|  |  |
| --- | --- |
|  | **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: «Анализ реликтового излучения»

Студент: Лавренов М.А.

Группа ИУ5-54Б

Преподаватель: Канев А.И.

2023г.

# **ВВЕДЕНИЕ**

Реликтовое излучение, или космический микроволновый фон (КМФ), представляет собой слабое излучение, охватывающее весь Вселенную, является своего рода "эхом" Большого взрыва. Оно наиболее ярко в микроволновом диапазоне электромагнитных волн. Исследователям нашей вселенной приходится непросто в том случае, если возникает необходимость в данных о конкретном спектре реликтового излучения. Возникает проблема получения необходимых для исследований данных. По этой причине было решено разработать систему для анализа реликтового излучения, которая сделает сканирование спектров реликтового излучения более простым и удобным.

Целью является разработка системы для заказов на космические аппараты сканирования реликтового излучения с выбранным спектром, включающую в себя веб-сервис, веб-приложение, десктопное приложение и выделенный сервис для сканирования.

Система предназначена для пользователей и модераторов сервиса сканирования. В системе предусмотрен доступ к выбору спектров для сканирования. Для получения доступа пользователю необходимо создать заявку на сканирование. Система предоставляет автоматизированный способ создания, учета и ведения заявок. Также она позволяет модераторам принимать или отклонять заявки.

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 Golang
  4. Реализовать интерфейс гостя на технологии React.
  5. Задеплоить на Github Pages.
  6. Добавить авторизацию и аутентификацию в веб-сервис.
  7. Реализовать интерфейс пользователя в React.
  8. Реализовать интерфейс модератора и администратора в React.
  9. Создать нативное IOS приложение на Swift.
  10. Создать выделенный сервис для расчета уровня игроков на Python Django.
  11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

# **БИЗНЕС-ПРОЦЕСС**

Существует много различных спектров, которые хотят отсканировать пользователи [1]. Для того, чтобы получить данные обо всех необходимых спектрах, пользователи могут оставлять заявки на сканирование. Каждая заявка вначале создается как черновик, в который можно добавлять и из которого можно удалять спектры. В одной заявке можно указать сразу несколько спектров.

Когда пользователь определится со спектрами, данные о которых ему необходимы, он формирует заявку и она отправляется на модерирование. После этого её уже нельзя редактировать. Можно также посмотреть историю своих заявок. На случай если пользователь передумает также присутствует возможность отменить заявку.

Спектры обладают разной длинной волны и частотой. Особенности спектра видны по его изображению и текстовому описанию. Во время работы системы на спектрах могут обнаружиться какие-то ошибки, и на этот случай модератор может временно скрыть спектр. Для увеличения разнообразия модерация может привнести изменения в спектры. Она может отредактировать какой-либо спектр, обновить его или исправить какие-то ошибки. Также можно создать совершенно новый спектр. Функции пользователей с разными ролями описаны на диаграммах прецедентов (рис. 1).

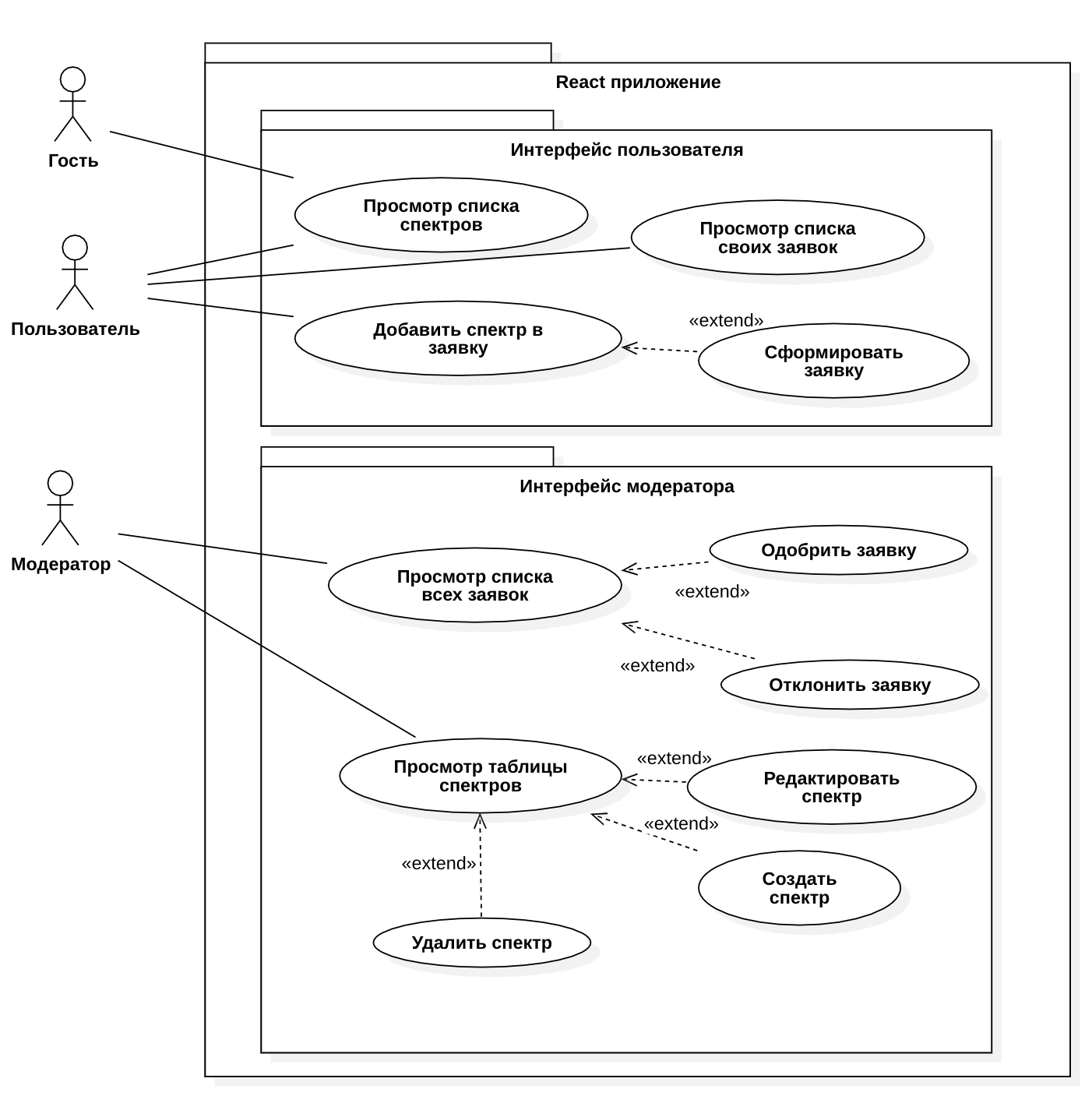


Рисунок 1 - Диаграмма прецедентов

Гостям доступен просмотр спектров. Зарегистрированные гости – игроки. Они могут добавлять спектры в заявку, просматривать список своих заявок и сформировывать текущую заявку. Заявки обрабатываются модераторами и администраторами. В результате обработки заявки её либо одобряют, либо отклоняют. Помимо возможностей модератора администратору также доступны уникальные функции для работы с картами, а именно: просмотр всех спектров, редактирование, создание и удаление спектров, а также просмотр списка всех спектров в табличном виде. Процесс оформления заявки отражен на диаграмме деятельности (рис. 2).

