



Универзитет „Св. Кирил и Методиј“ – Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО



ДИПЛОМСКА РАБОТА

Тема:

Веб апликација за планински туризам

Ментор:

проф. д-р Иван Китановски

Студент:

Елена Ртоска, 171200

Скопје, октомври 2021

Ментор:

Проф. д-р Иван Китановски,
ФИНКИ

Членови на комисија:

Проф. д-р Иван Китановски,
ФИНКИ

Проф. д-р Ивица Димитровски
ФИНКИ

Проф. м-р Влатко Спасев
ФИНКИ

Кандидат:

Елена Ртоска, 171200, КНИ

Апстракт

Во оваа дипломска работа е опишана целата постапка за резервација на планински туризам, односно креирање на веб апликација за онлајн резерваци на планински тури, која користи библиотеки и технологии што ни ја олеснуваат работата за креирањето на самата веб апликација. Целта на дипломската работа е да се прикаже целосно содржината поврзана со планински туризам на едно место – во случајов во веб апликација каде што сè се наоѓа тука, со цел да не се пребаруваат одредени настани или организации на различни сајтови за резервација на планинска тура.

Клучни зборови

Online резервации, SQL, веб апликација, веб програмирање, C#, JavaScript, Ajax, jQuery, ASP .NET MVC, CSS, HTML, Google Maps, GPS, кориснички интерфејс, API, Microsoft, авторизација, автентикација, Administrator, Editor, User, Application DB Context, веб-сервер, HTTP, GET, POST, Bootstrap, Date Time Picker

Содржина

1.	Вовед.....	5
1.1.	Како до идејата за оваа апликација	6
1.2.	Цел на овој проект и апликацијата	6
2.	Развој на системот	7
2.1.	Користени технологии, библиотеки и програмски јазици за реализација на веб апликацијата	7
2.1.1.	SQL	7
2.1.2.	JavaScript	8
2.1.3.	Ajax.....	8
2.1.4.	jQuery.....	9
2.1.5.	CSS & HTML.....	9
2.1.6.	C#	10
2.1.7.	ASP .NET MVC	10
2.1.8.	API	11
2.1.9.	Bootstrap.js & bootstrap-datetimepicker.js	11
2.2.	Етапи при развојот на веб апликацијата	12
3.	Функционалност на системот	13
3.1.	Регистрација	13
3.2.	Авторизација.....	14
3.3.	Автентикација	14
4.	Имплементација на апликацијата	14
4.1.	Архитектура на апликацијата	14
4.2.	Backend имплементација	19
4.3.	Frontend имплементација	28
5.	Приказ на апликацијата	33
6.	Заклучок.....	40
7.	Референци	41

1. Вовед

Здравиот живот, правилната исхрана, физичка спремност, кондиција, време поминато во природа, планинарење, трчање и многу други области се дел од нашето секојдневие и сè повеќе и повеќе имаат подем. Луѓето целат кон здрав живот и тоа е неопходно за секој од нас. Сè повеќе излегуваат статии за тоа како да се одржува кондицијата, препораки за вежби, организации за настани, online резервации и слично.

Оваа веб апликација има за цел да опфати различен тип на корисници и да придонесе за развој на планинскиот туризам во нашата земја бидејќи не е доволно развиен, иако Македонија има навистина многу убави планни. Целта е да се подобри планинскиот туризам и да има повеќе туристи од нашата, а и од други земји.

Сите начини на кои се пренесуваат препораки и се организираат настани се повеќе се дигитализираат и влегуваат во светот на технологијата, т.е се користи технологија за побрз пренос на работите, започнувајќи од настани на социјалните мрежи па се до веб апликации за резервации.

Со самото користење на технологиите се заштедува и време и пари, а има и подобар квалитет.

Денес речиси секој човек се сретнал на некој начин со технологијата, без разлика дали работи со неа или е корисник. Се очекува во иднина се повеќе да се зголемува бројот на корисници, бидејќи секојдневие то ни налага на некој начин да зависиме од технологијата, бидејќи денес е вметната и во самото образование.

Користењето на технологијата ни овозможува некои работи да ги завршеме во било кое време, побрзо и поедноставно благодарение на луѓе кои работат на тоа и со различни иновации се трудат да најдат решение за одредени барања и потреби.

Технологијата е широк концепт кој го опфаќа користењето и разбирањето на алатки и вештини од страна на некое суштество и како тоа влијае на неговата способност да ја контролира и да се приспособи на својата природна околина.

Живееме во време на дигитална ера, каде огромен број на информации не делат со само неколку клика на нашиот мобилен телефон. Дигитализацијата донесе многу позитивни и корисни промени во начинот на кој живееме. Периодов со ситуацијата поврзана со КОВИД – 19, не насочи кон еден нов начин на извршување на нашите работни задачи и активности, што дополнително се појави потребата голем дел од услугите од институциите да ги добиваме по дигитален пат. Голем дел од факултетите наидоа на унифициран и ефикасен начин на кој ќе изведуваат онлајн предавања и онлајн испити, воедно најголем дел на учебниците за студентите да се достапни онлајн.

Технологијата ни е повеќе од потребна. Светската популација секоја година се зголемува за над 80 милиони луѓе, еколошките катастрофи веќе ни се секојдневие. Само со помош на технологија можеме да произведеме доволно храна, да обезбедиме здравствена заштита, а истовремено да преземеме превентивни мерки за климатските промени. Автономни возила, роботи кои ќе произведуваат храна, паметни градови, дрoнови во земјоделието..., но сето тоа во служба на човекот.

1.1. Како до идејата за оваа апликација

Идејата за оваа тема, односно веб апликација за online резервации за планински туризам произлезе од потребата луѓето кои сакаат да поминат време во природа да можат да најдат се што е поврзано со планинарењето на едно место – во случајов во веб апликација.

Нема потреба луѓето да отворат социјални медиуми за да пронајдат некој настан, да го контактираат организаторот преку мобилен телефон, да бараат каде се наоѓа локацијата на Google Maps, рути на Strava, смештај или да проверуваат прогноза и препорачана опрема(што не значи дека сите овие работи што ќе ги најдат на различни места се точни).

Поголем дел од луѓето сакаат едноставни работи и немаат многу време да пребаруваат, поради сопствени обврски, па мислам дека ова е повеќе од потребно.

1.2. Цел на овој проект и апликацијата

Целта на секој труд е главниот и најважен дел во целокупната реализација и претставува придобивка од користењето на некој продукт или услуга.

Оваа веб апликација има за цел да опфати различен тип на корисници и да придонесе за планинскиот туризам во нашата земја бидејќи не е доволно развиен иако Македонија има навистина многу убави планни, туристички места и други знаменитости. Целта е да се подобри планинскиот туризам и да има повеќе туристи од нашата и од другите земји.

2. Развој на системот

Овој проект, односно веб апликацијата за планински туризам треба да е едноставна, брза и лесно разбирлива за корисниците.

Поради тоа развојот на оваа апликација се водеше кон разбирлив кориснички интерфејс, а воедно и да ги опфати сите потребни услуги кои би им биле од корист на корисниците.

2.1. Користени технологии, библиотеки и програмски јазици за реализација на веб апликацијата

Овој дел ги опфаќа технологиите, библиотеките и програмските јазици кои се користени за реализација на проектот и веб апликацијата. Каде се чуваат информациите, базата и како се пристапува до нив, корисничкиот интерфејс и што се е користено при дизајнот на корисничкиот интерфејс.

2.1.1. SQL



Слика 1: Лого на SQL

На слика 1 е прикажано логото на SQL

SQL [1] (Structured Query Language) е стандардизиран програмски јазик што се користи за управување со релациони бази на податоци и извршување на различни операции на податоците во нив. Првично создаден во 1970 -тите, SQL редовно се користи не само од администраторите на базата на податоци, туку и од развивачите кои пишуваат скрипти за интеграција на податоци и аналитичари на податоци кои поставуваат и одговараат аналитички прашања.

2.1.2. JavaScript



Слика 2: Лого на JavaScript

На слика 2 е прикажано логото на JavaScript.

JavaScript [2] е текст-базиран програмски јазик што се користи и од клиентска страна и од серверска. Овозможува веб-страниците да се направат интерактивни. Каде што HTML и CSS се јазици што даваат структура и стил на веб -страниците, JavaScript им дава на веб -страниците интерактивни елементи што вклучуваат употреба во апликацијата.

2.1.3. Ajax



Слика 3: Лого на Ajax

На слика 3 е прикажано логото на Ajax

Ajax [3] (Asynchronous JavaScript and XML) е збир на веб-развојни техники кои користат разни веб-технологии од клиентска страната за да креираат асинхрони веб-апликации.

Ајакс не е технологија, туку програмски концепт.

2.1.4. jQuery



Слика 4: Лого на jQuery

На слика 4 е прикажано логото на jQuery. jQuery [4] е брза, мала и богата со карактеристики JavaScript библиотека. Ги олеснува работите како прелистување и манипулација со HTML-документи, ракување со настани, анимација и Ajax. Многу е едноставен и лесен за употреба.

jQuery го смени начинот на кој милиони луѓе пишуваат JavaScript.

2.1.5. CSS & HTML



Слика 5: Лого на HTML & CSS

На слика 5 е прикажано логото на HTML и CSS. HTML [5] (Hypertext Markup Language) и CSS [6] (Cascading Style Sheets) се две од основните технологии за градење веб -страници. HTML обезбедува структура на страницата, CSS (визуелен и звучен) распоред, за различни уреди. Заедно со графика и скриптирање, HTML и CSS се основа за градење веб -страници и веб -апликации.

2.1.6. C#



Слика 6: Лого на C#

На слика 6 е прикажано логото на C#.

C# [7] е general-purpose(јазик за општа намена), multi-paradigm(мулти парадигма) програмски јазик. Се користи за развој на десктоп апликации, веб апликации и веб услуги. Се користи за креирање на апликации на Мајкрософт во големи размери. C# исто така се користи во развој на игри во Unity.

2.1.7. ASP .NET MVC



Слика 7: Лого на ASP .NET MVC

На слика 7 е прикажано логото на ASP .NET MVC.

ASP.NET MVC [8] е web application framework (рамка за веб апликации) развиена од Microsoft која ја имплементира шемата за MVC, односно model–view–controller (модел -приказ -контролер).

ASP .NET е целина од ASP.NET, ASP.NET MVC, ASP.NET Web API, и ASP.NET Web Pages.

2.1.8. API



Слика 8: Лого на API

На слика 8 е прикажано логото на API.

API [9] е кратенка за Application Programming Interface (апликациски програмски интерфејс), кој е софтверски посредник кој им овозможува на две апликации да разговараат едни со други. Секој пат кога користите апликација како Фејсбук, испраќате инстант порака или го проверувате времето на вашиот телефон, користите API.

2.1.9. Bootstrap.js & bootstrap-datetimepicker.js



Слика 9: Лого на Bootstrap

На слика 9 е прикажано логото на Bootstrap.

Bootstrap [10] е бесплатен framework со отворен код, насочена кон респонзивен, mobile-first front-end веб девелопмент.

Содржи шаблони за дизајн базирани на CSS и JavaScript за типографија, форми, копчиња, навигација и други компоненти на интерфејс.

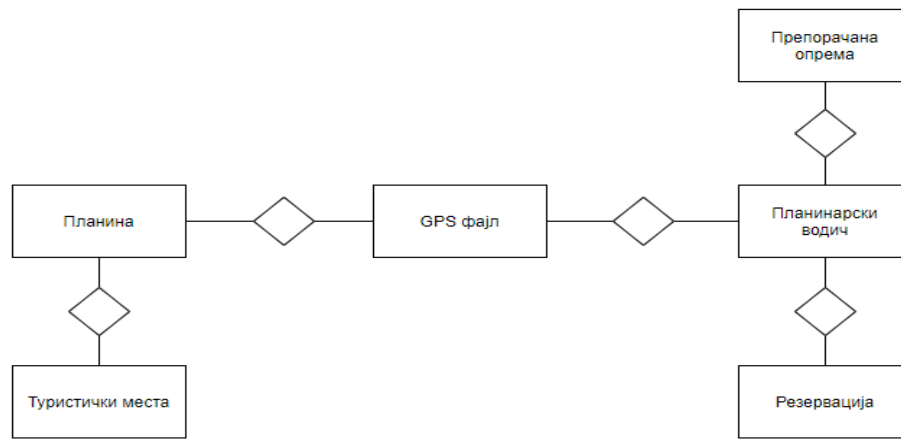
Покрај bootstrap.js, за мојата веб апликација е искористена и bootstrap-datetimepicker.js [11] библиотеката за креирање на календар со датум и време.

