|  |  |
| --- | --- |
|  | **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: «Компьютерная игра «Эволюция»»

Студент: Лукьянов К.Е.

Группа ИУ5-53Б

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 игру “Эволюция” с автоматической проверкой выполненных ходов и большим разнообразием карт.

Целью работы является реализация системы для совершения ходов в пошаговой игре “Эволюция”, включающую в себя веб-сервис, веб-приложение, десктопное приложение и выделенный сервис проверки соблюдения правил, при совершении хода.

Система предназначена для игроков и гейм мастеров игры. В игре предусмотрен ограниченный доступ к совершению ходов. Для получения доступа пользователю необходимо создать заявку хода, который он хочет сделать и выбрать использующиеся карты. Система предоставляет автоматизированный способ создания, учета и ведения заявок. Также она позволяет модераторам принимать или отклонять заявки. Гейм мастеры 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, которые игроки могут использовать во время игры[1]. Пользователи могут оставлять заявки на участие в игре с выбранными картами для игры. 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 каждая из них будет разыграна по очереди.

Когда игрок определится с картами, которые ему хочется разыграть в этот ход, он формирует заявку и она отправляется на модерирование. После этого её уже нельзя редактировать. Можно также посмотреть историю своих игр. Чтоб игра проходила полностью с соответствием правилами, каждый ход проходит через модерацию гейм мастера. Для этого он отклоняют и принимают заявки таким образом, чтобы все ходы были сделаны верно. На случай если игрок передумает играть также присутствует возможность отменить заявку.

Карты были взяты из оригинальной версии настольной игры “Эволюция”, все карты были уже сбалансированы и требовалось их перенести в электронное пространство для того, чтоб игроки могли выбирать, какими картами им играть в этот раз[2].Особенности карты видны по её изображению и текстовому описанию. Во время игры с картами могут обнаружиться какие-то ошибки, и на этот случай администратор может временно скрыть карту, чтоб игроки не могли ее использовать. Играть с одними и теми же картами 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).

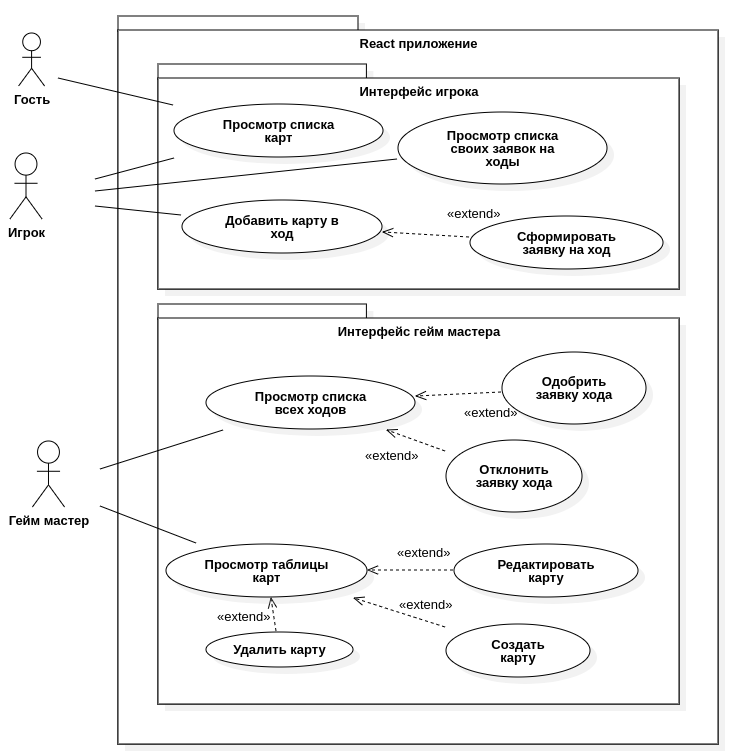


Рисунок 1 - Диаграмма прецедентов

Гостям доступен просмотр карт. Зарегистрированные гости – игроки. Они могут добавлять карты в ходы, просматривать список своих ходов и формировать текущую заявку. Заявки обрабатываются гейм мастерами. В результате обработки заявки её либо одобряют, либо отклоняют. Помимо возможностей принятие/отказа, гейм мастеру 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).

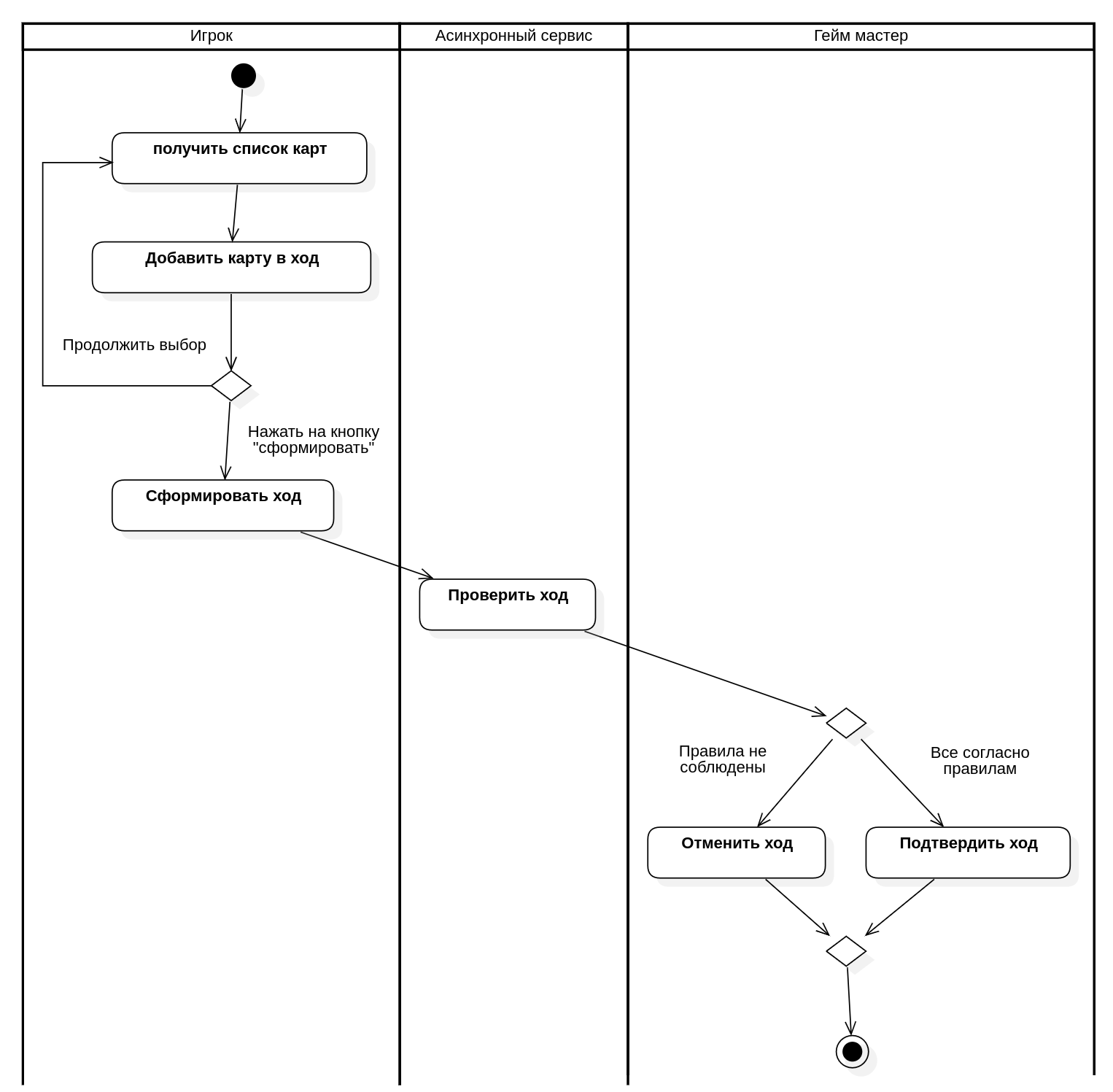


Рисунок 2 - Диаграмма деятельности

Пользователь выбирает карты, затем формирует на основе выбранных карт заявку. Эту заявку затем обрабатывает асинхронный сервис, а затем и гейм мастер. В случае одобрения заявки запускается процесс проверки соблюдения всех правил в асинхронном сервисе, вычисляющий новый статус хода. Возможные состояния заявки отражены на диаграмме состояний (рис. 3).