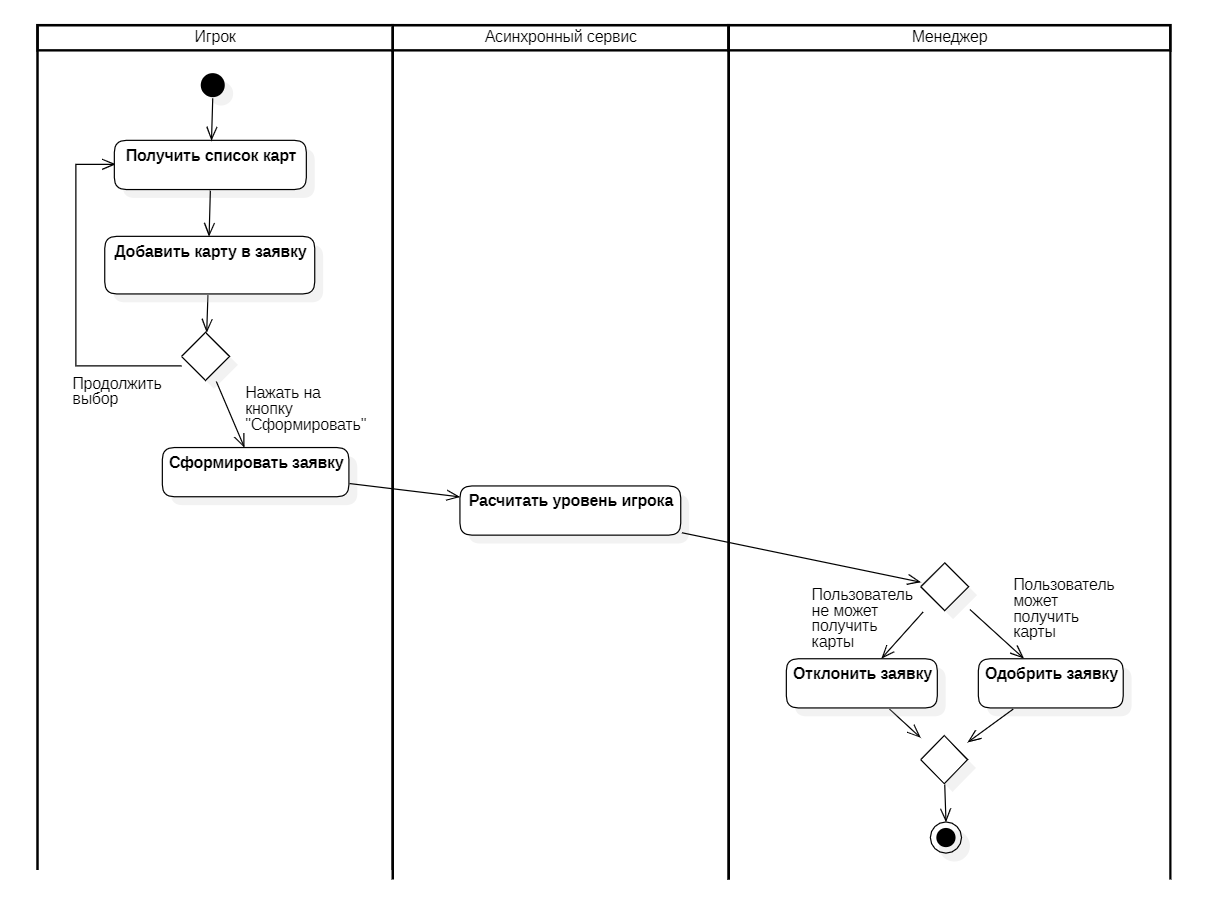


Рисунок 2 - Диаграмма деятельности

Пользователь выбирает спектры, затем формирует на основе выбранных спектров заявку. Эту заявку затем обрабатывает асинхронный сервис, а затем и менеджер. Менеджером может выступать как модератор, так и администратор В случае одобрения для заявки запускается расчёт уровня игрока в асинхронном сервисе. Возможные состояния заявки отражены на диаграмме состояний (рис. 3).

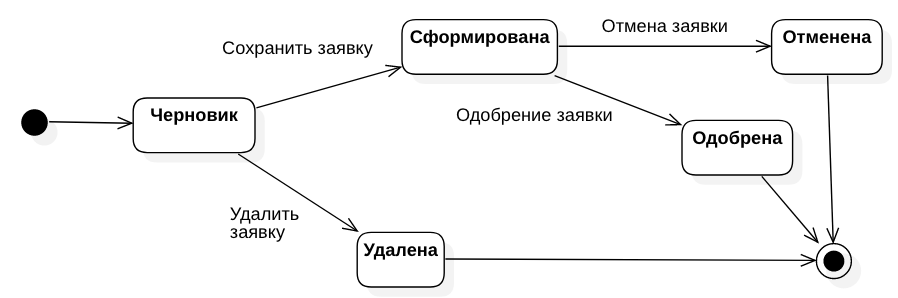


Рисунок 3 - Диаграмма состояний заявки

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

# **АРХИТЕКТУРА**

Архитектура системы отображена на диаграмме развертывания (рис. 4). Бекенд разворачивается в кластере докер 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 [4]. Сервис level-calcualtor [3] разворачивается отдельно и необязательно на том же компьютере, что и кластер основных контейнеров.

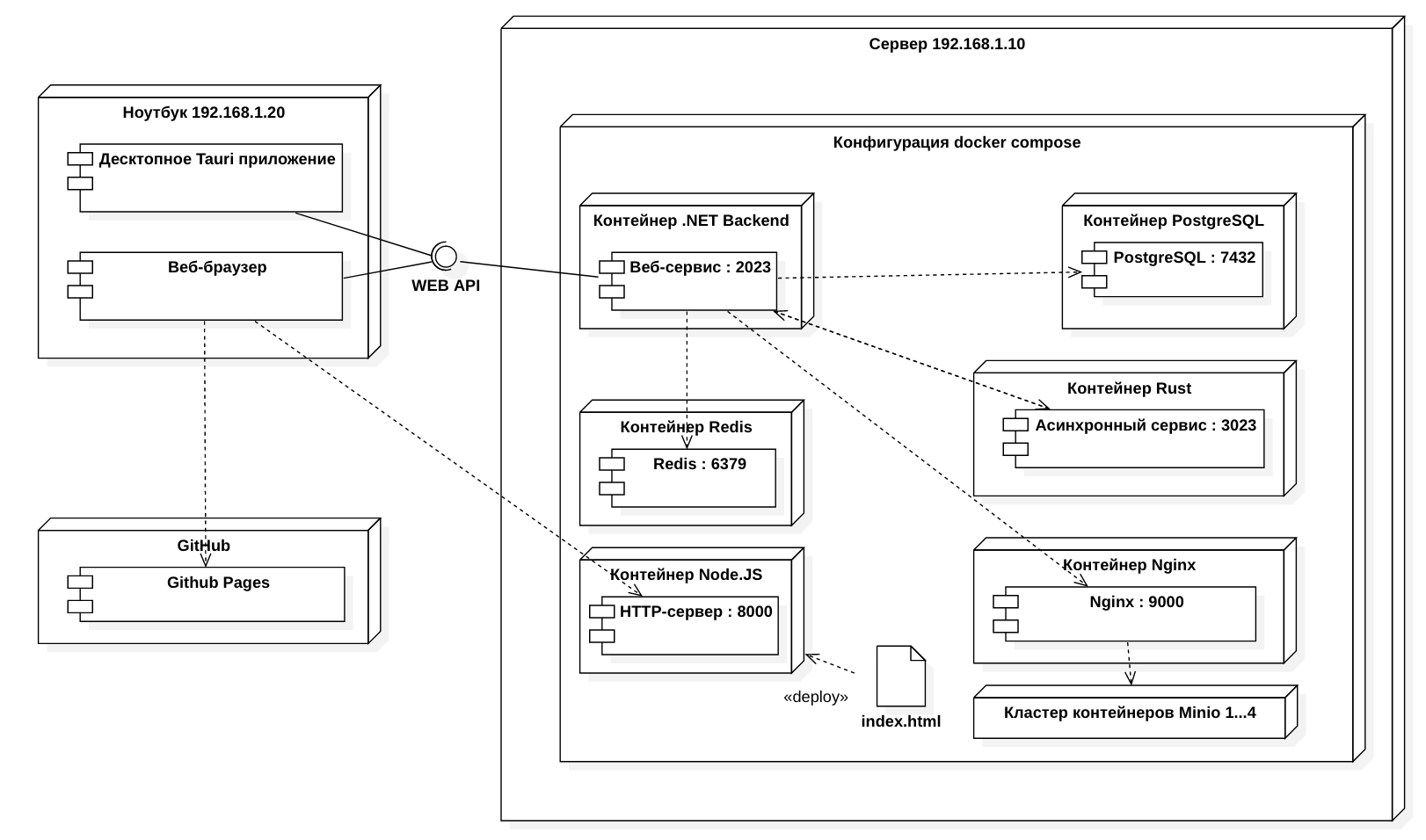


Рисунок 4 - Диаграмма развертывания

Нативное [5] и браузерное приложения [6, 7] обращаются к веб-сервису на базе технологии ASP.NET Core 7 [8, 9] через REST Web-API. Использование ASP.NET 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. Помимо базовых полей, пользователь также имеет поле level, отражающее его уровень, получаемый за игру на картах. Модель спектров представляет собой набор полей, необходимых исключительно для бизнес-логики. Для хранения в одной заявке нескольких спектров используется промежуточная таблица RequestMap, реализующая связь М-М.