2.2. Етапи при развојот на веб апликацијата

Веб апликацијата за планински туризам овозможува регистрација, најава, преглед, креирање и резервирање на планински тури online.

За тоа да се овозможи користена е SQL база, распределба на податоците во различни модели(табели), релации меѓу нив, авторизација и автентикација.

За манипулација со податоците, приказ на View-ата и логиката во нив се користат контролери. ЕР дијаграмот на базата е претставен на слика 10.



Слика 10: ЕР дијаграм на базата

Веб апликацијата опфаќа три етапи:

- Регистрација, авторизација и автентикација [12]
- Листање
- Preview на селекција и регистрација

Автентикација значи дека веб апликацијата има можност за најава, и без најава корисникот нема да може да пристапи до веб апликацијата. Регистрација е кога корисникот креира ново корисничко име и лозинка со кој би можел подоцна да се најави на веб апликацијата, додека пак авторизација е кога корисникот на веб апликацијата има одредена улога и во зависност од таа улога има различни пермисии кон одредени содржини.

Интересно за оваа апликација е тоа што постои приказ на GPS запис директно на мапа. Тоа е овозможено со Google Maps API, а воедно и со помош на JavaScript.

Постои опција за приказ на GPS запис, така што доволно е корисникот само да избери фајл од File Explorer и да даде опис за записот за да биде попрегледно. Покрај тоа има опција кога еден корисник ќе прикачи GPS фајл на веб страната,

друг корисник да го симне тој фајл и да го стави во друг уред. Што навистина е корисно за двете страни.

Администраторот може да креира, да ги уредува и да ги брише записите креирани од него или од некој корисник доколку би имал пермисија за креирање на записи.

Исто така, како посебен дел има препорачана опрема која корисникот може да ја разгледа и да ја спореди со временските услови, и посебен дел за препорака на атрактивни туристички места во близина на планината која корисникот ја избрал.

Пренасочувањето од еден кон друг веб сајт е едноставно имплементирано, со користење на готови функции за Redirect.

При избор на секој нареден чекор од страна на корисникот, се памти што тој всушност избрал преку View(преносот на параметри е со @ViewBag).

Во последниот дел за резервации корисникот има можност да направи преглед на досегашната селекција од претходните страни и да избере датум и време на кои би сакал да учествува во планинската тура.

Со успешна регистрација на корисникот се креира нов запис во посебен дел 'Резервации'. Исто така се зголемува бројот на планинари за еден за соодветната тура со соодветниот планинарски водич, за секој следен корисник да знае колку планинари се пријавени на таа тура со тој планинарски водич.

3. Функционалност на системот

Како што беше претходно напишано во делот за 'Цел на веб апликацијата' крајната етапа е најважна и мора да е функционална за да им биде од корист на корисниците.

Веб апликацијата може да има повеќе делови на функционалност во зависност од улогата која управува со неа.

3.1. Регистрација

Опција корисникот да се регистрира, при тоа да креира ново корисничко име и лозинка кои подоцна може да ги користи за најава на веб апликацијата.

Тоа значи дека за користење на оваа веб апликација секој корисник мора прво да е регистриран.

3.2. Авторизација

Авторизација [12] е кога корисникот на веб апликацијата има одредена улога и во зависност од таа улога има различни пермисии кон одредени содржини.

Улоги што се доделени на корисниците на веб апликацијата за резервации на планински туризам се Administrator, Editor и User.

Секој регистриран корисник може да има само една улога. За пристап до одредена содржима што ја овозможува оваа веб апликација, мора корисникот да биде автентизиран – да е регистриран и да е најавен со точно корисничко име и лозинка.

3.3. Автентикација

Автентикација [12] значи дека веб апликацијата има можност за најава, и без најава корисникот нема да може да пристапи до веб апликацијата.

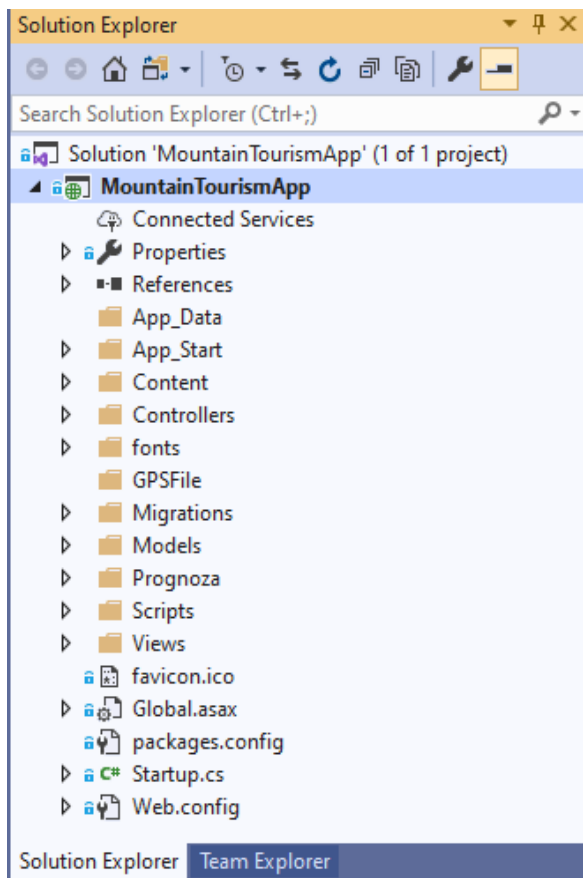
4. Имплементација на апликацијата

Во оваа секција е опишан начинот на кој ја изработив апликацијата, правилата според кои се водев и како е организирана. Прво е опишана архитектурата на апликацијата, а потоа имплементацијата на Backend и Frontend делот.

4.1. Архитектура на апликацијата

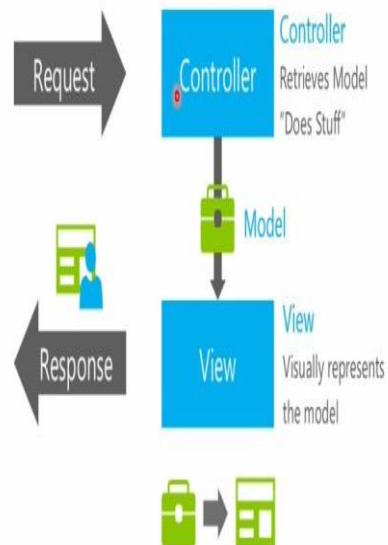
Апликацијата е ASP .NET MVC.

На слика 11 е прикажана архитектурата на ASP .Net MVC апликација.



Models, Views, and Controllers

What does MVC look like?

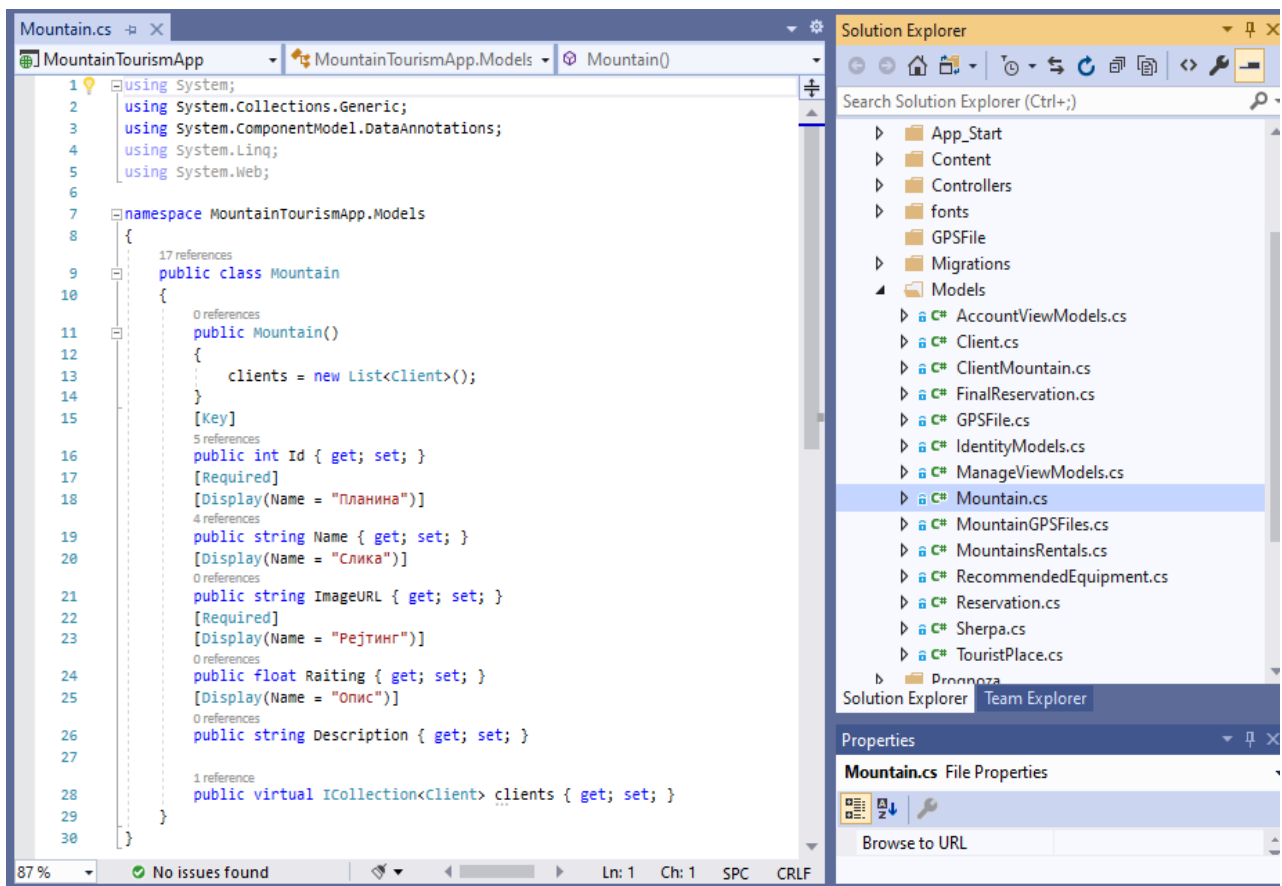


Слика 11: Архитектура на ASP .Net MVC апликација

MVC значи дека апликацијата се состои од 3 компоненти:

- Model – ја претставува формата на податоците. Може да се опише како табела, променливите кои ги има моделот се всушност атрибути на табелата доколку го видиме тоа во SQL база. Како што може да се примети на следната слика моделите се .cs фајлови.

