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: «Походы викингов»

Студент: Пермяков Д.К.

Группа ИУ5-53Б

Преподаватель: Канев А.И.

2023г.

ВВЕДЕНИЕ

Историческая область, посвященная походам викингов, стремительно развивается, привлекая внимание историков со всего мира. В силу увеличения интереса к этой теме возникает потребность в удобном инструменте для обмена информацией и создания исторических нарративов. С учетом того, что многие историки предпочитают участвовать в обсуждениях и делиться своими исследованиями, решено создать специализированный веб-ресурс.

Целью работы является реализация классификации и предложения тем для исследовательских проектов в области викингов, включающую в себя веб-сервис, веб-приложение, мобильное приложение и выделенный сервис проверки грамотности текста для поддержания высокого уровня качества исследовательских материалов и расчёта рейтинга пользователя.

Система предназначена для историков и модераторов сайта. На сайте предусмотрен ограниченный доступ к городам. Для получения доступа пользователю необходимо создать поход на получение доступа к выбранному городу. Система предоставляет автоматизированный способ создания, учета и ведения заявок. Также она позволяет модераторам принимать или отклонять заявки, 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 GoLang 1.20.
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 в React.

9. Создать мобильное приложение на SwiftUI.

10. Создать выделенный сервис проверки грамотности и расчёта рейтинга пользователя на Python.

11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

1. БИЗНЕС-ПРОЦЕСС

В бизнес-процессе участвуют люди. Они могут иметь роли исторического исследователя, исторического куратора, или не иметь роль вовсе – тогда они считаются гостями.

Города викингов были основаны в VIII-XI веках в ходе их морских экспансий [1]. К примеру, Бирка в Швеции и Хафнарфьордур в Исландии служили центрами торговли, где викинги обменивались товарами и культурными влияниями. Эти поселения также были оборудованы домами, верфями и ковчегами для хранения сокровищ, отражая разносторонний облик викингского общества. Города викингов были не только центрами торговли, но и важными точками для социального и культурного обмена в эпоху викингов. Система хранит список городов. Города имеют название, описание и фотографию. Викинги славились своими дерзкими исследованиями и морскими походами в VIII-XI веках. Они осуществляли экспедиции по Северной Европе, Атлантическому океану и даже Средиземному морю, завоевывая и торгуя. Их драккары, легкие и быстрые корабли, позволяли викингам преодолевать огромные расстояния, внушая страх и воздвигая свой мифический облик в глазах современников. Для того, чтобы сформировать статью о походе викингов, пользователи выбирают города, которые участвуют в походах, добавляя их в черновую заявку. В одном походе может быть сразу несколько городов. Города можно добавлять и удалять из похода. Черновую заявку также можно удалить в любой момент.

Когда пользователь выберет все желаемые города, он формирует поход, и он отправляется на модерирование. После этого его уже нельзя модерировать. Можно также посмотреть список предыдущих походов. Сформированный поход затем рассматриваются историческим куратором. Он принимает решение об одобрении и отклонении похода. После вынесения вердикта историческим куратором у пользователя обновляется статус, сформированного похода на «принято» или «отклонено», в зависимости от

того, что решил проверяющий исторический куратор. Написанные статьи требуют проверку на грамотность содержимого. На сколько текст грамотен – решает внешний сервис literacy-calculator [2], который в зависимости от уровня грамотности текста выставляет оценку статьи.

Более подробно бизнес-процесс описан на диаграммах прецедентов (рис. 1), деятельности и состояний:

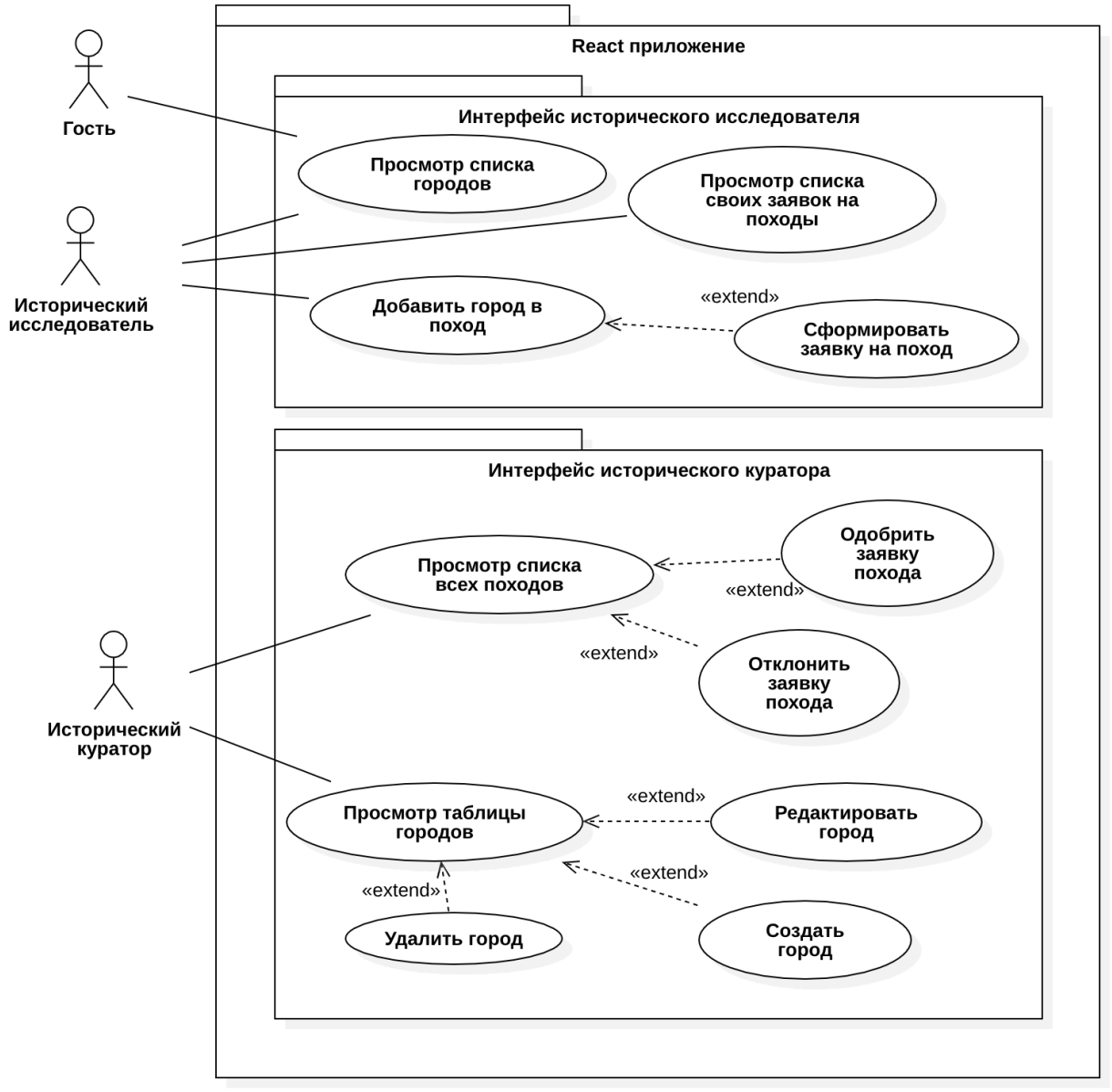

Рисунок 1 - Диаграмма прецедентов

Гостям доступен просмотр городов. Зарегистрированные гости – исторические исследователи. Они могут добавлять города в поход, просматривать список своих походов и сформировывать текущий поход.

Походы обрабатываются историческим куратором. В результате обработки похода её либо одобряют, либо отклоняют. Помимо возможностей принятие/отказа, историческому куратору также доступны уникальные функции для работы с городами, а именно: просмотр всех городов, редактирование, создание и удаление городов, а также просмотр списка всех городов в табличном виде. Процесс оформления похода отражен на диаграмме деятельности (рис. 2).

