*Федеральное государственное бюджетное образовательное учреждение высшего образования*

*«Владимирский государственный университет*

*имени Александра Григорьевича и Николая Григорьевича Столетовых»*

*Кафедра информационных систем и программной инженерии*

***ПОЯСНИТЕЛЬНАЯ ЗАПИСКА***

***к курсовому проекту по дисциплине   
"Технологии программирования"***

***на тему***

*Проектирование и разработка программной системы   
информационной системы*

*«Помощник путешественника по Владимиру»»*

*Выполнил: студент гр. ИСТ-220*

*Емельянов В. А.*

*Принял: доц. кафедры ИСПИ*

*Вершинин В.В.*

*Владимир, 2022*

АННОТАЦИЯ

В данном курсовом проекте производилась разработка программной системы информационной системы «помощник путешественника по Владимиру». Проект состоит из 3 этапов, включающих описание предметной области, словарь предметной области, описание функциональных требований и приложения, состоящего из исходного кода программной системы и структуры таблиц БД.

В рамках выполнения курсового проектирования были освоены навыки разработки полноценных программных систем.

Курсовой проект представлен на 88 страницах, рисунков - 17, таблиц - 6, использованных источников - 3, приложений – 2, иллюстрационный материал на 2 листах формата А1.

In this course project, the development of the software system of the information system "assistant traveler in Vladimir" was carried out. The project consists of 3 stages, including a description of the subject area, a dictionary of the subject area, a description of functional requirements and an application consisting of the source code of the software system and the structure of the database tables.

As part of the course design, the skills of developing full-fledged software systems were mastered.

The course project is presented on 88 pages, drawings - 17, tables - 6, used sources - 3, applications - 2, illustrations on 2 sheets of A1 format.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc122657929)

[**1**  **ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ** 5](#_Toc122657930)

[1.1 Анализ 5](#_Toc122657931)

[1.2 Функционал системы 6](#_Toc122657932)

[1.3 Пользователи разрабатываемой системы 6](#_Toc122657933)

[1.4 Словарь предметной области 7](#_Toc122657934)

[**2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ** 8](#_Toc122657935)

[2.1 Диаграммы UML 8](#_Toc122657936)

[**2.1.1. Диаграммы прецедентов** 8](#_Toc122657937)

[**2.1.2 Описание прецедентов** 8](#_Toc122657938)

[2.2. Диаграмма классов 10](#_Toc122657939)

[2.3. Диаграмма последовательностей 11](#_Toc122657940)

[2.4. Диаграмма состояний 13](#_Toc122657941)

[**3 РЕАЛИЗАЦИЯ СИСТЕМЫ** 15](#_Toc122657942)

[3.1 Стек используемых технологий 15](#_Toc122657943)

[3.2. Используемые сторонние API 15](#_Toc122657944)

[3.3 Демонстрация программной системы 16](#_Toc122657945)

[**ЗАКЛЮЧЕНИЕ** 23](#_Toc122657946)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 24](#_Toc122657947)

[**ПРИЛОЖЕНИЕ А** 25](#_Toc122657948)

[**ПРИЛОЖЕНИЕ Б** 87](#_Toc122657949)

## ВВЕДЕНИЕ

В ходе выполнения курсового проекта требовалось разработать программную систему информационной системы «помощник путешественника по Владимиру»

Целью курсового проектирования является изучение процесса разработки сложных систем на примере конкретной программной системы, определяемой тематикой курсового проекта.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучение особенностей конкретной предметной области, относящихся к теме курсового проекта;
* анализ возможных подходов и методов решения с обоснованием выбранного подхода;
* разработка моделей (математической, структурной, информационной, данных) программной системы;
* выбор алгоритмов с учетом их точности и спецификой предметной области;
* разработка программного обеспечения;
* анализ полученных результатов работы ПО.

# **1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

## 1.1 Анализ

Одна из ключевых функций - интерактивная динамическая карта, на которой расположены точки со всеми объектами для посещения, например, достопримечательности, заведения, интересные места. Управлением каждого объекта занимаются конкретные люди – владельцы или уполномоченные лица. У каждого объекта присутствуют отзывы. Реализована система авторизации пользователей. Для пользователей со статусом аккаунта Владелец разработан функционал добавления точек интереса при помощи заполнения формы с данными и контактами. После модерирования объект добавляют на сайт. Для улучшения основного аккаунта до владельца заведения необходимо заполнить форму с данными, которую в последствии рассмотрит и подтвердит администратор системы.

Для каждого объекта авторизованный путешественник может создать встречу для совместного посещения данного объекта. Чат создается автоматически в мессенджере Telegram. Приглашение происходит по ссылке.

## 1.2 Функционал системы

Программная система «Помощник путешественника по Владимиру» выполняет:

* просмотр различных объектов, которые можно будет посетить в будущем;
* предоставление динамической карты с точками всех объектов в системе;
* предоставление возможности создание встреч для совместного посещения объектов.

## 1.3 Пользователи разрабатываемой системы

1) **Путешественник** – человек, авторизовавшийся в программной системе. Присутствует возможность оставлять отзывы к заведениям, создавать встречи для совместного посещения. Есть возможность улучшить аккаунт до владельца.

2) **Гость** – неавторизованный путешественник системы. Ограничен функционал взаимодействия с системой. Возможен только просмотр объектов

3) **Владелец** – особый пользователь системы, подтвержденный модераторами сервиса. По мимо функций обычного пользователя присутствует возможность добавления и модерирование своих заведений.

4) **Администратор** – человек, в чьи обязанности входит отслеживание новых заявок по заведениям, модерирование отзывов, слежение за выполнением правил сервиса

## Словарь предметной области

1) Личный кабинет – особый раздел на сайте, где пользователи могут просматривать, редактировать и в зависимости от типа аккаунта выполнять различные действия.  
2) Точка интереса (объект, заведение) – место, которое можно посетить одному или с участниками группы, ранее созданной в мессенджере Telegram. У каждого объекта есть свой рейтинг и отзывы для более информативного ознакомления.

3) Встреча – мероприятие для совместного посещения объекта или заведения. Привязывается к конкретной дате и к конкретному месту. Для удобства общения, организации и сбора людей создается автоматически беседа в мессенджере Telegram и выдается ссылка для вступления.

**4) Интерактивная карта –** динамическая карта, на которой будут расположены различные точки для посещения. По нажатию на точку можно будет перейти в ее детальное описание.

**5) Отзыв –** словесное описание от посещенного объекта или заведения. Может быть удален администратором

# **2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ**

## 2.1 Диаграммы UML

**2.1.1. Диаграммы прецедентов**

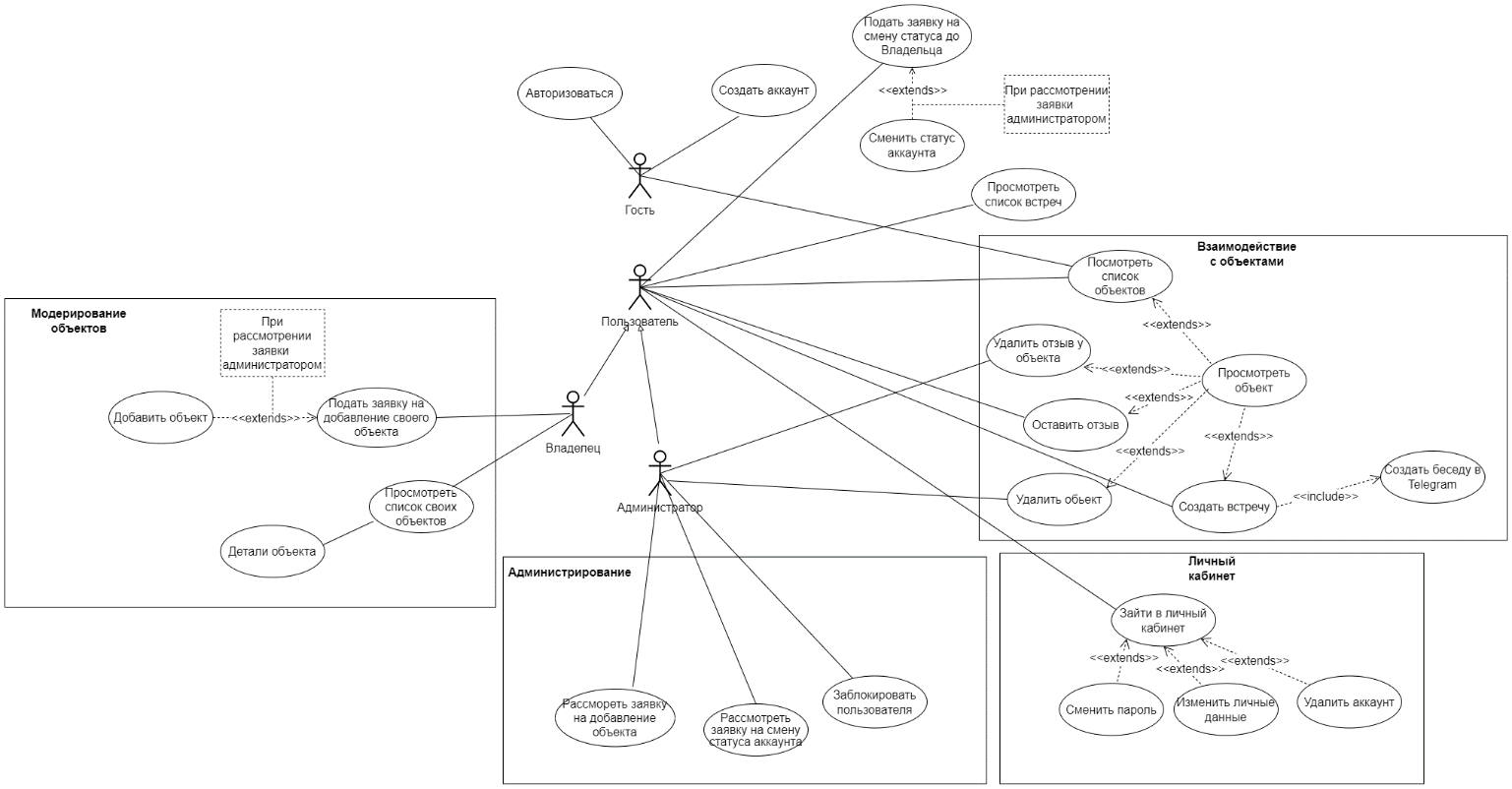
В ходе проектирования программной системы была разработана диаграмма прецедентов, представленная на рисунке 1.

Рисунок №1. Диаграмма прецедентов.

### **2.1.2 Описание прецедентов**

*Название:* Создать встречу

*Актер:* Путешественник или владелец

*Предусловие:* Человек авторизовался в системе и выбрал объект из общего списка

*Основной поток:* Создание сущности встреча

Выполняется создание встречи и производится привязывание этой сущности к создавшему пользователю.

*Альтернативный поток:* Ошибка обработки запроса сервером.

*Постусловие:* Создание беседы в мессенджере Телеграм и предоставлении ссылки на вступление.

*Название*: Оставить отзыв

*Актер:* Путешественник или Владелец

*Предусловие:* Участник системы авторизовался и выбрал конкретный объект или заведение и перешел в его детали.

*Основной поток:* Написание отзыва.

Путешественник выражает свои мысли, эмоции и оценку от посещенного мероприятия.

*Альтернативный поток:* Ошибка обработки запроса сервером.

*Постусловие:* Обновление текущей страницы сайта

*Название*: Подать заявку на смену статуса до владельца

*Актер:* Путешественник

*Предусловие:* Участник системы авторизован

*Основной поток:* Подача заявки на смену статуса.

Путешественник, захотевший сменить свой статус аккаунта до владельца для добавления своих объектов или заведений, подает необходимую юридическую информацию и отправляет заявку на рассмотрение администратором.

*Альтернативный поток:*

Ошибка обработки запроса сервером;

Отклонение заявки.

*Постусловие:* Переадресация человека на главную страницу сайта.

*Название*: Подать заявку на добавление своего объекта.

*Актер:* Владелец

*Предусловие:* Участник системы авторизован

*Основной поток:* Подача заявки на добавление объекта.

Владелец, решивший добавить свой объект или заведение, заполняет особую форму с данными и посылает заявку на рассмотрение администратором

*Альтернативный поток:*

Ошибка обработки запроса сервером;

Отклонение заявки.

*Постусловие:* Переадресация человека на главную страницу сайта.

## 2.2. Диаграмма классов

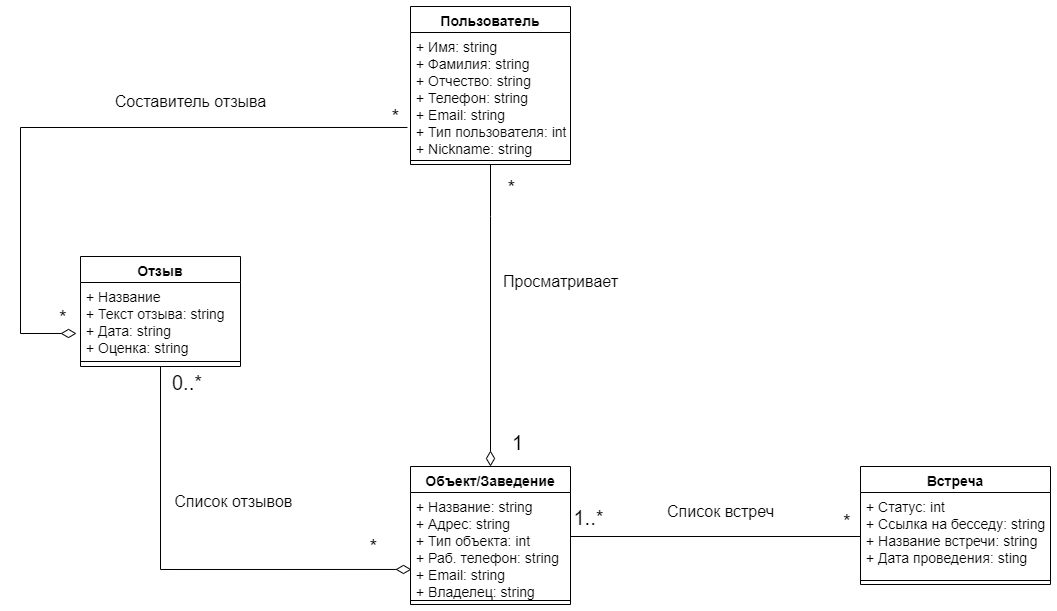
В ходе проектирования системы была разработана диаграмма классов на рисунке 2.

Рисунок №2. Диаграмма классов

## 2.3. Диаграмма последовательностей

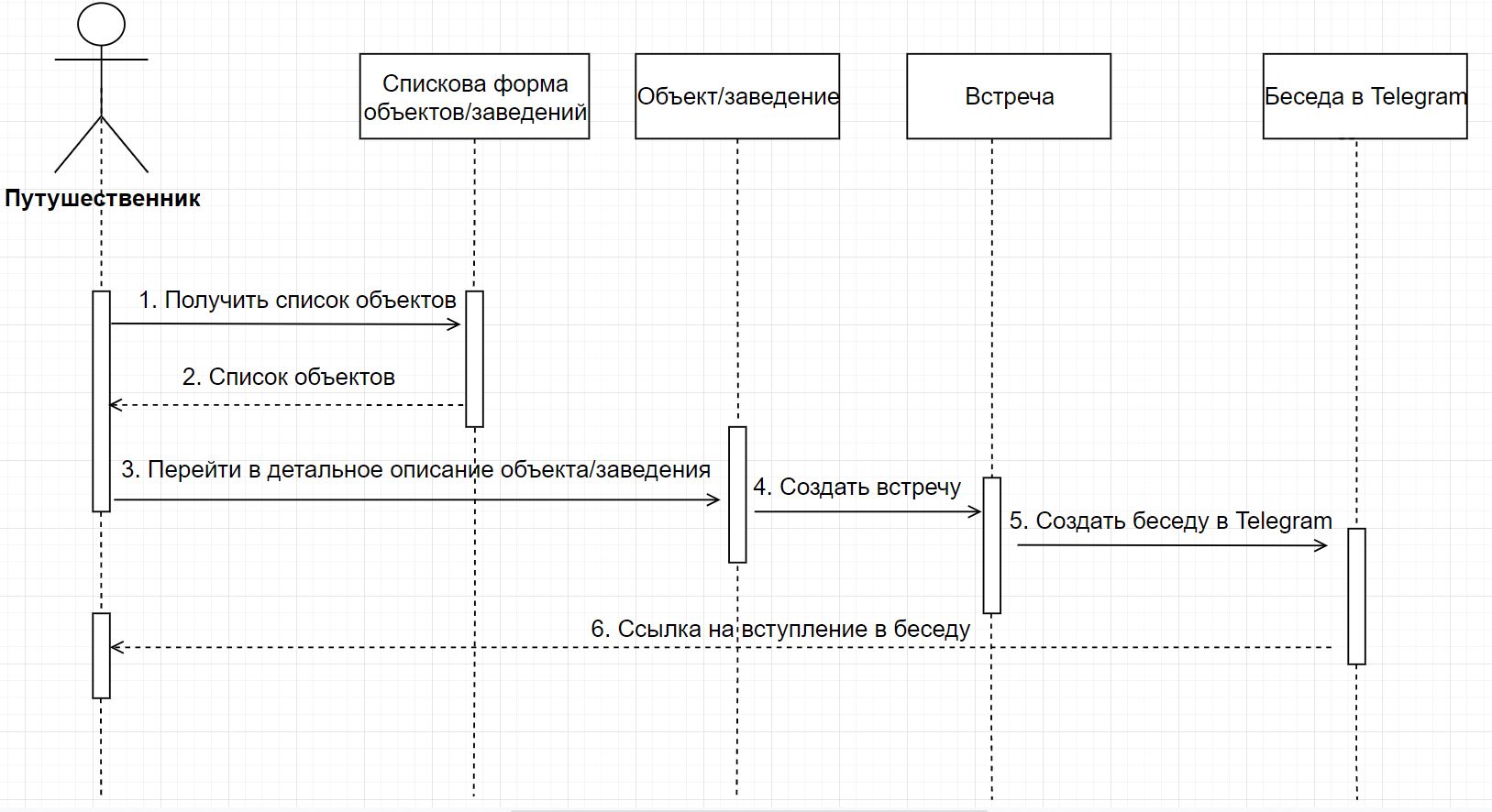
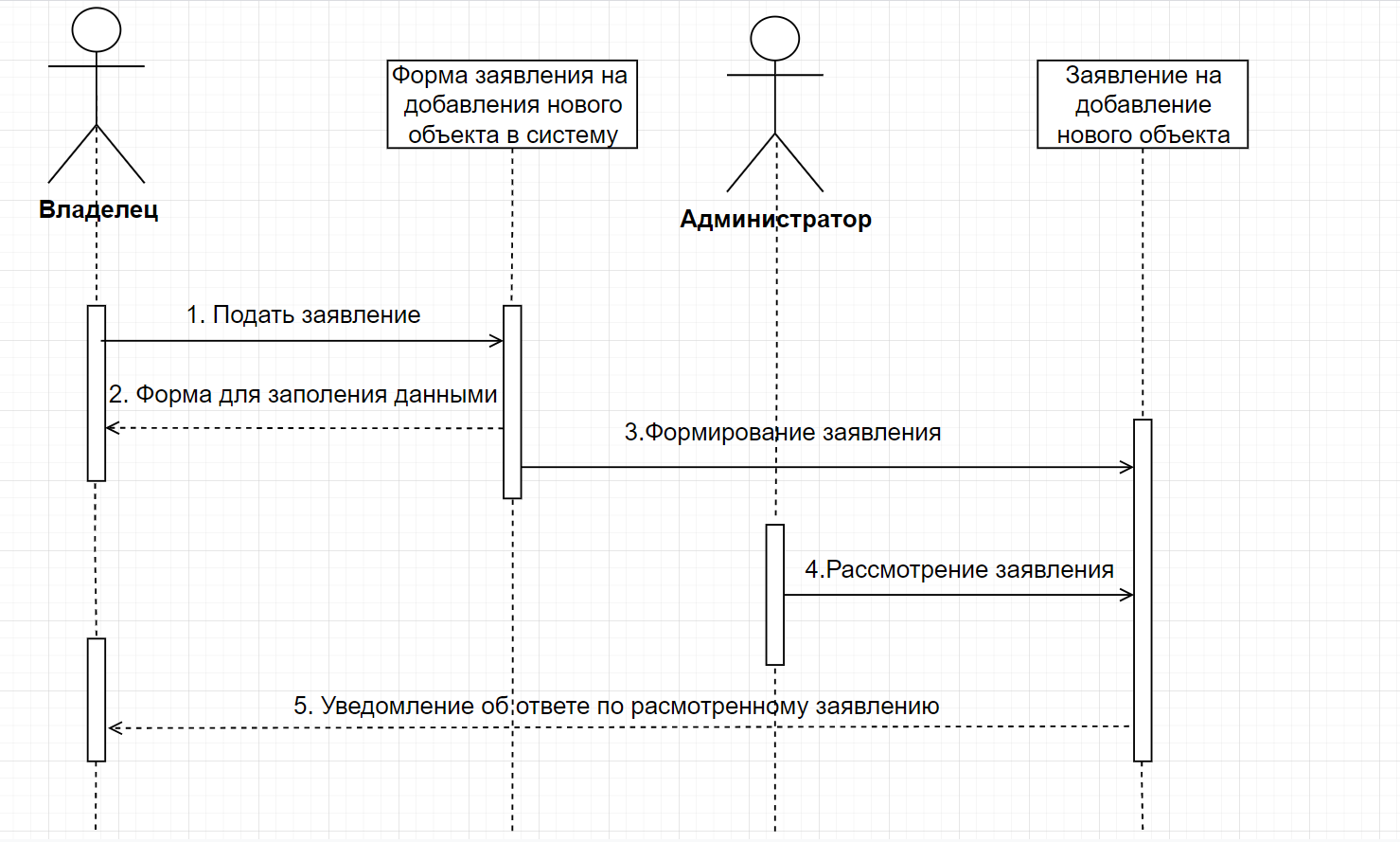
В ходе проектирование системы были разработаны основные диаграммы последовательности, описывающие основные взаимодействия между объектами системы и актерами. Данные диаграммы представлены на рисунках 3, 4, 5

Рисунок №3. Диаграмма последовательности для прецедента

«Создать встречу».