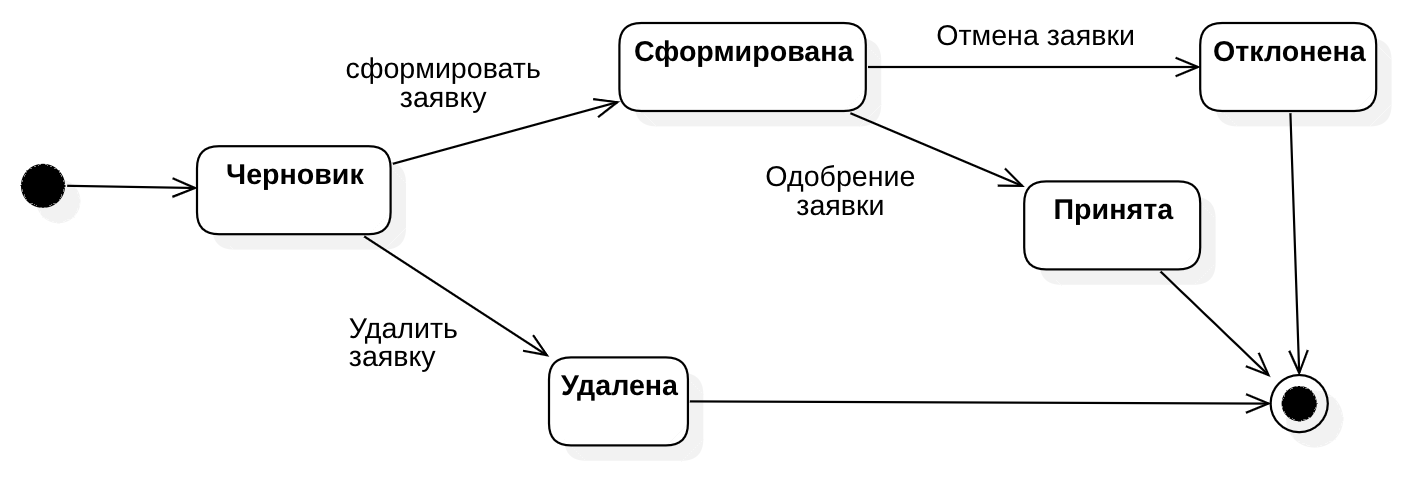


Рисунок 3 - Диаграмма состояний заявки

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 гейм мастер. Он может одобрить или отклонить её.

# **АРХИТЕКТУРА**

Архитектура системы представлена на диаграмме развертывания (рис. 4). Бэкенд разворачивается на локальном ноутбуке. Такое решение было принято, поскольку бэкенд не требует масштабируемости и может успешно функционировать в отдельном окружении. В дополнение к бэкенду, в системе присутствует асинхронный сервис, который разворачивается в контейнере Docker. Этот сервис предоставляет дополнительные функциональности и может быть масштабирован в зависимости от потребностей. Дополнительно, в системе используются следующие Docker-образы, описанные в docker-compose.yml файле [3]: PostgreSQL, Redis, Minio. Этот подход позволяет легко развертывать и управлять каждым компонентом системы независимо, а также обеспечивает гибкость в выборе платформы для запуска Docker-контейнера. Сервис turns-checker разворачивается отдельно и не обязательно на том же компьютере, что и основной сервис.

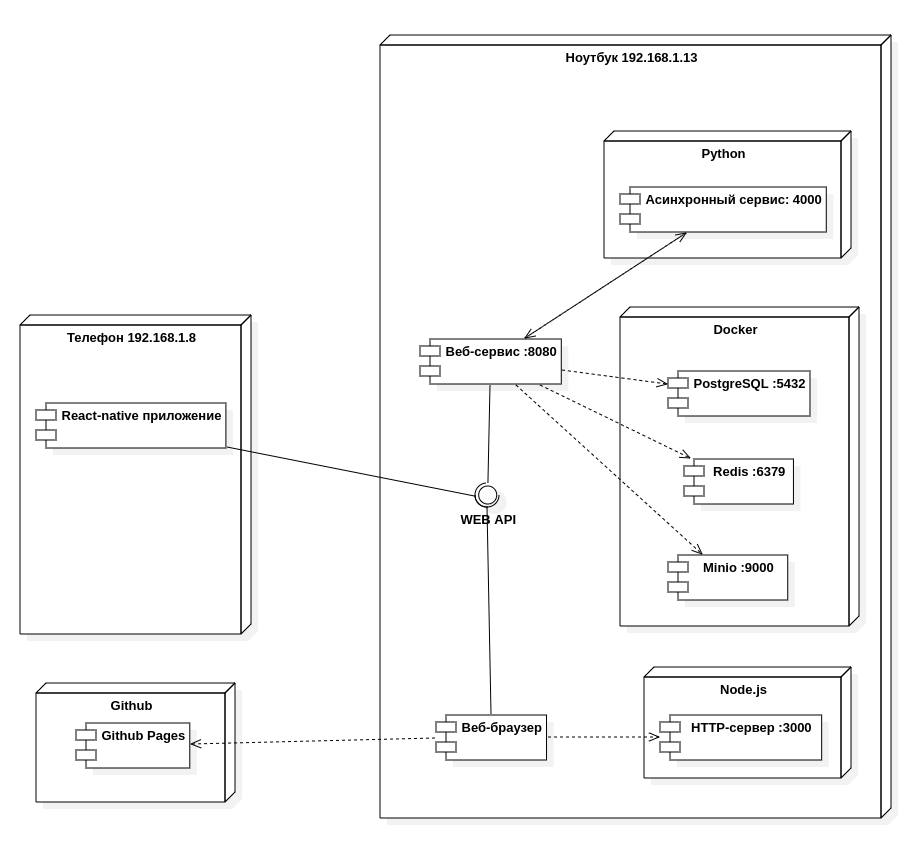
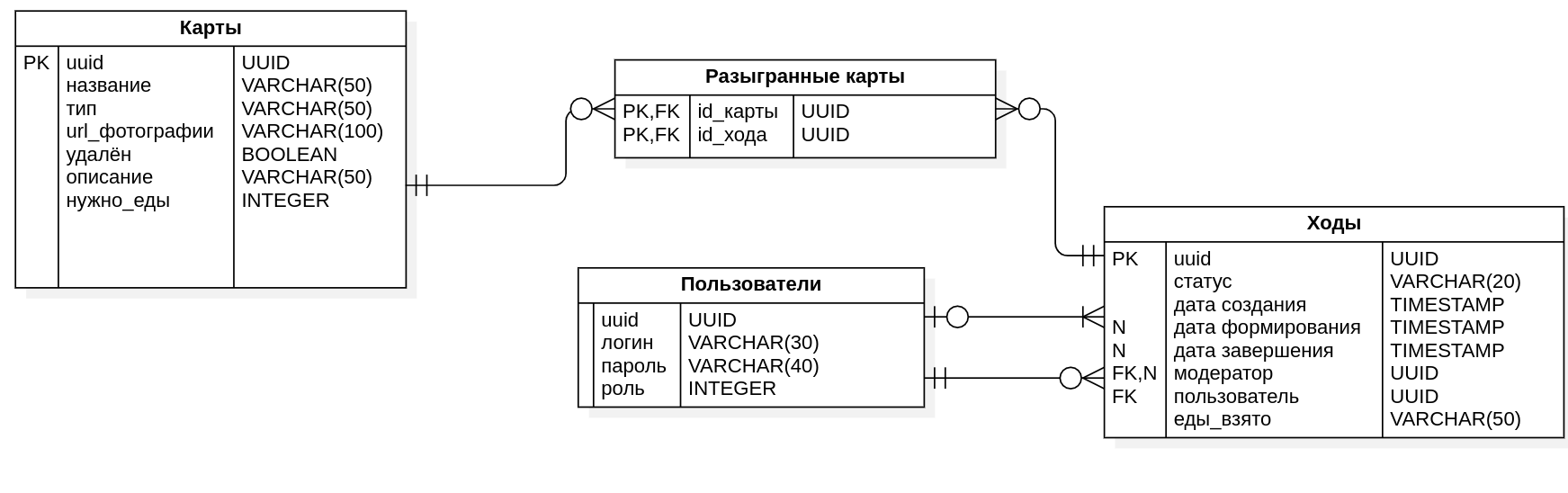
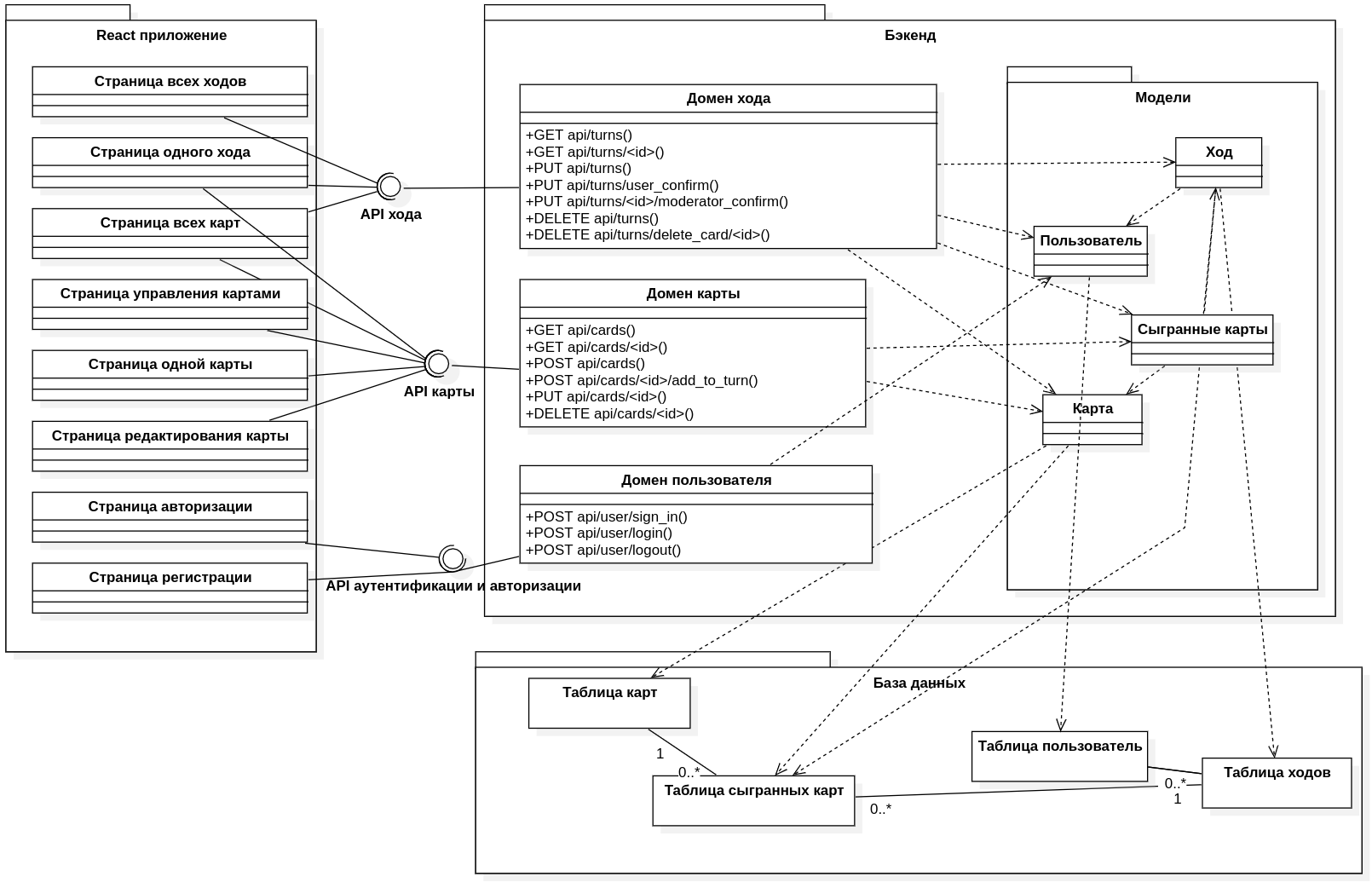


