|  |  |
| --- | --- |
|  | **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: «Автоматический подбор карт для игроков»

Студент: Оразов А.В.

Группа ИУ5-51Б

Преподаватель: Канев А.И.

2023г.

# **ВВЕДЕНИЕ**

Игровая индустрия сильно расширилась за последние несколько лет и продолжает развиваться. С её ростом также увеличивается количество игроков. Учитывая, что основную массу игроков составляют казуальные игроки, можно сделать вывод, что чем проще игра, тем охотнее в неё играют игроки. Для этого было решено разработать систему автоматического подбора карт для игроков, которая сделает поиск и выбор карт для игроков более простым и удобным.

Целью работы является реализация системы для подборка игроками карт (игровых локаций), включающую в себя веб-сервис, веб-приложение, десктопное приложение и выделенный сервис расчета уровня игрока.

Система предназначена для игроков, модераторов и администраторов игры. В игре предусмотрен ограниченный доступ к картам. Для получения доступа пользователю необходимо создать заявку на получение доступа к выбранной карте. Система предоставляет автоматизированный способ создания, учета и ведения заявок. Также она позволяет модераторам и администраторам принимать или отклонять заявки. Администраторы имеют возможность редактировать существующие и создавать новые карты.

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 dotnet 7.0.
  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 десктопное приложение на Tauri.
  10. Создать выделенный сервис для расчета уровня игроков на Rust и axum.
  11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

# **БИЗНЕС-ПРОЦЕСС**

Существует много различных карт, в которые хотят поиграть игроки [1]. Для того, чтобы вручную не искать других игроков для совместной игры, пользователи могут оставлять заявки на участие в игре на определенной карте. Каждая заявка вначале создается как черновик, в который можно добавлять и из которого можно удалять карты. В одной заявке можно указать сразу несколько карт, и тогда игра будет происходить на каждой карте по очереди.

За каждую игру на картах игроки повышают свой уровень. Когда игрок определится с картами, на которых ему хочется сыграть, он формирует заявку и она отправляется на модерирование. После этого её уже нельзя редактировать. Можно также посмотреть историю своих игр. Уровень игроков учитывается при подборе игроков для совместной игры, так как нежелательно, чтобы игроки высоких уровней играли с игроками низких уровней – это может привести к несбалансированным командам и, как следствие, к неинтересной игре. Для этого менеджеры отклоняют и принимают заявки таким образом, чтобы все игроки на одной карте были примерно одного уровня. На случай если игрок передумает играть также присутствует возможность отменить заявку.

Карта представляет собой какую-то игровую локацию определенных размеров и рассчитанную на конкретное количество людей. Карты могут обладать совершенно разными размерами и количеством игроков, будь это небольшая карта для дуэлей или масштабное поле битвы на тысячу человек [2]. Особенности карты видны по её изображению и текстовому описанию. Во время игры на картах могут обнаружиться какие-то ошибки, и на этот случай администратор может временно скрыть карту. Играть на одних и тех же картах рано или поздно может надоесть игрокам, поэтому для увеличения разнообразия администрация может привнести изменения в карты. Она может отредактировать какую-либо карту, обновить её, или исправить какие-то ошибки. Также можно создать совершенно новую карту. Функции пользователей с разными ролями описаны на диаграммах прецедентов (рис. 1).

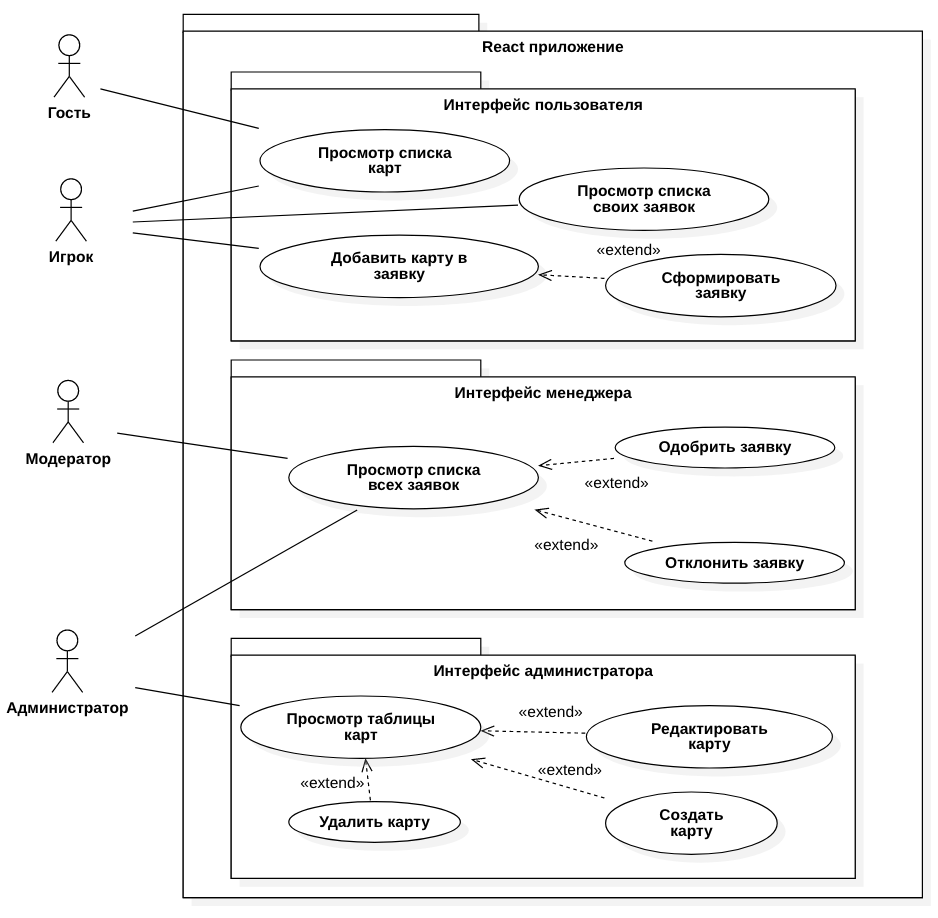


Рисунок 1 - Диаграмма прецедентов

Гостям доступен просмотр карт. Зарегистрированные гости – игроки. Они могут добавлять карты 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 карт, редактирование, создание и удаление карт, а также просмотр списка всех карт в табличном виде. Процесс оформления заявки отражен на диаграмме деятельности (рис. 2).