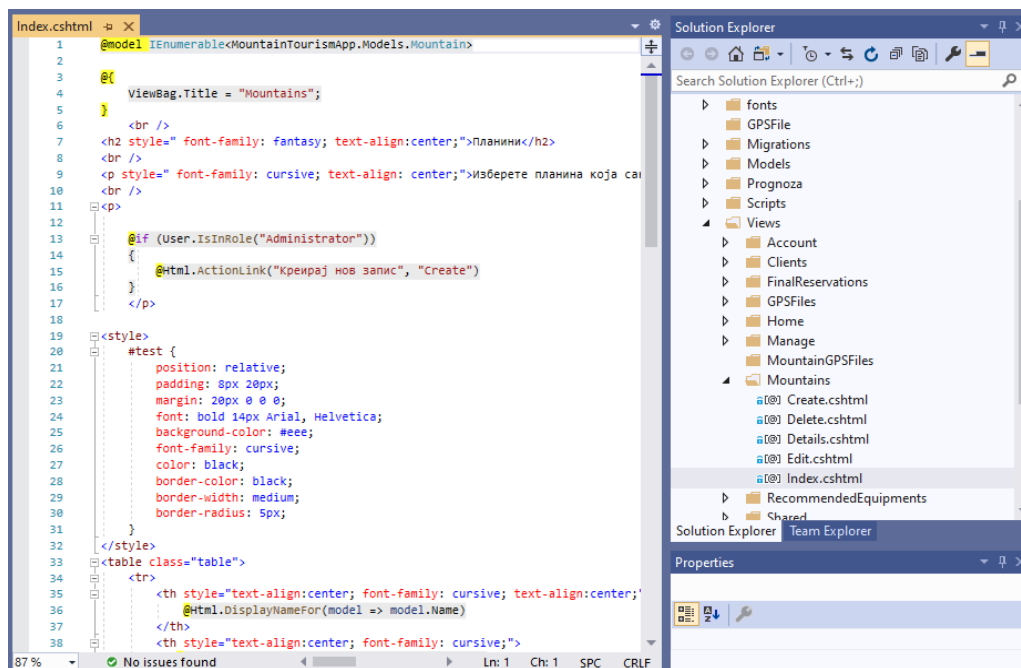
На слика 12 е прикажана структурата на моделот.



Слика 12: Структура на Model

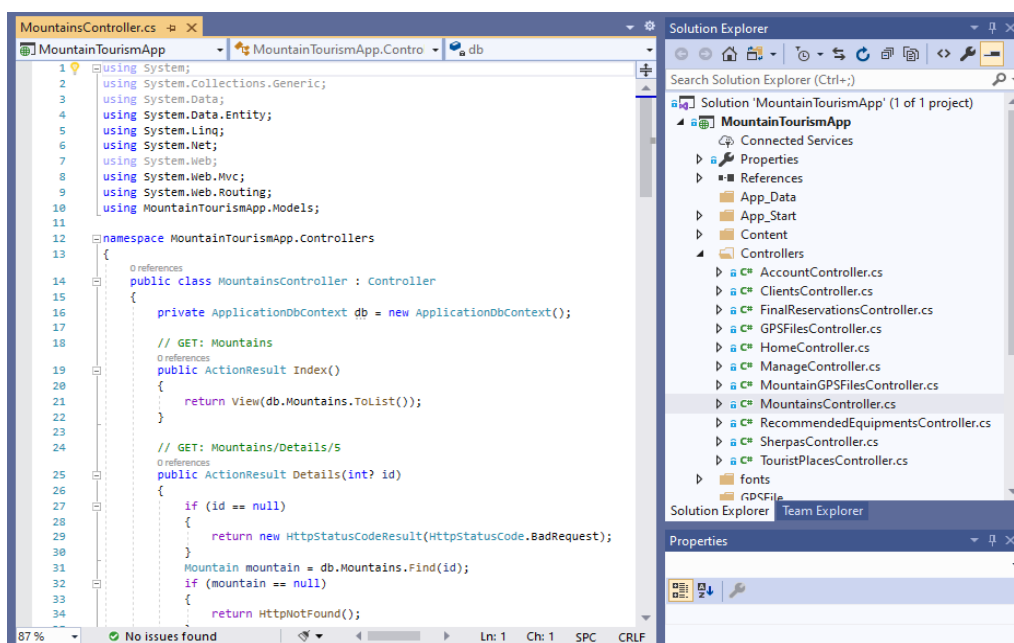
- View – во MVC претставува кориснички интерфејс. Ги прикажува податоците од моделот и овозможува тие да бидат изменети од страна на корисникот. View-ата се .cshtml фајлови. Папката со име Views содржи повеќе папки во неа, секоја папка ги содржи View-ата од истоимениот Model.

На слика 13 е прикажана структурата на View.



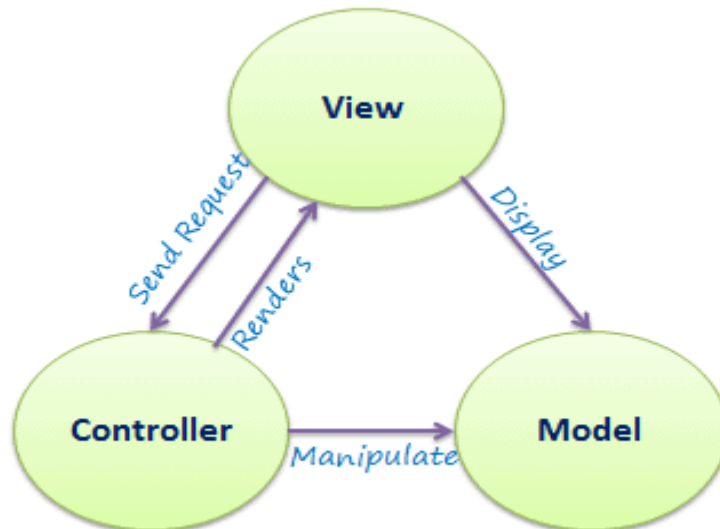
Слика 13: Структура на View

- Controller - се справува со корисничкото барање. Обично, корисникот го користи View-то и покренува HTTP барање, кое ќе го управува контролерот. Контролерот го обработува барањето и го враќа соодветниот приказ како одговор. На слика 14 е прикажана структурата на Controller.



Слика 14: Структура на Controller

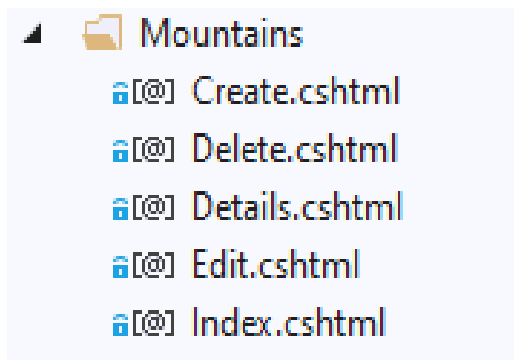
Следната слика(слика 15) ја илустрира интеракцијата помеѓу Model, View и Controller



Слика 15: MVC дијаграм

Во оваа веб апликација постојат неколку модели - Mountain, GPSFile, Sherpa, RecommendedEquipment, TouristPlace, Client и FinalReservation.

За секој од овие модели имам креирано по неколку View-а(покрај тие на сликата за некои модели постојат повеќе). На слика 16 е прикажана организацијата на View за Mountains.



Слика 16: Организација на View

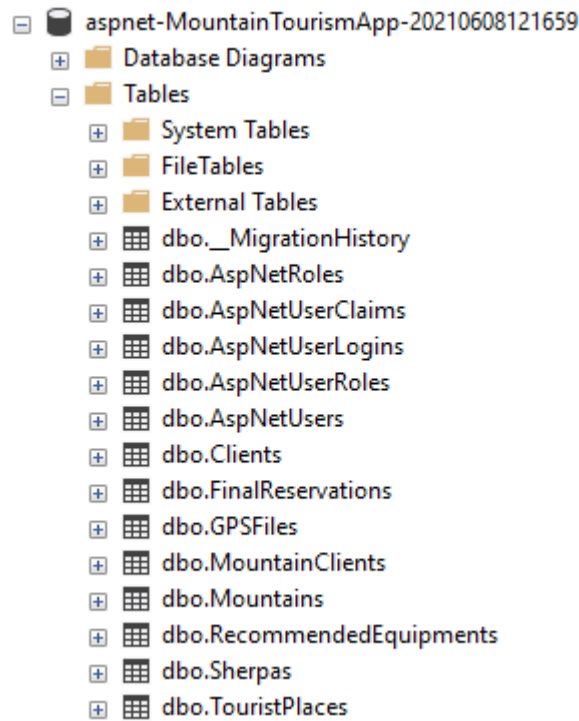
Index.cshtml ја прикажува листата од планини, Create се повикува кога сакаме да креираме нов запис за планина, Detail за повеќе детали и Edit и Delete за промена и бришење.

4.2. Backend имплементација

Во претходниот дел е опишано дека апликацијата користи MVC framework, како се поврзани и за што служат Model, View и Controller-от.

Најголем дел од логиката која припаѓа на Backend делот е имплементирана во моделите и контролерите со примена на C# програмски јазик.

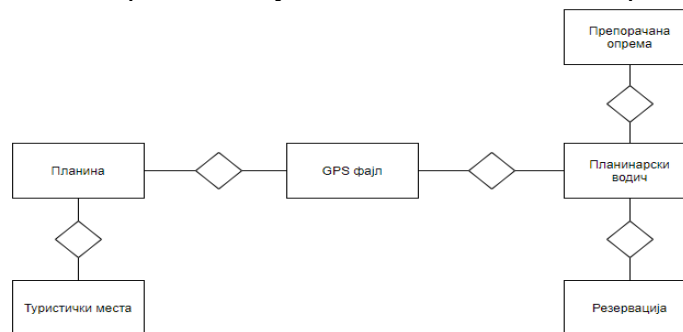
За апликацијата е користена SQL база која е поврзана со апликацијата преку connection string во Web.config фајлот од апликацијата. На слика 17 е прикажана SQL базата во SQL Management Studio.



Слика 17: SQL базата

Прво ги креирав потребните модели и им додадов атрибути.

После секоја промена на модел која сакаме да ја видиме во SQL база мора право да се додаде миграција и да се направи update на базата преку Package Manager Console. Табелите се поврзани меѓу себе со 1:N или N:M врски, како на слика 18.



Слика 18: Релации меѓу ентитетите

Секој ентитет(модел) се состои од атрибути каде што мора да има барем еден клуч како што е прикажано на слика 19. На секој модел се додава клучниот збор [Key] над атрибутот кој што сакаме да биде клуч во ентитетот, и секој атрибут има get и set функции во моделот.

За одреден атрибут во цела апликација да се прикажува со исто име на клиентска страна потребно е над него да се напише клучниот збор [Display(Name = "име")].

На следната слика има 1:N врска меѓу Mountain и Client, затоа во конструкторот на Mountain се иницијализира нова листа од клиенти(колекција).

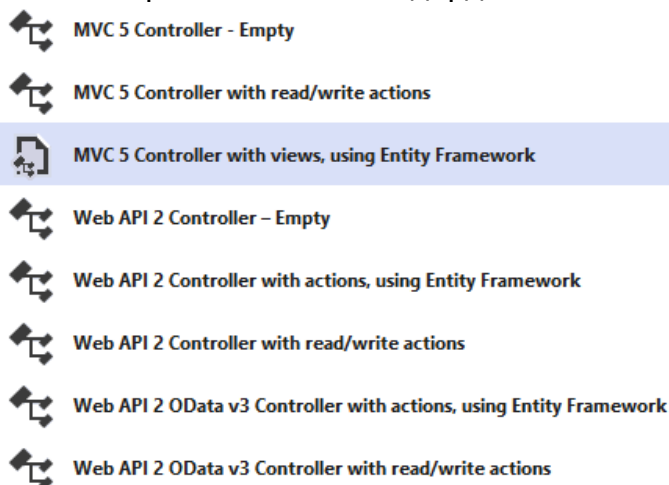
```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace MountainTourismApp.Models
{
    17 references
    public class Mountain
    {
        0 references
        public Mountain()
        {
            clients = new List<Client>();
        }
        [Key]
        5 references
        public int Id { get; set; }
        [Required]
        [Display(Name = "планина")]
        4 references
        public string Name { get; set; }
        [Display(Name = "Слика")]
        0 references
        public string ImageURL { get; set; }
        [Required]
        [Display(Name = "Рејтинг")]
        0 references
        public float Rating { get; set; }
        [Display(Name = "Опис")]
        0 references
        public string Description { get; set; }

        1 reference
        public virtual ICollection<Client> clients { get; set; }
    }
}
```

Слика 19: Атрибути на моделот

Потоа, откако ги креирав моделите и ги мапирав, со помош на Entity Framework како на слика 20, креирав контролери а со тоа и стандардни view-а за секој од моделите.



Слика 20: Начини за креирање на нов контролер

Секој контролер припаѓа на еден ист Data Context – ApplicationDbContext кој се креира при креирање на апликацијата со MVC framework како на слика 21.