Рисунок 4 - Диаграмма развертывания

Нативное [4] и браузерное приложения [5, 6] обращаются к веб-сервису, основанному на технологии Gin v1.9.1 [7, 8], через REST Web-API. Использование фреймворка Gin обусловлено тем, что эта технология является проверенным и быстродействующим решением, стандартом индустрии. Современность языка GoLang также склоняет к выбору технологии. Данные хранятся в СУБД PostgreSQL [9], 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 роли. Модель карты представляет собой набор полей, необходимых исключительно для бизнес-логики. Для хранения в одной заявке нескольких карт используется промежуточная таблица разыгранные карты, реализующая связь М-М. Устройство бэкенда приложения изображено на диаграмме классов бэкенда (рис 6.).

Рисунок 5 - ER диаграмма

Устройство бэкенда приложения изображено на диаграмме классов бэкенда (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 бэкенда отражена на диаграмме классов фронтенда (рис. 7). Ключевые страницы имеют связь с API аутентификации, т.к. доступ к ним осуществляется только для авторизированных пользователей (игроков) с определенными правами (ролями).

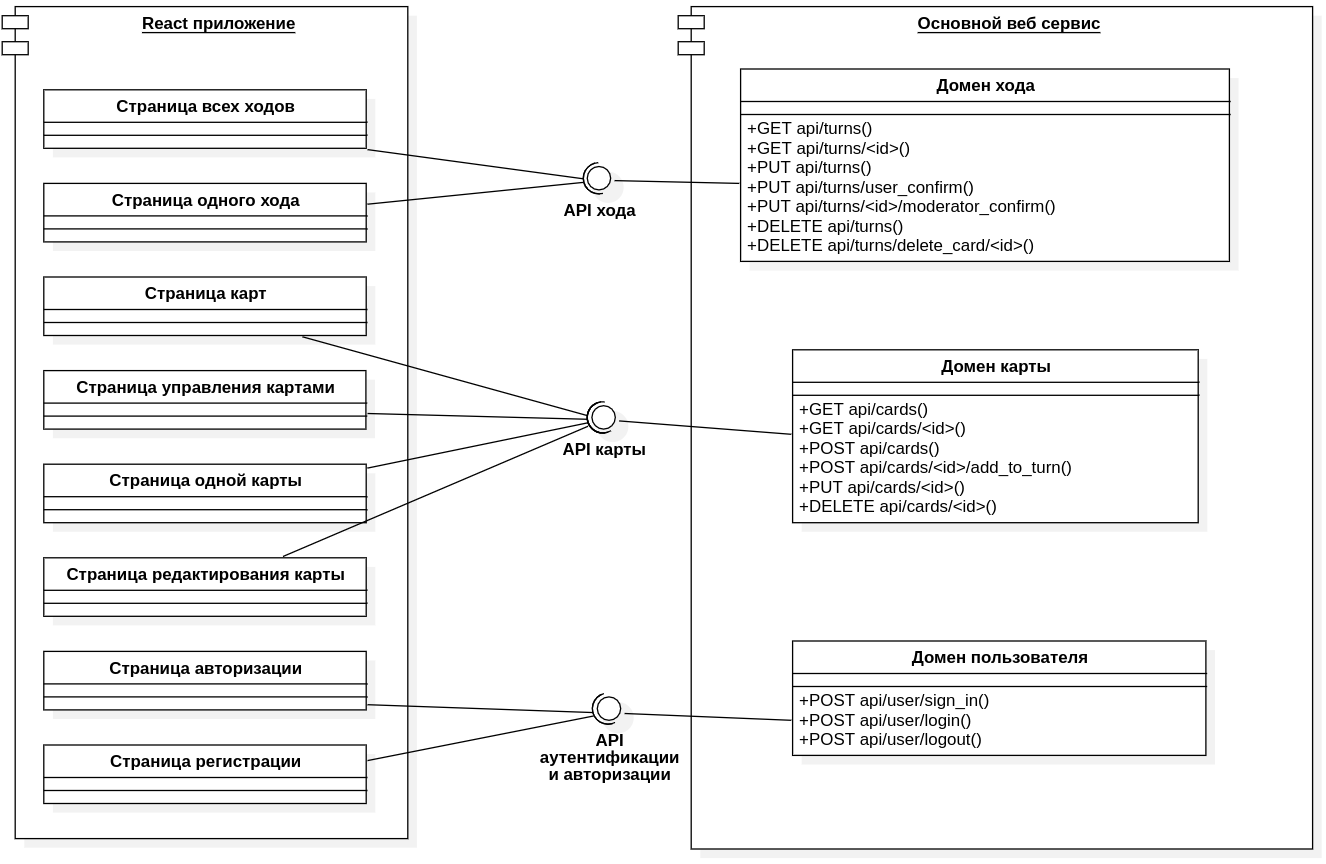
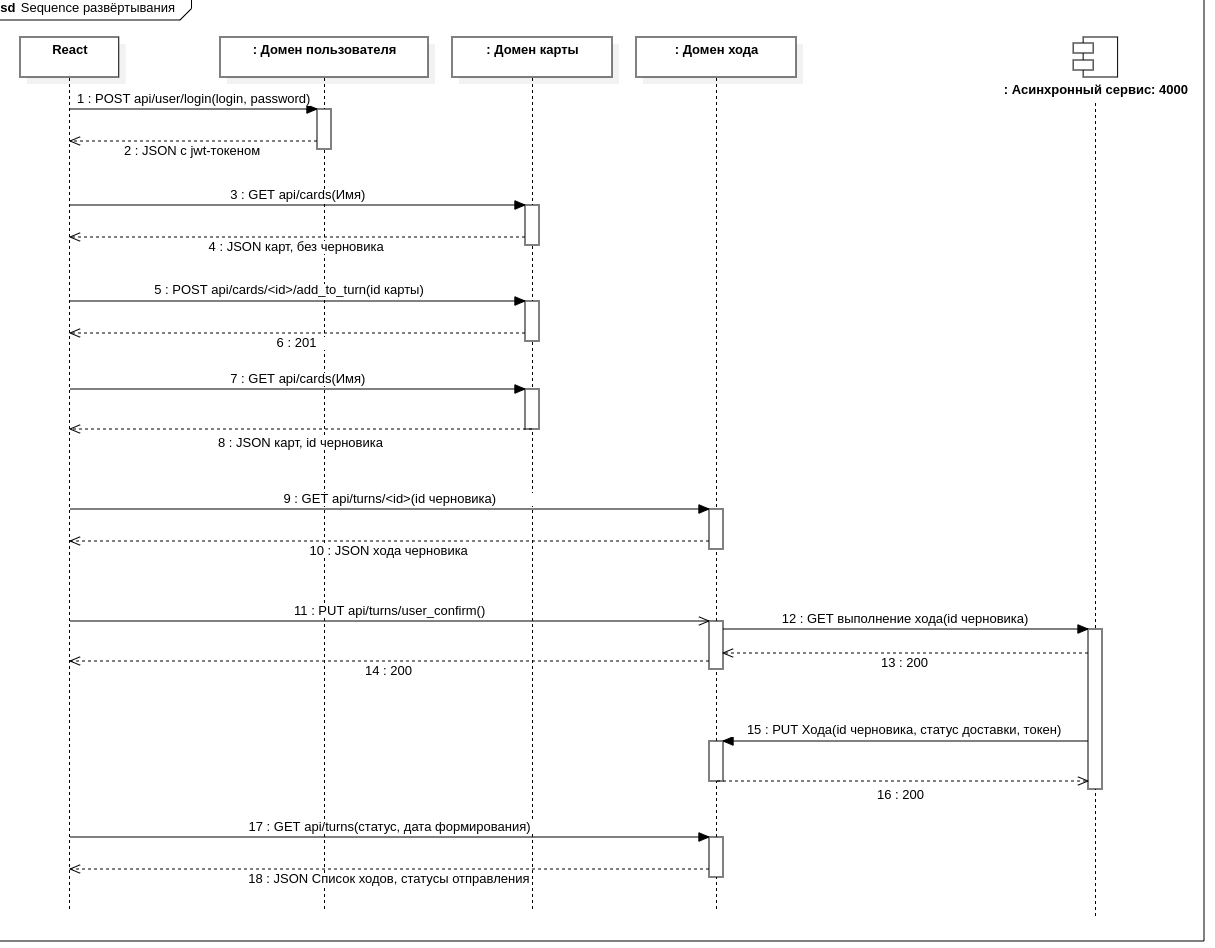


