

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий
Кафедра Информационные системы и технологии
Специальность 1–40 01 01 Программное обеспечение информационных техноло-
гий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Web-приложение «турагентство»

Выполнил студент Дмитрук Илья Игоревич
(Ф.И.О.)

Руководитель проекта преп.–ст. Некрасова А.П.
(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доцент Смелов В.В.
(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовая работа защищена с оценкой _____

Минск 2024

Содержание

Введение	6
1 Постановка задачи и обзор аналогичных решений	7
1.1 Постановка задачи.....	7
1.2 Обзор аналогичных решений.....	7
1.2.1 Веб-приложение «People Travel»	7
1.2.2 Веб-приложение «Coral Travel»	8
1.2.3 Веб-приложение «Tez Tour».....	9
1.3 Выводы	9
2 Проектирование веб–приложения.....	10
2.1 Диаграмма вариантов использования.....	10
2.2 Проектирование базы данных.....	12
2.3 Архитектура веб–приложения.....	20
2.4 Выводы.....	21
3 Реализация веб–приложения	22
3.1 Программная платформа ASP.NET Core.....	22
3.2 СУБД MariaDB	22
3.3 ORM Entity Framework Core	22
3.4 Библиотеки и фреймворки	23
3.5 Структура серверной части.....	23
3.6 Реализация функциональности.....	25
3.6.1 Реализация метода входа	25
3.6.2 Реализация метода регистрации.....	26
3.6.3 Реализация метода авторизации.....	26
3.6.4 Реализация метода получения списка туров.....	26
3.6.5 Реализация метода получения списка отфильтрованных туров.....	26
3.6.6 Реализация метода получения типов туров	26
3.6.7 Реализация метода получения характеристик для фильтрации туров...	26
3.6.8 Реализация метода получения характеристик для редактирования туров	26
3.6.9 Реализация метода получения типов питания	27
3.6.10 Реализация метода получения тура	27
3.6.11 Реализация метода получения пунктов отправлений	27
3.6.12 Реализация метода получения городов отправлений	27
3.6.13 Реализация метода получения типов транспорта.....	27
3.6.14 Реализация метода получения отзывов к туру	27
3.6.15 Реализация метода получения регионов	27
3.6.16 Реализация метода получения стран	28
3.6.17 Реализация метода получения городов	28
3.6.18 Реализация метода получения отелей	28
3.6.19 Реализация метода получения направлений	28
3.6.20 Реализация метода получения списка туров для редактирования	28
3.6.21 Реализация метода получения тура для редактирования	28
3.6.22 Реализация метода добавления тура.....	29

3.6.23 Реализация метода редактирования тура	29
3.6.24 Реализация метода удаления тура.....	29
3.6.25 Реализация метода оставления отзыва	29
3.6.26 Реализация метода удаления отзыва.....	29
3.6.27 Реализация метода бронирования тура	29
3.6.28 Реализация метода получения броней.....	29
3.6.29 Реализация метода получения броней для менеджера	30
3.6.30 Реализация метода подтверждения брони	30
3.6.31 Реализация метода удаления брони	30
3.6.32 Реализация метода блокировки/разблокировки пользователя.....	30
3.6.33 Реализация метода удаления пользователя.....	30
3.6.34 Реализация метода смены роли пользователя	30
3.6.35 Реализация метода смены получения пользователей	30
3.7 Структура клиентской части.....	30
3.8 Выводы.....	33
4 Тестирование веб–приложения	33
4.1 Функциональное тестирование	33
4.2 Выводы.....	35
5 Руководство пользователя	36
5.1 Регистрация	36
5.2 Авторизация	36
5.3 Просмотр туров	37
5.4 Просмотр тура	37
5.5 Фильтрация туров	38
5.6 Бронирование туров.....	40
5.6 Оставлять отзывы к турам	41
5.8 Добавление туров.....	41
5.9 Редактирование туров.....	43
5.10 Удаление туров	43
5.11 Подтверждении/отмена брони туров	44
5.12 Удаление отзывов к турам	44
5.13 Смена роли пользователям	44
5.14 Блокировка/разблокировка пользователей	45
5.15 Удаление пользователей	46
Заключение	47
Список используемых источников.....	48
Приложение А	49
Приложение Б.....	53
Приложение В	59
Приложение Г.....	61
Приложение Д	67
Приложение Е.....	70

Введение

Проект посвящён разработке веб-приложения, которое позволит пользователям заказывать туры в разные города и страны. Основная идея веб-приложения заключается в предоставлении туров для бронирования, включающих в себя бронь транспорта и отеля в определённом городе на определённую дату. Всё что включено в тур, будет забронировано вместе с ним.

Целью проекта является упрощение организации путешествий за счёт единовременного бронирования всего того, что необходимо для отправки в тур.

Задачи, поставленные для достижения целей, включают:

- постановка задачи и обзор аналогичных решений (раздел 1);
- проектирование веб-приложения (раздел 2);
- реализация веб-приложения (раздел 3);
- тестирование веб-приложения (раздел 4);
- создание руководства пользователя (раздел 5).

Веб-приложение предназначено для различных людей, которые путешествуют, например, любителям отдохнуть на море, любителям посмотреть на культурные и исторические объекты разных стран, любителям увидеть разнообразную природу различных мест, а также людям, которым необходимо куда-то отправиться по работе или по личным причинам.

Серверная часть реализована на платформе ASP.NET Core[1], клиентская часть — с использованием библиотеки React[2]. В качестве СУБД используется MariaDB[3], с подключением через ORM Entity Framework Core[4] Веб-приложение развёрнуто в Docker[5].

Процесс разработки включал обзор существующих решений для формирования оптимального набора функций, проектирование архитектуры приложения, реализацию серверной и клиентской частей, а также тестирование для проверки функциональности и стабильности.

1 Постановка задачи и обзор аналогичных решений

1.1 Постановка задачи

Необходимо разработать веб-приложение, которое поможет людям в организации путешествий. Веб-приложение должно предоставлять пользователям туры, в которых будут включена бронь номера в отеле и транспорта. Пользователи должны иметь возможность забронировать тур. Необходимо предоставить возможность менеджерам в добавлении, удалении, редактировании туров. Так же они должны принимать заявки на брони туров. Необходима модерация, чтобы следить за тем, чтобы отзывы пользователей не были оскорбительными и неприличными, а также следить, чтобы пользователи не спамили бронями. Для этого необходимы администраторы, которые смогут удалять неприемлемые отзывы, а также блокировать и удалять пользователей.

1.2 Обзор аналогичных решений

1.2.1 Веб-приложение «People Travel»

Веб-приложение People Travel[6] предоставляет пользователям широкий спектр возможностей для подбора нужного тура. В первую очередь, подбор тура происходит по странам, которые предоставляются пользователям на главной странице, затем уже пользователи более подробно настраивают себе отображение туров. Есть несколько видов туров. При просмотре отдельных туров, пользователю предоставляется очень подробная информация о нём. Но при этом у дизайна сайта есть проблемы. При заходе на сайт, сразу почти на весь экран отображается реклама. Для взаимодействия с основным функционалом сайта, необходимо прокрутить вниз, и выбрать страну, то есть для получения доступа к основному функционалу, необходимо выполнить несколько действий. Так же интерфейс не единообразный, то есть у разных стран, разные виды туров, и при их выборе, настройки отображения туров будут разными. Интерфейс веб-приложения представлен на рисунке 1.1.

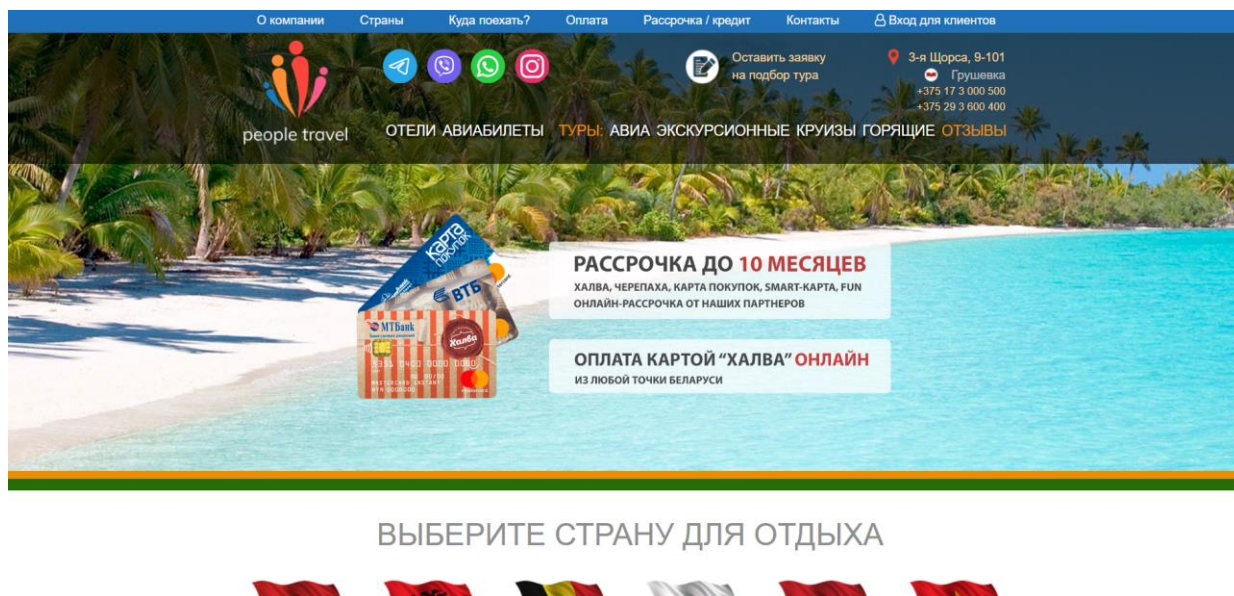


Рисунок 1.1 – Веб-приложение «People Travel»

1.2.2 Веб-приложение «Coral Travel»

Дизайн сайта веб-приложения «Coral Travel»[7] похож на дизайн сайта «People Travel», рассматриваемого ранее, даже фрагменты интерфейса, например, настройки отображаемых туров, идентичные. При входе на сайт, нам сразу предоставляется возможность выбрать отображаемые туры по основным критериям. На главной странице так же отображаются самые популярные туры, однако, чтобы до них добраться, необходимо пролистать огромную часть страницы, на которой размещена информация о турагентстве. Отображение информации о конкретном туре идентично отображению на сайте «People Travel». В целом данный сайт напоминает предыдущий, однако с небольшими преимуществами в более удобном расположении элементов интерфейса. Интерфейс веб-приложения представлен на рисунке 1.2

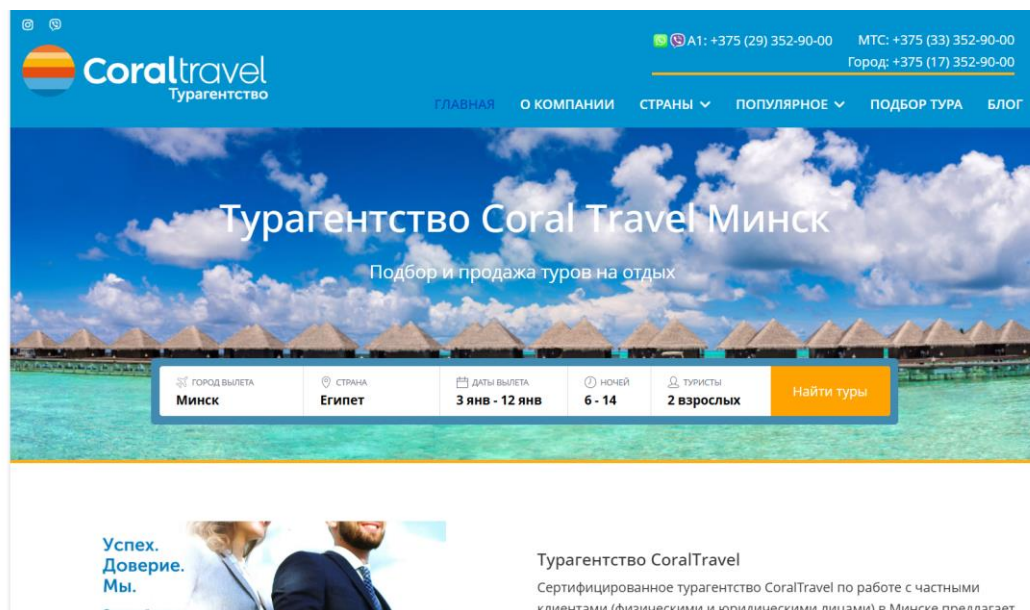


Рисунок 1.2 – Веб-приложение «Coral Travel»

1.2.3 Веб-приложение «Tez Tour»

Веб-приложение «Tez Tour»[8] на главной странице сайта сразу предоставляет возможность удобно настраивать отображение туров по основным критериям. На сайте можно посмотреть информацию о разных аспектах туров, таких как отели, транспорт и т.д. В меню навигации есть разделение на ссылки на страницы, связанные с информацией о турагентстве, и ссылки, связанные с информацией о турах. На главной странице в самом начале отображается реклама разных направлений туров. Хотя реклама занимает большую часть страницы, однако из-за того, что она связана с тематикой сайта, то она хорошо вписывается. Ниже рекламы располагается список самых популярных туров. В веб-приложении поддерживается выбор огромного количества видов туров. При просмотре отдельного тура, можно увидеть довольно подробную информацию о нём, однако она расположена не удобно и представляет с собой набор хаотично разбросанных характеристик. Интерфейс веб-приложения представлен на рисунке 1.3

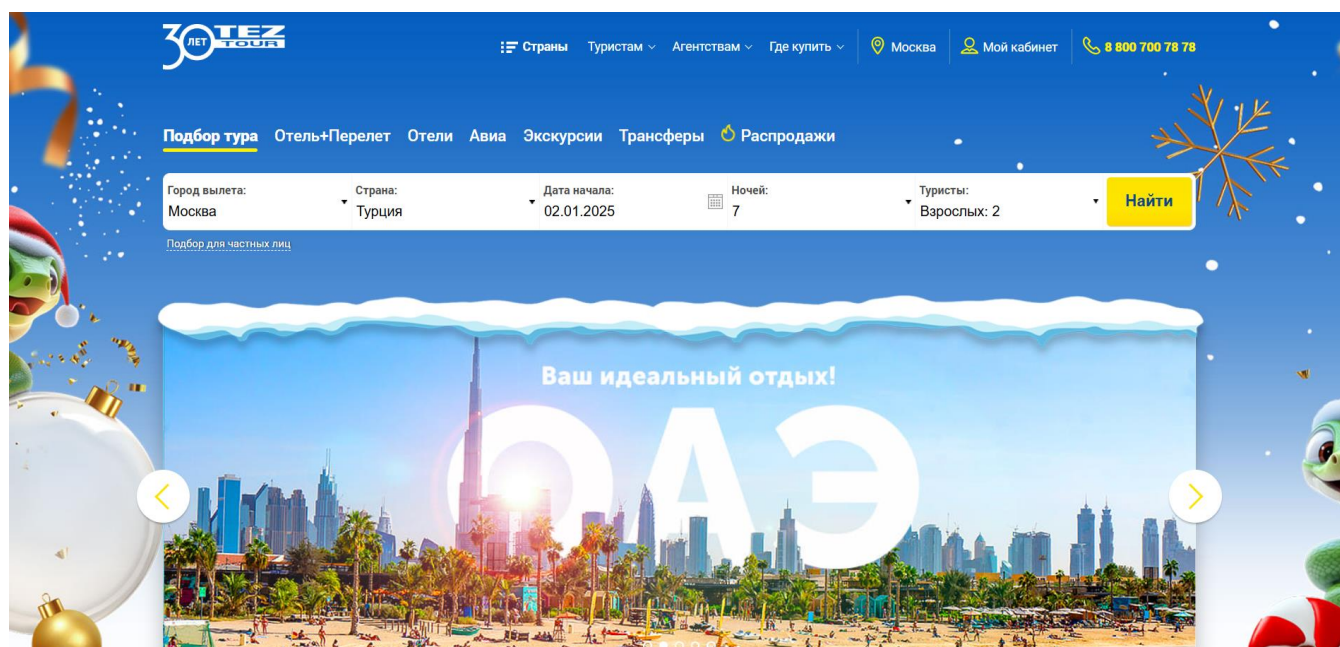


Рисунок 1.3 – Веб-приложение «Tez Tour»

1.3 Выводы

1. Проанализировав аналогичные решения и постановив для себя необходимые задачи, было принято решение реализовать веб-приложение, с поддержкой ролей «Администратор», «Менеджер», «Пользователь» и «Гость». Пользователь, не прошедший авторизацию, имеет роль «Гость» и может только просматривать туры. «Пользователь» имеет возможность бронирования туров, «Менеджер» – добавления, удаления и редактирования туров, а также принятия или отклонения заявок на брони, «Администратор» – удаление отзывов, блокирование и удаление пользователей.