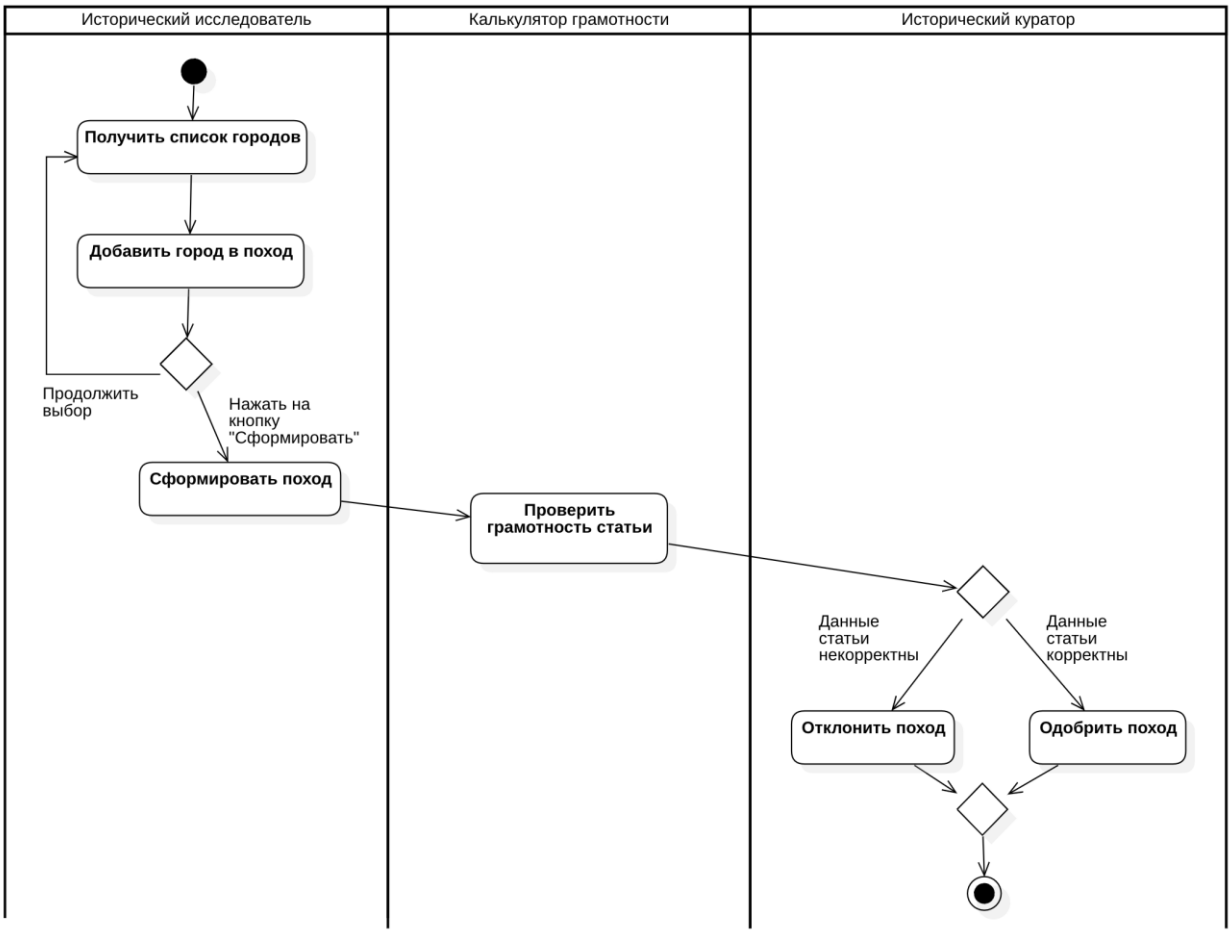

Рисунок 2 - Диаграмма деятельности

Исторический исследователь выбирает города, затем формирует на основе выбранных городов поход. Этот поход затем обрабатывает асинхронный сервис, а затем и исторический куратор. В случае одобрения похода запускается процесс проверки грамотности содержимого статьи в асинхронном сервисе, вычисляющий новый рейтинг исторического

исследователя. Возможные состояния похода отражены на диаграмме состояний (рис. 3).

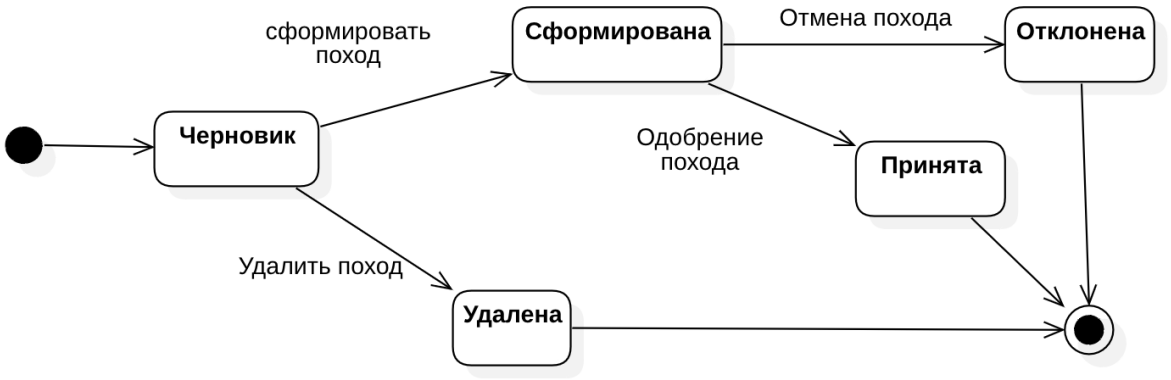

Рисунок 3 - Диаграмма состояний похода

При выборе первого города формируется черновик. Последующие выбранные города добавляются в этот черновик. Пользователь затем формирует поход, удаляет её или выходит из приложения. Сформированный поход обрабатывает исторический куратор. Он может одобрить или отклонить её.

2. 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 [5]: 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 literal-calculator [2] разворачивается отдельно и необязательно на том же компьютере, что и основной сервис.

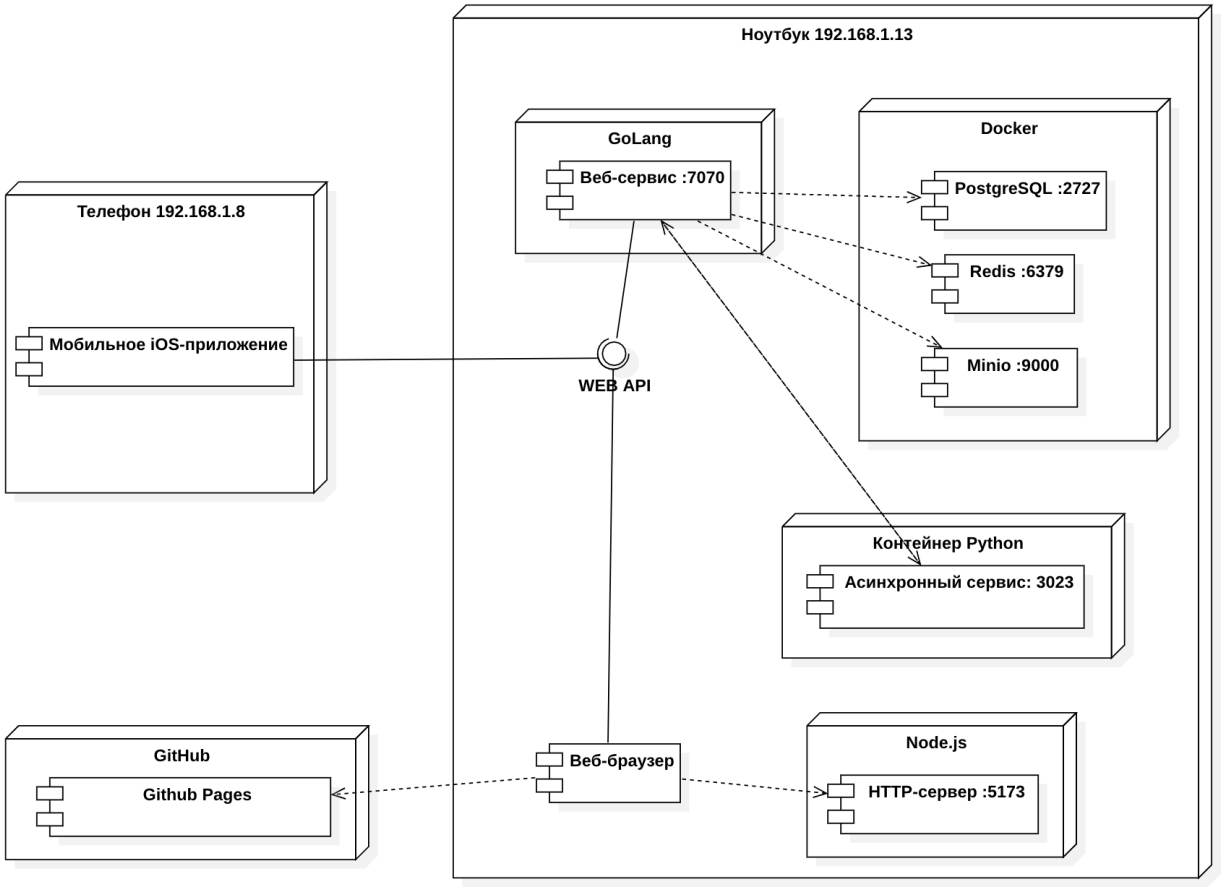

Рисунок 4 - Диаграмма развертывания

Нативное [3] и браузерное приложения [6, 7] обращаются к веб-сервису, основанному на технологии Gin v1.9.1 [4], 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 [8], 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 профессия, отражающее его профессию и поле рейтинг, отражающий текущий рейтинг пользователя за все его написанные статьи. Модель города представляет собой набор полей, необходимых исключительно для бизнес-логики. Для хранения в одной заявке нескольких карт используется промежуточная таблица места_назначения_походов, реализующая связь М-М. Устройство бекенда приложения изображено на диаграмме классов бекенда (рис 6.).