Рисунок №4. Диаграмма последовательности для прецедента

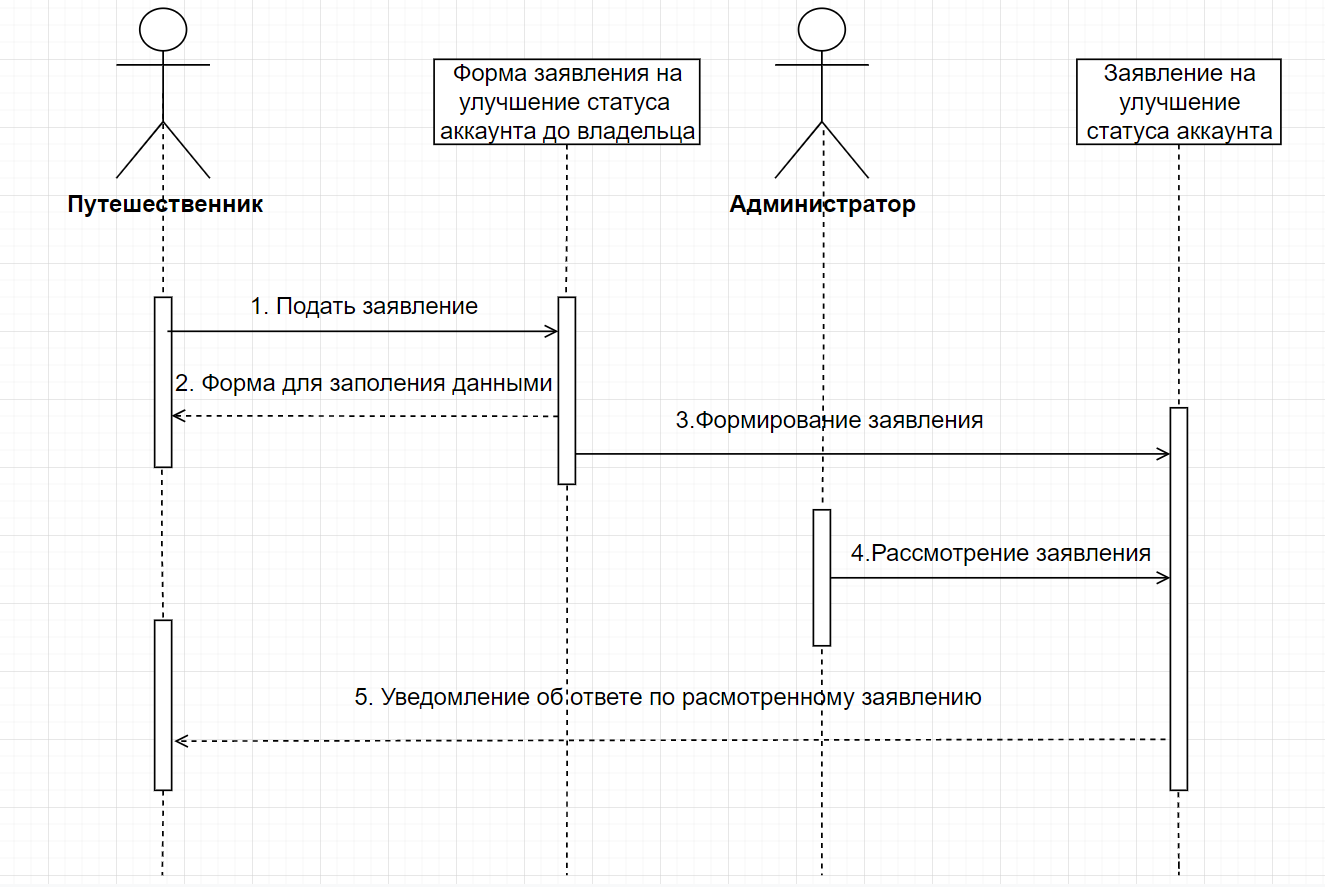
«Подать заявку на добавление своего объекта».

Рисунок №5. Диаграмма последовательности для прецедента

«Подать заявку на смену статуса до Владельца».

## 2.4. Диаграмма состояний

В ходе проектирование системы были разработаны диаграммы состояний, описывающие жизненные циклы объектов системы. Данные диаграммы представлены на рисунках 6, 7.

Рисунок №6. Диаграмма состояния жизненного цикла

объекта «Заявка на добавление нового объекта в систему» и

объекта «Заявка на улучшение статуса аккаунта до Владельца».

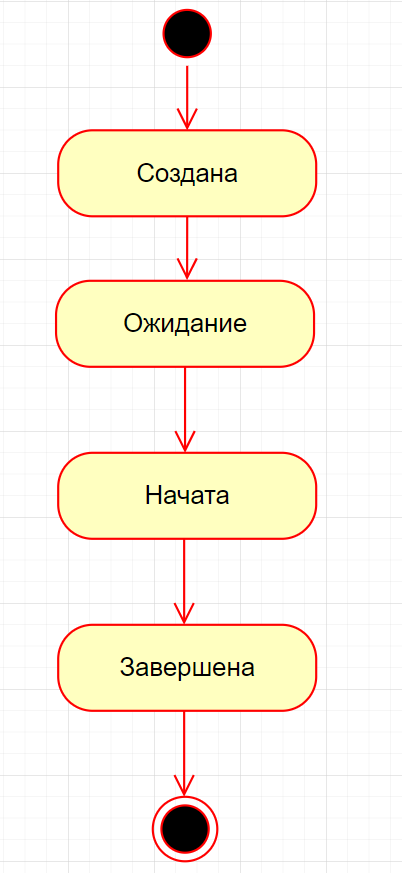


Рисунок №7. Диаграмма состояния жизненного цикла

объекта «Встреча».

# **3 РЕАЛИЗАЦИЯ СИСТЕМЫ**

**3.1 Стек используемых технологий**

Для реализации программной системы была выбрана концепция разработки приложения MVC. Основной идеей MVC является разделение приложения на 3 отдельных компонента: пользовательский интерфейс, контроллер, интерпретирующий действия пользователя и модель, которая предоставляет данные и реагирует на команды контроллера.

Для реализации компонентов программной системы следующие технологии:

* Основной платформой для написания бэкенда был использован фреймворк .NET Core 6.0. Эта платформа представляет многочисленный функционал, который доступен в формате «из коробки»,
* При разработке фронтэнд части использовался фреймворк Bootstrap. Данный фреймворк позволяет верстать быстрые и адаптивные под разное разрешение веб-страницы,
* В качестве источника хранения данных была использована СУБД MySQL, поскольку она является довольно популярной и имеет достаточно быструю производительность;

## 3.2. Используемые сторонние API

Для того чтобы самому не реализовывать сложные системы и компоненты различными разработчиками пишутся API – Application Programming Interface. По своей сути API является точкой соприкосновения одной компьютерной программы с другой или несколькими сразу. С помощью API можно не беспокоиться о создании некоторых компонентов системы самому, например, не писать свой сервис для прогноза погоды, а использовать готовый API сервиса с погодой.

Для реализации задуманного были использованы следующие API:

* Yandex Maps JavaScript API – программная библиотека для работы с динамическими и интерактивными картами в браузере. Данный API применяется при добавлении объектов в систему, выводе общей карты с точками всех объектов, присутствующих в системе,
* Telegram API – API, позволяющий построить свой пользовательский Telegram клиент, имеет полный функциональный набор возможностей всеми привычного мессенджера. Данный API применяется для автоматизированного создания чата;

## 3.3 Демонстрация программной системы

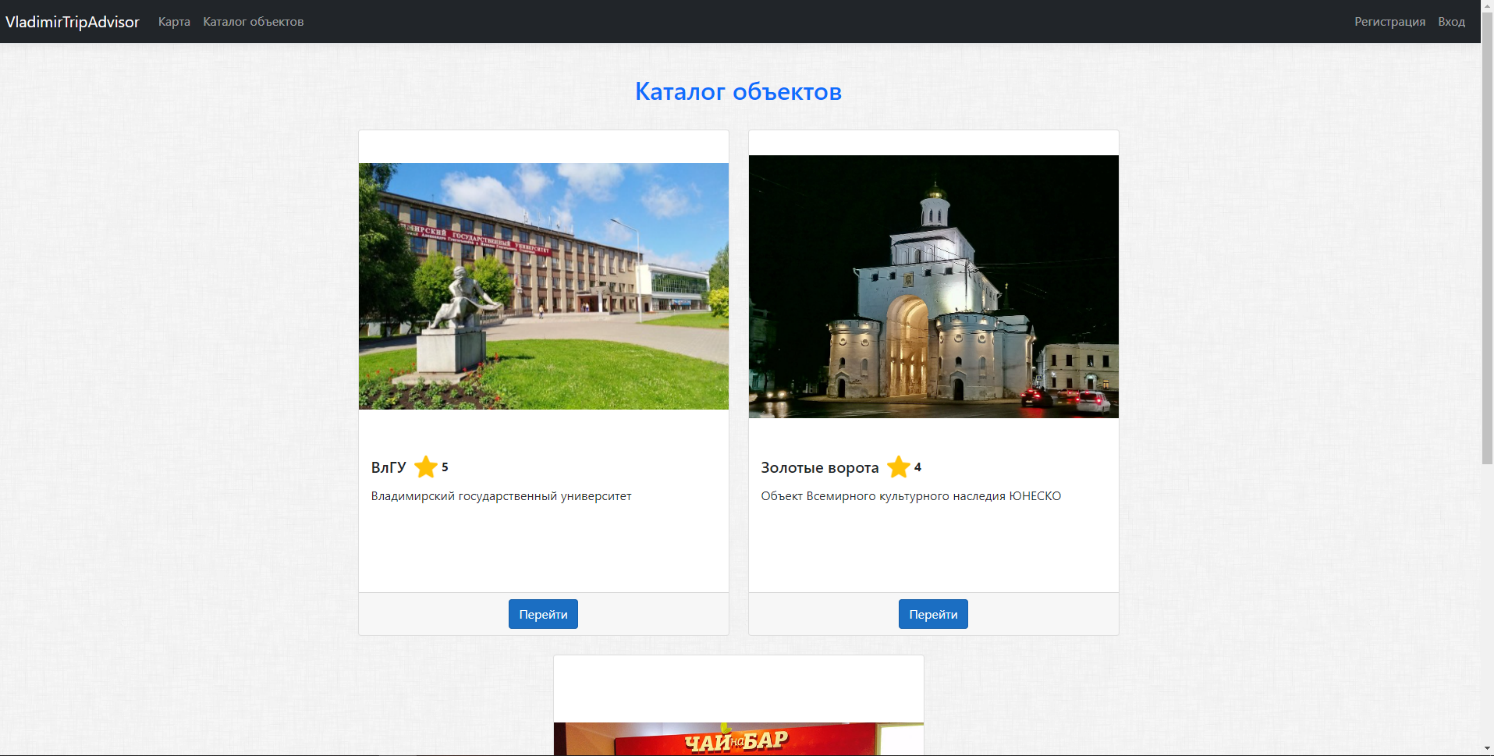
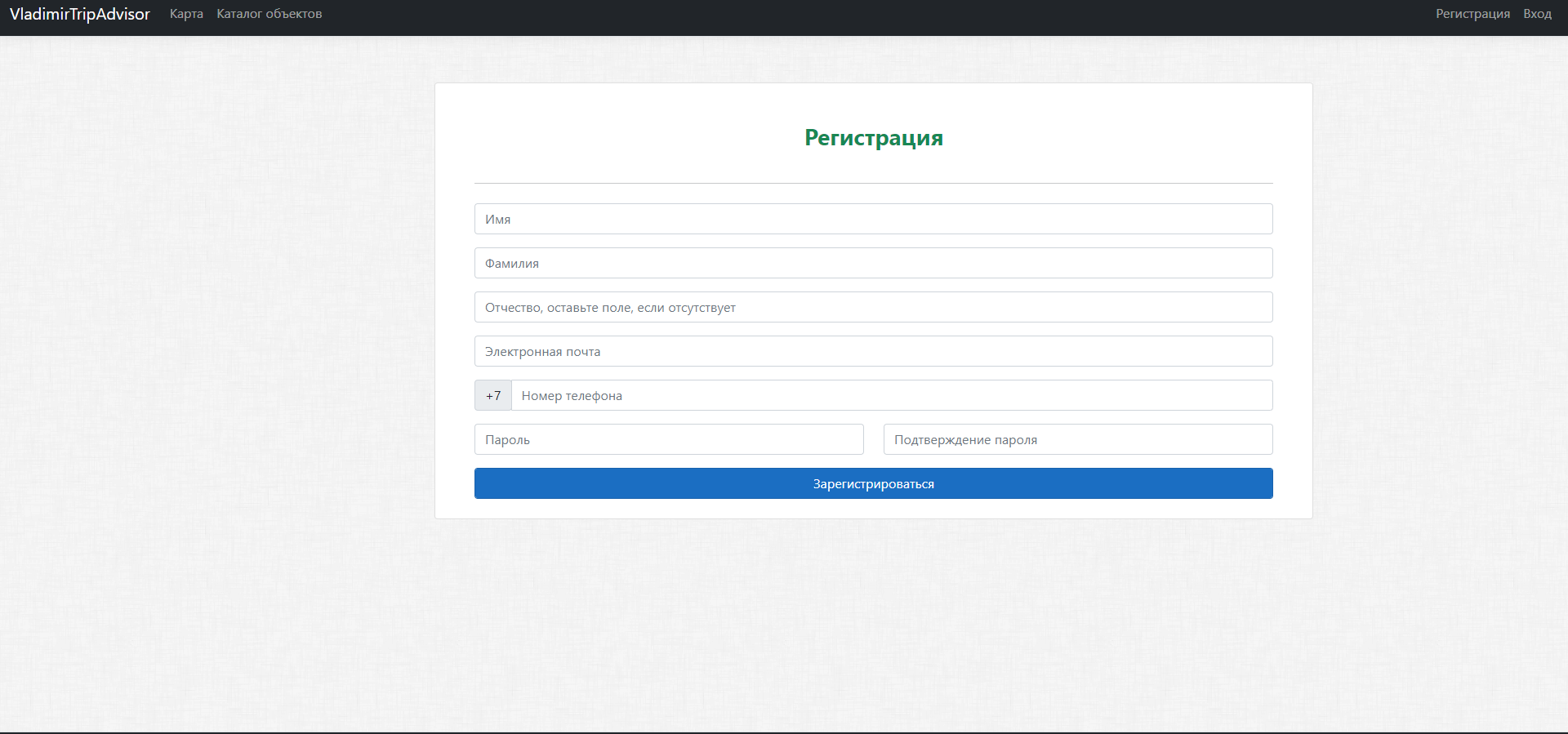
****Демонстрация работы программной системы представлены на рисунках 8-