Рисунок 7 *-* Диаграмма классов фронтенда

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

Рисунок 8 - Диаграмма последовательности

В начале бизнес-процесса происходит аутентификация игрока. Для этого он отправляет через графический интерфейс запрос, передавая в нем имя, 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 добавить, и, нажимая на кнопку «плюс» 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 асинхронному сервису на то, чтобы он проверил соблюдение правил. Когда заявка будет одобрена, пользователь сможет увидеть через некоторое время, что его ход одобрили.

Процесс рассмотрения заявок происходит также через графический интерфейс. Гейм мастер может 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. Также через графический интерфейс они могут управлять непосредственно картами. Им доступны такие функции, как создание и редактирование карты, просмотр списка карт и удаление их. Для каждой из этих функция присутствует свой метод, отправляемый на основной веб-сервис.

# **ОПИСАНИЕ ИНТЕРФЕЙСА**

Страница с формой авторизации (рис. 9)

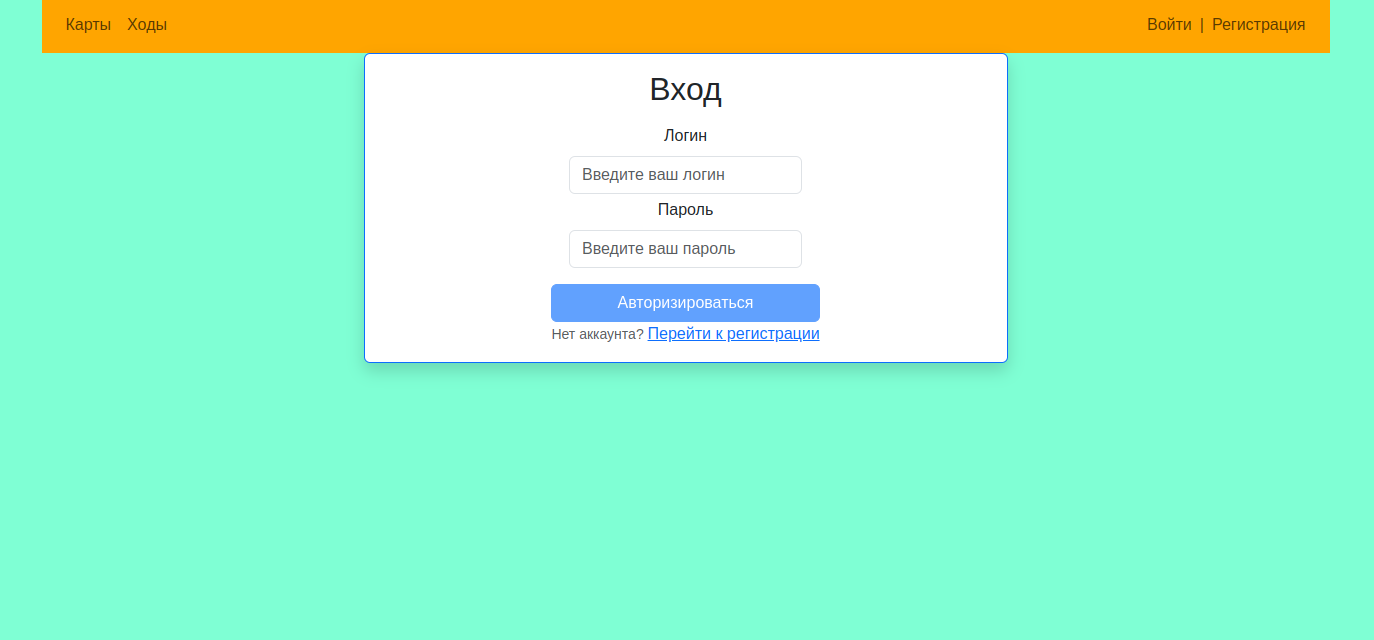


Рисунок 9 - Страница авторизации

На этой странице 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.

Страница с формой регистрации (рис. 10)

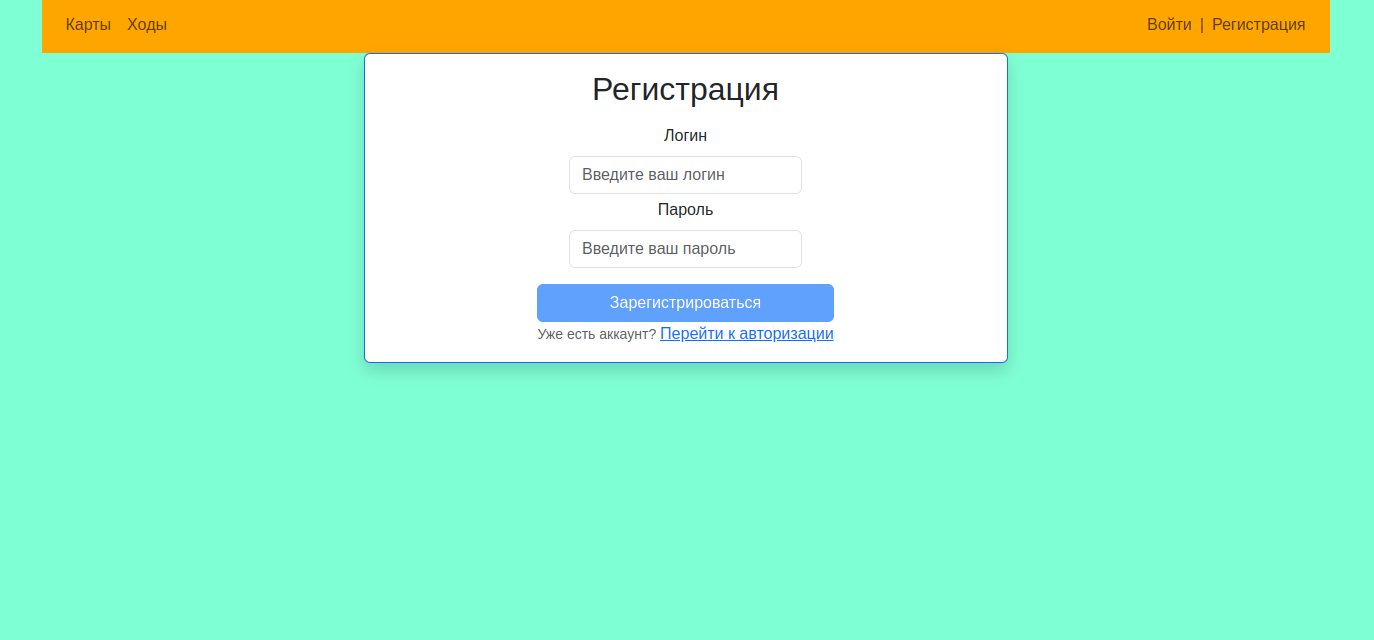
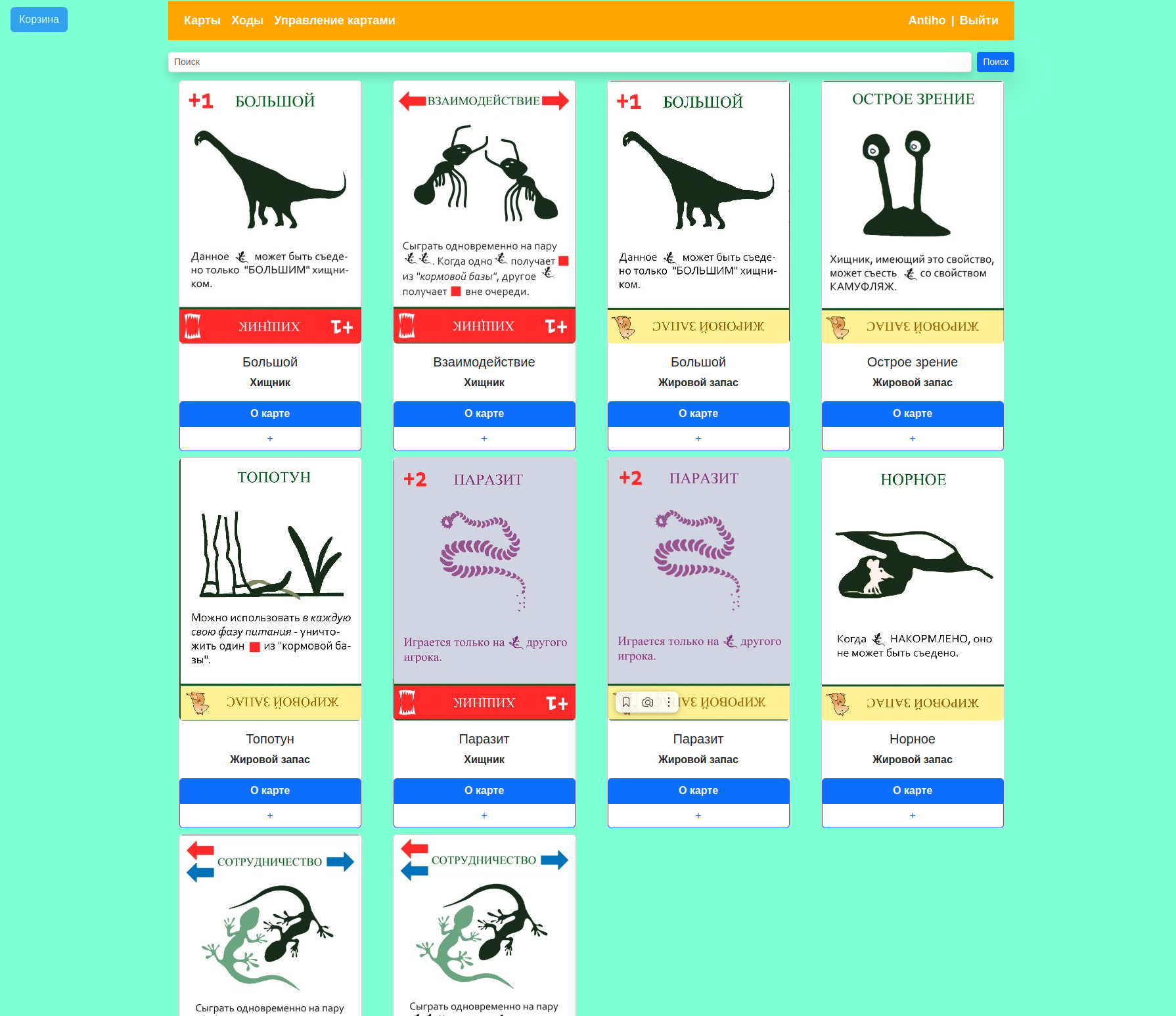


