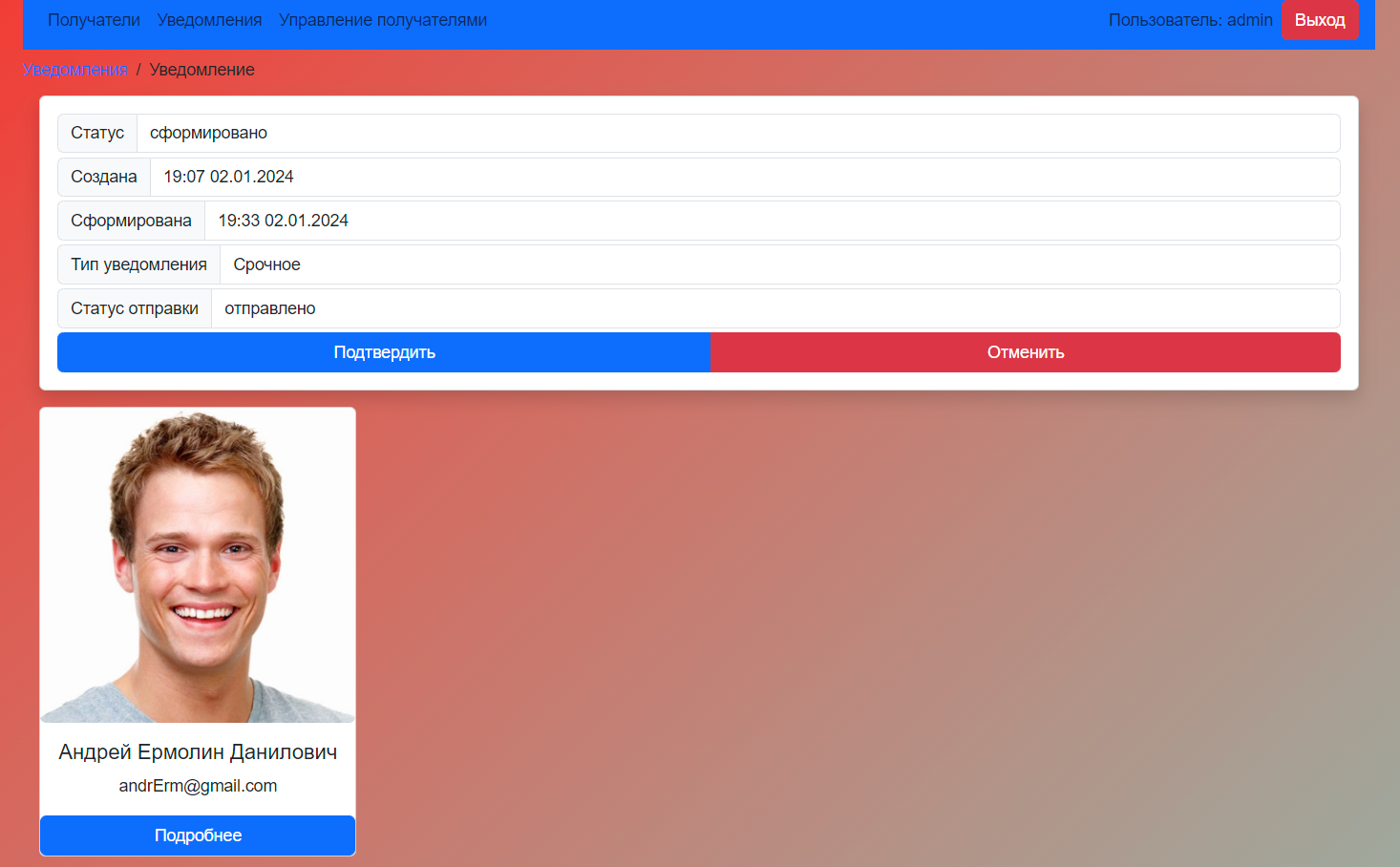


Рисунок 17 - Страница с подробным описанием уведомления

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 нажать на кнопки «Изменить» или «Удалить» для изменения (рис. 19), либо удаления получателя. Можно отфильтровать получателей по ФИО, либо добавить нового получателя (рис. 20).