Рисунок №8. Главная страница

Рисунок №9. Форма регистрации

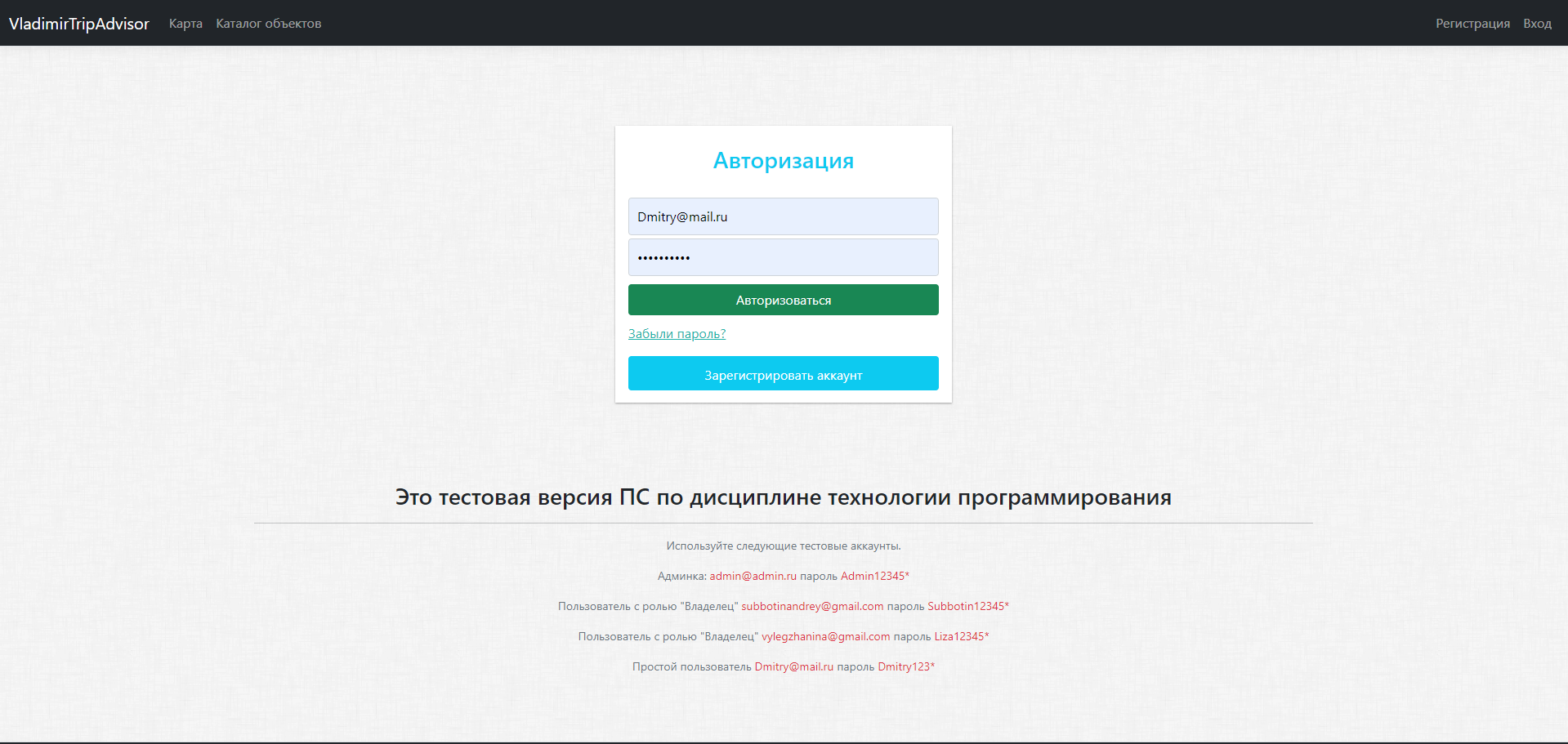


Рисунок №10. Форма с авторизацией

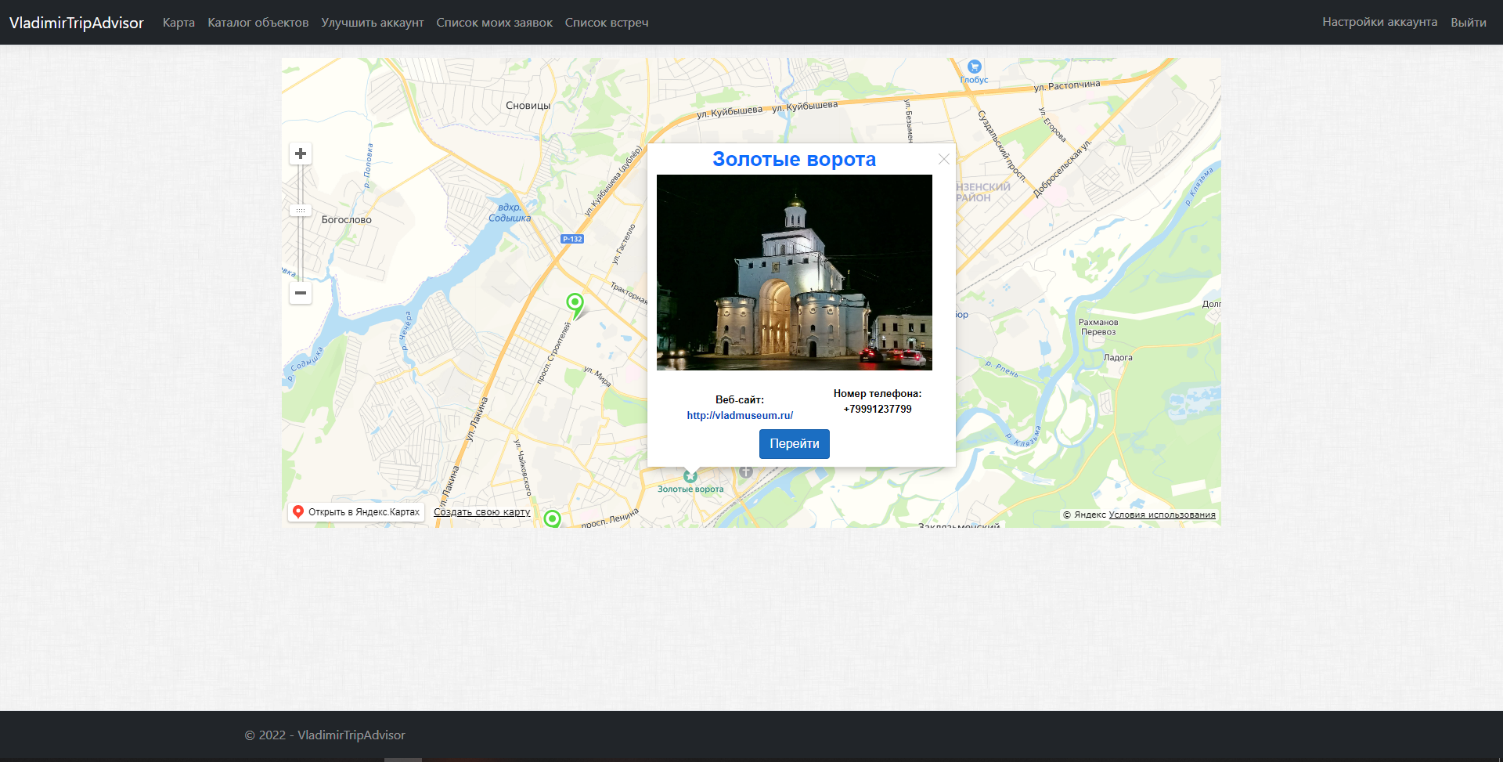
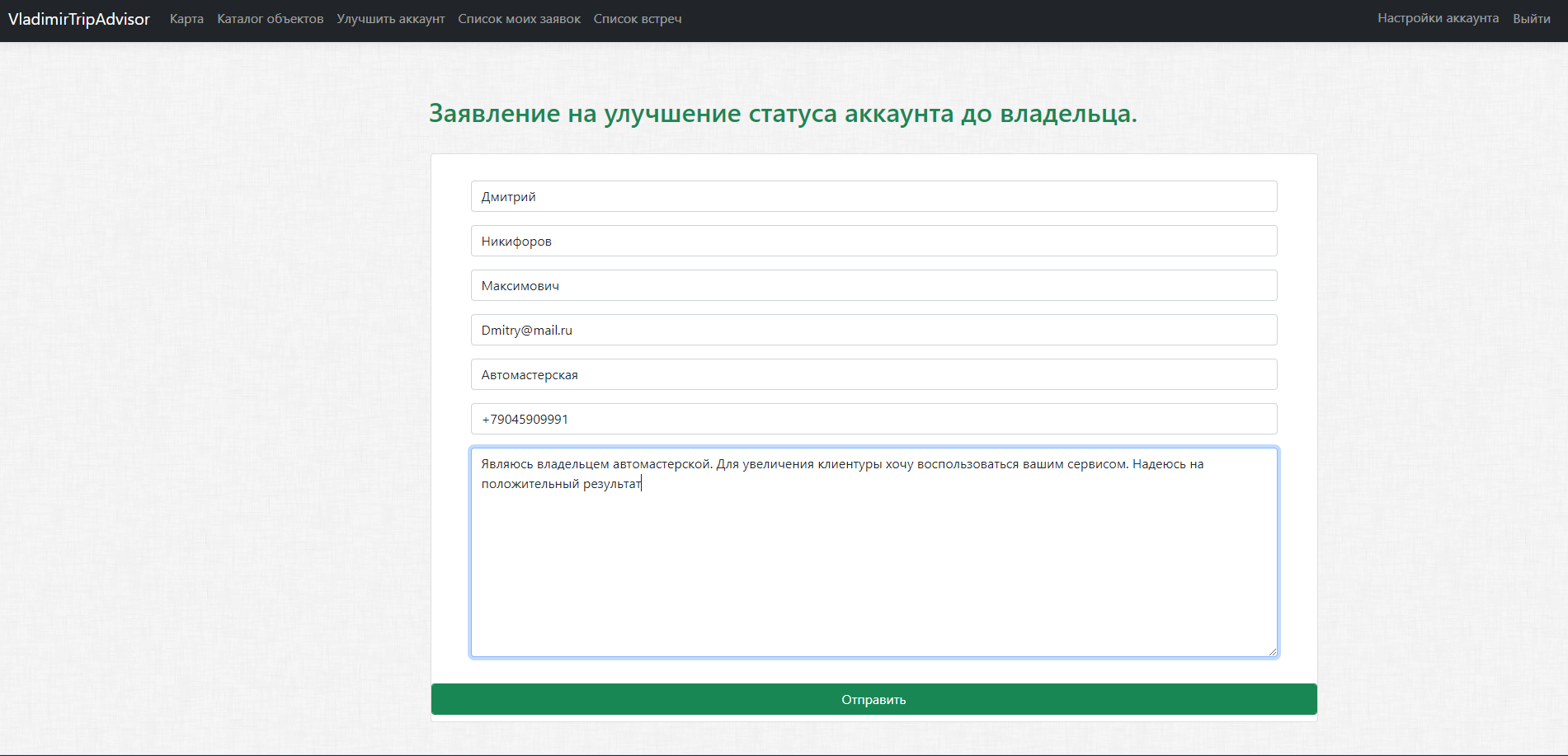
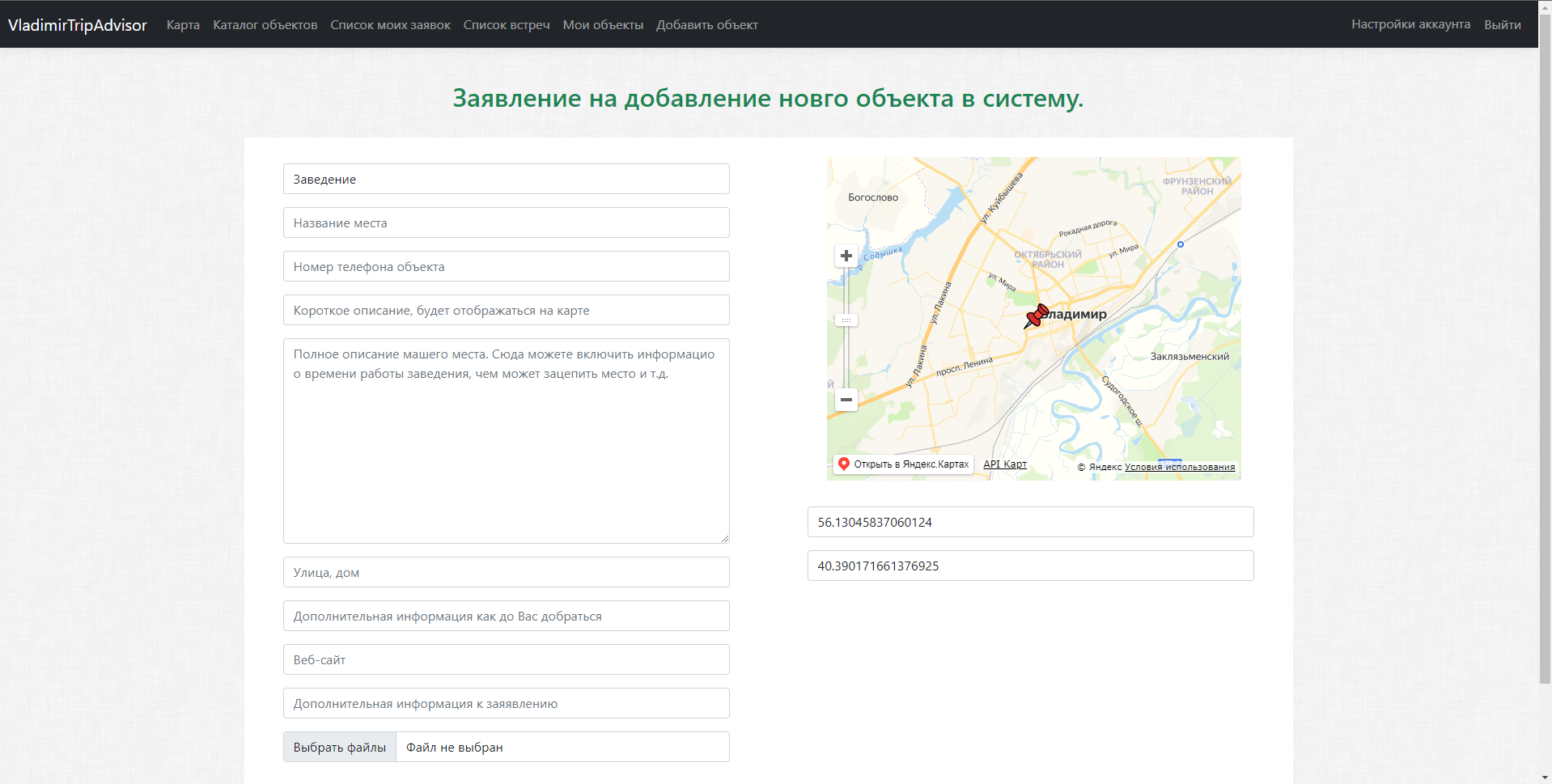
Рисунок №11. Карта с точками всех объектов

Рисунок №12. Заявление на улучшение статуса аккаунта.

Рисунок №13. Форма заявления на добавление нового объекта.

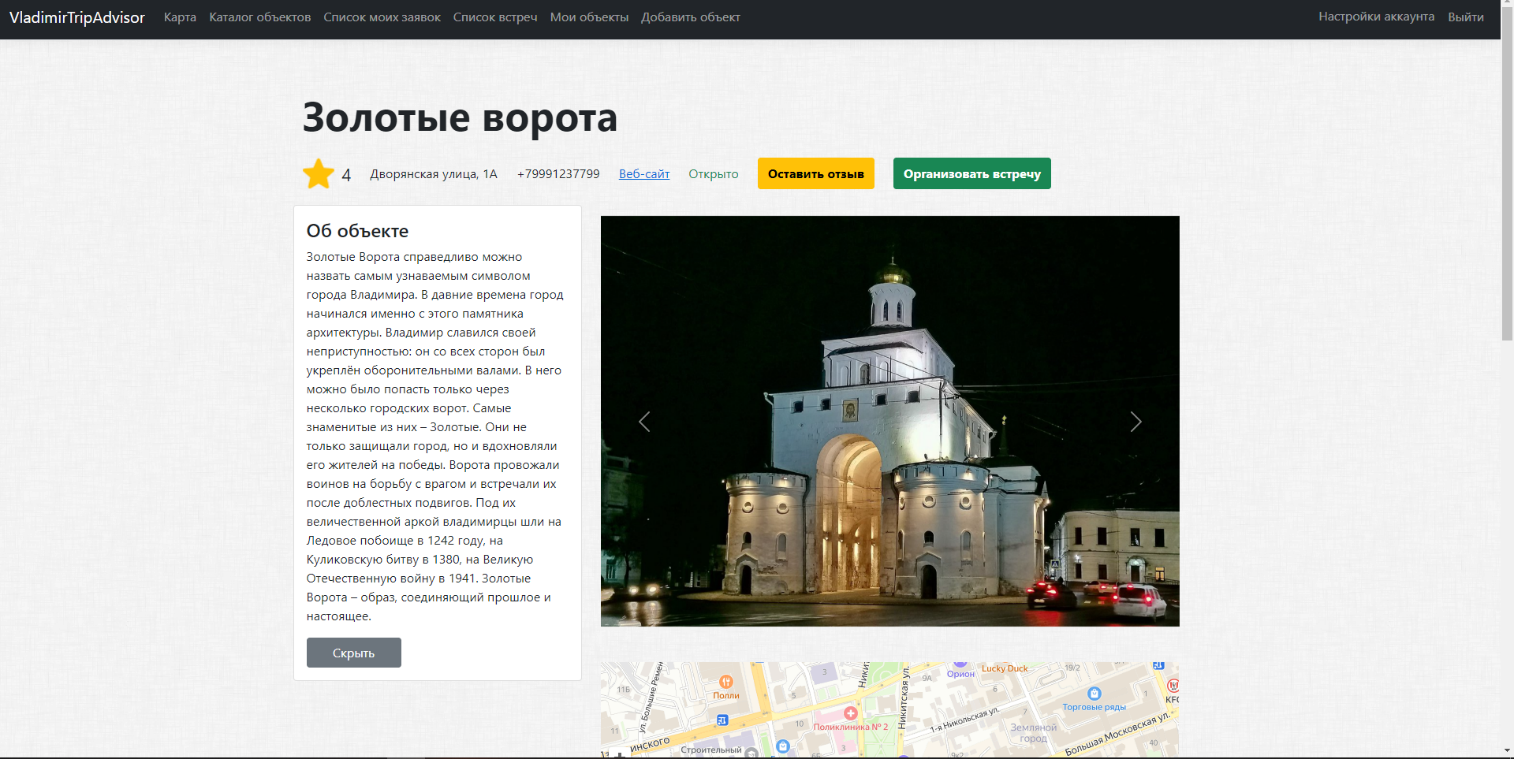
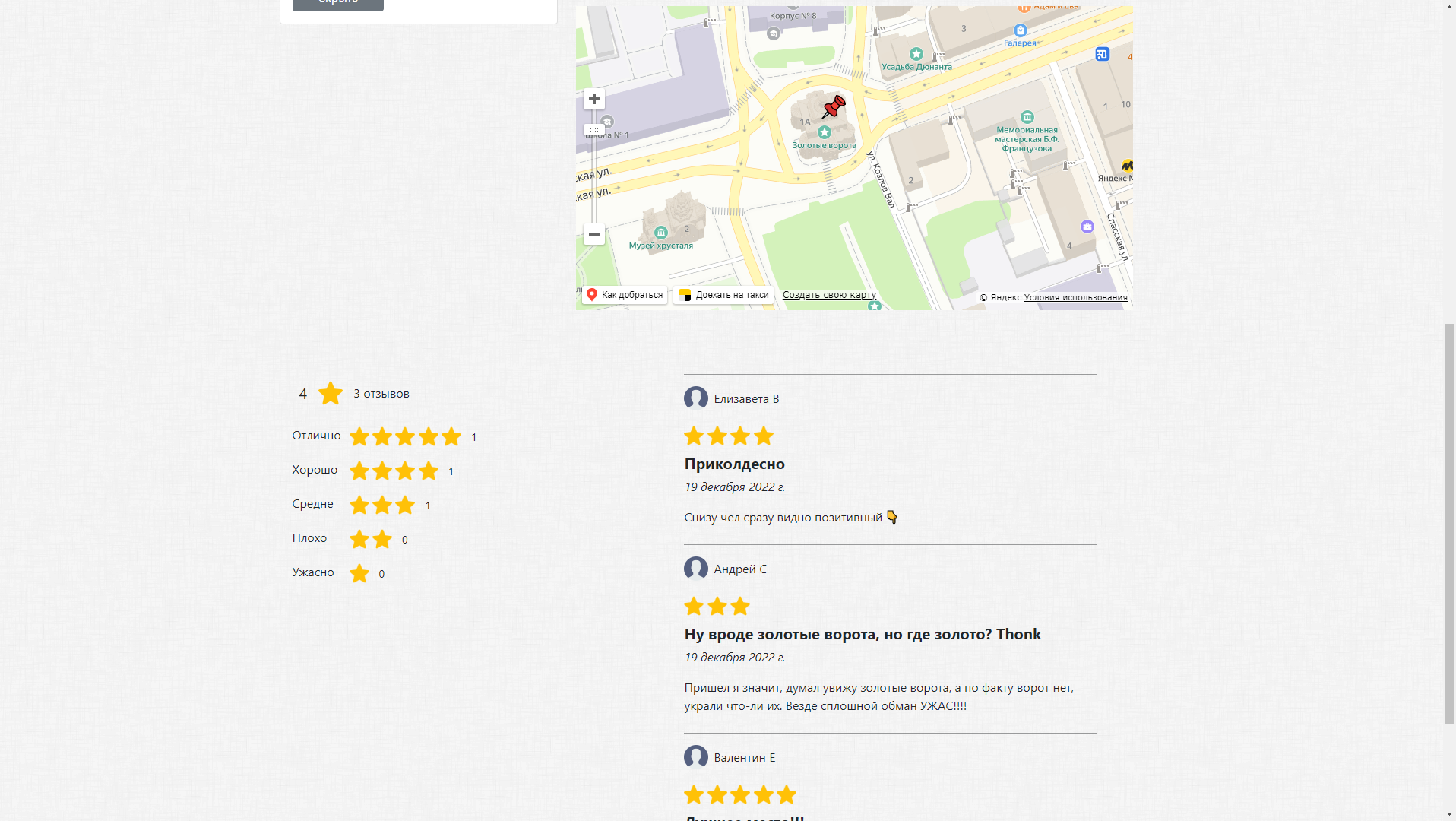


Рисунок №14. Детальный просмотр объекта

Рисунок №15. Секция с расположением объекта на карте и списком отзывов

от пользователей

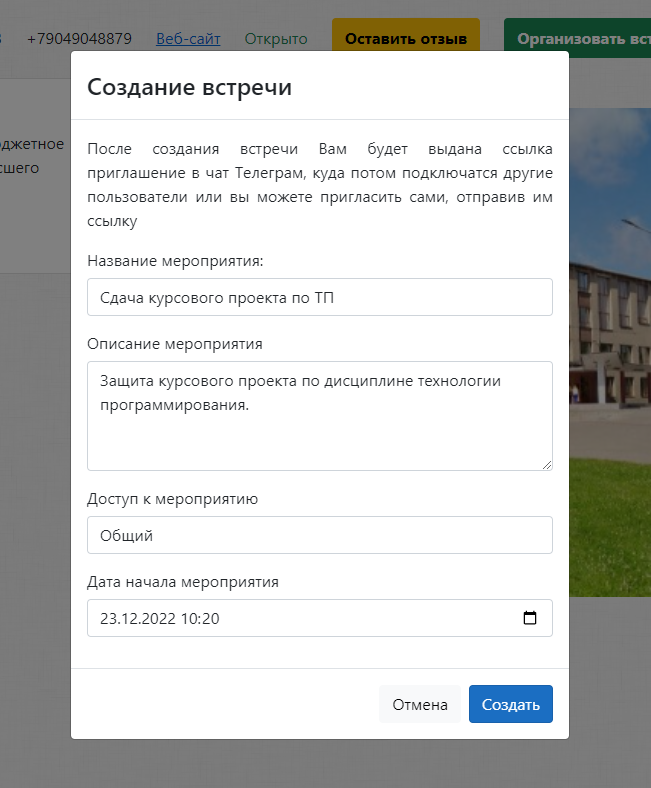


Рисунок №16. Форма создания встречи.

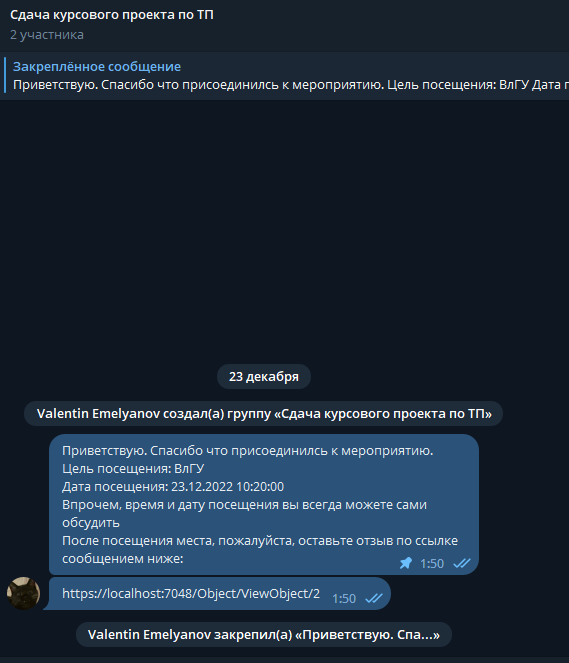


Рисунок №17. Создание чата и автоматическая отправка сообщений

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта был произведен анализ предметной области, определены основные процессы, происходящие в системе, спроектирована подсистема хранения данных, разработан дизайн пользовательского интерфейса, написаны функциональные возможности.

Были разработаны страницы пользовательского интерфейса – форма регистрации, форма авторизации, список всех объектов в системе, заявление на создание новых объектов в системе, заявление на улучшение статуса аккаунта, детальный просмотр объекта, карта с точками всех объектов в системе.

Была спроектирована подсистема хранения данных и описана в виде ER-диаграмм в приложении Б.