Рисунок 10 - Страница регистрации

На этой странице гости могут завести аккаунт. Для этого нужно указать логин и пароль, и повторить пароль.

Страница со списком карт (рис. 11)

Рисунок 11 - Страница со списком карт

На этой странице отображается список карт в виде карточек. У каждой карты есть хендлер нажатия для перехода в режим «О карте», переносящая пользователя на страницу с подробной информацией о карте.

Страница с подробным описанием карты (рис. 12)



Рисунок 12 - Страница с подробным описанием карты

На этой странице отображается подробная информация о карте: название, тип, сколько требует еды и текстовое описание.

Страница со списком заявок (рис. 13, 14)

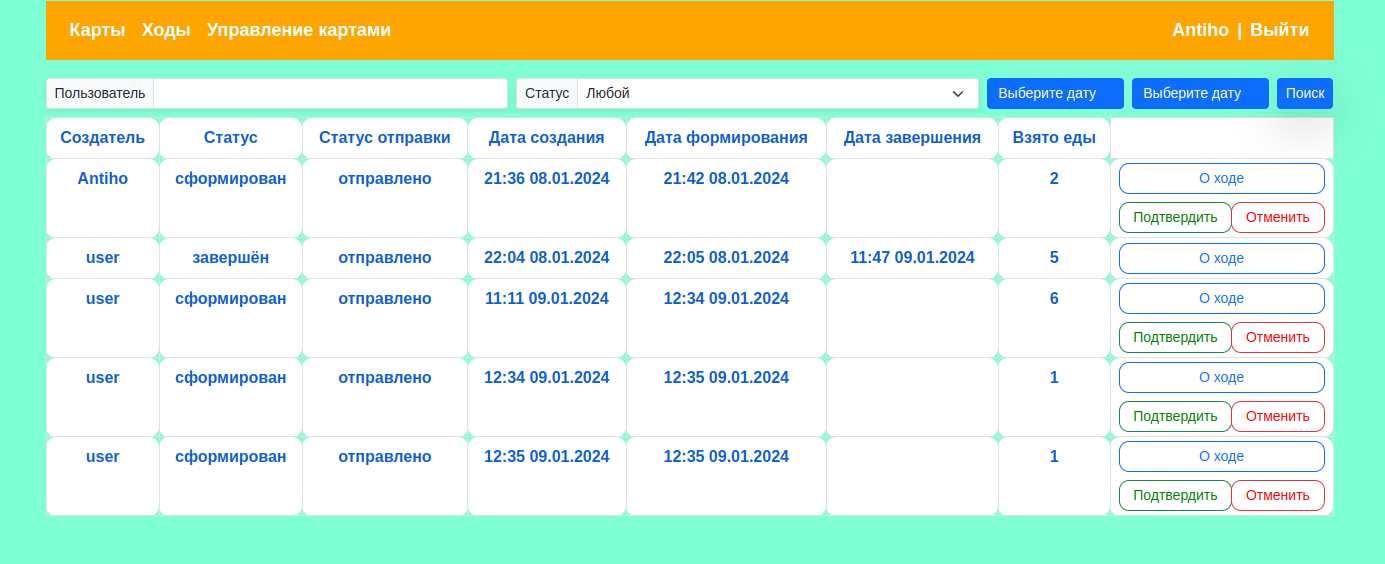
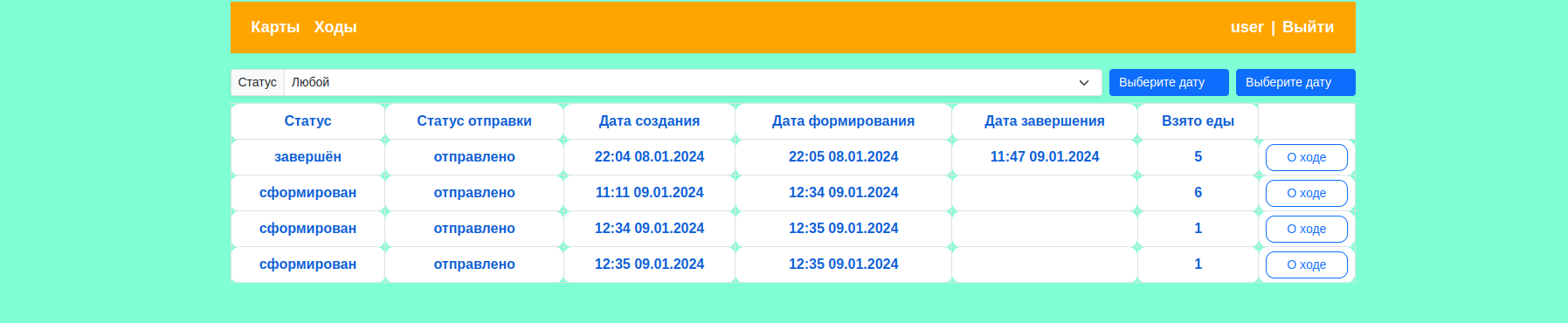
****