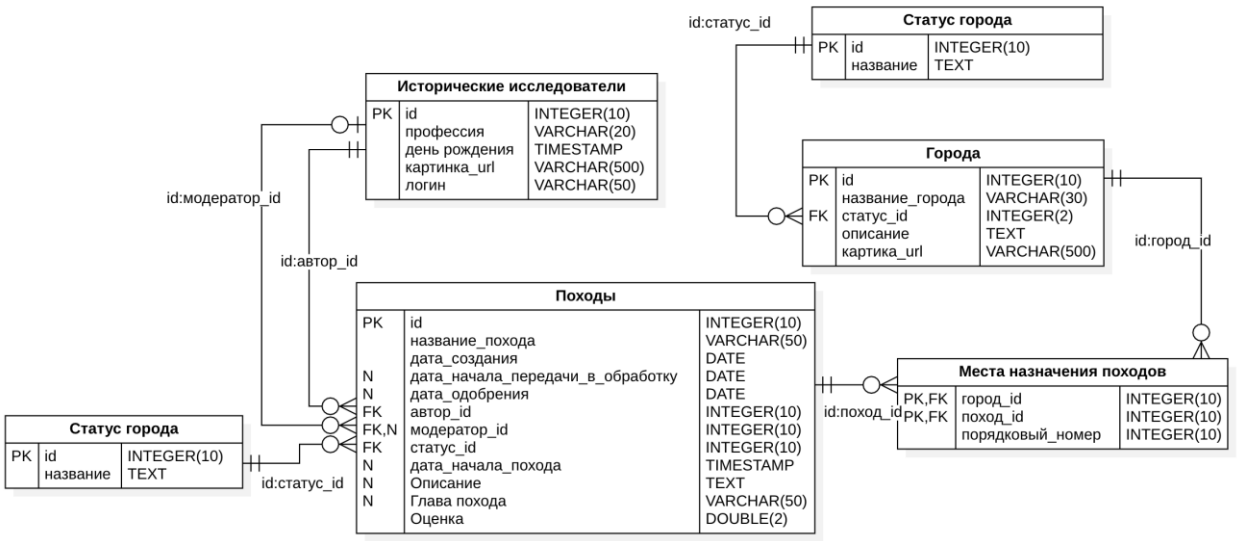

Рисунок 5 - ER диаграмма

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, города

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

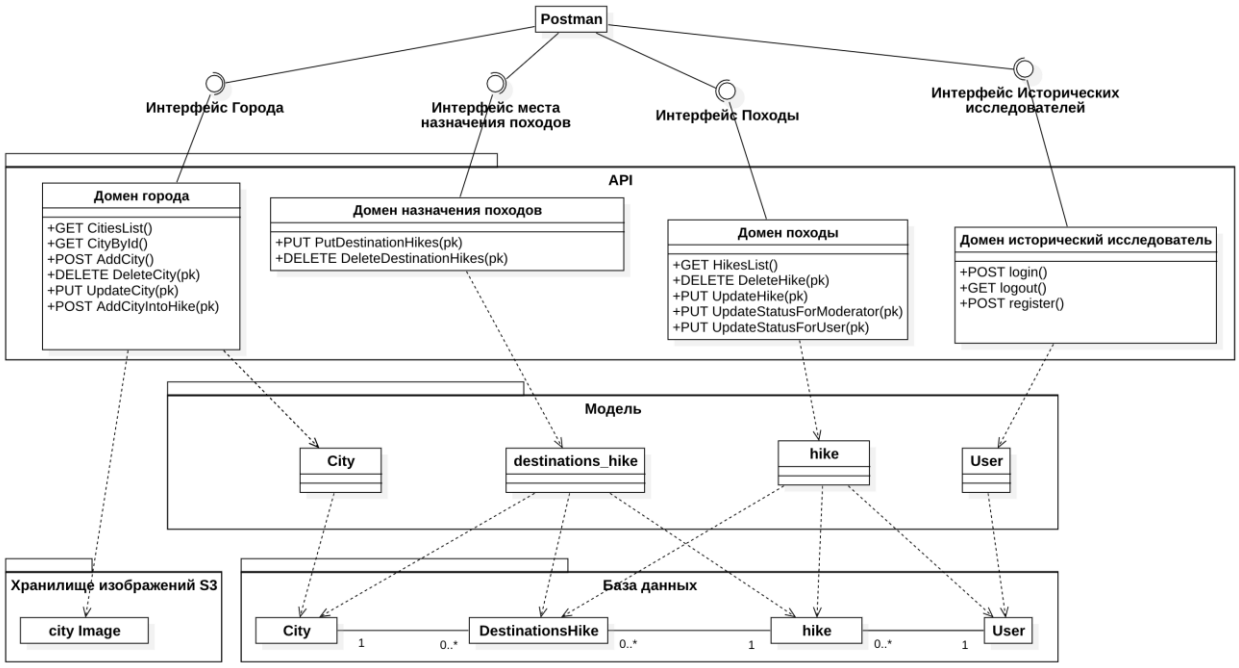

Рисунок 6 - Диаграмма классов бекенда

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 (исторических исследователей) с определенными правами (ролями).

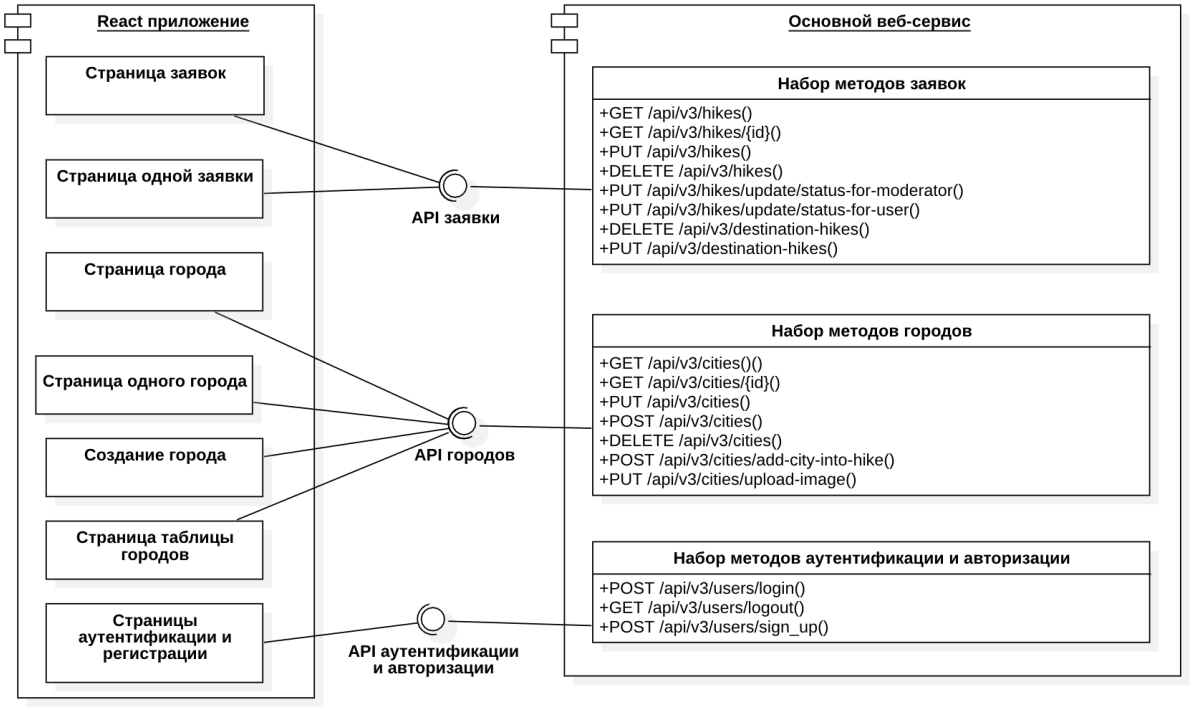

Рисунок 7 - Диаграмма классов фронтенда

3. 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: города, походы и исторические исследователи. Методы следуют правилам REST API.