2 Проектирование веб–приложения

2.1 Диаграмма вариантов использования.

Диаграмма вариантов использования иллюстрирует ключевые действия, доступные различным ролям пользователей в веб-приложении. Диаграмма представлена на рисунке 2.1.

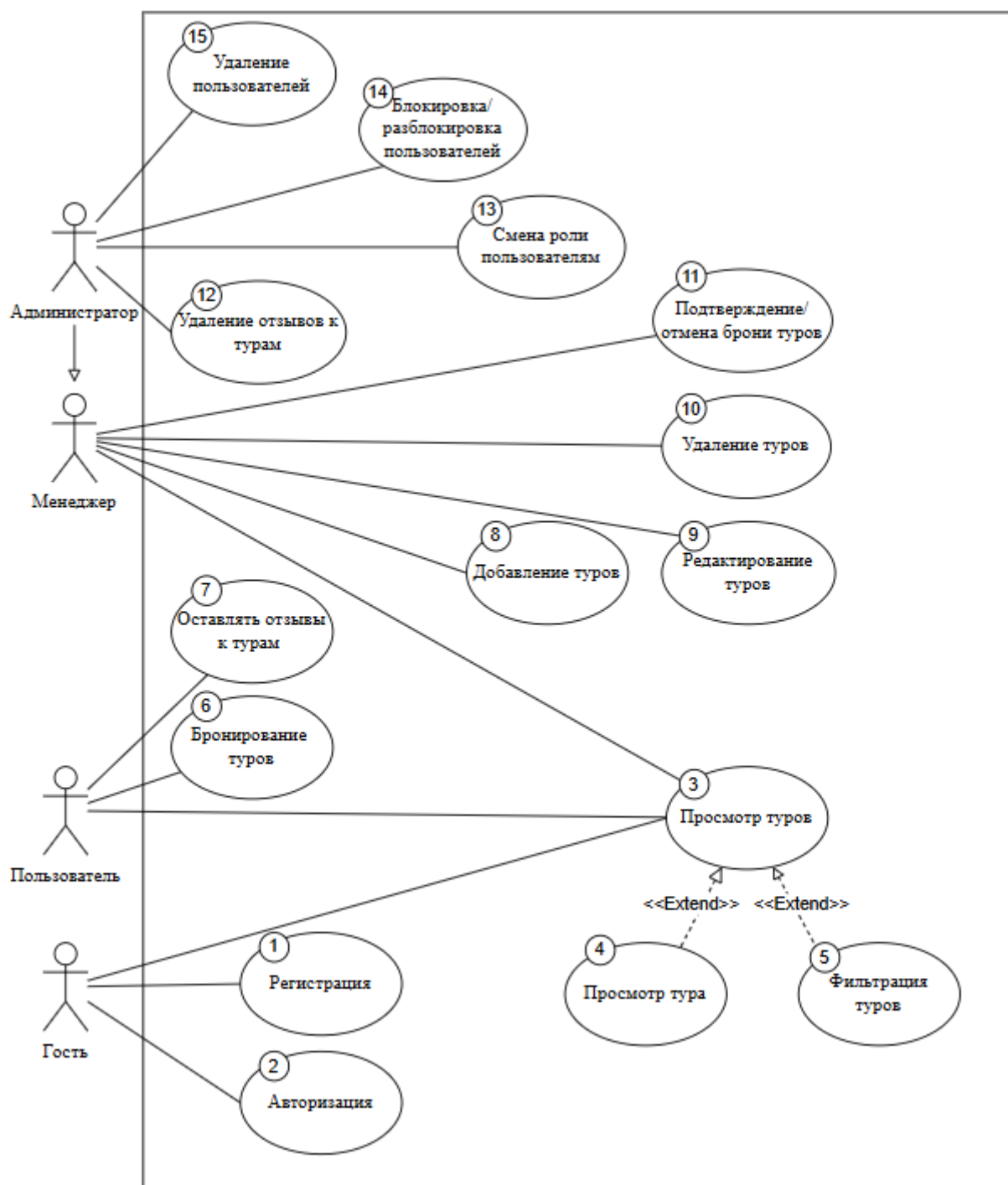


Рисунок 2.1 – Диаграмма вариантов использования

Описание ролей, отображённых на диаграмме, представлена в таблице 2.1.

Таблица 2.1 – Описание ролей

Роль	Описание
Гость	Пользователь, не прошедший авторизацию. Имеет доступ только к просмотру туров.
Пользователь	Может бронировать туры и оставлять к ним отзывы
Менеджер	Может добавлять, удалять, редактировать туры, а также подтверждать брони туров.
Администратор	Может удалять отзывы к турам, а также удалять, блокировать пользователей и изменять им роль. Так же может выполнять функции менеджера.

Описание функций ролей приложения представлены в таблице 2.2.

Таблица 2.2 – Описание функций

№	Функция	Описание	Доступна ролям
1	Регистрация	Создание новой учётной записи, с указанием email, пароля, имени, фамилии и номера телефона.	Гость
2	Аутентификация	Вход в систему с помощью email и пароля.	Гость
3	Просмотр туров	Просмотр списка туров	Гость, Пользователь
4	Просмотр тура	Просмотр подробной информации о выбранном туре	Гость, Пользователь, Менеджер, Администратор
5	Фильтрация туров	Просмотр списка туров с выбранными характеристиками	Гость, Пользователь, Менеджер, Администратор
6	Бронирование туров	Отправление заявки на бронирование выбранного тура	Пользователь
7	Оставлять отзывы к турам	Оставление отзыва к выбранному туру	Пользователь
8	Добавление туров	Создание новых туров	Менеджер, Администратор
9	Редактирование туров	Изменение данных существующих туров.	Менеджер, Администратор
10	Удаление туров	Удаление существующих туров	Менеджер, Администратор
11	Подтверждение/отмена брони туров	Подтверждение заявки на бронь тура, отправленного от пользователя, либо же её отмена.	Менеджер, Администратор
12	Удаление отзывов к турам	Удаление у туров отзывов, нарушающие правила платформы	Администратор

Продолжение таблицы 2.2

13	Смена роли пользователям	Изменение роли у пользователя, что приводит к смене его прав.	Администратор
14	Блокировка/разблокировка пользователей	Блокировка у пользователя всех возможностей, кроме просмотра туров. А также разблокировка, то есть возвращение ему прав.	Администратор
15	Удаление пользователей	Удаление существующих учётных записей пользователей	Администратор

2.2 Проектирование базы данных.

Диаграмма базы данных иллюстрирует структуру базы данных и отношений между таблицами, которые хранятся в этой базе данных. Диаграмма логической схемы базы данных представлена на рисунке 2.2.

Для веб–приложения была спроектирована база данных, включающая 18 таблиц. В таблице 2.3 показано описание таблиц базы данных.

Таблица 2.3 – Описание таблиц базы данных

Название таблицы	Описание таблицы
roles	Хранит данные о ролях пользователей.
users	Хранит данные о пользователях.
tours	Хранит данные о турах.
routes	Хранит данные о маршрутах.
bookings	Хранит данные о бронях.
tourtypes	Хранит данные о типах туров.
characteristictypes	Хранит данные о типах характеристик.
characteristics	Хранит данные о характеристиках
charcteristics_tourtypes	Описывает связь между характеристиками и типами туров.
descriptions	Описывает связь между характеристиками и турами, а также хранит значение характеристики.
nutritionetypes	Хранит данные о типах питания.
hotels	Хранит данные о отелях.
cities	Хранит данные о городах.
countries	Хранит данные о странах.
regions	Хранит данные о регионах мира.
departmentdepartures	Хранит данные о пунктах отправления.
reviews	Хранит данные о отзывах.
transporttypes	Хранит данные о типах транспортов.

В таблицах 2.4 – 2.21 показаны описания полей таблицы базы данных.

Таблица 2.4– Описание структуры таблицы «roles»

Поле	Тип данных	Ограничения	Описание
ID	TINYINT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор роли
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название роли

Таблица 2.5 – Описание структуры таблицы «users»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор пользователя
Email	VARCHAR(255)	UNIQUE, NOT NULL	Адрес электронной почты пользователя
Password	VARCHAR(255)	NOT NULL	Пароль пользователя
RoleId	TINYINT	DEFAULT 1, FOREIGN KEY	Идентификатор роли пользователя
BlockedStatus	BOOLEAN	DEFAULT 0	Статус блокировки
Name	VARCHAR(255)		Имя пользователя

Surname	VARCHAR(255)		Фамилия пользователя
PhoneNumber	VARCHAR(255)		Номер телефона пользователя
Photo	LOB		Фотография пользователя

Таблица 2.6 – Описание структуры таблицы «tourtypes»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор типа тура
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название типа тура
Image	LOB		Изображение типа тура

Таблица 2.7 – Описание структуры таблицы «characteristictypes»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор типа характеристики
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название типа характеристики

Таблица 2.8 – Описание структуры таблицы «characteristics»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор характеристики
CharacteristicTypeID	INT	FOREIGN KEY	Идентификатор типа характеристики
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название характеристики

Таблица 2.9 – Описание структуры таблицы «characteristics_tourtypes»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор связи
CharacteristicID	INT	FOREIGN KEY	Идентификатор характеристики
TourTypeID	INT	FOREIGN KEY	Идентификатор типа тура

Таблица 2.10 – Описание структуры таблицы «regions»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор региона
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название региона
Image	LONGBLOB		Изображение региона

Таблица 2.11 – Описание структуры таблицы «countries»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор страны
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название страны
Flag	LONGBLOB		Флаг страны
RegionId	INT	FOREIGN KEY	Идентификатор региона

Таблица 2.12 – Описание структуры таблицы «cities»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор города
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название города
CountryId	INT	FOREIGN KEY	Идентификатор страны

Таблица 2.13 – Описание структуры таблицы «departmentdepartures»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор пункта отправления
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название пункта отправления
CityId	INT	FOREIGN KEY	Идентификатор города

Таблица 2.14 – Описание структуры таблицы «transporttypes»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор типа транспорта
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название типа транспорта

Таблица 2.15 – Описание структуры таблицы «hotels»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор отеля
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название отеля
StarsNumber	INT	NOT NULL	Количество звезд у отеля
CityId	INT	FOREIGN KEY	Идентификатор города

Таблица 2.16 – Описание структуры таблицы «nutritiontypes»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор типа питания
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название типа питания

Таблица 2.17 – Описание структуры таблицы «tours»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор тура
Name	VARCHAR(255)	UNIQUE, NOT NULL	Название тура
MainDescription	VARCHAR(255)		Основное описание тура
TourTypeId	INT	FOREIGN KEY	Идентификатор типа тура
NutritionTypeId	INT	FOREIGN KEY	Идентификатор типа питания
HotelId	INT	FOREIGN KEY	Идентификатор отеля
Photo	LONGBLOB		Изображение тура

Таблица 2.18 – Описание структуры таблицы «descriptions»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор описания
Value	BOOLEAN	DEFAULT 0, NOT NULL	Значение характеристики
CharacteristicID	INT	FOREIGN KEY	Идентификатор характеристики
TourID	INT	FOREIGN KEY	Идентификатор тура

Таблица 2.19 – Описание структуры таблицы «reviews»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор отзыва
UserId	INT	FOREIGN KEY	Идентификатор пользователя
TourId	INT	FOREIGN KEY	Идентификатор тура
ReviewText	VARCHAR(255)		Текст отзыва

Таблица 2.20 – Описание структуры таблицы «routes»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор маршрута
LandingDateOfDeparture	DATE		Дата отправления
LandingTimeOfDeparture	TIME		Время отправления
ArrivalDateOfDeparture	DATE		Дата прибытия
ArrivalTimeOfDeparture	TIME		Время прибытия
LandingDateOfReturn	DATE		Дата отправления обратно
LandingTimeOfReturn	TIME		Время отправления обратно
ArrivalDateOfReturn	DATE		Дата возвращения
ArrivalTimeOfReturn	TIME		Время возвращения
Price	INT		Стоимость маршрута
SeatsNumber	INT		Количество мест
DepartmentDepartureId	INT	FOREIGN KEY	Идентификатор пункта отправления
TransportTypeId	INT	FOREIGN KEY	Идентификатор типа транспорта
TourId	INT	FOREIGN KEY	Идентификатор тура

Таблица 2.21 – Описание структуры таблицы «bookings»

Поле	Тип данных	Ограничения	Описание
ID	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор бронирования
OrderSeatsNumber	INT		Количество забронированных мест
UserId	INT	FOREIGN KEY	Идентификатор пользователя
RouteId	INT	FOREIGN KEY	Идентификатор маршрута
Status	BOOLEAN	DEFAULT 0	Статус бронирования

Описание связей таблиц базы данных представлено в таблице 2.22

Таблица 2.22 – Описание связей таблиц

Таблица 1	Поле (PK)	Таблица 2	Поле (FK)	Тип связи
roles	ID	users	RoleId	Один–ко–многим
users	ID	bookings	UserId	Один–ко–многим
tourtypes	ID	characteristics_tourtypes	TourTypeId	Один–ко–многим
characteristics	ID	characteristics_tourtypes	CharacteristicId	Один–ко–многим
characteristicstypes	ID	characteristics	CharacteristicId	Один–ко–многим
characteristics	ID	descriptions	CharacteristicId	Один–ко–многим
tours	ID	routes	TourId	Один–ко–многим
tours	ID	descriptions	TourId	Один–ко–многим
nutritiontypes	ID	tours	NutritionTypeId	Один–ко–многим
tourtypes	ID	tours	TourTypeId	Один–ко–многим
hotels	ID	tours	HotelId	Один–ко–многим
cities	ID	hotels	CityId	Один–ко–многим
cities	ID	departmentdepartures	CityId	Один–ко–многим
countries	ID	cities	CountryId	Один–ко–многим
regions	ID	countries	RegionId	Один–ко–многим
routes	ID	bookings	RouteId	Один–ко–многим
departmentdepartures	ID	routes	DepartmentDepartureId	Один–ко–многим
transporttypes	ID	routes	TransportTypeId	Один–ко–многим
tours	ID	reviews	TourId	Один–ко–многим

2.3 Архитектура веб-приложения

Архитектура приложения представлена на рисунке 2.3.