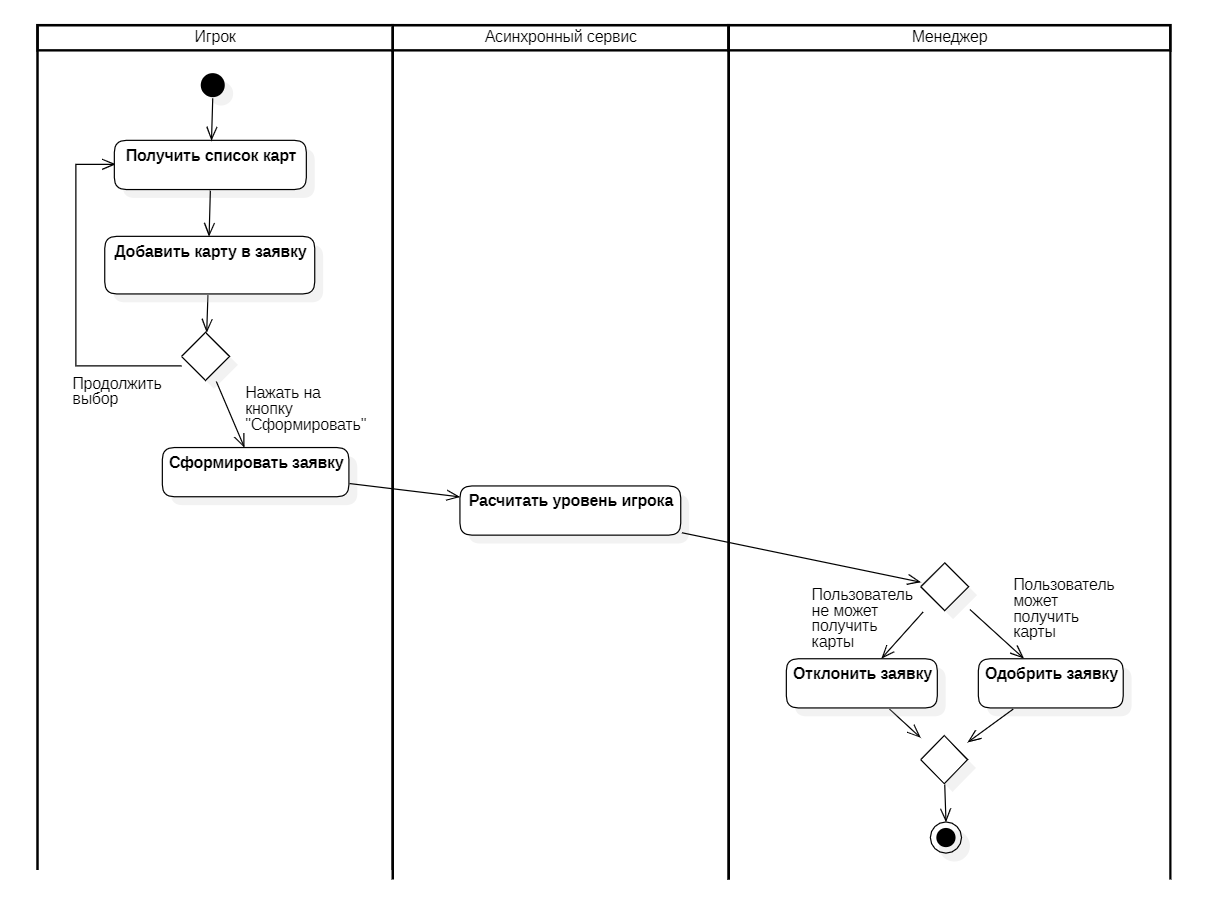


Рисунок 2 - Диаграмма деятельности

Пользователь выбирает карты, затем формирует на основе выбранных карт 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).

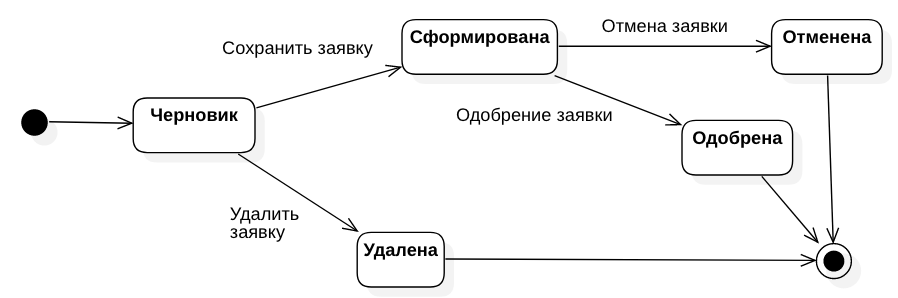


Рисунок 3 - Диаграмма состояний заявки

При выборе первой карты формируется черновик. Последующие выбранные карты 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.

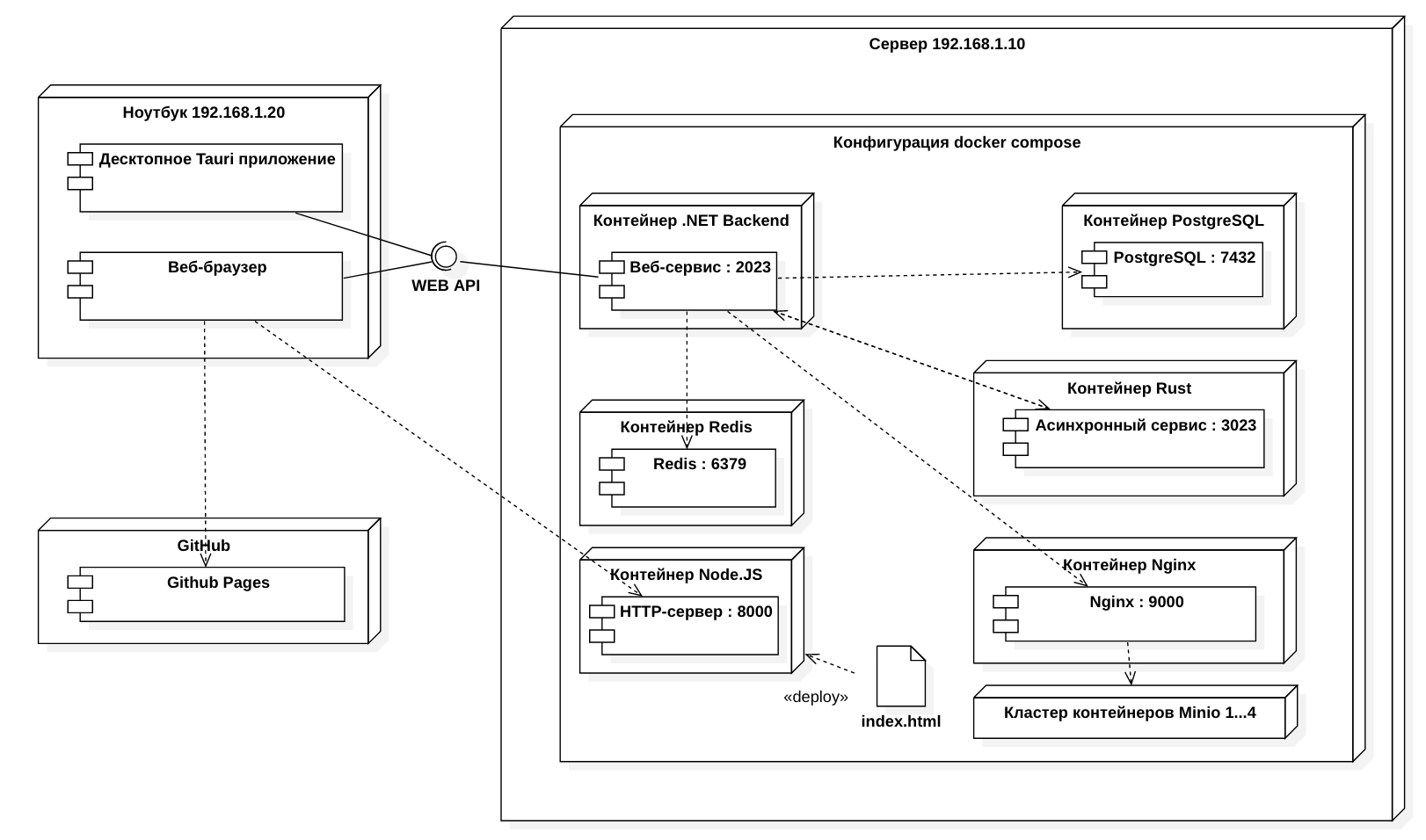


Рисунок 4 - Диаграмма развертывания

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 карт представляет собой набор полей, необходимых исключительно для бизнес-логики. Для хранения в одной заявке нескольких карт используется промежуточная таблица RequestMap, реализующая связь М-М.