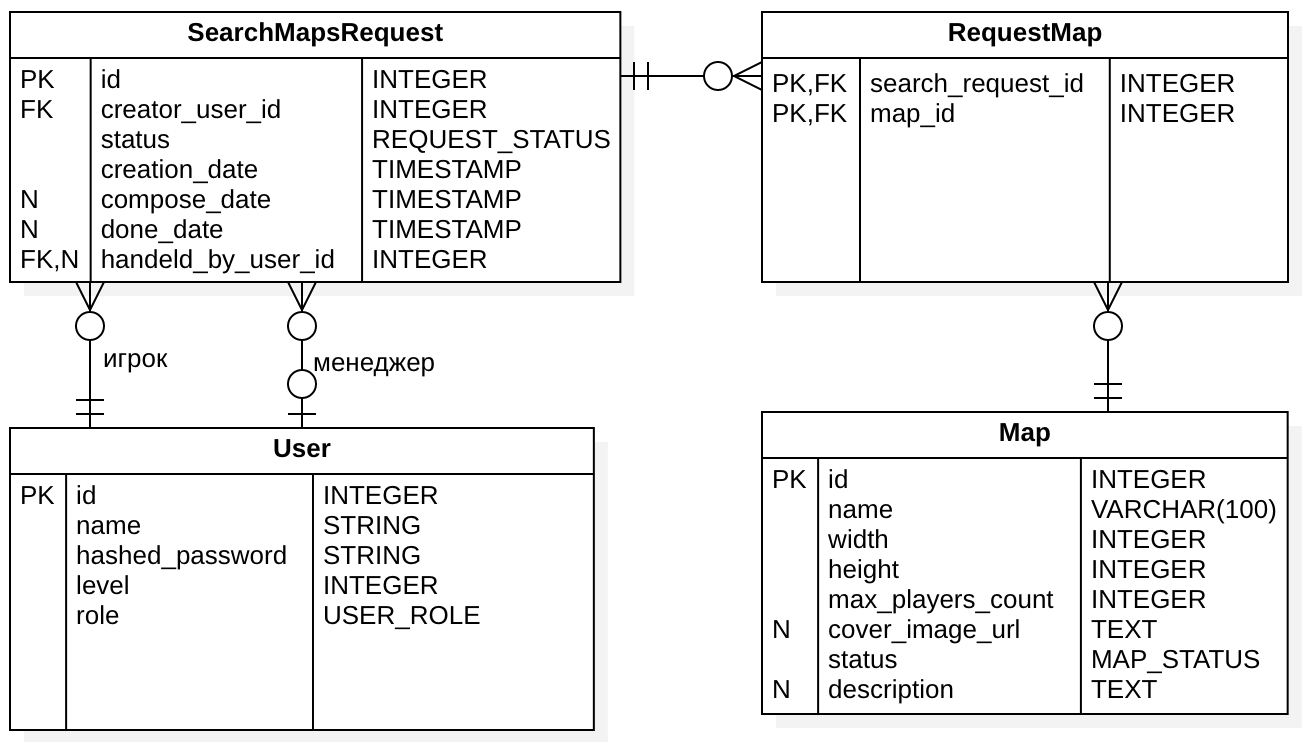


Рисунок 5 - ER диаграмма

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, спектры имеют связь с сервером статических файлов, т.к. в картах хранится ссылка на их изображение, хранимое на сервере статических файлов.

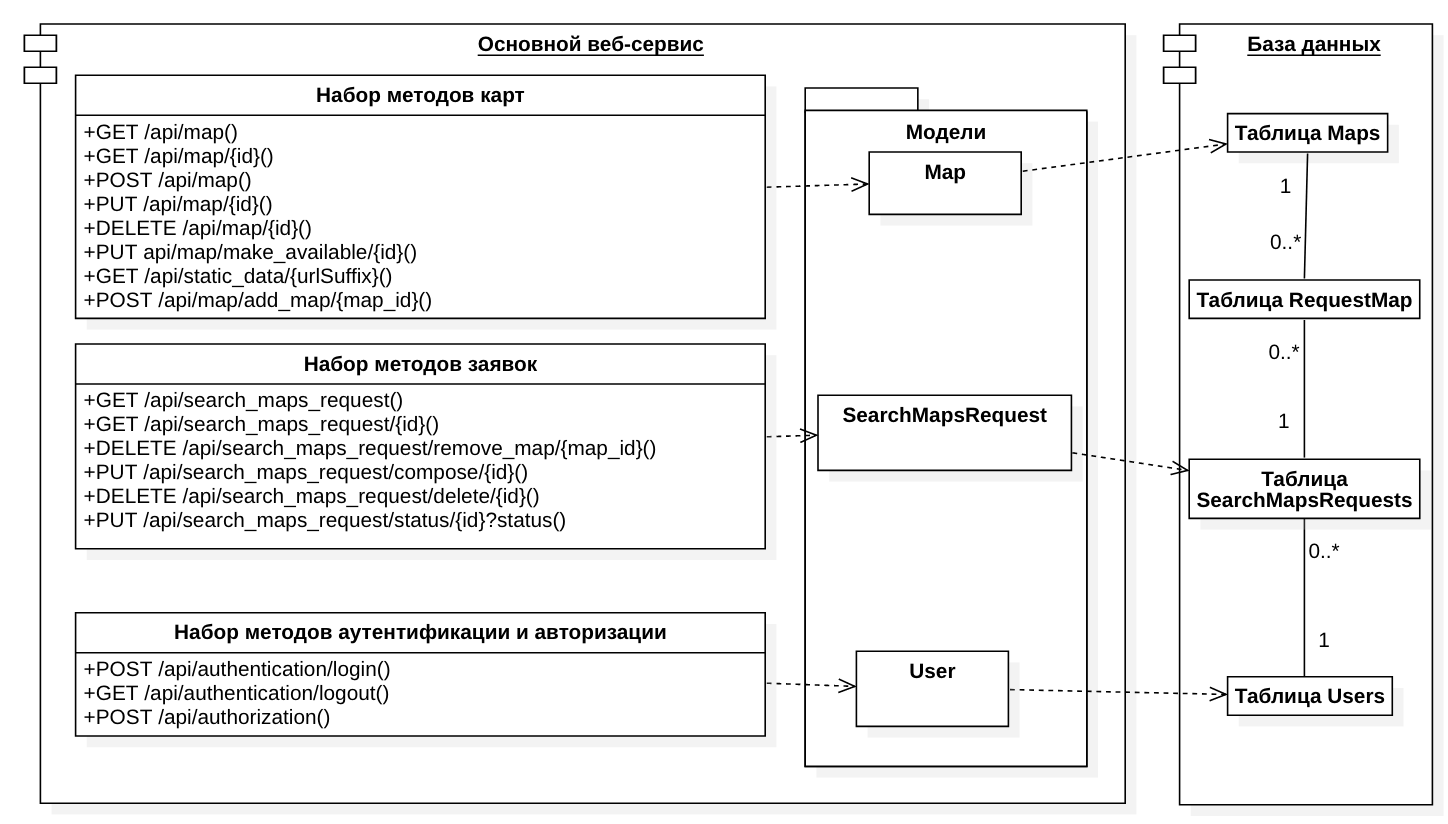


Рисунок 6 *-* Диаграмма классов бекенда

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).

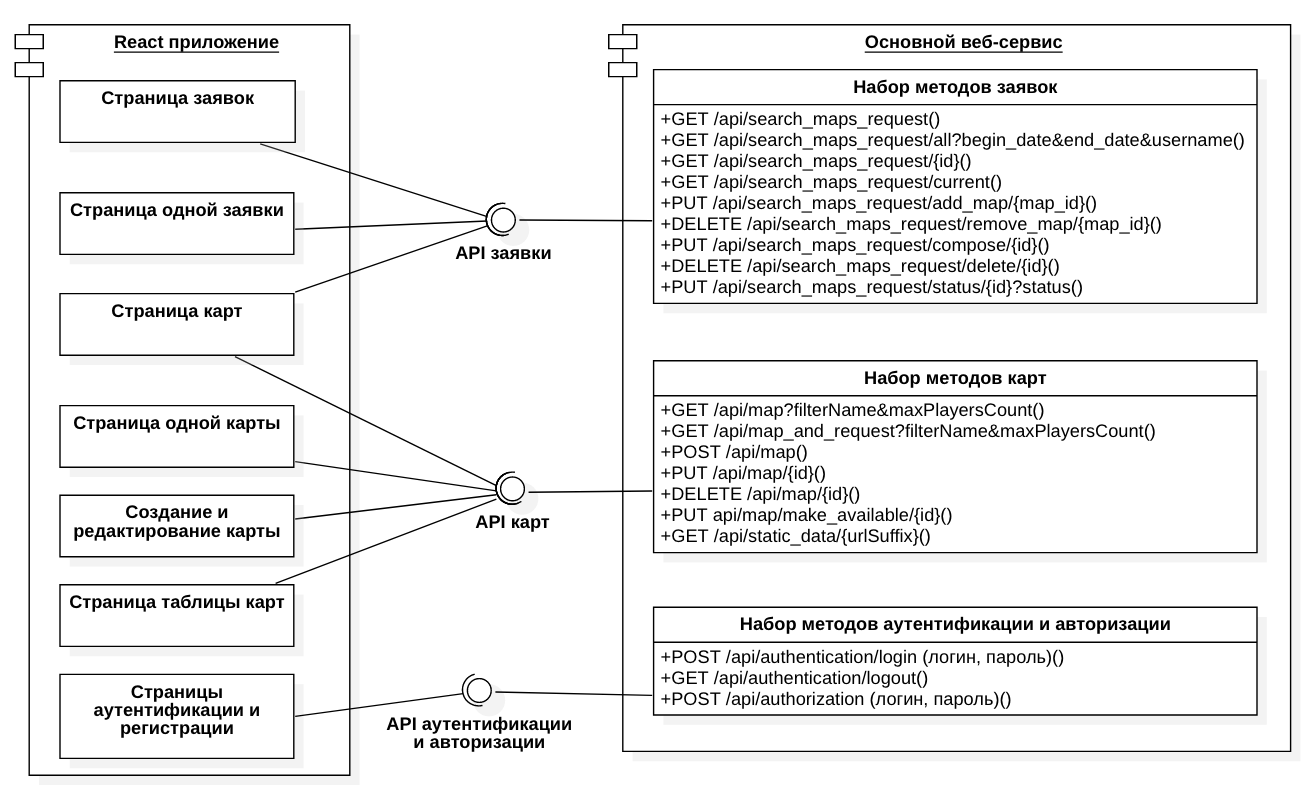


Рисунок 7 *-* Диаграмма классов фронтенда

# **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: спектры, заявки, пользователи и аутентификация. Методы следуют правилам REST API.