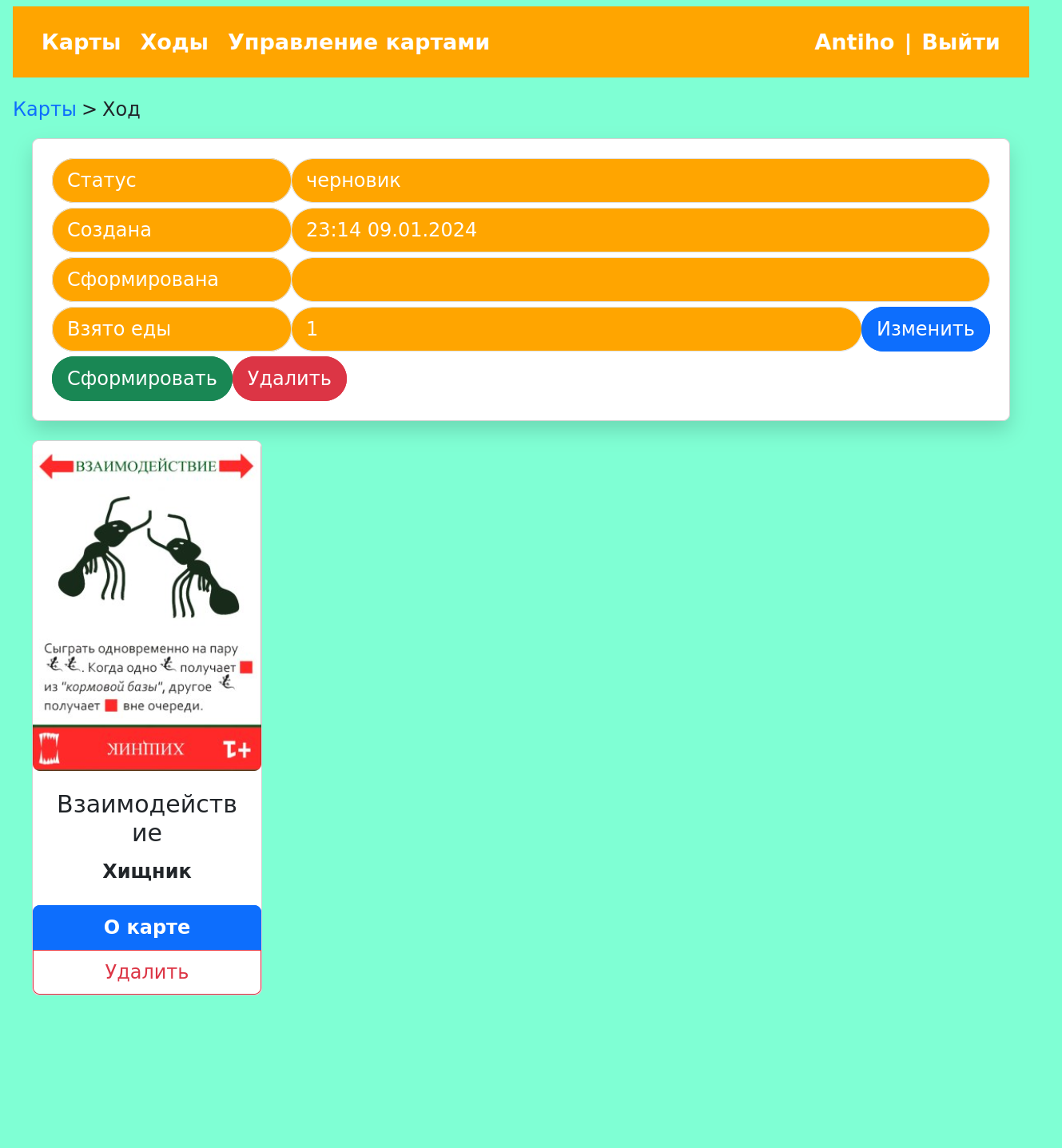
Рисунок 13 - Страница со списком заявок (гейм мастер)

На этой странице отображается список заявок. В зависимости от типа пользователя этот список будет функционально отличаться. Так, для пользователей отображается список созданных ими заявок: Статус, статус отправки, дата создания, дата окончания хода, дата формирования, дата принятия, дата начала хода, и количество еды.

Рисунок 14 - Страница со списком заявок (игрок)

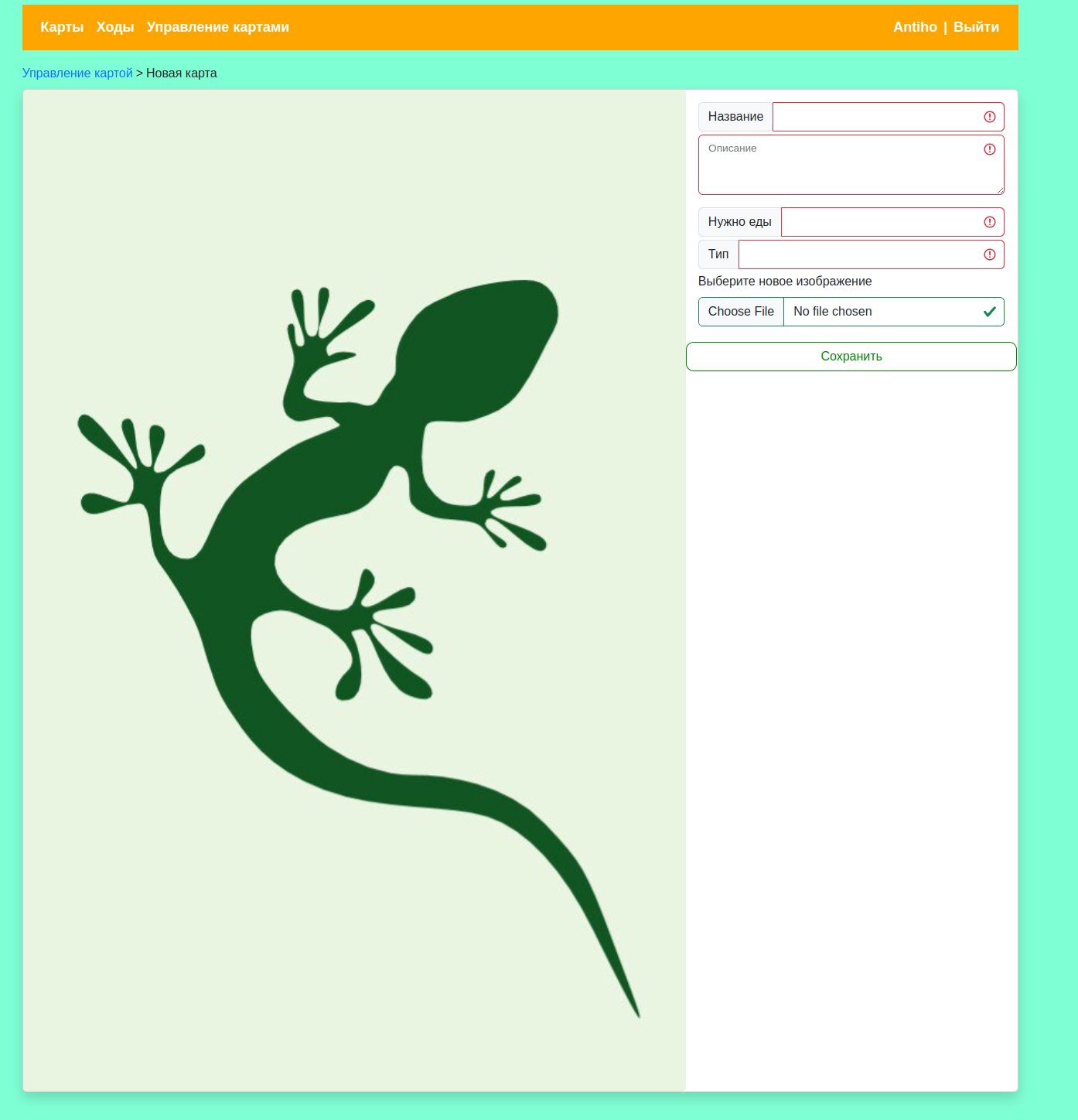
Для гейм мастера функционал этой страницы шире. Для них отображается список всех заявок всех пользователей с более подробной информацией: Создатель, статус, статус отправки, дата создания, дата формирования, дата завершения, количество взятой еды и возможность принять или отклонить. Кроме того, присутствует фильтрация по пользователю.

Страница с подробным описанием заявки (рис. 15)

Рисунок 15 - Страница с подробным описанием заявки

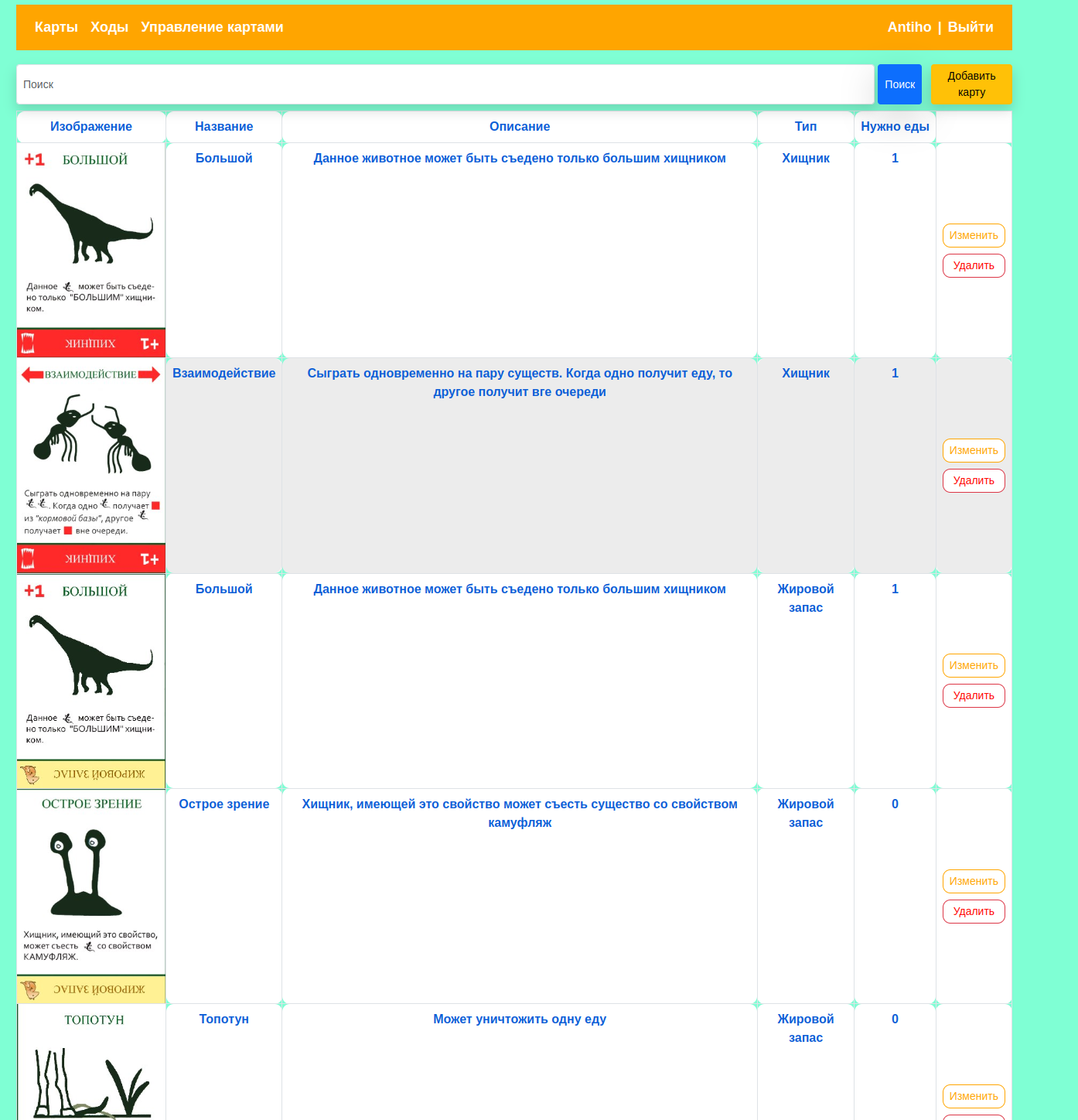
На этой странице отображается подробная информация о заявке. Список выбранных карт в виде самих карточек, а также статус заявки, дата формирования и её автор.