На слика 21 е прикажана содржината на Application DB Context-от каде што се дефинирани DB Set-ови за секој модел што би требало да претставува ентитет во базата, која е поврзана со connection string, во случајов името на тој string е “DefaultConnection”

```
23 references
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    7 references
    public DbSet<Client> Clients { get; set; }
    11 references
    public DbSet<Mountain> Mountains { get; set; }

    18 references
    public DbSet<GPSFile> GPSFile { get; set; }
    20 references
    public DbSet<Sherpa> Sherpa { get; set; }
    7 references
    public DbSet<RecommendedEquipment> RecommendedEquipment { get; set; }

    9 references
    public ApplicationDbContext()
        : base("DefaultConnection", throwIfV1Schema: false)
    {
    }

    1 reference
    public static ApplicationDbContext Create()
    {
        return new ApplicationDbContext();
    }

    8 references
    public System.Data.Entity.DbSet<MountainTourismApp.Models.TouristPlace> TouristPlaces { get; set; }

    9 references
    public System.Data.Entity.DbSet<MountainTourismApp.Models.FinalReservation> FinalReservations { get; set; }
}
```

Слика 21: Application DB Context

Се што треба да се прикаже кај корисникот, и што треба да се одговори на барањата е работа на контролерот.

Функциите кои што ги содржи контролерот се дефинирани како GET или POST акции и се повикуваат во View со или без параметри во повикот.

На слика 22 е прикажана дефиницијата за GET и POST функциите за Create во контролерот Mountains.

На слика 23 е прикажан повик на GET функцијата ‘Create’ . На клиентска страна содржината каде што ќе се повикува оваа функција ќе биде достапна како Action link со текст ‘Креирај нов запис’. Истото ќе се прикаже доколку корисникот има улога на администратор.

На слика 23 се повикува GET методот за Create, а на слика 24 е прикажан повик на POST методот за Create.

```

// GET: Mountains/Create
0 references
public ActionResult Create()
{
    return View();
}

// POST: Mountains/Create
// To protect from overposting attacks, please enable the specific properties you want to bind to, for
// more details see https://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Create([Bind(Include = "Id,Name,ImageUrl,Raiting,Description")] Mountain mountain)
{
    if (ModelState.IsValid)
    {
        db.Mountains.Add(mountain);
        db.SaveChanges();
        //return RedirectToAction("Index");
    }

    return View(mountain);
}

```

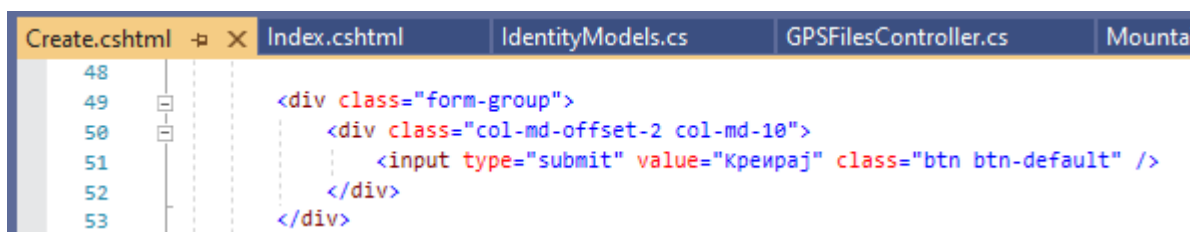
Слика 22: Дефиниција на GET и POST функциите за Create во контролеот Mountains

```

@if (User.IsInRole("Administrator"))
{
    @Html.ActionLink("Креирај нов запис", "Create")
}

```

Слика 23: Повик на GET методот во View



```

48
49
50
51
52
53
<div class="form-group">
  <div class="col-md-offset-2 col-md-10">
    <input type="submit" value="Креирај" class="btn btn-default" />
  </div>
</div>

```

Слика 24: Повик на POST методот во View

За креирање/прикачување на GPS фајл креирав нова функција која е прикажана на слика 25.

Пред дефиницијата на функцијата се додавава [HttpPost] за да може да се испрати текст до веб-сервер за обработка.

Во одредени случаи, POST се користи за добивање податоци на сличен начин како и командата HTTP GET.

```

[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Create([Bind(Include = "Id,mountainId,title,description,distance,positiveD,negativeD,uploadFile")] GPSFile gPSFile)
{
    string fileName = System.IO.Path.GetFileNameWithoutExtension(gPSFile.uploadFile.FileName);
    string extension = System.IO.Path.GetExtension(gPSFile.uploadFile.FileName);
    fileName = fileName + DateTime.Now.ToString("yymmssfff") + extension;
    gPSFile.filePath = "~/GPSFile/" + fileName;
    gPSFile.title = fileName;
    fileName = Path.Combine(Server.MapPath("~/GPSFile/"), fileName);
    gPSFile.uploadFile.SaveAs(fileName);

    db.GPSFile.Add(gPSFile);
    db.SaveChanges();

    ModelState.Clear();
    return View();
}

```

Слика 25: Функција за креирање на нов GPS запис

[ValidateAntiForgeryToken] атрибутот проверува дали барањето го содржи безбедносниот токен или не, и ако не го содржи токенот, тогаш ја генерира следната грешка:

“A required anti-forgery token was not supplied or was invalid”.

[Bind] атрибутот може да се примени на класа или како параметар на метод.

Преку овај клучен збор се пренесува барањето за да се пополнат својствата на наведените имиња од страна на model binder.

FileName е променлива од името на фајлот кој што е прикачен, вредноста се дефинира со веќе постоечка функција System.IO.Path.GetFileNameWithoutExtension, каде што како параметар се зема model.file.nameOfFile, тоа може да се пристапи на овој начин бидејќи во моделот претходно фајлот е дефиниран од тип HttpPostedFileBase (слика 26).

```

[NotMapped]
3 references
public HttpPostedFileBase uploadFile { get; set; }

```

Слика 26: Дефиниција на GPS фајлот во моделот

GPS фајлот се зачувува во база и во проектот локално во папката ‘GPSFile’, а името на фајлот е со наставка од датумот кога е прикачен и екстензија од фајлот, тоа е овозможено со употреба на готови функции System.IO.Path.GetExtension и DateTime.Now.

За зачувување во база потребно е прво да се креира инстанца (слика 27) до базата, и потоа се пристапува како на слика 28, прво се додава записот во база, а потоа се зачувуваат промените.

```

public class GPSFilesController : Controller
{
    private ApplicationDbContext db = new ApplicationDbContext();
}

```

Слика 27: Инстанца до базата

```
db.GPSFile.Add(gPSFile);
db.SaveChanges();
```

Слика 28: Манипулација со база

Освен можноста за прикачување на нов фајл, овозможено е превземање на фајлот од страна на друг корисник како што е прикажано на слика 29.

Патеката на фајлот кој е избран за превземање се пристапува преку сервер со употреба на `Server.MapPath()`.

Потоа на `Response` се прави `Clear`, `Flush` за испраќање на резултатот веднаш, и `TransmitFile` за испраќање на фајлот директно преку HTTP response.

```
[HttpGet]
0 references
public ActionResult View(int id)
{
    GPSFile GPSFileModel = new GPSFile();
    GPSFileModel = db.GPSFile.Where(x => x.Id == id).FirstOrDefault();
    this.downloadFile(GPSFileModel.title);
    return RedirectToAction("Index", "Mountain");
}

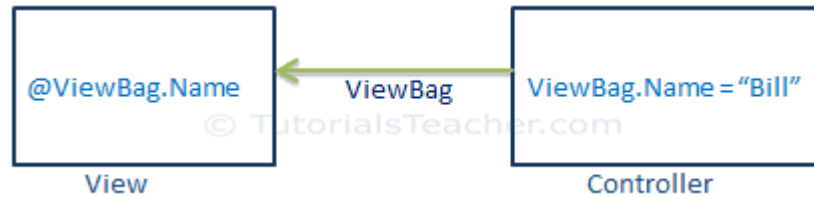
1 reference
public void downloadFile(String file)
{
    try
    {
        String fileName = file;
        String filePath = Server.MapPath("~/GPSFile/" + fileName);

        Response.Clear();
        Response.ClearHeaders();
        Response.ClearContent();
        Response.AddHeader("Content-Disposition", "attachment; fileName=" + fileName);
        Response.Flush();
        // HttpContext.ApplicationInstance.CompleteRequest();
        Response.TransmitFile(filePath);
        Response.End();

        ViewBag.message = "";
    }
    catch (Exception ex)
    {
        ViewBag.message = ex.Message.ToString();
    }
}
```

Слика 29: Превземање на GPS фајл

Покрај релациите на ентитетите(моделите) во базата од едно во друго View може да се испраќаат параметри преку ViewBag како што е прикажано на слика 30. Параметрите во ViewBag се креираат и се иницијализираат во некоја од акциите(функциите) во контролерот и се користат за приказ на вредностите во View. ViewBag во ASP.NET MVC се користи за пренос на привремени податоци (кои не се вклучени во моделот) од Controller до View.(слика 31 и 32)



Слика 30: ViewBag пренос

```

public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Sherpa sherpa = db.Sherpa.Find(id);
    if (sherpa == null)
    {
        return HttpNotFound();
    }

    ViewBag.parm = id;

    var userEmail = this.User.Identity.Name;
    ViewBag.userEmail = userEmail;

    var finalSherpa = db.Sherpa.Where(b => b.Id == id).ToList().FirstOrDefault().name;
    var gpsFileSherpa = db.Sherpa.Where(b => b.Id == id).ToList().FirstOrDefault().GPSFileId;
    ViewBag.finalSherpa = finalSherpa;

    var gpsFile = db.GPSFile.Where(b => b.Id == gpsFileSherpa).ToList().FirstOrDefault().description;
    var mountainGPS = db.GPSFile.Where(b => b.Id == gpsFileSherpa).ToList().FirstOrDefault().mountainId;
    ViewBag.gpsFile = gpsFile;

    var mountain = db.Mountains.Where(b => b.Id == mountainGPS).ToList().FirstOrDefault().Name;
    ViewBag.mountain = mountain;
    return View(sherpa);
}

```

Слика 31: Иницијализација на ViewBag во контролер

```

73 @Html.HiddenFor(model => model.Id)
74
75
76
77 <div class="card">
78 
80 <h4><b>Изберете датум и време:</b></h4>
81 <h6>
82 Почитуван кориснику <i>@ViewBag userEmail</i> изберете датум и време за старт доклку сакате резервацијат
83 да биде успешна и притиснете на копчето <i>Резервирај</i>.
84 </h6>
85 <br />
86
87
88 <h6>
89 <b>@Html.LabelFor(model => model.dateTime)</b>
90
91 @Html.EditorFor(model => model.dateTime, new { htmlAttributes = new { @type = "datetime-local" } })
92 </h6>
93
94
95
96 <h6><b>Планина:</b>@ViewBag.mountain</h6>
97 <h6><b>Рута:</b>@ViewBag.gpsFile</h6>
98 <h6><b>Планинарски водич:</b>@ViewBag.finalSherpa</h6>
99 <br />
100 <br />
101 <div class="form-group">
102 <div >
103 <input id="test" type="submit" value="Резервирај" class="btn btn-default" />
104 </div>
105 </div>
106 </div>
107 </div>

```

Слика 32: Употреба на ViewBag во View

Листањето од еден во друг модел е имплементирано преку филтрирање со id бидејќи постои релација меѓу табелите, така што во следниот пример(слика 33, 34 и 35) се филтрираат сите планинарски водичи за даден GPS запис.

```

<div class="btn" id="test">
    @Html.ActionLink("Планинарски водич", "ListSherpas", new { id = item.Id })
</div>

```

Слика 33: Повик на методот ListSherpas во View 'Index' од GPSFiles

```

public ActionResult ListSherpas(int? id)
{
    return RedirectToAction("Index", new RouteValueDictionary(new { Controller = "Sherpas", Action = "Index", Id = id }));
}

```

Слика 34: Дефиниција на методот ListSherpas во GPSFilesController

```
// GET: Sherpas
0 references
public ActionResult Index(int? id)
{
    ViewBag.parm = id;

    var mountainGPS = db.GPSFile.Where(b => b.Id == id).ToList().FirstOrDefault().mountainId;
    ViewBag.mountain = mountainGPS;

    return View(db.Sherpa.Where(b => b.GPSFileId == id).ToList());
}
```

Слика 35: Дефиниција на методот Index со параметар id во SherpasController

Во однос на делот за авторизација, при регистрација на веб апликацијата се креира запис во SQL базата во AspNetUsers табелата(слика 36), во оваа табела се наоѓаат сите регистрирани корисници.