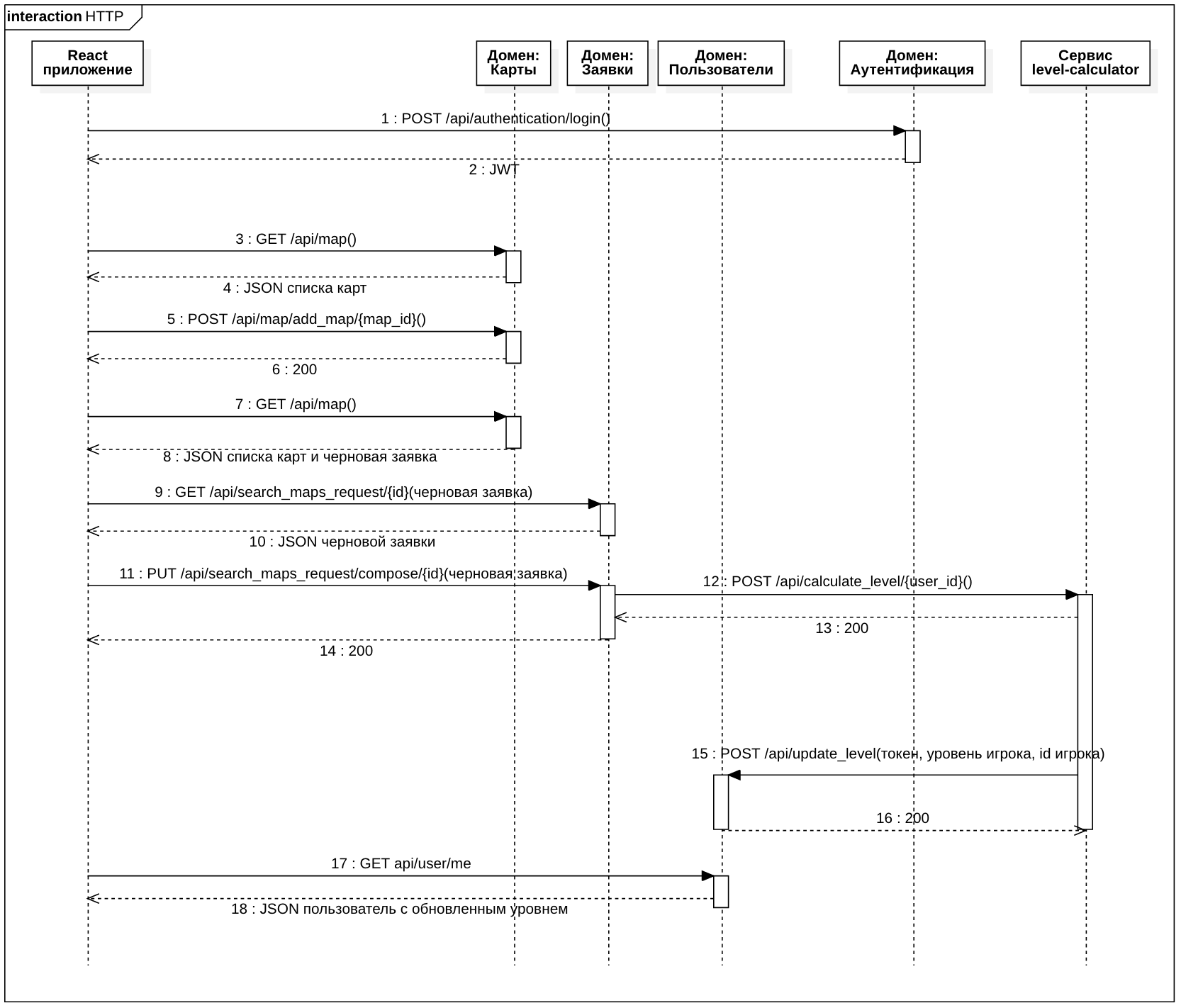


Рисунок 8 - Диаграмма последовательности

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 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 спектров, которые возвращаются в JSON формате. Пользователь выбирает карту, которую хочет получить, и, нажимая на кнопку «добавить» в графическом интерфейсе, отправляет запрос на добавление спектра в свою черновую заявку. 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 сервису level-calculator на то, чтобы он рассчитал уровень игрока, который будет у него, когда он сыграет на выбранных картах. Когда заявка будет одобрена, пользователь сможет увидеть через некоторое время, что его уровень изменился благодаря картам, которые он выбрал для игры.

Процесс рассмотрения заявок происходит также через графический интерфейс. Менеджеры могут просматривать списки всех заявок и, нажимая на соответствующие кнопки, отправлять запросы на одобрение или отклонение заявок 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. У администраторов функционал еще больше, через графический интерфейс они могут управлять непосредственно картами. Им доступны такие функции, как создание и редактирование спектров, просмотр списка спектров 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).

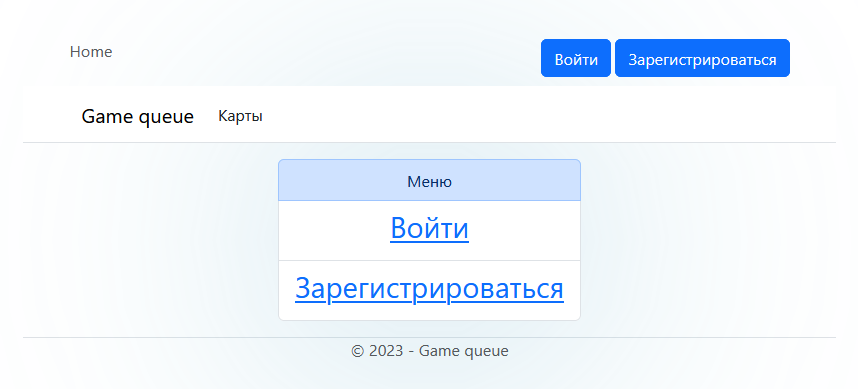
****

Рисунок 9 - Главное меню (неавторизированный пользователь)

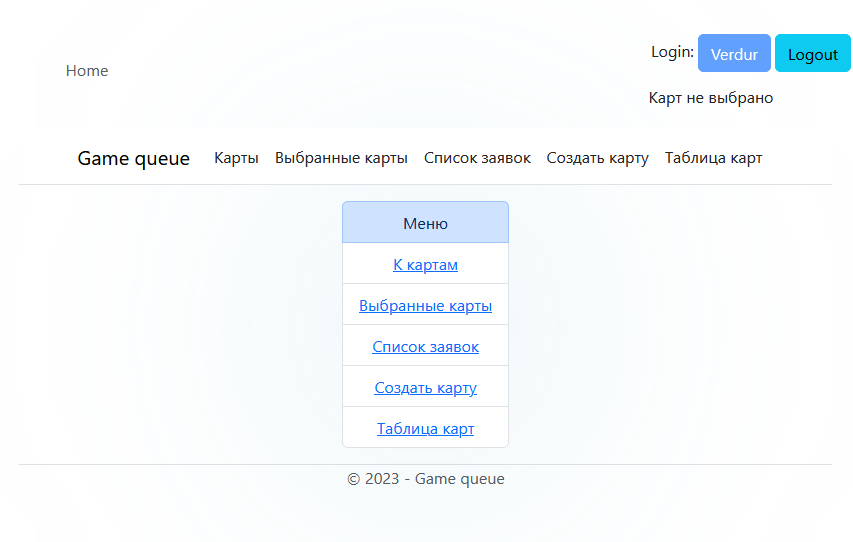
****

Рисунок 10 - Главное меню (администратор)

Первоначальная страница для всех пользователей и гостей. В зависимости от типа пользователя её содержимое меняется. Для гостей, например, там отображаются только кнопки «войти» и «зарегистрироваться», а для пользователя – «список спектров», «выбранные спектра» и «список заявок».

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.