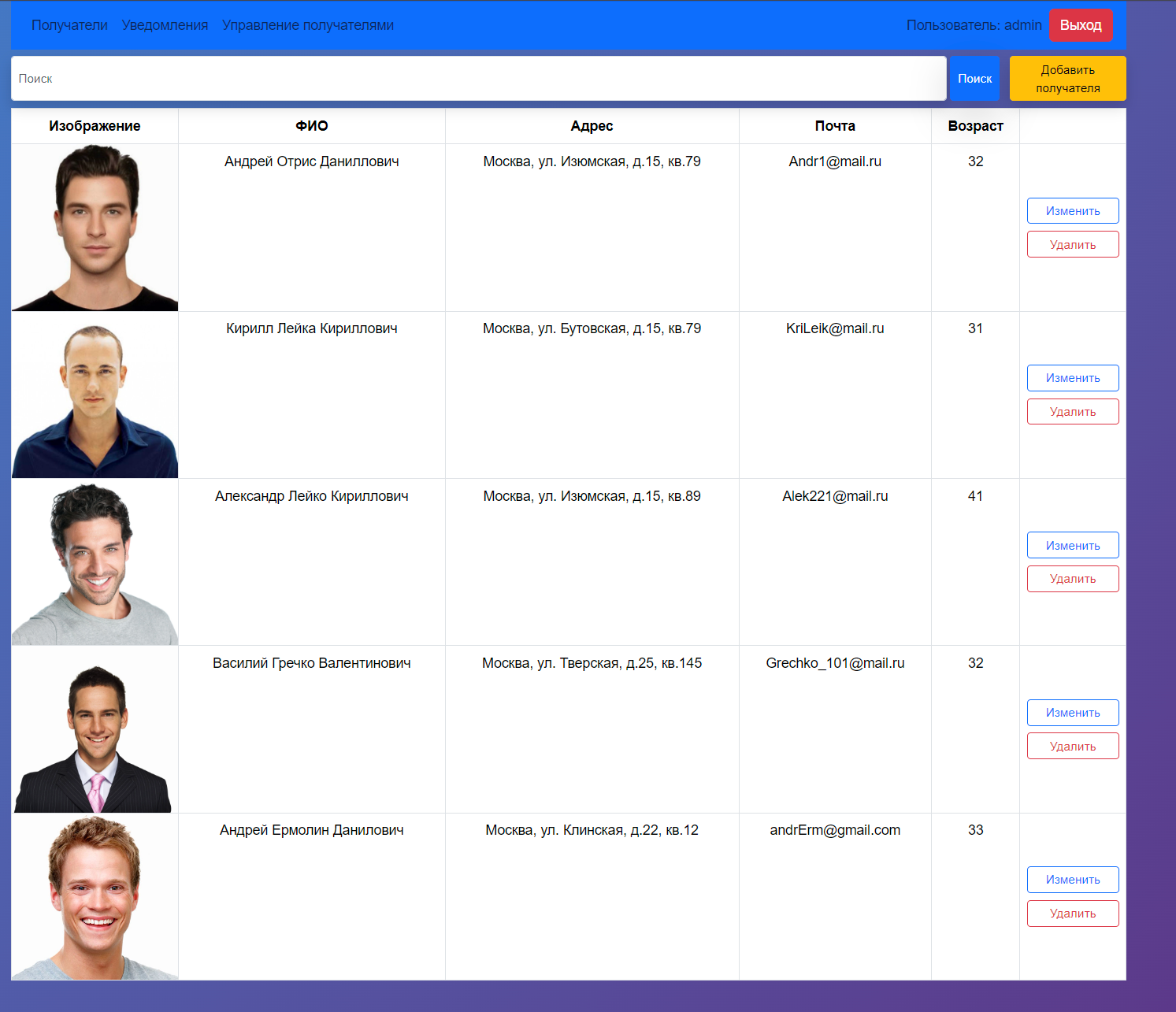


Рисунок 18 - Страница с таблицей получателей

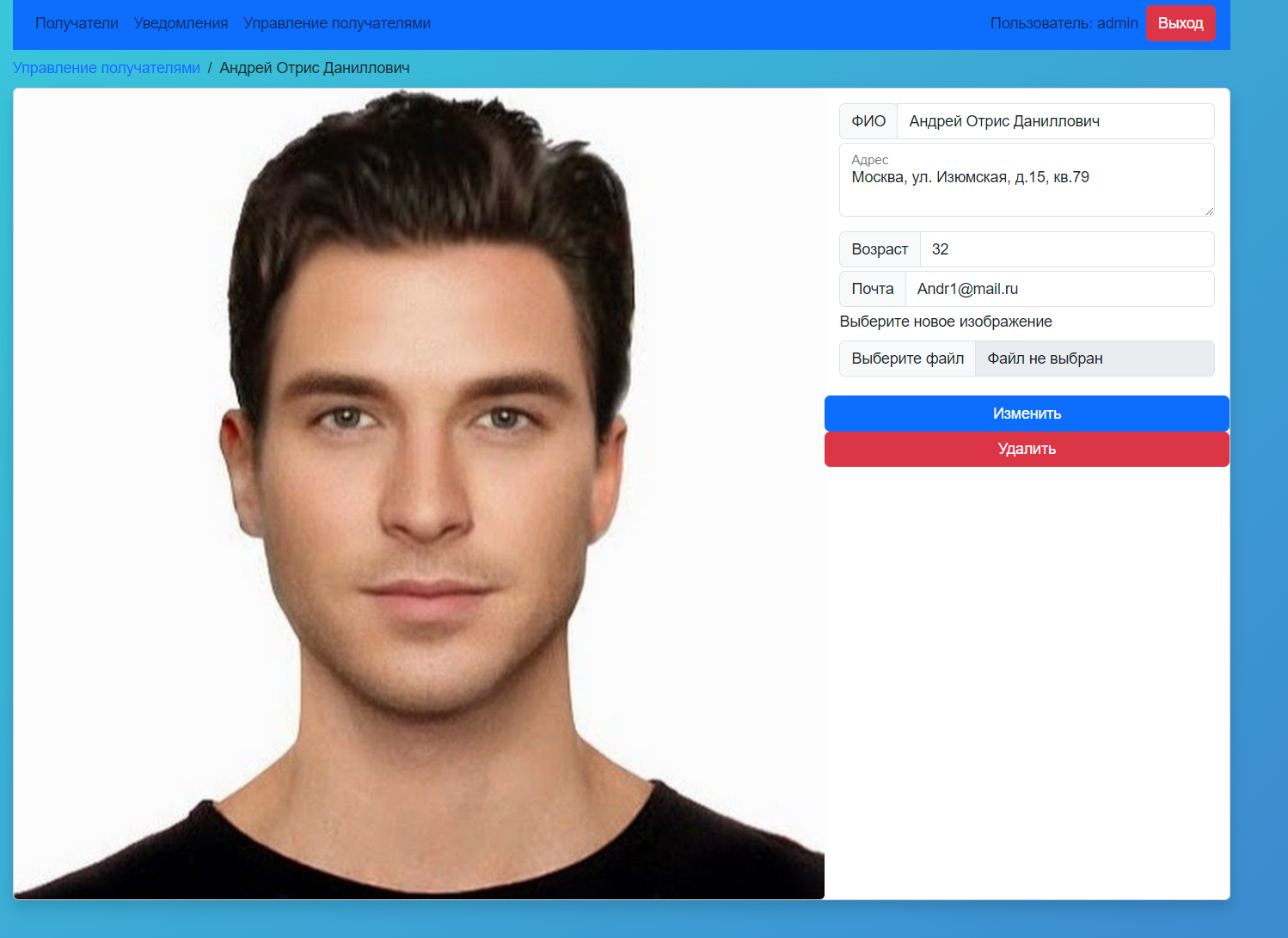


Рисунок 19 - Страница редактирования получателя

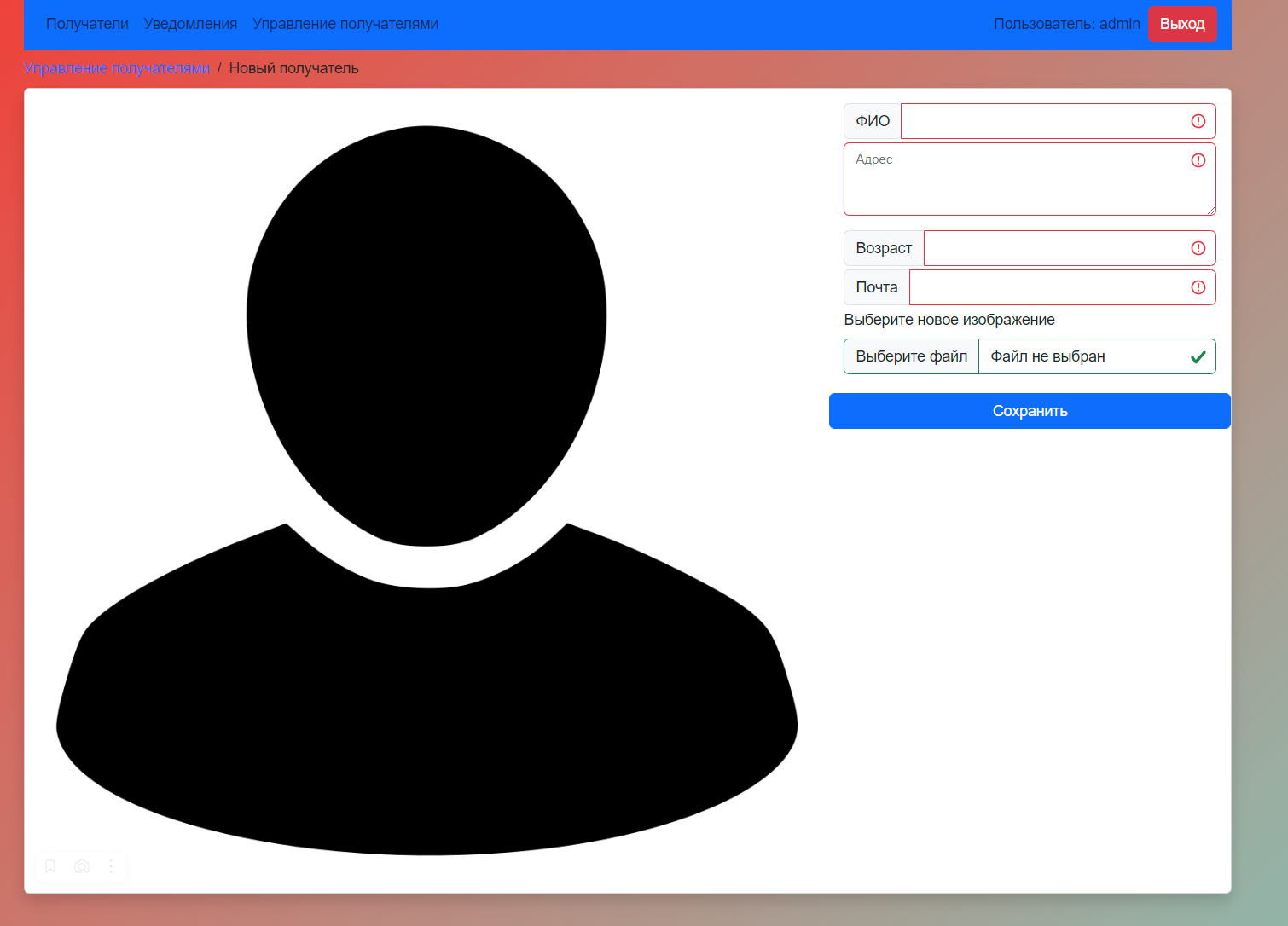


Рисунок 20 - Страница создания получателя

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

На страницу создания получателя (рис. 20) модератор может создать нового получателя. Для заполнения 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 GO с REST\_API.
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://qaleka.github.io/R\_I\_P\_labs\_front/recipients.
6. В веб-сервис добавлена авторизация через JWT, 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 docker кластере.