Данная система имеет базовые функциональные возможности. В дальнейшем планируется работа над данной программной системой, улучшение пользовательского интерфейса, разработка новых возможностей, проведение рефакторинга кода с целью улучшения производительности и читаемости и развертывание на сервер для всех пользователей.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Документация .NET [Электронный ресурс]. – URL: <https://learn.microsoft.com/en-us/dotnet>. Режим доступа: Публичный. Дата обращения: 22.12.2022
2. Документация Yandex Maps JavaScript API [Электронный ресурс]. – URL: <https://yandex.ru/dev/maps/jsapi/doc/2.1/quick-start/index.html?from=jsapi> Режим доступа: Публичный. Дата обращения: 22.12.2022
3. Документация Telegram API [Электронный ресурс]. – URL: <https://core.telegram.org/methods> Режим доступа: Публичный. Дата обращения: 22.12.2022

# **ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММНОЙ СИСТЕМЫ**

**ApplicationModelDB.cs**

using VladimirTripAdvisor.Logic.Application;

namespace VladimirTripAdvisor.Models

{

public class ApplicationModelDB

{

[Key]

public long Id { get; set; }

[Column("application\_type")]

public ApplicationType ApplicationType { get; set; }

[Column("application\_status")]

public ApplicationStatus ApplicationStatus { get; set; }

[Column("application\_data")]

public string ApplicationData { get; set; }

[Column("id\_user")]

public string UserId { get; set; }

[ForeignKey("UserId")]

public virtual UserModel User { get; set; }

}

}

**EventModel.cs**

namespace VladimirTripAdvisor.Models

{

public class EventModel

{

[Key]

public long Id { get; set; }

[Column("name")]

[StringLength(maximumLength: 50)]

public string Name { get; set; }

[Column("description")]

[StringLength(maximumLength: 500)]

public string Description { get; set; }

[Column("start\_date")]

public DateTime StartDate { get; set; }

[Column("end\_date")]

public DateTime EndDate { get; set; }

[Column("event\_type")]

public EventType EventType { get; set; }

[Column("telegram\_link")]

[StringLength(maximumLength:500)]

public string TelegramLink { get; set; }

[Column("id\_creator")]

public string CreatorId { get; set; }

[ForeignKey("CreatorId")]

public virtual UserModel User { get; set; }

[Column("id\_place")]

public long PlaceId { get; set; }

[ForeignKey("PlaceId")]

public virtual ObjectOfVisitModel ObjectOfVisit { get; set; }

}

}

**ImageModel.cs**

namespace VladimirTripAdvisor.Models

{

[Table("image\_object")]

public class ImageModel

{

[Key]

public long Id { get; set; }

[Required]

[Column("image\_byte")]

public byte[] ImageByte { get; set; }

[Column("id\_application")]

public long ApplicationId { get; set; }

[ForeignKey("ApplicationId")]

public virtual ApplicationModelDB? Application { get; set; }

[Column("id\_object")]

public long? ObjectId { get; set; }

[ForeignKey("ObjectId")]

public virtual ObjectOfVisitModel? ObjectOfVisit { get; set; }

}

}

**ObjectOfVisitModel.cs**

using Microsoft.EntityFrameworkCore.Metadata.Internal;

namespace VladimirTripAdvisor.Models

{

[Table("object\_of\_visit")]

public class ObjectOfVisitModel

{

[Key]

public long Id { get; set; }

[Required(ErrorMessage = "Это поле обязательно")]

[StringLength(maximumLength:50)]

public string Name { get; set; }

[Required(ErrorMessage = "Это поле обязательно")]

[StringLength(maximumLength: 1500)]

[Column("place\_description")]

public string Description { get; set; }

[Required(ErrorMessage = "Это поле обязательно")]

[StringLength(maximumLength: 250)]

[Column("place\_short\_description")]

public string ShortDescription { get; set; }

[Column("place\_type")]

public PlaceType PlaceType { get; set; }

[StringLength(maximumLength: 50)]

public string Latitude { get; set; }

[StringLength(maximumLength: 50)]

public string Longitude { get; set; }

[StringLength(maximumLength: 150)]

[Column("street\_address")]

public string StreetAddress { get; set; }

[StringLength(maximumLength: 150)]

[Column("additional\_address\_info")]

public string? AdditionalAddressInfo { get; set; }

[Column("average\_rating")]

public float? AverageRating { get; set; }

[Phone]

[Column("phone\_number")]

public string? PhoneNumber { get; set; }

[StringLength(maximumLength: 150)]

[Column("place\_url")]

public string? PlaceURL { get; set; }

[Column("id\_owner")]

public string? IdOwner { get; set; }

[ForeignKey("IdOwner")]

public UserModel? User { get; set; }

}

}

**ReviewModel.cs**

using System.ComponentModel.DataAnnotations.Schema;

namespace VladimirTripAdvisor.Models

{

public class ReviewModel

{

public ReviewModel()

{

ReviewDate = DateTime.Now;

}

[Key]

public long Id { get; set; }

[Column("review\_name")]

[StringLength(maximumLength: 50)]

[Required(ErrorMessage = "Данное поле обязательно")]

public string ReviewName { get; set; }

[Column("review\_description")]

[StringLength(maximumLength:1000)]

[Required(ErrorMessage ="Данное поле обязательно")]

public string ReviewDescription { get; set; }

[Column("score")]

public ReviewScore Score { get; set; }

[Column("review\_date")]

[Required(ErrorMessage ="Данное поле обязательно")]

public DateTime ReviewDate { get; set; }

[Column("id\_object")]

public long ObjectId { get; set; }

[ForeignKey("ObjectId")]

public virtual ObjectOfVisitModel? ObjectOfVisit { get; set; }

[Column("id\_user")]

public string UserId { get; set; }

[ForeignKey("UserId")]

public virtual UserModel? User { get; set; }

}

}

**UserModel.cs**

using Microsoft.AspNetCore.Identity;

using System.ComponentModel.DataAnnotations;

namespace VladimirTripAdvisor.Models

{

public class UserModel : IdentityUser

{

[StringLength(maximumLength:50)]

public string Name { get; set; }

[StringLength(maximumLength: 50)]

public string Surname { get; set; }

[StringLength(maximumLength: 50)]

public string? Midname { get; set; }

}

}

**ApplicationAccountViewModel.cs**

using VladimirTripAdvisor.Logic.Application;

namespace VladimirTripAdvisor.ViewModels

{

public class ApplicationAccountViewModel

{

public string UserID { get; set; }

public long? ApplicationId { get; set; }

public ApplicationStatus? ApplicationStatus { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

[MaxLength(50, ErrorMessage = "Данное поле не должно превышать 50 символов")]

public string? Name { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

[MaxLength(50, ErrorMessage = "Данное поле не должно превышать 50 символов")]

public string? Surname { get; set; }

[MaxLength(50, ErrorMessage = "Данное поле не должно превышать 50 символов")]

public string? Midname { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

[MaxLength(12, ErrorMessage = "Данное поле не должно превышать 50 символов")]

[Phone]

public string? PhoneNumber { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

[EmailAddress]

public string? Email { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public string? CompanyName { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public string? AdditionalInfo { get; set; }

public string? AdminComment { get; set; }

}

}

**ApplicationObjectViewModel.cs**

using System.Buffers.Text;

using System.Collections.Generic;

using VladimirTripAdvisor.Logic.Application;

namespace VladimirTripAdvisor.ViewModels

{

public class ApplicationObjectViewModel

{

public string UserID { get; set; }

public long? ApplicationId { get; set; }

public ApplicationStatus? ApplicationStatus { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

[StringLength(maximumLength: 50)]

public string? Name { get; set; }

[StringLength(maximumLength: 1500)]

[Required(ErrorMessage = "Данное поле обязательно")]

public string? Description { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

[StringLength(maximumLength: 250)]

public string? ShortDescription { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public string? Latitude { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public string? Longitude { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public string? Address { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public PlaceType PlaceType { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public string? PhoneNumber { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

[JsonIgnore]

public IList<IFormFile> Images { get; set; }

[JsonIgnore]

public IList<string>? ImagesBase64 { get; set; }

public string? AdditionalInfoAddress { get; set; }

[Url]

public string? ObjectURL { get; set; }

public string? AdditionalInfoApplication { get; set; }

public string? AdminComment { get; set; }

}

}

**ApplicationObjectViewModelWithoutRequiredImage.cs**

using System.Buffers.Text;

using System.Collections.Generic;

using VladimirTripAdvisor.Logic.Application;

namespace VladimirTripAdvisor.ViewModels

{

public class ApplicationObjectViewModelWithoutRequiredImage

{

public string UserID { get; set; }

public long? ApplicationId { get; set; }

public ApplicationStatus? ApplicationStatus { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

[StringLength(maximumLength: 50)]

public string? Name { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

[StringLength(maximumLength: 1500)]

public string? Description { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

[StringLength(maximumLength: 250)]

public string? ShortDescription { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public string? Latitude { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public string? Longitude { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public string? Address { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public string? PhoneNumber { get; set; }

[Required(ErrorMessage = "Данное поле обязательно")]

public PlaceType PlaceType { get; set; }

[JsonIgnore]

public IList<IFormFile> Images { get; set; }

[JsonIgnore]

public IList<string>? ImagesBase64 { get; set; }

public string? AdditionalInfoAddress { get; set; }

[Url]

public string? ObjectURL { get; set; }

public string? AdditionalInfoApplication { get; set; }

public string? AdminComment { get; set; }

}

}

**EventModelWithPhotoViewModel.cs**

namespace VladimirTripAdvisor.ViewModels

{

public class EventModelWithPhotoViewModel

{

public EventModel? EventModel { get; set; }

public string? ImageMainBase64 { get; set; }

public string? PlaceName { get; set; }

}

}

**EventViewModel.cs**

namespace VladimirTripAdvisor.ViewModels

{

public class EventViewModel

{

public EventModel Event { get; set; }

public string PlaceName { get; set; }

}

}

**ObjectWithPhotoViewModel.cs**

namespace VladimirTripAdvisor.ViewModels

{

public class ObjectWithPhotoViewModel

{

public ObjectWithPhotoViewModel()

{

Place = new ObjectOfVisitModel();

Images = new List<string>();

}

public ObjectOfVisitModel Place{ get; set; }

public string? ImageMainBase64{ get; set; }

public IList<string>? Images { get; set; }

public ReviewModel ReviewModel { get; set; }

public EventModel EventModel { get; set; }

}

}

**ReviewSummary.cs**

namespace VladimirTripAdvisor.ViewModels

{

public class ReviewSummary

{

public ReviewSummary()

{

ExcellentReviews = 0;

GoodReviews = 0;

AverageReviews = 0;

PoorReviews = 0;

TerribleReviews = 0;

ReviewsCount= 0;

AverageRating = 0f;

}

public float AverageRating { get; set; }

public int ReviewsCount { get; set; }

public int ExcellentReviews{ get; set; }

public int GoodReviews { get; set; }

public int AverageReviews { get; set; }

public int PoorReviews { get; set; }

public int TerribleReviews { get; set; }

}

}

**ReviewViewModel.cs**

namespace VladimirTripAdvisor.ViewModels

{

public class ReviewViewModel

{

public ReviewModel Review{ get; set; }

public string Name { get; set; }

public string Surname { get; set; }

}

}

**UserViewModel.cs**

namespace VladimirTripAdvisor.ViewModels

{

public class UserViewModel

{

public UserModel? User{ get; set; }

public string? AccountType { get; set; }

}

}

**AdminController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using System.Data;

using VladimirTripAdvisor.ViewModels;

namespace VladimirTripAdvisor.Controllers

{

public class AdminController : Controller

{

private readonly ApplicationDbContext \_db;

private readonly UserManager<IdentityUser> \_userManager;

public AdminController(ApplicationDbContext db, UserManager<IdentityUser> userManager)

{

\_db = db;

\_userManager = userManager;

}

public IActionResult AllUsers()

{

var users = \_db.Users.ToList();

IList<UserViewModel> userList = new List<UserViewModel>();

foreach (var user in users)

{

var userRoles = \_userManager.GetRolesAsync(user).Result;

string mainRole;

if (userRoles.Contains(WC.OwnerRole))

{

mainRole = WC.OwnerRole;

}

mainRole = userRoles.FirstOrDefault();

UserViewModel userViewModel = new UserViewModel

{

User = (UserModel)user,

AccountType = mainRole

};

userList.Add(userViewModel);

}

return View(userList);

}

public IActionResult DeleteUser(string userId)

{

if (userId == null)

{

return NotFound();

}

var user = \_db.User.Find(userId);

if (user == null)

{

return NotFound();

}

\_db.User.Remove(user);

\_db.SaveChanges();

return View("AllUsers");

}

[Authorize(Roles = WC.AdminRole)]

public IActionResult CheckApplication()

{

IEnumerable<ApplicationModelDB> apps = \_db.Application.Include(x => x.User);

return View(apps);

}

}

}

**ApplicationController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using NuGet.Versioning;

using System;

using System.Collections;

using System.Data;

using System.Net.Mime;

using System.Security.Claims;

using System.Xml.Linq;

using VladimirTripAdvisor.Logic.Application;

using VladimirTripAdvisor.Logic.ImageHandler;

using VladimirTripAdvisor.Logic.YandexMap;

using VladimirTripAdvisor.Models;

using VladimirTripAdvisor.ViewModels;

namespace VladimirTripAdvisor.Controllers

{

[Authorize]

public class ApplicationController : Controller

{

private readonly ApplicationDbContext \_db;

private readonly UserManager<IdentityUser> \_userManager;

private readonly IWebHostEnvironment \_hostEnvironment;

public ApplicationController(ApplicationDbContext db, RoleManager<IdentityRole> roleManager, UserManager<IdentityUser> userManager, IWebHostEnvironment hostEnvironment)

{

\_db = db;

\_userManager = userManager;

\_hostEnvironment = hostEnvironment;

}

public IActionResult Index()

{

return View();

}

public IActionResult CreateApplicationObject()

{

if (!User.IsInRole(WC.OwnerRole))

{

return Forbid();

}

ApplicationObjectViewModel application = new ApplicationObjectViewModel();

application.UserID = \_db.User.Where(x => x.Id == User.FindFirstValue(ClaimTypes.NameIdentifier)).FirstOrDefault().Id;

return View(application);

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult CreateApplicationObject(ApplicationObjectViewModel application)

{

if (!User.IsInRole(WC.OwnerRole))

{

return Forbid();

}

if (!ModelState.IsValid)

{

return View();

}

UserModel? user = \_db.User.Where(x => x.Id == User.FindFirstValue(ClaimTypes.NameIdentifier)).FirstOrDefault();

if (user == null)

{

return NotFound();

}

ApplicationJSONSerialization jSONSerialization = new ApplicationJSONSerialization();

string applicationDataJSON = jSONSerialization.SerializeApplication<ApplicationObjectViewModel>(application);

if (application.ApplicationStatus == 0 || application.ApplicationStatus == null)

{

ApplicationModelDB applicationModelDB = new ApplicationModelDB();

applicationModelDB.ApplicationData = applicationDataJSON;

applicationModelDB.ApplicationType = ApplicationType.AddObject;

applicationModelDB.ApplicationStatus = ApplicationStatus.Created;

applicationModelDB.UserId = user.Id;

\_db.Application.Add(applicationModelDB);

\_db.SaveChanges();

ImageHandler imageHandler = new ImageHandler();

if (application.Images == null)

{

return BadRequest();

}

foreach (var file in application.Images)

{

var bytes = imageHandler.ConvertFileToByteArray(file);

ImageModel imageModel = new ImageModel()

{

ImageByte = bytes.Result,

ApplicationId = applicationModelDB.Id

};

\_db.Image.Add(imageModel);

}

\_db.SaveChanges();

}

else if (application.ApplicationStatus == ApplicationStatus.Edit)

{

ApplicationModelDB? applicationModelDB = \_db.Application.Find(application.ApplicationId);

if (applicationModelDB == null)

{

return NotFound();

}

applicationModelDB.ApplicationData = applicationDataJSON;

applicationModelDB.ApplicationStatus = ApplicationStatus.Processing;

\_db.Application.Update(applicationModelDB);

\_db.SaveChanges();

}

return RedirectToAction("Index", "Home");

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult SubmitApplicationObjectEdit(ApplicationObjectViewModelWithoutRequiredImage application)

{

if (!User.IsInRole(WC.OwnerRole))

{

return Forbid();

}

if (!ModelState.IsValid)

{

return View("ViewApplicationObject");

}

UserModel? user = \_db.User.Where(x => x.Id == User.FindFirstValue(ClaimTypes.NameIdentifier)).FirstOrDefault();

if (user == null)

{

return NotFound();

}

ApplicationJSONSerialization jSONSerialization = new ApplicationJSONSerialization();

string applicationDataJSON = jSONSerialization.SerializeApplication(application);

ApplicationModelDB? applicationModelDB = \_db.Application.Find(application.ApplicationId);

if (applicationModelDB == null)

{

return NotFound();

}

applicationModelDB.ApplicationData = applicationDataJSON;

applicationModelDB.ApplicationStatus = ApplicationStatus.Processing;

\_db.Application.Update(applicationModelDB);

\_db.SaveChanges();

return RedirectToAction("Index", "Home");

}

[Authorize(Roles = WC.UserRole)]

public IActionResult CreateApplicationAccount()

{

if (User.IsInRole(WC.OwnerRole))

{

return Forbid();

}

UserModel? user = \_db.User.Where(x => x.Id == User.FindFirstValue(ClaimTypes.NameIdentifier)).FirstOrDefault();

if (user == null)

{

return NotFound(user);

}

ApplicationAccountViewModel applicationData = new ApplicationAccountViewModel()

{

UserID = user.Id,

Name = user.Name,

Surname = user.Surname,

Midname = user.Midname,

PhoneNumber = user.PhoneNumber,

Email = user.Email

};

return View("UpgradeAccountApplication", applicationData);

}

[Authorize(Roles = WC.UserRole)]

public IActionResult UpgradeAccountApplication(long? id)

{

if (User.IsInRole(WC.OwnerRole))

{

return Forbid();

}

ApplicationModelDB? application = \_db.Application.Find(id);

if (application == null)

{

return NotFound();

}

ApplicationJSONSerialization jSONSerialization = new ApplicationJSONSerialization();

ApplicationAccountViewModel applicationData = jSONSerialization.DeserializeApplication<ApplicationAccountViewModel>

(application.ApplicationData, ApplicationType.UpgradeAccount);

return View(applicationData);

}

[HttpPost]

[ValidateAntiForgeryToken]

[Authorize(Roles = WC.UserRole)]

public IActionResult UpgradeAccountApplication(ApplicationAccountViewModel application)

{

if (User.IsInRole(WC.OwnerRole))

{

return Forbid();

}

if (!ModelState.IsValid)

{

return View(application);

}

ApplicationJSONSerialization jsonSerialization = new ApplicationJSONSerialization();

string jsonApplicationData = jsonSerialization.SerializeApplication(application);

if (application.ApplicationStatus == 0 || application.ApplicationStatus == null)

{

ApplicationModelDB applicationModelDB = new ApplicationModelDB();

applicationModelDB.ApplicationData = jsonApplicationData;

applicationModelDB.ApplicationType = ApplicationType.UpgradeAccount;

applicationModelDB.ApplicationStatus = ApplicationStatus.Created;

applicationModelDB.UserId = application.UserID;

\_db.Application.Add(applicationModelDB);

\_db.SaveChanges();

}

if (application.ApplicationStatus == ApplicationStatus.Edit)

{

ApplicationModelDB? applicationModelDB = \_db.Application.Find(application.ApplicationId);

if (applicationModelDB == null)

{

return NotFound();

}

applicationModelDB.ApplicationData = jsonApplicationData;

applicationModelDB.ApplicationStatus = ApplicationStatus.Processing;

\_db.Application.Update(applicationModelDB);

\_db.SaveChanges();

}

return RedirectToAction("Index", "Home");

}

public IActionResult ViewApplication(long? id)

{

if (id == null || id == 0)

{

return NotFound();

}

ApplicationModelDB? application = \_db.Application.Find(id);

if (application == null)

{

return NotFound();

}

ApplicationJSONSerialization jSONSerialization = new ApplicationJSONSerialization();

if (application.ApplicationType == ApplicationType.UpgradeAccount)

{

var applicationData = jSONSerialization

.DeserializeApplication<ApplicationAccountViewModel>(application.ApplicationData, ApplicationType.UpgradeAccount);

applicationData.ApplicationId = id;

applicationData.ApplicationStatus = application.ApplicationStatus;

if (application.ApplicationStatus == ApplicationStatus.Created)

{

application.ApplicationStatus = ApplicationStatus.Taked;

\_db.Application.Update(application);

\_db.SaveChanges();

}

return View("ViewApplicationAccount", applicationData);

}

else

{

var applicationData = jSONSerialization

.DeserializeApplication<ApplicationObjectViewModelWithoutRequiredImage>(application.ApplicationData, ApplicationType.AddObject);

applicationData.ApplicationId = id;

applicationData.ApplicationStatus = application.ApplicationStatus;

var images = \_db.Image.Where(x => x.ApplicationId == id);

if (images == null)

{

return NotFound();

}

applicationData.ImagesBase64 = new List<string>();

foreach (var image in images)

{

applicationData.ImagesBase64.Add(Convert.ToBase64String(image.ImageByte, 0, image.ImageByte.Length));

}

if (application.ApplicationStatus == ApplicationStatus.Created)

{

application.ApplicationStatus = ApplicationStatus.Taked;

\_db.Application.Update(application);

\_db.SaveChanges();

}

return View("ViewApplicationObject", applicationData);

}

}

[HttpPost]

[ValidateAntiForgeryToken]

[Authorize(Roles = WC.AdminRole)]

public IActionResult DenyApp(long? id)

{

if (id == null || id == 0)

{

return NotFound();

}

ApplicationModelDB? application = \_db.Application.Find(id);

if (application == null)

{

return NotFound();

}

application.ApplicationStatus = ApplicationStatus.Canceled;

\_db.Application.Update(application);

\_db.SaveChanges();

return RedirectToAction("CheckApplication", "Admin");

}

[HttpPost]

[ValidateAntiForgeryToken]

[Authorize(Roles = WC.AdminRole)]

public async Task<IActionResult> SubmitApplicationAccount(long? id)

{

if (id == null || id == 0)

{

return NotFound();

}

ApplicationModelDB? application = \_db.Application.Find(id);

if (application == null)

{

return NotFound();

}

IdentityUser user = await \_userManager.FindByIdAsync(application.UserId);

if (user == null)

{

return NotFound();

}

await \_userManager.AddToRoleAsync(user, WC.OwnerRole);

application.ApplicationStatus = ApplicationStatus.Done;

\_db.Application.Update(application);

await \_db.SaveChangesAsync();

return RedirectToAction("CheckApplication", "Admin");

}

[HttpPost]

[ValidateAntiForgeryToken]

[Authorize(Roles = WC.AdminRole)]

public IActionResult SubmitApplicationObject(long? id)

{

if (!ModelState.IsValid)

{

return View("ViewApplicationObject");

}

if (id == null || id == 0)

{

return NotFound();

}

ApplicationModelDB? application = \_db.Application.Find(id);

if (application == null)

{

return NotFound();

}

var jSONSerialization = new ApplicationJSONSerialization();

var applicationJsonData = jSONSerialization.DeserializeApplication<ApplicationObjectViewModel>

(application.ApplicationData, ApplicationType.AddObject);

application.ApplicationStatus = ApplicationStatus.Done;

if (applicationJsonData == null)

{

return BadRequest();

}

ObjectOfVisitModel place = new ObjectOfVisitModel()

{

IdOwner = application.UserId,

Name = applicationJsonData.Name,

Description = applicationJsonData.Description,

StreetAddress = applicationJsonData.Address,

Latitude = applicationJsonData.Latitude,

Longitude = applicationJsonData.Longitude,

AdditionalAddressInfo = applicationJsonData.AdditionalInfoAddress,

PlaceURL = applicationJsonData.ObjectURL,

PlaceType = applicationJsonData.PlaceType,

ShortDescription = applicationJsonData.ShortDescription,

PhoneNumber = applicationJsonData.PhoneNumber,

AverageRating = 0

};