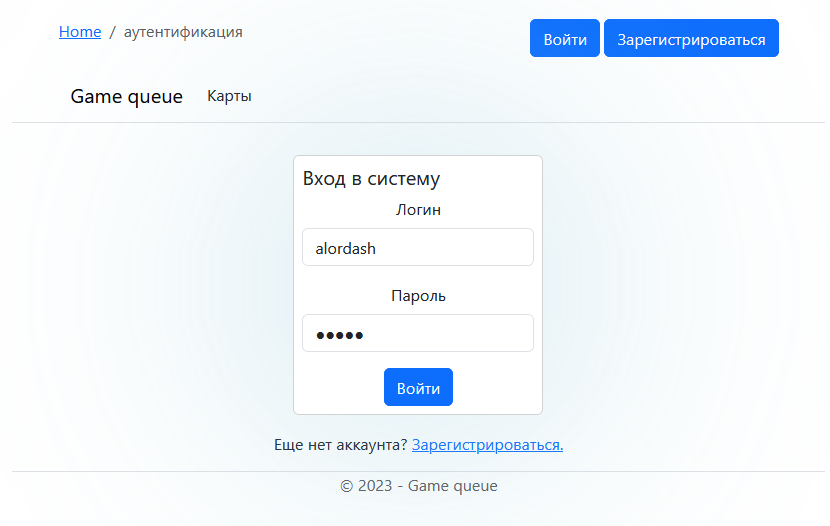
****

Рисунок 11 - Страница авторизации

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

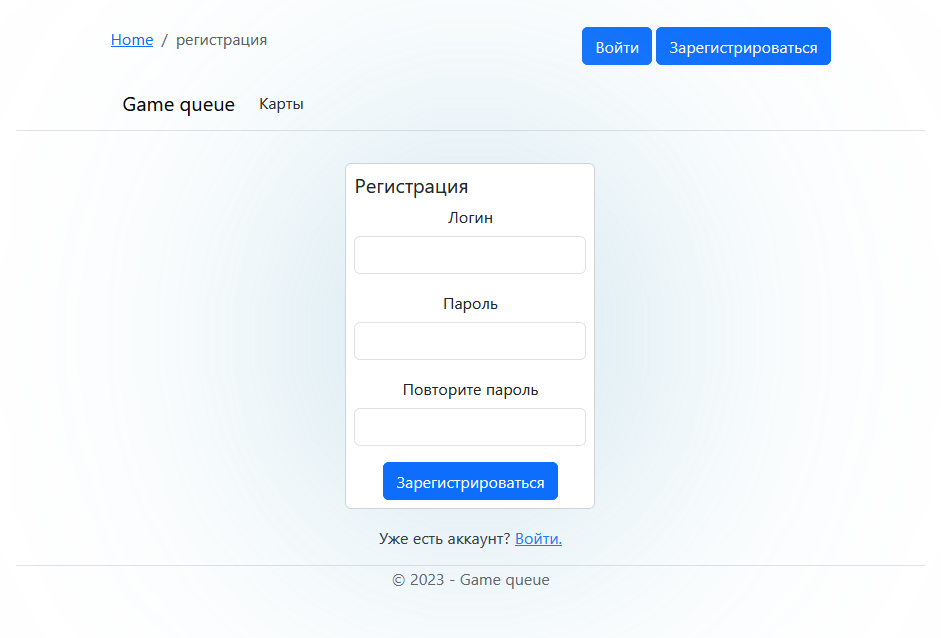
****

Рисунок 12 - Страница регистрации

На странице со списком спектров (рис. 13) отображается список спектров в виде карточек. У каждой карточки есть кнопка «Подробнее», переносящая пользователя на страницу с подробной информацией о карте, и кнопки «выбрать карту»/«снять выбор» в зависимости от того выбрал ли пользователь уже эту карту или нет. Слева находится фильтр спектров, там можно ввести фрагмент названия спектра, который должен присутствовать в названии спектров, а также максимальное количество игроков.

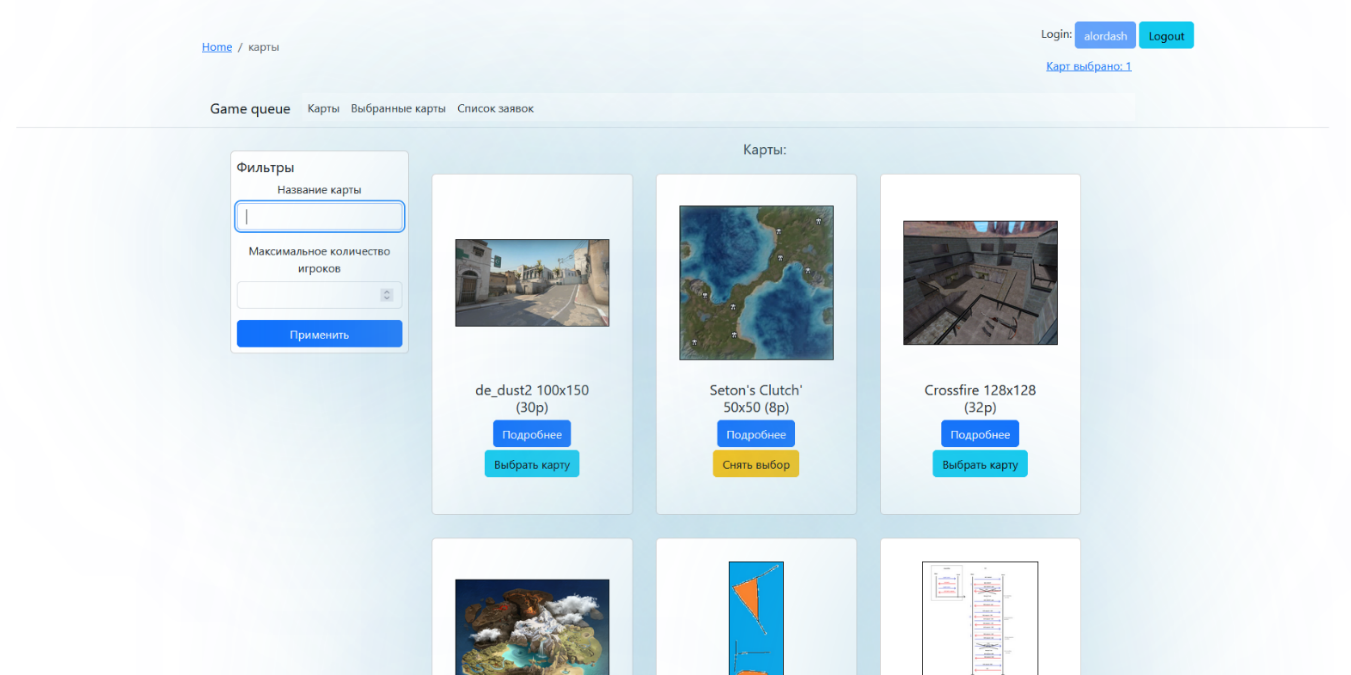
****

Рисунок 13 - Страница со списком спектров

На странице с подробным описанием спектра (рис. 14) отображается подробная информация о карте: название, размер, максимальное количество игроков и текстовое описание. Можно добавить карту в заявку или убрать из неё, также можно вернуться назад.