Id	Email	EmailConfirmed	PasswordHash	SecurityStamp	PhoneNumber	PhoneNumber...	TwoFactorEna...	LockoutEndDa...	LockoutEnabled	Acces
1d-e571bd00c864	user@test.com	False	AliNxSDMMOH...	8e4f3031-20ce...	NULL	False	False	NULL	True	0
968e711d-0fb6-...	elena@test.com	False	AFOnvyFpM44...	d44b111d-bea3...	NULL	False	False	NULL	True	0
b0fca062-4fb0-...	editor@test.com	False	AJT2kHykORiv...	143458ca-9eb8...	NULL	False	False	NULL	True	0
ccc7ba59-2a0f-...	admin@admin...	False	AP/BQQS5Moj...	0e706239-56b5...	NULL	False	False	NULL	True	0
d68b5565-98a7...	elena@admin.c...	False	APXb8f51CHJ5...	38de2c9d-1e33...	NULL	False	False	NULL	True	0
fdc8de6d-ad5c...	elena@user.com	False	ABtDvKKQLxGi...	bc294d1d-f0e4...	NULL	False	False	NULL	True	0

Слика 36: AspNetUsers табела во SQL база

Креирав 3 улоги во табелата AspNetRoles(слика 37) со кои може да располагаат корисниците.

Id	Name
1	Administrator
2	Editor
3	User

Слика 37: AspNetRoles табела во SQL база

Потоа дефинирав кој корисник која улога ја има во табелата AspNetUserRoles.(слика 38)

Userid	RoleId
ccc7ba59-2a0f-...	1
d68b5565-98a7...	1
b0fca062-4fb0-...	2
69a55987-9b0c-...	3
fdc8de6d-ad5c...	3

Слика 38: AspNetUserRoles табела во SQL база

4.3. Frontend имплементација

Frontend е се она со кое што корисникот има интеракција.

Од гледна точка на корисникот, Frontend е синоним за корисничкиот интерфејс.

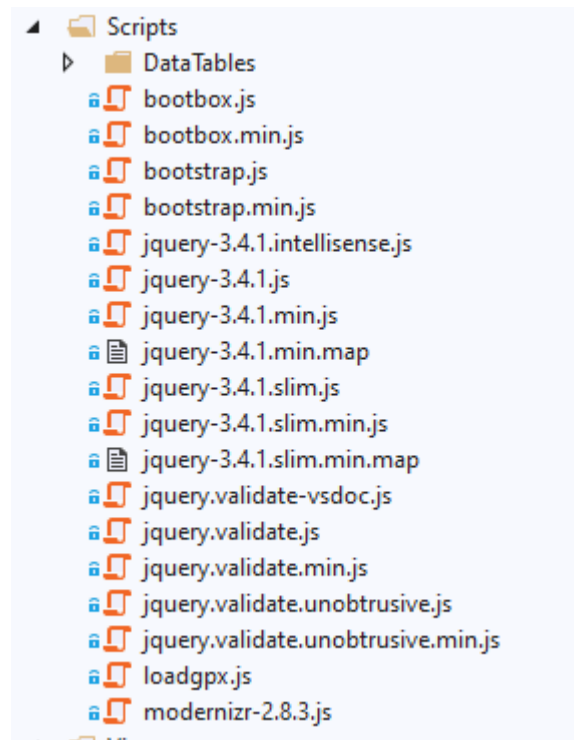
Од гледна точка на девелоперот, Frontend е дизајнот на интерфејсот и програмирањето со кое се овозможува интерфејсот да функционира.

Во рамки на овој дипломски труд користев JavaScript, Ajax, jQuery, HTML & CSS и API за делот за frontend.

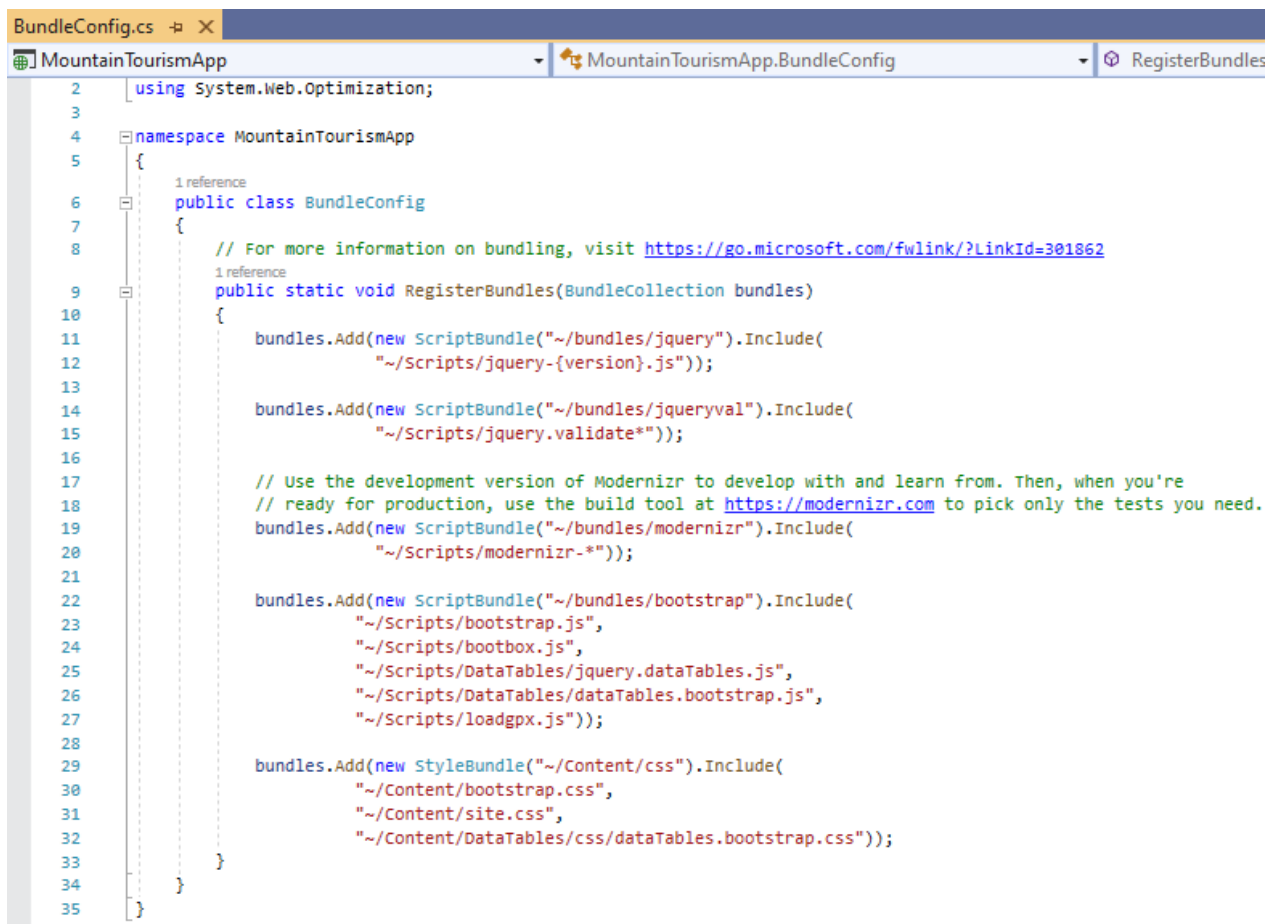
Со користење на ASP .NET MVC framework и креирање на 'MVC 5 Controller with views, using Entity Framework' се креираат и стандардни View-а за моделите каде што е претставен изгледот на корисничкиот интерфејс на веб апликацијата.

Покрај тој код, што јас дополнително го изменив, имплементирав и нови функции со помош на JavaScript.

Додадов некои библиотеки(слика 39) кои ги инсталирав преку Package Manager Console, и потоа ги вклучив преку BundleConfig.cs. (слика 40)



Слика 39: Скрипти во веб апликацијата



Слика 40: BundleConfig.cs

Библиотеките loadgpx.js и jquery.js га користев за приказ на GPS рутите на мапа.

За приказ на мапата користев Google Maps API.

Google Map's API се користи за креирање на сопствена мапа, мапа што може да се пребарува, да се манипулира и на која што можат да се применат функции, да се исцртаат рути, да се означуваат локации и слично.

Функциите со JavaScript кои што ги креирав како и CSS-от за апликацијата се дефинирани во <script> тагови.

Креирани се class и id атрибути за кои е дефиниран дизајнот со CSS во <style> тагови, директно е повикан HTML тагот во style таговите или во самиот HTML таг имаме посебен <style> тагови.

\$ajax се користи за приказ на GPS рутата на мапа, прикажано е на слика 41.

```

@Scripts.Render("~/bundles/jquery")
<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDIVDhsp43HtlgD0zCWLQg_XRYGRnf_IVU"></script>

<script type="text/javascript">

    function loadGPXFileIntoGoogleMap(map, filename) {
        $.ajax({
            url: filename,
            dataType: "xml",
            success: function (data) {
                var parser = new GPXParser(data, map);
                parser.setTrackColour("#ff0000");
                parser.setTrackWidth(5);
                parser.setMinTrackPointDelta(0.001);
                parser.centerAndZoom(data);
                parser.addTrackpointsToMap();
                parser.addWaypointsToMap();
            }
        });
    }

    $(document).ready(function () {
        var map = new google.maps.Map(document.getElementById("attribute_<%= @item.Id%>"), {
            center: { lat: 41.99646, lng: 21.43141 },
            zoom: 10
        });

        loadGPXFileIntoGoogleMap(map, "https://localhost:44386/GPSFile/" + "@item.title");
    });

</script>

```

Слика 41: Употреба на loadgpx.js, jquery.js, Google Maps API и Ajax

Ајах е збир на веб-развојни техники кои користат разни веб-технологии од клиентска страната за да креираат асинхрони веб-апликации. не е технологија, туку програмски концепт. На слика 42 е прикажана употреба на CSS во <style> тагови.

```

<style>
    #test {
        position: relative;
        padding: 8px 20px;
        margin: 20px 0 0 0;
        font: bold 14px Arial, Helvetica;
        background-color: #eee;
        font-family: cursive;
        color: black;
        border-color: black;
        border-width: medium;
        border-radius: 5px;
    }
</style>

```

Слика 42: CSS

```
#pricing-table h3 span {
    display: block;
    font: bold 25px/100px Georgia, Serif;
    color: #777;
    background: #fff;
    border: 5px solid #fff;
    height: 100px;
    width: 100px;
    margin: 10px auto -65px;
    -moz-border-radius: 100px;
    -webkit-border-radius: 100px;
    border-radius: 100px;
    -moz-box-shadow: 0 5px 20px #ddd inset, 0 3px 0 #999 inset;
    -webkit-box-shadow: 0 5px 20px #ddd inset, 0 3px 0 #999 inset;
    box-shadow: 0 5px 20px #ddd inset, 0 3px 0 #999 inset;
}
```

Слика 43: CSS за повеќе HTML елементи

Во примерот на слика 43 CSS атрибутите се користени за повеќе HTML елементи, односно се користат кај HTML елементите што имаат атрибут `id = "pricing-table"`, `<h3>` тагови и `` тагови.

