МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Веб-приложение для планирования и учета совместных путешествий TravelWithFriends

Курсовой проект

по дисциплине

Технологии программирования

09.03.04 Программная инженерия

Информационные системы и сетевые технологии

Преподаватель _____ В.С. Тарасов, ст. преподаватель __.__.20__

Обучающийся _____ Е.С. Воронежская, 3 курс, д/о

Обучающийся _____ М.С. Бондарев, 3 курс, д/о

Обучающийся _____ В.Г. Деревянко, 3 курс, д/о

Руководитель _____ Е.Д. Проскуряков, ассистент

Воронеж 2024

Содержание

Определения, обозначения и сокращения

Термин	Определение
Авторизация	процесс предоставления пользователю или группе пользователей определенных разрешений, прав доступа и привилегий в компьютерной системе.
Аутентификация	процесс проверки подлинности пользователя, предоставляющего учетные данные (логин и пароль) для доступа к системе или сервису
Администратор	компьютер, использующий ресурсы сервера и предоставляющий пользователю возможность взаимодействия с системой
Адрес электронной почты (email)	уникальный идентификатор, используемый для регистрации пользователя в системе
Аккаунт	персональная учетная запись пользователя, которая позволяет ему получить доступ к определенным ресурсам или функциям в рамках системы или сервиса
База данных (БД)	организованное совокупность данных, обычно хранящихся и обрабатываемых с использованием компьютерных систем
Библиотека	набор функций или классов, предназначенных для решения определенной задачи или облегчения разработки программного обеспечения
Браузер	программное обеспечение для просмотра веб-страниц и других ресурсов в Интернете
Веб-приложение	программное обеспечение, доступное через браузер и предназначенное для выполнения определенных функций через Интернет
Программное обеспечение (ПО)	совокупность программных инструкций, данные и документации, предназначенных для работы компьютерной системы или устройства
Отладка	процесс поиска и исправления ошибок в программном коде
Пользователь	авторизованный в системе человек, использующий веб-приложение
Гость	неавторизованный в системе человек, использующий часть функционала веб-приложения
Пароль	секретная комбинация символов, используемая для аутентификации пользователя и обеспечения безопасности данных
Цифровизация	это процесс превращения аналоговых данных и рабочих процессов в цифровой формат

Сервер, серверная часть	компьютер или программа, предоставляющая ресурсы или услуги другим компьютерам или программам
Система управления базами данных (СУБД)	программное обеспечение для управления базами данных
Фреймворк	набор библиотек и инструментов, облегчающих разработку программного обеспечения
Header (хедер)	это верхняя часть веб-страницы, которая содержит логотип компании, навигационные ссылки и другие элементы для управления и навигации по сайту
C#	объектно-ориентированный язык программирования общего назначения
.NET	модульная платформа для разработки программного обеспечения с открытым исходным кодом
ASP.NET Core	свободно-распространяемый кроссплатформенный фреймворк для создания веб-приложений на платформе .NET с открытым исходным кодом
Backend (бэкенд)	часть программного обеспечения, отвечающая за обработку данных и взаимодействие с базой данных
Frontend (фронтенд)	часть программного обеспечения, отвечающая за визуальное представление данных и взаимодействие с пользователем
TypeScript (TS)	скриптовый (сценарный) язык программирования, используемый для создания интерактивных веб-страниц
React	это фреймворк JavaScript с открытым кодом для создания внешних пользовательских интерфейсов
HTTP	протокол передачи данных в сети Интернет, который используется для передачи информации между клиентом и сервером
HTTPS	защищенная версия протокола HTTP, использующая шифрование для безопасной передачи данных
REST API (REST, Representational State Transfer)	архитектурный стиль веб-служб, который использует протокол HTTP для передачи данных между клиентом и сервером
SQL	язык структурированных запросов, используемый для взаимодействия с базами данных

Введение

В современном мире путешествия становятся все более доступным способом проведения свободного времени.

Согласно отчету World Tourism Organization, в 2023 году количество международных поездок превысило 1,4 миллиарда, демонстрируя восстановление туристической индустрии после пандемии COVID-19. В 2024 году количество поездок по России должно составить 80 миллионов. Примерно две трети путешественников предпочтут приобретать билеты и бронировать отели без посредников. При этом около 60% россиян будут ездить только по России. Ожидается, что к 2027 году количество путешествий по стране должно достигнуть отметки в 103 миллиона.

Совместные поездки могут быть сложным организационным испытанием, особенно когда речь идет о планировании маршрутов и учете финансов. Тем не менее, все больше людей обращаются к путешествиям в кругу друзей как к предпочтительному варианту отпуска, особенно в летнее время. Проблема эффективного планирования и учета финансов при совместных путешествиях часто ведет к напряженным отношениям между участниками и даже к разногласиям после возвращения домой.

В данной курсовой работе был реализован сервис для организации и планирования совместных путешествий, что включает в себя составление плана поездки, визуализацию маршрутов на карте, ведение расходов и построение статистики на их основе.

1 Постановка задачи

Целью данного проекта является создание сервиса совместных путешествий. Этот сервис предоставит клиентам возможность пригласить участников в созданное путешествие, составить маршрут, доступный каждому участнику поездки. Каждый участник поездки сможет внести трату в общий список трат, которая отразится в статистике, а также указать участников этой траты. После чего каждый участник сможет посмотреть наглядную статистику расходов.

1.1 Требования к разрабатываемой системе

1.1.1 Функциональные требования

К разрабатываемому сервису выдвигаются следующие функциональные требования:

— просмотр общей информации опубликованных поездок;

— создание и планирование поездки авторизованным пользователем;

— отслеживание маршрута, проходящего через активности, отмеченные на карте, авторизованным пользователем;

— добавление расходов в путешествие авторизованным пользователем;

— просмотр статистики расходов за путешествие авторизованным пользователем;

— сохранение истории путешествий авторизованным пользователем;

— оформление «подписки» авторизованным пользователем;

— управление статусом пользователя администратором;

— удаление поездок и пользователей администратором.

1.1.2 Нефункциональные требования

К разрабатываемому сервису выдвигаются следующие нефункциональные требования:

— сервис должен обладать интерфейсом, выполненном в едином стиле со всем необходимым набором функций;

— сервис должен использовать современные технологии и инструменты разработки.

1.2 Требования к архитектуре

Список требований к архитектуре:

— приложение должно быть построено с использованием протоколов HTTPS;

— для хранения информации необходимо использовать реляционную базу данных;

— клиентская часть приложения должна быть написана с использованием технологий frontend разработки, таких как HTML, CSS, TypeScript с фреймворком React. Выбор этого фреймворка объясняется тем, что он обладает простым синтаксисом, позволяет обновлять только те элементы, которые требуют изменений и использовать повторно уже существующие элементы;

— серверная часть приложения должна быть написана с использованием технологий backend разработки, таких как язык программирования C# и фреймворк ASP.NET Core [1]. Этот кроссплатформенный фреймворк обладает высокой производительностью и эффективностью благодаря использованию языка C# и оптимизациям внутри платформы. Имеет интегрированную поддержку для разработки RESTful API.

Обеспечивает широкий спектр инструментов и библиотек для разработчиков, включая Entity Framework для работы с базами данных, инструменты автоматизации развертывания.

1.3 Задачи, решаемые в процессе разработки

Процесс организации данного веб-приложения построен на основе гибкой методологии Kanban.

В процессе разработки сервиса планирования и учета совместных путешествий будут решаться следующие задачи:

- анализ предметной области: необходимо изучить специфику планирования совместных путешествий;
- проектирование базы данных: на основе полученных требований необходимо разработать структуру базы данных, которая будет использоваться в приложении;
- разработка серверной части приложения: на этом этапе необходимо разработать серверную часть приложения, которая будет отвечать за обработку запросов клиента и взаимодействие с базой данных. Для этого используется фреймворк ASP.NET Core;
- разработка клиентской части приложения: клиентская часть приложения должна быть написана с использованием современных технологий frontend разработки, таких как HTML, CSS, TypeScript;
- тестирование и отладка: на этом этапе необходимо провести тестирование и отладку приложения, чтобы убедиться, что оно соответствует требованиям, определенным в начале проекта.

2 Анализ предметной области

2.1 Глоссарий предметной области

Активность — любое действие или занятие, которое человек совершает или планирует совершить во время своего пребывания в поездке. Может включать в себя различные виды деятельности, такие как экскурсии, посещение достопримечательностей, пешие или велосипедные прогулки, занятия спортом, походы, культурные мероприятия и другие формы отдыха или развлечений.

Денежные траты — сумма денег, которую человек использует для покупки товаров или услуг.

Маршрут — путь, который следует пройти или проехать от одной точки (начальной) к другой (конечной).

Поездка — путешествие с целью отдыха, туризма, включает в себя формирование плана, определение дат начала и окончания.

Расходы — денежные траты, совершенные с той или иной целью.

Участник поездки — пользователь, который принимает участие в поездке вместе с другими пользователями.

2.2 Обзор аналогов

2.2.1 Troupe

Troupe – приложение для планирования групповых поездок. В troupe есть различные разделы для облегчения планирования, такие как опросы, проживание, мероприятия, маршрут. Интерфейс удобный и понятный. Есть интеграция с рядом отелей за рубежом. Из недостатков можно выделить отсутствие учета расходов и отсутствие истории путешествий. Интерфейс приложения представлен на Рисунке 1.

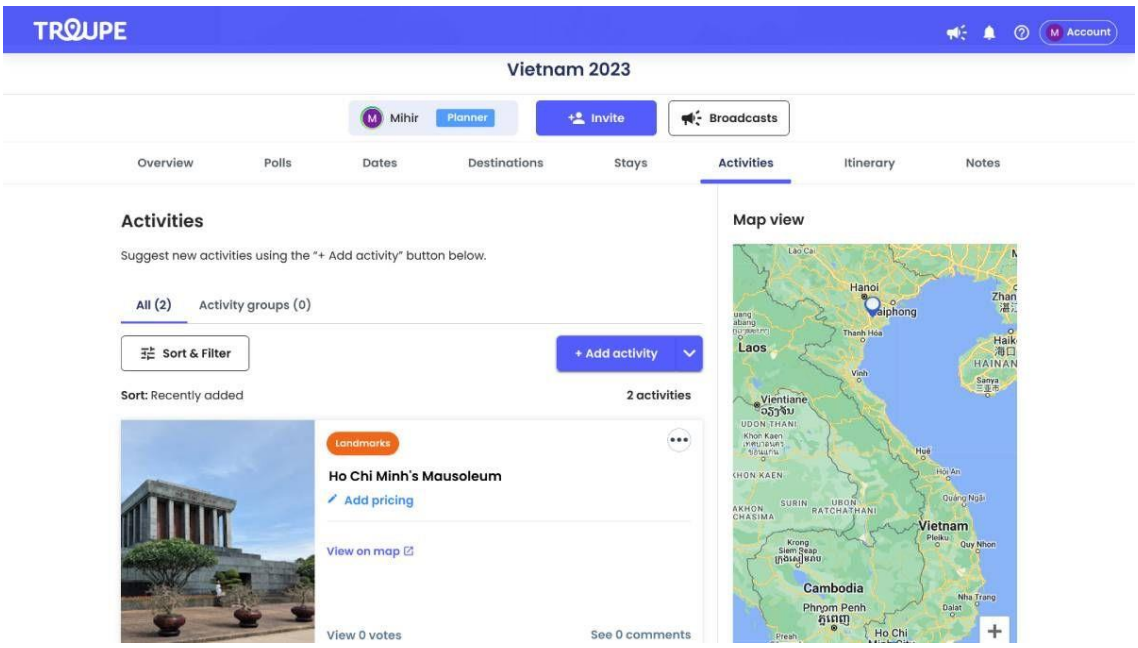

Рисунок 1 — Внешний вид добавления мероприятия «Troupe»

2.2.2 Plan Harmony

Plan Harmony – календарное планирование и отслеживание бюджета для групповых поездок. Plan Harmony позволяет создавать несколько поездок, в которые можно пригласить друзей, просто поделившись ссылкой, после чего можно совместно их спланировать. После установки дат поездки, по умолчанию отображается календарь, поэтому планирование происходит, как при использовании общего календаря Google. Из преимуществ, это приложение позволяет вести совместный учет расходов, предоставляет статистику расходов. Из недостатков, нет визуализации маршрутов. Интерфейс приложения представлен на Рисунке 2.