\_db.Application.Update(application);

\_db.ObjectOfVisit.Add(place);

\_db.SaveChanges();

var objectImages = \_db.Image.Where(x => x.ApplicationId == application.Id);

foreach (var item in objectImages)

{

item.ObjectId = place.Id;

}

\_db.UpdateRange(objectImages);

\_db.SaveChanges();

BaloonJson baloonJson = new BaloonJson(\_hostEnvironment);

baloonJson.AddObject(place, Convert.ToBase64String(objectImages.FirstOrDefault().ImageByte));

return RedirectToAction("CheckApplication", "Admin");

}

[Authorize(Roles = WC.UserRole)]

public IActionResult MyApplicationAccount()

{

IEnumerable<ApplicationModelDB> apps = \_db.Application.Where(y => y.UserId == User.FindFirstValue(ClaimTypes.NameIdentifier));

return View(apps);

}

[Authorize(Roles = WC.AdminRole)]

public IActionResult SendOnEditApplicationAccount(ApplicationAccountViewModel application)

{

if (application == null)

{

return NotFound();

}

ApplicationJSONSerialization jsonSerialization = new ApplicationJSONSerialization();

string jsonApplicationData = jsonSerialization.SerializeApplication(application);

ApplicationModelDB? applicationModelDB = \_db.Application.Find(application.ApplicationId);

if (applicationModelDB == null)

{

return NotFound();

}

applicationModelDB.ApplicationData = jsonApplicationData;

applicationModelDB.ApplicationStatus = ApplicationStatus.Edit;

\_db.Application.Update(applicationModelDB);

\_db.SaveChanges();

return RedirectToAction("CheckApplication", "Admin");

}

public IActionResult SendOnEditApplicationObject(ApplicationObjectViewModel application)

{

if (application == null)

{

return NotFound();

}

ApplicationJSONSerialization jsonSerialization = new ApplicationJSONSerialization();

string jsonApplicationData = jsonSerialization.SerializeApplication<ApplicationObjectViewModel>(application);

ApplicationModelDB? applicationModelDB = \_db.Application.Find(application.ApplicationId);

if (applicationModelDB == null)

{

return NotFound();

}

applicationModelDB.ApplicationData = jsonApplicationData;

applicationModelDB.ApplicationStatus = ApplicationStatus.Edit;

\_db.Application.Update(applicationModelDB);

\_db.SaveChanges();

return RedirectToAction("CheckApplication", "Admin");

}

}

}

**EventController.cs**

using Microsoft.AspNetCore.Mvc;

using System.Security.Claims;

using VladimirTripAdvisor.Logic.Telegram;

using VladimirTripAdvisor.Models;

using VladimirTripAdvisor.ViewModels;

namespace VladimirTripAdvisor.Controllers

{

public class EventController : Controller

{

private readonly ApplicationDbContext \_db;

public EventController(ApplicationDbContext db)

{

\_db = db;

}

public IActionResult CreateEvent(EventModel eventModel)

{

Telegram telegram = new Telegram();

ObjectOfVisitModel? place = \_db.ObjectOfVisit.Find(eventModel.PlaceId);

if (place == null)

{

return NotFound();

}

string telegramLink = telegram.CreateTelegramChannel(eventModel.Name, eventModel.StartDate, place.Name, place.Id).Result;

eventModel.TelegramLink = telegramLink;

\_db.Event.Add(eventModel);

\_db.SaveChanges();

return View("MyEvents");

}

public IActionResult AllEvents()

{

IList<EventModelWithPhotoViewModel> eventList = new List<EventModelWithPhotoViewModel>();

var events = \_db.Event.Where(x => x.EventType == EventType.Public)

.Where(x => x.StartDate.CompareTo(DateTime.Now.AddHours(-1)) > 0).ToList();

foreach (var item in events)

{

var eventViewModel = new EventModelWithPhotoViewModel();

eventViewModel.EventModel = item;

ImageModel? ImageBase64 = \_db.Image.Where(x => x.ObjectId == item.PlaceId).FirstOrDefault();

if(ImageBase64 != null)

{

eventViewModel.ImageMainBase64 = Convert.ToBase64String(ImageBase64.ImageByte);

}

ObjectOfVisitModel? place = \_db.ObjectOfVisit.Find(item.PlaceId);

if(place != null)

{

eventViewModel.PlaceName = place.Name;

}

eventList.Add(eventViewModel);

}

return View(eventList);

}

public IActionResult MyEvents()

{

IList<EventViewModel> eventList = new List<EventViewModel>();

var events = \_db.Event.Where(x => x.CreatorId == User.FindFirstValue(ClaimTypes.NameIdentifier)).ToList();

foreach (var item in events)

{

var eventViewModel = new EventViewModel();

eventViewModel.Event = item;

var place = \_db.ObjectOfVisit.Find(item.PlaceId);

if (place != null)

{

eventViewModel.PlaceName = place.Name;

}

eventList.Add(eventViewModel);

}

return View(eventList);

}

}

}

**MapController.cs**

using Microsoft.AspNetCore.Mvc;

using System.Diagnostics;

using VladimirTripAdvisor.Models;

namespace WebApplication1.Controllers

{

public class MapController : Controller

{

private readonly ILogger<MapController> \_logger;

public MapController(ILogger<MapController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

}

}

**ObjectController.cs**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System.Collections;

using System.Diagnostics;

using System.Security.Claims;

using VladimirTripAdvisor.Logic.YandexMap;

using VladimirTripAdvisor.Models;

using VladimirTripAdvisor.ViewModels;

namespace VladimirTripAdvisor.Controllers

{

public class ObjectController : Controller

{

private readonly ApplicationDbContext \_db;

private readonly IWebHostEnvironment \_hostEnvironment;

public ObjectController(ApplicationDbContext db, IWebHostEnvironment hostEnvironment)

{

\_db = db;

\_hostEnvironment = hostEnvironment;

}

// POST: ObjectController/Delete/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Delete(int id, IFormCollection collection)

{

try

{

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

public async Task<IActionResult> AllObjects()

{

IList<ObjectOfVisitModel> objectList = \_db.ObjectOfVisit.ToList();

IList<ObjectWithPhotoViewModel> objectVM = new List<ObjectWithPhotoViewModel>();

foreach (var item in objectList)

{

ImageModel imgByte = await \_db.Image.FirstOrDefaultAsync(x => x.ObjectId == item.Id);

var obj = new ObjectWithPhotoViewModel()

{

Place = item,

ImageMainBase64 = Convert.ToBase64String(imgByte.ImageByte)

};

objectVM.Add(obj);

}

return View(objectVM);

}

public async Task<IActionResult> ViewMyObjects()

{

IList<ObjectOfVisitModel> objectList = \_db.ObjectOfVisit.Where(x=>x.IdOwner == User.FindFirstValue(ClaimTypes.NameIdentifier)).ToList();

IList<ObjectWithPhotoViewModel> objectVM = new List<ObjectWithPhotoViewModel>();

foreach (var item in objectList)

{

ImageModel imgByte = await \_db.Image.FirstOrDefaultAsync(x => x.ObjectId == item.Id);

var obj = new ObjectWithPhotoViewModel()

{

Place = item,

ImageMainBase64 = Convert.ToBase64String(imgByte.ImageByte)

};

objectVM.Add(obj);

}

return View(objectVM);

}

public IActionResult ViewObject(long? id)

{

if (id == null)

{

return NotFound();

}

var place = \_db.ObjectOfVisit.Find(id);

if (place == null)

{

return NotFound();

}

var objectVM = new ObjectWithPhotoViewModel()

{

Place = place

};

var objectImages = \_db.Image.Where(x => x.ObjectId == id);

foreach (var item in objectImages)

{

objectVM.Images.Add(Convert.ToBase64String(item.ImageByte));

}

return View(objectVM);

}

public IActionResult RemoveObject(long? id)

{

if (id == null)

{

return NotFound();

}

ObjectOfVisitModel? place = \_db.ObjectOfVisit.Find(id);

if(place == null)

{

return BadRequest();

}

\_db.ObjectOfVisit.Remove(place);

\_db.SaveChanges();

return RedirectToAction("AllObjects");

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

public IActionResult Map()

{

return View();

}

}

}

**ReviewController.cs**

using Microsoft.AspNetCore.Mvc;

using NuGet.Protocol.Plugins;

using System.Collections;

namespace VladimirTripAdvisor.Controllers

{

public class ReviewController : Controller

{

private readonly ApplicationDbContext \_db;

public ReviewController(ApplicationDbContext db)

{

\_db = db;

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult AddReview(ReviewModel reviewModel)

{

if (!ModelState.IsValid)

{

return RedirectToAction("Index", "Home");

}

\_db.Add(reviewModel);

\_db.SaveChanges();

UpdateAverageObjectRating(reviewModel.ObjectId);

return RedirectToAction("ViewObject", "Object", new {id = reviewModel.ObjectId});

}

private void UpdateAverageObjectRating(long? placeId)

{

var placeReviews = \_db.Review.Where(x => x.ObjectId == placeId);

float averageRating = 0;

foreach (var review in placeReviews)

{

averageRating += (float) review.Score;

}

averageRating /= placeReviews.Count();

averageRating = MathF.Round(averageRating, 2);

var place = \_db.ObjectOfVisit.Find(placeId);

place.AverageRating = averageRating;

\_db.Update(place);

\_db.SaveChanges();

}

[HttpGet]

public IActionResult RemoveReview(long? reviewId, long? objectId)

{

if (reviewId == null || objectId == null)

{

return NotFound();

}

ReviewModel? review = \_db.Review.Find(reviewId);

if (review == null)

{

return NotFound();

}

\_db.Remove(review);

\_db.SaveChanges();

UpdateAverageObjectRating(objectId);

return RedirectToAction("ViewObject", "Object", new { id = objectId });

}

}

}

**ApplicationDbContext.cs**

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using System.Reflection.Metadata;

using VladimirTripAdvisor.Models;

namespace VladimirTripAdvisor.Data

{

public class ApplicationDbContext : IdentityDbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)

{

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<ImageModel>(entity =>

{

entity.Property(x => x.ImageByte).HasColumnType("longblob");

});

modelBuilder.Entity<ApplicationModelDB>(entity =>

{

entity.Property(x => x.ApplicationData).HasColumnType("JSON");

});

}

public DbSet<UserModel> User { get; set; }

public DbSet<ObjectOfVisitModel> ObjectOfVisit { get; set; }

public DbSet<ImageModel> Image{ get; set; }

public DbSet<ReviewModel> Review { get; set; }

public DbSet<ApplicationModelDB> Application { get; set; }

public DbSet<EventModel> Event { get; set; }

}

}

**Commentary.cs**

using Microsoft.AspNetCore.Mvc;

using VladimirTripAdvisor.Data;

using VladimirTripAdvisor.ViewModels;

namespace VladimirTripAdvisor.Components

{

public class Commentary : ViewComponent

{

private readonly ApplicationDbContext \_db;

public Commentary(ApplicationDbContext db)

{

\_db = db;

}

public IViewComponentResult Invoke(long? id)

{

var reviews = \_db.Review.Where(x => x.ObjectId == id).OrderByDescending(x => x.ReviewDate);

IList<ReviewViewModel> reviewViewModels = new List<ReviewViewModel>();

foreach (var review in reviews.ToList())

{

var userReview = \_db.User.Find(review.UserId);

review.ReviewDate = review.ReviewDate.Date;

ReviewViewModel reviewView = new ReviewViewModel()

{

Review = review,

Name = userReview.Name,

Surname = userReview.Surname,

};

reviewViewModels.Add(reviewView);

}

return View("Commentary", reviewViewModels);

}

}

}

**CommentarySummary.cs**

using Microsoft.AspNetCore.Mvc;

using VladimirTripAdvisor.Data;

using VladimirTripAdvisor.ViewModels;

namespace VladimirTripAdvisor.Components

{

public class CommentarySummary : ViewComponent

{

private readonly ApplicationDbContext \_db;

public CommentarySummary(ApplicationDbContext db)

{

\_db = db;

}

public async Task<IViewComponentResult> InvokeAsync(long? id)

{

var reviews = \_db.Review.Where(x => x.ObjectId == id);

ReviewSummary reviewSummary = new ReviewSummary();

reviewSummary.ReviewsCount = reviews.Count();

var place = await \_db.ObjectOfVisit.FindAsync(id);

reviewSummary.AverageRating = (float)place.AverageRating;

foreach (var review in reviews)

{

switch (review.Score)

{

case ReviewScore.Excellent:

reviewSummary.ExcellentReviews++;

break;

case ReviewScore.Good:

reviewSummary.GoodReviews++;

break;

case ReviewScore.Average:

reviewSummary.AverageReviews++;

break;

case ReviewScore.Poor:

reviewSummary.PoorReviews++;

break;

case ReviewScore.Terrible:

reviewSummary.TerribleReviews++;

break;

default:

break;

}

}

return View("CommentarySummary", reviewSummary);

}

}

}

**GetUserId.cs**

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

namespace VladimirTripAdvisor.Components

{

public class GetUserId : ViewComponent

{

private readonly UserManager<IdentityUser> \_userManager;

public GetUserId(UserManager<IdentityUser> userManager)

{

\_userManager = userManager;

}

public async Task<string> InvokeAsync()

{

var appUser = await \_userManager.FindByNameAsync(HttpContext.User.Identity.Name);

return appUser.Id;

}

}

}

**ApplicationJSONSerialization.cs**

using Newtonsoft.Json.Linq;

using VladimirTripAdvisor.ViewModels;

namespace VladimirTripAdvisor.Logic.Application

{

public class ApplicationJSONSerialization

{

public string SerializeApplication<T>(T application)

{

var options = new JsonSerializerOptions { Encoder = System.Text.Encodings.Web.JavaScriptEncoder.UnsafeRelaxedJsonEscaping, DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull };

string result = JsonSerializer.Serialize(application, options);

return result;

}

public T DeserializeApplication<T>(string applicationData, ApplicationType type)

{

var options = new JsonSerializerOptions { Encoder = System.Text.Encodings.Web.JavaScriptEncoder.UnsafeRelaxedJsonEscaping, DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull };

if(type == ApplicationType.UpgradeAccount)

{

T application = JsonSerializer.Deserialize<T>(applicationData);

return (T)Convert.ChangeType(application, typeof(T));

}

else

{

T application = JsonSerializer.Deserialize<T>(applicationData);

return (T)Convert.ChangeType(application, typeof(T));

}

}

}

}

**ApplicationStatus.cs**

using System.ComponentModel;

namespace VladimirTripAdvisor.Logic.Application

{

public enum ApplicationStatus

{

[Description("Заявление создано")]

Created = 1,

[Description("Принята администратором создано")]

Taked = 2,

[Description("В обработке")]

Processing = 3,

[Description("На редактировании")]

Edit = 4,

[Description("Обработана успешно")]

Done = 5,

[Description("Отклонена")]

Canceled = 6

}

}

**ApplicationType.cs**

namespace VladimirTripAdvisor.Logic.Application

{

public enum ApplicationType

{

UpgradeAccount = 1,

AddObject = 2

}

}

**ImageHandler.cs**

namespace VladimirTripAdvisor.Logic.ImageHandler

{

public class ImageHandler

{

public async Task<byte[]> ConvertFileToByteArray(IFormFile file)

{

long length = file.Length;

if (length <= 0)

{

return null;

}

using var fileStream = file.OpenReadStream();

byte[] bytes = new byte[length];

await fileStream.ReadAsync(bytes, 0, (int)file.Length);

return bytes;

}

}

}

**Telegram.cs**

using Microsoft.Build.Execution;

using System.Threading.Channels;

using TL;

using WTelegram;

namespace VladimirTripAdvisor.Logic.Telegram

{

public class Telegram

{

private static string Config(string what)

{

switch (what)

{

case "api\_id": return "28837445";

case "api\_hash": return "f75175336330b8fb0deb87b39c5969bf";

case "phone\_number": return "+79106648980";

case "verification\_code": Console.Write("Code: "); return Console.ReadLine();

case "first\_name": return "John"; // if sign-up is required

case "last\_name": return "Doe"; // if sign-up is required

case "password": return "450912"; // if user has enabled 2FA

default: return null; // let WTelegramClient decide the default config

}

}

public void InitialiseTelegramClient()

{

using var client = new WTelegram.Client(Config);

}

public async Task<string> CreateTelegramChannel(string channelName, DateTime startDate, string placeName, long placeId)

{

using (var client = new WTelegram.Client(Config))

{

await client.ConnectAsync();

var users = new InputUser[1];

for (int i = 0; i < users.Length; i++)

{

users[i] = new InputUser(5802032660, -1117118480243990476);

}

var chat = await client.Messages\_CreateChat(users, channelName);

long chatId = chat.Chats.First().Value.ID;

var telegraMessage = await client.Messages\_SendMessage(new InputPeerChat(chatId), $"Приветствую. Спасибо что присоединилсь к мероприятию." +

$"\nЦель посещения: {placeName}\nДата посещения: {startDate}\nВпрочем, время и дату посещения вы всегда можете сами обсудить" +

$"\nПосле посещения места, пожалуйста, оставьте отзыв по ссылке сообщением ниже:", new Random().NextInt64());

await client.Messages\_SendMessage(new InputPeerChat(chatId), $"https://localhost:7048/Object/ViewObject/{placeId}", new Random().NextInt64());

await client.Messages\_UpdatePinnedMessage(new InputPeerChat(chatId), (telegraMessage.UpdateList.First() as TL.UpdateMessageID).id);

ChatInviteExported exportedChatInvite = (ChatInviteExported)await client.Messages\_ExportChatInvite(new InputPeerChat(chatId));

return exportedChatInvite.link;

}

}

}

}

**BaloonModel.cs**

public class BaloonModel

{

public BaloonModel()

{

}

public BaloonModel(Feature[] ballonCollections)

{

type = "FeatureCollection";

features = ballonCollections;

}

public string type { get; set; }

public IList<Feature> features { get; set; }

}

**Feature.cs**

public class Feature

{

public Feature()

{

}

public Feature(long objectId, string latitude, string longitude,

string contentHeader, string contentBody, string contentFooter = null)

{

id = objectId;

type = "Feature";

geometry = new Geometry(latitude, longitude);

properties = new Properties(contentHeader, contentBody, contentFooter);

}

public string type { get; set; }

public long id { get; set; }

public Geometry geometry { get; set; }

public Properties properties { get; set; }

}

**Geometry.cs**

public class Geometry

{

public Geometry()

{

}

public Geometry(string latitude, string longitude)

{

type = "Point";

coordinates = new[] { double.Parse(latitude.Replace('.',',')),

double.Parse(longitude.Replace('.',',')) };

}

public string type { get; set; }

public double[] coordinates { get; set; }

}

**Properties.cs**

public class Properties

{

public Properties()

{

}

public Properties(string contentHeader, string contentBody, string contentFooter)

{

balloonContentHeader = contentHeader;

balloonContentBody = contentBody;

balloonContentFooter = contentFooter;

}

public string balloonContentHeader { get; set; }

public string balloonContentBody { get; set; }

public string balloonContentFooter { get; set; }

public string? clustercaption { get; set; }

public string? hintcontent { get; set; }

}

**BaloonJson.cs**

using System.Text;

using VladimirTripAdvisor.Logic.YandexMap.JsonModel;

namespace VladimirTripAdvisor.Logic.YandexMap

{

public class BaloonJson

{

private readonly IWebHostEnvironment \_appEnvironment;

public BaloonJson(IWebHostEnvironment webHostEnvironment)

{

\_appEnvironment = webHostEnvironment;

}

public void AddObject(ObjectOfVisitModel place, string imageBase64)

{

var objectBaloons = LoadJson();

if (objectBaloons != null)

{

Feature baloon = new Feature(place.Id, place.Latitude, place.Longitude,

GenerateBaloonHeaderContent(place.Name), GenerateBaloonBodyContent(imageBase64, place.PlaceURL, place.PhoneNumber, place.Id));

objectBaloons.features.Add(baloon);

SaveJson(objectBaloons);

}

}

private string GenerateBaloonHeaderContent(string placeName)

{

string content = $"<div class='d-flex justify-content-center'><p class='fw-bold text-primary' style='font-size: 26px;margin-bottom: 5px;'>{placeName}</p> </div>";

return content;

}

private string GenerateBaloonBodyContent(string imageBase64, string placeURL, string phoneNumber, long placeId)

{

string content = $"<div class='d-flex justify-content-center'><img src='data:image/jpg;base64, {imageBase64}' style='object-fit: contain;height: 250px;' />" +

$"</div><div class='d-flex justify-content-around' style='margin-top: 20px;'>" +

$"<p class='fw-bold text-center d-flex flex-column justify-content-center' " +

$"style='width: max-content;margin-left: 25px;margin-bottom: 0px;'>Веб-сайт:<a href='{placeURL}' " +

$"style='padding-left: 0px;margin-left: 0px;'>{placeURL}</a></p><p class='fw-bold text-center' " +

$"style='width: max-content;margin-left: 25px;margin-bottom: 15px;'>Номер телефона: <br />{phoneNumber}</p></div>" +

$"<div class='d-flex justify-content-center'><a href='/Object/ViewObject/{placeId}' class='btn btn-primary'> Перейти </a></div>";

return content;

}

private BaloonModel LoadJson()

{

string baloons;

string pathJson = \_appEnvironment.WebRootPath + "\\ymaps\\objects.json";

var options = new JsonSerializerOptions { Encoder = System.Text.Encodings.Web.JavaScriptEncoder.UnsafeRelaxedJsonEscaping, DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull };

using (StreamReader streamReader = new StreamReader(pathJson, Encoding.UTF8))

{

baloons = streamReader.ReadToEnd();

}

BaloonModel baloonJson = JsonSerializer.Deserialize<BaloonModel>(baloons, options);

return baloonJson;

}

private async Task SaveJson(BaloonModel baloons)