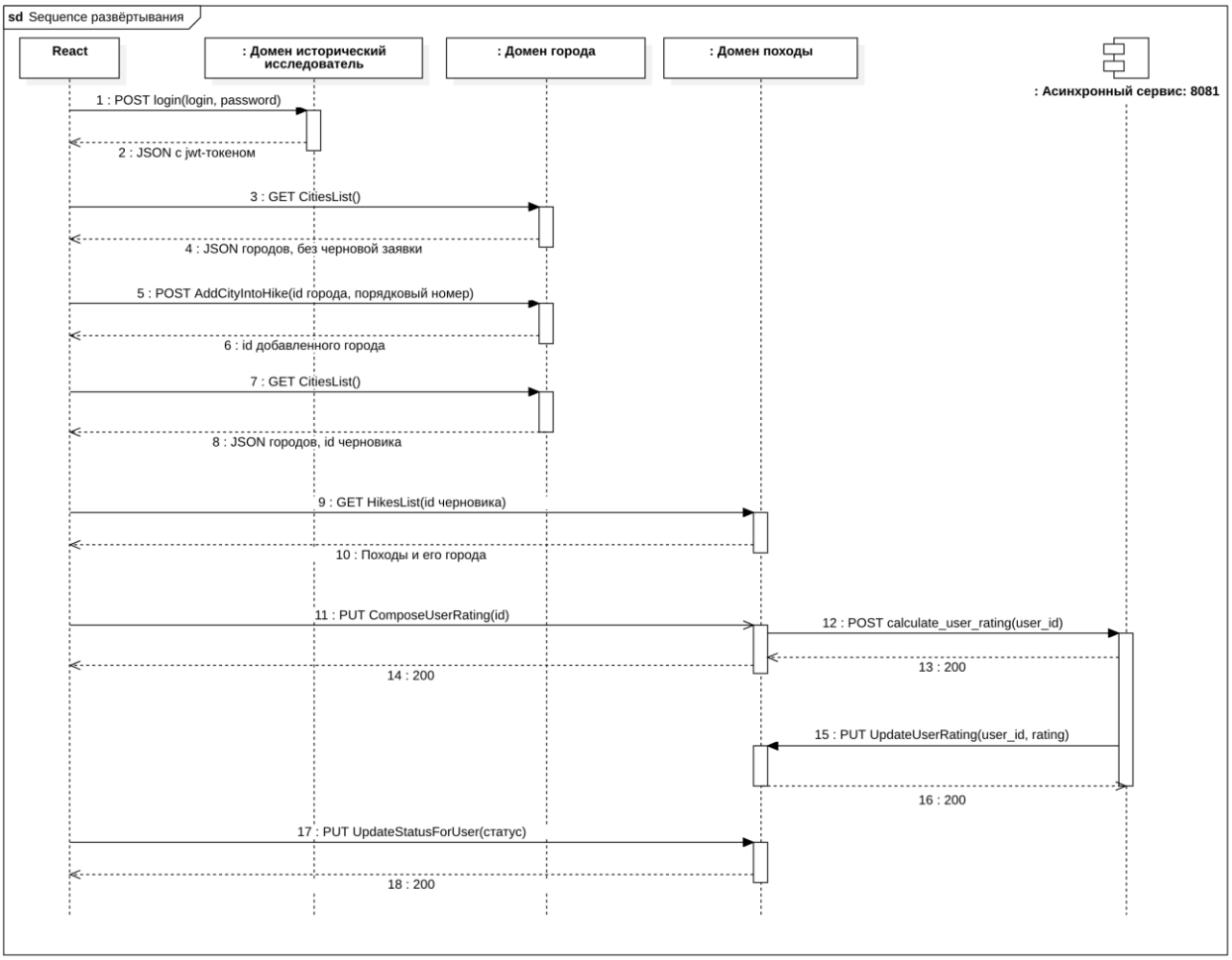

Рисунок 8 - Диаграмма последовательности

В начале бизнес-процесса происходит аутентификация исторического исследователя. 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 городов, которые возвращаются в JSON формате. Пользователь выбирает город, которую хочет добавить, и, нажимая на кнопку «плюс» в графическом интерфейсе, отправляет запрос на добавление города 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 этого похода. В этот момент основной веб-сервис выполняет асинхронный запрос к сервису literal-calculator на то, чтобы он проверил грамотность статьи. Когда поход и его города будут одобрены, пользователь сможет увидеть через некоторое время, что оценка его статьи изменилась.

Процесс рассмотрения походов происходит также через графический интерфейс. Исторический куратор может просматривать списки всех походов и, нажимая на соответствующие кнопки, отправлять запросы на одобрение или отклонение похода в основной веб-сервис. В эти запросы также можно включить фильтры по имени создателя похода и диапазону дат, в котором должны были быть созданы походы. Также через графический интерфейс они могут управлять непосредственно городами. Им доступны такие функции, как создание и редактирование города, просмотр списка городов и удаление их. Для каждой из этих функция присутствует свой метод, отправляемый на основной веб-сервис.

4. ОПИСАНИЕ ИНТЕРФЕЙСА

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

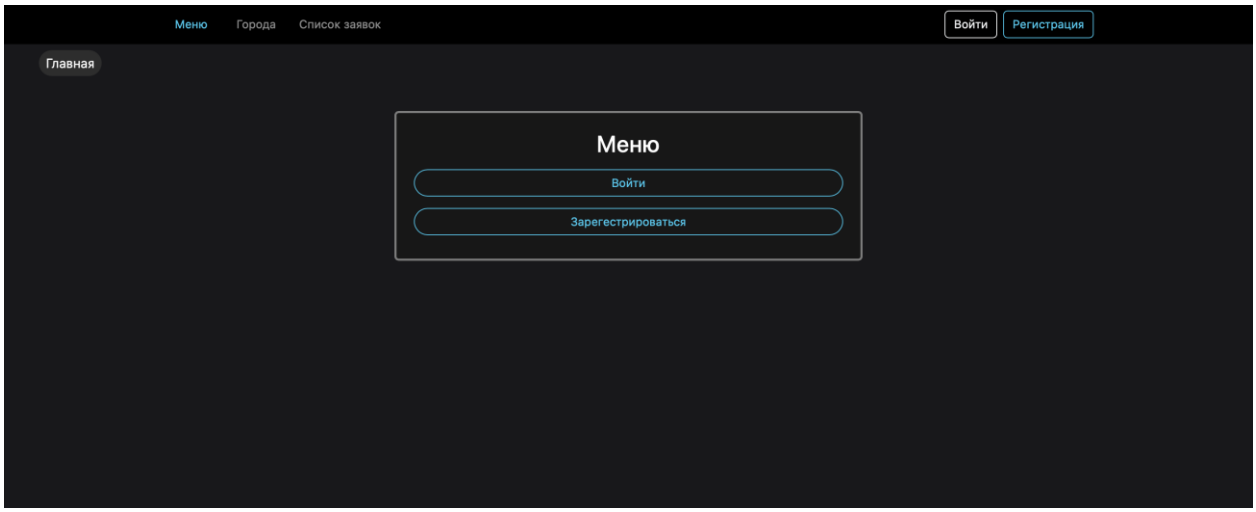

Рисунок 9 - Главное меню (неавторизированный пользователь)

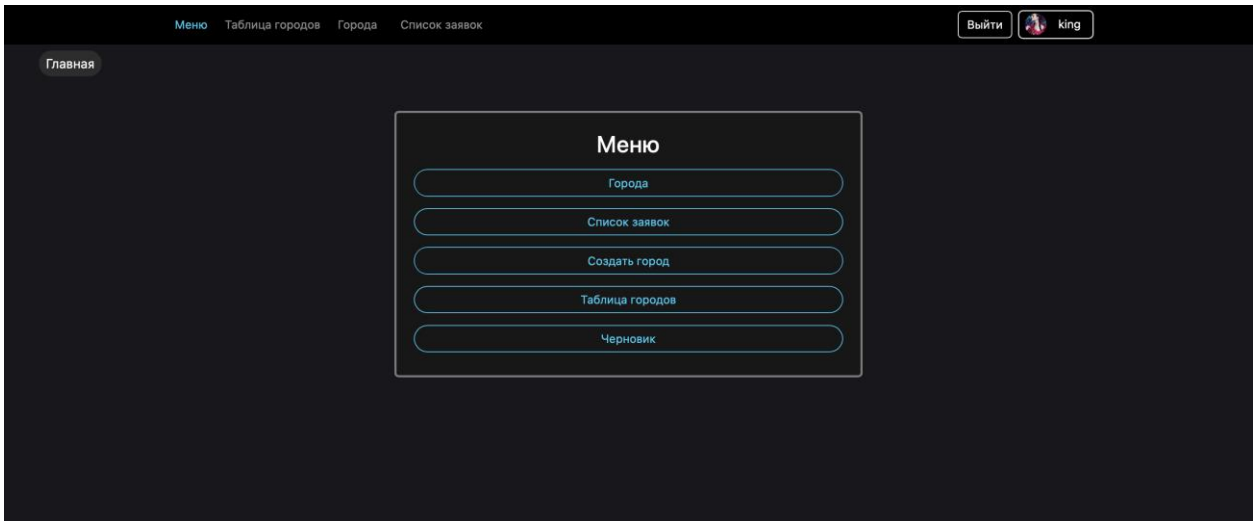

Рисунок 10 - Главное меню (исторический куратор)

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 городов», «черновик» и «список заявок».

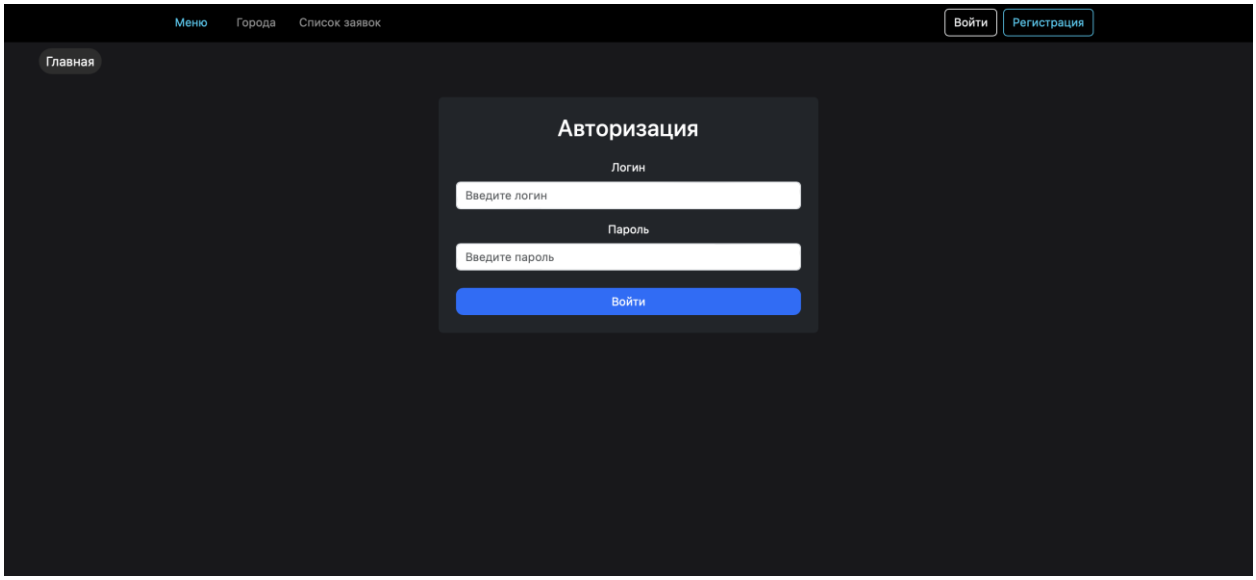

Рисунок 11 - Страница авторизации

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.