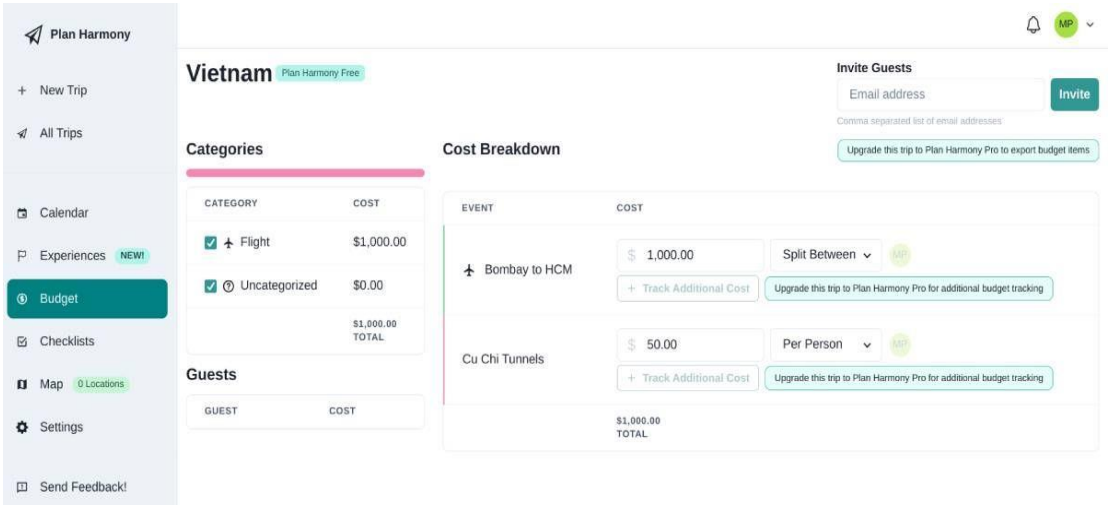

Рисунок 2 — Внешний вид добавления мероприятия «Plan Harmony»

2.2.3 MiTravel

MiTravel – приложение, использующее систему канбан-досок для планирования путешествий. На этой досте путешественники вместе добавляют различные элементы для совместной поездки и проводят опросы для быстрого принятия решений. Приложение бесплатно. Из недостатков, дизайн интуитивно не понятен. Интерфейс приложения представлен на Рисунке 3.

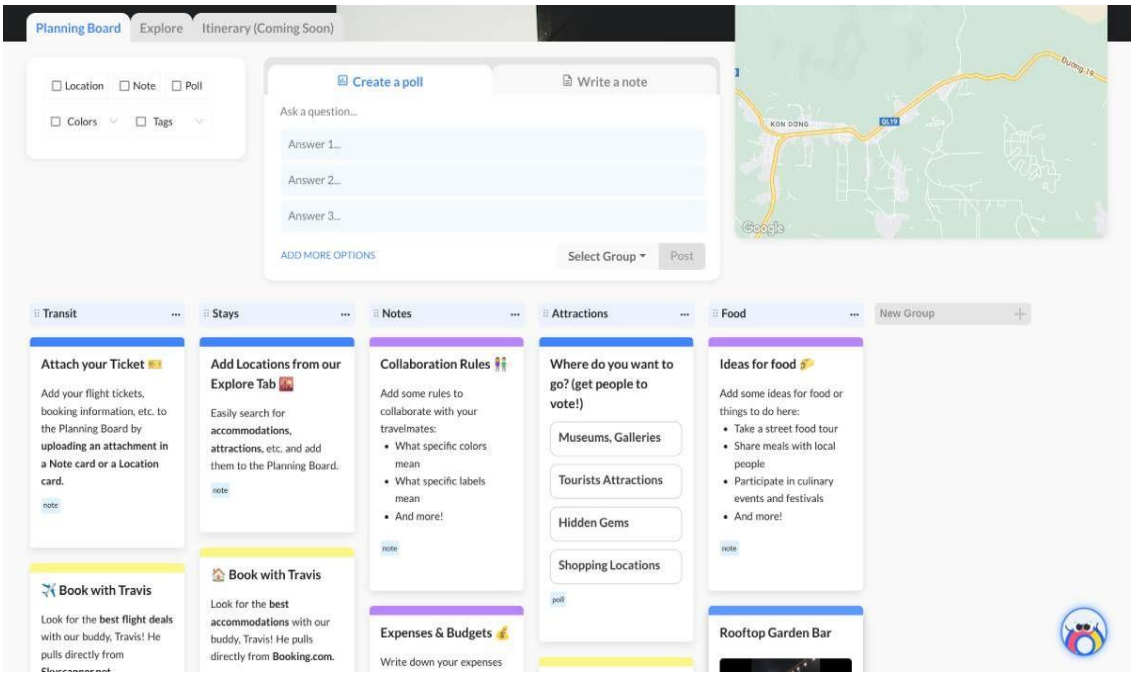

Рисунок 3 — Внешний вид доски «MiTravel»

2.2.4 Сравнительная таблица аналогов

В ходе исследования рынка приложений для совместных путешествий было выявлено 3 прямых конкурента.

Таблица содержит результаты проведённого конкурентного исследования.

Таблица 1 — Результаты конкурентного исследования

Характеристика	Troupe	Plan Harmony	MiTravel
Карта для отображения маршрута	+	-	+
Учет и статистика расходов	-	+	-
Встроенный чат	+	+	-
История путешествий	-	+	-
Удобный интерфейс	+	+	-

2.3 Диаграммы, иллюстрирующие работу системы

2.3.1 Диаграмма прецедентов

Диаграмма прецедентов представлена для трёх типов акторов: неавторизованного пользователя, авторизованного пользователя, администратора. У каждого из них своя модель поведения, которую можно проследить на Рисунках 4-6.

Неавторизованный пользователь может:

— зарегистрироваться в системе;

— авторизоваться в системе;

— просматривать опубликованные путешествия;

— просматривать главную страницу.

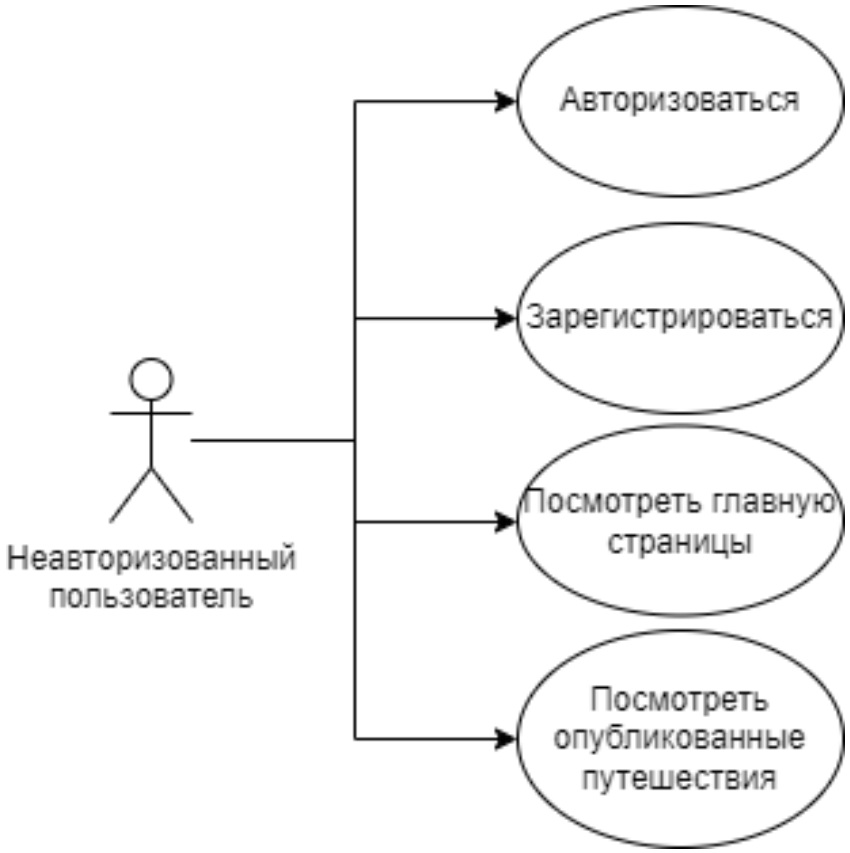

Рисунок 4 — Диаграмма прецедентов для неавторизованного пользователя

Авторизованный пользователь помимо функций, доступных неавторизованному пользователю, может:

— создавать путешествия: добавлять пользователей в команду, добавлять даты начала и окончания поездки;

— формировать список активностей, включая просмотр маршрута между точками активностей, внесение трат на активность, указание членов команды, участвовавших в данной активности;

— просмотреть статистику расходов за поездку;

— взаимодействовать с личным кабинетом: посмотреть личную информацию, указанную на сайте, посмотреть историю путешествий;

— оформить подписку.

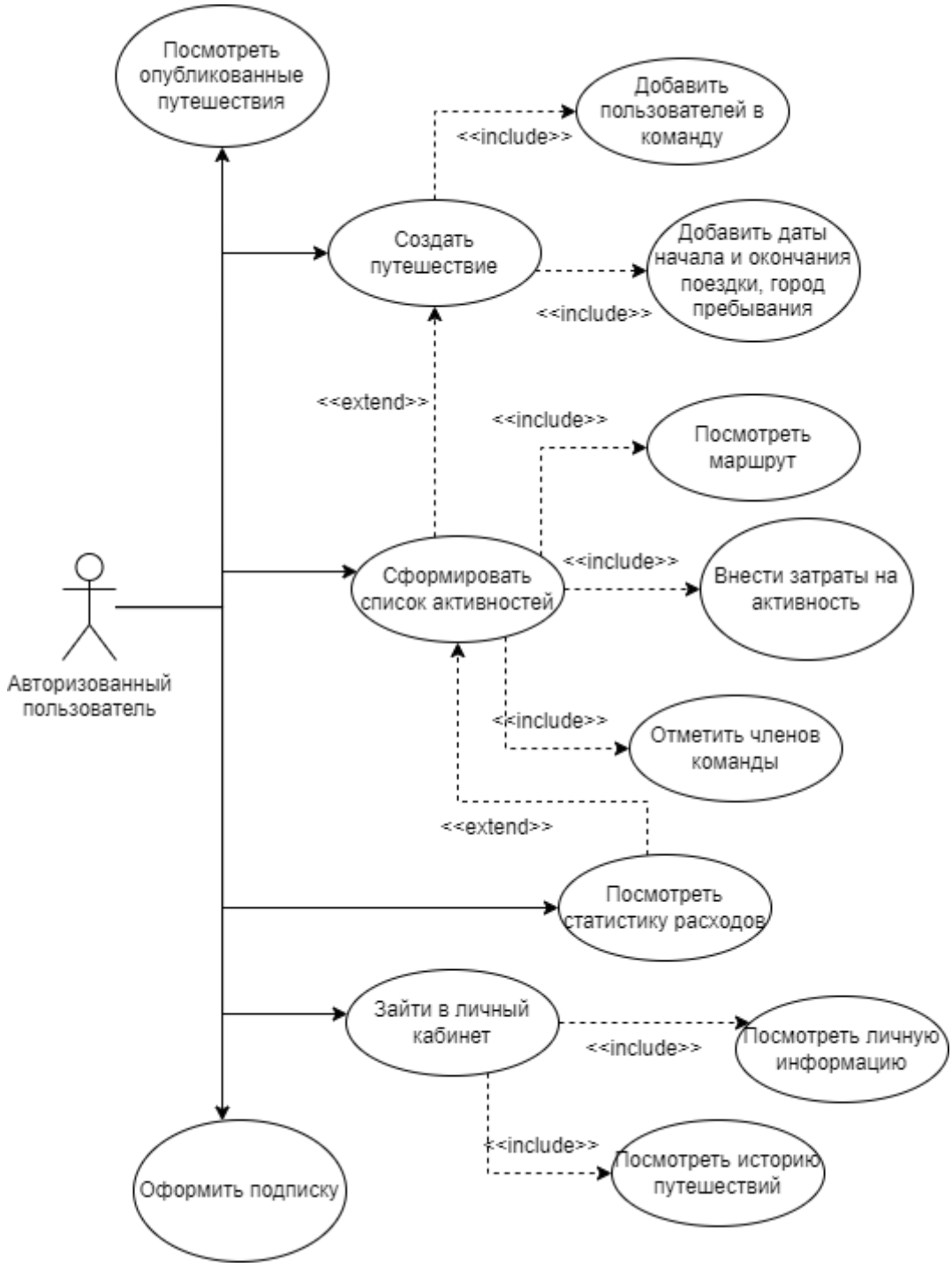

Рисунок 5 — Диаграмма прецедентов для авторизованного пользователя

Существуют следующие функции, доступные администратору:

— удалить пользователей;

— управлять подпиской пользователей;

— удалить любое путешествие;

— просматривать страницу опубликованных путешествий.