Последниот дел за резервација прикажан на слика 44 е претставен со картичка каде што можат да се прегледаат избраните ставки, да се избере датум и да се заврше со резервацијата.

```
<div class="card">
  
  <div class="container">
    <h4><b>Изберете датум и време:</b></h4>
    <h6>
      Почитуван кориснику <i>@ViewBag userEmail</i> изберете датум и време за старт доклку сакате резервацијата<br />
      да биде успешна и притиснете на копчето <i>Резервирај</i>.
    </h6>
    <br />
    <h6>
      <b>Датум и време:</b>
    </h6>

    <div class='input-group date col-md-3'>
      @Html.EditorFor(model => model.dateTime, new { htmlAttributes = new { @class = "form-control", @id = "datetimepicker1" } })
      <span class="input-group-addon">
        <span class="glyphicon glyphicon-calendar"></span>
      </span>
    </div>

    <br />

    <h6><b>Планина:</b> @ViewBag.mountain</h6>
    <h6><b>Рута:</b> @ViewBag.gpsFile</h6>
    <h6><b>Планинарски водич:</b> @ViewBag.finalSherpa</h6>
    <br />
    <br />
    <div class="form-group">
      <div>
        <input id="test" type="submit" value="Резервирај" class="btn btn-default" />
      </div>
    </div>
  </div>
</div>
```

Слика 44: HTML картичка за резервација

Датумот и времето кои се користат при резервација за планинската тура се креирани со Bootstrap Date Time Picker, односно со bootstrap-datetimepicker.js библиотека. За да се искористи bootstrap-datetimepicker.js библиотеката прво треба да се додадат референци кон jquery.js, bootstrap.js, moment.js и jquery-ui.js како на слика 45. На слика 46 можат да се видат како се креирани слотови за датум и време.

```
1 reference
public static void RegisterBundles(BundleCollection bundles)
{
    bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
        "~/Scripts/jquery-3.4.1.js"));

    bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
        "~/Scripts/jquery.validate*"));

    // Use the development version of Modernizr to develop with and learn from. Then, when you're
    // ready for production, use the build tool at https://modernizr.com to pick only the tests you need.
    bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
        "~/Scripts/modernizr-*"));

    bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
        "~/Scripts/bootstrap.js",
        "~/Scripts/bootbox.js",
        "~/Scripts/DataTables/jquery.dataTables.js",
        "~/Scripts/DataTables/dataTables.bootstrap.js",
        "~/Scripts/loadgpx.js"));

    bundles.Add(new ScriptBundle("~/bundles/moment").Include(
        "~/Scripts/moment-with-locales.min.js",
        "~/Scripts/moment.min.js"));

    bundles.Add(new ScriptBundle("~/bundles/datetimepicker").Include(
        "~/Scripts/bootstrap-datetimepicker.js"));

    bundles.Add(new ScriptBundle("~/bundles/jqueryui").Include(
        "~/Scripts/jquery-ui-1.12.1.js"));

    bundles.Add(new StyleBundle("~/Content/css").Include(
        "~/Content/bootstrap.css",
        "~/Content/bootstrap-datetimepicker.css",
        "~/Content/site.css"));
}
```

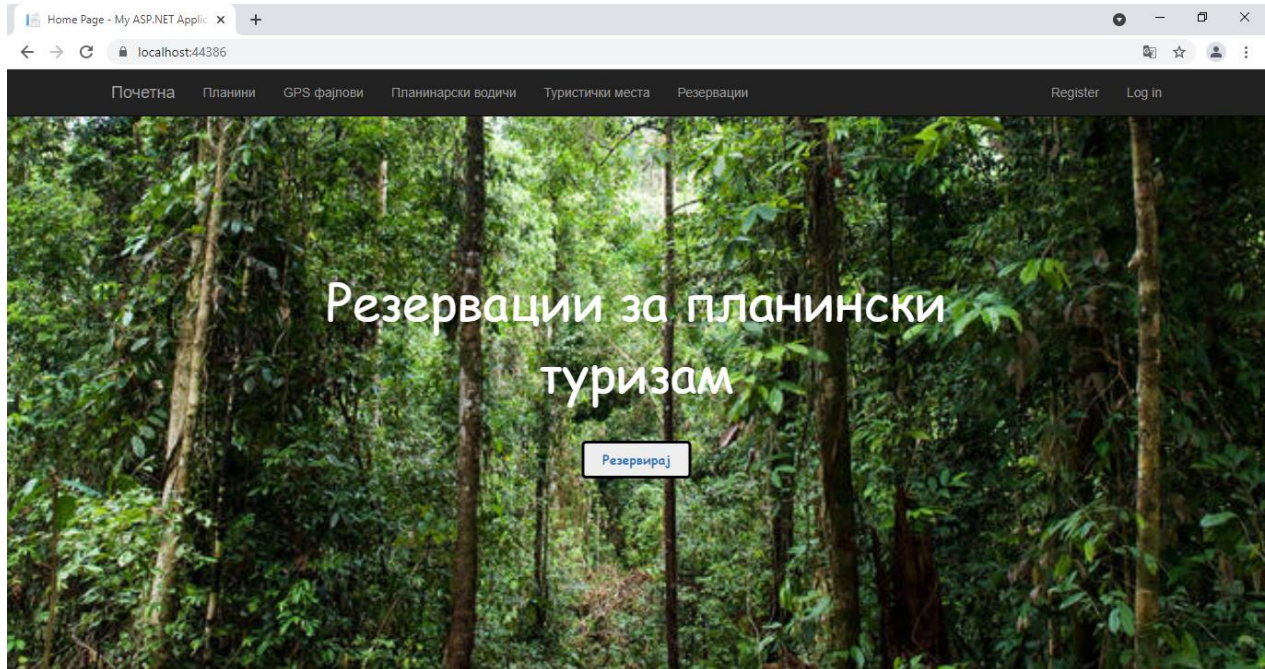
Слика 45: Вклучување на библиотеките во BundleConfig.cs

```
@section Scripts {
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @Scripts.Render("~/bundles/moment")
    @Scripts.Render("~/bundles/jqueryui")
    @Scripts.Render("~/bundles/datetimepicker");

    <script>
        $(document).ready(function () {
            $('#datetimepicker1').datetimepicker({
                enabledHours: [6, 11],
                daysOfWeekDisabled: [1, 2, 3, 4, 5],
                format: 'MM/DD/YYYY hh:00:00 A',
                minDate: new Date()
            });
        });
    </script>
}
```

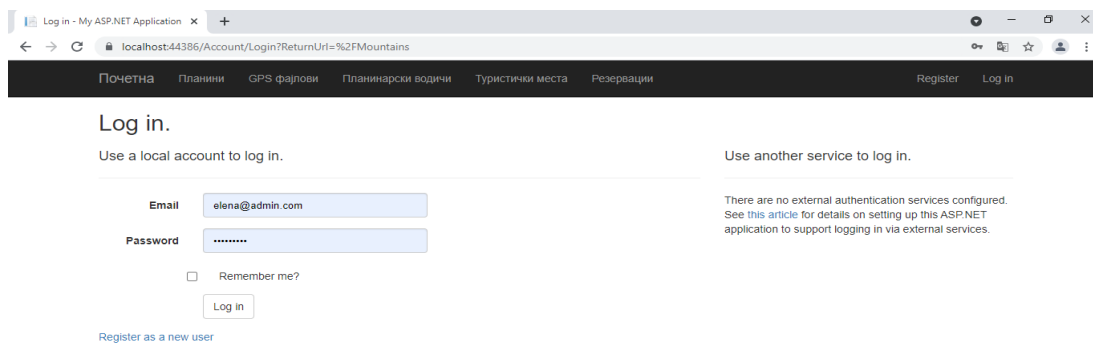
Слика 46: Употреба на Date Time Picker и bootstrap-datetimepicker.js

5. Приказ на апликацијата



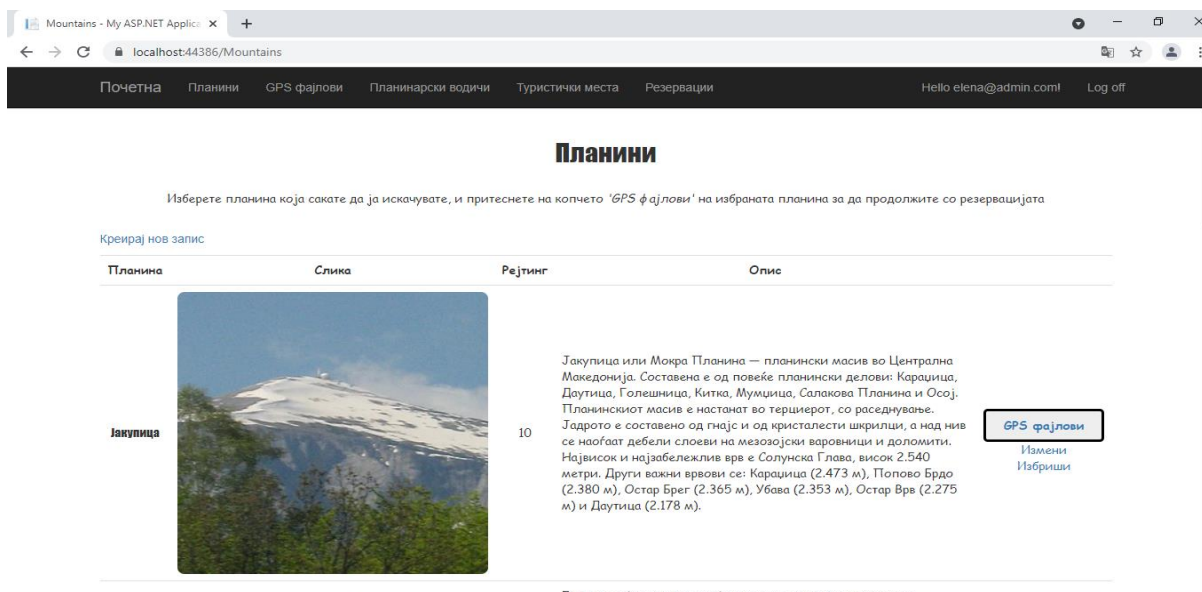
Слика 47: Почетна страна на веб апликацијата

На слка 47 е прикажана почетната страна на апликацијата. Секоја акција што ќе ја изведеме следно ќе побара најава на веб апликацијата.



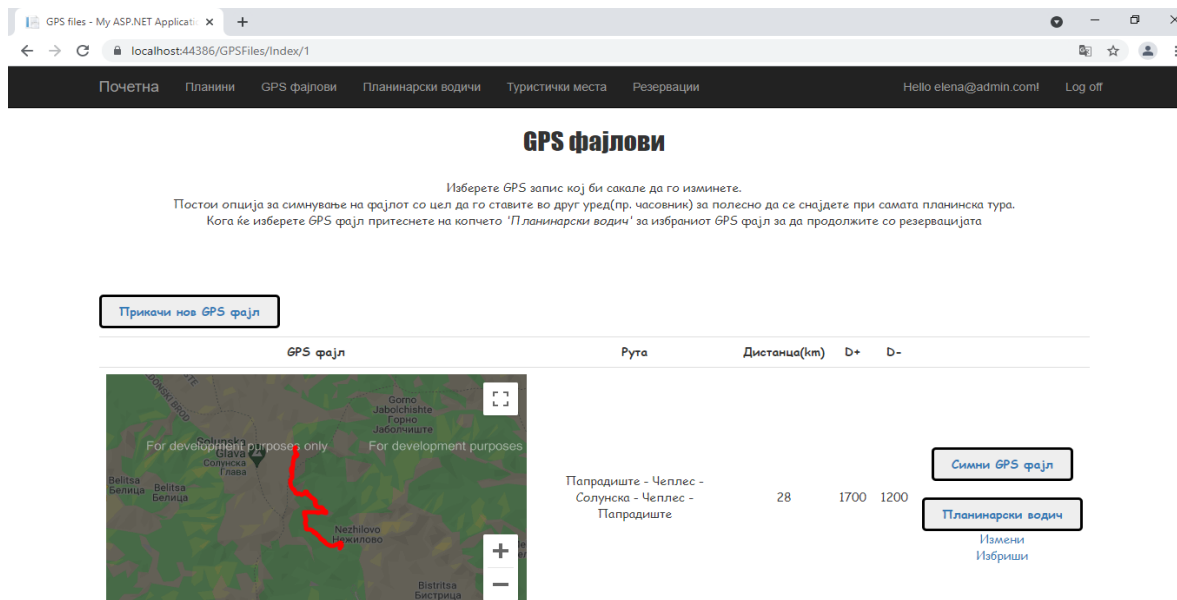
Слика 48: Најава на веб апликацијата

На слка 48 е прикажана страната за најава на веб апликацијата.