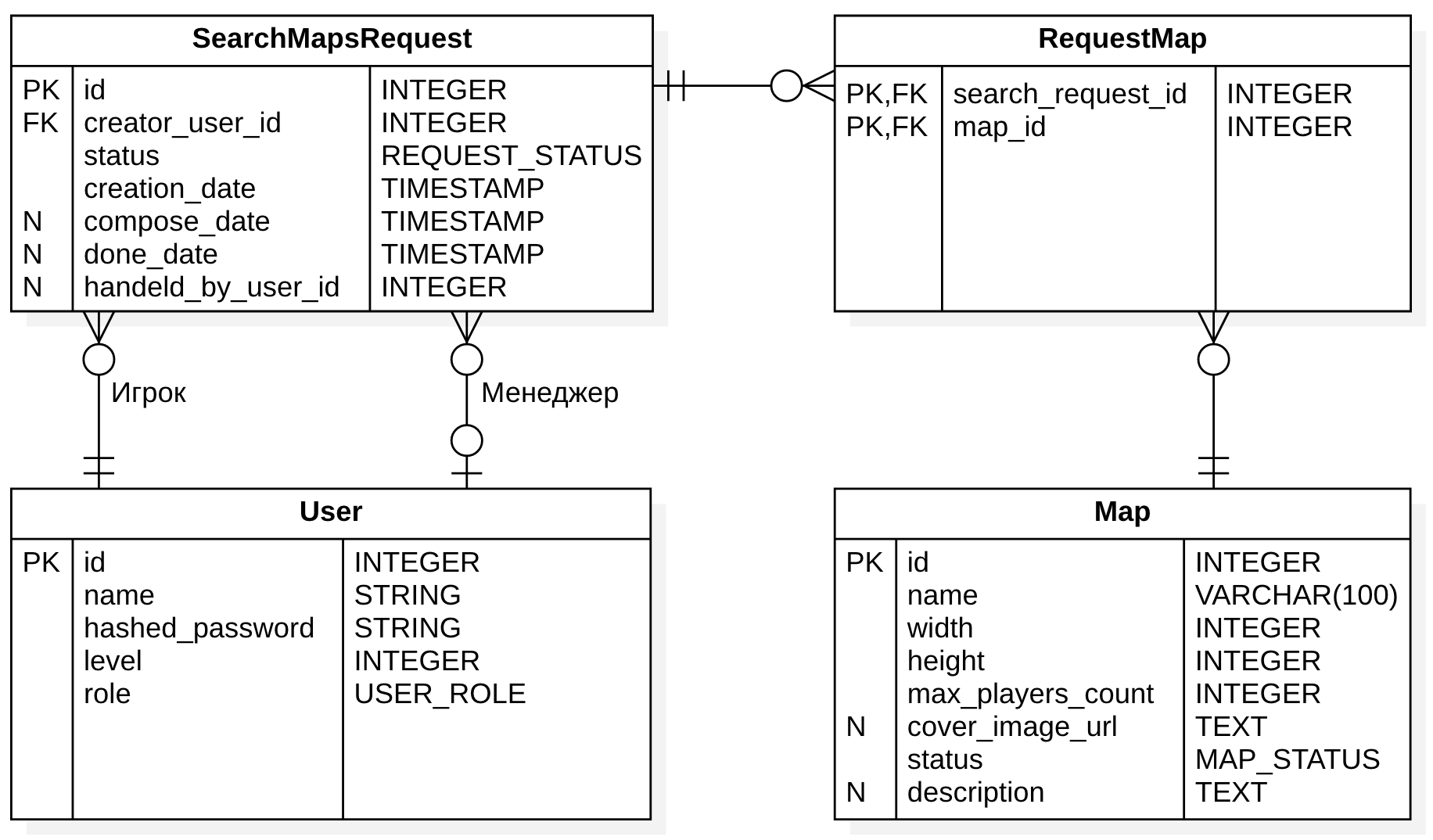


Рисунок 5 - ER диаграмма

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

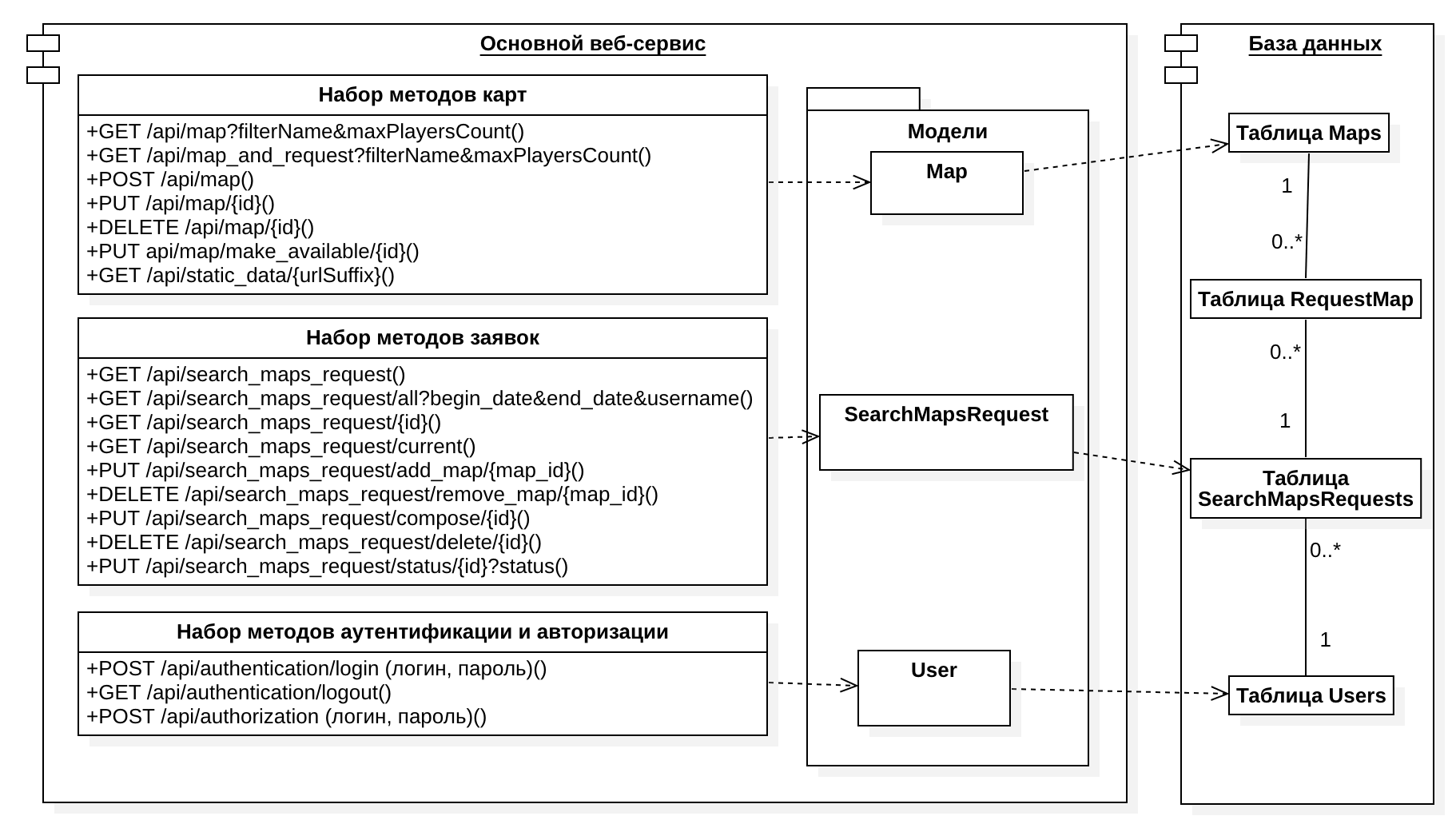


Рисунок 6 *-* Диаграмма классов бекенда

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).