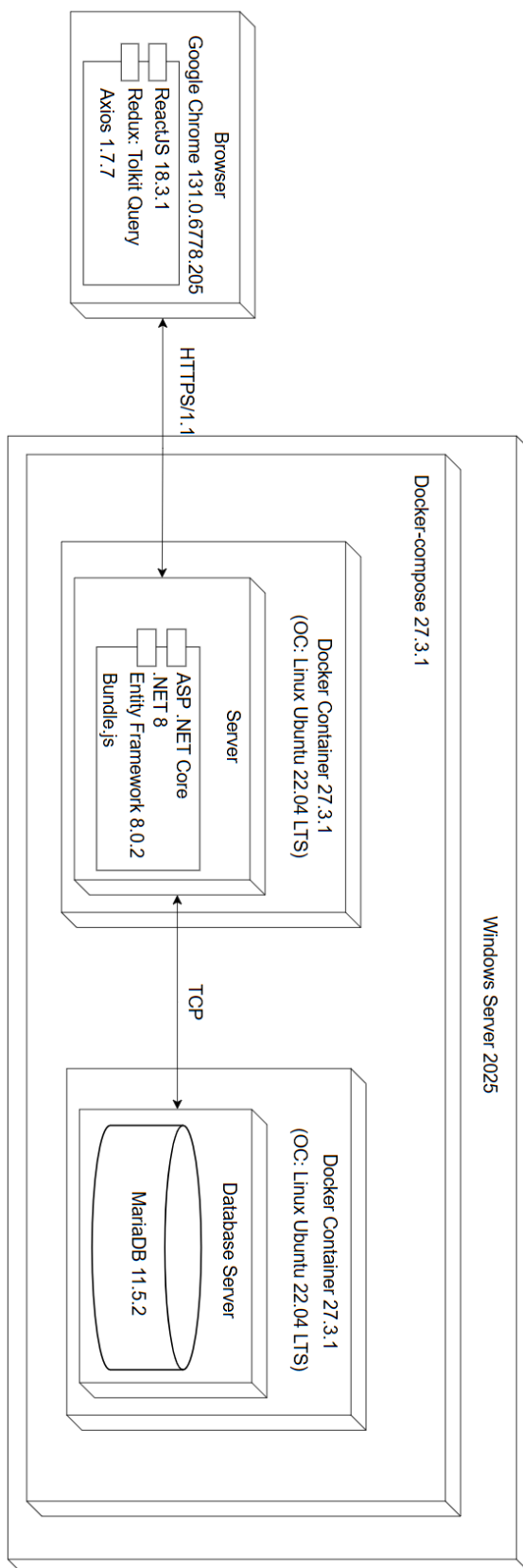


Рисунок 2.3 – Архитектура приложения

Система представляет собой архитектуру, включающую серверную и клиентскую части, а также базу данных, развёрнутые в контейнерах Docker на операционных системах Linux Ubuntu 22.04 LTS[9]. Серверная часть обрабатывает запросы HTTPS/1.1[10]. В качестве СУБД используется MariaDB. Важным компонентом системы является Docker Compose[11]. Этот инструмент используется для управления несколькими контейнерами Docker.

2.4 Выводы

1. Web-приложение поддерживает четыре роли пользователей: «Гость», «Администратор», «Менеджер», «Пользователь». Гостям предоставляется возможность только простора туров. Что бы получить доступ к остальному функционалу, они должны зарегистрироваться или пройти авторизацию. Администратор может удалять, блокировать учетные записи и изменять пользователям роли. Также он может выполнять функции Менеджера. Менеджер может добавлять, удалять, редактировать туры и принимать заявки на брони. Пользователь может бронировать тур и оставлять на него отзыв.

2. Логическая схема базы данных веб-приложения включает 18 таблиц и предназначена для хранения данных о ролях пользователей, пользователях, турах, маршрутах, бронях, типах туров, типах характеристик, характеристиках, типах питания, отелях, городах, странах, регионах, пунктах отправления, отзывах, типах транспортных.

3. Веб-приложение основано на монолитной архитектуре.

3 Реализация веб–приложения

3.1 Программная платформа ASP.NET Core

Для реализации серверной части веб-приложения используется платформа ASP.NET Core. Эта платформа является кроссплатформенным и высокопроизводительным фреймворком для создания веб-приложений, который поддерживает разработку REST API.

3.2 СУБД MariaDB

Для проекта была выбрана СУБД MariaDB — мощную реляционную СУБД, известную надежностью, расширяемостью и активным сообществом. Она обеспечивает хранение данных, поддержку транзакций и эффективную обработку сложных запросов, что делает её отличным выбором для систем с большими объёмами данных и сложной бизнес-логикой.

3.3 ORM Entity Framework Core

Для управления данными используется ORM Entity Framework Core, которая обеспечивает удобное взаимодействие с базой данных. Данная ORM позволяет взаимодействовать с объектами базы данных, как с программными объектами. Код контекста БД предоставлен в приложении А. Сопоставление моделей и таблиц базы данных представлено в таблице 3.1.

Таблица 3.1 – Описание связей таблиц

Таблицы	Модель
roles	Role
users	User
tours	Tour
routes	Route
bookings	Booking
tourtypes	TourType
characteristictypes	CharacteristicType
characteristics	Characteristic
descriptions	Description
nutritionetypes	NutritioneType
hotels	Hotel
cities	City
countries	Country
regions	Region
departmentdepartures	DepartmentDeparture
reviews	Review
transporttypes	TransportType

3.4 Библиотеки и фреймворки

В процессе разработки серверной части web-приложения были использованы программные библиотеки, все библиотеки указаны в приложении Б, основные представлены в таблице 3.4.

Таблица 3.2 – Программные библиотеки серверной части

Библиотека	Версия	Назначение
Entity Framework Core	V8.0.2	Используется для взаимодействия с базой данных по принципу ORM (Object-Relational Mapping).
JWT Bearer Authentication [12]	v8.0.11	Используется для реализации аутентификации на основе токенов JSON Web Token (JWT).

В процессе разработки клиентской части web-приложения были задействованы программные библиотеки, представленные в таблице 3.5.

Таблица 3.3 – Программные библиотеки клиентской части

Библиотека	Версия	Назначение
axios [13]	v1.7.7	Библиотека для выполнения HTTP-запросов.
react [14]	v18.3.1	Библиотека для создания пользовательских интерфейсов, используется для построения компонентов и управления состоянием UI
react-router-dom [15]	v6.26.2	Библиотека для маршрутизации в React-приложениях.

3.5 Структура серверной части

В таблице 3.4 приведён список директорий проекта разработки серверной части веб-приложения и назначение файлов, хранящихся в этих директориях.

Директория	Назначение
Configurations	Содержит файлы с конфигурациями, например, для настройки аутентификации с помощью JWT токенов.
Controllers	Содержит контроллеры, которые обрабатывают HTTP-запросы и возвращают ответы клиенту.
DB	Содержит контекст базы данных для Entity Framework.
Models	Содержит все виды моделей: DTO, данные из форм и модели Entity Framework.
Services	Содержит файлы, которые реализуют бизнес-логику для вспомогательных действий в контролерах.

В таблице 3.5 описаны все маршруты веб-сервера.

Таблица 3.5 – Описание маршрутов

URL	Метод	Кон-троллер	№ функ-ций	Описание
/auth/register	POST	Auth	1	Регистрация нового пользователя.
/auth/login	POST	Auth	2	Вход пользователя с логином и паролем
/auth/auth	GET	Auth	2	Авторизация пользователя с помощью токена
/booking/add	POST	Booking	6	Добавление новой брони тура
/booking/bookings	GET	Booking	6	Получение броней пользователя
/booking/bookings_for_manager	GET	Booking	11	Получение всех броней для их редактирования
/booking/delete	PATCH	Booking	6, 11	Удаление брони
/booking/confirm	PUT	Booking	11	Подтверждение брони
/direction/regions	GET	Directions	5, 8, 9	Получение регионов
/direction/countries	GET	Directions	5, 8, 9	Получение стран
/direction/cities	GET	Directions	5, 8, 9	Получение городов
/direction/get	GET	Directions	5, 8, 9	Получение отеля, города, страны, региона, соответствующих друг другу
/review/reviews	GET	Review	4, 9	Получение отзывов к туру
/review/add	POST	Review	7	Добавление отзыва к туру
/review/delete	DELETE	Review	12	Удаление отзыва у тура
/route/department_departures	GET	Route	8, 9	Получение пунктов отправлений
/route/departure_cities	GET	Route	5	Получение городов пунктов отправлений
/route/transport_types	GET	Route	5, 8, 9	Получений типов транспортов
/tour/tour_types	GET	Tour	5, 8, 9	Получение типов туров
/tour/characteristics	GET	Tour	8, 9	Получение характеристик туров с определённым типом туров для редактора тура

/tour/ characteristics_to_filter	GET	Tour	5	Получение характеристик туров с определённым типом туров для фильтра туров
/tour/nutrition_types	GET	Tour	5, 8, 9	Получение типов питания
/tour/tour_to_edit	GET	Tour	9	Получение данных тура для редактора
/tour/tours	GET	Tour	3	Получение туров
/tour/add	POST	Tour	8	Добавление нового тура
/tour/edit	PUT	Tour	9	Редактирование тура
/tour/delete	DELETE	Tour	10	Удаление тура
/user/block	PATCH	User	14	Блокировка/разблокировка пользователя
/user/delete	DELETE	User	15	Удаление пользователя
/user/change_role	PATCH	User	13	Изменение роли пользователя
/user/users	GET	User	13, 14, 15	Получение пользователей

Для передачи большинства запросов и ответов используется формат JSON[16]. Он позволяет эффективно передавать структурированные данные, такие как объекты, массивы, строки, числа и другие базовые типы.

В случае передачи данных с форм использовался формат «multipart/form-data»[17]. Он позволяет передавать файлы в составе HTTP-запроса наряду с другими данными, такими как текстовые поля.

3.6 Реализация функциональности

Листинг кода всех методов, перечисленных в данном разделе, представлен в приложении В.

3.6.1 Реализация метода входа

Метод «Login» обрабатывает процесс входа пользователя в систему. Он в начале получает пользователя из базы данных с переданным в метод email, затем сравнивает хеш пароля пользователя с хешом пароля, переданного в метод. Если данный пользователь существует и пароли совпадают, то метод возвращает токен, если нет, то ответ с 401 статусом.

3.6.2 Реализация метода регистрации

Метод «Register» отвечает за регистрацию нового пользователя. Сначала он проверяет, существует ли пользователь с переданным в метод email, если существует, то возвращает ответ с 409 статусом. Если нет, то хеширует переданный в метод. Затем возвращает токен.

3.6.3 Реализация метода авторизации

Метод «Auth» предназначен для предоставления пользователю его учётных данных токеном. У метода установлен атрибут Authorize, который проверяет корректность полученного токена. Затем метод получает из базы данных данные пользователя по его email, полученного из токена и возвращает эти данные. Если пользователь не будет найден, или токен не корректен, то возвращает ответ с 401 статусом.

3.6.4 Реализация метода получения списка туров

Метод «GetTours» возвращает список всех туров. Он получает из базы данных все туры с их маршрутами, у которых есть ещё места и дата отправления больше текущей, фильтрует их по количеству оставшихся мест и возвращает их.

3.6.5 Реализация метода получения списка отфильтрованных туров

Метод «GetFiltredTours» возвращает список отфильтрованных туров. Он получает из базы данных все туры с маршрутами, затем выбирает те туры, данные которых соответствуют данным фильтра, переданных в метод, и возвращает эти туры.

3.6.6 Реализация метода получения типов туров

Метод «GetTourTypes» получает из базы данных список типов туров, необходимых для фильтрации или редактирования туров, и возвращает его.

3.6.7 Реализация метода получения характеристик для фильтрации туров

Метод «GetCharacteristicsToFilter» возвращает характеристики, соответствующих типу тура, для фильтрации туров. Метод получает из базы данных характеристики, соответствующих типу тура, id которого предаётся в метод. По этим характеристикам пользователи могут фильтровать список туров.

3.6.8 Реализация метода получения характеристик для редактирования туров

Метод «GetCharacteristics» возвращает характеристики со значениями, соответствующих определённому типу туров, для редактирования тура. Метод получает из базы данных характеристики со значениями, которые соответствуют типу тура

по id, переданного в метод. В значения характеристик устанавливается false, так как вызывается при смене типа тура в редакторе тура, в следствии чего значение характеристик обнуляются.

3.6.9 Реализация метода получения типов питания

Метод «GetNutritionTypes» получает из базы данных список типов питания, необходимых для фильтрации или редактирования туров, и возвращает его.

3.6.10 Реализация метода получения тура

Метод «GetTour» возвращает данные о туре. Метод получает из базы данных тур с определённым id, переданного в метод, и возвращает его со всей информацией.

3.6.11 Реализация метода получения пунктов отправлений

Метод «GetDepartureCities» получает из базы данных список пунктов отправления, необходимых редактирования туров, и возвращает его.

3.6.12 Реализация метода получения городов отправлений

Метод «GetDepartmentDepartures» получает из базы данных список пунктов отправлений, группирует его по городам, необходимых для фильтрации туров, и возвращает его.

3.6.13 Реализация метода получения типов транспорта

Метод «GetTransportTypes» получает из базы данных список типов транспорта, необходимых для фильтрации или редактирования туров, и возвращает его.

3.6.14 Реализация метода получения отзывов к туру

Метод «GetReviews» получает из базы данных отзывы у определённого тура, id которого передаётся в метод, и возвращает их.

3.6.15 Реализация метода получения регионов

Метод «GetRegions» получает из базы данных список регионов мира, необходимых для фильтрации туров или для поиска отеля при редактировании туров, и возвращает его.

3.6.16 Реализация метода получения стран

Метод «GetCountries» получает из базы данных список стран региона, id которого передаётся в метод, необходимых для фильтрации туров и для поиска отеля при редактировании туров, и возвращает его.

3.6.17 Реализация метода получения городов

Метод «GetCities» получает из базы данных список городов страны, id которой передаётся в метод, необходимых для фильтрации туров и для поиска отеля при редактировании туров, и возвращает его.

3.6.18 Реализация метода получения отелей

Метод «GetHotels» получает из базы данных список отелей города, id которого передаётся в метод, необходимых для редактирования туров, и возвращает его.

3.6.19 Реализация метода получения направлений

Метод «GetDirection» возвращает направление, которое включает в себя название региона, страны, города, отеля, id которых передаются в метод. Метод сначала проверяет id отеля, если он не равен нулю или null, то получает из базы данных отель и все географические сущности, соответствующих ему. Если нет, то метод получает из базы данных город по его id, и все его географические сущности. Если же и его id равен нулю или null, то делается то же самое со страной, а затем и с регионом. Таким образом получается объект с названием географических сущностей, которые отобразятся на странице при их выборе в фильтре или при поиске отеля в редакторе туров.

3.6.20 Реализация метода получения списка туров для редактирования

Метод «GetToursToEdit» возвращает список туров, предназначенных для редактирования. Метод получает из базы данных туры, которые соответствуют типу туров, id которого передаётся в метод, и возвращает их. Отличие от метода «GetTours» заключается том, что он не извлекает информацию о туре, связанную с маршрутом, а передаёт информацию именно о туре.