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/Qaleka?tab=repositories.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

* 1. Нескучный матчмейкинг без дисбаланса и очередей: практическое руководство [Электронный ресурс] // Habr. URL: https://habr.com/ru/companies/pixonic/articles/475548/ (дата обращения 02.10.2023).
  2. Data Structures and Algorithms for Game Developers [Книга] // Allen Sherrod. (дата обращения 04.10.2023).
  3. Rust – axum [Электронный ресурс] // Axum. URL: https://docs.rs/axum/latest/axum/ (дата обращения: 01.12.2023).
  4. Полное практическое руководство по Docker [Электронный ресурс] // Habr. URL: https://habr.com/ru/articles/310460/ (дата обращения: 20.10.2023).
  5. Tauri guide [Электронный ресурс] // Tauri. URL: https://tauri.app/v1/guides/ (дата обращения: 24.11.2023).
  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. Руководство по ASP.NET Core 7 [Электронный ресурс] // Metanit. URL: https://metanit.com/sharp/aspnet6/ (дата обращения: 04.10.2023).
  9. ASP.NET документация [Электронный ресурс] // Microsoft. URL: https://learn.microsoft.com/en-us/aspnet/core/ (дата обращения: 04.10.2023).
  10. Руководство по PostgreSQL [Электронный ресурс] // Metanit. URL: https://metanit.com/sql/postgresql/ (дата обращения: 05.10.2023).