Страница создания карточки (рис. 16)

Рисунок 16 – Страница создания карты

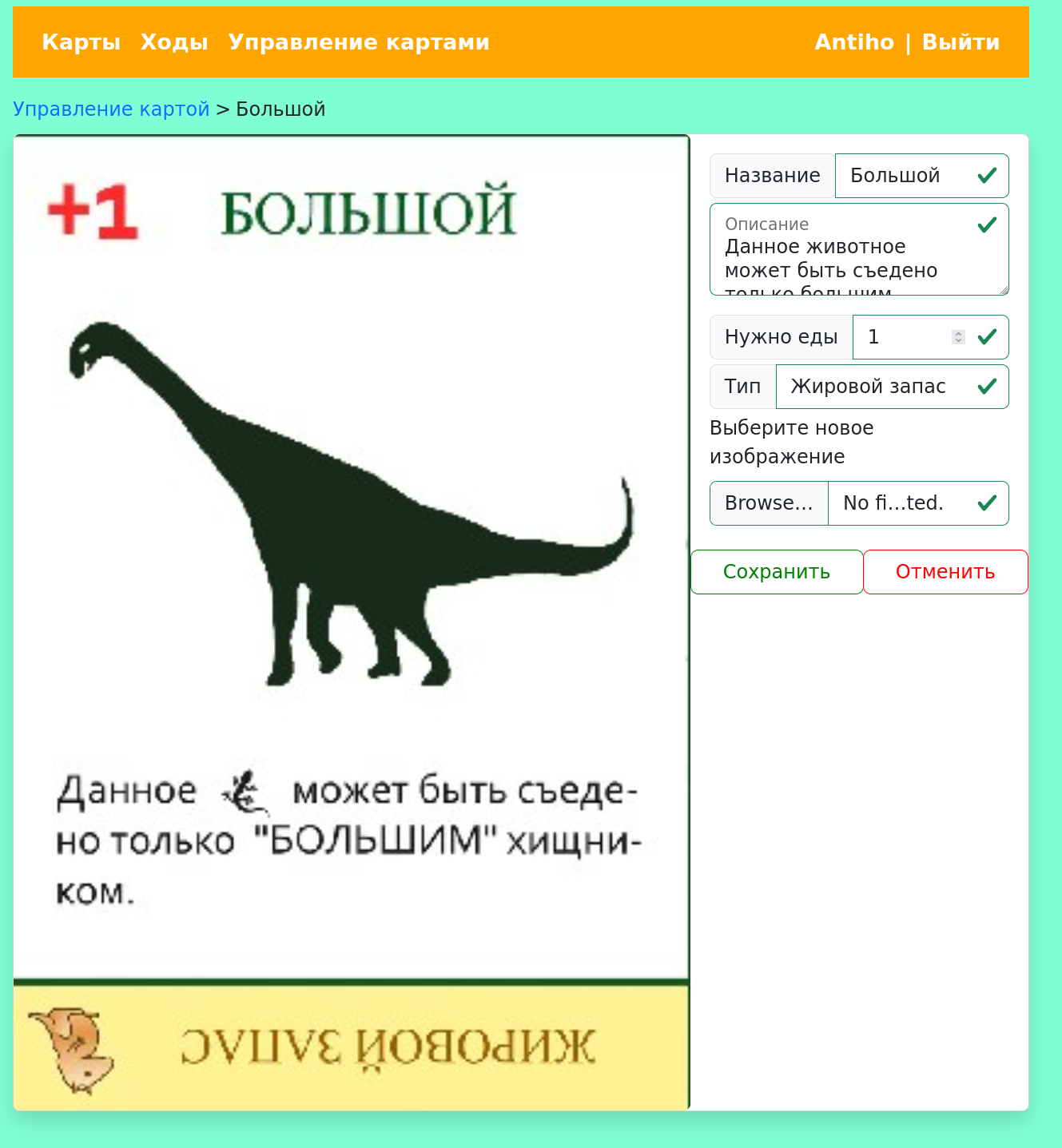
На этой странице модератор может создать новую карту. Для создания доступны все поля: название, описание и фотография картинки, тип и сколько требует еды.

Страница с таблицей карт (рис. 17)

Рисунок 17 - Страница с таблицей карт

На этой странице исторический куратор может в компактном и удобном формате просмотреть список всех карт, существующих в системе и редактировать информацию о карте.

Страница с редактированием карты (рис. 18)

Рисунок 18 - Страница с редактированием карты

Здесь гейм мастер может отредактировать параметры карты для ее баланса и для устранения ошибок.

# **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 1.20, с использованием веб-фреймворка Gin.
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://antihoman.github.io/RIP\_labs\_frontend/.
6. В веб-сервис добавлена авторизация через JWT, а методы задокументированы через Swagger.
7. Реализован интерфейс пользователя был реализован. Доступ к нему имеют только авторизованные пользователи.
8. Реализован интерфейс модератора карт для создания новых ходов, редактирования существующих.
9. Было реализовано мобильное приложение на SwiftUI, повторяющее интерфейс веб-приложения на React.
10. Подготовлен набор документации, включающий РПЗ, ТЗ и набор UML диаграмм.
11. Исходный код проекта доступен в GitHub https://github.com/Antihoman/RIP\_labs

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Руководство по правилам игры “Эволюция” [Электронный ресурс] // Documentation URL: https://add-hobby.ru/evolution.html (дата обращения: 05.08.2023)
2. All Evolution cards [Электронный ресурс] // Documentation URL: https://printgames.ru/wp-content/uploads/2016/01/evolustions.pdf (дата обращения 05.08.2023).
3. Полное практическое руководство по Docker [Электронный ресурс] // Habr. URL: https://habr.com/ru/articles/310460/ (дата обращения: 20.10.2023).
4. Tauri guide [Электронный ресурс] // Tauri. URL: https://tauri.app/v1/guides/ (дата обращения: 02.11.2023).
5. Руководство по React [Электронный ресурс] // Metanit. URL: https://metanit.com/web/react/ (дата обращения: 12.09.2023).
6. Quick Start – React [Электронный ресурс] // React. URL: https://react.dev/learn (дата обращения: 10.09.2023).
7. Руководство по ASP.NET Core 7 [Электронный ресурс] // Metanit. URL: https://metanit.com/sharp/aspnet6/ (дата обращения: 04.10.2023).
8. ASP.NET документация [Электронный ресурс] // Microsoft. URL: https://learn.microsoft.com/en-us/aspnet/core/ (дата обращения: 04.10.2023).
9. 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»**

Техническое задание

по дисциплине «Разработка интернет приложений»

Тема предметной области «Компьютерная игра «Эволюция»»

Студент: Лукьянов К.Е.

Группа ИУ5-53Б

Преподаватель: Канев А.И.

Москва, 2023

# **1. Цель**

Разработка пошаговой компьютерной игры "Эволюция", состоящей из веб-сервиса, веб-приложения, кроссплатформенного приложения и службы, позволяющей обеспечивать эффективное взаимодействие между игроками.

# **2. Назначение**

Система предназначена для управления компьютерной игрой "Эволюция", в которой участвуют зарегистрировавшиеся игроки. Система предоставляет интерфейс для просмотра и изменения информации как о самой игре, так и о ходах развития. Игровые мастера могут изменять список доступных эволюционных путей и выполнять модерацию событий, то есть завершать и отклонять их. Главная задача системы – автоматизация процесса регистрации ходов и контроль за их осуществлением в игре "Эволюция".