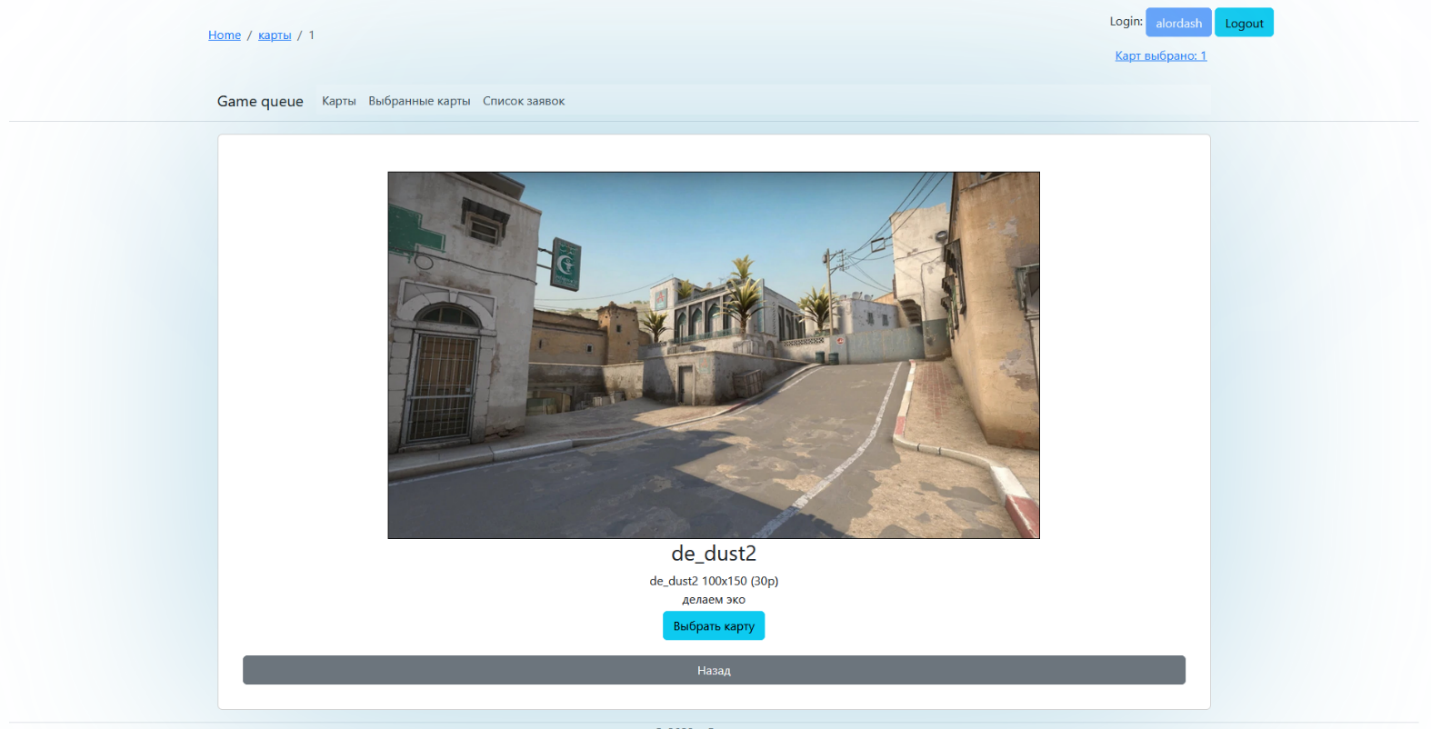
****

Рисунок 14 - Страница с подробным описанием спектра

На странице со списком заявок (рис. 15, 16) отображается список заявок. В зависимости от типа пользователя этот список будет функционально отличаться. Так, для пользователей отображается список созданных ими заявок: номер, статус, дата создания, дата формирования, дата завершения, список выбранных спектров и кнопка для открытия заявки.

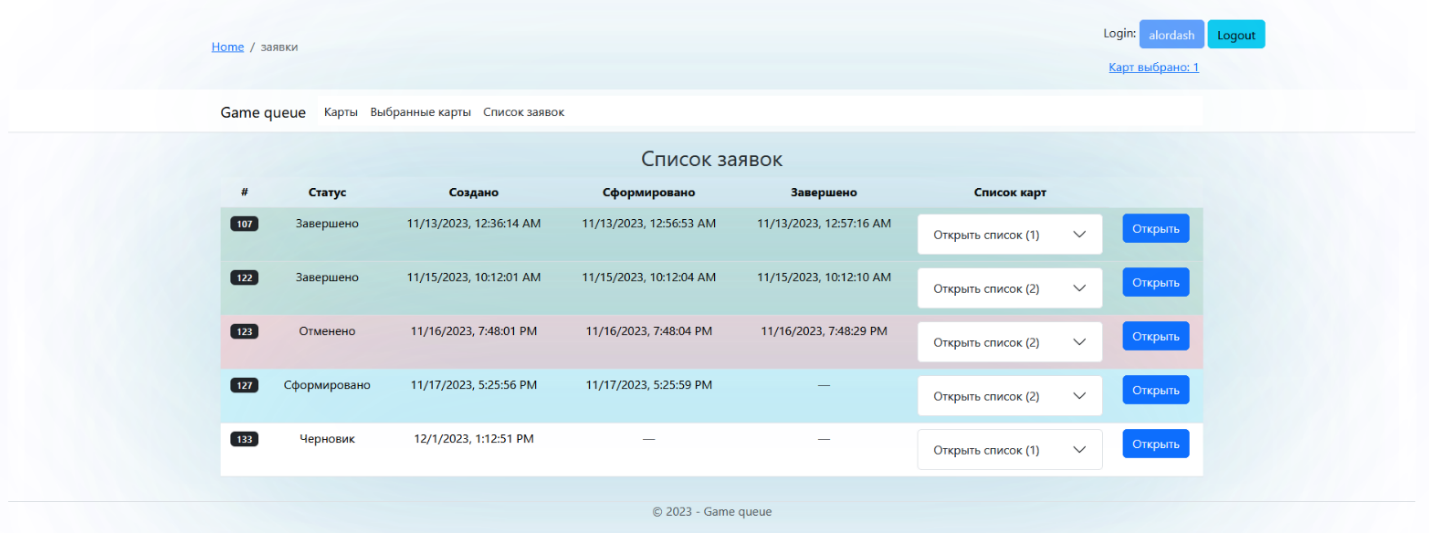
****

Рисунок 15 - Страница со списком заявок (пользователь)

Для администратора или модератора функция этой страницы шире. Для них отображается список всех заявок всех пользователей с более подробной информацией: номер заявки, автор заявки, дата создания, дата формирования, дата завершения, кто её обработал, список выбранных спектров и кнопка для открытия заявки. У заявок со статусом «черновик» также присутствует кнопка для установления статуса заявки.

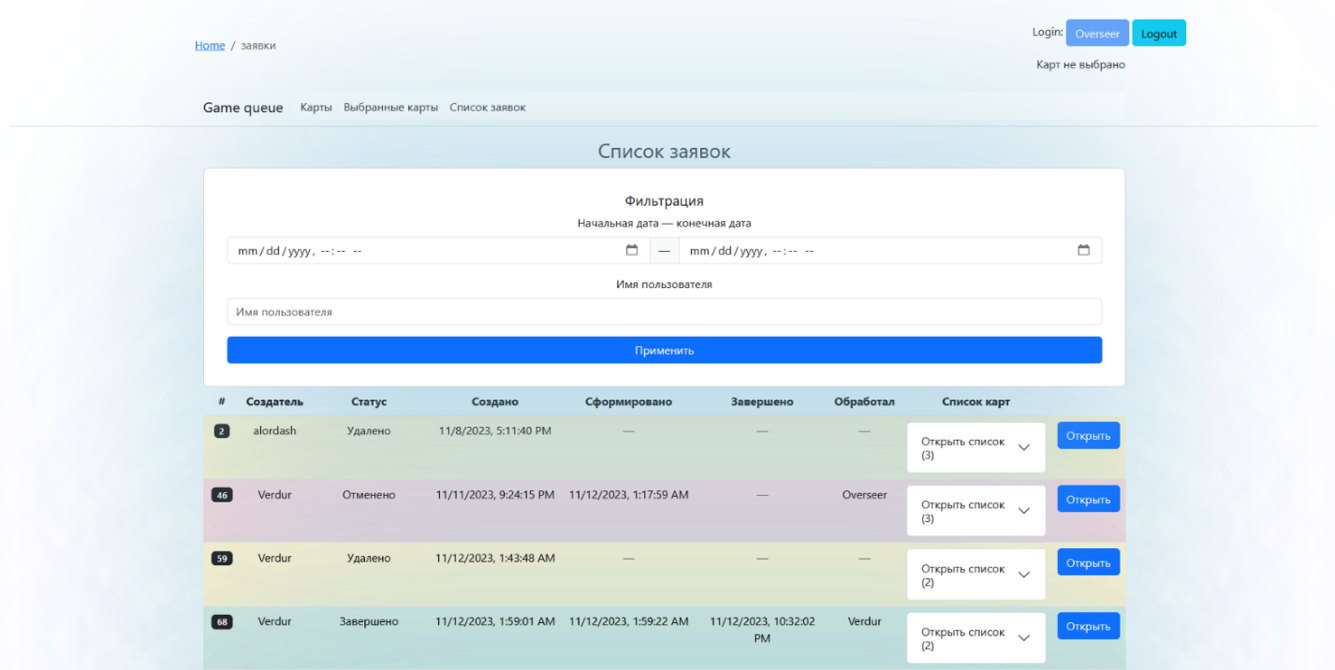
****

Рисунок 16 - Страница со списком заявок (модератор/администратор)

На странице с подробным описанием заявки (рис. 17) отображается подробная информация о заявке. Список выбранных спектров в виде карточек, а также статус заявки.