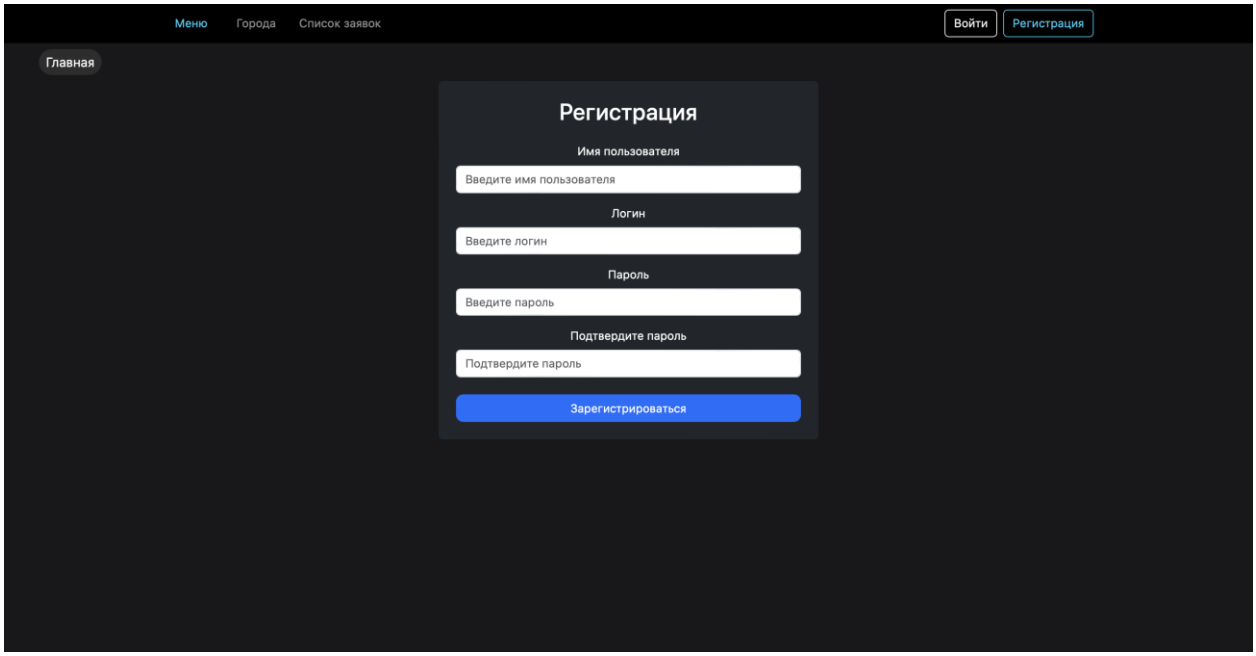

Рисунок 12 - Страница регистрации

На этой странице гости могут завести аккаунт. Для этого нужно указать имя, логин и пароль, и повторить пароль. Если введенный логин уже занят система попросит пользователя сменить его.

Рисунок 13 - Страница со списком городов

На этой странице отображается список городов в виде карточек. У каждой города есть хендлер нажатия для перехода в режим «Подробнее», переносящая пользователя на страницу с подробной информацией о городе.

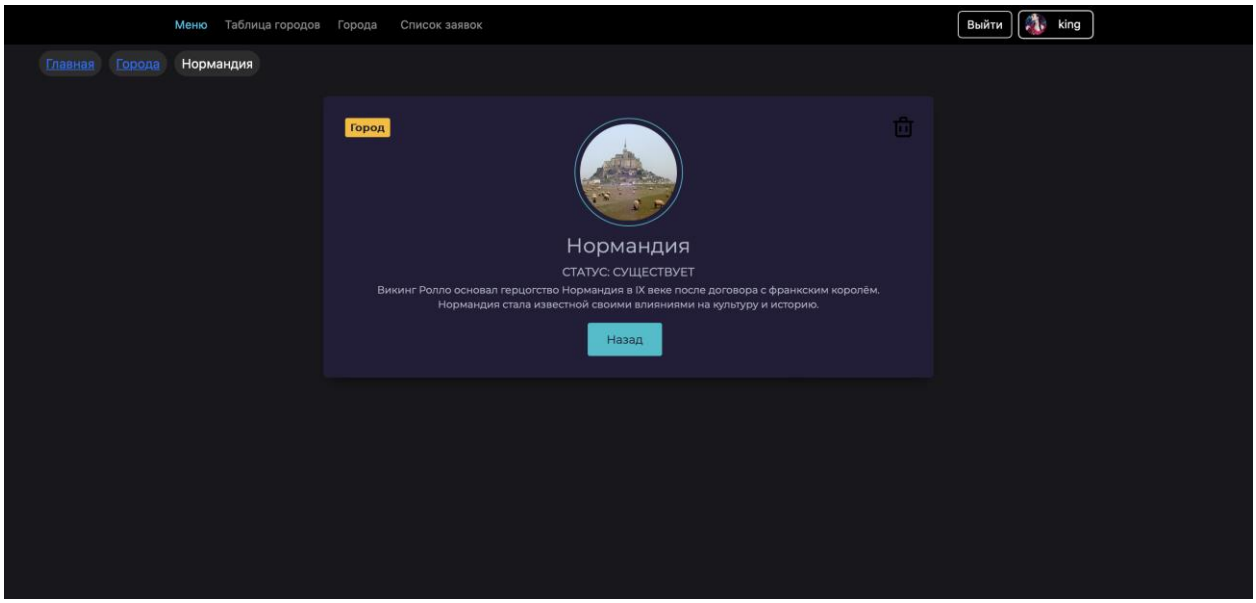

Рисунок 14 - Страница с подробным описанием города

На этой странице отображается подробная информация о городе: название, статус и текстовое описание. Историческому куратору есть доступ удаления города при нажатии на кнопку корзины, также можно вернуться назад для всех пользователей.

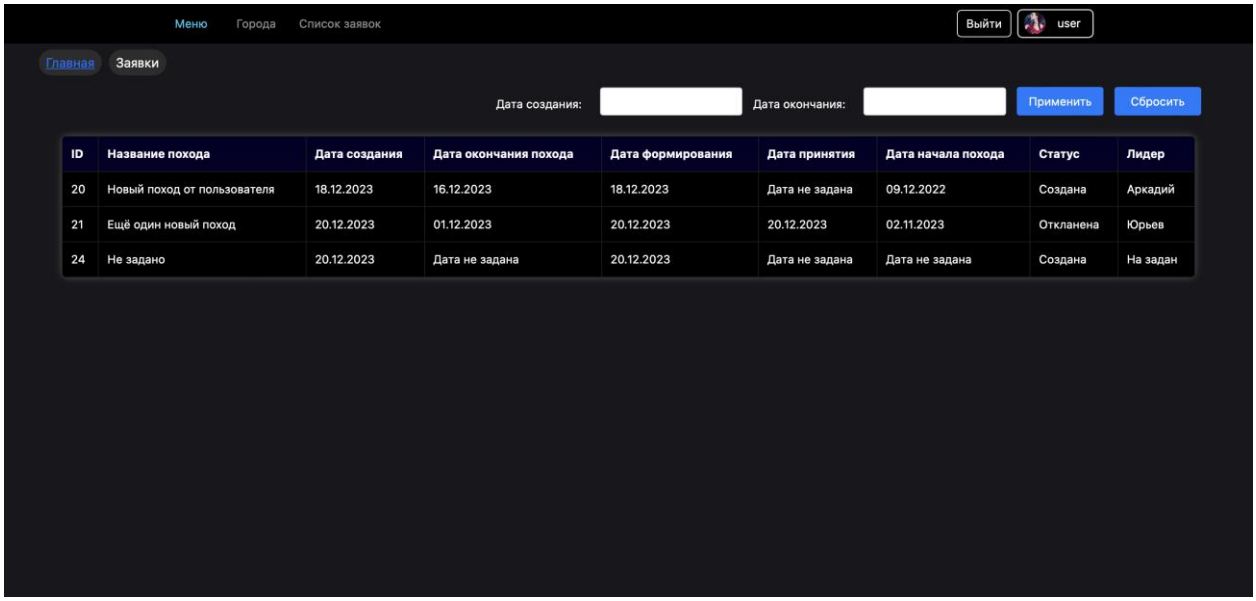

Рисунок 15 - Страница со списком заявок (исторический исследователь)

На этой странице отображается список заявок. В зависимости от типа пользователя этот список будет функционально отличаться. Так, для пользователей отображается список созданных ими заявок: ID, название похода, дата создания, дата окончания похода, дата формирования, дата принятия, дата начала похода, статус и лидер похода.