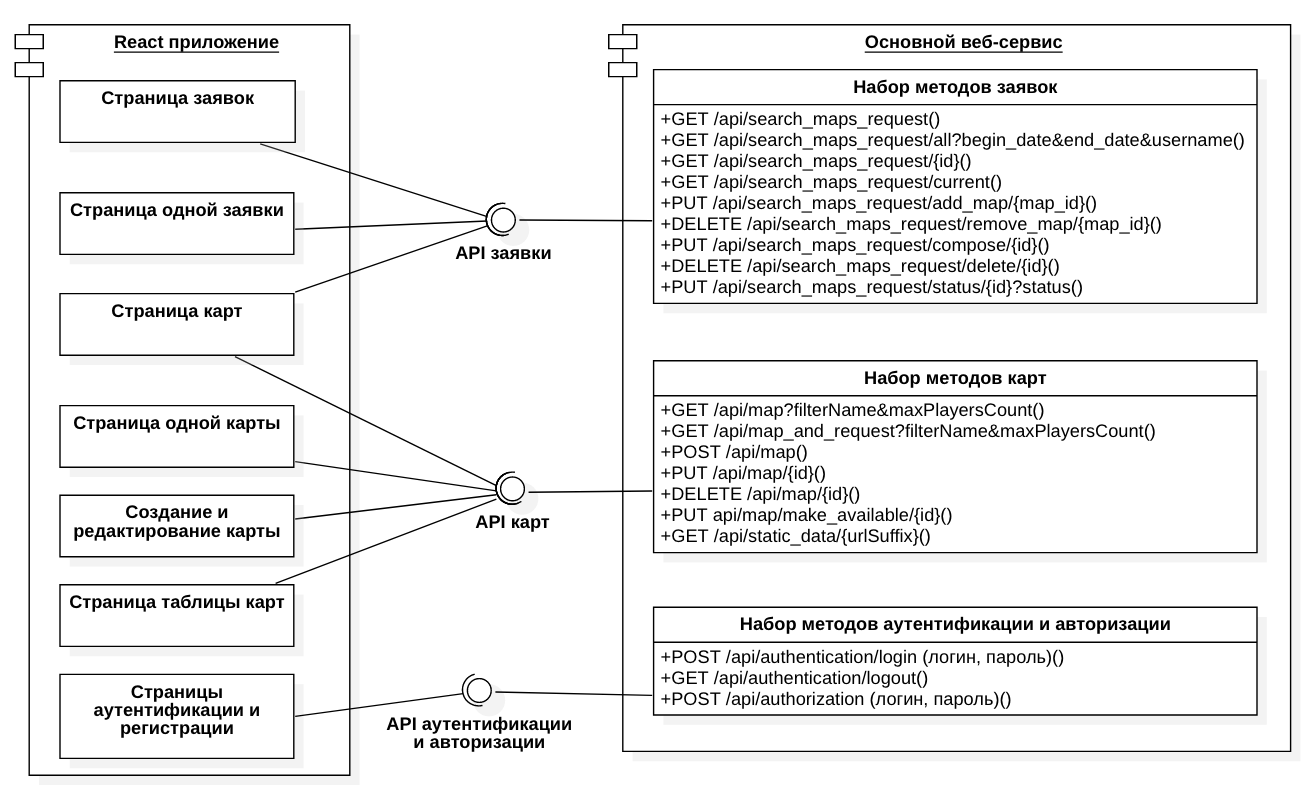


Рисунок 7 *-* Диаграмма классов фронтенда

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

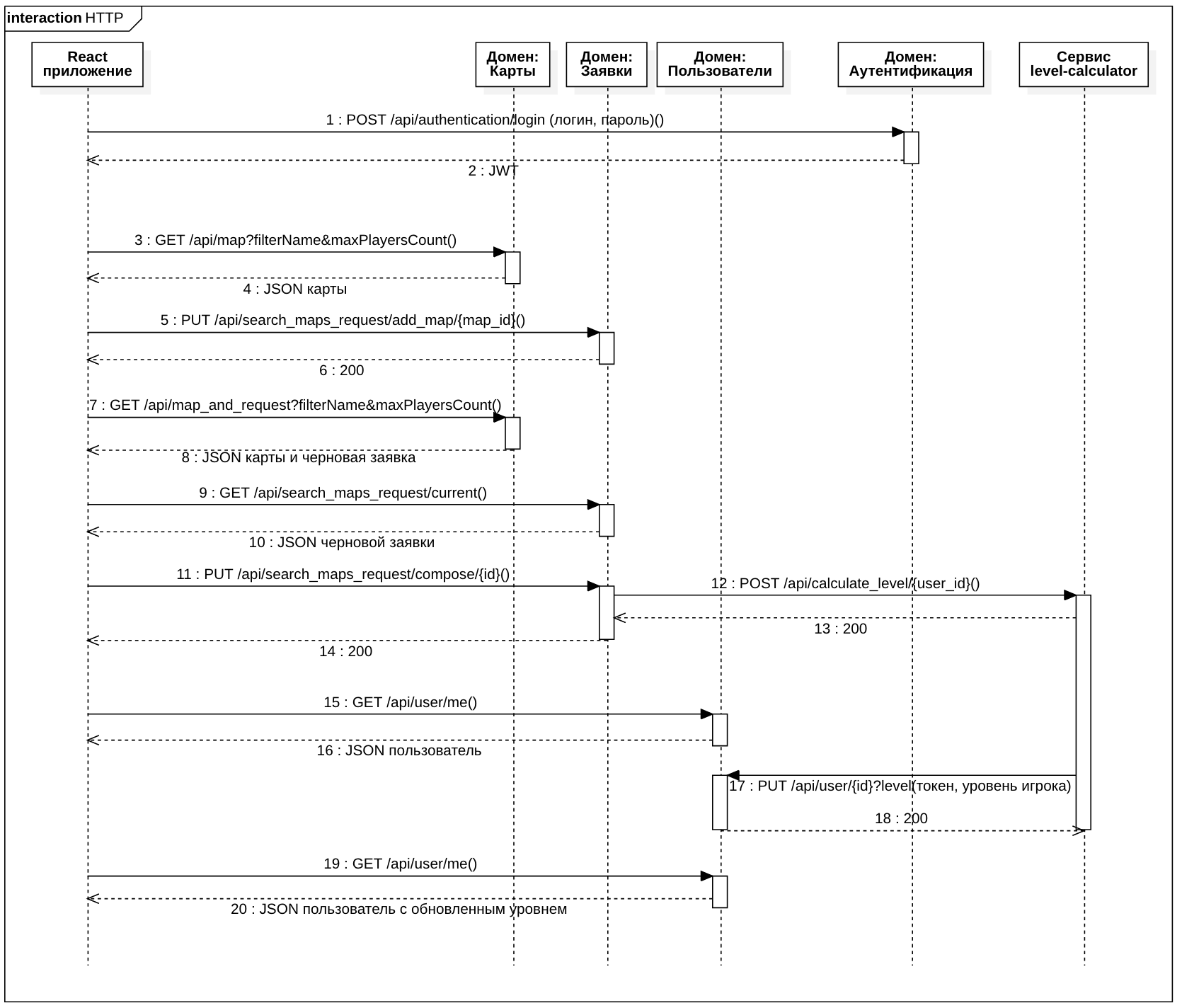


Рисунок 8 - Диаграмма последовательности

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 карт, которые возвращаются в JSON формате. Пользователь выбирает карту, которую хочет получить, и, нажимая на кнопку «добавить» в графическом интерфейсе, отправляет запрос на добавление карты 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 карт, просмотр списка карт 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).

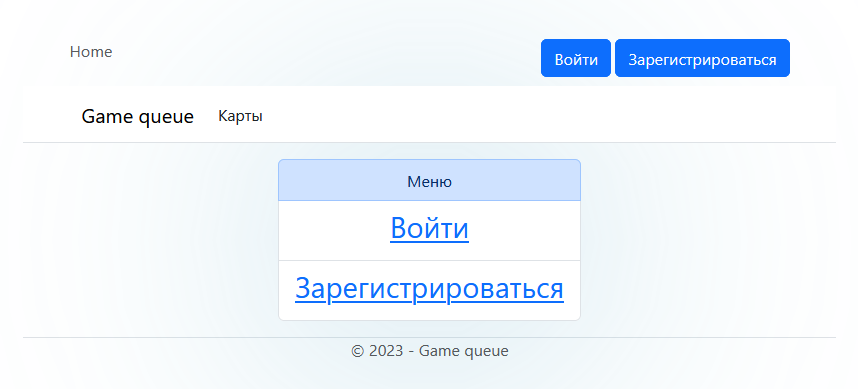
****

Рисунок 9 - Главное меню (неавторизированный пользователь)

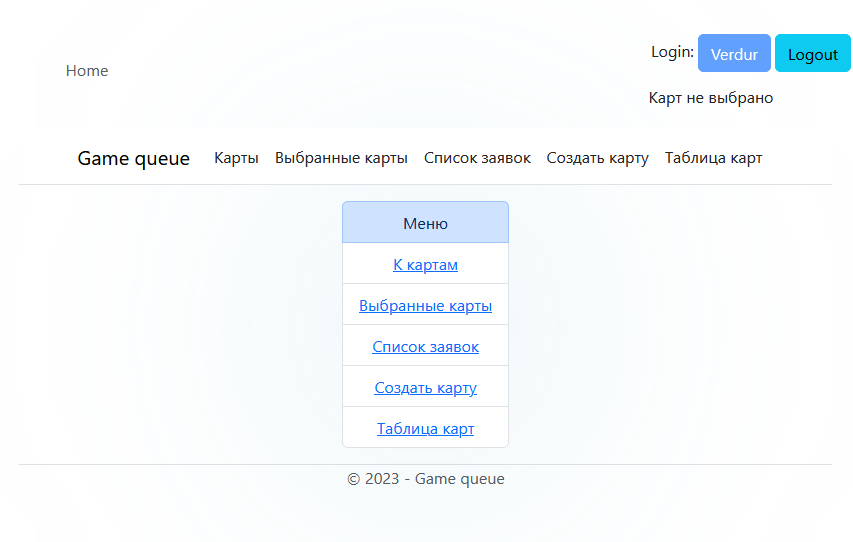
****

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 карт», «выбранные карты» 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.