{

string pathJson = \_appEnvironment.WebRootPath + "\\ymaps\\objects.json";

var options = new JsonSerializerOptions { Encoder = System.Text.Encodings.Web.JavaScriptEncoder.UnsafeRelaxedJsonEscaping, DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull, WriteIndented = true};

string baloonJsonString = JsonSerializer.Serialize(baloons,options);

await File.WriteAllTextAsync(pathJson, baloonJsonString);

}

}

}

**EventType.cs**

namespace VladimirTripAdvisor.Logic

{

public enum EventType

{

[Display(Name = "Частный")]

Private = 1,

[Display(Name = "Общий")]

Public = 2

}

}

**PlaceType.cs**

namespace VladimirTripAdvisor.Logic

{

public enum PlaceType

{

[Display(Name = "Заведение")]

Attraction = 1,

[Display(Name = "Культурный объект")]

Cultural = 2,

[Display(Name = "Ресторан")]

Restaurant = 3,

[Display(Name = "Жилье")]

Accommodation =4

}

}

**ReviewScore.cs**

namespace VladimirTripAdvisor.Logic

{

public enum ReviewScore

{

[Display(Name = "Отлично")]

Excellent = 5,

[Display(Name = "Хорошо")]

Good = 4,

[Display(Name = "Средне")]

Average = 3,

[Display(Name = "Плохо")]

Poor = 2,

[Display(Name = "Ужасно")]

Terrible =1

}

}

**btn\_readmore.js**

$(".toggle").click(function () {

$(this).removeClass("d-inline-flex").addClass("d-none");

$(this).siblings("button").removeClass("d-none").addClass("d-inline-flex");

$(this).parent("").children("p").toggleClass("truncate");

});

**ymaps\_create\_application.js**

jQuery(function () {

ymaps.ready(init);

});

function init() {

var map = new ymaps.Map('map', {

center: [56.13, 40.39],

zoom: 12

}, {

searchControlProvider: 'yandex#search'

}),

markerElement = jQuery('#marker'),

dragger = new ymaps.util.Dragger({

// Драггер будет автоматически запускаться при нажатии на элемент 'marker'.

autoStartElement: markerElement[0]

}),

// Смещение маркера относительно курсора.

markerOffset,

markerPosition;

map.controls.remove('searchControl');

map.controls.remove('geolocationControl');

map.controls.remove('trafficControl');

map.controls.remove('fullscreenControl');

map.controls.remove('rulerControl');

map.controls.remove('typeSelector');

dragger.events

.add('start', onDraggerStart)

.add('move', onDraggerMove)

.add('stop', onDraggerEnd);

function onDraggerStart(event) {

var offset = markerElement.offset(),

position = event.get('position');

// Сохраняем смещение маркера относительно точки начала драга.

markerOffset = [

position[0] - offset.left,

position[1] - offset.top

];

markerPosition = [

position[0] - markerOffset[0],

position[1] - markerOffset[1]

];

applyMarkerPosition();

}

function onDraggerMove(event) {

applyDelta(event);

}

function onDraggerEnd(event) {

applyDelta(event);

markerPosition[0] += markerOffset[0];

markerPosition[1] += markerOffset[1];

// Переводим координаты страницы в глобальные пиксельные координаты.

var markerGlobalPosition = map.converter.pageToGlobal(markerPosition),

// Получаем центр карты в глобальных пиксельных координатах.

mapGlobalPixelCenter = map.getGlobalPixelCenter(),

// Получением размер контейнера карты на странице.

mapContainerSize = map.container.getSize(),

mapContainerHalfSize = [mapContainerSize[0] / 2, mapContainerSize[1] / 2],

// Вычисляем границы карты в глобальных пиксельных координатах.

mapGlobalPixelBounds = [

[mapGlobalPixelCenter[0] - mapContainerHalfSize[0], mapGlobalPixelCenter[1] - mapContainerHalfSize[1]],

[mapGlobalPixelCenter[0] + mapContainerHalfSize[0], mapGlobalPixelCenter[1] + mapContainerHalfSize[1]]

];

// Проверяем, что завершение работы драггера произошло в видимой области карты.

if (containsPoint(mapGlobalPixelBounds, markerGlobalPosition)) {

// Теперь переводим глобальные пиксельные координаты в геокоординаты с учетом текущего уровня масштабирования карты.

var geoPosition = map.options.get('projection').fromGlobalPixels(markerGlobalPosition, map.getZoom());

document.getElementById("latitude").value = geoPosition[0];

document.getElementById("longitude").value = geoPosition[1];

}

}

function applyDelta(event) {

// Поле 'delta' содержит разницу между положениями текущего и предыдущего события драггера.

var delta = event.get('delta');

markerPosition[0] += delta[0];

markerPosition[1] += delta[1];

applyMarkerPosition();

}

function applyMarkerPosition() {

markerElement.css({

left: markerPosition[0],

top: markerPosition[1]

});

}

function containsPoint(bounds, point) {

return point[0] >= bounds[0][0] && point[0] <= bounds[1][0] &&

point[1] >= bounds[0][1] && point[1] <= bounds[1][1];

}

function getTileCoordinate(coords, zoom, tileSize) {

return [

Math.floor(coords[0] \* zoom / tileSize),

Math.floor(coords[1] \* zoom / tileSize)

];

}

}

**ymaps\_object\_manager.js**

ymaps.ready(init);

function init() {

var myMap = new ymaps.Map('map', {

center: [56.13, 40.39],

zoom: 14

}),

objectManager = new ymaps.ObjectManager({

// Чтобы метки начали кластеризоваться, выставляем опцию.

clusterize: true,

// ObjectManager принимает те же опции, что и кластеризатор.

gridSize: 32,

clusterDisableClickZoom: true

});

// Чтобы задать опции одиночным объектам и кластерам,

// обратимся к дочерним коллекциям ObjectManager.

objectManager.objects.options.set('preset', 'islands#greenDotIcon');

objectManager.clusters.options.set('preset', 'islands#greenClusterIcons');

myMap.geoObjects.add(objectManager)

myMap.controls.remove('searchControl');

myMap.controls.remove('geolocationControl');

myMap.controls.remove('trafficControl');

myMap.controls.remove('fullscreenControl');

myMap.controls.remove('rulerControl');

myMap.controls.remove('typeSelector');

$.ajax({

url: "/ymaps/objects.json"

}).done(function (data) {

objectManager.add(data);

});

}

**WC.cs**

namespace VladimirTripAdvisor

{

public static class WC

{

[Display(Name ="Администратор")]

public const string AdminRole = "Admin";

[Display(Name ="Владелец")]

public const string OwnerRole = "Owner";

[Display(Name = "Простой пользователь")]

public const string UserRole = "User";

}

}

**AllUsers.cshtml**

@model IEnumerable<VladimirTripAdvisor.ViewModels.UserViewModel>

@{

ViewData["Title"] = "AllUsers";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div class="container">

<div class="row pt-4">

<div class="col-6">

<h2 class="text-primary">Список пользователей</h2>

</div>

</div>

<br />

@if (Model.Count() > 0)

{

<table class="table table-bordered table-striped" style="width:100%">

<thead>

<tr>

<th>

Логин

</th>

<th>

Фамилия

</th>

<th>

Имя

</th>

<th>

Тип аккаунта

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var obj in Model)

{

<tr>

<td width="15%">@obj.User.UserName</td>

<td width="15%">@obj.User.Surname</td>

<td width="15%">@obj.User.Name</td>

<td width="15%">@obj.AccountType</td>

<td class="text-center">

<div class="btn-group" role="group">

<button class="btn btn-danger" type="button"

data-bs-toggle="modal" data-bs-target="#deleteUserAccountModal\_@obj.User.Id">Удалить аккаунт</button>

</div>

</td>

</tr>

<div id="deleteUserAccountModal\_@obj.User.Id" class="modal fade" role="dialog" tabindex="-1">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-body d-flex justify-content-center align-items-center align-content-center">

<p class="text-center" style="padding-bottom: 0px;margin-bottom: 0px; font-size:20px;">Удалить аккаунт?</p>

</div>

<div class="modal-footer">

<button class="btn btn-primary" type="button" data-bs-dismiss="modal">Нет</button>

<button class="btn btn-danger">

<**a** **asp-route-userId**="@obj.User.Id" **asp-controller**="Admin" **asp-action**="DeleteUser" style="color: #ffffff; text-decoration:none">

Да

</**a**>

</button>

</div>

</div>

</div>

</div>

}

</tbody>

</table>

}

else

{

<p>Список пустой</p>

}

</div>

**CheckApplication.cshtml**

@model IEnumerable<VladimirTripAdvisor.Models.ApplicationModelDB>

@{

ViewData["Title"] = "CheckApplication";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div class="container">

<div class="row pt-4">

<div class="col-6">

<h2 class="text-primary">Список заявок на улучшение статуса аккаунта</h2>

</div>

</div>

<br />

@if (Model.Count() > 0)

{

<table class="table table-bordered table-striped" style="width:100%">

<thead>

<tr>

<th>

Фамилия

</th>

<th>

Имя

</th>

<th>

Тип заявления

</th>

<th>

Статус заявления

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var obj in Model)

{

if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Created)

{

<tr>

<td width="20%">@obj.User.Surname</td>

<td width="20%">@obj.User.Name</td>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="20%">Улучшение аккаунта</td>

}

else

{

<td width="20%">Добавление объекта</td>

}

<td width="20%">Создана</td>

<td class="text-center">

<div class="w-75 btn-group" role="group">

<**a** **asp-controller**="Application" **asp-route-Id**="@obj.Id" **asp-action**="ViewApplication" class="btn btn-primary">

Принять

</**a**>

</div>

</td>

</tr>

}

else if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Taked)

{

<tr>

<td width="20%">@obj.User.Surname</td>

<td width="20%">@obj.User.Name</td>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="20%">Улучшение аккаунта</td>

}

else

{

<td width="20%">Добавление объекта</td>

}

<td width="20%">Принята администратором</td>

<td class="text-center">

<div class="w-75 btn-group" role="group">

<a asp-controller="Application" asp-route-Id="@obj.Id" asp-action="ViewApplication" class="btn btn-primary">

Открыть

</a>

</div>

</td>

</tr>

}

else if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Processing)

{

<tr>

<td width="20%">@obj.User.Surname</td>

<td width="20%">@obj.User.Name</td>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="20%">Улучшение аккаунта</td>

}

else

{

<td width="20%">Добавление объекта</td>

}

<td width="20%">В обработке</td>

<td class="text-center">

<div class="w-75 btn-group" role="group">

<**a** **asp-controller**="Application" **asp-route-Id**="@obj.Id" **asp-action**="ViewApplication" class="btn btn-primary">

Открыть

</**a**>

</div>

</td>

</tr>

}

else if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Edit)

{

<tr>

<td width="20%">@obj.User.Surname</td>

<td width="20%">@obj.User.Name</td>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="20%">Улучшение аккаунта</td>

}

else

{

<td width="20%">Добавление объекта</td>

}

<td width="20%">На редактировании</td>

<td></td>

</tr>

}

else if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Done)

{

<tr>

<td width="20%">@obj.User.Surname</td>

<td width="20%">@obj.User.Name</td>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="20%">Улучшение аккаунта</td>

}

else

{

<td width="20%">Добавление объекта</td>

}

<td width="20%">Обработана</td>

<td></td>

</tr>

}

else if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Canceled)

{

<tr>

<td width="20%">@obj.User.Surname</td>

<td width="20%">@obj.User.Name</td>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="20%">Улучшение аккаунта</td>

}

else

{

<td width="20%">Добавление объекта</td>

}

<td>Отклонена администратором</td>

<td></td>

</tr>

}

}

</tbody>

</table>

}

else

{

<p>Список пустой</p>

}

</div>

**CreateApplicationObject.cshtml**

@model VladimirTripAdvisor.ViewModels.ApplicationObjectViewModel

@{

ViewData["Title"] = "Создание заявления";

}

<div class="col-12 text-success text-center">

<**form** method="post" enctype="multipart/form-data">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<div class="row form-group">

<h2 class="col-12 text-success text-center py-4">Заявление на добавление новго объекта в систему.</h2>

<div class="col-12">

<div class="bg-white">

<div class="px-lg-5 pt-4 col-6 float-end">

<div class="container justify-content-center">

<div id="map" class="col-6" style="width: 512px; height: 400px">

<div id="marker"></div>

</div>

</div>

<div class="pt-4">

<div class="py-2">

<**input** **asp-for**="Latitude" **type**="text" id="latitude" class="form-control" placeholder="Широта">

<**span** **asp-validation-for**="Latitude" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Longitude" **type**="text" id="longitude" class="form-control" placeholder="Долгота">

<**span** **asp-validation-for**="Longitude" class="text-danger"></**span**>

</div>

</div>

</div>

<div class="card-body px-lg-5 pt-4 col-6">

<**input** **asp-for**="UserID" **type**="hidden" class="form-control">

<**input** **asp-for**="ApplicationId" **type**="hidden" class="form-control">

<div class="py-2">

<**select** **asp-for**="PlaceType" class="form-control" **asp-items**="Html.GetEnumSelectList<PlaceType>()"></**select**>

<**span** **asp-validation-for**="PlaceType" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Name" **type**="text" class="form-control" placeholder="Название места">

<**span** **asp-validation-for**="Name" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="PhoneNumber" **type**="text" class="form-control" placeholder="Номер телефона объекта">

<**span** **asp-validation-for**="PhoneNumber" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="ShortDescription" **type**="text" class="form-control" placeholder="Короткое описание, будет отображаться на карте">

<**span** **asp-validation-for**="ShortDescription" class="text-danger"></**span**>

</div>

<div class="py-2">

<**textarea** **asp-for**="Description" type="text" rows="10" class="form-control" placeholder="Полное описание машего места. Сюда можете включить информацио о времени работы заведения, чем может зацепить место и т.д."></**textarea**>

<**span** **asp-validation-for**="Description" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Address" class="form-control" placeholder="Улица, дом">

<**span** **asp-validation-for**="Address" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="AdditionalInfoAddress" **type**="text" class="form-control" placeholder="Дополнительная информация как до Вас добраться">

<**span** **asp-validation-for**="AdditionalInfoAddress" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="ObjectURL" **type**="text" class="form-control" placeholder="Веб-сайт">

<**span** **asp-validation-for**="ObjectURL" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="AdditionalInfoApplication" class="form-control" placeholder="Дополнительная информация к заяявлению">

<**span** **asp-validation-for**="AdditionalInfoApplication" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Images" class="form-control" **type**="file" id="formFileMultiple" multiple accept="image/jpeg" />

<**span** **asp-validation-for**="Images" class="text-danger"></**span**>

</div>

</div>

<div class="py-2 align-content-center">

<**button** type="submit" **asp-action**="CreateApplicationObject" class="btn btn-success">Отправить</**button**>

</div>

</div>

</div>

</div>

</**form**>

</div>

<style>

#marker {

background-image: url('/images/pin.png');

width: 32px;

height: 32px;

position: absolute;

top: 35%;

left: 65%;

z-index: 1;

}

.header {

padding: 5px;

}

</style>

<script src="https://api-maps.yandex.ru/2.1/?lang=ru-RU&amp;apikey=2901eb02-0489-4004-96be-a5b85aaa9984" type="text/javascript"></script>

<script src="https://yandex.st/jquery/2.2.3/jquery.min.js" type="text/javascript"></script>

<**script** src="~/js/ymaps\_create\_application.js" type="text/javascript"></**script**>

@section Scripts {

<**partial** **name**="\_ValidationScriptsPartial" />

}

**MyApplicationAccount.cshtml**

@model IEnumerable<VladimirTripAdvisor.Models.ApplicationModelDB>

@{

ViewData["Title"] = "Мои заявки";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div class="container">

<div class="row pt-4">

<div class="col-6">

<h2 class="text-primary">Список заявок на улучшение статуса аккаунта</h2>

</div>

</div>

<br />

@if (Model.Count() > 0)

{

<table class="table table-bordered table-striped" style="width:100%">

<thead>

<tr>

<th>

Тип заявки

</th>

<th>

Статус заявления

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var obj in Model)

{

if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Created)

{

<tr>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="40%">Улучшение аккаунта</td>

}

else

{

<td width="40%">Добавление объекта</td>

}

<td width="40%">Создана</td>

<td class="text-center">

</td>

</tr>

}

else if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Taked)

{

<tr>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="40%">Улучшение аккаунта</td>

}

else

{

<td width="40%">Добавление объекта</td>

}

<td width="40%">На рассмотрении администратором</td>

<td class="text-center"></td>

</tr>

}

else if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Processing)

{

<tr>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="40%">Улучшение аккаунта</td>

}

else

{

<td width="40%">Добавление объекта</td>

}

<td width="40%">В обработке</td>

<td></td>

</tr>

}

else if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Edit)

{

<tr>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="40%">Улучшение аккаунта</td>

}

else

{

<td width="40%">Добавление объекта</td>

}

<td width="40%">На редактировании</td>

<td>

<div class="w-75 btn-group" role="group">

<**a** **asp-controller**="Application" **asp-route-Id**="@obj.Id" **asp-action**="ViewApplication" class="btn btn-primary">

Открыть

</**a**>

</div>

</td>

</tr>

}

else if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Done)

{

<tr>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="40%">Улучшение аккаунта</td>

}

else

{

<td width="40%">Добавление объекта</td>

}

<td width="40%">Принята</td>

<td></td>

</tr>

}

else if (obj.ApplicationStatus == Logic.Application.ApplicationStatus.Canceled)

{

<tr>

@if (obj.ApplicationType == Logic.Application.ApplicationType.UpgradeAccount)

{

<td width="40%">Улучшение аккаунта</td>

}

else

{

<td width="40%">Добавление объекта</td>

}

<td>Отклонена администратором</td>

<td class="text-center"></td>

</tr>

}

}

</tbody>

</table>

}

else

{

<p>Список пустой</p>

}

</div>

**UpgradeAccountApplication.cshtml**

@model VladimirTripAdvisor.ViewModels.ApplicationAccountViewModel

@{

ViewData["Title"] = "Создание заявления";

}

<div class="col-12 text-success text-center py-4">

<**form** method="post">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<div class="row form-group">

<h2 class="col-12 text-success text-center py-4">Заявление на улучшение статуса аккаунта до владельца.</h2>

<div class="col-10 offset-2">

<div class="card">

<div class="card-body px-lg-5 pt-4">

<**input** **asp-for**="UserID" **type**="hidden" class="form-control">

<**input** **asp-for**="ApplicationId" **type**="hidden" class="form-control">

<div class="py-2">

<**input** **asp-for**="Name" **type**="text" class="form-control" placeholder="Ваше имя">

<**span** **asp-validation-for**="Name" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Surname" **type**="text" class="form-control" placeholder="Ваша фамилия">

<**span** **asp-validation-for**="Surname" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Midname" **type**="text" class="form-control" placeholder="Ваше отчество, если отсутствует, оставьте поле пустым">

<**span** **asp-validation-for**="Midname" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Email" **type**="email" class="form-control" placeholder="Электронная почта">

<**span** **asp-validation-for**="Email" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="CompanyName" **type**="text" class="form-control" placeholder="Наименование организации">

<**span** **asp-validation-for**="CompanyName" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="PhoneNumber" **type**="text" class="form-control" placeholder="Номер телефона организации">

<**span** **asp-validation-for**="PhoneNumber" class="text-danger"></**span**>

</div>

<div class="py-2">

<**textarea** **asp-for**="AdditionalInfo" rows="10" class="form-control" placeholder="Опишите здесь различную информацию для положительного рассмотрения заявления. Например, есть ли у вас заведения, различные места для посещения"></**textarea**>

<**span** **asp-validation-for**="AdditionalInfo" class="text-danger"></**span**>

</div>

</div>

<div class="py-2">

<**button** type="submit" **asp-action**="UpgradeAccountApplication" class="btn btn-success form-control">Отправить</**button**>

</div>

</div>

</div>

</div>

</**form**>

</div>

@section Scripts {

<**partial** **name**="\_ValidationScriptsPartial" />

}

**ViewApplicationAccount.cshtml**

@model VladimirTripAdvisor.ViewModels.ApplicationAccountViewModel

@{

ViewData["Title"] = "Заявка";

}

<div class="col-12 text-success text-center py-4">

<**form** method="post">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<div class="row form-group">

<h2 class="col-12 text-success text-center py-4">Заявление на улучшение статуса аккаунта до владельца.</h2>

<div class="col-10 offset-2">

<div class="card">

@if (Model.ApplicationStatus == Logic.Application.ApplicationStatus.Taked

|| Model.ApplicationStatus == Logic.Application.ApplicationStatus.Processing

|| Model.ApplicationStatus == Logic.Application.ApplicationStatus.Created)

{

<div class="card-body px-lg-5 pt-4">

<**input** **asp-for**="UserID" **type**="hidden" class="form-control">

<**input** **asp-for**="ApplicationId" **type**="hidden" class="form-control">

<div class="py-2">

<**input** **asp-for**="Name" class="form-control" readonly placeholder="Имя">

<**span** **asp-validation-for**="Name" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Surname" class="form-control" readonly placeholder="Фамилия">

<**span** **asp-validation-for**="Surname" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Midname" class="form-control" readonly placeholder="Отчество">

<**span** **asp-validation-for**="Midname" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Email" **type**="email" class="form-control" readonly placeholder="Отчество">

<**span** **asp-validation-for**="Email" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="CompanyName" class="form-control" readonly placeholder="Отчество">

<**span** **asp-validation-for**="CompanyName" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="PhoneNumber" class="form-control" readonly placeholder="Отчество">

<**span** **asp-validation-for**="PhoneNumber" class="text-danger"></**span**>

</div>

<div class="py-2">

<**textarea** **asp-for**="AdditionalInfo" rows="10" readonly class="form-control" placeholder="Опишите здесь различную информацию для положительного рассмотрения заявления. Например, есть ли у вас заведения, различные места для посещения"></**textarea**>