3.6.21 Реализация метода получения тура для редактирования

Метод «GetTourToEdit» возвращает данные тура, предназначенных для редактирования. Метод получает из базы данных тура, id которого передаётся в метод, и возвращает их. При чём, извлекаются из базы данных все маршруты, связанные с турами. Если же id тура не передан или равен нулю, то метод создаёт новый объект тура со свойствами по умолчанию, и возвращает его.

3.6.22 Реализация метода добавления тура

Метод «AddTour» создаёт новый тур. Метод принимает данные тура и сохраняет их в базу данных.

3.6.23 Реализация метода редактирования тура

Метод «EditTour» изменяет существующий тур. Метод принимает данные тура и изменяет на их данные у тура в базе данных, id которого соответствуя id переданного в метод тура.

3.6.24 Реализация метода удаления тура

Метод «DeleteTour» удаляет существующий тур. Метод принимает id тура и удаляет тур с соответствующим id в базе данных вместе со всеми сущностями, соединённых с туром.

3.6.25 Реализация метода оставления отзыва

Метод «AddReview» создаёт новый отзыв к туру. Метод данные отзыва и добавляет его в базу данных.

3.6.26 Реализация метода удаления отзыва

Метод «DeleteReview» удаляет существующий отзыв к туру. Метод принимает id отзыва и удаляет отзыв с соответствующим id из базы данных.

3.6.27 Реализация метода бронирования тура

Метод «AddBooking» создаёт новую бронь. Метод принимает id пользователя и маршрута, и проверяет, есть ли в базе данных бронь тура по данному маршруту у данного пользователя, если есть, то метод возвращает ответ со статусом 409. Затем проверяет есть ли данный маршрут, если нет, то возвращает ответ со статусом 400, затем проверяет есть ли данный пользователь, если нет, то возвращает ответ со статусом 401. Если все проверки пройдут успешно, то метод получает из базы данных маршрут и отнимает у него от количества мест количество заказанных мест, переданных в метод. Если результат меньше нуля, то возвращает ответ со статусом 409, если нет, то бронь успешно добавляется в базу данных.

3.6.28 Реализация метода получения броней

Метод «GetBookings» возвращает список броней. Метод принимает id пользователя и получает из базы данных брони пользователя и возвращает их.

3.6.29 Реализация метода получения броней для менеджера

Метод «GetBookingsForManager» получает из базы данных все брони, и возвращает их с информацией о пользователе, отправивших заявку на данную бронь.

3.6.30 Реализация метода подтверждения брони

Метод «ConfirmBooking» меняет статус брони на true. Принимает id брони и в базе данных у этой брони меняет статус.

3.6.31 Реализация метода удаления брони

Метод «DeleteBooking» удаляет существующую бронь. Метод принимает id брони и удаляет бронь с соответствующим id в базе данных.

3.6.32 Реализация метода блокировки/разблокировки пользователя

Метод «BlockUser» изменяет статус блокировки пользователя на противоположный. Метод принимает id пользователя и изменяет статус блокировки у соответствующего пользователя.

3.6.33 Реализация метода удаления пользователя

Метод «DeleteUser» удаляет существующего пользователя. Метод принимает id пользователя и удаляет пользователя с соответствующим id в базе данных.

3.6.34 Реализация метода смены роли пользователя

Метод «ChangeRole» изменяет роль у пользователя. Метод изменяет у пользователя с id, переданного в метод, id роли на id, так же переданной в метод.

3.6.35 Реализация метода смены получения пользователей

Метод «GetUsers» получает из базы данных список всех пользователей, и возвращает его.

3.7 Структура клиентской части

Файлы проекта организованы таким образом, чтобы они были сгруппированы в директории по их назначению. Основная логика и элементы пользовательского интерфейса размещены в директории src. В таблице 3.6 приведен список директорий проекта.

Таблица 3.6 – Директории проекта и их назначение

Директория	Назначение
styles	Содержит файлы стилей.
pages	Содержит компоненты страниц.
img	Содержит изображения элементов, используемых на страницах.
context	Содержит глобальные данные, которые могут использоваться во всех компонентах.
components	Содержит компоненты элементов страниц.

При написании клиента было разработано 13 компонентов страниц. Они описаны в таблице 3.7

Таблица 3.7 – Описание компонентов страниц

Название	Описание	Назначение
auth	Страница авторизации	Предоставляет пользователю возможность войти или зарегистрироваться
company	Страница с информацией о турагентстве	Предоставляет пользователям информации о турагентстве.
error	Страница ошибок	Предоставляет пользователю информацию об ошибке.
managerBookings	Страница броней для менеджера	Предоставляет список броней, которые можно подтверждать.
payment	Страница с информацией о оплате	Предоставляет пользователям информации о оплате забронированных туров.
tour	Страница с информацией о туре	Предоставляет пользователям информации о выбранном им туре, с возможностью его бронирования
tourEditor	Страница редактора	Предоставляет тур для редактирования или создания нового.
tours	Страница с турами. Главная страница	Предоставляет пользователю список туров.
toursEditor	Страница с турами для редактирования.	Предоставляет список туров, переходя по которым, попадаешь в редактор тура.
travelInfo	Страница с советами для путешественников	Предоставляет пользователям информации о том, куда лучше отправиться.
user	Страница пользователя	Предоставляет пользователю информацию о его профиле.
userBookings	Страница с бронями	Предоставляет пользователю список его броней.
users	Страница с пользователями	Предоставляет список пользователей.

Компоненты и страницы реализуют все необходимые функции для различных ролей.

3.8 Выводы

1. В web-приложении использована программная платформа ASP .NET Core для разработки серверной части приложения. Для хранения данных используется база данных MariaDB. Для работы с базой данных применяется ORM Entity Framework.

2. Для разработки клиентской части использовалась библиотека React, которая позволила реализовать компонентный подход.

3. Был разработан необходимый функционал для всех ролей: гостя, пользователя, менеджера и администратора. Количество функций 15.

4 Тестирование веб–приложения

4.1 Функциональное тестирование

Для проверки корректности работы всех функций разработанного веб-приложения было проведено ручное тестирование, описание и итоги которого представлены в таблице 4.1.

Таблица 4.1 – Примеры тестирования функций

Функция	Описание тестирования	Ожидаемый результат	Итог
Регистрация	Действие: отправить POST запрос на адрес /auth/register/, указав в теле запроса email «ilya@gmail.com», имя (name) «ilya», фамилию (surname) «dmitruk», пароль (password) «1234», номер телефона (phonePassword) «+375336461866».	В базе данных появится новый пользователь. Сервер вернёт токен.	Успешно.
Авторизация	Действие: отправить POST запрос на адрес /auth/login/, указав в теле запроса email «ilya@gmail.com», пароль (password) «1234».	Сервер должен вернуть токен.	Успешно.
Просмотр туров	Действие: необходимо отправить GET-запрос на адресс tour/tours/	Сервер должен вернуть все туры	Успешно.
Просмотр тура	Отправить GET-запрос на адрес /tour/get?tourId=1	Сервер должен вернуть тур с id равного 1	Успешно

Фильтрация туров	Отправить POST-запрос на адрес /tour/filtred_tours, указав в теле запроса countryId «1».	Сервер должен вернуть тур, отель которого находится в стране с id, равного 1	Успешно
Бронирование туров	Отправить POST-запрос на адрес booking/add, указав в теле количество заказываемых мест (orderSeatsNumber) «2», routeId «1», userId «1».	В базе данных появится новый тур, с количеством заказанных мест равного 2, и связанный с маршрутом с id 1 и пользователем с id 1	Успешно
Оставление отзывов к туру	Отправить POST-запрос на адрес review/add, указав в теле userId «1», tourId «1», текст отзыва (ReviewText) «хороший тур».	В базе данных появится отзыв с текстом «хороший тур», у маршрута с id 1, связанный с пользователем с id 1	Успешно
Добавление туров	Отправить POST-запрос на адрес tour/add, указав в теле name «mmm», MainDescription «nnn», hotelId «1», tourTypeId «1», nutritionTypeId «1».	В базе данных появится тур с названием «mmm», основным описание «nnn», с отелем, типом тура, типом питания, id которых 1.	Успешно
Редактирование туров	Отправить PUT-запрос на адрес tour/edit, указав в теле id «1», name «mmm», MainDescription «nnn», hotelId «1», tourTypeId «1», nutritionTypeId «1».	В базе данных у тура, с id 1 изменится название на «mmm», основное описание на «nnn», id отеля, типа тура, типа питания на 1.	Успешно
Удаление туров	Отправить DELETE-запрос на адрес tour/delete?tourId=1	В базе данных удалиться тур, с id равного 1.	Успешно
Подтверждение брони	Отправить PATCH-запрос на адрес booking/confirm?bookingId=1	Статус брони в базе данных, id у которой равно 1, сменится на true.	Успешно

Удаление отзывов к турам	Отправить DELETE-запрос на адрес review/delete?reviewId=1	В базе данных удалиться отзыв с id, равного 1.	Успешно
Смена роли пользователям	Отправить PATCH-запрос на адрес users/change_role?userId=1&roleId=2	В базе данных, у пользователя с id 1, смениться id роли на 2 (Менеджер).	Успешно
Блокировка/разблокировка пользователей	Отправить PATCH-запрос на адрес users/block?userId=1	В базе данных, у пользователя с id 1, сменится статус блокировки на противоположный	Успешно
Удаление пользователя	Отправить DELETE-запрос на адрес users/delete?userId=1	В базе данных удалиться пользователь с id, равного 1.	Успешно

Таким образом, были протестированы все ключевые функции web-приложения.

4.2 Выводы

1. Тестирование проводилось ручным методом. Проведено тестирование всех 15 функций приложения. Все операции работают корректно.
2. Покрытия тестами веб-приложения составляет 80%.

5 Руководство пользователя

5.1 Регистрация

Для регистрации необходимо нажать кнопку войти в правом верхнем углу главной страницы, выбрать регистрацию и в поля формы ввести email, имя, фамилию, номер телефона, пароль и повторный пароль. Форма регистрации с введенными данными представлена на рисунке 5.1.



The image shows a registration form for 'World Tours'. At the top left is a logo featuring a globe with a location pin and a paper airplane. To the right of the logo is the text 'World Tours'. Below the logo and text are two buttons: 'Вход' (Login) and 'Регистрация' (Registration). The 'Регистрация' button is highlighted in blue. Below these buttons are several input fields, each with a horizontal line indicating where to enter data. The fields contain the following text: 'mondaxfeall1@gmail.com', 'Илья', 'Дмитрук', '+375336461866', and two fields with four dots '....' representing a password and its repeat. At the bottom of the form is a large blue button labeled 'Зарегистрироваться' (Register).

Рисунок 5.1 – Форма регистрации с введенными данными

После ввода данных, необходимо нажать кнопку «Зарегистрироваться», которая находится под формой. Затем выполнится валидация, и если она пройдет успешно, то в базе данных появятся ваши учетные данные, а вас перенаправит на главную страницу.

5.2 Авторизация

Если вы уже авторизовывались, то вам ничего делать не надо. Вы просто заходите на сайт, токен сам отправится на сервер и ваши учетные данные сами подгрузятся. Если же вы неавторизовывались, то вам необходимо нажать кнопку войти в правом верхнем углу главной страницы, и ввести в форму входа свои

аутентификационные данные, такие как email и пароль. Форма входа с введенными данными представлена на рисунке 5.2.



Рисунок 5.2 – Форма входа с введенными данными


После ввода данных, необходимо нажать кнопку «Войти», которая находится под формой. Затем на сервере выполнится поиск пользователя с введенным email и паролем, и данные найденного пользователя отправляются вам, и вас перенаправит на главную страницу.

5.3 Просмотр туров


Для просмотра туров вам достаточно просто зайти на главную страницу сайта, даже не обязательно авторизовываться. Перед вами отобразится список туров с

5.4 Просмотр тура


Для того что бы просмотреть отдельный тур, вам необходимо нажать на него в списке туров. Вам откроется страница с подробной информацией о туре, включающие изображение, название, отель, пункт направления, тип питания, описание, характеристики. Справа же будет информация о определенном маршруте. Там же кнопка для бронирования. Страница тура представлена на рисунке 5.4.



[О компании](#)
[Оплата](#)
[Куда поехать?](#)



Жаркий Таиланд



Andaman Pearl Resort

Таиланд, Пхукет

★★★★

Тип питания: AI (All inclusive) — всё включено — завтрак, обед и ужин (шведский стол)

490 BYN Осталось мест: 14

Дата отправления 29.01.2025	Дата возвращения 13.02.2025
Отправление Москва	Прибытие Пхукет

Количество людей:

Заявка на тур

✚

15.08 - 08.08
29.01.2025
29.01.2025

Москва - Пхукет

✚

04.04 - 21.04
12.02.2025
13.02.2025

Пхукет - Москва

Рисунок 5.4 – Страница тура

5.5 Фильтрация туров

Для фильтрации туров на главной странице расположены ряд инструментов. В верху страницы расположен главный фильтр. Он представлен на рисунке 5.5.

Выберите направление
Регион, страна, город

Выберите город отправления
Не важно ▾

Отправления с
ДД.ММ.ГГГГ 📅

Отправления до
ДД.ММ.ГГГГ 📅


Выберите транспорт
Не важно ▾


🔍


Рисунок 5.5 – Главный фильтр


После выбора своих параметров фильтрации, необходимо нажать на кнопку с лупой.


Ниже главного фильтра расположилось навигационное меню с типами туров. Что бы сделать так, чтобы отображались туры с определённым типом, необходимо нажать на этот тип. Фильтр по типу туров представлен на рисунке 5.6.



 Все виды туров


 Отдых на море


 Горнолыжный курорт


 Путешествия по природе


 Культурный туризм


 Обычная поездка



 Фильтры

Рисунок 5.6 – Фильтр по типу туров

Справа от типов туров есть кнопка Фильтр. Нажав на неё, откроется модальное окно с более подробными фильтрами. Оно представлено на рисунке 5.7.

✕

Фильтрация туров

Мин. цена

Макс. цена

Мин. звёзды отеля: ★☆☆☆☆

Макс. звёзды отеля: ★★★★★

Питание:

Не важно ▾

	Не важно	Да	Нет
Пляж рядом	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Наличие Wi-fi	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Раздельные кровати	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Раздельный санузел	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Бассейн	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Джакузи	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Дискотеки	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Бильярд	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Теннис	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

Очистить всё

Показать

Рисунок 5.7 – Модальное окно более подробных фильтров

На рисунке 5.8 показан результат фильтрации. Я в главном фильтре выбрал направление: страна ОАЭ со всеми городами, город отправления: Москва, отправление в тур: в промежутке 01.05.2025 – 31.05.2025, и тип транспорта: самолёт. Мне отобразились туры, соответствующие фильтру.