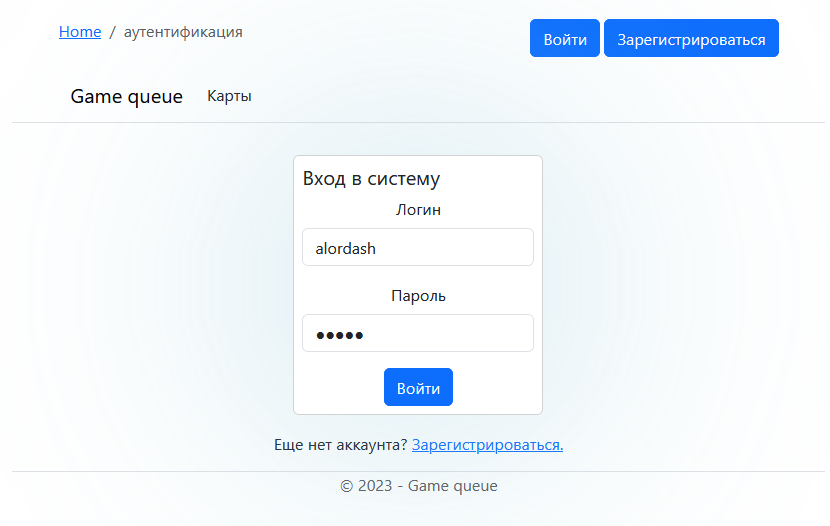
****

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.

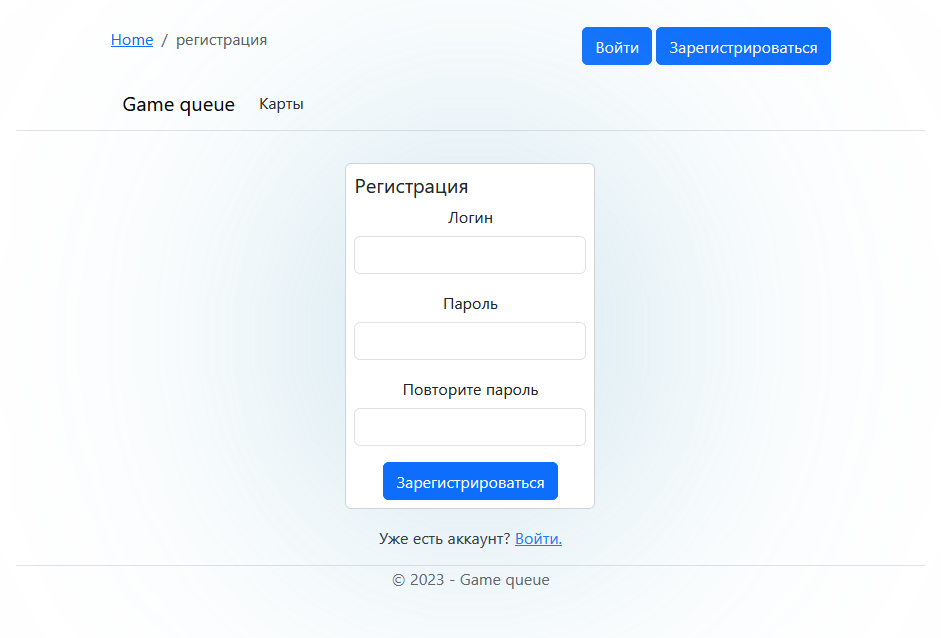
****

Рисунок 12 - Страница регистрации

На странице со списком карт (рис. 13) отображается список карт 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 карт, там можно ввести фрагмент названия карты, который должен присутствовать в названии карт, а также максимальное количество игроков.

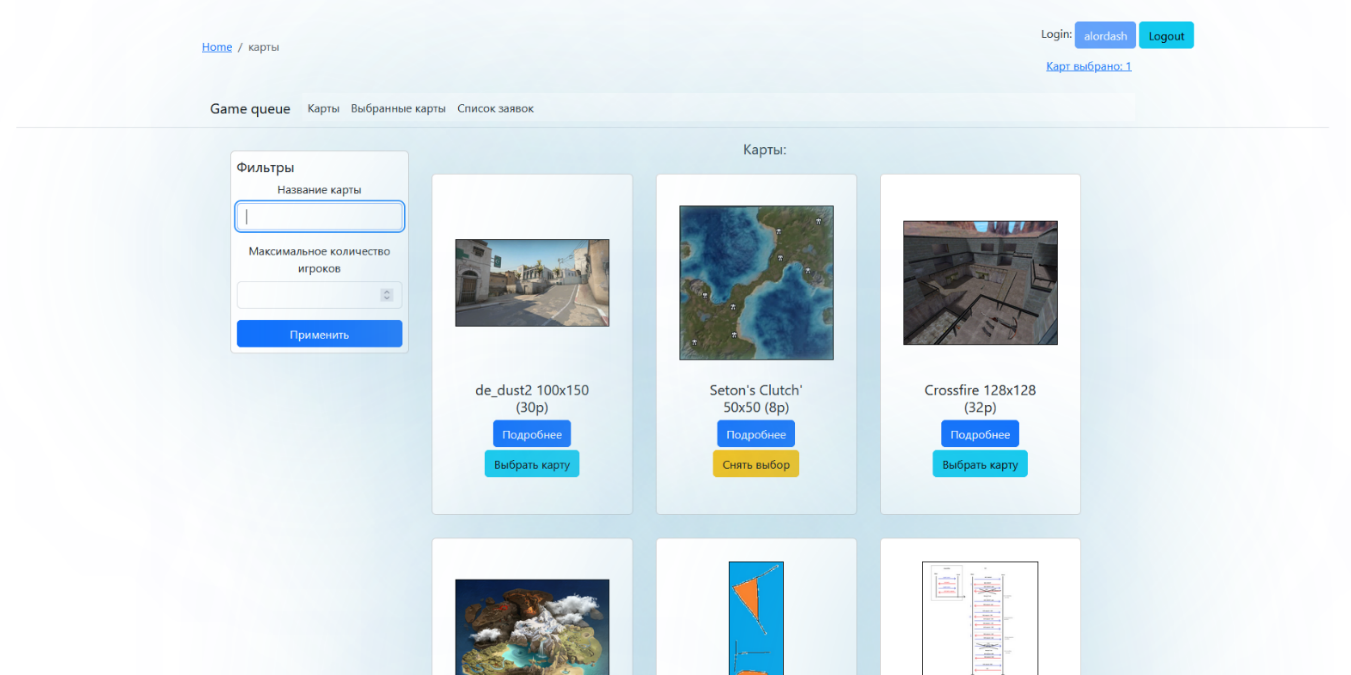
****

Рисунок 13 - Страница со списком карт

На странице с подробным описанием карты (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.