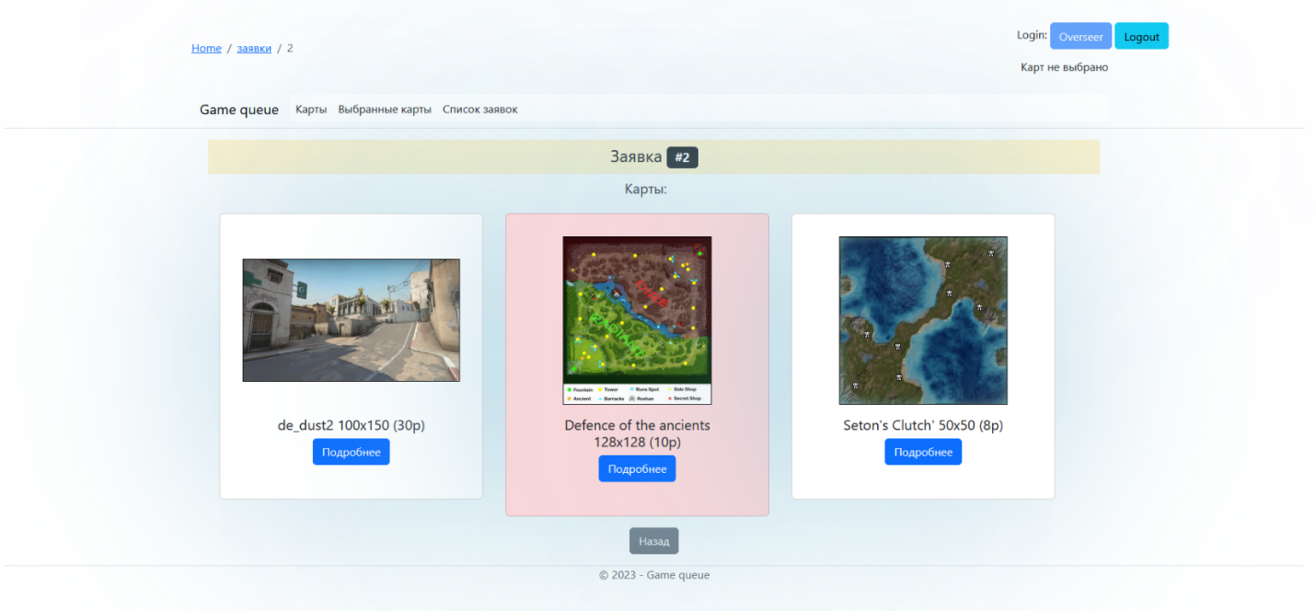
****

Рисунок 17 - Страница с подробным описанием заявки

На странице с таблицей спектров (рис. 18) администратор может в компактном и удобном формате просмотреть список всех спектров, существующих в системе. Отображаются следующие поля: номер, изображение-обложка, название, ширина, высота, максимальное количество игроков, описание и статус спектра. Также можно открыть существующую карту или перейти на страницу создания новой (рис. 19).

****

Рисунок 18 - Страница с таблицей спектров

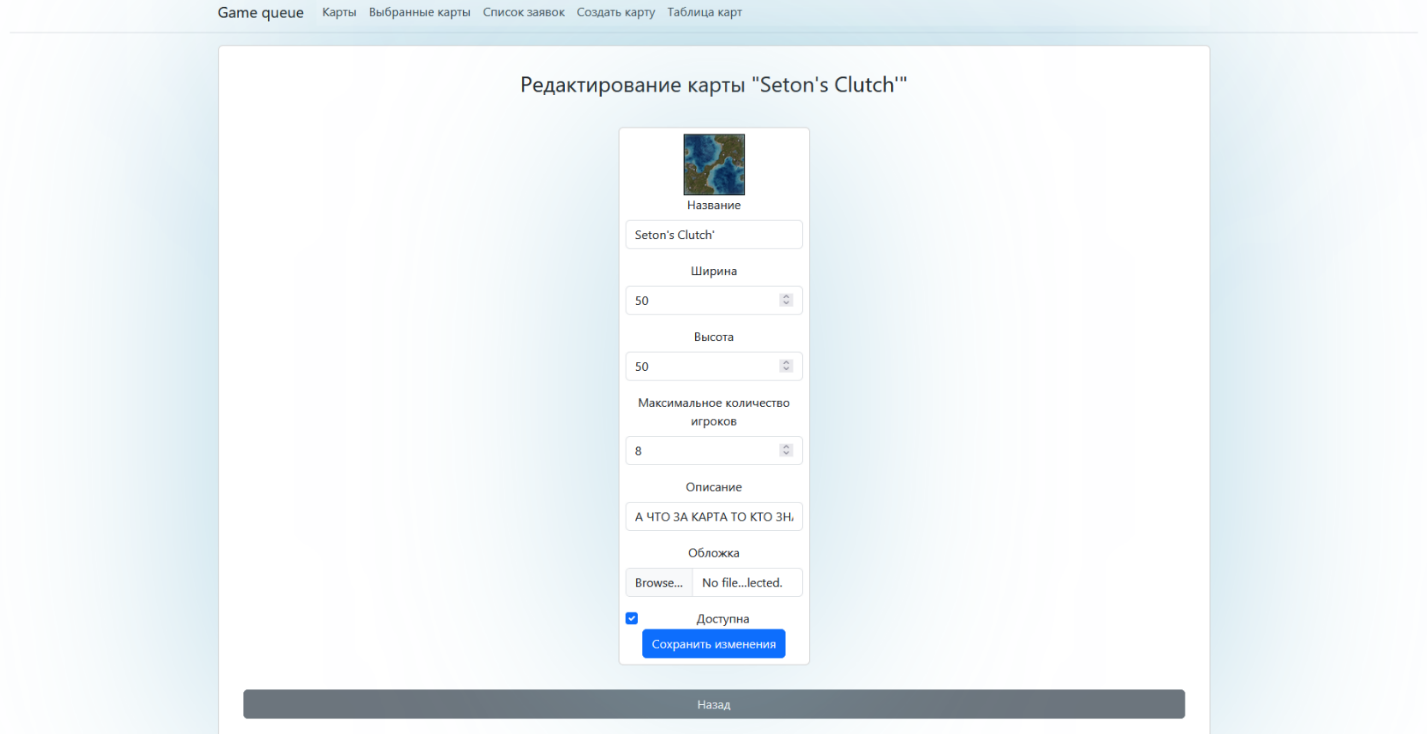
****

Рисунок 19 - Страница редактирования/создания спектра

На странице редактирования/создания спектра (рис. 19) администратор может отредактировать существующую, или создать новую карту. Для редактирования доступны все поля: название, ширина, высота спектра, максимальное количество игроков, описание и изображение-обложка. Также можно скрыть или отобразить карту в списке спектров, отображаемом для пользователей.

# **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 Golang.
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://web-3-game-queue.github.io/front/index.
6. В веб-сервис добавлена авторизация через JWT, а методы задокументированы через Swagger.
7. Реализован интерфейс игрока был реализован. Доступ к нему имеют только авторизированные пользователи.
8. Выделенный сервис был разработан и развернут в отдельном docker кластере.
9. Реализован интерфейс модератора игровых спектров для подтверждения новых игр и редактирования игровых локаций-спектров.
10. Было реализовано десктопное приложение на Tauri, 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/orgs/web-3-game-queue/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).

**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: «Анализ реликтового излучения»

Студент: Лавренов М.А.

Группа ИУ5-54Б

Преподаватель: Канев А.И.

2023г.

1. **Цель работы**

Целью является разработка системы для заказов на космические аппараты сканирования реликтового излучения с выбранным спектром, включающую в себя веб-сервис, веб-приложение, десктопное приложение и выделенный сервис для сканирования.

1. **Назначение**

Система предназначена для пользователей и модераторов сервиса сканирования. В системе предусмотрен доступ к выбору спектров для сканирования. Для получения доступа пользователю необходимо создать заявку на сканирование. Система предоставляет автоматизированный способ создания, учета и ведения заявок. Также она позволяет модераторам принимать или отклонять заявки.