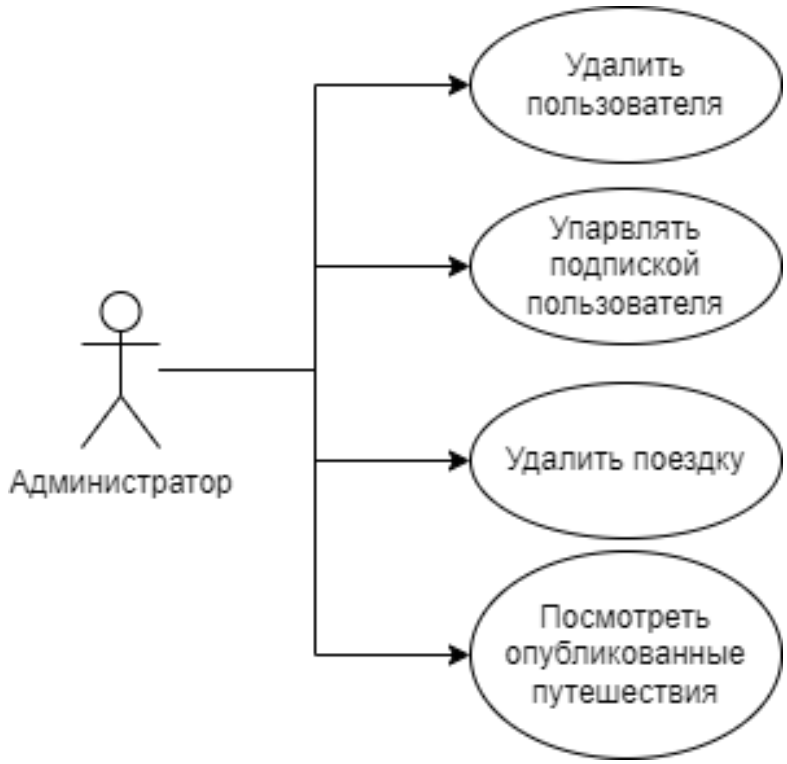

Рисунок 6 — Диаграмма прецедентов для администратора

2.3.2 Диаграмма последовательности

Существует также диаграмма последовательностей (Рисунок 7), на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента. Участником данной системы является пользователь, а объектами – клиент, сервер и база данных.

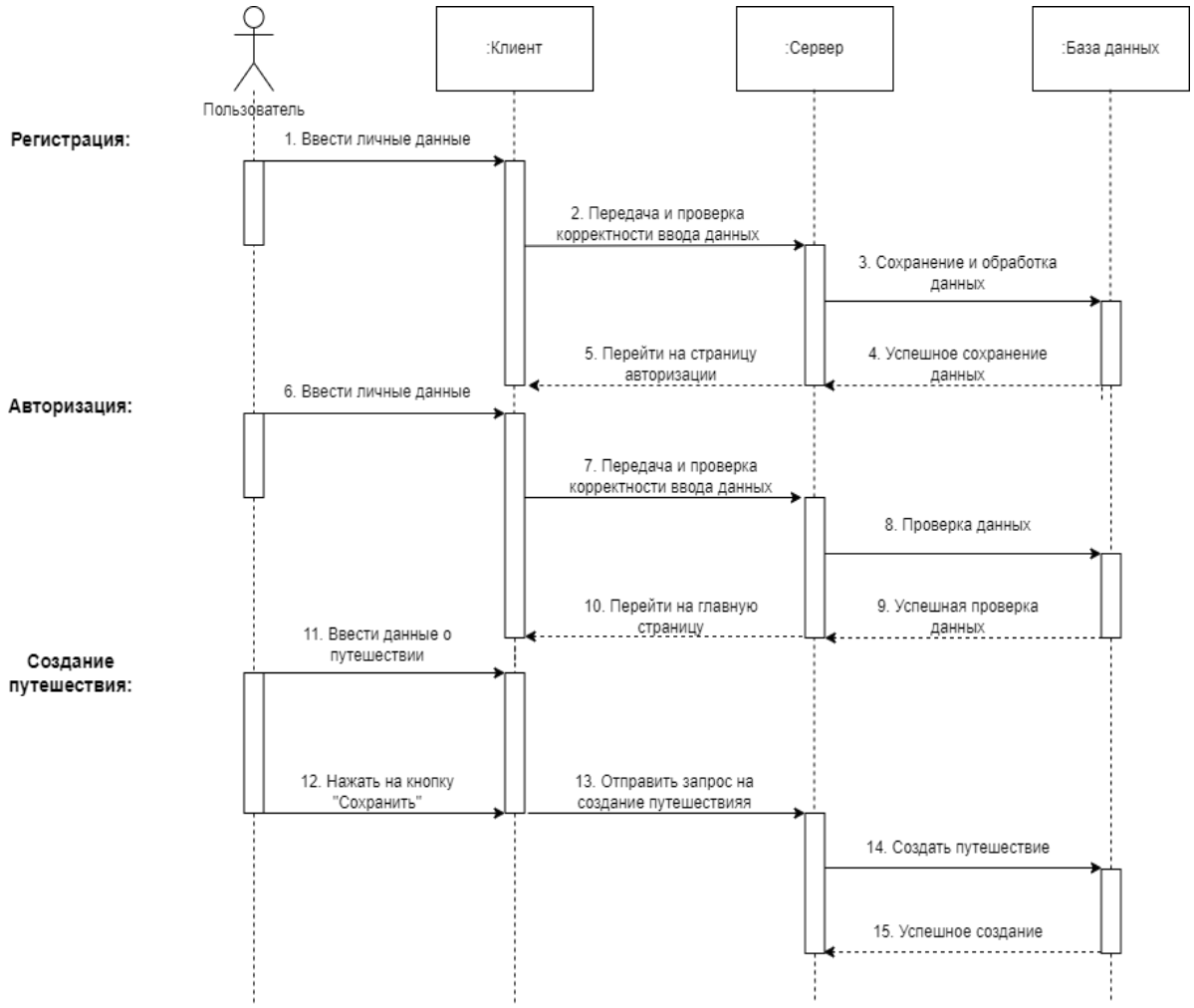

Рисунок 7 — Диаграмма последовательности

2.3.3 Диаграмма состояний

Диаграмма состояний (Рисунок 8) отражает внутренние состояния объекта в течение его жизненного цикла от момента создания до разрушения. На данной диаграмме рассмотрены состояния от момента входа в систему до полного выхода из нее.

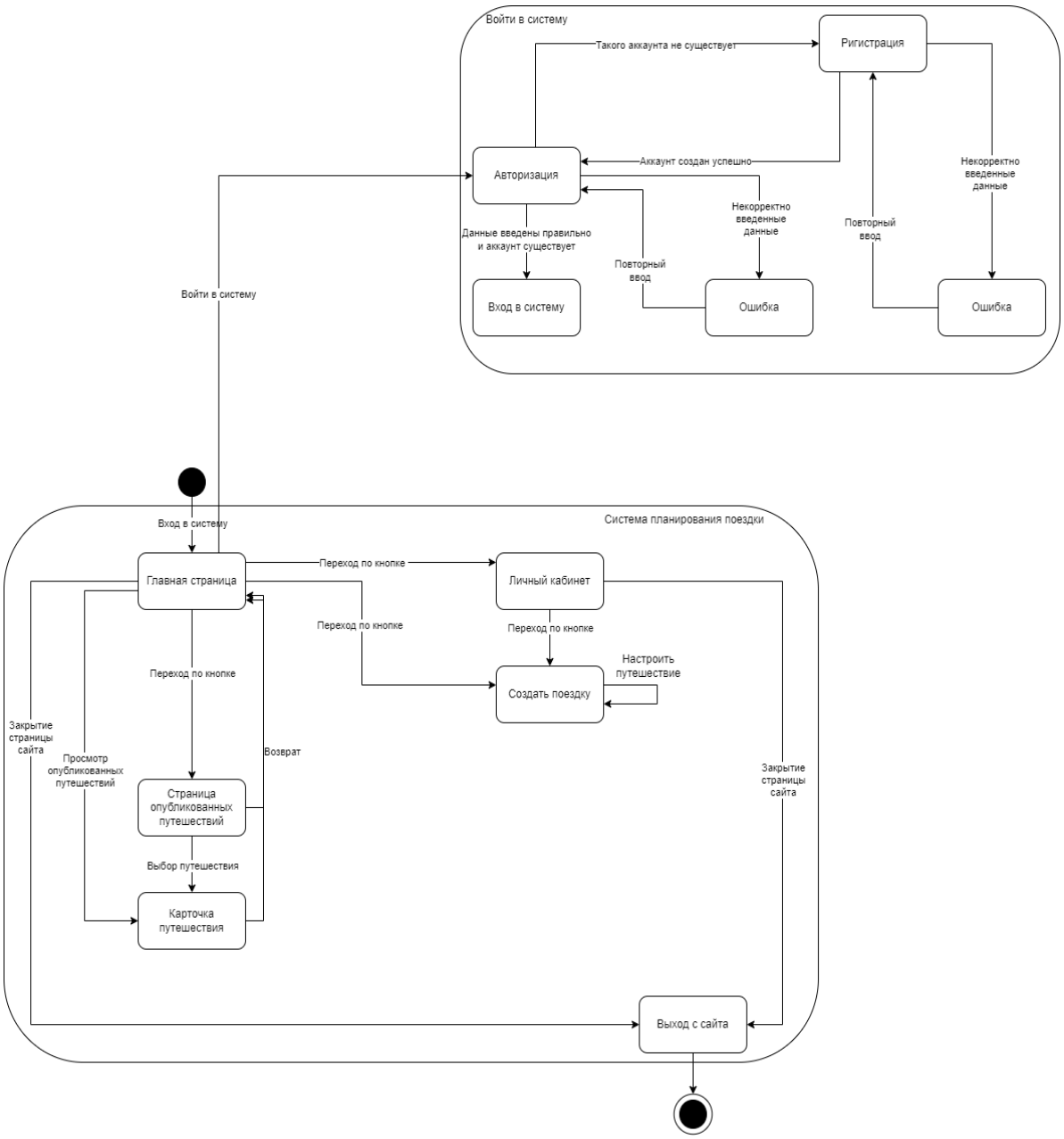

Рисунок 8 — Диаграмма состояний

2.3.4 Диаграмма деятельности

Диаграмма деятельности (Рисунок 9) представляет собой диаграмму, на которой показаны действия, состояния которых описаны на диаграмме состояний. Она описывает действия системы или людей, выполняющих действия, и последовательный поток этих действий.

В данном случае рассмотрен путь действий пользователя.

Диаграмма показывает, что пользователь может использовать основные функции сайта только будучи авторизованным.