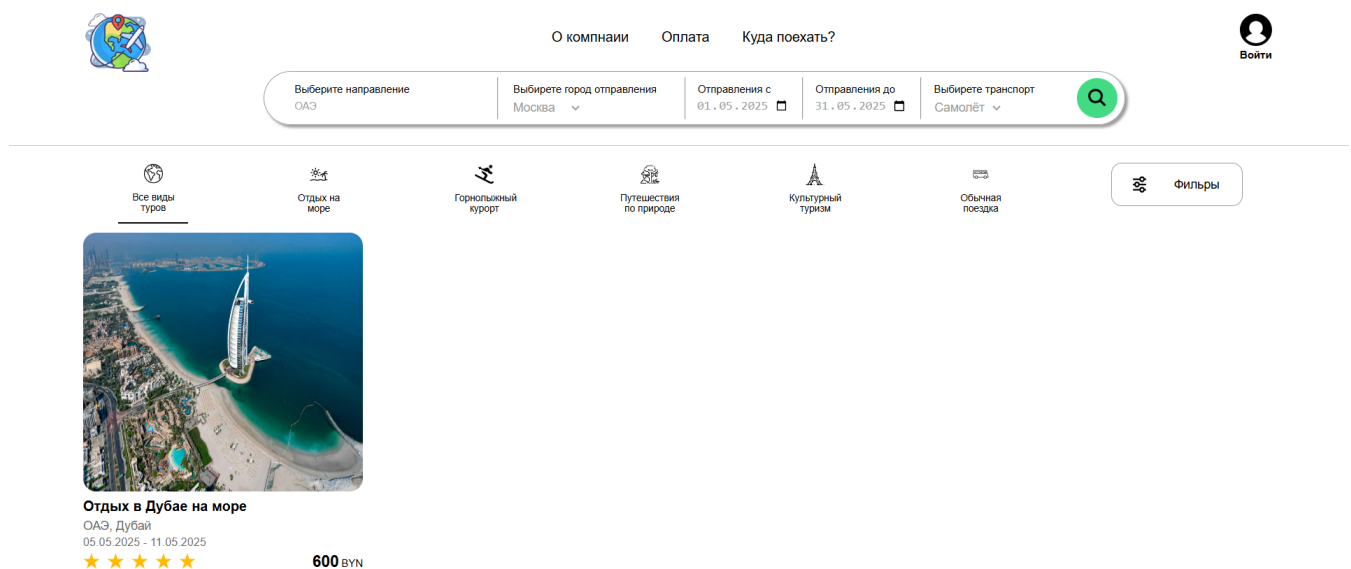


Рисунок 5.8 – Результат фильтрации

5.6 Бронирование туров

Для бронирования необходимо войти на страницу тура, в правом меню указать количество заказываемых мест (при их смене цена будет умножаться на количество указанных мест), затем нажать кнопку «Заявка на тур». Если тур не был заказан ранее и есть свободные места, то отобразиться сообщение о успешном бронировании тура. Результат бронирования представлен на рисунке 5.9.

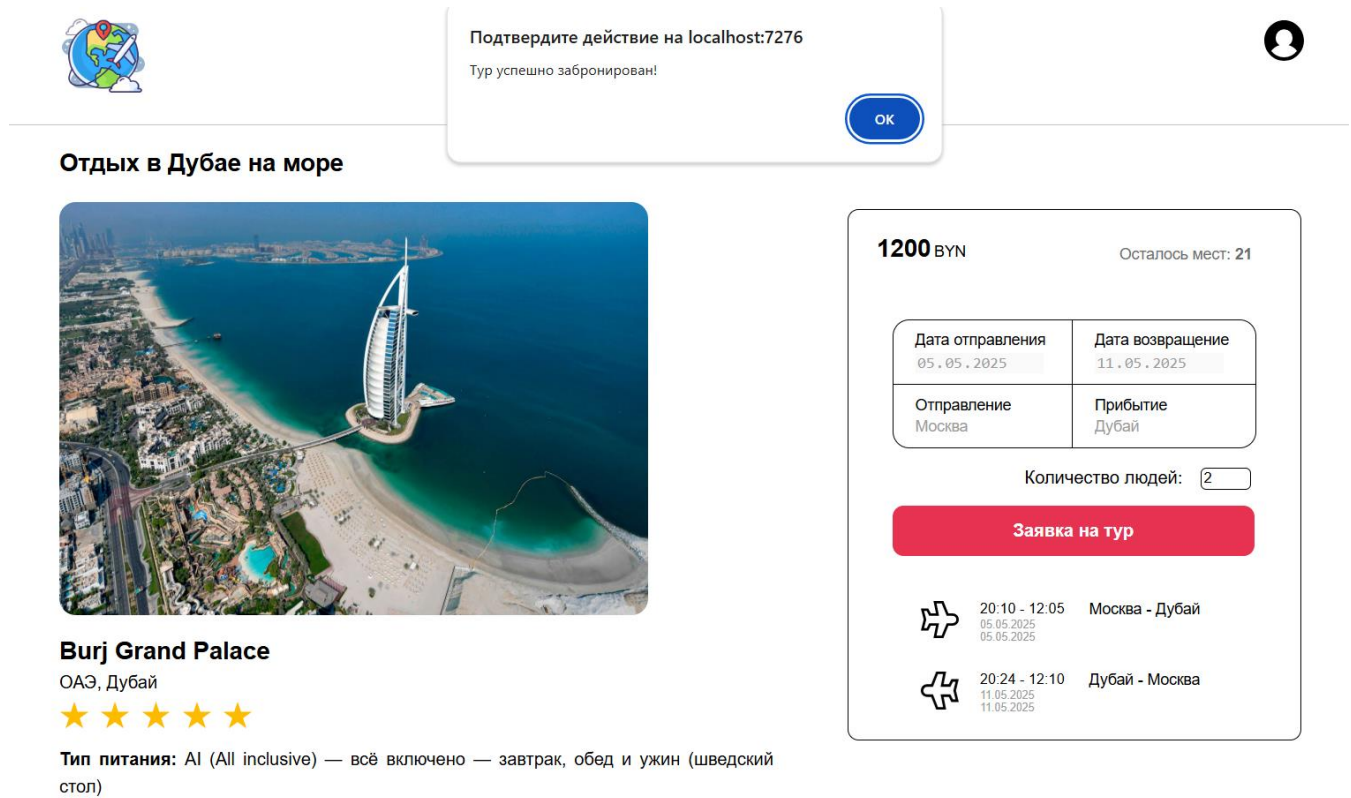


Рисунок 5.9 – Результат бронирования тура

5.6 Оставлять отзывы к турам

Что бы оставить отзыв, вам необходимо перейти на страницу тура, и пролистать вниз. Там можно увидеть другие отзывы, и в специальное поле вписать свой отзыв, и, нажав кнопку отправить, оставить отзыв. Оставленный отзыв отображён на рисунке 5.10.

Отзывы

Это был лучший тур!

Отправить

Илья Дмитрук

Это был лучший тур!

Рисунок 5.10 – Оставленный отзыв

5.8 Добавление туров

Для того что бы добавить тур, вам необходимо нажать на главной странице кнопку с плюсиком возле типов туров. Вам перебросит на страницу редактора туров с пустыми полями. Страница с добавления тура с незаполненными данными представлена на рисунке 5.11.

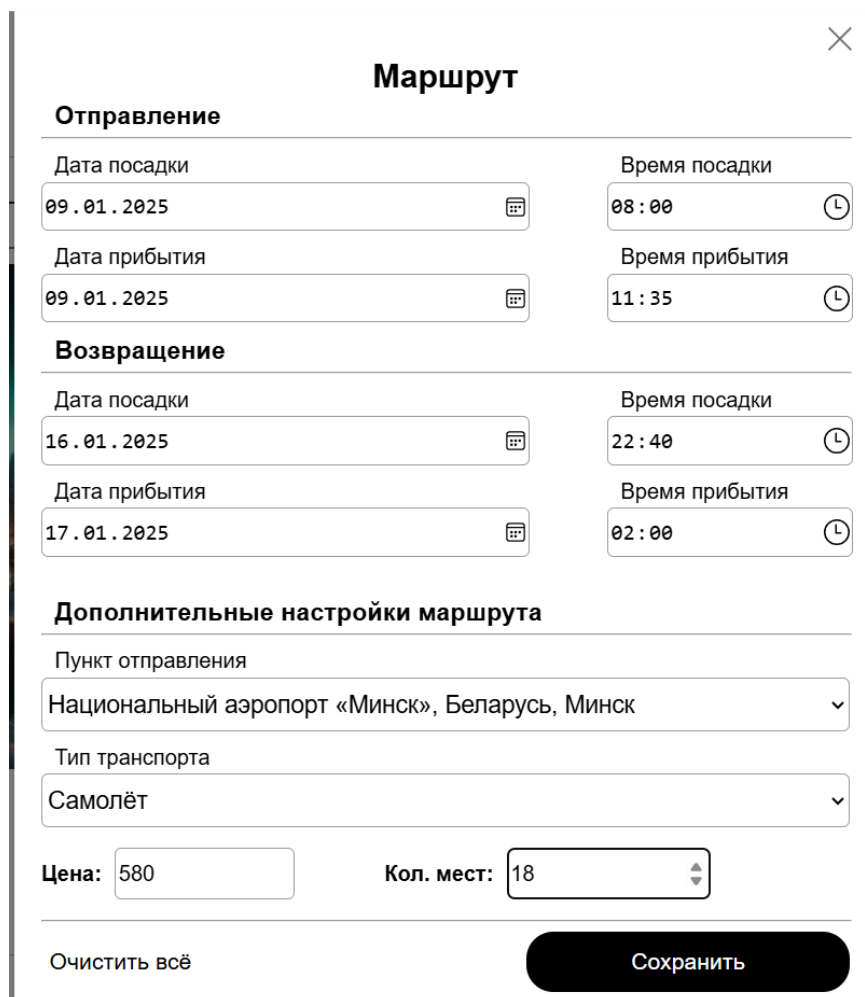
The screenshot shows a web interface for creating a new tour. At the top, there is a navigation bar with a globe icon, links for 'О компании' (About us), 'Оплата' (Payment), and 'Куда поехать?' (Where to go?), and a user profile icon. The main form area contains several input fields and buttons:

- A text input field labeled 'Название тура' (Tour name).
- A large grey rectangular area with the text 'Нажмите, что бы загрузить изображение' (Click to upload image).
- A button labeled 'Нажмите, что бы добавить пункт направления' (Click to add direction point).
- A text input field labeled 'Описание' (Description).
- A section titled 'Маршруты:' (Routes:) with a blue plus icon in the top right corner.
- A red button labeled 'Сохранить тур' (Save tour) at the bottom right of the routes section.

Рисунок 5.11 – Страница добавления тура с незаполненными данными

Вам необходимо ввести название тура, загрузить его изображение, под изображением необходимо нажать кнопку с надписью «Нажмите, чтобы добавить пункт направления», выбрать страну, город отель из предложенных списков, ввести описание тура, выбрать тип питания, выбрать тип тура и выделить те характеристики, что присущи этому туру. В правой части странице находится меню маршрутов тура. Что бы добавить новый маршрут, необходимо нажать на кнопку с плюсиком в данном меню,

затем ввести все данные маршрута, такие как дату посадки для отправления и дату прибытия в пункт назначения, вместе с временем и те же даты и время для возвращения. Далее необходимо выбрать пункт отправления и тип транспорта. Затем ввести цену и количество мест в данном маршруте. Модальное окно ввода данных маршрута представлено на рисунке 5.12.



Маршрут

Отправление

Дата посадки: 09.01.2025
Время посадки: 08:00

Дата прибытия: 09.01.2025
Время прибытия: 11:35

Возвращение

Дата посадки: 16.01.2025
Время посадки: 22:40

Дата прибытия: 17.01.2025
Время прибытия: 02:00

Дополнительные настройки маршрута

Пункт отправления: Национальный аэропорт «Минск», Беларусь, Минск


Тип транспорта: Самолёт

Цена: 580 Кол. мест: 18


Очистить всё Сохранить

Рисунок 5.12 – Модальное окно ввода данных маршрута

Затем нажать кнопку сохранить. В списке маршрутов появится новый маршрут. Его можно удалить, наведясь на него и нажав на крестик в правом верху маршрута. Маршрутов можно добавить несколько. Страница с добавлением тура с заполненными данными представлена на рисунке 5.13




[О компании](#)
[Оплата](#)
[Куда поехать?](#)



Название тура

Зимняя сказка



Aurora Lights Inn
 Норвегия, Тронхейм
 ★ ★ ★ ★

Описание

Маршруты:

+

Отправление

Беларусь, Минск → Норвегия, Тронхейм

2025-01-09, 08:00 → 2025-01-09, 11:35

Возвращение

Беларусь, Минск ← Норвегия, Тронхейм

2025-01-17, 02:00 ← 2025-01-16, 22:40

Тип транспорта:

Самолёт

Кол. мест: 18

Цена: 580 BYN

Отправление

Беларусь, Минск → Норвегия, Тронхейм

2025-01-16, 08:00 → 2025-01-23, 11:35

Возвращение

Беларусь, Минск ← Норвегия, Тронхейм

2025-02-07, 02:00 ← 2025-01-30, 22:40

Тип транспорта:

Самолёт

Кол. мест: 12

Цена: 530 BYN

Сохранить тур

Рисунок 5.13 – Страница добавления тура с заполненными данными

Ну и что-бы добавить тур, необходимо нажать кнопку «Сохранить тур».

5.9 Редактирование туров

Процесс редактирования туров в целом схож с доаввлением, только у вас при открытии редактора туров, будут подгружены данные тура. Что бы открыть страницу для редактирования тура, необходимо нажать на тур в главном окне.

5.10 Удаление туров

Что бы удалить тур, необходимо навестись на него на главной странице и нажать на крестик в правом верхнем углу. Тур с крестиком для удаления отображён на рисунке 5.14.



Рисунок 5.14 – Тур с крестиком для удаления

5.11 Подтверждения/отмена брони туров

Для того что бы перейти на страницу подтверждения броней туров, необходимо нажать на значёк профиля в правом верхнем углу, и нажать во всплывшем меню «Заявка на брони». Затем необходимо в необходимой брони в нижнем правом углу нажать кнопку подтверждения. После чего, кнопка поменяет цвет на синий и текст на «Подтвержденно». Для отмены брони, необходимо на неё навестись и нажать на крестик в правом верхнем углу. Страница с бронями для подтверждения представленна на рисунке 5.15.

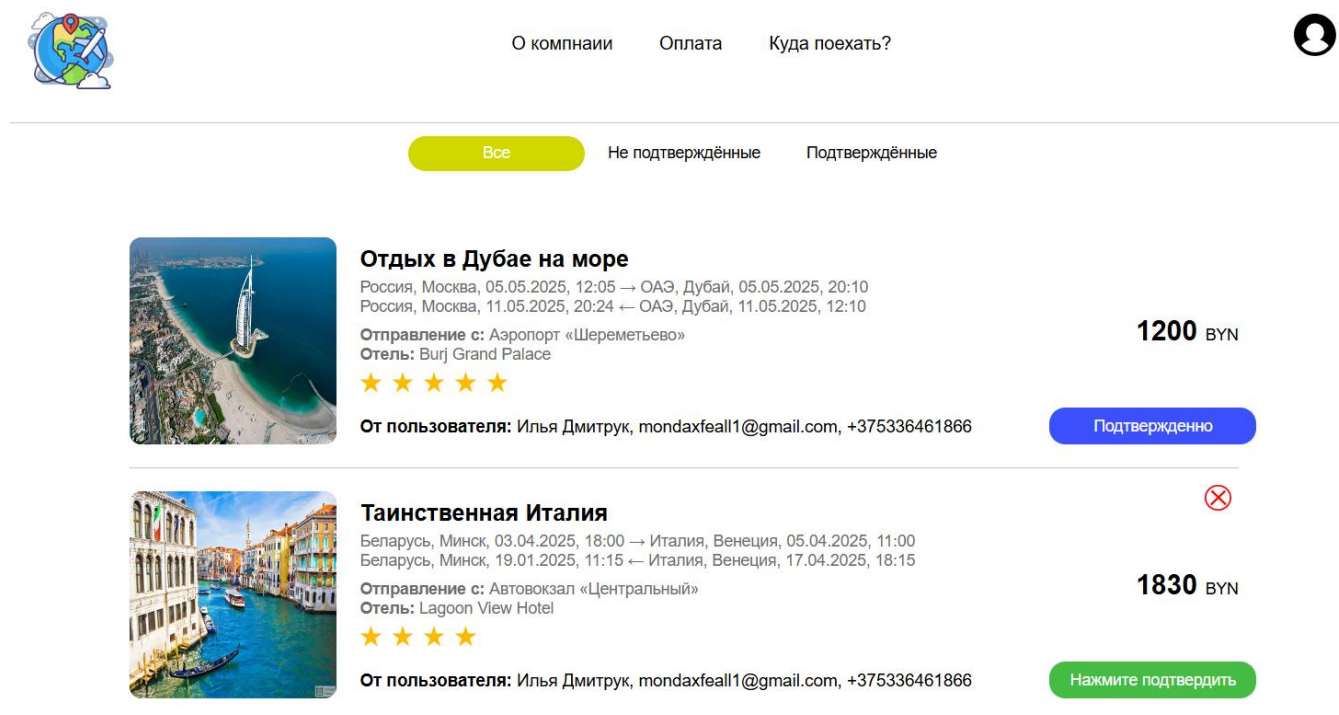


Рисунок 5.15 – Страница с бронями для подтверждения

5.12 Удаление отзывов к турам

Для того что бы удалить отзыв к туру, необходимо на него навестись и нажать на крестик справа. Отзыв с крестиком для удаления представлен на рисунке 5.16.