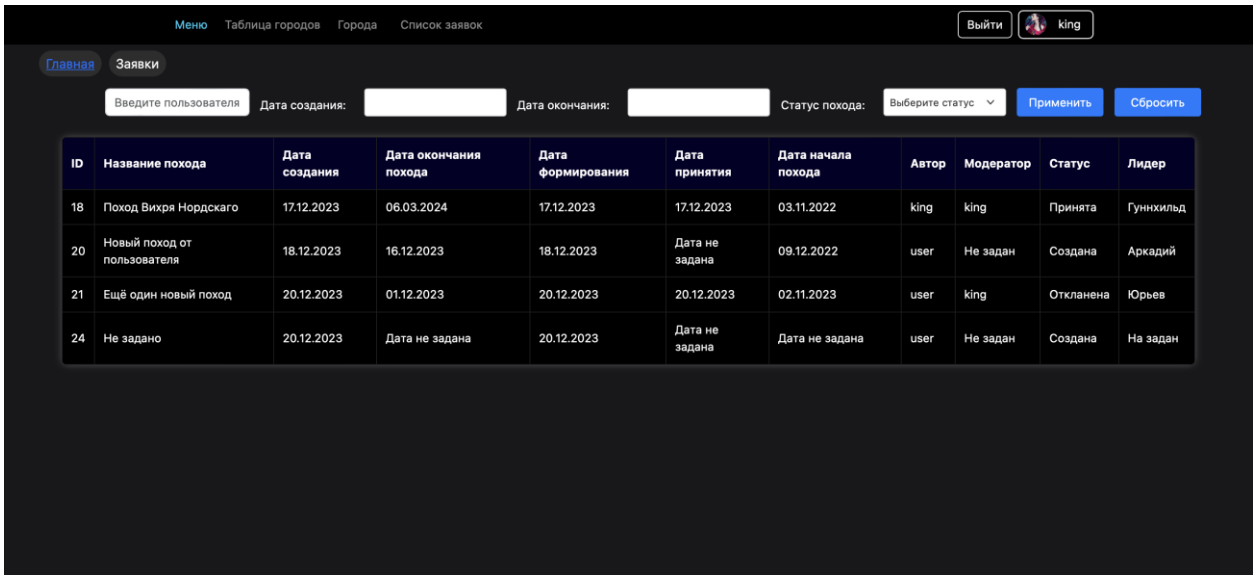

Рисунок 16 - Страница со списком заявок (исторический куратор)

Для исторического куратора функционал этой страницы шире. Для них отображается список всех заявок всех пользователей с более подробной информацией: ID, автор, дата создания, дата формирования, дата завершения, дата начала похода, дата конца похода, дата обработки, кто её обработал,

лидер похода и статус похода. Кроме того, присутствует фильтрация по пользователю.

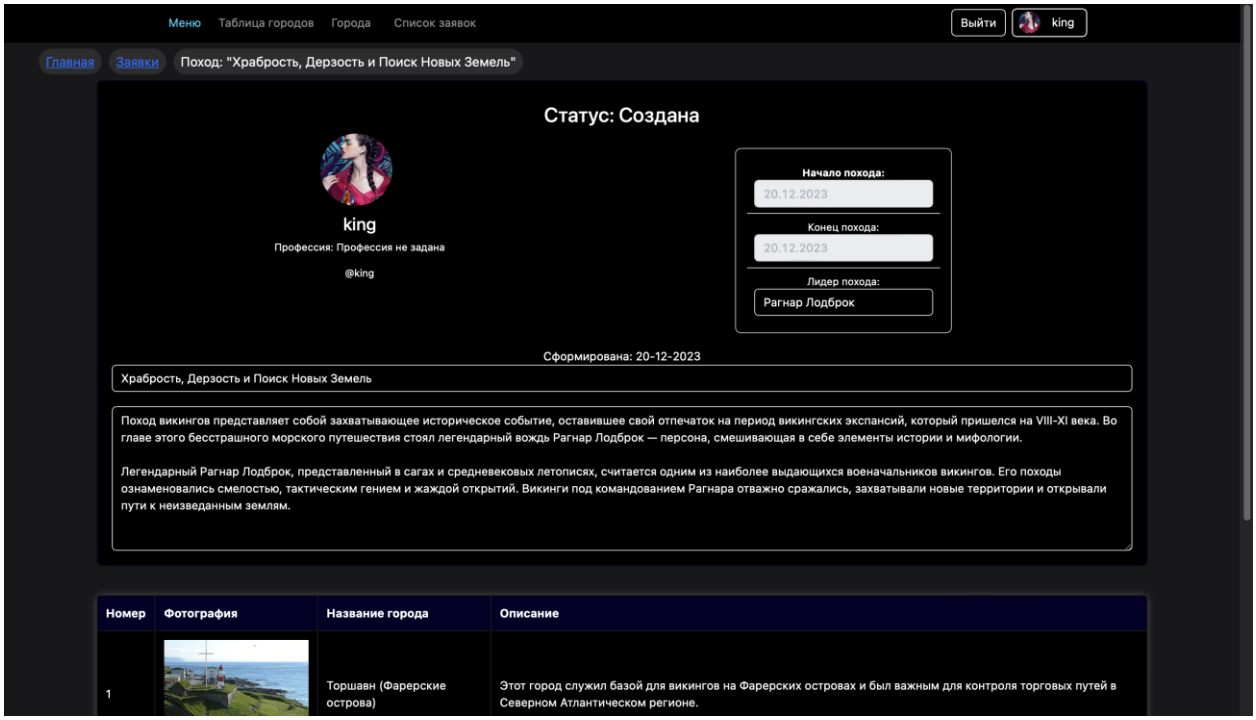

Рисунок 17 - Страница с подробным описанием похода

На этой странице отображается подробная информация о заявке. Список выбранных городов в виде таблицы, а также статус похода, дата формирования и её автор.

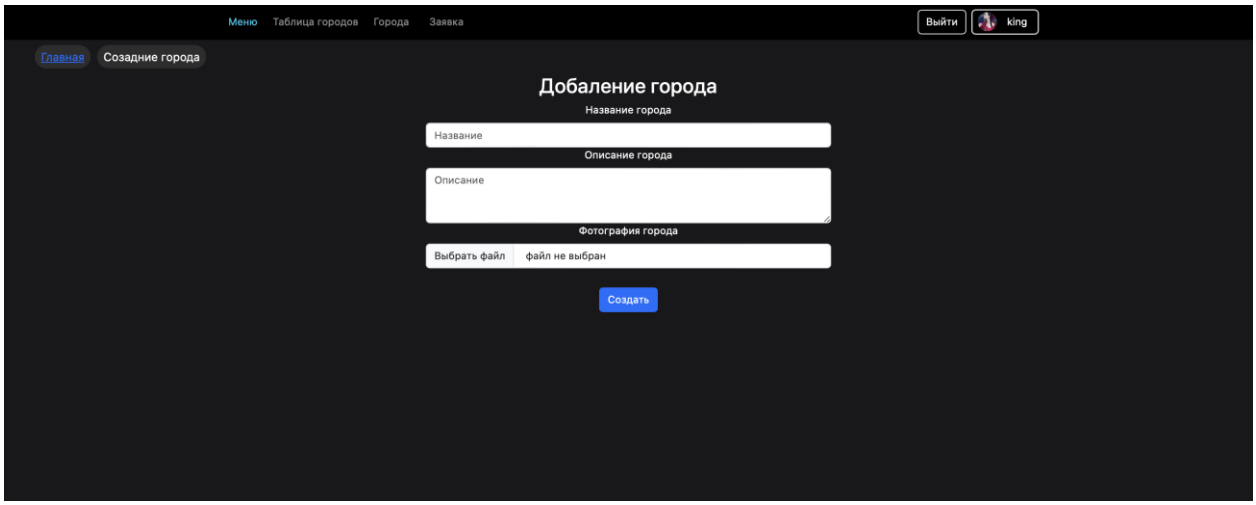

Рисунок 18 - Страница создания города

На этой странице модератор может создать новый город. Для создания доступны все поля: название, описание и фотография города.