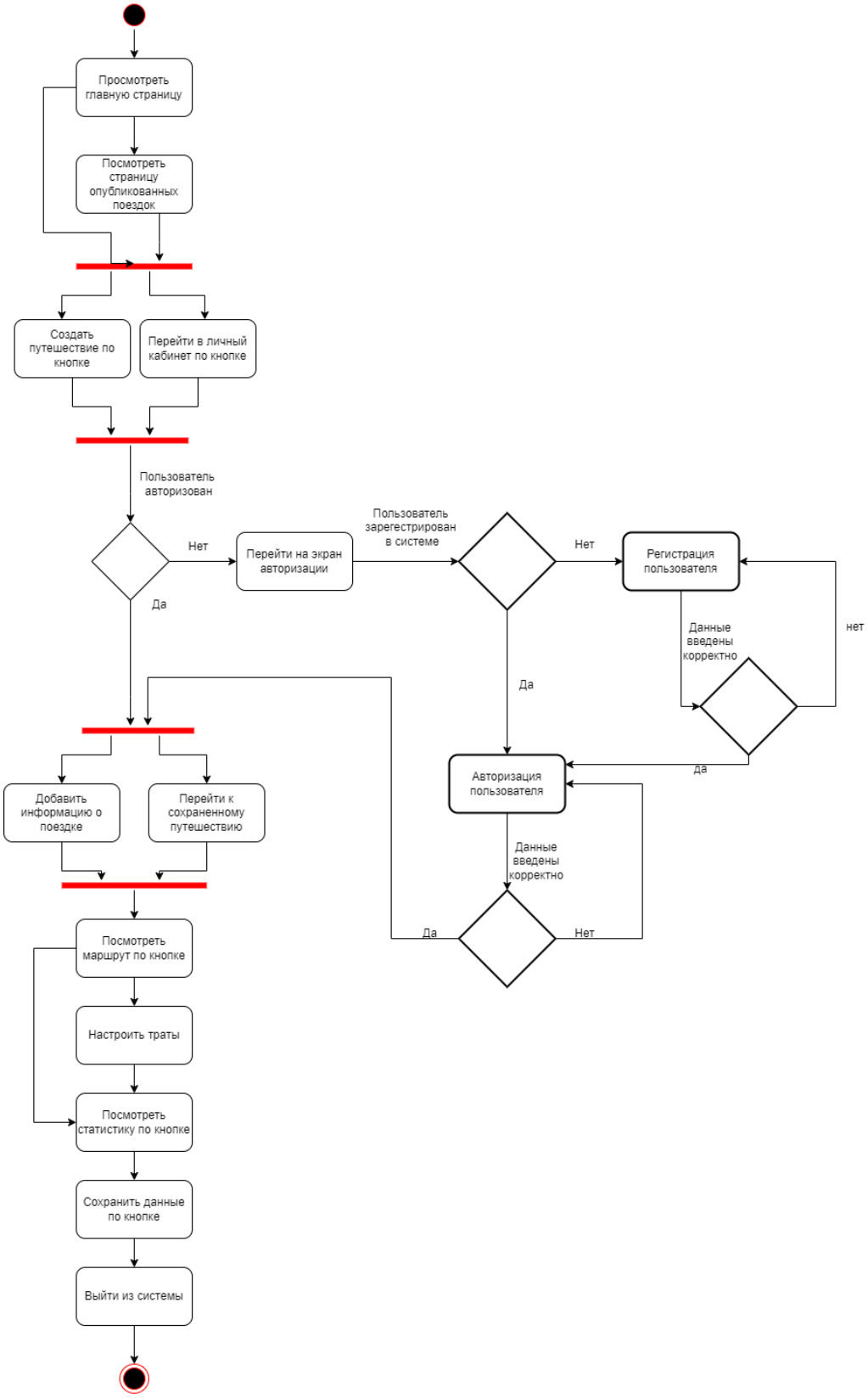

Рисунок 9 — Диаграмма деятельности

2.3.5 Диаграмма классов

Диаграмма классов (Рисунок 10) демонстрирует общую структуру иерархии классов системы, их кооперaций, атрибутов, методов, интерфейсов и взаимосвязей между ними. В данной системе рассмотрены следующие классы:

— класс «Пользователь»;

— класс «Поездка»;

— класс «День поездки»;

— класс «Активность»;

— класс «Категория».

У каждого из классов существуют свои атрибуты.

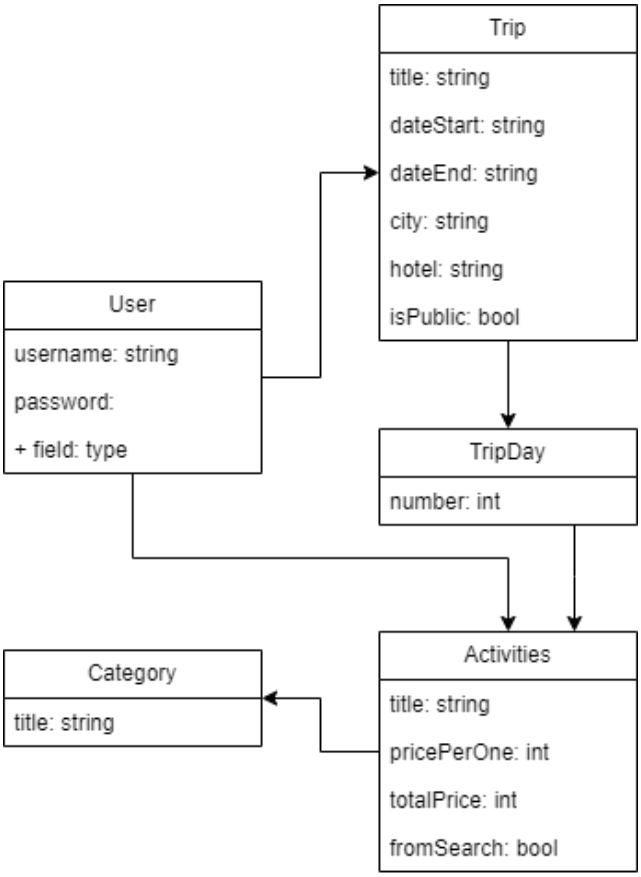

Рисунок 10 — Диаграмма классов

2.3.6 Диаграмма объектов

Была создана диаграмма объектов. (Рисунок 11)

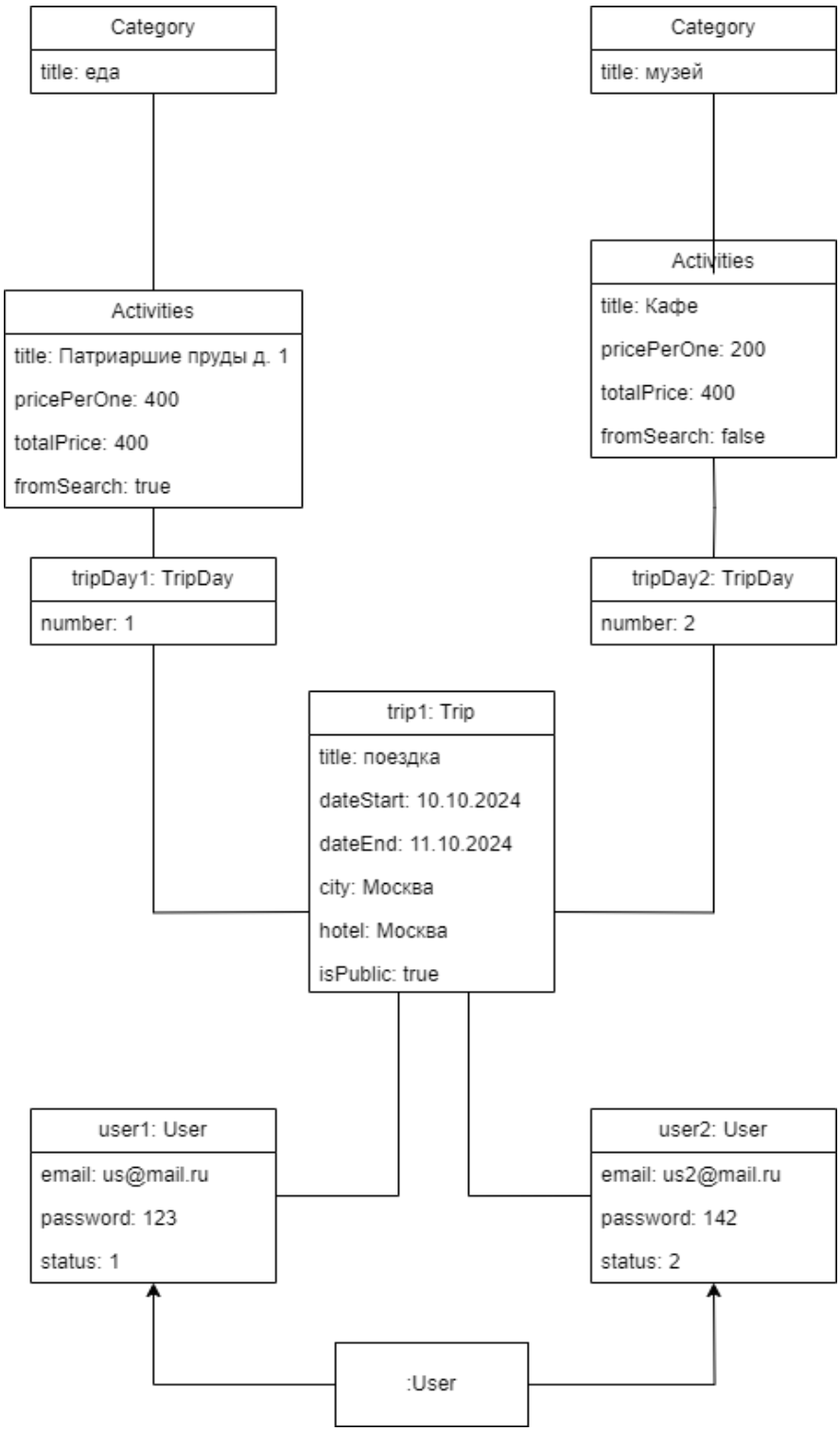

Рисунок 11 — Диаграмма объектов

2.3.7 Диаграмма сотрудничества

Диаграмма сотрудничества (Рисунки 12-13) — это вид диаграммы взаимодействия, в котором основное внимание сосредоточено на структуре взаимосвязей объектов, принимающих и отправляющих сообщения.

Рисунок 12 — Диаграмма сотрудничества при авторизации

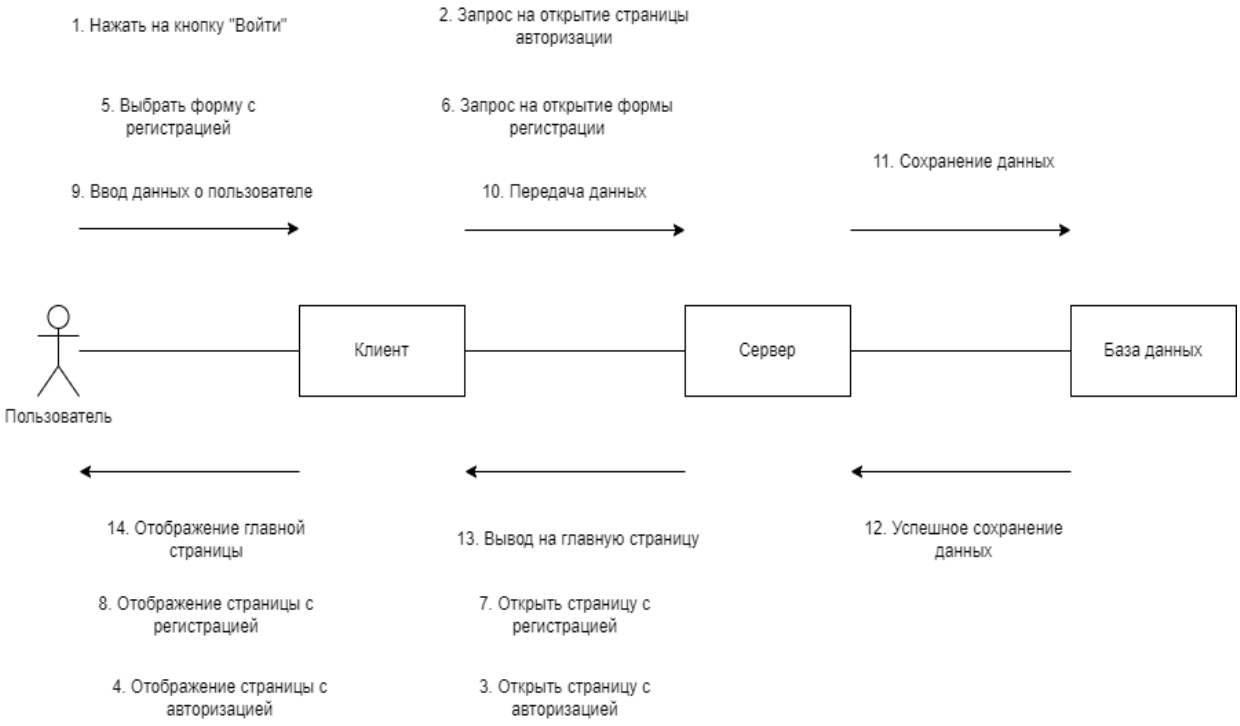

Рисунок 13 — Диаграмма сотрудничества при регистрации

2.3.8 Диаграмма развертывания

Диаграмма развертывания (Рисунок 14) предназначена для представления общей конфигурации или топологии распределенной программной системы [3].