Отзывы

Илья Дмитрук
Это был лучший тур!



Рисунок 5.16 – Отзыв с крестиком для удаления

5.13 Смена роли пользователям

Что бы изменить роли пользователю, вам необходимо перейти на страницу с пользователями. Для этого нажмите на значок профиля в правом верхнем углу и в всплывающем меню нажмите на «Пользователи». Затем у нужного пользователя

нажмите «Сменить роль». У вас раскроются кнопки с ролями, выберите нужную. Смена роли пользователя представлена на рисунке 5.17.

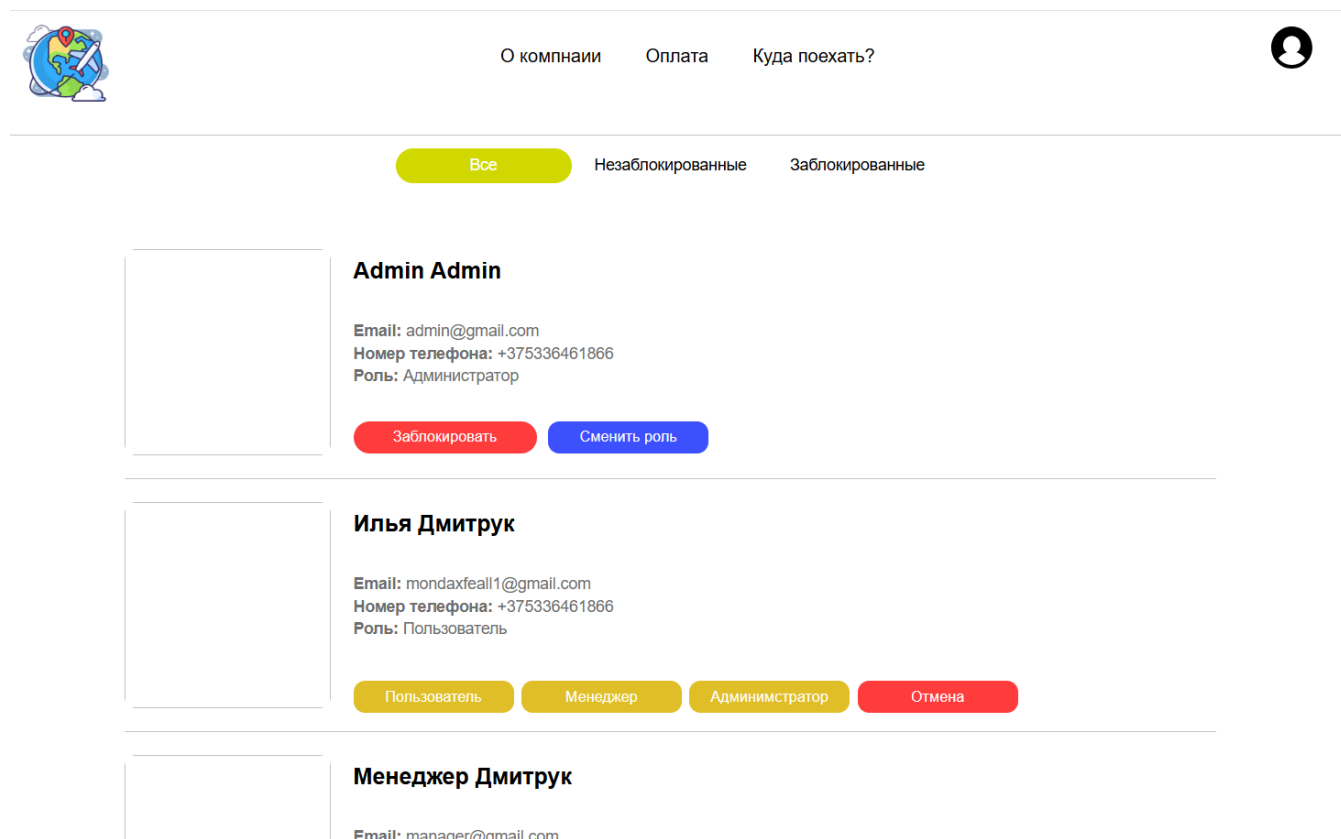


Рисунок 5.17 – Смена роли пользователю

5.14 Блокировка/разблокировка пользователей

Для того что заблокировать пользователя, необходимо нажать на соответствующую кнопку у нужного пользователя. Что бы разблокировать, так же необходимо нажать на соответствующую кнопку у нужного пользователя. Заблокированные и разблокированные пользователи представлены на рисунке 5.18.

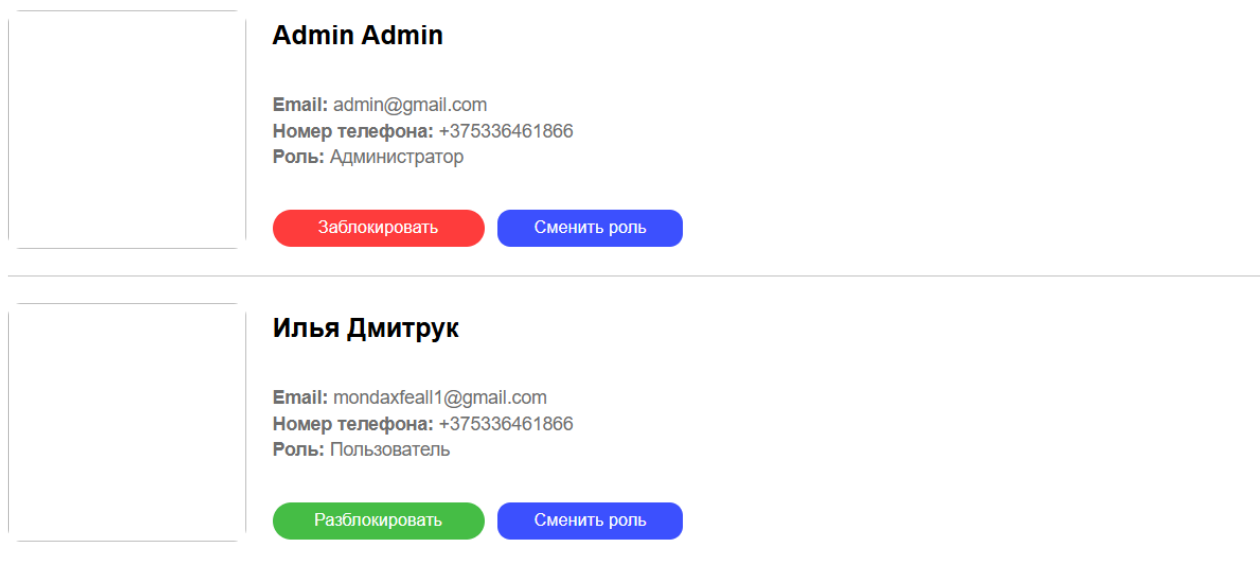


Рисунок 5.18 – Заблокированные и разблокированные пользователи

5.15 Удаление пользователей

Что бы удалить пользователя, необходимо навестись на него и нажать на крестик в правом верхнем углу. Пользователь с крестиком для удаления представлен на рисунке 5.19.

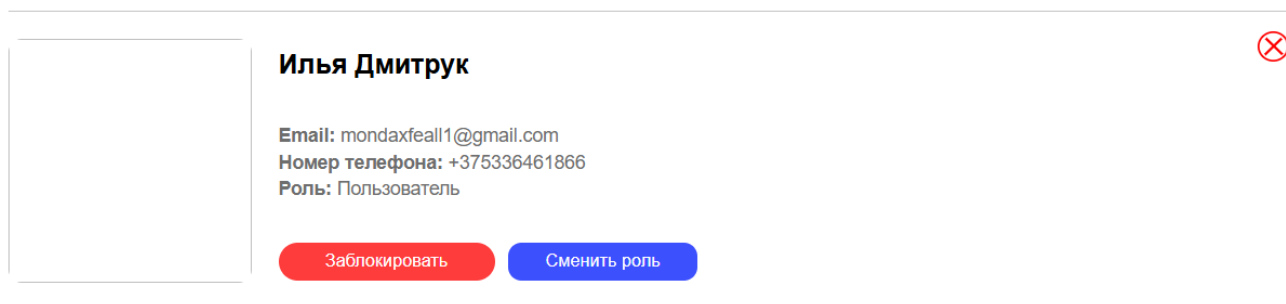


Рисунок 5.18 – Пользователь с крестиком для удаления

Заключение

В ходе данного курсового проекта было реализовано веб-приложение турагентство для заказа туров в разные страны. Основные результаты разработки:

1. Web-приложение поддерживает 4 роли: гость, пользователь, менеджер, администратор.
2. В ходе рассмотрения аналогов разрабатываемого приложения были выявлены основные функции и элементы.
3. Реализовано 15 функций.
4. База данных включает 18 таблиц.
5. Использована REST-архитектура, обеспечивающая удобное взаимодействие клиента и сервера через HTTPS. Модели данных описывают таблицы базы данных и их связи, разработаны с помощью ORM Entity Framework Core.
6. Объем кода составил более 6 000 строк кода
7. Было разработано 15 тестов, которые покрыли 80% функционала web-приложения.

В результате реализации веб-приложения были выполнены все поставленные задачи, такие как:

- поддерживать роли гостя, пользователя, менеджера и администратора;

Гость со следующим функционалом:

- регистрация;
- авторизация;
- просмотр туров.

Пользователь со следующим функционалом:

- просмотр туров;
- бронирование туров;
- оставлять отзывы к турам.

Менеджер со следующим функционалом:

- добавление туров;
- удаление туров;
- редактирование туров;
- подтверждение/отмена брони туров.

Администратор со следующим функционалом:

- добавление туров;
- удаление туров;
- редактирование туров;
- подтверждение/отмена брони тура;
- удаление отзывов к турам;
- смена роли пользователям;
- блокировка/разблокировка пользователей;
- удаление пользователей.

Список используемых источников

1. ASP.NET Core | Полное руководство [Электронный ресурс] / Режим доступа: <https://metanit.com/sharp/aspnet6/> – Дата доступа: 04.10.2024.
2. React Official Documentation. [Электронный ресурс] – Режим доступа: <https://react.dev/learn> – Дата обращения: 26.11.2024.
3. MariaDB Documentation. [Электронный ресурс] – Режим доступа: <https://mariadb.org/documentation/> – Дата обращения: 26.11.2024.
4. Entity Framework Core [Электронный ресурс] / Режим доступа: <https://learn.microsoft.com/ru-ru/ef/core/> – Дата обращения: 20.10.2024.
5. Docker [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://docs.docker.com/get-started/docker-overview/> – Дата обращения: 22.12.2024;
6. People Travel [Электронный ресурс] – Режим доступа: <https://peopletravel.by/>. – Дата обращения: 22.10.2024.
7. Coral Travel. [Электронный ресурс] – Режим доступа: <https://grusha-travel.by/>. – Дата обращения: 22.10.2024
8. Tez Tour. [Электронный ресурс] – Режим доступа: <https://www.tez-tour.com/> – Дата обращения: 22.10.2024.
9. Linux Ubuntu 22.04. [Электронный ресурс] – Режим доступа: <https://help.ubuntu.com/22.04/ubuntu-help/index.html>. – Дата обращения: 22.10.2024.
10. HTTPS. [Электронный ресурс] – Режим доступа: <https://developer.mozilla.org/ru/docs/Glossary/HTTPS/>. – Дата обращения: 22.10.2024.
11. Docker Compose Docs [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://docs.docker.com/compose/> – Дата обращения: 22.12.2024;
12. JWT [Электронный ресурс]. – Режим доступа: <https://jwt.io> – Дата обращения: 28.10.2024.
13. Axios GitHub Repository. [Электронный ресурс] – Режим доступа: <https://github.com/axios/axios>. – Дата обращения: 26.11.2024.
14. React Docs [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://react.dev/learn> – Дата обращения: 22.12.2024.
15. react-router-dom [Электронный ресурс]. – Режим доступа: <https://reactrouter.com/> – Дата обращения: 12.12.2024.
16. Json standart, technical specifications. [Электронный ресурс] – Режим доступа: <https://www.json.org/json-en.html>. – Дата обращения: 25.11.2024.
17. multipart/form-data [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST> – Дата обращения: 01.11.2024