Слика 49: Листа од планини

На слка 49 е прикажана листата од планини на апликацијата. Која може да се отвори на првиот таб во хоризонталното мени со претходна најава на веб апликацијата.



Слика 50: Листа од GPS записи

На слка 50 е прикажана листата од GPS записи на апликацијата. Секој корисник може да додава нови GPS рути, да симнува GPS фајлови и да избира планинарски водичи.

Креирање на нов GPS фајл

Внесете ги потребните податоци и прикачете го фајлот од GPS записот, после тоа притиснете на копчето 'Зачувај'

Наслов

Рута

Дистанца(km)

D+

D-

Прикачи фајл No file chosen

Слика 51: Креирање на нов GPS фајл

На слка 51 е прикажано креирање на нов GPS фајл. Постои опција да се зачуваат внесените вредности или да се вратиме назад кон листата од GPS фајлови.

Планинарски водич

Изберете планинарски водич со кој би сакале да ја резервираше вашата планинска тура. Следните планинарски водичи се регистрирани за планината и рутата што ги избравте на претходните страни. Кога ќе избарете планинарски водич притиснете на копчето 'Резервирај' за да избарете датум и време за турата и каде ќе имате преглед на досегашните избрани ставки. Исто така на секој запис од планинарски водич постои 'Туристички места' каде што можете да видите кои туристички места се препорачани да ги посетите (во близина на рутата која имате план да ја изминете). Во горниот десен агол постои копче 'Препорачана опрема' каде што може да се провери за која температура која опрема е препорачана да ја понесете со себе, потоа следат три пренасочувања: правилник, прогноза и смештај.

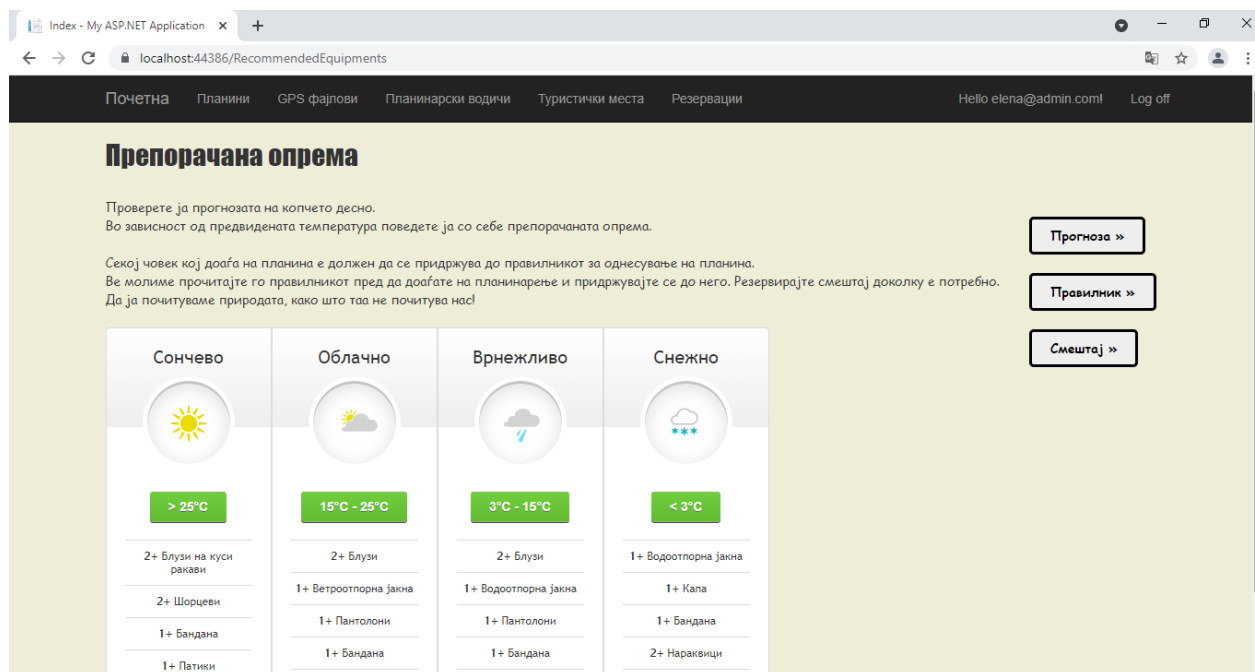
Додај нов планинарски водич

Слика	Име	Опис	Година искуство	Лиценца	Планинарски клуб	Број на планинари
	Петар Илиевски	Планинарски водич со 10 години искуство, има искочено повеќе врвови од кои и Солунска глава, Голем Короб, Титов врв и Пелистер	10	<input checked="" type="checkbox"/>	Македон	2

[Избриши](#)

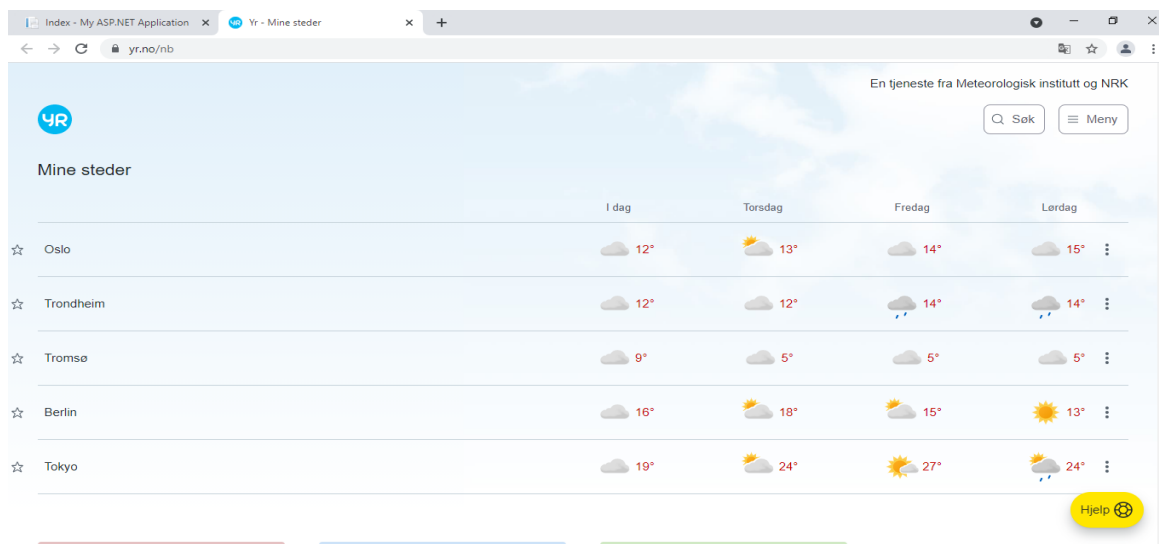
Слика 52: Листа од планинарски водичи

На слка 52 е прикажана листата од планинарски водичи. Оваа страна не пренасочува кон препорачана опрема, смештај и резервација на планинска тура.



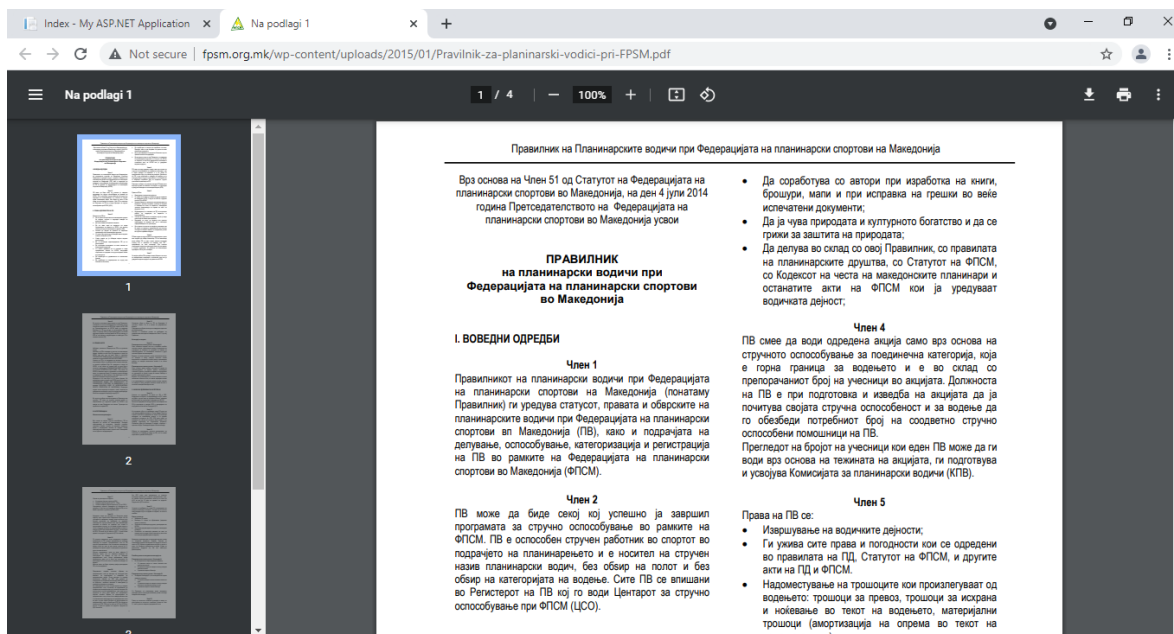
Слика 53: Препорачана опрема во зависност од временски услови

На слка 53 е прикажана препорачана опрема во зависност од временските услови. Постојат копчиња кои пренасочуваат кон прогноза, правилник и смештај.



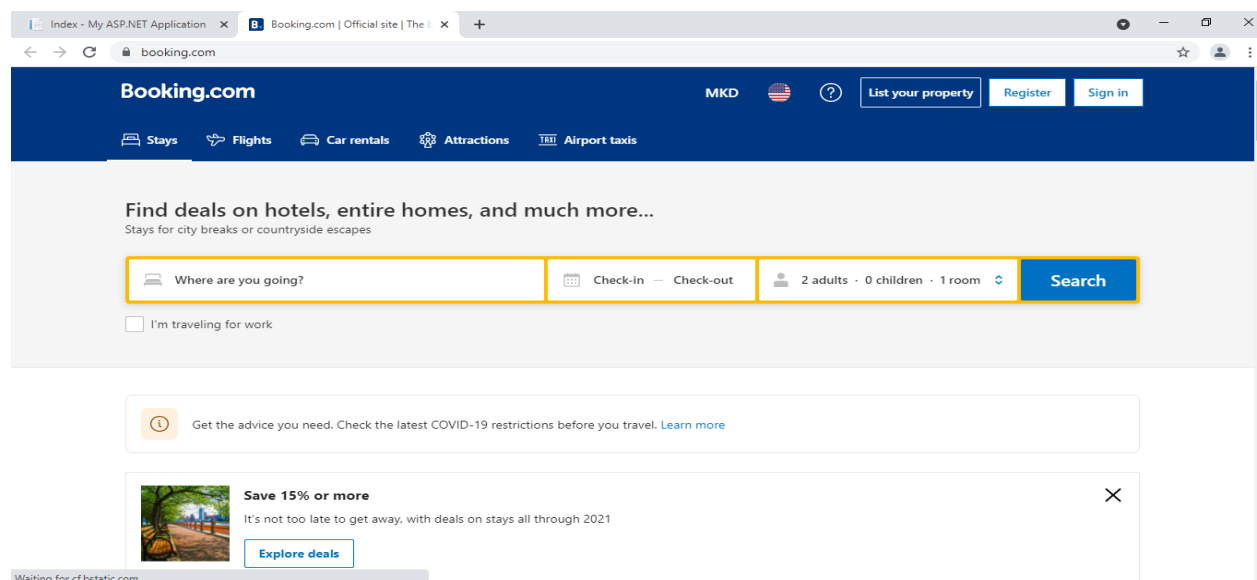
Слика 54: Временска прогноза – yr.no

На слка 54 е прикажан сајтот на yr.no/nb [13]. Се повикува кога ќе се притисне копчето 'Прогноза'.