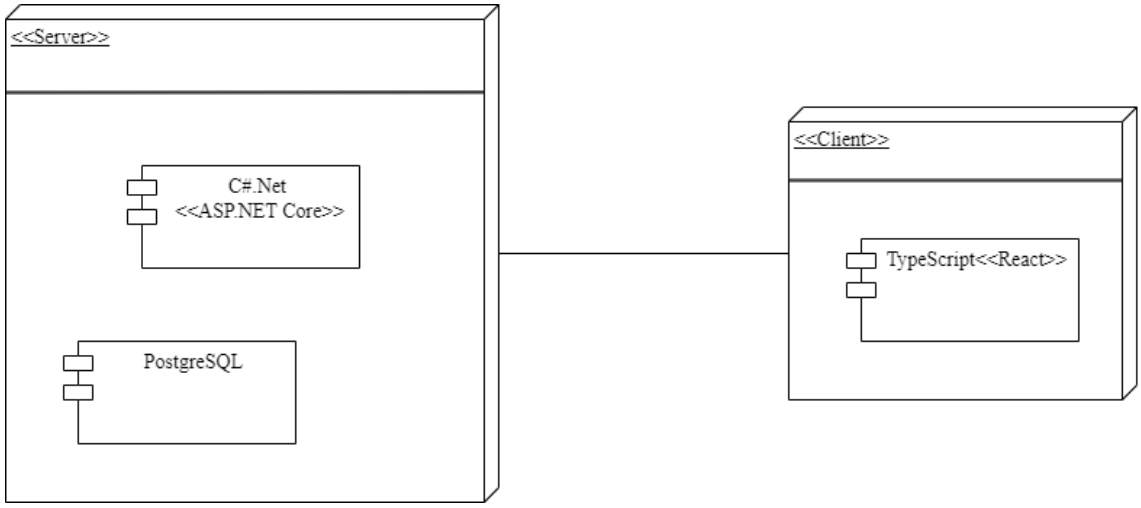

Рисунок 14 — Диаграмма развертывания

2.3.9 ER-диаграмма

ER-диаграмма — это графическое представление модели данных, которая используется для описания концептуальной структуры базы данных. В такой диаграмме есть сущности, которые представляют объекты, с которыми работает система, и связи между сущностями, описывающие их взаимодействия. ER-диаграмма помогает наглядно описать структуру базы данных и увидеть связи между ее элементами (Рисунок 15).

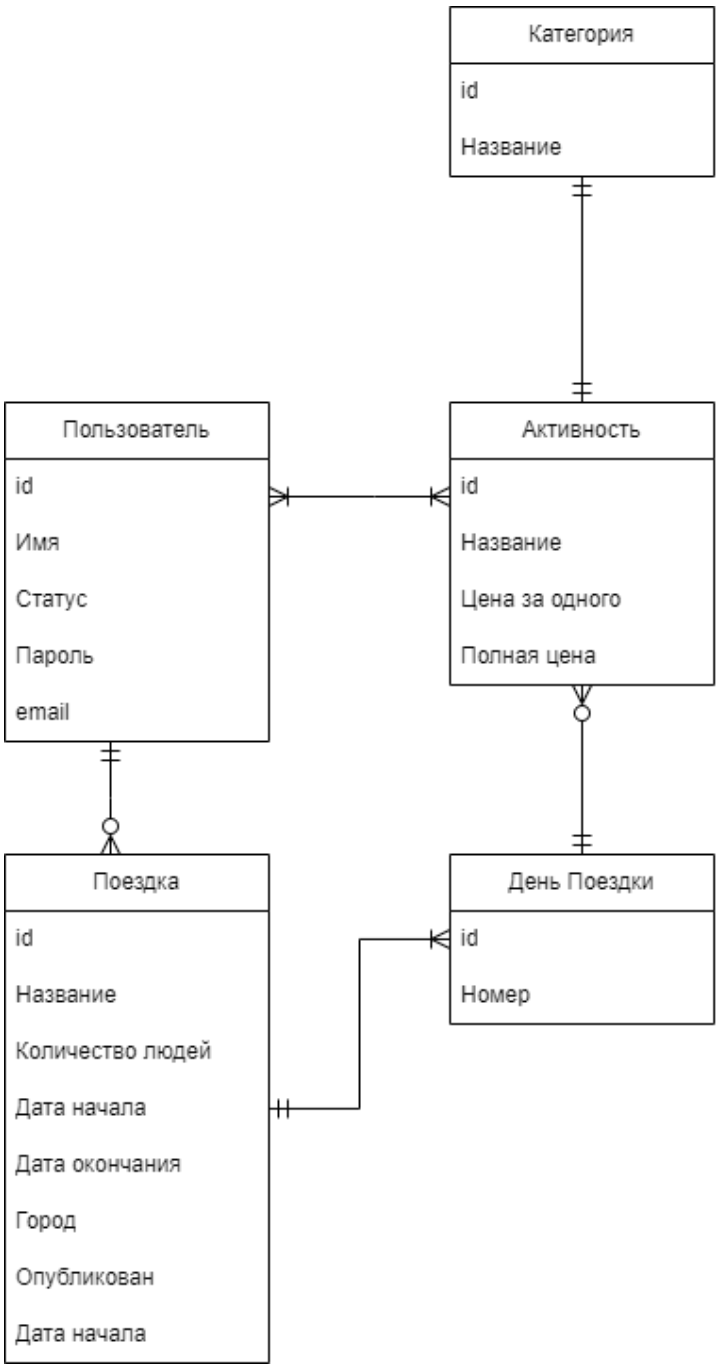

Рисунок 15 — ER-диаграмма

3 Реализация

3.1 Средства реализации

3.1.1 Средства реализации серверной части приложения

Для разработки серверной части приложения был выбран следующий стек технологий:

— язык программирования C# — объектно-ориентированный язык программирования общего назначения, позволяет писать быстрый и эффективный код благодаря использованию JIT-компиляции (Just-In-Time) и оптимизациям в рамках платформы .NET;

— ASP.NET Core 8 — это кроссплатформенный фреймворк для создания серверной части веб-приложений от Microsoft. Он обеспечивает высокую производительность и масштабируемость, предлагая модульную архитектуру, которая позволяет включать только необходимые компоненты. Поддерживает работу на Windows, macOS и Linux, что делает его гибким и удобным для разработки backend части приложения;

— СУБД PostgreSQL 16.2 — это объектно-реляционная система управления базами данных (СУБД), которая предоставляет мощные средства для хранения, организации и манипулирования данными. PostgreSQL является свободным и открытым программным обеспечением и использует SQL для работы с данными. Он предлагает расширяемость, высокую надежность, многофункциональность, поддержку геопространственных данных, JSON-данных и многое другое. PostgreSQL активно развивается и используется в крупных проектах по всему миру;

— IdentityServer — это фреймворк для управления идентификацией и доступом, построенный на основе ASP.NET Core. Он предоставляет

инструменты для реализации аутентификации и авторизации, поддерживает протоколы OpenID Connect и OAuth 2.0, позволяя создавать безопасные и масштабируемые системы единого входа (SSO), управление доступом к ресурсам и токенами. IdentityServer хорошо интегрируется с различными клиентами и ресурсами, что делает его отличным выбором при разработке;

— Swagger — инструмент для документирования и тестирования API. Он позволяет создавать интерактивную документацию для вебсервисов, что упрощает их использование и интеграцию. Swagger автоматически генерирует документацию на основе аннотаций и комментариев в коде, что позволяет разработчикам сосредоточиться на написании логики приложения, а не на создании и поддержке документации. Благодаря Swagger, разработчики могут изучить доступные параметры, модели данных и примеры запросов и ответов.

3.1.2 Средства реализации клиентской части приложения

Для разработки клиентской части приложения был выбран следующий стек технологий:

— язык программирования TypeScript 4.9.5 — строго типизированный язык программирования, расширяющий возможности JavaScript. Обеспечивает удобство разработки, добавляет статическую типизацию и возможности объектно-ориентированного программирования;

— React 18.2.0 — фреймворк JavaScript с открытым исходным кодом. React [2] позволяет разрабатывать компоненты UI в виде функций или классов, которые могут быть многократно использованы, и которые просты в поддержке. React используется для создания одностраничных приложений (SPA), мобильных приложений,

административных дашбордов и других веб-приложений. React реализует виртуальный DOM, который дает библиотеке многие преимущества перед другими фреймворками и библиотеками для разработки пользовательских интерфейсов, такие как скорость, универсальность, простота и удобство тестирования;

— язык гипертекстовой разметки HTML5 — язык разметки для создания веб-страниц. Является стандартом для создания структуры веб-страниц и обеспечивает семантическую разметку;

— формальный язык описания внешнего вида документа CSS3 — каскадные таблицы стилей, используемые для оформления веб-страниц. Позволяет создавать дизайны для веб-страниц, а также обеспечивает адаптивность.

3.2 Реализация серверной части приложения

Серверная часть приложения (backend) была написана на языке C# с использованием фреймворка ASP.NET Core. Структура проекта представляет собой корневую папку решения TravelWithFriends и дополнительные — Data, Services, Shared и Systems (Рисунок 16).

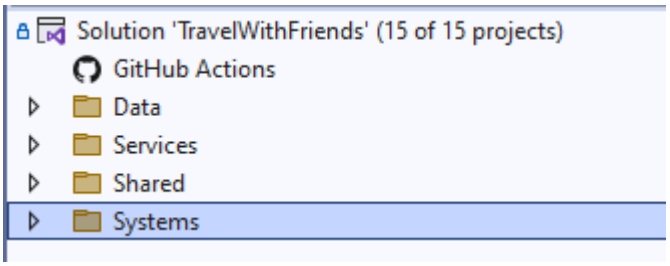

Рисунок 16 — Структура проекта

Папка Data (Рисунок 17) содержит 4 проекта:

— Travel.Context.Migrations – отвечает за миграции. Миграции нужны для управления изменениями структуры базы данных в ходе разработки. Они позволяют автоматически отслеживать изменения моделей данных и применять соответствующие изменения к базе

данных, упрощая процесс разработки и поддержки. С помощью миграций можно легко добавлять новые таблицы, изменять существующие, обновлять схемы и откатывать изменения, обеспечивая синхронизацию кода приложения с базой данных и облегчая командную работу над проектом;

— Travel.Context – отвечает за контекст базы данных (DbContext) и конфигурации. DbContext используется для управления соединением с базой данных и выполнения операций с данными. Он представляет собой основной класс Entity Framework Core, который позволяет взаимодействовать с базой данных, используя объектно-ориентированный подход. DbContext отслеживает изменения в сущностях, управляет запросами и командами SQL;

— Travel.Context.Entities – отвечает за описание сущностей;

— Travel.Context.Seeder – обеспечивает предзаполнение пустой базы данных при первом запуске в новом окружении;

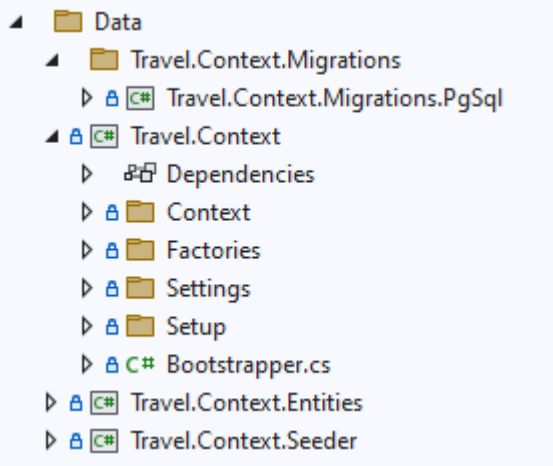

Рисунок 17 — Структура проекта

Папка Services (Рисунок 18) содержит 7 проектов:

— Travel.Services.Activities – включает описание моделей для создания и обновления активностей, а также методы бизнес-логики;

— Travel.Services.Categories – бизнес-логика работы с категориями, которые используются при работе с активностями;

— Travel.Services.Logger – сервис для обеспечения логгирования;

— Travel.Services.Settings – сервис для задания настроек;

— Travel.Services.Stat – сервис для вычисления статистики для каждого путешествия;

— Travel.Services.Trips – сервис путешествий, включает описание моделей для создания и обновления путешествий, а также методы бизнес-логики;

— Travel.Services.UserAccount – сервис работы с аккаунтами пользователей;

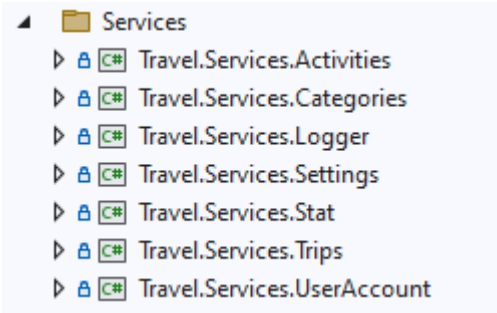

Рисунок 18 — Папка Services

Папка Shared включает в себя один проект Travel.Shared.Common. Это проект с общей конфигурацией проекта: он включает в себя блоки:

— Exception – описание исключений;

— Extension – настройки сборки, расширения некоторых системных классов;

— HealthCheck – отвечает за предоставление информации о текущем состоянии системы, проверяя такие аспекты, как доступность баз данных, подключение к внешним сервисам, состояние памяти и другие критически важные компоненты;

— Helpers – расширения для маппинга сущностей;

— Limits – описание лимитов пользователей;

— Responces – описание структуры ответов на запросы;

— Security – установление уровней доступа для пользователей;

— Settings – установление путей к файлам настроек, таких как appsettings.json;

— ValidationRules – правила валидации данных;

— Validator – подключение валидатора;

В папке Systems содержится 2 папки: Api и Identity.

Папка Api содержит проект Travel.Api, который является точкой входа в приложение. Он содержит контроллеры, настройки Swagger, а также некоторые конфигурации.

Папка Identity содержит проект Travel.Identity, который отвечает за Обеспечение авторизации с помощью IdentityServer, который предоставляет решение для управления доступом к ресурсам в веб-приложениях. Он обеспечивает механизмы для выдачи и проверки токенов доступа, позволяет настраивать правила доступа и роли пользователей, а также обеспечивает безопасность через шифрование и проверку подлинности токенов.

База данных была развернута в контейнере с помощью системы контейнеризации Docker. Такие данные как хост, порт, имя базы данных, имя пользователя и пароль задавались при помощи файла appsettings.json.

Для развертывания системы был написан файл docker-compose.yaml, который определяет структуру и конфигурацию многоконтейнерного приложения, включая службы (контейнеры), их параметры, зависимости, сети и тома данных.

Для каждой службы был написан свой Dockerfile, который содержит инструкции для создания образа Docker. Этот файл определяет шаги и команды, необходимые для создания контейнера.

Тажке Dockerfile был написан для клиентской части приложения, чтобы обеспечить создание и запуск контейнера.

Для защиты приложения от непосредственного доступа из внешней сети был подключен Nginx. Он обрабатывает входящие HTTP-запросы от клиентов, направляя их к соответствующим приложениям или сервисам. Nginx также выполняет терминацию SSL/TLS, обеспечивая безопасное соединение с клиентом, и пересылая нешифрованный трафик к серверам приложений внутри защищенной сети.

Для описания спецификации API использовался Swagger [5]. Для того чтобы интегрировать его в проект, нужно было внести изменения в код, добавив в Travel.Api класс SwaggerConfiguration, который представляет собой описание конфигурации и добавление поддержки Swagger.

3.3 Реализация клиентской части приложения

Клиентская часть приложения (frontend) разработана на языке TypeScript с использованием фреймворка React.

Одной из особенностей разработки приложения на React является декомпозиция приложения на независимые компоненты. Каждый компонент отвечает за определенный функциональный блок приложения и представляет собой функцию, которая возвращает HTML-код. Кроме того, компоненты могут содержать переменные и другие функции, что позволяет дополнительно упростить код и повысить его читабельность. Приложение загружается на страницу по мере необходимости, что уменьшает время загрузки страницы и повышает производительность приложения в целом.

Основной файл называется index.tsx, именно в него загружаются компоненты, которые будут выведены в браузере. В нём находится точка входа React-приложения, где вызывается основной компонент App.tsx, который будет меняться в зависимости от действий пользователя. Все остальные компоненты вызываются по мере необходимости.

Структура проекта на React включает в себя определенные основные элементы (Рисунок 19).

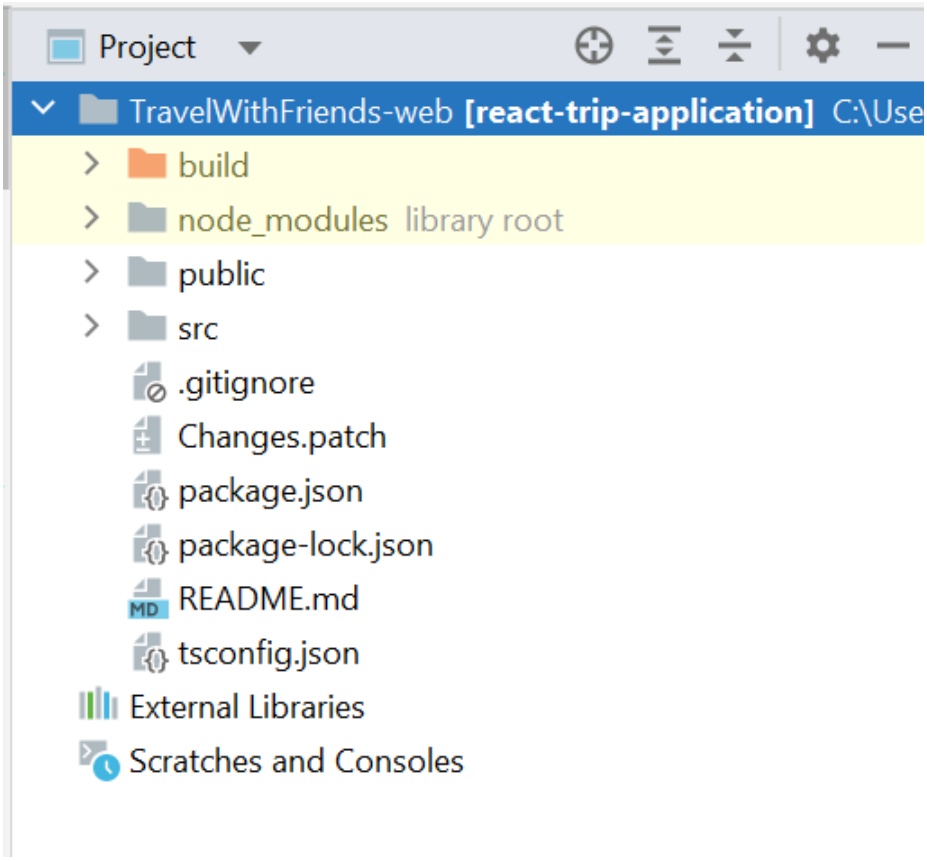

Рисунок 19 — Основные элементы React-проекта

— основная папка проекта src/, содержащая исходный код;

— папка public/, содержащая файлы, которые будут доступны публично;

— файл package.json, содержащий информацию о проекте, а также список зависимостей для установки;

— файл package-lock.json с информацией о текущей версии зависимостей.

В свою очередь в папке src/ (Рисунок 20) хранятся следующие разделы приложения.

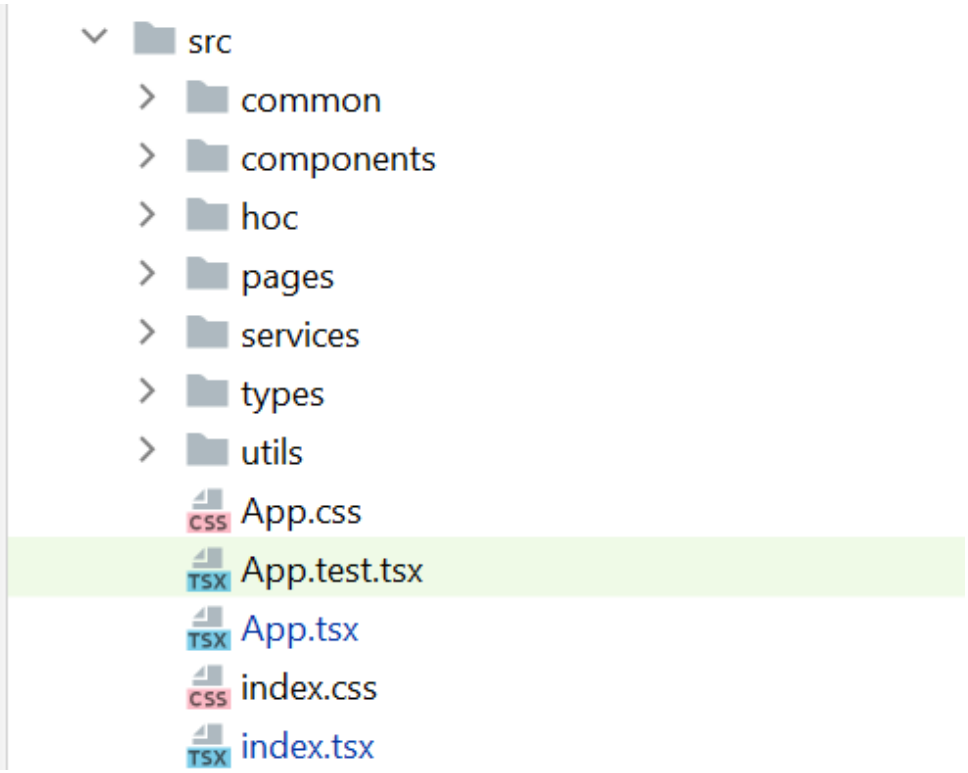

Рисунок 20 — Структура папки src/ проекта

— папка common/, хранящая общие утилиты и вспомогательные функции, содержит в себе EventBus, который используется для глобального управления событиями в приложении;

— папка components/, содержащая все компоненты приложения;

— папка hoc/ (Higher-Order Components), содержащая компоненты высшего порядка. Эти компоненты принимают другие компоненты в качестве аргументов и возвращают новые компоненты с дополнительной функциональностью;

— папка pages/, содержащая существующие страницы нашего приложения;

— папка services/, совершающая обращение к серверу через API;

— папка types/, содержащая определения типов и интерфейсов, используемых в нашем приложении;

— папка utils/, содержащая утилиты, которые используются в приложении.

Компоненты могут состоять из других компонентов. Поэтому в приложении помимо компонентов, описывающих конкретные страницы, есть компоненты, из которых состоят эти страницы. Компонентами приложения являются следующие разделы (Рисунок 21).

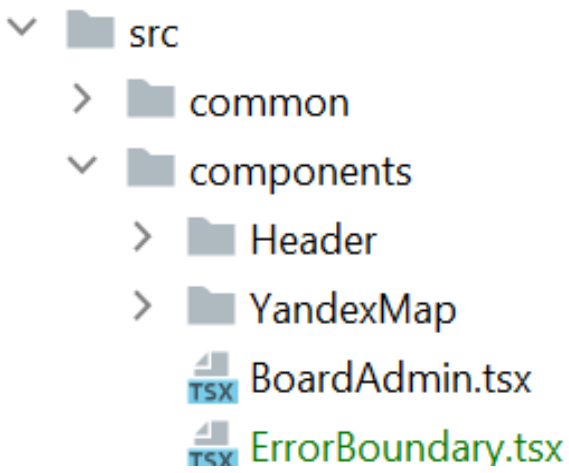

Рисунок 21 — Компоненты приложения

— папка Header – отображение основного заголовка и навигационных элементов в верхней части приложения;

— папка YandexMap – интегрирует карту Yandex в приложение, предоставляет функционал для работы с маршрутами, метками и поиском по карте;

— BoardAdmin.tsx – компонент для управления административной панелью;

— ErrorBoundary.tsx – компонент для обработки ошибок.