# **3. Задачи**

* 1. Разработка базового дизайна системы;
  2. Создание структуры базы данных PostgreSQL;
  3. Реализация REST-API на GO;
  4. Реализации интерфейса гостя на React на React-Vite;
  5. Добавление авторизации, аутентификации и Swagger;
  6. Добавление интерфейса создателя хода в игре;
  7. Добавление нативного приложения React Native, Redux Toolkit;
  8. Работа с асинхронными сервисами для определения результатов ходов;
  9. Разработка интерфейса модератора React;
  10. Разработка и документирование ТЗ, РПЗ и диаграммы.
  11. Развернуть веб-приложение React на Github Pages.

**4. Методы веб-сервиса**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № | Метод | URL | | Описание | Входные данные | | Выходные данные |
| Карточки | | | | | | | |
| 4.1.1 | GET | api/crads/ | | Возвращает список карт, отфильтрованные по их названию, если такое указывалось и черновую заявку пользователя | jwt token  {  name: string  } | | {  {  uuid:string card\_count:int  }  [  {  uuid:string,  name: string,  image\_url: varchar,  type: string,  need\_food: int,  description: string  },  …  {}  ]  } |
| 4.1.2 | POST | api/crads/ | | Добавление карты.  Доступно только модератору | jwt token  {  name: string,  type: string, image\_url: varchar,  need\_food: int,  description: string  } | | {  uuid: string,  name: string,  type: string, image\_url: varchar,  need\_food: int,  description: string  } |
| 4.1.3 | GET | api/crads/<id> | | Возвращает карту по уникальному id | id:int | | {  uuid:int,  name: string,  type: string, “image\_url”: string,  “need\_food”: int,  “description”: string  } |
| 4.1.4 | PUT | api/crads/<id> | | Обновляет информацию о карте.  Доступно только модератору | jwt token  {  id: int,  name: string,  type: string, image\_url: string,  need\_food: int,  description: string  } | | {  card\_id:int,  name: string,  type: string, image\_url: varchar,  need\_food: int,  description: string  } |
| 4.1.5 | DELETE | api/crads/<id>/ | | Меняет статус карты на «Удален». Доступно только модератору | id:int | |  |
| 4.1.6 | POST | api/crads/<id>/add\_to\_turn | | Добавляет в черновой ход карту {id}, возвращает число карт в черновом ходе.  Доступно только авторизированным пользователям | jwt token  id:int | | card\_count:int |
| Заявки - ходы | | | | | | | |
| 4.2.1 | GET | api/turns/ | | Возвращает список ходов. Доступна фильтрация ходов.  Доступно только авторизованным пользователям | jwt token  Status: string, formation\_date\_start: datetime,  formation\_date\_end: datetime | | {  turns:  [  {  uuid: int,  status: string, creation\_date: timestamp, formation\_date: timestamp, completion\_date:timestamp, moderator\_id: varchar,  customer\_id: varchar,  take\_food: string  },  …  {…}  ]  } |
| 4.2.2 | GET | api/turns/<id>/ | | Возвращает информацию о ходе и использованных в нем карт. Доступно авторизованным пользователям | jwt token  id:int | | {  turn:  {  uuid: int,  status: string, creation\_date: timestamp, formation\_date: timestamp, completion\_date:timestamp, moderator\_id: varchar,  customer\_id: varchar, take\_food: string,  }  crads:  [  {  card\_id:int,  name: string,  type: string, image\_url: varchar,  need\_food: int,  description: string  },  …  {…}  ]  } |
| 4.2.3 | PUT | api/turns/<id>/ | | Обновление информации о ходе. Доступно авторизованным пользователям | { take\_food:string  } | | {  uuid: int,  status: string, creation\_date: timestamp, formation\_date: timestamp, completion\_date:timestamp, moderator\_id: varchar,  customer\_id: varchar, take\_food: string  } |
| 4.2.4 | DELETE | api/turns/<id>/ | | Меняет статус заявки на «Удален». Доступно авторизованным пользователям | jwt token  id:int | |  |
| 4.2.5 | DELETE | api/turns/delete\_card/<id> | | Удаление карты из хода. Доступно авторизованным пользователям | jwt token  id: int | |  |
| 4.2.6 | PUT | api/turns/<id>/moderator\_confirm | | Меняет статус с «Сформирован» на «Отклонен» или «Завершен». Доступно модератору | jwt token  {  confirm: boolean  } | |  |
| 4.2.7 | PUT | api/turns/<id>/user\_confirm | | Меняет статус с «Черновик» на «Сформирован».  Доступно авторизованным пользователям | jwt token | |  |
| Внешний сервис | | | | | | | |
| 4.4.1 | POST | api/sending | | Рассчитывает новый статус доставки хода | {  id: int  } | | {  status: varchar,  token: varchar  } |
| Пользователь | | | | | | | |
| 4.3.1 | POST | | api/user/sign\_up | Регистрация | {  login: string,  password: varchar  } |  | |
| 4.3.2 | POST | | api/user/login | Авторизация | {  login: varchar,  password: varchar  } | jwt token | |
| 4.3.3 | POST | | api/user/logout | Выход из аккаунта | Auth jwt |  | |

# **Функциональные требования**

## Главное меню.

### Доступна всем пользователям.

### Действия

#### Регистрация – переход на страницу 5.2. Для гостей.

#### Войти – переход на страницу 5.3. Для гостей.

#### Список карт – переход на страницу 5.4. Для всех.

#### Список выбранных карт для хода – переход на страницу 5.7. Для авторизованных пользователей.

#### Список ходов – переход на страницу 5.6. Для авторизованных пользователей.

#### Добавить ход – переход на страницу 5.8. Для администраторов.

#### Таблица ходов – переход на страницу 5.9. Для администраторов.

## Страница регистрации.

### Доступна гостям.

### Действия

#### Регистрация – отправляет запрос (метод 4.3.1), в котором передаются введенные данные.

#### Войти – перенаправление на страницу 5.3.

## Страница авторизации.

### Доступна гостям.

### Действия

#### Войти – отправляет запрос (метод 4.3.2).

#### Регистрация – перенаправление на страницу 5.2.

## Страница со списком ходов (список карт и панель фильтрации).

### Доступна всем пользователям.

### Информация о каждой карте в виде отдельной карточки (метод 4.1.1)

#### Изображение карты.

#### Название карты.

#### Тип карты.

### Действия

#### Фильтрация карт по названию (метод 4.1.1).

#### «Выбранные карты» – перенаправление на страницу хода 5.7.

#### Добавить карту в ход (метод 4.1.6). Для авторизованных пользователей.

#### Подробнее – перенаправление на страницу 5.5.

## Страница с описанием карты.

### Доступна всем пользователям.

### Вывод информации о карте (метод 4.1.3).

#### Изображение карты.

#### Название карты.

#### Тип карты.

#### Сколько требует еды (от 0 до 2).

#### Описание карты.

## Страница со списком ходов.

### Доступна только авторизованным пользователям.

### Вывод информации о ходах (метод 4.2.1).

#### Фаза хода.

#### Дата создания.

#### Дата формирования.

#### Дата завершения.

#### Кем сформирована. Для администраторов.

#### Список добавленных карт.

#### Сколько взято еды.

### Действия

#### Открыть ход – перенаправление на страницу 5.7

#### Установить статус «завершен» или «отклонен» (метод 4.2.7). Для администраторов

#### Фильтр ходов по статусу и/или диапазону даты формирования (метод 4.2.1). Для авторизованных пользователей.

#### Страница с описанием карты.

##### Доступна только авторизованным пользователям.

##### Информация об ходе и списке карт в нем (метод 4.2.2).

###### Изображение карты.

###### Название карты.

###### Сколько взято еды.

##### Действия над черновиком

###### Сформировать ход (метод 4.2.7).

###### Удалить ход (4.2.4).

###### Удалить карту из хода (4.2.5).

###### Изменить фазу хода (4.2.3).

#### Страница изменения карт. Модератор может изменить или удалить карты.

##### Доступна только администраторам.

##### Информация о карте (метод 4.1.3).

###### Изображение карты.

###### Название карты.

###### Тип карты.

###### Сколько требует еды карта.

###### Описание карты.

##### Изменение полей карты (метод 4.1.4).

#### Страница со списком карт.

##### Доступна только администраторам.

##### Информация о картах (метод 4.2.2).

###### Изображение карты.

###### Название карты.

###### Тип карты.

###### Сколько требует еды карта.

###### Описание карты.

##### Действия

###### Удалить карту (метод 4.1.5).

###### Редактировать карту – перенаправление на страницу 5.8.

###### Добавить карту – перенаправление на страницу 5.8.

# **Требования к аппаратному обеспечению**

## Серверная часть

### Процессор минимум 2-ядерный от 2 Ггц

### Оперативная память от 8 Гб.

### Место на жестком диске от 2 Гб.

## Клиентская часть

### Процессор частотой от 1 ГГц.

### Оперативная память от 2 Гб.

# **Требования к программному обеспечению**

## Серверная часть

### ОС: Linux/Windows

### Docker 18.09.0 и выше

### СУБД: PostgreSQL 15.0

### GO 1.20.0 и выше

### minio RELEASE.2022-10-15T19-57-03Z

### nginx 1.19.2

### redis 7.2.2

### Node.js 20.10.0 и выше

## Клиентская часть

### OC: Windows, Linux, MacOs

### Браузер: Google Chrome 100+, Firefox 44+, Safari 12+, Yandex 20+