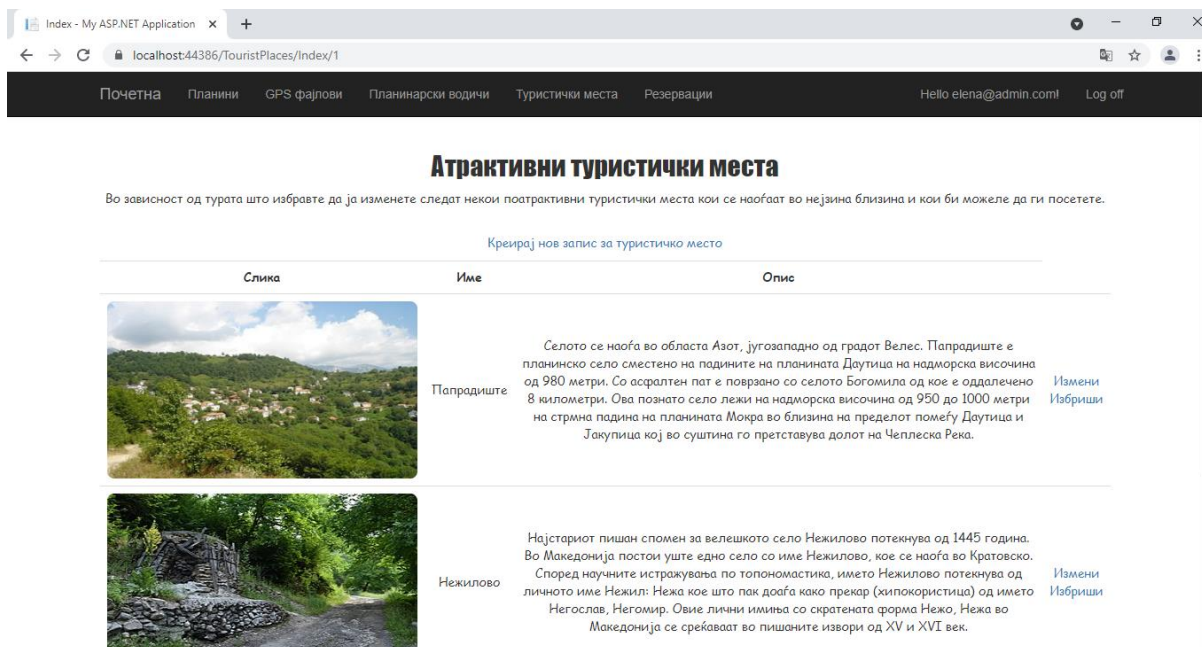
Слика 55: Правилник за планинарење – fpsm.org.mk

На слика 55 е прикажан правилник за планинарење – fpsm.org.mk [14]. Се повикува кога ќе се притисне копчето 'Правилник'.



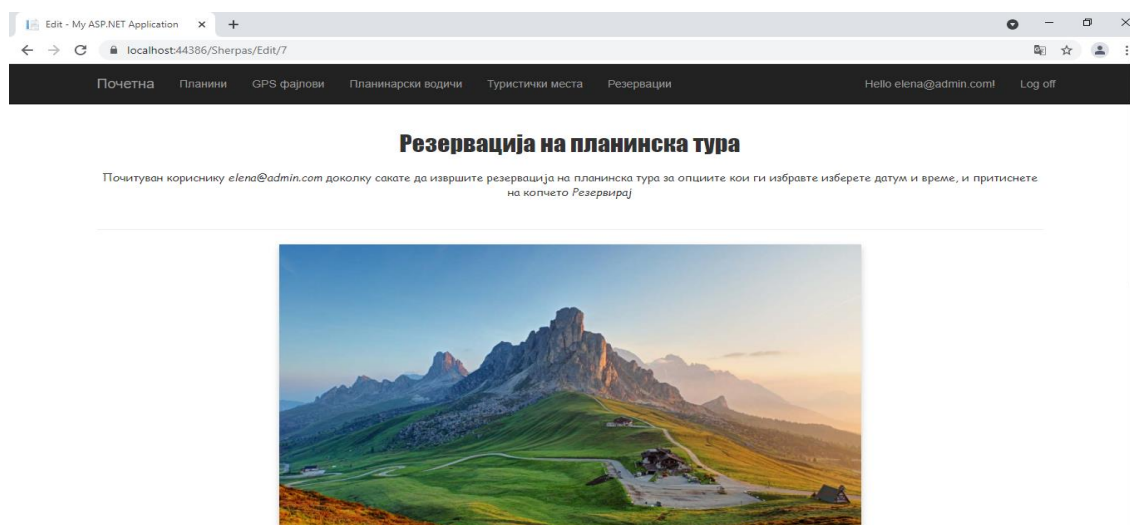
Слика 56: Сместај – booking.com

На слика 56 е прикажан сајтот за сместување booking.com [15]. Се повикува кога ќе се притисне копчето 'Сместај'.



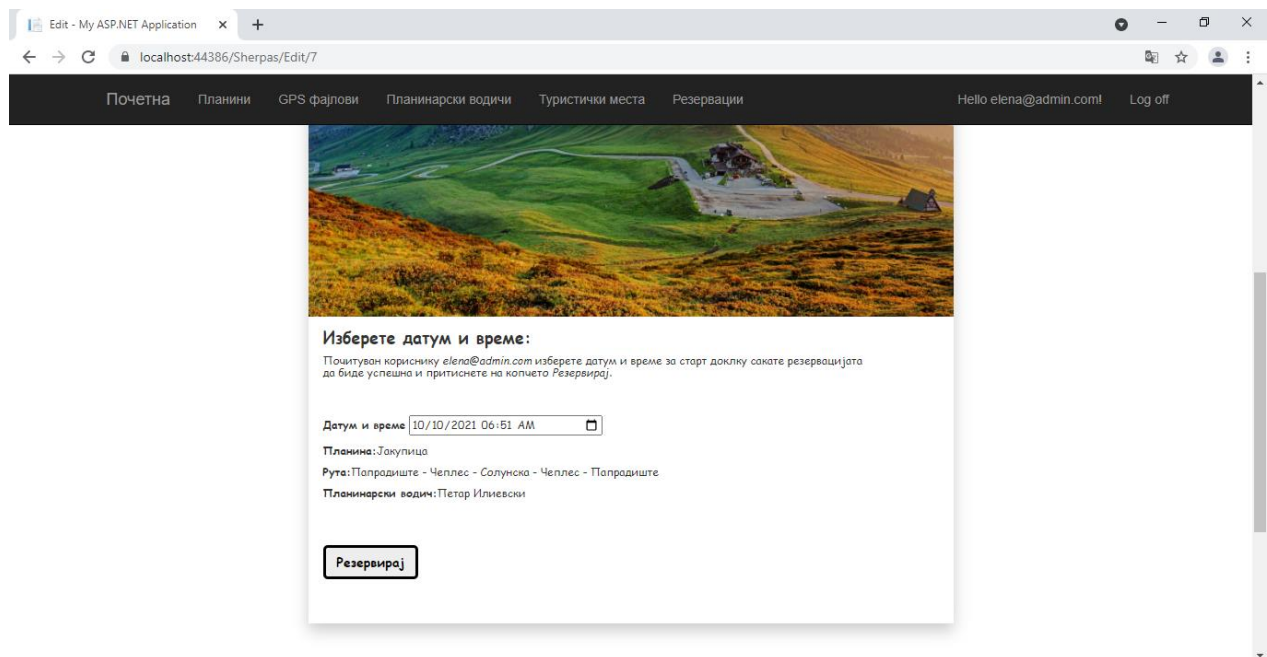
Слика 57: Атрактивни туристички места во близина на избраната планина

На слика 57 е прикажана листата од атрактивни туристички места во близина на избраната планина. Се повикува кога ќе се притисне копчето 'Туристички места' на страната кај што се листаат планинарските водичи. Секој корисник може да додава и да изменува записи за туристички места.



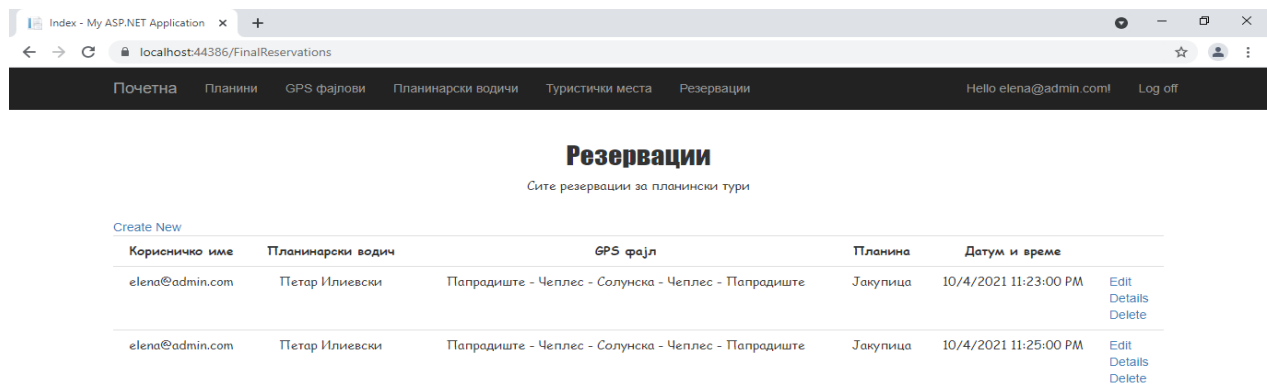
Слика 58: Резервација – 1

На слика 58 е прикажана картичката за резервација.



Слика 59: Резервација - 2

На слка 59 се прикажани избраните параметри за кои се извршува резервацијата. Тука се избира датум и време за планинарската тура и се завршува со регистрацијата.



Слика 60: Сите резервации

На слка 60 се прикажани сите резервации.

6. Заклучок

Во овој дипломски труд е изработена веб апликација за планински туризам. Имплементирана е веб апликација која можат да ја користат било каков тип на корисници кои се претходно регистрирани на истата.

Корисниците можат да додаваат нови GPS рути, да симнуваат постоечки, да прават преглед на понудени планини каде што се закажани настани за планинарење, можат да избираат планинарски водичи, и да направат преглед на планинарскиот водич дали тој има лиценца или не, колку години има планинарско искуство како и опис за самиот планинар.

Целта на овој дипломски труд е апликацијата да има разбирлив интерфејс, да овозможува лесна интеракција со корисниците и да одговара на потребите и барањата на корисниците.

Главната цел е да им се овозможи на корисниците се на едно место, да имаат јасен преглед на сите содржини поврзани со планински туризам.

Исто така, многу е важно со тоа да се развие планинскиот туризам, тоа би придонело и до поголем развој на самата веб апликација.

7. Референци

- [1] Structured Query Language – SQL, <https://en.wikipedia.org/wiki/SQL>
- [2] JavaScript, https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
- [3] Asynchronous JavaScript And XML - AJAX, https://www.w3schools.com/xml/ajax_intro.asp
- [4] jQuery, <https://jquery.com/>
- [5] Hypertext Markup Language - HTML, <https://developer.mozilla.org/en-US/docs/Web/HTML>
- [6] Cascading Style Sheets - CSS, <https://developer.mozilla.org/en-US/docs/Web/CSS>
- [7] C#, <https://docs.microsoft.com/en-us/dotnet/csharp/>
- [8] ASP .Net MVC, <https://dotnet.microsoft.com/apps/aspnet/mvc>
- [9] Application Programming Interface - API, <https://www.mulesoft.com/resources/api/what-is-an-api>
- [10] Bootstrap, <https://getbootstrap.com/>
- [11] Date Time Picker – Bootstrap, <https://www.malot.fr/bootstrap-datetimepicker/>
- [12] Авторизација и автентикација, <https://www.okta.com/identity-101/authentication-vs-authorization/>
- [13] yr.no/nb - <https://www.yr.no/nb>
- [14] fpsm.org.mk - <http://www.fpsm.org.mk/wp-content/uploads/2015/01/Pravilnik-za-planinarski-vodici-pri-FPSM.pdf>
- [15] booking.com - <https://www.booking.com/>