3.4 Навигация по приложению

3.4.1 Для неавторизованного пользователя

Первое, что видит пользователь — главный экран (Рисунок 22).

Далее у пользователя возникает выбор: он может войти в личный кабинет при помощи авторизации (Рисунок 23), если нет аккаунта – при помощи регистрации (Рисунок 24) или продолжить пользоваться сайтом в режиме неавторизованного пользователя, но с ограничениями (без возможности создавать поездки, отслеживать статистику, просматривать личный кабинет).

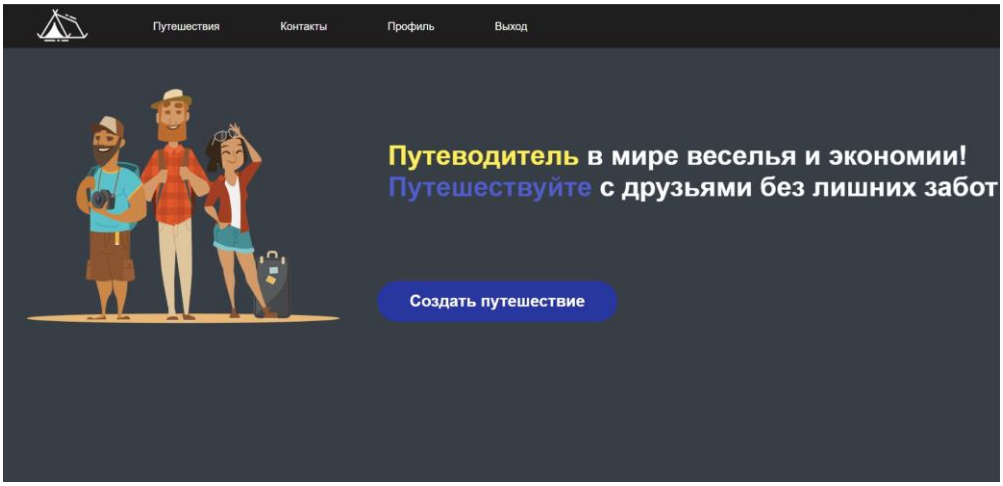

Рисунок 22 — Главная страница сайта

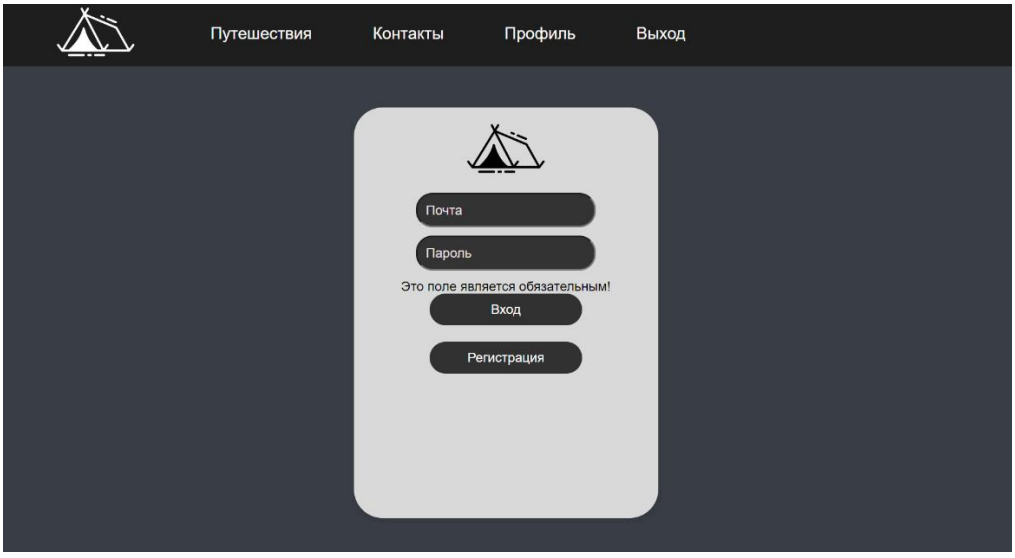

Рисунок 23 — Страница авторизации

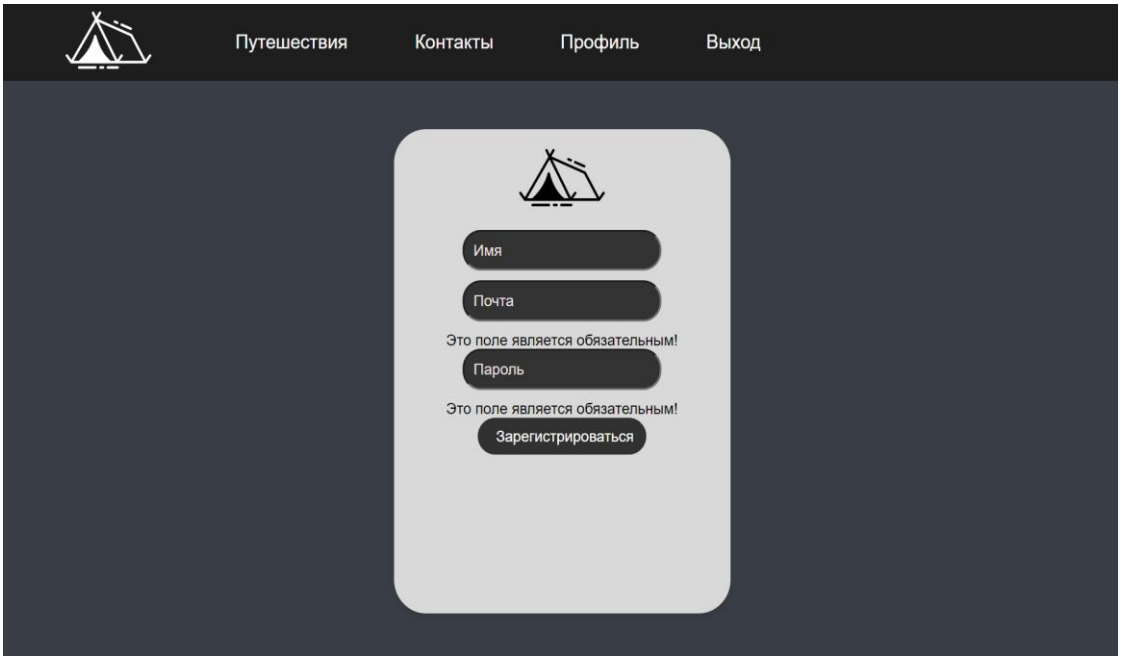

Рисунок 24 — Страница регистрации

Неавторизованному пользователю доступен просмотр опубликованных путешествий, где можно осуществить просмотр путешествий, созданных другими пользователями (Рисунок 25).

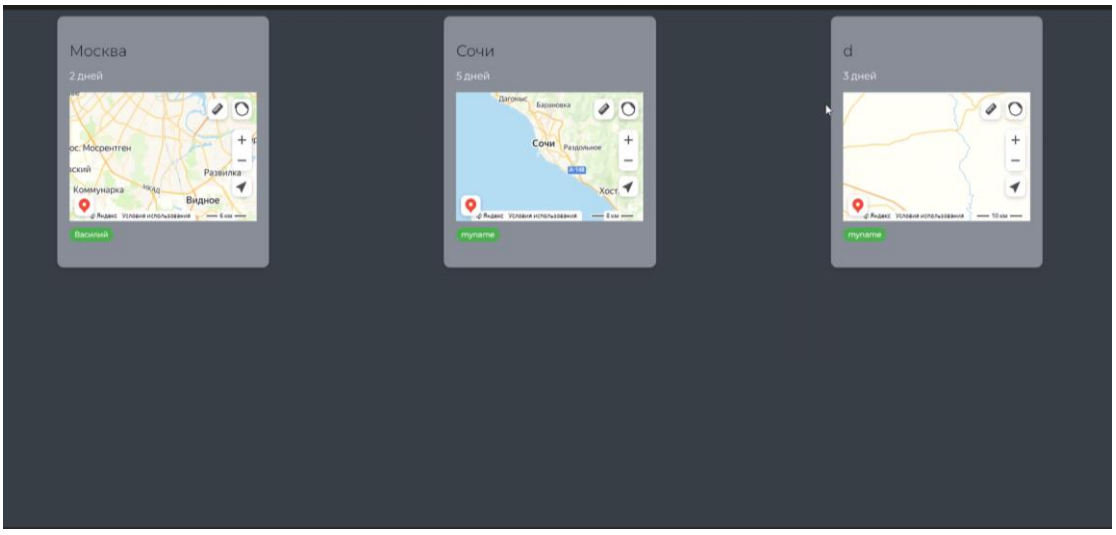

Рисунок 25 — Страница опубликованных путешествий

Можно просматривать активности определенного путешествия (Рисунок 26).

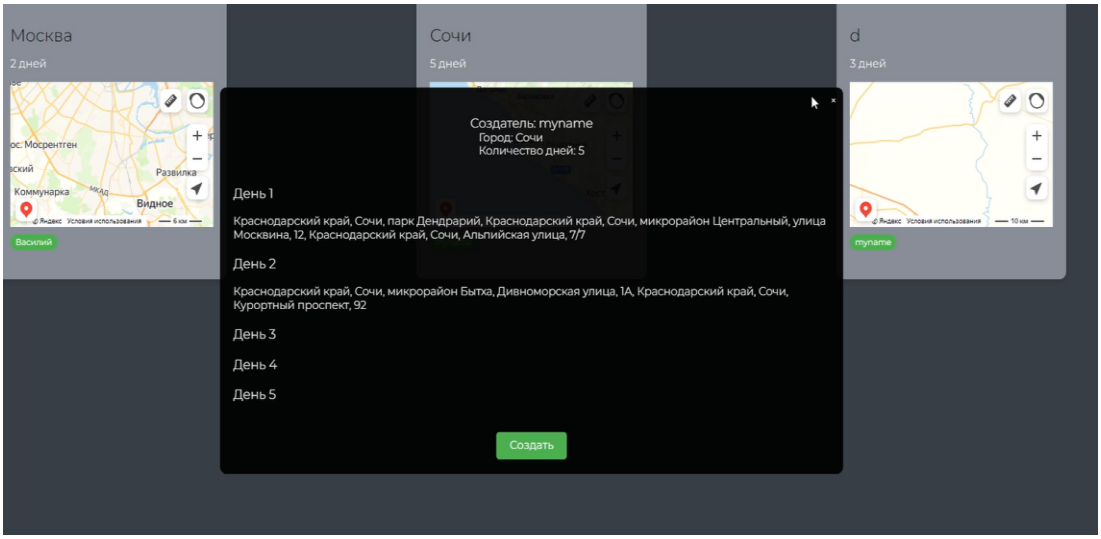

Рисунок 26 — Просмотр активностей поездки

3.4.2 Для авторизованного пользователя

После авторизации пользователю становится доступна возможность создавать и планировать путешествия (Рисунок 27). Для этого необходимо нажать кнопку «создать» в хедере, либо в личном кабинете.

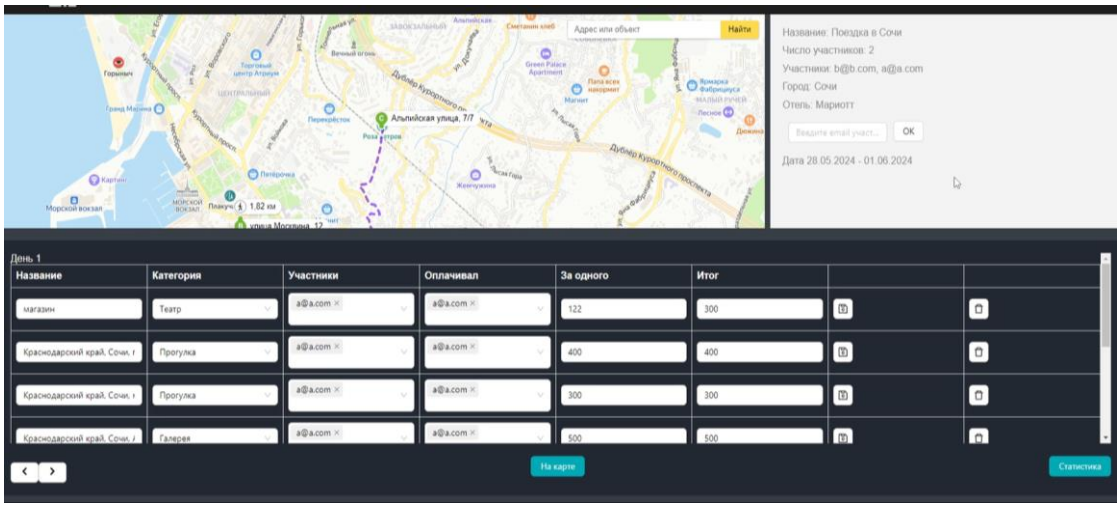

Рисунок 27 — Планирование путешествия

Список созданных поездок, как и информацию о себе, введенную при регистрации, пользователь может посмотреть в своем личном кабинете (Рисунок 28).