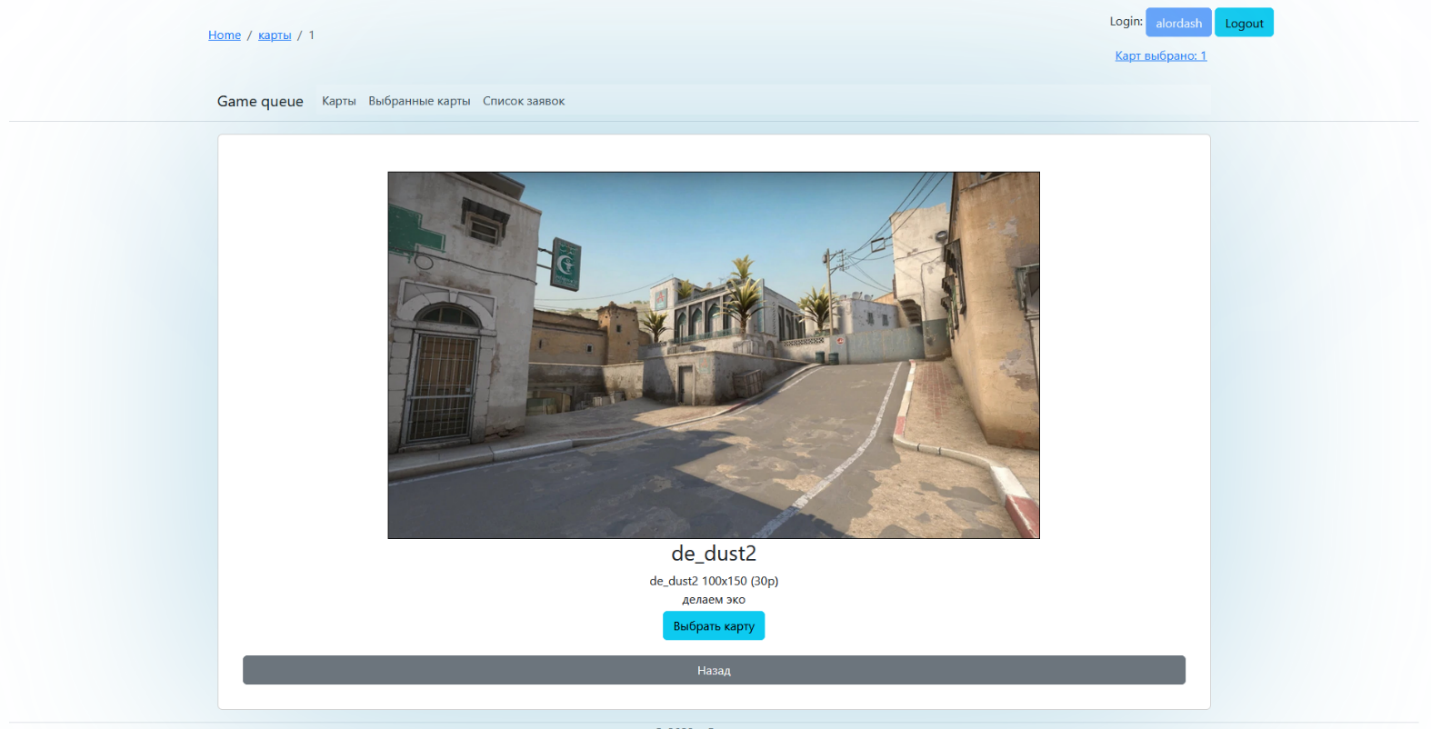
****

Рисунок 14 - Страница с подробным описанием карты

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 карт и кнопка для открытия заявки.

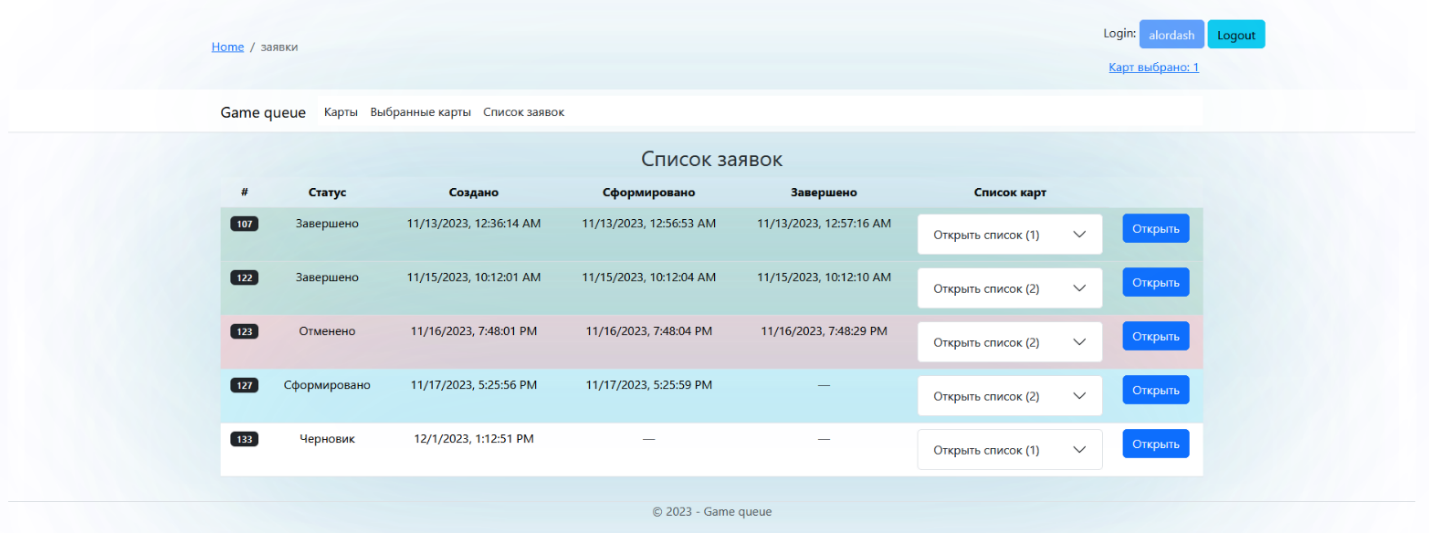
****

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 карт и кнопка для открытия заявки. У заявок со статусом «черновик» также присутствует кнопка для установления статуса заявки.