Приложение А

Код контекста базы данных

```

public class AppDbContext : DbContext
{
    public DbSet<User> Users { get; set; }
    public DbSet<TourType> TourTypes { get; set; }
    public DbSet<CharacteristicType> CharacteristicTypes { get; set; }
    public DbSet<Characteristic> Characteristics { get; set; }
    public DbSet<Description> Descriptions { get; set; }
    public DbSet<Region> Regions { get; set; }
    public DbSet<Country> Countries { get; set; }
    public DbSet<City> Cities { get; set; }
    public DbSet<DepartmentDeparture> DepartmentDepartures { get; set; }
    public DbSet<TransportType> TransportTypes { get; set; }
    public DbSet<Hotel> Hotels { get; set; }
    public DbSet<NutritionType> NutritionTypes { get; set; }
    public DbSet<Tour> Tours { get; set; }
    public DbSet<Models.Entity.Route> Routes { get; set; }
    public DbSet<Booking> Bookings { get; set; }
    public DbSet<Review> Reviews { get; set; }

    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
    {
        Database.EnsureCreated();    // создаем базу данных при первом
        // обращении
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        // Настройка связи многие-ко-многим
        modelBuilder.Entity<TourType>()
            .HasMany(tt => tt.Characteristics)
            .WithMany(c => c.TourTypes)
            .UsingEntity<Dictionary<string, object>>(
                "Charcteristics_TourTypes", // Название промежуточной
таблицы
                j => j
                    .HasOne<Characteristic>()
                    .WithMany()
                    .HasForeignKey("CharacteristicID")
                    .HasConstraintName("FK_Charcteris-
tics_TourTypes_Characteristics"),
                j => j
                    .HasOne<TourType>()
                    .WithMany()
    
```

```

        .HasForeignKey("TourTypeId")
        .HasConstraintName("FK_Characteris-
tics_TourTypes_TourTypes")
    );

    // Настройка User
    modelBuilder.Entity<User>()
        .HasIndex(u => u.Email)
        .IsUnique();

    // Настройка TourType
    modelBuilder.Entity<TourType>()
        .HasMany(tt => tt.Tours)
        .WithOne(t => t.TourType)
        .HasForeignKey(t => t.TourTypeId)
        .OnDelete(DeleteBehavior.SetNull);

    modelBuilder.Entity<CharacteristicType>()
        .HasMany(ct => ct.Characteristics)
        .WithOne(c => c.CharacteristicType)
        .HasForeignKey(c => c.CharacteristicTypeId)
        .OnDelete(DeleteBehavior.Cascade);

    // Настройка Characteristic
    modelBuilder.Entity<Characteristic>()
        .HasMany(c => c.Descriptions)
        .WithOne(d => d.Characteristic)
        .HasForeignKey(d => d.CharacteristicId)
        .OnDelete(DeleteBehavior.Cascade);

    // Настройка Description
    modelBuilder.Entity<Description>()
        .HasOne(d => d.Tour)
        .WithMany(t => t.Descriptions)
        .HasForeignKey(d => d.TourId)
        .OnDelete(DeleteBehavior.Cascade);

    // Настройка Region
    modelBuilder.Entity<Region>()
        .HasMany(r => r.Countries)
        .WithOne(c => c.Region)
        .HasForeignKey(c => c.RegionId)
        .OnDelete(DeleteBehavior.Cascade);

    // Настройка Country
    modelBuilder.Entity<Country>()
        .HasMany(c => c.Cities)

```

```

        .WithOne(city => city.Country)
        .HasForeignKey(city => city.CountryId)
        .OnDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<City>()
    .HasMany(city => city.DepartmentDepartures)
    .WithOne(dd => dd.City)
    .HasForeignKey(dd => dd.CityId)
    .OnDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<City>()
    .HasMany(city => city.Hotels)
    .WithOne(h => h.City)
    .HasForeignKey(h => h.CityId)
    .OnDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<DepartmentDeparture>()
    .HasMany(dd => dd.Routes)
    .WithOne(r => r.DepartmentDeparture)
    .HasForeignKey(r => r.DepartmentDepartureId)
    .OnDelete(DeleteBehavior.Restrict);

modelBuilder.Entity<TransportType>()
    .HasMany(tt => tt.Routes)
    .WithOne(r => r.TransportType)
    .HasForeignKey(r => r.TransportTypeId)
    .OnDelete(DeleteBehavior.Restrict);

modelBuilder.Entity<Hotel>()
    .HasMany(h => h.Tours)
    .WithOne(t => t.Hotel)
    .HasForeignKey(t => t.HotelId)
    .OnDelete(DeleteBehavior.SetNull);

modelBuilder.Entity<NutritionType>()
    .HasMany(nt => nt.Tours)
    .WithOne(t => t.NutritionType)
    .HasForeignKey(t => t.NutritionTypeId)
    .OnDelete(DeleteBehavior.SetNull);

modelBuilder.Entity<Tour>()
    .HasMany(t => t.Routes)
    .WithOne(r => r.Tour)
    .HasForeignKey(r => r.TourId)

```

```

        .onDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<Tour>()
    .HasMany(t => t.Routes)
    .WithOne(r => r.Tour)
    .HasForeignKey(r => r.TourId)
    .onDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<Tour>()
    .HasMany(t => t.Reviews)
    .WithOne(r => r.Tour)
    .HasForeignKey(r => r.TourId)
    .onDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<User>()
    .HasMany(u => u.Bookings)
    .WithOne(b => b.User)
    .HasForeignKey(b => b.UserId)
    .onDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<User>()
    .HasMany(u => u.Reviews)
    .WithOne(r => r.User)
    .HasForeignKey(r => r.UserId)
    .onDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<Models.Entity.Route>()
    .HasMany(r => r.Bookings)
    .WithOne(b => b.Route)
    .HasForeignKey(b => b.RouteId)
    .onDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<Models.Entity.Route>()
    .Property(r => r.Price)
    .HasDefaultValue(0);

modelBuilder.Entity<Models.Entity.Route>()
    .Property(r => r.SeatsNumber)
    .HasDefaultValue(0);

modelBuilder.Entity<Region>().HasIndex(r => r.Name).IsUnique();
modelBuilder.Entity<Country>().HasIndex(c => c.Name).IsUnique();
modelBuilder.Entity<City>().HasIndex(c => c.Name).IsUnique();
modelBuilder.Entity<DepartmentDeparture>().HasIndex(dd
dd.Name).IsUnique();
modelBuilder.Entity<Hotel>().HasIndex(h => h.Name).IsUnique();

```

```

        modelBuilder.Entity<NutritionType>().HasIndex(nt => nt.Name).Is-
Unique();
        modelBuilder.Entity<TourType>().HasIndex(tt => tt.Name).Is-
Unique();
        modelBuilder.Entity<Characteristic>().HasIndex(c => c.Name).Is-
Unique();
        modelBuilder.Entity<Tour>().HasIndex(t => t.Name).IsUnique();
    }
}

```

Приложение Б

Код моделей

```

public class User
{
    public int Id { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
    public UserRole Role { get; set; }
    public bool BlockedStatus { get; set; } = false;

    [MaxLength(255)]
    public string Name { get; set; }

    [MaxLength(255)]
    public string Surname { get; set; }

    [MaxLength(255)]
    public string PhoneNumber { get; set; }

    public byte[]? Photo { get; set; }

    public ICollection<Booking> Bookings { get; set; }
    public ICollection<Review> Reviews { get; set; }

    public User(int id, string email, string password, UserRole
role, string name, string surname, string phoneNumber)
    {
        Id = id;
        Email = email;
        Password = password;
        Role = role;
        Name = name;
        Surname = surname;
        PhoneNumber = phoneNumber;
    }
    public User(RegisterForm register, UserRole role = UserRole.User)
    {
        Email = register.Email;
        Password = register.Password;
        Name = register.Name;
    }
}

```

```

        Surname = register.Surname;
        PhoneNumber = register.PhoneNumber;
        Role = role;
    }
}
public class TransportType
{
    [Key]
    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    public string Name { get; set; }

    public ICollection<Route> Routes { get; set; }
}

public class TourType
{
    public int Id { get; set; }
    public string Name { get; set; }
    public byte[]? Image { get; set; }

    public ICollection<Characteristic> Characteristics { get; set; }
    public ICollection<Tour> Tours { get; set; }
}

public class Tour
{
    [Key]
    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    public string Name { get; set; }
    [MaxLength(255)]
    public string MainDescription { get; set; }
    public int? TourTypeId { get; set; }
    public int? NutritionTypeId { get; set; }
    public int? HotelId { get; set; }
    public byte[]? Photo { get; set; }

    [ForeignKey(nameof(TourTypeId))]
    public TourType TourType { get; set; }
    [ForeignKey(nameof(NutritionTypeId))]
    public NutritionType NutritionType { get; set; }
    [ForeignKey(nameof(HotelId))]
    public Hotel Hotel { get; set; }
    public ICollection<Route> Routes { get; set; }
    public ICollection<Description> Descriptions { get; set; }
    public ICollection<Review> Reviews { get; set; }
}

public class Route
{
    [Key]

```



```

    public int Id { get; set; }
    public DateTime? LandingDateOfDeparture { get; set; }
    public TimeSpan? LandingTimeOfDeparture { get; set; }
    public DateTime? ArrivalDateOfDeparture { get; set; }
    public TimeSpan? ArrivalTimeOfDeparture { get; set; }
    public DateTime? LandingDateOfReturn { get; set; }
    public TimeSpan? LandingTimeOfReturn { get; set; }
    public DateTime? ArrivalDateOfReturn { get; set; }
    public TimeSpan? ArrivalTimeOfReturn { get; set; }
    public int? Price { get; set; }
    public int? SeatsNumber { get; set; }
    public int? DepartmentDepartureId { get; set; }
    public int? TransportTypeId { get; set; }
    public int? TourId { get; set; }

    [ForeignKey(nameof(DepartmentDepartureId))]
    public DepartmentDeparture DepartmentDeparture { get; set; }

    [ForeignKey(nameof(TransportTypeId))]
    public TransportType TransportType { get; set; }

    [ForeignKey(nameof(TourId))]
    public Tour Tour { get; set; }
    public ICollection<Booking> Bookings { get; set; }
}

public class Review
{
    [Key]
    public int Id { get; set; }
    public int? UserId { get; set; }
    public int? TourId { get; set; }
    public string? ReviewText { get; set; }

    [ForeignKey(nameof(TourId))]
    public Tour Tour { get; set; }

    [ForeignKey(nameof(UserId))]
    public User User { get; set; }
}

public class Region
{
    [Key]
    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    public string Name { get; set; }
    public byte[]? Image { get; set; }
    public ICollection<Country> Countries { get; set; }
}

public class NutritionType

```

```

{
    [Key]
    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    public string Name { get; set; }

    public ICollection<Tour> Tours { get; set; }
}
public class Hotel
{
    [Key]
    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    public string Name { get; set; }
    public int StarsNumber { get; set; }
    public int CityId { get; set; }

    [ForeignKey(nameof(CityId))]
    public City City { get; set; }
    public ICollection<Tour> Tours { get; set; }
}
public class Description
{
    [Key]
    public int Id { get; set; }
    public bool Value { get; set; }
    public int CharacteristicId { get; set; }
    public int TourId { get; set; }

    [ForeignKey(nameof(CharacteristicId))]
    public Characteristic Characteristic { get; set; }
    [ForeignKey(nameof(TourId))]
    public Tour Tour { get; set; }
}
public class DepartmentDeparture
{
    [Key]
    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    public string Name { get; set; }
    public int CityId { get; set; }

    [ForeignKey(nameof(CityId))]
    public City City { get; set; }
    public ICollection<Route> Routes { get; set; }
}
public class Country
{
    [Key]

```

```

    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    public string Name { get; set; }
    public byte[]? Flag { get; set; }
    public int RegionId { get; set; }

    [ForeignKey(nameof(RegionId))]
    public Region Region { get; set; }
    public ICollection<City> Cities { get; set; }
}
public class City
{
    [Key]
    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    public string Name { get; set; }
    public int CountryId { get; set; }

    [ForeignKey(nameof(CountryId))]
    public Country Country { get; set; }
    public ICollection<DepartmentDeparture> DepartmentDepartures {
get; set; }
    public ICollection<Hotel> Hotels { get; set; }
}
public class CharacteristicType
{
    [Key]
    public int Id { get; set; }

    [Required]
    [MaxLength(255)]
    public string Name { get; set; }

    public ICollection<Characteristic> Characteristics { get; set; }
}
public class Characteristic
{
    [Key]
    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    public string Name { get; set; }
    public int CharacteristicTypeId { get; set; }

    [ForeignKey(nameof(CharacteristicTypeId))]
    public CharacteristicType CharacteristicType { get; set; }

    public ICollection<Description> Descriptions { get; set; }
    // Навигационное свойство для связи многие-ко-многим
    public ICollection<TourType> TourTypes { get; set; }
}

```

```
}  
public class Booking  
{  
    [Key]  
    public int Id { get; set; }  
    public int? OrderSeatsNumber { get; set; }  
    public int? RouteId { get; set; }  
    public int? UserId { get; set; }  
    public bool? Status { get; set; }  
  
    [ForeignKey(nameof(RouteId))]  
    public Route Route { get; set; }  
  
    [ForeignKey(nameof(UserId))]  
    public User User { get; set; }  
}
```

Приложение В

Код контроллера AuthController

```

public class AuthController : Controller
{
    private AppDbContext db;

    public AuthController(AppDbContext context)
    {
        db = context;
    }

    [Authorize]
    public async Task<IActionResult> Auth()
    {
        try
        {
            User user = await db.Users.FirstOrDefaultAsync(u =>
u.Email == User.FindFirst(ClaimTypes.Email).Value);

            return user != null ? Ok(new UserDto()
            {
                Id = user.Id,
                Email = user.Email,
                Name = user.Name,
                Surname = user.Surname,
                PhoneNumber = user.PhoneNumber,
                PhotoUrl = PhotoService.ConvertToBase64(user.Photo,
"png"),
                BlockedStatus = user.BlockedStatus,
                Role = user.Role
            }) : Unauthorized();
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [HttpPost]
    public async Task<IActionResult> Register([FromBody] RegisterForm
register)
    {
        try
        {
            if (await db.Users.FirstOrDefaultAsync(u => u.Email ==
register.Email) != null) return Conflict(new { message = "Этот email
уже используется." });

            register.Password =
HashService.ComputeHash(register.Password);
            await db.Users.AddAsync(new User(register));
            await db.SaveChangesAsync();
        }
    }
}

```

```

        return Ok(new { token =
TokenService.GenerateToken(register.Email, UserRole.User) });
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

[HttpPost]
public async Task<IActionResult> Login([FromBody] LoginForm
login)
{
    try
    {
        User user = await db.Users.FirstOrDefaultAsync(u =>
u.Email == login.Email);
        return user != null &&
HashService.VerifyHash(login.Password, user.Password) ? Ok(new {
token = TokenService.GenerateToken(login.Email, user.Role) }) :
Unauthorized();
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
}

```

Приложение Г

Код контроллера BookingController

```
[Route("booking")]

public class BookingController : Controller
{
    private AppDbContext db;

    public BookingController(AppDbContext context)
    {
        db = context;
    }

    [HttpPost("add")]

    [Authorize(Roles = "User")]
    public async Task<IActionResult> AddBooking([FromBody]
ApplicationForBookingForm applicationForBooking)
    {
        try
        {
            Booking booking = await
db.Bookings.FirstOrDefaultAsync(b => b.UserId ==
applicationForBooking.UserId && b.RouteId ==
applicationForBooking.RouteId);
            if (booking != null) return Conflict(new { message =
"Вы уже заказали данный тур." });

            Route route = await db.Routes.FirstOrDefault(r =>
r.Id == applicationForBooking.RouteId);
            if (route == null) return BadRequest();

            User user = await db.Users.FirstOrDefault(r =>
r.Id == applicationForBooking.UserId);
            if (user == null) return Unauthorized();

            int remainingSeatsNumber = (int)route.SeatsNumber -
(int)applicationForBooking.OrderSeatsNumber;

            if (remainingSeatsNumber >= 0)
            {
                using (var transaction = await
db.Database.BeginTransactionAsync())
                {
                    try
                    {
                        route.SeatsNumber =
remainingSeatsNumber;

                        await db.SaveChangesAsync();

                        await db.Bookings.AddAsync(new Booking()
                        {
```

```

        UserId =
applicationForBooking.UserId,
        RouteId =
applicationForBooking.RouteId,
        OrderSeatsNumber =
applicationForBooking.OrderSeatsNumber
    });
    await db.SaveChangesAsync();

    await transaction.CommitAsync();
    return Ok();
}
catch
{
    await transaction.RollbackAsync();
    return BadRequest();
}
}
else return Conflict(new { message = "Недостаточно
мест." });
}
catch (Exception ex)
{
    return BadRequest(ex.Message);
}
}

[Authorize(Roles = "User")]
[HttpGet("bookings")]
public async Task<IActionResult> GetBookings([FromQuery] int?
userId)
{
    try
    {
        List<Booking> bookings = await db.Bookings
            .Where(b => b.UserId == userId)
            .Include(b => b.Route)
            .ThenInclude(r => r.Tour)
            .ThenInclude(t => t.Hotel)
            .ThenInclude(h => h.City)
            .ThenInclude(c => c.Country)
            .ToListAsync();

        foreach (Booking booking in bookings)
        {
            booking.Route.DepartmentDeparture = await
db.DepartmentDepartures
                .Include(dd => dd.City)
                .ThenInclude(c => c.Country)
                .FirstOrDefaultAsync(dd => dd.Id ==
booking.Route.DepartmentDepartureId);
        }
    }
}