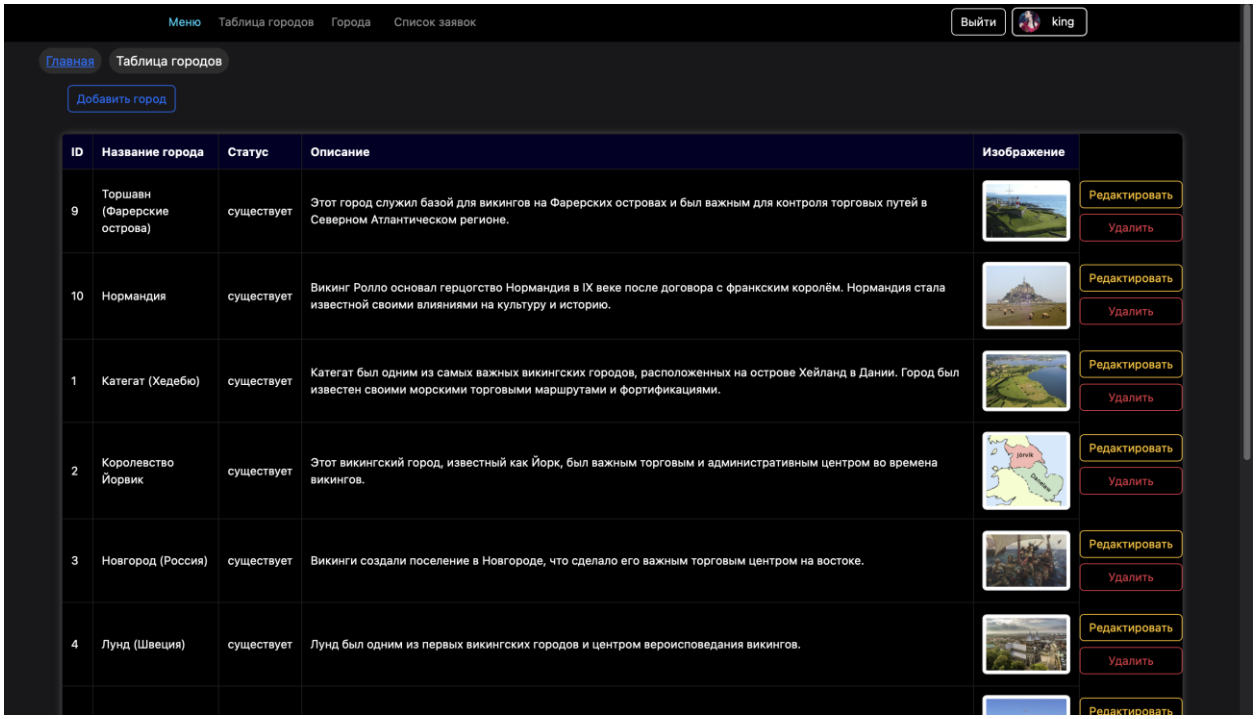

Рисунок 19 - Страница с таблицей городов

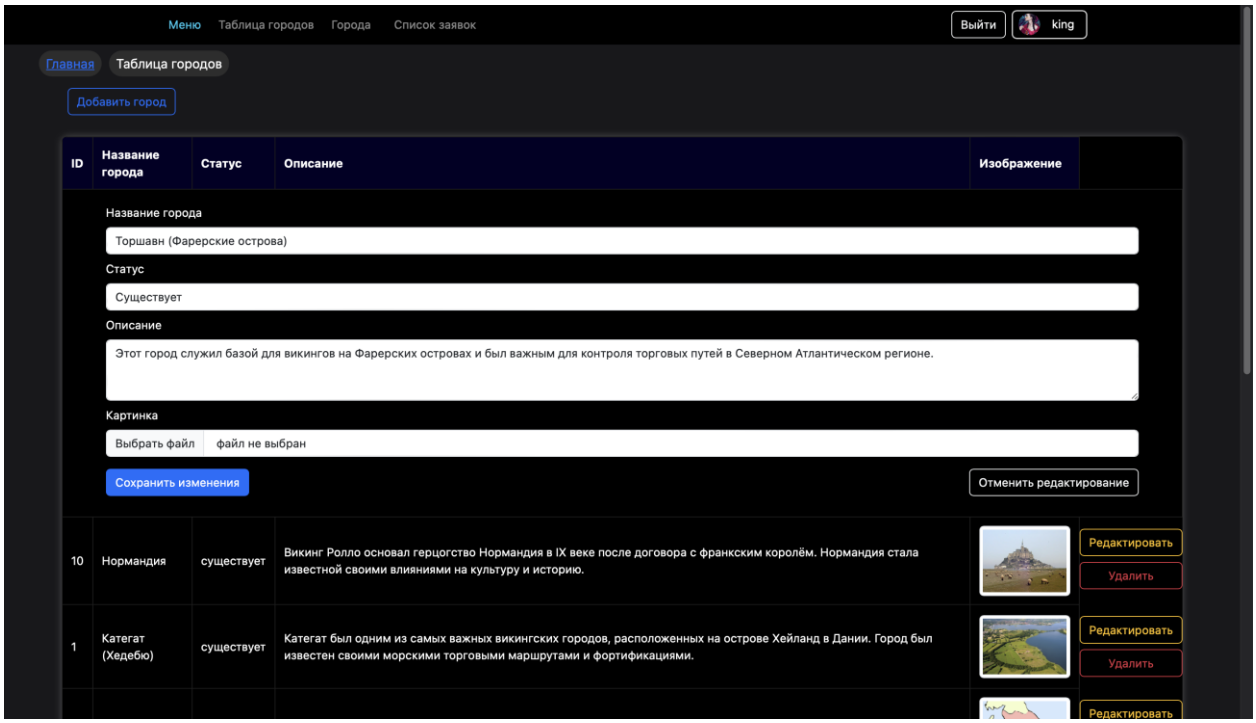

Рисунок 20 - Страница с таблицей городов в режиме редактирования города

На этой странице исторический куратор может в компактном и удобном формате просмотреть список всех городов, существующих в системе и отредактировать информацию о городе. Отображаются следующие поля: название, изображение-обложка, описание и статус города.

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://mightyk1ngrichard.github.io/DevelopmentNetworkApplicationFrontend/.
6. В веб-сервис добавлена авторизация через JWT, а методы задокументированы через Swagger.
7. Реализован интерфейс пользователя был реализован. Доступ к нему имеют только авторизированные пользователи.
8. Выделенный сервис был разработан и развернут в отдельном виртуальном окружении Python.
9. Реализован интерфейс модератора городов для создания новых городов, редактирования существующих.
10. Было реализовано мобильное приложение на SwiftUI, 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/mightyK1ngRichard/DevelopmentNetworkApplicationBackend

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Экспансия викингов (viii-xi вв.). Роль норманнов в создании единого европейского экономического пространства Роль викингов в европейской истории [Электронный ресурс] // Habr, URL: https://schoolperspektiva.ru/tolstojj-l-n-/ekspansiya-vikingov-viii-xi-vv-rol-normannov-v-sozdanii-edinogo-evropeiskogo/ (дата обращения 5.10.2023)
2. Data Structures and Algorithms for Game Developers [Книга] // Allen Sherrod. (дата обращения 04.10.2023).
3. Документация Apple [Электронный ресурс] // Documantation. URL: https://developer.apple.com/xcode/swiftui/ (дата обращения 02.10.2023).
4. Руководство по Gin Web Framework [Электронный ресурс] // GoLang. URL: https://github.com/gin-gonic/gin (дата обращения: 03.08.2023).
5. Полное практическое руководство по Docker [Электронный ресурс] // Habr. URL: https://habr.com/ru/articles/310460/ (дата обращения: 20.10.2023).
6. Руководство по React [Электронный ресурс] // Metanit. URL: https://metanit.com/web/react/ (дата обращения: 10.10.2023).
7. Quick Start – React [Электронный ресурс] // React. URL: https://react.dev/learn (дата обращения: 12.10.2023).
8. Руководство по PostgreSQL [Электронный ресурс] // Metanit. URL: https://metanit.com/sql/postgresql/ (дата обращения: 03.08.2023).

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: «Походы викингов»

Студент: Пермяков Д.К.

Группа ИУ5-53Б

Преподаватель: Канев А.И.

2023г.

1. Цель работы

Реализовать систему для организации походов викингов, включающей в себя веб-сервис, веб-приложение, мобильное приложение и выделенный сервис для проверки орфографии в статьях и обеспечения качества текстовых материалов в системе.

2. Назначение

Система предназначена для историков и модераторов сайта. На сайте предусмотрен ограниченный доступ к городам викингов. Для получения доступа пользователю необходимо создать поход на получение доступа к выбранному городу. Система предоставляет автоматизированный способ создания, учета и ведения заявок. Также она позволяет модераторам принимать или отклонять поход. Модераторы имеют возможность редактировать существующие и создавать новые походы.

3. Задачи:

3.1. Разработать дизайн приложения.

3.2. Создать базу данных в PostgreSQL.

3.3. Создать веб-сервис на технологии GoLang.

3.4. Реализовать интерфейс гостя на технологии React.

3.5. Развернуть веб-приложение React на Github Pages.

3.6. Добавить авторизацию и аутентификацию в веб-сервис.

3.7. Реализовать интерфейс пользователя в React.

3.8. Реализовать интерфейс модератора в React.

3.9. Создать мобильное приложение на SwiftUI.

3.10. Создать выделенный сервис для проверки орфографии в статьях.

3.11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

4. Методы веб-сервиса:

№	Метод	Описание	Url	Входные данные	Выходные данные
4.1. Методы авторизации и аутентификации					
4.1.1.	POST	Регистрация	/api/v3/users/sign_up	login: string password: string user_name: string	

4.1.2.	POST	Вход в аккаунт	/api/v3/users /login	login: string
				password: string
					{
					access_token: string,
					expires_type: string
					}

4.1.3.	GET	Выход из аккаунта Доступно только авторизированным пользователям	/api/v3/users/logout	auth cookies	status_code: int
4.2. Методы городов					

4.2.1.	PUT	Обновляет фотографию города.
		Доступно только модератору
			api/v3/cities/upload-image	auth header
				file: File
				city_id: Int
					{
					image_url: string,
					status: string
					}

4.2.2.	GET	Возвращает список городов, удовлетворяющих	api/v3/cities	auth header
				{
				city: string,
					{
					cities: **{**
					"id": int,
					city_name: string,
					status_id: int,
					status: {
					id: int,

		переданным критериям		search: string }	status_name: string }, Description: string image_url: string }[] }
4.2.3.	PUT	Добавляет в поход пользователя город {city_id}, возвращает id похода, в которую был добавлен город Доступно только авторизированным пользователям	api/v3/cities/add-city-into-hike	auth header { city_id: Int, serial_number: Int }	{ id: int, status: string }
4.2.4.	GET	Возвращает город	api/v3/cities/{id}	id: int	{ "id": int, city_name: string, status_id: int, status: { id: int, status_name: string }, Description: string image_url: string }
4.2.5.	POST	Создает город Доступно только модераторам	api/v3/cities	auth header { city_name: string, description: string,	{ city_id: string, status: string }

				image_url: file? }	

4.2.
6.
	PUT	Обновляет информацию о городе
		Доступно только модераторам
			api/v3/cities	auth
				header
				{
				city_name: string?,
				description: string?,
				id: int?,
				status_id: int?
				}
					{
					city_name: string,
					description: string,
					id: int,
					status_id: int
					}

4.2.
7.
	DELETE	Помечает город как удаленный
		Доступно только модераторам
			api/v3/cities	auth
				header
				id: int
					{
					deleted_id: int
					}