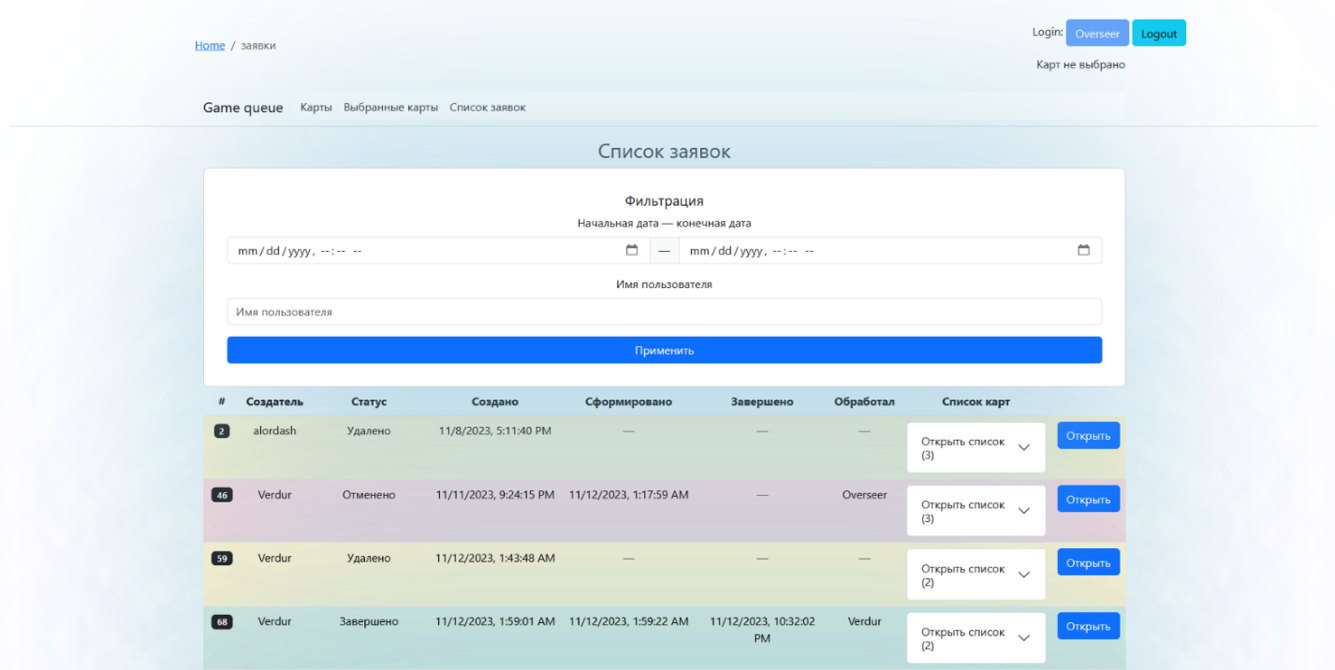
****

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

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

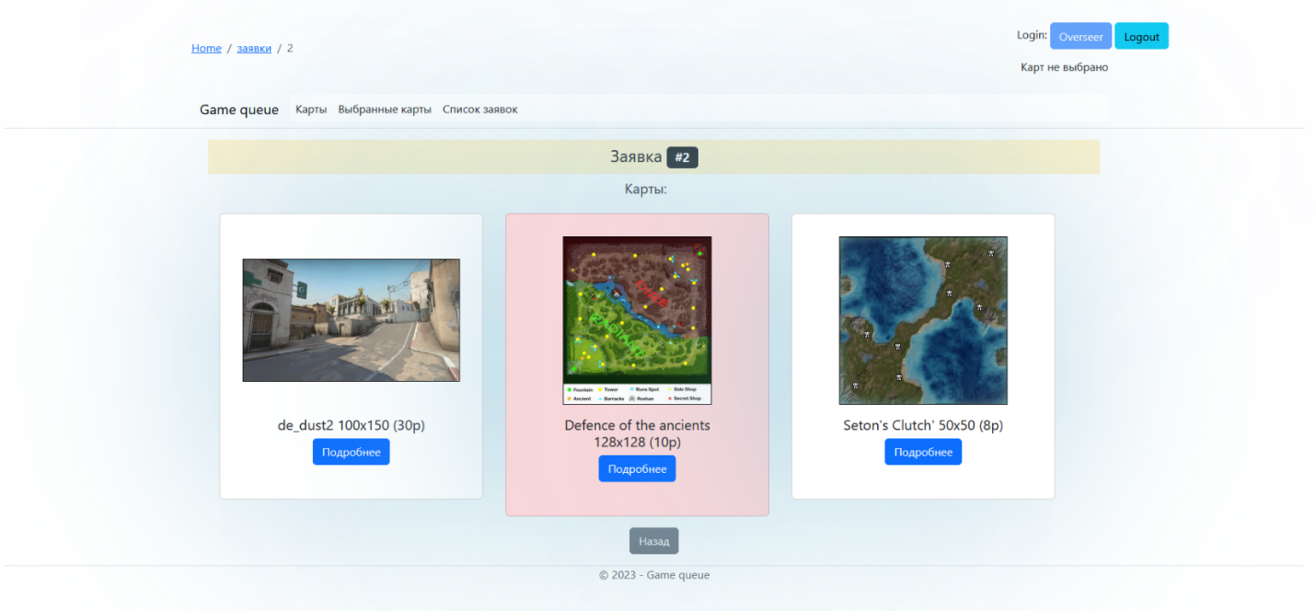
****

Рисунок 17 - Страница с подробным описанием заявки

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

****

Рисунок 18 - Страница с таблицей карт

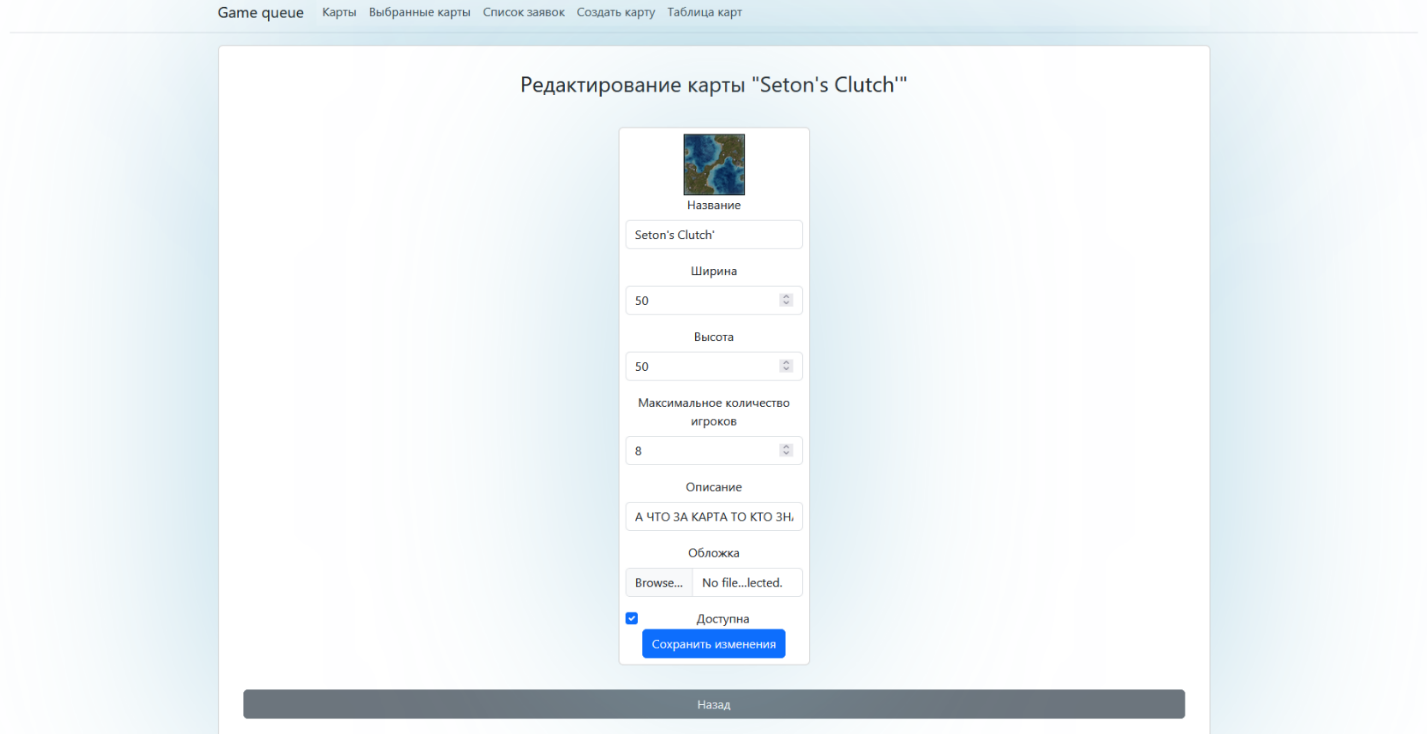
****

Рисунок 19 - Страница редактирования/создания карты

На странице редактирования/создания карты (рис. 19) администратор может отредактировать существующую, или создать новую карту. Для редактирования доступны все поля: название, ширина, высота карты, максимальное количество игроков, описание и изображение-обложка. Также можно скрыть или отобразить карту в списке карт, 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 dotnet 7.0, 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://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 карт для подтверждения новых игр и редактирования игровых локаций-карт.
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: «Автоматический подбор карт для игроков»

Студент: Оразов А.В.

Группа ИУ5-51Б

Преподаватель: Канев А.И.

2023г.

1. **Цель работы**

Реализовать систему для подборка игроками карт (игровых локаций), включающую в себя веб-сервис, веб-приложение, десктопное приложение и выделенный сервис расчета уровня игрока.

1. **Назначение**

Система предназначена для игроков, модераторов и администраторов игры. В игре предусмотрен ограниченный доступ к картам. Для получения доступа пользователю необходимо создать заявку на получение доступа к выбранной карте. Система предоставляет автоматизированный способ создания, учета и ведения заявок. Также она позволяет модераторам и администраторам принимать или отклонять заявки. Администраторы имеют возможность редактировать существующие и создавать новые карты.