1. **Задачи**:
   1. Разработать дизайн приложения.
   2. Создать базу данных в PostgreSQL.
   3. Создать веб-сервис на технологии Golang.
   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 игрока в React.
   8. Реализовать интерфейс модератора в React.
   9. Создать нативное IOS приложение на Swift.
   10. Создать выделенный сервис для сканирования на Python Django.
   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 | Регистрация | /signin | login: string  password: string  username: string |  |
| 4.1.2. | POST | Вход в аккаунт | /login | login: string  password: string | auth jwt |
| 4.1.3. | POST | Выход из аккаунта Доступно только авторизированным пользователям | /logout | auth jwt |  |
| * 1. Методы Спектров | | | | | |
| 4.2.1. | GET | Возвращает список Спектров, удовлетворяющих переданным критериям и черновую заявку пользователя | /Spectrums/ | auth jwt  {  search: string  } | {  id: int,  name: string  len: float,  freq: float,  desc: string,  ImageUrl: string?,  } |
| 4.2.2. | GET | Возвращает карту | /Spectrums/{id}/ | id: int | {  id: int,  name: string  len: float,  freq: float,  desc: string,  ImageUrl: string?  } |
| 4.2.3. | POST | Создает карту  Доступно только модераторам | /Spectrums | auth jwt  {  name: string  len: float,  freq: float,  desc: string,  ImageUrl: string?,  } |  |
| 4.2.4. | PUT | Обновляет информацию о карте  Доступно только модераторам | /Spectrums/{id} | auth jwt  {  id: int,  name: string  len: float,  freq: float,  desc: string,  ImageUrl: string?,  } |  |
| 4.2.5. | DELETE | Помечает карту как удаленную  Доступно только модераторам | /Spectrums/{id} | id: int |  |
| * 1. Методы заявок | | | | | |
| 4.3.1. | GET | Возвращает список всех заявок  Доступно только модераторам | /Satellites | auth jwt  userlogin:string?  datestart: datetime?  dateend: string?  Status:string? | {  Satellites[]{  ID: int,  DateCreate: date,  DateFormation: date?,  DateCompletion: date?,  Status: string,  Satellite: satellite,  ModerLogin: string,  UserLogin: string,  Percentage: string,  }  } |
| 4.3.2. | GET | Возвращает подробную информацию о заявке  Доступно только модераторам | /Satellites/{id} | auth jwt  id: int | {  ID: int,  DateCreate: date,  DateFormation: date?,  DateCompletion: date?,  Status: string,  Satellite: satellite,  ModerLogin: string,  UserLogin: string,  Percentage: string,  SpectrumRequests[]:{  id: int,  name: string  len: float,  freq: float,  desc: string,  ImageUrl: string?,  }  } |
| 4.3.3. | DELETE | Удаляет заявку пользователя{map\_id}  Доступно только модераторам | /Satellites | auth jwt  id: int |  |
| 4.3.4. | PUT | Изменяет переданные поля заявки «черновик»  Доступно только модераторам | /Satellites | auth jwt  {  "id": int,  "date\_created": date,  "date\_formed": date,  "date\_accepted": date,  "status": string,  "satellite": string,  "user\_id": int,  "moder\_id": int  } | {  "id": int,  "date\_created": date,  "date\_formed": date,  "date\_accepted": date,  "status": string,  "satellite": string,  "user\_id": int,  "moder\_id": int  } |
| 4.3.5. | PUT | Устанавливает заявке процент сканирования Доступен только выделенному сервису | /SatelliteAsyncStatus/{id} | auth passkey  id: int  percentage: string |  |
| 4.3.6. | PUT | Устанавливает заявке статус «в работе» или «завершена»  Только если её текущий статус «создан».  Также вызывает выделенный сервис.  Доступно только авторизованным пользователям | /SatellitesUser/{id} | Auth jwt,  id: int |  |
| 4.3.7. | PUT | Устанавливает заявке статус «завершена» или «отклонена» «удалена»  Только если её текущий статус «В работе»  Доступно только модераторам | /SatellitesModer/{id} | Auth jwt,  id: int |  |
| 4.3.8. | GET | Возвращает все заявки пользователя  Доступно только авторизованным пользователям | /UsersSatellite | User\_id: int |  |
| 4.3.9. | PUT | Изменяет значения полей в заявке на значения, заданные пользователем  Доступно авторизованным пользователям | /UsersSatelliteUpdate | Auth jwt,  {  "id": int,  "date\_created": date,  "date\_formed": date,  "date\_accepted": date,  "status": string,  "satellite": string,  "user\_id": int,  "moder\_id": int  } | {  "id": int,  "date\_created": date,  "date\_formed": date,  "date\_accepted": date,  "status": string,  "satellite": string,  "user\_id": int,  "moder\_id": int  } |
| * 1. Методы услуг в заявках | | | | | |
| 4.4.1. | POST | Добавляет услугу в заявку Доступно только авторизованным пользователям | /SpectrumsRequests | User\_id: int,  spectrum\_id: int | Updated\_spectrum\_request: string |
| 4.4.2. | DELETE | Удаляет спектр из заявки  Доступно только авторизованным пользователям | /SpectrumsRequests | auth jwt  satellite\_id: int,  spectrum\_id: int |  |
| 4.4.3. | PUT | Изменяет спектр в заявке  Доступно только авторизованным пользователям | /SpectrumsRequests | auth jwt  satellite\_id: int,  spectrum\_id: int |  |

1. **Функциональные требования:**
   1. Главное меню. На этой странице находится текст с приветствием, если пользователь не авторизирован, и меню со списком доступных пунктов.
      1. Доступна всем пользователям.
      2. Действия
         1. Войти – переход на страницу 5.2. Только для гостей.
         2. Зарегистрироваться – переход на страницу 5.3. Только для гостей.
         3. Открыть список Спектров – переход на страницу 5.4. Только для авторизированных пользователей.
         4. Открыть список выбранных Спектров – переход на страницу 5.7. Только для авторизированных пользователей.
         5. Список заявок – переход на страницу 5.6. Только для авторизированных пользователей.
         6. Создать карту – переход на страницу 5.8. Только для администраторов.
         7. Таблица Спектров – переход на страницу 5.9. Только для администраторов.
   2. Страница с формой авторизации. На этой странице находится форма авторизации.
      1. Доступна гостям
      2. Действия
         1. Войти – производит запрос (метод 4.1.2).
         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 4.1.1), в котором передаются введенные данные формы.
         2. Войти - перенаправляет пользователя на страницу 5.2.
   4. Страница со списком Спектров. На этой странице располагается список всех Спектров и панель фильтрации Спектров.
      1. Доступна всем пользователям.
      2. Выводится информация о картах в виде карточек (метод 4.2.2).
         1. Изображение спектра.
         2. Название спектра.
         3. Ширина спектра.
         4. Высота спектра.
         5. Максимальное количество игроков.
      3. Действия
         1. Отфильтровать спектры по указанному названию и максимальному количеству игроков (метод 4.2.2).
         2. «Спектров выбрано» – перенаправляет на страницу заявки 5.7.
         3. Добавить карту в заявку (метод 4.2.2). Только для авторизированных пользователей.
         4. Подробнее – перенаправляет на страницу 5.5.
   5. Страница с подробным описанием спектра
      1. Доступна всем пользователям.
      2. Выводится информация о карте в построчном формате (метод 4.2.4).
         1. Изображение спектра.
         2. Название спектра.
         3. Ширина спектра.
         4. Высота спектра.
         5. Максимальное количество игроков.
         6. Описание спектра.
   6. Страница со списком заявок
      1. Доступна только авторизированным пользователям.
      2. Выводится информация о заявках в табличном формате (методы 4.3.1).
         1. Статус заявки.
         2. Дата создания.
         3. Дата формирования.
         4. Дата завершения.
         5. Кем обработана. Только для модераторов и администраторов.
         6. Список выбранных Спектров (в выпадающем списке).
      3. Действия
         1. Открыть заявку – перенаправляет на страницу 5.7.
         2. Установить заявке статус «отклонена» или «завершена» (метод 4.3.6). Только для модераторов и администраторов.
         3. Отфильтровать список заявок по диапазону дат и/или имени автора заявки (метод 4.3.1). Только для модераторов и администраторов.
   7. Страница с подробным описанием заявки. Пользователь может изменить спектры в заявке. Модератор может просматривать заявки всех пользователей.
      1. Доступна только авторизированным пользователям.
      2. Выводится информация о заявке в карточной форме. Список Спектров, выбранных в заявке, в виде карточек (метод 4.3.2).
         1. Изображение спектра.
         2. Название спектра.
         3. Ширина спектра.
         4. Высота спектра.
         5. Максимальное количество игроков на карте.
      3. Действия, если заявка находится в статусе черновика и только для владельца заявки:
         1. Сформировать заявку (метод 4.3.4).
         2. Удалить заявку (метод 4.3.5).
         3. Удалить карту из заявки (метод 4.3.3).
   8. Страница редактирования/создания спектра. Администратор может изменить существующую или создать новую карту.
      1. Доступна только модераторам.
      2. Выводится информация о карте в построчном формате (метод 4.2.4).
         1. Изображение спектра.
         2. Название спектра.
         3. Ширина спектра.
         4. Высота спектра.
         5. Максимальное количество игроков на карте.
         6. Описание
         7. Доступность.
      3. Изменение полей спектра. Можно изменять все вышеперечисленные поля спектра (методы 4.2.5 и 4.2.6).
   9. Страница с таблицей Спектров. Предоставляет администратору удобный способ отображения всех Спектров.
      1. Доступна только модераторам.
      2. Выводит информация о картах в табличном виде (метод 4.2.3).
         1. Изображение спектра.
         2. Название спектра.
         3. Ширина спектра.
         4. Высота спектра.
         5. Максимальное количество игроков на карте.
         6. Описание (в выпадающем списке).
         7. Статус.
      3. Действия
         1. Удалить карту (метод 4.2.7)
         2. Редактировать карту – перенаправляет на страницу 5.8.
         3. Добавить карту – перенаправляет на страницу 5.8.
2. **Требования к программному обеспечению:**
   1. Серверная часть
      1. ОС: Linux/Windows
      2. Docker
      3. dotnet/sdk 7.0
      4. dotnet/aspnet 7.0
      5. redis 7.2.3-alpine
      6. nginx 1.19.2-alpine
      7. minio RELEASE.2022-10-15T19-57-03Z
      8. postgres 14
      9. rust 1.72.0
      10. scratch
   2. Клиентская часть
      1. ОС: Windows/MacOS/Linux
      2. Веб-браузер: Safari 11.1+/Chrome 40+/Opera 27+/Firefox 44+
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 1Гб