4.3. Методы походов					

4.3.
1.
	GET	Возвращает список заявок пользователя
		Доступно только авторизированным пользователям
			api/v3/hikes	auth
				header
					hikes: {
					date_approve: string,
					date_created: string,
					date_end: string,
					date_start_hike: string,
					date_start_of_processin: string,
					description: string,
					destination_hikes: {
					{
					city: {
					city_name: string,
					description: string,
					id: 0,
					image_url: string,
					status: {
					id: int,

					status_name: string }, status_id: int }, city_id: int, hike: string, hike_id: int, id: int, serial_number: int }[] }[]

4.3.2.	GET	Возвращает подробную информацию о заявке
		Доступно только авторизированным пользователям
			api/v3/hikes/{id}	auth
				cookies
				id: int
					{
					hikes: {
					date_approve: string,
					date_created: string,
					date_end:string,
					date_start_hike: string,
					date_start_of_processing: string,
					description:string,
					destination_hikes: City[],
					}[],
					status: string
					}

4.3.3.	GET	Возвращает текущую черновую заявку пользователя
		Доступно только авторизированным пользователям
			api/v3/hikes/current	auth
				cookies
					hikes: {
					date_approve: string,
					date_created: string,
					date_end: string,
					date_start_hike: string,
					date_start_of_processing: string,
					description: string,
					destination_hikes: City[]
					}

4.3.4.	DELETE	Удаляет из похода пользователя город {city_id} Доступно только авторизированным пользователям	api/v3/destination-hikes	auth header { Id: int }	{ deleted_destination_hike: int status": string" }
4.3.5.	PUT	Устанавливает заявке статус «сформирована» Только если её текущий статус «черновик» Доступно только авторизированным пользователям	Api/v3/hikes/update/status-for-user	auth header status_id: int?	Status_code: int
4.3.6.	DELETE	Устанавливает заявке статус «удалена» Только если её текущий статус «черновик» Доступно только авторизированным пользователям	api/v3/hikes	auth cookies { id: int }	{ Hike_id: int, Status: string }
4.3.7.	GET	Возвращает отфильтрованный по датам	Api/v3/hikes?status_id&start_date&end_date	auth cookies start_date: datetime?	{ Hikes:Hikes[], Status:string }

		создания и статусу похода. Доступно только модераторам		end_date: datetime? Status_id: int	
4.3.8.	PUT	Устанавливает заявке статус «отклонена» или «завершена» Только если её текущий статус «сформирована» Доступно только модераторам	api/v3/hikes/update/status-for-moderator	{ Hike_id: int, Status_id: int }	Status code
4.4. Методы внешнего сервиса literacy-calculator					
4.4.1.	POST	Проверяет качество написанного текста. Метод внешнего сервиса literacy-calculator	api/check_literacy	hike_id: int	

5. Функциональные требования:

5.1. Главное меню. На этой странице находится меню со списком доступных пунктов.

5.1.1. Доступна всем пользователям.

5.1.2. Действия

5.1.2.1. Войти – переход на страницу 5.2. Только для гостей.

5.1.2.2. Зарегистрироваться – переход на страницу 5.3. Только для гостей.

5.1.2.3. Открыть список городов – переход на страницу 5.4. Только для авторизированных пользователей.

5.1.2.4. Открыть список выбранных городов – переход на страницу 5.7. Только для авторизированных пользователей.

5.1.2.5. Список заявок – переход на страницу 5.6. Только для авторизированных пользователей.

5.1.2.6. Создать город – переход на страницу 5.8. Только для администраторов.

5.1.2.7. Таблица городов – переход на страницу 5.9. Только для модератора.

5.2. Страница с формой авторизации. На этой странице находится форма авторизации.

5.2.1. Доступна гостям

5.2.2. Действия

5.2.2.1. Войти – производит запрос (метод 4.1.2.).

5.2.2.2. Зарегистрироваться – перенаправляет на страницу 5.3.

5.3. Страница с формой регистрации. На этой странице находится форма авторизации.

5.3.1. Доступна гостям

5.3.2. Действия

5.3.2.1. Зарегистрироваться – производит запрос (метод 4.1.1.), в котором передаются введенные данные формы.

5.3.2.2. Войти - перенаправляет пользователя на страницу 5.2.

5.4. Страница со списком городов. На этой странице располагается список всех городов и панель фильтрации городов.

5.4.1. Доступна всем пользователям.

5.4.2. Выводится информация о городах в виде карточек (метод 4.2.2.).

5.4.2.1. Изображение города.

5.4.2.2. Название города.

5.4.3. Действия

5.4.3.1. Отфильтровать города по введённому названию города (метод 4.2.2.).

5.4.3.2. Добавить город в поход (метод 4.2.3.). Только для авторизированных пользователей.

5.4.3.3. Нажатие на карточку – перенаправляет на страницу 5.5.

5.5. Страница с подробным описанием города

5.5.1. Доступна всем пользователям.

5.5.2. Выводится информация о городе в построчном формате.

5.5.2.1. Изображение города.

5.5.2.2. Название города.

5.5.2.3. Статус города

5.5.2.4. Описание города.

5.5.3. Действия

5.5.3.1. Удалить город (метод 4.2.7.). Только для модератора.

5.5.3.2. Назад – перенаправляет на предыдущую страницу.

5.6. Страница со списком заявок

5.6.1. Доступна только авторизированным пользователям.

5.6.2. Выводится информация о походах в табличном формате (методы 4.3.1. и 4.3.8.).

5.6.2.1. Название похода.

5.6.2.2. Дата создания.

5.6.2.3. Дата окончания похода.

5.6.2.4. Дата начала процесса.

5.6.2.5. Дата принятия.

5.6.2.6. Дата начала похода

5.6.2.7. Автор статьи о походе.

5.6.2.8. Кем обработана. Только для модераторов.

5.6.3. Действия

5.6.3.1. Открыть поход – перенаправляет на страницу 5.6.

5.6.3.2. Отфильтровать список заявок по диапазону дат и/или статусу похода (метод 4.3.8.). Только для модераторов.

5.7. Страница с подробным описанием похода. Пользователь может удалить город из похода, отредактировать содержимое статьи о походе. Модератор может просматривать заявки всех пользователей.

5.7.1. Доступна только авторизированным пользователям.

5.7.2. Выводится информация о заявке. Список городов, выбранных в заявке, в виде таблицы (метод 4.3.2.).

5.7.2.1. Изображение автора статьи.

5.7.2.2. Имя автора статьи.

5.7.2.3. Профессия автора статьи.

5.7.2.4. Логин автора статьи.

5.7.2.5. Дата начала похода.

5.7.2.6. Дата окончания похода.

5.7.2.7. Название похода.

5.7.2.8. Лидер похода.

5.7.2.9. Описание похода.

5.7.2.10. Посещённые города.

5.7.2.11. Статус похода.

5.7.2.12. Дата формирования похода.

5.7.3. Действия

5.7.3.1. Удалить город из похода (метод 4.3.4.). Если заявка похода находится в статусе черновика. Только для владельца заявки.

5.7.3.2. Сформировать поход (метод 4.3.5.). Если заявка похода находится в статусе черновика. Только для владельца заявки.

5.7.3.3. Удалить поход (метод 4.3.6.). Если заявка похода находится в статусе черновика. Только для владельца заявки и модератора.

5.7.3.4. Принять поход (метода 4.3.8). Если заявка сформирована. Только для модератора.

5.7.3.5. Отклонить поход (метод 4.3.8). Если заявка сформирована. Только для модератора.

5.8. Страница создания города.

5.8.1. Доступна только модераторам.

5.8.1.1. Название города.

5.8.1.2. Описание города.

5.8.1.3. Фотография города.

5.8.2. Действия

5.8.2.1. Создать город (метод 4.2.5)

5.9. Страница с таблицей городов. Предоставляет модератору удобный способ отображения всех городов в табличной форме с возможностью редактирования.

5.9.1. Доступна только модераторам.

5.9.1.1. Название города.

5.9.1.2. Статус города.

5.9.1.3. Описание города.

5.9.1.4. Фотография города.

5.9.2. Действия

5.9.2.1. Изменение полей города. Можно изменять все вышеперечисленные поля города.

5.9.2.2. Сохранить изменения (методы 4.2.5. и 4.2.6.).

5.9.2.3. Удалить город (метод 4.2.7)

5.9.2.4. Добавить город. Открывает страницу 5.8.

6. Требования к программному обеспечению:

6.1. Серверная часть

6.1.1. ОС: MacOS Sonoma 14.2 (23C64)

6.1.2. Docker

6.1.3. Докер образы

6.1.4. GoLang 1.20

6.1.5. redis 7.2.3-alpine

6.1.6. nginx 1.19.2-alpine

6.1.7. quay.io minio RELEASE.2022-10-15T19-57-03Z

6.1.8. postgres 12

6.2. Клиентская часть

6.2.1. Веб-браузер: Safari 11.1+/Chrome 40+/Opera 27+/Firefox 44+

6.2.2. IPhone: iOS17+

7. Требования к аппаратному обеспечению:

7.1. Серверная часть

7.1.1. Процессор минимум 2-ядерный с частотой от 2 ГГц.

7.1.2. Оперативная память от 4 Гб.

7.1.3. Место на жестком диске от 2 Гб.

7.2. Клиентская часть

7.2.1. Процессор с частотой от 1ГГц.

7.2.2. Оперативная память от 512 Мб.