1. **Задачи**:
   1. Разработать дизайн приложения.
   2. Создать базу данных в PostgreSQL.
   3. Создать веб-сервис на технологии dotnet 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 игрока в React.
   8. Реализовать интерфейс модератора и администратора в React.
   9. Создать десктопное приложение на Tauri.
   10. Создать выделенный сервис для расчета уровня игроков на Rust и axum.
   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 | /api/authorization | login: string  password: string |  |
| 4.1.2. | POST | Вход в аккаунт | /api/authentication/login | login: string  password: string | auth cookies |
| 4.1.3. | GET | Выход из аккаунта Доступно только авторизированным пользователям | /api/authentication/logout | auth cookies |  |
| * 1. Методы карт | | | | | |
| 4.2.1. | GET | Возвращает файл, сохраненный в файловом хранилище сервиса. urlSuffix – путь до файла. | api/static\_data/{urlSuffix} | urlSuffix: string | Файл |
| 4.2.2. | GET | Возвращает список карт, удовлетворяющих переданным критериям | api/map/ | {  filterName: string,  maxPlayersCount: int  } | {  id: int,  name: string  width: int,  height: int,  maxPlayersCount: int,  coverImageUrl: string?,  mapStatus: MapStatus  }[] |
| 4.2.3. | GET | Возвращает список карт, удовлетворяющих переданным критериям и черновую заявку пользователя | api/map/ | auth cookies  {  filterName: string,  maxPlayersCount: int  } | {  id: int,  name: string  width: int,  height: int,  maxPlayersCount: int,  coverImageUrl: string?,  mapStatus: MapStatus  }[],  searchMapRequest |
| 4.2.4. | GET | Возвращает карту | api/map/{id}/ | id: int | {  id: int,  name: string  width: int,  height: int,  maxPlayersCount: int,  coverImageUrl: string?,  mapStatus: MapStatus  } |
| 4.2.5. | POST | Создает карту  Доступно только администраторам | api/map | auth cookies  {  name: string,  width: int,  height: int,  maxPlayersCount: int,  coverImageUrl: string?,,  coverImageFile: file,  } |  |
| 4.2.6. | PUT | Обновляет информацию о карте  Доступно только администраторам | api/map/{id} | auth cookies  {  id: int,  name: string?,  width: int?,  height: int?,  maxPlayersCount: int?,  coverImageUrl: string?,  coverImageFile: file?  } |  |
| 4.2.7. | DELETE | Помечает карту как удаленную  Доступно только администраторам | api/map/{id} | id: int |  |
| 4.2.8. | PUT | Делает карту доступной  Доступно только администраторам | api/map/make\_available/{id} | id: int |  |
| * 1. Методы заявок | | | | | |
| 4.3.1. | GET | Возвращает список заявок пользователя  Доступно только авторизированным пользователям | api/search\_maps\_request | auth cookies | searchMapsRequest: {  id: int,  creatorUserId: int,  status: SearchMapsRequestStatus,  creationDate: DateTime,  composeDate: DateTime?,  doneDate: DateTime?,  mapsCount: int,  } |
| 4.3.2. | GET | Возвращает подробную информацию о заявке  Доступно только авторизированным пользователям | api/search\_maps\_request/{id} | auth cookies  id: int | {  id: int,  creatorUser: User,  status: SearchMapsRequestStatus,  creationDate: DateTime,  composeDate: DateTime?,  doneDate: DateTime?,  maps: Map[],  } |
| 4.3.3. | GET | Возвращает текущую черновую заявку пользователя  Доступно только авторизированным пользователям | api/search\_maps\_request/current | auth cookies | {  id: int,  creatorUser: User,  status: SearchMapsRequestStatus,  creationDate: DateTime,  composeDate: DateTime?,  doneDate: DateTime?,  maps: Map[],  }? |
| 4.3.4. | PUT | Добавляет в заявку пользователя карту {map\_id}, возвращает id заявки, в которую была добавлена карта  Доступно только авторизированным пользователям | api/search\_maps\_request/add\_map/{map\_id} | auth cookies  map\_id: int | searchMapsRequestId: int |
| 4.3.5. | DELETE | Удаляет из заявки пользователя карту {map\_id}  Доступно только авторизированным пользователям | api/search\_maps\_request/remove\_map/{map\_id} | auth cookies  map\_id: int |  |
| 4.3.6. | PUT | Устанавливает заявке статус «сформирована»  Только если её текущий статус «черновик»  Доступно только авторизированным пользователям | api/search\_maps\_request/compose/{id} | auth cookies  id: int |  |
| 4.3.7. | DELETE | Устанавливает заявке статус «удалена»  Только если её текущий статус «черновик»  Доступно только авторизированным пользователям | api/search\_maps\_request/delete/{id} | auth cookies  id: int |  |
| 4.3.8. | GET | Возвращает отфильтрованный по датам создания и имени пользователя список заявок  Доступно только модераторам | api/search\_maps\_request/all/?begin\_date &end\_date&username | auth cookies  begin\_date: dtetime?  end\_date: datetime?  username: string? | searchMapsRequest[] |
| 4.3.9. | PUT | Устанавливает заявке статус «отклонена» или «завершена»  Только если её текущий статус «сформирована»  Доступно только модераторам | api/search\_maps\_request/status/{id}?status | id: int  status: “Finished” || “Cancelled” |  |
| * 1. Методы пользователей | | | | | |
| 4.4.1. | PUT | Устанавливает уровень пользователя  Вызывается внешним сервисом level-calculator | api/user/{id}?level | id: int  level: int |  |
| 4.4.2. | GET | Возвращает информацию об аккаунте пользователя. | api/user/me | auth cookies | {  id: int,  name: string,  level: int,  role: UserRole  } |
| * 1. Методы внешнего сервиса level-calculator | | | | | |
| 4.5.1. | POST | Рассчитывает уровень игрока после его участия в картах, указанных в заявке Метод внешнего сервиса level-calculator | api/calculate\_level | user\_id: int  search\_maps\_request\_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 карт – переход на страницу 5.4. Только для авторизированных пользователей.
         4. Открыть список выбранных карт – 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 карт – 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 карт. На этой странице располагается список всех карт и панель фильтрации карт.
      1. Доступна всем пользователям.
      2. Выводится информация о картах в виде карточек (метод 4.2.2.).
         1. Изображение карты.
         2. Название карты.
         3. Ширина карты.
         4. Высота карты.
         5. Максимальное количество игроков.
      3. Действия
         1. Отфильтровать карты по указанному названию и максимальному количеству игроков (метод 4.2.2.).
         2. Добавить карту в заявку (метод 4.3.4.). Только для авторизированных пользователей.
         3. Удалить карту из заявки (метод 4.3.5.). Только для авторизированных пользователей.
         4. Подробнее – перенаправляет на страницу 5.5.
   5. Страница с подробным описанием карты
      1. Доступна всем пользователям.
      2. Выводится информация о карте в построчном формате.
         1. Изображение карты.
         2. Название карты.
         3. Ширина карты.
         4. Высота карты.
         5. Максимальное количество игроков.
         6. Описание карты.
      3. Действия
         1. Добавить карту в заявку (метод 4.3.4.). Если карта доступна. Только для авторизированных пользователей.
         2. Удалить карту из заявки (метод 4.3.5.). Только для авторизированных пользователей.
         3. Редактировать – перенаправляет на страницу 5.8. Только для администраторов.
   6. Страница со списком заявок
      1. Доступна только авторизированным пользователям.
      2. Выводится информация о заявках в табличном формате (методы 4.3.1. и 4.3.8.).
         1. Статус заявки.
         2. Дата создания.
         3. Дата формирования.
         4. Дата завершения.
         5. Кем обработана. Только для модераторов и администраторов.
         6. Список выбранных карт (в выпадающем списке).
      3. Действия
         1. Открыть заявку – перенаправляет на страницу 5.7.
         2. Установить заявке статус «отклонена» или «завершена» (метод 4.3.9.). Только для модераторов и администраторов.
         3. Отфильтровать список заявок по диапазону дат и/или имени автора заявки (метод 4.3.8.). Только для модераторов и администраторов.
   7. Страница с подробным описанием заявки. Пользователь может изменить карты в заявке. Модератор может просматривать заявки всех пользователей.
      1. Доступна только авторизированным пользователям.
      2. Выводится информация о заявке в карточной форме. Список карт, выбранных в заявке, в виде карточек (метод 4.3.2.).
         1. Изображение карты.
         2. Название карты.
         3. Ширина карты.
         4. Высота карты.
         5. Максимальное количество игроков на карте.
      3. Действия
         1. Удалить карту из заявки (метод 4.3.5.). Если заявка находится в статусе черновика. Только для владельца заявки.
         2. Вернуть карту в заявку (метод 4.3.4.). Если заявка находится в статусе черновика. Только для владельца заявки.
         3. Сформировать заявку (метод 4.3.6.). Если заявка находится в статусе черновика. Только для владельца заявки.
         4. Удалить заявку (метод 4.3.7.). Если заявка находится в статусе черновика. Только для владельца заявки.
         5. Открыть карту – перенаправляет на страницу 5.5.
   8. Страница редактирования/создания карты. Администратор может изменить существующую или создать новую карту.
      1. Доступна только администраторам.
      2. Выводится информация о карте в построчном формате.
         1. Изображение карты.
         2. Название карты.
         3. Ширина карты.
         4. Высота карты.
         5. Максимальное количество игроков на карте.
         6. Описание
         7. Доступность.
      3. Изменение полей карты. Можно изменять все вышеперечисленные поля карты (методы 4.2.5. и 4.2.6.).
   9. Страница с таблицей карт. Предоставляет администратору удобный способ отображения всех карт.
      1. Доступна только администраторам.
      2. Выводит информация о картах в табличном виде (метод 4.2.2.).
         1. Изображение карты.
         2. Название карты.
         3. Ширина карты.
         4. Высота карты.
         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 5.5.
         3. Добавить карту – перенаправляет на страницу 5.8.
2. **Требования к программному обеспечению:**
   1. Серверная часть
      1. ОС: Linux/Windows
      2. Docker
      3. Докер образы
      4. dotnet/sdk 7.0
      5. dotnet/aspnet 7.0
      6. redis 7.2.3-alpine
      7. nginx 1.19.2-alpine
      8. quay.io minio RELEASE.2022-10-15T19-57-03Z
      9. postgres 14
      10. rust 1.72.0
      11. 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 512 Мб.