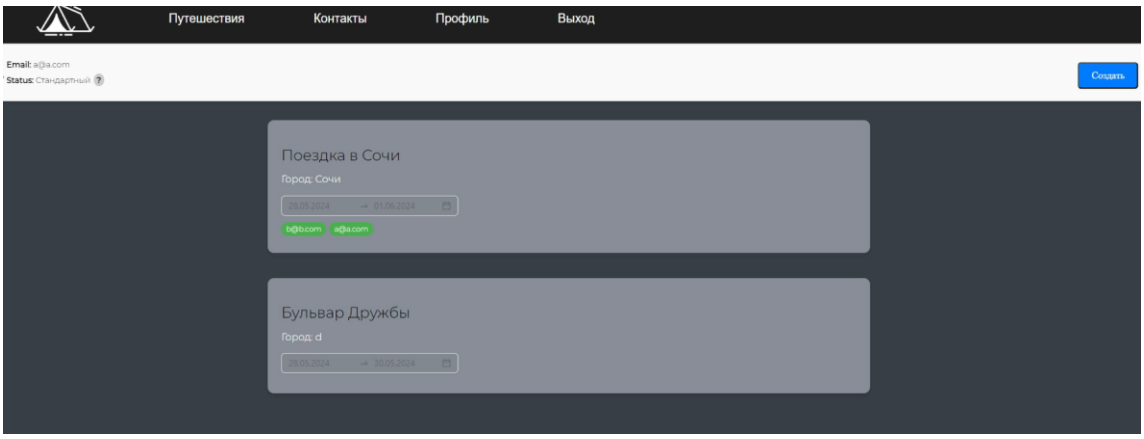

Рисунок 28 — Личный кабинет пользователя со списком созданных путешествий

Также авторизованному пользователю доступна функция просмотра статистики. Для этого на странице создания путешествий необходимо нажать на кнопку «Статистика», после нажатия на которую показывается или скрывается форма с диаграммами личной статистикой трат и общими тратами участников (Рисунок 29).

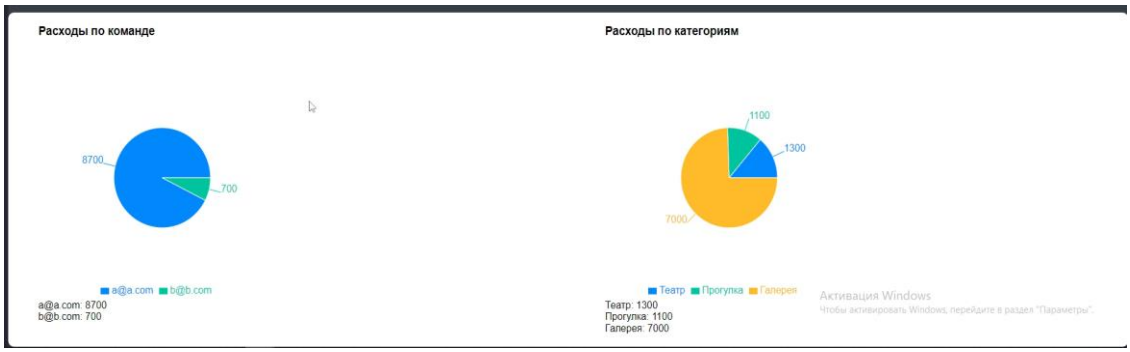

Рисунок 29 — Просмотр статистики трат в виде диаграмм

3.4.3 Для администратора

Для администратора доступна страница администрирования, где, он может управлять статусом и удалять пользователей и удалять поездки (Рисунок 30).

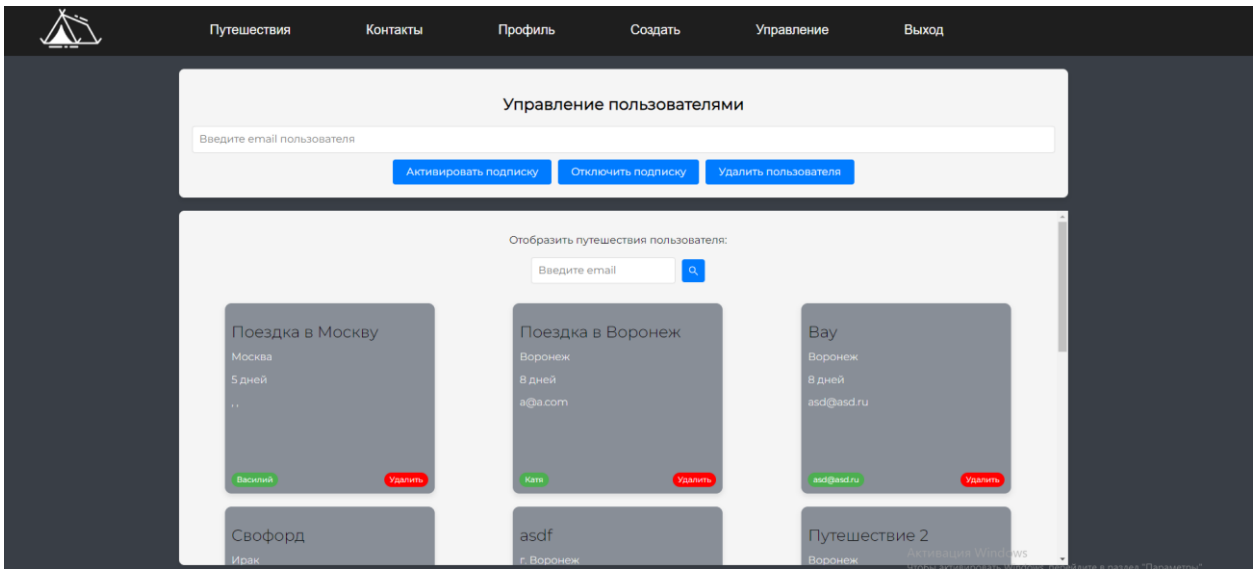

Рисунок 30 — Главная страница с панелью для администратора

4 Тестирование

4.1 Дымовое тестирование

Дымовое тестирование — это тип тестирования, при котором производится кратковременная проверка основной функциональности системы, чтобы убедиться в отсутствии серьезных проблем, ошибок или сбоев.

Такой тип тестирования полезен, чтобы обнаружить крупные проблемы в функциональности системы до того, как будут проведены более подробные и насыщенные тесты. Если в процессе дымового тестирования найдены ошибка или проблемы, то это может стать причиной для проведения дополнительных и более детальных тестов, чтобы устранить недочеты и проблемы в работе программы.

В ходе дымового тестирования выполняются базовые операции или сценарии, которые предполагаются как наиболее важные и часто используемые пользователем. Далее представлены результаты дымового тестирования для основных сценариев неавторизованного, авторизованного пользователей и администратора (Таблица 2-4).

Таблица 2 — Результаты дымового тестирования для неавторизованного пользователя

Тестовый сценарий	**Результат теста**
Регистрация	Пройден
Авторизация	Пройден
Просмотр опубликованных поездок	Пройден
Просмотр активностей выбранной поездки	Пройден

Таблица 3 — Результаты дымового тестирования для авторизованного пользователя

Тестовый сценарий	**Результат теста**
Просмотр опубликованных поездок	Пройден
Просмотр активностей выбранной поездки	Пройден
Создание путешествия	Пройден
Добавление и удаление активности	Пройден
Просмотр карты с отображением маршрута	Пройден
Добавление участников поездки	Пройден
Добавление и удаление траты	Пройден
Просмотр статистики	Пройден
Просмотр личного кабинета	Пройден

Таблица 4 — Результаты дымового тестирования для администратора

Тестовый сценарий	**Результат теста**
Авторизация	Пройден
Изменение статуса пользователя	Пройден
Удаление поездки	Пройден
Удаление пользователя	Пройден

4.2 Тестирование пользовательского интерфейса

Тестирование пользовательского интерфейса (GUI-тестирование) — это процесс тестирования элементов управления в приложении, который помогает убедиться, что интерфейс соответствует ожидаемой функциональности и корректно отображает интерфейс. Включает проверку различных функций и элементов, таких как окна, диалоговые окна, кнопки, переключатели, выпадающие списки, формы.

Задача проведения GUI-тестов — убедиться, что в функциях пользовательского интерфейса отсутствуют дефекты. В Таблице 5 представлена часть результатов тестирования пользовательского интерфейса. В ней отражены тестовые сценарии для неавторизованного пользователя.

Таблица 5 — Результаты GUI-тестирования

Тестовый сценарий	**Ожидаемый результат**	**Статус теста**
Нажатие на кнопку «Путешествия»	Переход на страницу опубликованных путешествий	Пройден
Нажатие на логотип	Переход на главную страницу	Пройден
Нажатие на кнопку «Создать путешествие»	Переход на страницу авторизации	Пройден
Нажатие на кнопку «Войти»	Переход на страницу авторизации	Пройден
Нажатие на кнопку «Контакты»	Переход на страницу для связи с администратором	Пройден
Нажатие на карточку опубликованного путешествия	Отображение формы с активностями для выбранной поездки	Пройден
Нажатие на кнопку «Войти»	Переход на главную страницу	Пройден
Нажатие на ссылку «Зарегистрироваться»	Переход на страницу регистрации	Пройден
Нажатие на ссылку «Войти»	Переход на страницу авторизации	Пройден

Заключение

В ходе выполнения курсовой работы были выполнены все поставленные задачи. Было разработано веб-приложение с возможностью планирования и учета путешествий, просмотра статистики трат, оформления подписки.

В начале разработки был проведен анализ предметной области, определены основные требования к разрабатываемой системе, определены основные сценарии веб-приложения. По результатам разработки проводился ряд тестов с целью проверки работоспособности системы.

Были реализованы следующие сценарии для пользователя:

— ознакомиться с путешествиями других людей;

— создать и спланировать совместное путешествие, с возможностью отследить маршрут на карте и добавить участников;

— посмотреть статистику трат за путешествие;

— посмотреть свои созданные путешествия и личную информацию.

И следующие для администратора:

— удалять путешествия пользователей;

— управлять статусом пользователей;

— удалить пользователя.

Таким образом, итоги разработки, проверенные в ходе тестирования, позволяют достигнуть поставленных заказчиком целей и решают сформулированные в начале разработки задачи.

Список использованной литературы

1. Что такое ASP.NET Core [Электронный ресурс]. — Режим доступа: https://metanit.com/sharp/aspnet6/1.1.php.— Заглавие с экрана. — (Дата обращения: 14.03.2024).

2. React – JavaScript фреймворк для создания пользовательских интерфейсов [Электронный ресурс]. — Режим доступа: https://ru.react.js.org/?ref=dtf.ru.— Заглавие с экрана. — (Дата обращения: 16.03.2024).

3. Полное руководство по 14 типам диаграмм UML [Электронный ресурс]. — Режим доступа: https://www.cybermedian.com/ru/a-comprehensive-guide-to-14-types-of-uml-diagram/. — Заглавие с экрана. — (Дата обращения: 18.03.2024).

4. Диаграммы сотрудничества [Электронный ресурс]. — Режим доступа: https://helpiks.org/9-53448.html. — Заглавие с экрана. — (Дата обращения: 19.03.2024).

5. Тестирование API с помощью Swagger: особенности и преимущества [Электронный ресурс]. — Режим доступа: https://blog.ithillel.ua/ru/articles/apitesting-with-swagger. — Заглавие с экрана. — (Дата обращения: 20.05.2024).