```



```

        return Ok(bookings.Select(b => new BookingCardDto()
        {
            Id = b.Id,
            TourName = b.Route.Tour.Name,
            TourId = b.Route.Tour.Id,
            RouteId = b.Route.Id,
            TourPhotoUrl =
PhotoService.ConvertToBase64(b.Route.Tour.Photo, "png"),
            LandingDateOfDeparture =
b.Route.LandingDateOfDeparture?.ToString("dd.MM.yyyy"),
            LandingDateOfReturn =
b.Route.LandingDateOfReturn?.ToString("dd.MM.yyyy"),
            LandingTimeOfDeparture =
b.Route.LandingTimeOfDeparture?.ToString(@"hh\:mm"),
            LandingTimeOfReturn =
b.Route.LandingTimeOfReturn?.ToString(@"hh\:mm"),
            ArrivalDateOfDeparture =
b.Route.ArrivalDateOfDeparture?.ToString("dd.MM.yyyy"),
            ArrivalDateOfReturn =
b.Route.ArrivalDateOfReturn?.ToString("dd.MM.yyyy"),
            ArrivalTimeOfDeparture =
b.Route.ArrivalTimeOfDeparture?.ToString(@"hh\:mm"),
            ArrivalTimeOfReturn =
b.Route.ArrivalTimeOfReturn?.ToString(@"hh\:mm"),
            Price = b.Route.Price,
            OrderSeatsNumber = b.OrderSeatsNumber,
            Status = b.Status,
            Direction = new DirectionDto()
            {
                Hotel = b.Route.Tour.Hotel.Name,
                StarsNumber = b.Route.Tour.Hotel.StarsNumber,
                City = b.Route.Tour.Hotel.City.Name,
                Country =
b.Route.Tour.Hotel.City.Country.Name,
            },
            DepartmentDeparture = new DepartmentDepartureDto()
            {
                Name = b.Route.DepartmentDeparture.Name,
                City = b.Route.DepartmentDeparture.City.Name,
                Country =
b.Route.DepartmentDeparture.City.Country.Name,
            }
        }
        ));
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

[Authorize(Roles = "Manager, Admin")]
[HttpGet("bookings_for_manager")]

```

```

public async Task<IActionResult> GetBookingsForManager()
{
    try
    {
        List<Booking> bookings = await db.Bookings
            .Include(b => b.Route)
            .ThenInclude(r => r.Tour)
            .ThenInclude(t => t.Hotel)
            .ThenInclude(h => h.City)
            .ThenInclude(c => c.Country)
            .ToListAsync();

        foreach (Booking booking in bookings)
        {
            booking.Route.DepartmentDeparture = await
db.DepartmentDepartures
            .Include(dd => dd.City)
            .ThenInclude(c => c.Country)
            .FirstOrDefaultAsync(dd => dd.Id ==
booking.Route.DepartmentDepartureId);
        }

        foreach (Booking booking in bookings)
        {
            booking.User = await
db.Users.FirstOrDefaultAsync(u => u.Id == booking.UserId);
        }

        return Ok(bookings.Select(b => new
BookingCardForManagerDto()
        {
            Id = b.Id,
            TourName = b.Route.Tour.Name,
            TourId = b.Route.Tour.Id,
            RouteId = b.Route.Id,
            TourPhotoUrl =
PhotoService.ConvertToBase64(b.Route.Tour.Photo, "png"),
            LandingDateOfDeparture =
b.Route.LandingDateOfDeparture?.ToString("dd.MM.yyyy"),
            LandingDateOfReturn =
b.Route.LandingDateOfReturn?.ToString("dd.MM.yyyy"),
            LandingTimeOfDeparture =
b.Route.LandingTimeOfDeparture?.ToString(@"hh\:mm"),
            LandingTimeOfReturn =
b.Route.LandingTimeOfReturn?.ToString(@"hh\:mm"),
            ArrivalDateOfDeparture =
b.Route.ArrivalDateOfDeparture?.ToString("dd.MM.yyyy"),
            ArrivalDateOfReturn =
b.Route.ArrivalDateOfReturn?.ToString("dd.MM.yyyy"),
            ArrivalTimeOfDeparture =
b.Route.ArrivalTimeOfDeparture?.ToString(@"hh\:mm"),
            ArrivalTimeOfReturn =
b.Route.ArrivalTimeOfReturn?.ToString(@"hh\:mm"),

```

```

        Price = b.Route.Price,
        OrderSeatsNumber = b.OrderSeatsNumber,
        Status = b.Status,
        User = new UserDto()
        {
            Name = b.User.Name,
            Surname = b.User.Surname,
            Email = b.User.Email,
            PhoneNumber = b.User.PhoneNumber,
            PhotoUrl =
PhotoService.ConvertToBase64(b.User.Photo, "png"),
        },
        Direction = new DirectionDto()
        {
            Hotel = b.Route.Tour.Hotel.Name,
            StarsNumber = b.Route.Tour.Hotel.StarsNumber,
            City = b.Route.Tour.Hotel.City.Name,
            Country =
b.Route.Tour.Hotel.City.Country.Name,
        },
        DepartmentDeparture = new DepartmentDepartureDto()
        {
            Name = b.Route.DepartmentDeparture.Name,
            City = b.Route.DepartmentDeparture.City.Name,
            Country =
b.Route.DepartmentDeparture.City.Country.Name,
        }
    }));
}
catch (Exception ex)
{
    return BadRequest(ex.Message);
}

[Authorize]
[HttpDelete("delete")]
public async Task<IActionResult> DeleteBooking([FromQuery] int?
bookingId)
{
    try
    {
        using (var transaction = await
db.Database.BeginTransactionAsync())
        {
            try
            {
                Booking removedBooking = await
db.Bookings.FirstOrDefaultAsync(b => b.Id == bookingId);
                if (removedBooking == null) return
NotFound();
            }
        }
    }
}

```

```

        Route route = await
db.Routes.FirstOrDefaultAsync(r => r.Id == removedBooking.RouteId);
        if (route != null)
        {
            route.SeatsNumber +=
removedBooking.OrderSeatsNumber;
        }

        User user = await
db.Users.FirstOrDefaultAsync(u => u.Email ==
User.FindFirst(ClaimTypes.Email).Value);
        if (user == null) return StatusCode(403);
        if (User.FindFirst(ClaimTypes.Role).Value ==
nameof(UserRole.User) && user.Id != removedBooking.UserId) return
StatusCode(403);

        db.Bookings.Remove(removedBooking);
        await db.SaveChangesAsync();
        await transaction.CommitAsync();
        return Ok();
    }
    catch
    {
        await transaction.RollbackAsync();
        return BadRequest();
    }
}
catch (Exception ex)
{
    return BadRequest(ex.Message);
}
}

[Authorize(Roles = "Manager, Admin")]
[HttpPatch("confirm")]
public async Task<IActionResult> ConfirmBooking([FromQuery] int?
bookingId)
{
    try
    {
        Booking confirmedBooking = await
db.Bookings.FirstOrDefaultAsync(b => b.Id == bookingId);
        if (confirmedBooking == null) return NotFound();

        confirmedBooking.Status = true;
        await db.SaveChangesAsync();
        return Ok();
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
}
}

```

Приложение Д

Код контроллера UserController

```
[Route("user")]
public class UserController : Controller
{
    private AppDbContext db;
    public UserController(AppDbContext context)
    {
        db = context;
    }

    [Authorize]
    [HttpPut("edit")]
    public async Task<IActionResult> EditUser([FromForm] UserForm
user)
    {
        try
        {
            User editUser = await db.Users.FirstOrDefaultAsync(u
=> u.Id == user.Id);
            if (editUser == null) return NotFound();

            editUser.Name = user.Name;
            editUser.Surname = user.Surname;
            editUser.PhoneNumber = user.PhoneNumber;
            if (user.PhotoFile != null) editUser.Photo = await
PhotoService.ConvertToBytes(user.PhotoFile);

            await db.SaveChangesAsync();
            return Ok();
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [Authorize(Roles = "Admin")]
    [HttpPatch("block")]
    public async Task<IActionResult> BlockUser([FromQuery] int?
userId)
    {
        try
        {
            User blockedUser = await
db.Users.FirstOrDefaultAsync(u => u.Id == userId);
            if (blockedUser == null) return NotFound();

            blockedUser.BlockedStatus =
!blockedUser.BlockedStatus;
            await db.SaveChangesAsync();
            return Ok();
        }
    }
}
```

```

        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [Authorize(Roles = "Admin")]
    [HttpDelete("delete")]
    public async Task<IActionResult> DeleteUser([FromQuery] int?
userId)
    {
        try
        {
            User removedUser = await
db.Users.FirstOrDefaultAsync(u => u.Id == userId);
            if (removedUser == null) return NotFound();

            db.Users.Remove(removedUser);
            await db.SaveChangesAsync();
            return Ok();
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [Authorize(Roles = "Admin")]
    [HttpPatch("change_role")]
    public async Task<IActionResult> ChangeRole([FromQuery] int?
userId, [FromQuery] UserRole? roleId)
    {
        try
        {
            User editedUser = await db.Users.FirstOrDefaultAsync(u
=> u.Id == userId);
            if (editedUser == null) return NotFound();

            editedUser.Role = (UserRole)roleId;
            await db.SaveChangesAsync();
            return Ok();
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [Authorize(Roles = "Admin")]
    [HttpGet("users")]
    public async Task<IActionResult> GetUsers()
    {

```

```
try
{
    return Ok(await db.Users.Select(u => new UserDto()
    {
        Id = u.Id,
        Name = u.Name,
        Surname = u.Surname,
        PhotoUrl = PhotoService.ConvertToBase64(u.Photo,
"png"),
        Email = u.Email,
        PhoneNumber = u.PhoneNumber,
        Role = u.Role,
        BlockedStatus = u.BlockedStatus,
    }).ToListAsync());
}
catch (Exception ex)
{
    return BadRequest(ex.Message);
}
}
```

Приложение Е

Скрипт создания базы данных

```
CREATE TABLE `Roles` (  
  `ID` TINYINT PRIMARY KEY AUTO_INCREMENT,  
  `Name` VARCHAR(255) UNIQUE NOT NULL  
);  
  
CREATE TABLE `Users` (  
  `ID` INT PRIMARY KEY AUTO_INCREMENT,  
  `Email` VARCHAR(255) UNIQUE NOT NULL,  
  `Password` VARCHAR(255) NOT NULL,  
  `Role` TINYINT DEFAULT 1,  
  `BlockedStatus` BOOLEAN DEFAULT 0,  
  `Name` VARCHAR(255),  
  `Surname` VARCHAR(255),  
  `PhoneNumber` VARCHAR(255),  
  `Photo` LONGBLOB,  
  FOREIGN KEY (`Role`) REFERENCES `Roles`(`ID`)  
);  
  
CREATE TABLE TourTypes (  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  NAME VARCHAR(255) UNIQUE NOT NULL,  
  Image LONGBLOB  
);  
  
CREATE TABLE CharacteristicTypes (  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  NAME VARCHAR(255) UNIQUE NOT NULL  
);  
  
CREATE TABLE Characteristics (  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  CharacteristicTypeID INT,  
  NAME VARCHAR(255) UNIQUE NOT NULL,  
  FOREIGN KEY (`CharacteristicTypeID`) REFERENCES  
`CharacteristicTypes`(`ID`)  
);  
  
CREATE TABLE Characteristics_TourTypes (  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  CharacteristicID INT,  
  TourTypeID INT,  
  FOREIGN KEY (`CharacteristicID`) REFERENCES  
`Characteristics`(`ID`),  
  FOREIGN KEY (`TourTypeID`) REFERENCES `TourTypes`(`ID`)  
);  
  
CREATE TABLE Regions (  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  Name VARCHAR(255) UNIQUE NOT NULL,  
  Image LONGBLOB
```



```

);

CREATE TABLE Countries (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) UNIQUE NOT NULL,
    Flag LONGBLOB,
    RegionId INT,
    FOREIGN KEY (`RegionId`) REFERENCES `Regions`(`ID`)
);

CREATE TABLE Cities (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) UNIQUE NOT NULL,
    CountryId INT,
    FOREIGN KEY (`CountryId`) REFERENCES `Countries`(`ID`)
);

CREATE TABLE DepartmentDepartures (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) UNIQUE NOT NULL,
    CityId INT,
    FOREIGN KEY (`CityId`) REFERENCES `Cities`(`ID`)
);

CREATE TABLE TransportTypes (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) UNIQUE NOT NULL
);

CREATE TABLE Hotels (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) UNIQUE NOT NULL,
    StarsNumber INT NOT NULL,
    CityId INT,
    FOREIGN KEY (`CityId`) REFERENCES `Cities`(`ID`)
);

CREATE TABLE NutritionTypes (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) UNIQUE NOT NULL
);

CREATE TABLE Tours (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) UNIQUE NOT NULL,
    MainDescription VARCHAR(255),
    TourTypeId INT,
    NutritionTypeId INT,
    HotelId INT,
    Photo LONGBLOB
);

CREATE TABLE Descriptions (

```

```

        ID INT PRIMARY KEY AUTO_INCREMENT,
        Value BOOLEAN DEFAULT 0 NOT NULL,
        CharacteristicID INT,
        TourID INT,
        FOREIGN KEY (`CharacteristicID`) REFERENCES
`Characteristics`(`ID`),
        FOREIGN KEY (`TourID`) REFERENCES `Tours`(`ID`)
);

CREATE TABLE Reviews (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    UserId INT,
    TourId INT,
    ReviewText VARCHAR(255),
    FOREIGN KEY (`UserId`) REFERENCES `Users`(`ID`),
    FOREIGN KEY (`TourId`) REFERENCES `Tours`(`ID`)
);

CREATE TABLE Routes (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    LandingDateOfDeparture DATE,
    LandingTimeOfDeparture TIME,
    ArrivalDateOfDeparture DATE,
    ArrivalTimeOfDeparture TIME,
    LandingDateOfReturn DATE,
    LandingTimeOfReturn TIME,
    ArrivalDateOfReturn DATE,
    ArrivalTimeOfReturn TIME,
    Price INT,
    SeatsNumber INT,
    DepartmentDepartureId INT,
    TransportTypeId INT,
    TourId INT,
    FOREIGN KEY (`DepartmentDepartureId`) REFERENCES
`DepartmentDepartures`(`ID`),
    FOREIGN KEY (`TransportTypeId`) REFERENCES
`TransportTypes`(`ID`),
    FOREIGN KEY (`TourId`) REFERENCES `Tours`(`ID`)
);

CREATE TABLE Bookings (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    OrderSeatsNumber INT,
    UserId INT,
    RouteId INT,
    Status BOOLEAN DEFAULT 0,
    FOREIGN KEY (`UserId`) REFERENCES `Users`(`ID`),
    FOREIGN KEY (`RouteId`) REFERENCES `Routes`(`ID`)
);

```