<**span** **asp-validation-for**="AdditionalInfo" class="text-danger"></**span**>

</div>

<div class="py-2">

<**textarea** **asp-for**="AdminComment" rows="10" class="form-control" placeholder="Комментарий администратора"></**textarea**>

<**span** **asp-validation-for**="AdminComment" class="text-danger"></**span**>

</div>

</div>

<div class="py-2">

<**button** type="submit" **asp-action**="SubmitApplicationAccount" **asp-route-Id**="@Model.ApplicationId" class="btn btn-success form-control">Подтвердить</**button**>

<**button** type="submit" **asp-action**="SendOnEditApplicationAccount" class="btn btn-warning form-control">Отправить на доработку</**button**>

<**button** type="submit" **asp-action**="DenyApp" **asp-route-Id**="@Model.ApplicationId" class="btn btn-danger form-control">Отклонить</**button**>

</div>

}

else if (Model.ApplicationStatus == Logic.Application.ApplicationStatus.Edit)

{

<div class="card-body px-lg-5 pt-4">

<**input** **asp-for**="UserID" **type**="hidden" class="form-control">

<**input** **asp-for**="ApplicationId" **type**="hidden" class="form-control">

<**input** **asp-for**="ApplicationStatus" **type**="hidden" class="form-control">

<div class="py-2">

<**input** **asp-for**="Name" class="form-control" placeholder="Имя">

<**span** **asp-validation-for**="Name" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Surname" class="form-control" placeholder="Фамилия">

<**span** **asp-validation-for**="Surname" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Midname" class="form-control" placeholder="Отчество">

<**span** **asp-validation-for**="Midname" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="Email" **type**="email" class="form-control" placeholder="Отчество">

<**span** **asp-validation-for**="Email" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="CompanyName" class="form-control" placeholder="Отчество">

<**span** **asp-validation-for**="CompanyName" class="text-danger"></**span**>

</div>

<div class="py-2">

<**input** **asp-for**="PhoneNumber" class="form-control" placeholder="Отчество">

<**span** **asp-validation-for**="PhoneNumber" class="text-danger"></**span**>

</div>

<div class="py-2">

<**textarea** **asp-for**="AdditionalInfo" rows="10" class="form-control" placeholder="Опишите здесь различную информацию для положительного рассмотрения заявления. Например, есть ли у вас заведения, различные места для посещения"></**textarea**>

<**span** **asp-validation-for**="AdditionalInfo" class="text-danger"></**span**>

</div>

<div class="py-2">

<**textarea** **asp-for**="AdminComment" rows="10" readonly class="form-control"></**textarea**>

<**span** **asp-validation-for**="AdminComment" class="text-danger"></**span**>

</div>

</div>

<div class="py-2">

<**button** type="submit" **asp-action**="UpgradeAccountApplication" **asp-route-Id**="@Model.ApplicationId" class="btn btn-success form-control">Отправить</**button**>

</div>

}

</div>

</div>

</div>

</**form**>

</div>

@section Scripts {

<**partial** **name**="\_ValidationScriptsPartial" />

}

**ViewApplicationObject.cshtml**

@model VladimirTripAdvisor.ViewModels.ApplicationObjectViewModelWithoutRequiredImage

@{

ViewData["Title"] = "Создание заявления";

}

<div class="col-12 text-success text-center">

<form method="post" enctype="multipart/form-data">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="row form-group">

<h2 class="col-12 text-success text-center py-4">Заявление на добавление новго объекта в систему.</h2>

<div class="col-12">

<div class="bg-white">

@if (Model.ApplicationStatus == Logic.Application.ApplicationStatus.Taked

|| Model.ApplicationStatus == Logic.Application.ApplicationStatus.Processing

|| Model.ApplicationStatus == Logic.Application.ApplicationStatus.Created)

{

<div class="px-lg-5 pt-4 col-6 float-end">

<div class="container justify-content-center">

<div id="map" class="col-6" style="width: 512px; height: 400px">

<div id="marker"></div>

</div>

</div>

<div class="pt-4">

<div class="py-2">

<input asp-for="Latitude" type="text" readonly id="latitude" class="form-control" placeholder="Широта">

<span asp-validation-for="Latitude" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="Longitude" type="text" readonly id="longitude" class="form-control" placeholder="Долгота">

<span asp-validation-for="Longitude" class="text-danger"></span>

</div>

</div>

</div>

<div class="card-body px-lg-5 pt-4 col-6">

<input asp-for="UserID" type="hidden" class="form-control">

<input asp-for="ApplicationId" type="hidden" class="form-control">

<div class="py-2">

<textarea asp-for="AdminComment" type="text" rows="10" class="form-control" placeholder="Комментарий администратора"></textarea>

<span asp-validation-for="AdminComment" class="text-danger"></span>

</div>

<div class="py-2">

<select asp-for="PlaceType" disabled class="form-control" asp-items="Html.GetEnumSelectList<PlaceType>()"></select>

<span asp-validation-for="PlaceType" class="text-danger"></span>

</div>

<div class="py-2">

<select asp-for="PlaceType" hidden class="form-control" asp-items="Html.GetEnumSelectList<PlaceType>()"></select>

<span asp-validation-for="PlaceType" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="Name" type="text" readonly class="form-control" placeholder="Название места">

<span asp-validation-for="Name" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="PhoneNumber" type="text" readonly class="form-control" placeholder="Номер телефона объекта">

<span asp-validation-for="PhoneNumber" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="ShortDescription" readonly type="text" class="form-control" placeholder="Короткое описание, будет отображаться на карте">

<span asp-validation-for="ShortDescription" class="text-danger"></span>

</div>

<div class="py-2">

<textarea asp-for="Description" readonly type="text" rows="10" class="form-control" placeholder="Полное описание машего места. Сюда можете включить информацио о времени работы заведения, чем может зацепить место и т.д."></textarea>

<span asp-validation-for="Description" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="Address" readonly class="form-control" placeholder="Улица, дом">

<span asp-validation-for="Address" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="AdditionalInfoAddress" readonly type="text" class="form-control" placeholder="Дополнительная информация как до Вас добраться">

<span asp-validation-for="AdditionalInfoAddress" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="ObjectURL" type="text" readonly class="form-control" placeholder="Веб-сайт">

<span asp-validation-for="ObjectURL" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="AdditionalInfoApplication" readonly class="form-control" placeholder="Дополнительная информация к заяявлению">

<span asp-validation-for="AdditionalInfoApplication" class="text-danger"></span>

</div>

<div id="carousel-1" class="carousel slide py-2" data-bs-ride="false" style="width: 100%;">

<div class="carousel-inner">

<div class="carousel-item active"><img class="w-100 d-block" src="data:image/jpg;base64, @Model.ImagesBase64.First()" alt="Slide Image" style="height: 450px; width:750px" /></div>

@for (int i = 1; i < Model.ImagesBase64.Count; i++)

{

<div class="carousel-item" data-bs-interval="">

<img class="w-100 d-block" src="data:image/jpg;base64, @Model.ImagesBase64[i]" alt="Slide Image" style="height: 450px; width:750px" />

</div>

}

</div>

<div><a class="carousel-control-prev" href="#carousel-1" role="button" data-bs-slide="prev"><span class="carousel-control-prev-icon"></span><span class="visually-hidden">Previous</span></a><a class="carousel-control-next" href="#carousel-1" role="button" data-bs-slide="next"><span class="carousel-control-next-icon"></span><span class="visually-hidden">Next</span></a></div>

</div>

</div>

<div class="py-2">

<button type="submit" asp-action="SubmitApplicationObject" asp-route-Id="@Model.ApplicationId" class="btn btn-success form-control">Подтвердить</button>

<button type="submit" asp-action="SendOnEditApplicationObject" class="btn btn-warning form-control">Отправить на доработку</button>

<button type="submit" asp-action="DenyApp" asp-route-Id="@Model.ApplicationId" class="btn btn-danger form-control">Отклонить</button>

</div>

}

else if (Model.ApplicationStatus == Logic.Application.ApplicationStatus.Edit)

{

<div class="px-lg-5 pt-4 col-6 float-end">

<div class="container justify-content-center">

<div id="map" class="col-6" style="width: 512px; height: 400px">

<div id="marker"></div>

</div>

</div>

<div class="pt-4">

<div class="py-2">

<input asp-for="Latitude" type="text" id="latitude" class="form-control" placeholder="Широта">

<span asp-validation-for="Latitude" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="Longitude" type="text" id="longitude" class="form-control" placeholder="Долгота">

<span asp-validation-for="Longitude" class="text-danger"></span>

</div>

</div>

</div>

<div class="card-body px-lg-5 pt-4 col-6">

<input asp-for="UserID" type="hidden" class="form-control">

<input asp-for="ApplicationId" type="hidden" class="form-control">

<div class="py-2">

<textarea asp-for="AdminComment" type="text" readonly rows="10" class="form-control" placeholder="Комментарий администратора"></textarea>

<span asp-validation-for="AdminComment" class="text-danger"></span>

</div>

<div class="py-2">

<select asp-for="PlaceType" class="form-control" asp-items="Html.GetEnumSelectList<PlaceType>()"></select>

<span asp-validation-for="PlaceType" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="Name" type="text" class="form-control" placeholder="Название места">

<span asp-validation-for="Name" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="PhoneNumber" type="text" class="form-control" placeholder="Номер телефона объекта">

<span asp-validation-for="PhoneNumber" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="ShortDescription" type="text" class="form-control" placeholder="Короткое описание, будет отображаться на карте">

<span asp-validation-for="ShortDescription" class="text-danger"></span>

</div>

<div class="py-2">

<textarea asp-for="Description" type="text" rows="10" class="form-control" placeholder="Полное описание машего места. Сюда можете включить информацио о времени работы заведения, чем может зацепить место и т.д."></textarea>

<span asp-validation-for="Description" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="Address" class="form-control" placeholder="Улица, дом">

<span asp-validation-for="Address" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="AdditionalInfoAddress" type="text" class="form-control" placeholder="Дополнительная информация как до Вас добраться">

<span asp-validation-for="AdditionalInfoAddress" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="ObjectURL" type="text" class="form-control" placeholder="Веб-сайт">

<span asp-validation-for="ObjectURL" class="text-danger"></span>

</div>

<div class="py-2">

<input asp-for="AdditionalInfoApplication" class="form-control" placeholder="Дополнительная информация к заяявлению">

<span asp-validation-for="AdditionalInfoApplication" class="text-danger"></span>

</div>

<div class="py-2">

<input class="form-control" type="file" id="formFileMultiple" multiple />

<span asp-validation-for="Images" class="text-danger"></span>

</div>

<div id="carouselIndicators" class="carousel slide py-2" data-bs-ride="true">

<div class="carousel-inner">

<div class="carousel-item active" data-bs-interval="">

<img src="data:image/jpg;base64, @Model.ImagesBase64[0]" class="d-block w-100" alt="...">

</div>

@for (int i = 1; i < Model.ImagesBase64.Count; i++)

{

<div class="carousel-item" data-bs-interval="">

<img src="data:image/jpg;base64, @Model.ImagesBase64[i]" class="d-block w-100" alt="...">

</div>

}

</div>

<button class="carousel-control-prev" type="button" data-bs-target="#carouselIndicators" data-bs-slide="prev">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

<span class="visually-hidden">Previous</span>

</button>

<button class="carousel-control-next" type="button" data-bs-target="#carouselIndicators" data-bs-slide="next">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

<span class="visually-hidden">Next</span>

</button>

</div>

</div>

<div class="py-2 align-content-center">

<button type="submit" asp-action="SubmitApplicationObjectEdit" class="btn btn-success">Отправить</button>

</div>

}

</div>

</div>

</div>

</form>

</div>

<style>

#marker {

background-image: url('/images/geo\_pin.png');

width: 32px;

height: 32px;

position: absolute;

top: 35%;

left: 65%;

z-index: 1;

}

.header {

padding: 5px;

}

</style>

<script src="https://api-maps.yandex.ru/2.1/?lang=ru-RU&amp;apikey=2901eb02-0489-4004-96be-a5b85aaa9984" type="text/javascript"></script>

<script src="https://yandex.st/jquery/2.2.3/jquery.min.js" type="text/javascript"></script>

<script src="~/js/ymaps\_create\_application.js" type="text/javascript"></script>

@section Scripts {

<partial name="\_ValidationScriptsPartial" />

}

**AllEvents.cshtml**

@model IEnumerable<VladimirTripAdvisor.ViewModels.EventModelWithPhotoViewModel>

@{

ViewData["Title"] = "AllEvents";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div class="container">

<div class="row pt-4">

<div class="col-6">

<h2 class="text-primary">Список публичных встреч</h2>

</div>

</div>

<br />

@if (Model.Count() > 0)

{

<div class="row row-cols-1 row-cols-md-2 g-4">

@foreach (var obj in Model)

{

<div class="card">

<img class="img-fluid card-img-top w-100 d-block card-img-top w-100 card-img-top w-100 card-img-top w-100 card-img-top" alt="..."

style="height: 400px;object-fit: contain;" src="data:image/jpg;base64, @obj.ImageMainBase64" />

<div class="card-body">

<div class="d-flex flex-row align-items-center" style="margin-bottom: 10px;">

<h5 class="card-title" style="width: max-content;padding-bottom: 0px;margin-bottom: 0px;">@obj.EventModel.Name</h5>

</div>

<p class="card-text">@obj.EventModel.Description</p>

<p class="card-text">Цель встречи: @obj.PlaceName</p>

<p class="card-text">Дата встречи: @obj.EventModel.StartDate.ToString("dd MMMM hh:mm")</p>

</div>

<div class="card-footer">

<div class="d-flex justify-content-around">

<a href="@obj.EventModel.TelegramLink" class="btn btn-primary">

Участвовать

</a>

</div>

</div>

</div>

}

</div>

}

else

{

<p>Список пустой</p>

}

</div>

**MyEvents.cshtml**

@model IEnumerable<VladimirTripAdvisor.ViewModels.EventViewModel>

@{

ViewData["Title"] = "Мои встречи";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div class="container">

<div class="row pt-4">

<div class="col-6">

<h2 class="text-primary">Список моих встреч</h2>

</div>

</div>

<br />

@if (Model.Count() > 0)

{

<table class="table table-bordered table-striped" style="width:100%">

<thead>

<tr>

<th>

Название

</th>

<th>

Место встречи

</th>

<th>

Дата встречи

</th>

<th>

Ссылка

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var obj in Model)

{

<tr>

<td width="25%">@obj.Event.Name</td>

<td width="25%">@obj.PlaceName</td>

<td width="25%">@obj.Event.StartDate</td>

<td width="25%">

<a href="@obj.Event.TelegramLink">@obj.Event.TelegramLink</a>

</td>

</tr>

}

</tbody>

</table>

}

else

{

<p>Список пустой</p>

}

</div>

**Commentary.cshtml**

@model IEnumerable<VladimirTripAdvisor.ViewModels.ReviewViewModel>

@foreach (var review in Model)

{

<div style="height: auto;border-top: 1px solid rgba(55,57,59,0.48);margin-bottom: 20px;">

<div class="d-flex justify-content-between" style="width: 100%;height: 35px;border-color: var(--bs-purple);margin-top: 15px;">

<div class="d-flex flex-row" style="height: 32px;width: auto;">

<span style="width: 32px;height: 32px;"><**img** src="~/images//user.png" style="width: 100%;height: 100%;" /></span>

<p class="d-flex justify-content-center align-items-center align-content-center" style="height: 100%;padding-left: 7px;padding-right: 7px;margin-bottom: 0px;">@review.Name @review.Surname.Substring(0,1)</p>

</div>

@if (User.IsInRole(WC.AdminRole))

{

<div class="d-flex justify-content-center align-items-center" style="width: 69px;">

<button class="d-flex justify-content-start align-items-center" type="button"

data-bs-toggle="modal" data-bs-target="#removeReviewModal\_@review.Review.Id"

style="height: 100%;font-weight: bold;font-size: 34px; border-style:none; border-radius: 0px;">

<strong>

<span style="color:red;"><i class="fa-solid fa-trash"></i></span>

</strong>

</button>

</div>

}

</div>

<div style="margin-top: 16px;">

<span>

@if (review.Review.Score == ReviewScore.Excellent)

{

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

}

else if (review.Review.Score == ReviewScore.Good)

{

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

}

else if (review.Review.Score == ReviewScore.Average)

{

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

}

else if (review.Review.Score == ReviewScore.Poor)

{

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

}

else

{

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

}

</span>

</div>

<div style="margin-top: 8px;margin-bottom: 4px;">

<p class="fw-bold" style="font-size: 20px;text-align: justify;margin-bottom: 0px;">@review.Review.ReviewName</p>

</div>

<div>

<p style="font-style: italic;">@review.Review.ReviewDate.ToLongDateString()</p>

</div>

<div>

<p style="margin-bottom: 4px;">@review.Review.ReviewDescription</p>

</div>

</div>

<div id="removeReviewModal\_@review.Review.Id" class="modal fade" role="dialog" tabindex="-1">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-body d-flex justify-content-center align-items-center align-content-center">

<p class="text-center" style="padding-bottom: 0px;margin-bottom: 0px; font-size:20px;">Удалить отзыв?</p>

</div>

<div class="modal-footer">

<button class="btn btn-primary" type="button" data-bs-dismiss="modal">Нет</button>

<button class="btn btn-danger">

<**a** **asp-route-reviewId**="@review.Review.Id" **asp-route-objectId**="@review.Review.ObjectId"

**asp-controller**="Review" **asp-action**="RemoveReview" style="color: #ffffff; text-decoration:none">

Да

</**a**>

</button>

</div>

</div>

</div>

</div>

}

**CommentarySummary.cshtml**

@model VladimirTripAdvisor.ViewModels.ReviewSummary

<div class="col-4 d-flex align-self-start align-self-sm-center align-self-xl-start" style="height: auto;width: auto;margin-bottom: 60px;">

<div class="d-flex flex-column" style="width: 100%;height: auto;">

<div class="d-flex" style="padding-bottom: 0px;height: 59px;">

<p class="fw-semibold text-start d-flex align-items-center" style="margin-bottom: 0px;padding-left: 14px;font-size: 20px;">@Model.AverageRating</p>

<span class="d-flex justify-content-center align-items-center" style="padding-left: 1px;margin-left: 14px;">

<**img** src="~/images/star.png" />

</span>

<p class="fw-semibold d-flex justify-content-center align-items-center" style="height: auto;margin-bottom: 0px;margin-left: 14px;">@Model.ReviewsCount отзывов</p>

</div>

<div class="d-flex align-content-center" style="min-height: auto;margin-top: 13px;height: auto;">

<p class="fw-semibold" style="width: 80px;padding-left: 5px;height: 32px;margin-bottom: 0px;">Отлично</p>

<span>

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

</span>

<p class="fw-normal" style="width: 80px;padding-left: 13px;height: 32px;margin-bottom: 0px;font-size: 14px;padding-top: 4px;">@Model.ExcellentReviews</p>

</div>

<div class="d-flex align-content-center" style="min-height: auto;margin-top: 13px;height: auto;">

<p class="fw-semibold" style="width: 80px;padding-left: 5px;height: 32px;margin-bottom: 0px;">Хорошо</p>

<span>

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

</span>

<p class="fw-normal" style="width: 80px;padding-left: 13px;height: 32px;margin-bottom: 0px;font-size: 14px;padding-top: 4px;">@Model.GoodReviews</p>

</div>

<div class="d-flex align-content-center" style="min-height: auto;margin-top: 13px;height: auto;">

<p class="fw-semibold" style="width: 80px;padding-left: 5px;height: 32px;margin-bottom: 0px;">Средне</p>

<span>

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

</span>

<p class="fw-normal" style="width: 80px;padding-left: 13px;height: 32px;margin-bottom: 0px;font-size: 14px;padding-top: 4px;">@Model.AverageReviews</p>

</div>

<div class="d-flex align-content-center" style="min-height: auto;margin-top: 13px;height: auto;">

<p class="fw-semibold" style="width: 80px;padding-left: 5px;height: 32px;margin-bottom: 0px;">Плохо</p>

<span>

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

</span>

<p class="fw-normal" style="width: 80px;padding-left: 13px;height: 32px;margin-bottom: 0px;font-size: 14px;padding-top: 4px;">@Model.PoorReviews</p>

</div>

<div class="d-flex align-content-center" style="min-height: auto;margin-top: 13px;height: auto;">

<p class="fw-semibold" style="width: 80px;padding-left: 5px;height: 32px;margin-bottom: 0px;">Ужасно</p>

<span>

<**img** src="~/images/star.png" style="width: 26px;height: 26px;" />

</span>

<p class="fw-normal" style="width: 80px;padding-left: 13px;height: 32px;margin-bottom: 0px;font-size: 14px;padding-top: 4px;">@Model.TerribleReviews</p>

</div>

</div>

</div>

**AllObjects.cshtml**

@model IList<VladimirTripAdvisor.ViewModels.ObjectWithPhotoViewModel>

@{

ViewData["Title"] = "Каталог объектов";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div class="container">

<div class="row pt-4 d-flex justify-content-center">

<div class="col-6 d-flex justify-content-center">

<h2 class="text-primary">Каталог объектов</h2>

</div>

</div>

<br />

@if (Model.Count() > 0)

{

<div class="row row-cols-2 d-flex justify-content-center g-4">

@foreach (var obj in Model)

{

<div class="col" style="width: 500px;">

<div class="card">

<img class="img-fluid card-img-top w-100 d-block card-img-top w-100 card-img-top" alt="..."

style="height: 400px;object-fit: contain;" src="data:image/jpg;base64, @obj.ImageMainBase64" />

<div class="card-body" style="height:192px">

<div class="d-flex flex-row align-items-center" style="margin-bottom: 10px;">

<h5 class="card-title" style="width: max-content;padding-bottom: 0px;margin-bottom: 0px;">@obj.Place.Name</h5>

<div class="d-flex flex-row justify-content-center align-items-center align-content-center" style="margin-left: 10px;">

<span><**img** height="30" width="30" src="~/images/star.png" /></span>

<p style="font-weight: bold;margin-bottom: 0px;margin-left: 5px;">@obj.Place.AverageRating</p>

</div>

</div>

<p class="card-text">@obj.Place.ShortDescription</p>

</div>

<div class="card-footer">

@if (User.IsInRole(WC.AdminRole))

{

<div class="d-flex justify-content-around">

<**a** **asp-controller**="Object" **asp-route-Id**="@obj.Place.Id" **asp-action**="ViewObject" class="btn btn-primary">

Перейти

</**a**>

<**a** **asp-controller**="Object" **asp-route-Id**="@obj.Place.Id" **asp-action**="RemoveObject" class="btn btn-danger">

Удалить

</**a**>

</div>

}

else

{

<div class="d-flex justify-content-center">

<**a** **asp-controller**="Object" **asp-route-Id**="@obj.Place.Id" **asp-action**="ViewObject" class="btn btn-primary">

Перейти

</**a**>

</div>

}

</div>

</div>

</div>

}

</div>

}

else

{

<p>Список пустой</p>

}

</div>

**Map.cshtml**

@{

ViewData["Title"] = "Карта";

}

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<!--

Укажите свой API-ключ. Тестовый ключ НЕ БУДЕТ работать на других сайтах.

Получить ключ можно в Кабинете разработчика: https://developer.tech.yandex.ru/keys/

-->

<script src="https://api-maps.yandex.ru/2.1/?lang=ru-RU&amp;apikey=2901eb02-0489-4004-96be-a5b85aaa9984" type="text/javascript"></script>

<script src="https://yandex.st/jquery/2.2.3/jquery.min.js" type="text/javascript"></script>

<script src="/js/ymaps\_object\_manager.js" type="text/javascript"></script>

<style>

#map {

width: 100%;

height: 100%;

padding: 0;

margin: 0;

}

a {

color: #04b; /\* Цвет ссылки \*/

text-decoration: none; /\* Убираем подчеркивание у ссылок \*/

}

a:hover {

color: #f50000; /\* Цвет ссылки при наведении на нее курсора мыши \*/

}

</style>

<body>

<div class="container d-flex justify-content-center">

<div id="map" class="col-6" style="width: 1200px; height: 600px"></div>

</div>

</body>

**ViewMyObjects.cshtml**

@model IList<VladimirTripAdvisor.ViewModels.ObjectWithPhotoViewModel>

@{

ViewData["Title"] = "ViewMyObjects";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div class="container">

<div class="row pt-4">

<div class="col-6">

<h2 class="text-primary">Каталог объектов</h2>

</div>

</div>

<br />

@if (Model.Count() > 0)

{

<div class="row row-cols-1 row-cols-md-2 g-4">

@foreach (var obj in Model)

{

<div class="col" style="width: 500px;">

<div class="card">

<img class="img-fluid card-img-top w-100 d-block card-img-top w-100 card-img-top" alt="..."

style="height: 400px;object-fit: contain;" src="data:image/jpg;base64, @obj.ImageMainBase64" />

<div class="card-body">

<div class="d-flex flex-row align-items-center" style="margin-bottom: 10px;">

<h5 class="card-title" style="width: max-content;padding-bottom: 0px;margin-bottom: 0px;">@obj.Place.Name</h5>

<div class="d-flex flex-row justify-content-center align-items-center align-content-center" style="margin-left: 10px;">

<span><**img** height="30" width="30" src="~/images/star.png" /></span>

<p style="font-weight: bold;margin-bottom: 0px;margin-left: 5px;">@obj.Place.AverageRating</p>

</div>

</div>

<p class="card-text">@obj.Place.ShortDescription</p>

</div>

<div class="card-footer">

@if (User.IsInRole(WC.AdminRole))

{

<div class="d-flex justify-content-around">

<**a** **asp-controller**="Object" **asp-route-Id**="@obj.Place.Id" **asp-action**="ViewObject" class="btn btn-primary">

Перейти

</**a**>

<**a** **asp-controller**="Object" **asp-route-Id**="@obj.Place.Id" **asp-action**="RemoveObject" class="btn btn-danger">

Удалить

</**a**>

</div>

}

else

{

<div class="d-flex justify-content-center">

<**a** **asp-controller**="Object" **asp-route-Id**="@obj.Place.Id" **asp-action**="ViewObject" class="btn btn-primary">

Перейти

</**a**>

</div>

}

</div>

</div>

</div>

}

</div>

}

else

{

<p>Список пустой</p>

}

</div>

**ViewObject.cshtml**

@using Microsoft.AspNetCore.Identity;

@inject UserManager<IdentityUser> UserManager

@model VladimirTripAdvisor.ViewModels.ObjectWithPhotoViewModel

@{

ViewData["Title"] = "ViewObject";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<style>

.truncate {

display: -webkit-box;

-webkit-line-clamp: 3;

text-overflow: ellipsis;

-webkit-box-orient: vertical;

overflow: hidden;

}

</style>

<body>

<div class="container flex-column" style="display: flex;padding: 40px;position: relative;overflow: visible;max-width: 100%;height: auto;width: 1216px;">

<p class="d-flex" style="font-weight: bold;font-size: 51px;">@Model.Place.Name</p>

<div class="row text-start d-flex flex-column flex-lg-row flex-xl-row flex-xxl-row" style="position: relative;display: flex;overflow: visible;padding-left: 0px;padding-right: 0px;height: auto;width: auto;margin-right: 12px;">

<div class="col-auto d-flex flex-row" style="width: auto;">

<span><img class="d-xxl-flex align-items-xxl-center" src="~/images/star.png" width="40" height="40" /></span>

<p class="fw-semibold d-flex align-items-center" style="margin-bottom: 0px;padding-bottom: 0px;padding-left: 10px;font-size: 22px;width: auto;padding-top: 3px;">@Model.Place.AverageRating</p>

</div>

<div class="col-auto col-lg-auto d-xxl-flex justify-content-xxl-center align-items-xxl-center" style="height: auto;width: auto;">

<p class="fw-semibold d-flex flex-grow-0 align-items-center" style="margin-bottom: 0px;width: 100%;height: 100%;">@Model.Place.StreetAddress</p>

</div>

<div class="col-auto col-lg-auto d-flex d-md-flex d-xxl-flex align-items-center justify-content-xxl-center" style="height: auto;width: auto;">

<p class="fw-semibold d-flex align-items-center" style="margin-bottom: 0px;width: auto;">@Model.Place.PhoneNumber</p>

</div>

<div class="col-auto col-lg-auto d-flex d-xxl-flex align-items-center justify-content-xxl-center" style="width: auto;">

<p class="fw-semibold d-flex align-items-center" style="padding-bottom: 0px;margin-bottom: 0px;width: 100%;height: 100%;"><a href="@Model.Place.PlaceURL">Веб-сайт</a></p>

</div>

<div class="col-auto col-lg-auto d-flex d-xxl-flex align-items-center justify-content-xxl-center" style="width: auto;">

<p class="fw-semibold text-success d-flex align-items-center" style="padding-bottom: 0px;margin-bottom: 0px;width: 100%;height: 100%;">Открыто</p>

</div>

@if (User.IsInRole(WC.UserRole) || User.IsInRole(WC.OwnerRole) || User.IsInRole(WC.AdminRole))

{

<div class="col-auto col-lg-auto d-flex d-xxl-flex align-items-center justify-content-xxl-center" style="width: auto;">

<button class="btn btn-warning d-flex justify-content-center align-items-center" type="button" style="height: 40px;" data-bs-target="#addReviewModal" data-bs-toggle="modal">

<strong>Оставить отзыв</strong>

</button>

</div>

<div class="col-auto col-lg-auto d-flex d-xxl-flex align-items-center justify-content-xxl-center" style="width: auto;">

<button class="btn btn-success d-flex justify-content-center align-items-center" type="button" style="height: 40px;" data-bs-target="#CreateEventModal" data-bs-toggle="modal">

<strong>Организовать встречу</strong>

</button>

</div>

}

</div>

<div class="row d-flex flex-column align-content-center flex-sm-column flex-md-column flex-lg-row flex-xl-row flex-xxl-row" style="padding-left: 0px;padding-top: 20px;height: auto;width: 100%;">

<div class="col-4 col-sm-12 col-lg-4 align-self-start" style="padding-left: 0px;height: auto;">

<div class="card" style="padding-right: 0px;margin-right: 0px;width: 100%;">

<div class="card-body">

<h4 class="card-title">Об объекте</h4>

<p class="flex-wrap card-text truncate">@Model.Place.Description</p><button id="toggle" class="btn btn-secondary toggle" type="button">Подробнее</button><button id="toggle" class="btn btn-secondary text-center d-none justify-content-xxl-center toggle" type="button" style="width: 120px;text-align: center;">Скрыть</button>

</div>

</div>

</div>

<div class="col order-first order-lg-last order-xl-last order-xxl-last col-md" style="height: auto;">

<div id="carousel-1" class="carousel slide" data-bs-ride="false" style="width: 100%;">

<div class="carousel-inner">

<div class="carousel-item active">

<img class="w-100 d-block" src="data:image/jpg;base64, @Model.Images.First()" alt="Slide Image"

style="object-fit: contain;object-position: center;overflow: hidden;height: 550px;" />

</div>

@for (int i = 1; i < Model.Images.Count; i++)

{

<div class="carousel-item">

<img class="w-100 d-block" src="data:image/jpg;base64, @Model.Images[i]" alt="Slide Image"

style="object-fit: contain;object-position: center;overflow: hidden;height: 550px;" />

</div>

}

</div>

<div>

<a class="carousel-control-prev" href="#carousel-1" role="button" data-bs-slide="prev">

<span class="carousel-control-prev-icon"></span>

<span class="visually-hidden">Previous</span>

</a>

<a class="carousel-control-next" href="#carousel-1" role="button" data-bs-slide="next">

<span class="carousel-control-next-icon"></span>

<span class="visually-hidden">Next</span>

</a>

</div>

</div>

<div class="row g-0" style="padding-top: 30px;height: auto;width: 100%;padding-bottom: 30px;">

<div class="col d-flex" style="padding-top: 0px;width: 100%;height: 100%;">

<div class="d-flex justify-content-center align-items-center" style="height: 100%;width: 100%;">

<div id="map" style="height: 400px;width: 100%;">

</div>

</div>

</div>

</div>

</div>

</div>

<div class="row d-flex flex-row flex-wrap flex-sm-column flex-xl-row" style="padding-top: 50px;width: 100%;height: auto;">

@await Component.InvokeAsync("CommentarySummary", new {id = Model.Place.Id})

<div class="col-6 offset-xl-2 offset-xxl-2 d-flex flex-column align-self-sm-center align-self-xl-start" style="height: auto;border-color: var(--bs-blue);padding-top: 4px;">

@await Component.InvokeAsync("Commentary", new {id = Model.Place.Id})

</div>

</div>

<div id="addReviewModal" class="modal fade" role="dialog" tabindex="-1">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title">Отзыв</h4><button class="btn-close" type="button" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<form method="post">

<div class="modal-body">

<input asp-for="ReviewModel.UserId" class="form-control form-control-sm" hidden type="text" value="@UserManager.GetUserId(User)" />

<input asp-for="ReviewModel.ObjectId" class="form-control form-control-sm" hidden type="text" value="@Model.Place.Id" />

<div style="margin-bottom: 10px;">

<label class="form-label" style="font-weight: bold;">Название:</label>

<input asp-for="ReviewModel.ReviewName" class="form-control form-control-sm" type="text" />

<span asp-validation-for="ReviewModel.ReviewName" class="text-danger"></span>

</div>

<div style="margin-bottom: 10px;">

<label class="form-label" style="font-weight: bold;">Опишите Ваши впечатления о посещенном месте:</label>

<textarea asp-for="ReviewModel.ReviewDescription" class="form-control" rows="4"></textarea>

<span asp-validation-for="ReviewModel.ReviewDescription" class="text-danger"></span>

</div>

<label class="form-label" style="font-weight: bold;">Ваша оценка:</label>

<select asp-for="ReviewModel.Score" class="form-control" asp-items="Html.GetEnumSelectList<ReviewScore>()"></select>

</div>

<div class="modal-footer">

<button class="btn btn-light" type="button" data-bs-dismiss="modal">Закрыть</button>

<button asp-controller="Review" asp-action="AddReview" class="btn btn-primary" type="submit">Готово</button>

</div>

</form>

</div>

</div>

</div>

<div id="CreateEventModal" class="modal fade" role="dialog" tabindex="-1">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header d-flex">

<h4 class="modal-title">Создание встречи</h4>

</div>

<form method="post">

<input asp-for="EventModel.PlaceId" class="form-control" hidden type="text" style="margin-bottom: 0px;" value="@Model.Place.Id" />

<input asp-for="EventModel.CreatorId" class="form-control form-control-sm" hidden type="text" value="@UserManager.GetUserId(User)" />

<div class="modal-body">

<p style="text-align: justify;">После создания встречи Вам будет выдана ссылка приглашение в чат Телеграм, куда потом подключатся другие пользователи или вы можете пригласить сами, отправив им ссылку</p>

<div style="margin-bottom: 15px;">

<p style="margin-bottom: 6px;">Название мероприятия:</p>

<input asp-for="EventModel.Name" class="form-control" type="text" style="margin-bottom: 0px;" />

<span asp-validation-for="EventModel.Name" class="text-danger"></span>

</div>

<div style="margin-bottom: 15px;">

<p style="margin-bottom: 6px;">Описание мероприятия</p>

<textarea asp-for="EventModel.Description" rows="4" class="form-control" type="text" style="margin-bottom: 0px;" >

</textarea>

<span asp-validation-for="EventModel.Description" class="text-danger"></span>

</div>

<div style="margin-bottom: 15px;">

<p style="margin-bottom: 6px;">Доступ к мероприятию</p>

<select asp-for="EventModel.EventType" class="form-control" asp-items="Html.GetEnumSelectList<EventType>()"></select>

</div>

<div style="margin-bottom: 15px;">

<p style="margin-bottom: 6px;">Дата начала мероприятия</p>

<input asp-for="EventModel.StartDate" class="form-control" type="datetime-local" style="margin-bottom: 0px;" />

<span asp-validation-for="EventModel.StartDate" class="text-danger" ></span>

</div>

@\* <div style="margin-bottom: 15px;">

<p style="margin-bottom: 6px;">Дата окончания мероприятия</p><input class="form-control form-control" type="datetime-local" style="margin-bottom: 0px;" />

<p class="text-danger" ></p>

</div>\*@

</div>

<div class="modal-footer">

<button class="btn btn-light" type="button" data-bs-dismiss="modal">Отмена</button>

<button asp-action="CreateEvent" asp-controller="Event" class="btn btn-primary" type="submit">Создать</button>

</div>

</form>

</div>

</div>

</div>

</div>

</body>

@section Scripts {

<partial name="\_ValidationScriptsPartial" />

}

<script src="https://api-maps.yandex.ru/2.1/?lang=ru-RU&amp;apikey=2901eb02-0489-4004-96be-a5b85aaa9984" type="text/javascript"></script>

<script type="text/javascript">

ymaps.ready(function () {

var myMap = new ymaps.Map('map', {

center: [@Model.Place.Latitude, @Model.Place.Longitude],

zoom: 16

}, {

searchControlProvider: 'yandex#search'

}),

// Создаём макет содержимого.

MyIconContentLayout = ymaps.templateLayoutFactory.createClass(

'<div style="color: #AAAAAA; font-weight: bold;">$[properties.iconContent]</div>'

),

myPlacemark = new ymaps.Placemark(myMap.getCenter(), {

}, {

iconLayout: 'default#image',

// Своё изображение иконки метки.

iconImageHref: 'https://localhost:7048/images/pin.png',

// Размеры метки.

iconImageSize: [32, 32],

});

myMap.controls.remove('searchControl');

myMap.controls.remove('geolocationControl');

myMap.controls.remove('trafficControl');

myMap.controls.remove('fullscreenControl');

myMap.controls.remove('rulerControl');

myMap.controls.remove('typeSelector');

myMap.geoObjects

.add(myPlacemark)

});

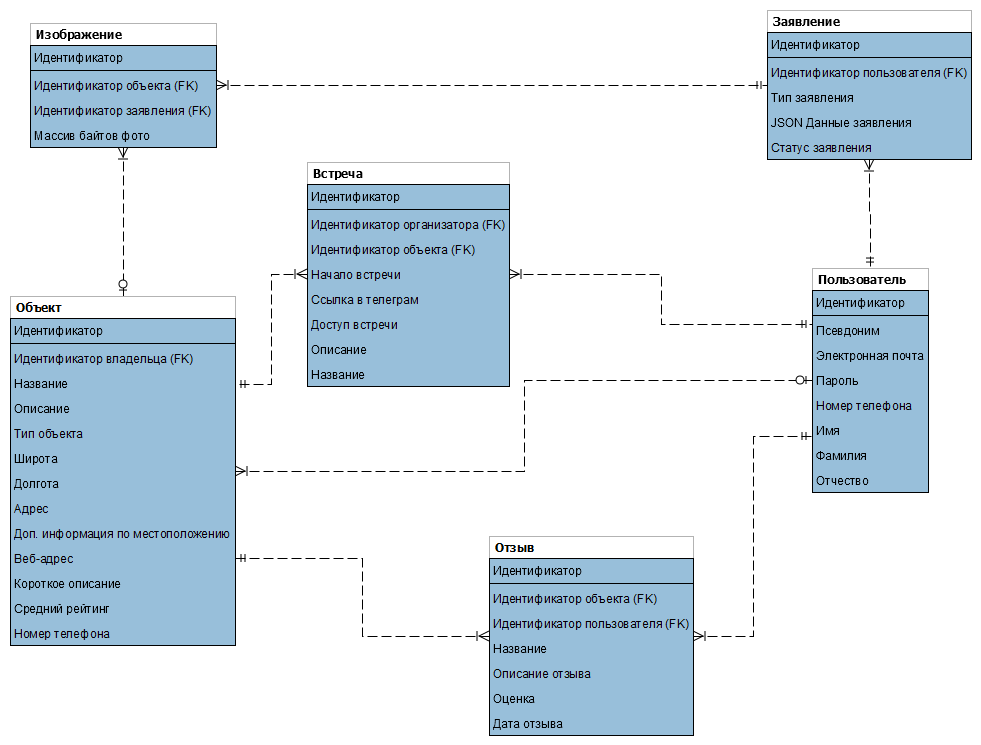
</script>

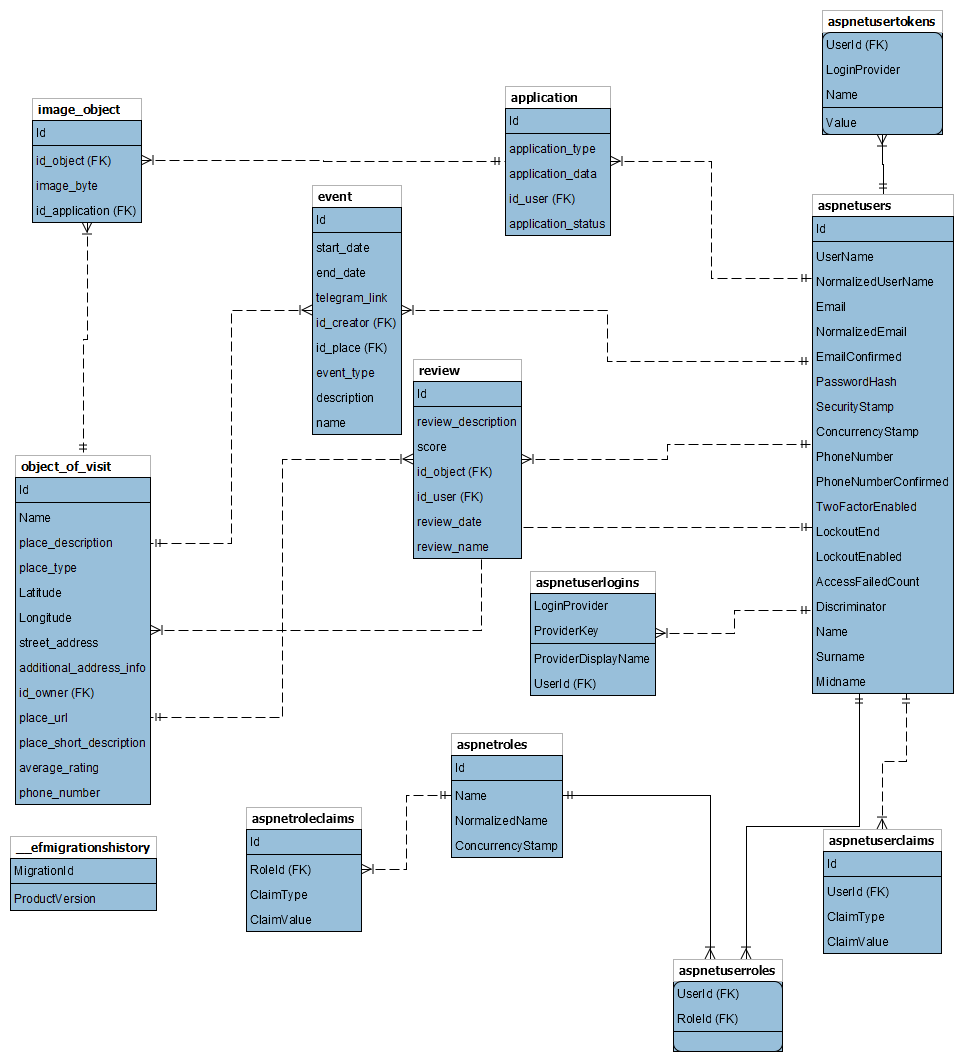
<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/js/btn\_readmore.js"></script>

# **ПРИЛОЖЕНИЕ Б**

**Структура таблиц БД**

****

****