

B2 d.o.o.
VIŠJA STROKOVNA ŠOLA

DIPLOMSKO DELO

Jure Jurečič

B2 d.o.o.
VIŠJA STROKOVNA ŠOLA

DIPLOMSKO DELO

**Razvoj spletne aplikacije, namenjene
plezalcem**

Ljubljana, september 2020

Jure Jurečič

IZJAVA O AVTORSTVU

Spodaj podpisani Jure Jurečič, študent B2 Višje strokovne šole v Ljubljani, izjavljam, da sem avtor diplomskega dela z naslovom Razvoj spletne aplikacije, namenjene plezalcem, pripravljenega v sodelovanju z mentorjem Aleksandrom Lazarevičem.

Izrecno izjavljam, da v skladu z določili Zakona o avtorskih in sorodnih pravicah (Ur. l. RS, št. 21/1995 s spremembami) dovolim objavo diplomskega dela na spletnih straneh šole in objavo bibliografskih podatkov s povzetkom v sistemu Cobiss.

S svojim podpisom zagotavljam, da

- sta predloženi tiskana in elektronska verzija besedila diplomskega dela enaki;
- je predloženo besedilo rezultat izključno mojega lastnega raziskovalnega dela;
- je predloženo besedilo jezikovno korektno (lektorirano) in tehnično pripravljeno v skladu z Navodili za izdelavo diplomskih del B2 Višje strokovne šole, kar pomeni, da sem
 - poskrbel, da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam v diplomskem delu, citirana oziroma navedena v skladu z Navodili za izdelavo diplomskih del B2 Višje strokovne šole in
 - pridobil vsa dovoljenja za uporabo avtorskih del, ki so v celoti (v pisni ali grafični obliki) uporabljena v besedilu, in sem to v besedilu tudi jasno zapisal;
- se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku (Ur. l. RS, št. 55/2008 s spremembami).

V Ljubljani, 28. avgust 2020

Podpis avtorja: _____

ZAHVALA

Zahvaljujem se mentorju Aleksandru Lazareviću za pomoč in spodbudo pri izdelavi diplomske naloge.

POVZETEK

V diplomskem delu je predstavljena tehnologija Blazor. S pomočjo tega novega ogrodja sem sprogramiral spletno aplikacijo VPlati, ki je namenjena ljubiteljem plezanja in hrani podatke o slovenskih plezališčih ter plezalcih, ki portal uporabljajo. Aplikacija bi potrebovala še kar nekaj razvoja, da bi jo lahko uporabili v praksi, je pa dovolj obsežna, da vsebuje vse osnovne elemente Blazor ogrodja.

KLJUČNE BESEDE

VPlati, Spletna aplikacija CSS, Blazor, HTML, ASP.NET Core, Entity Framework Core, Identity Framework, Visual Studio 2019, Razor Pages, Razor, Plezanje

KAZALO VSEBINE

1	Uvod	1
2	Orodja in tehnologije.....	1
2.1	HTML	1
2.2	CSS – Bootstrap.....	1
2.3	C#.....	2
2.4	ASP.NET Core	2
2.5	Blazor	2
2.5.1	Komponente	2
2.5.2	Blazor WebAssembly	3
2.5.3	Blazor Server.....	4
2.5.4	Primerjava Blazor WebAssembly and Blazor Server	5
2.6	Entity Framework Core.....	6
2.7	Identity framework	6
2.8	Visual Studio 2019.....	6
3	Načrt in razvoj aplikacije.....	7
3.1	Analiza obstoječih rešitev	7
3.1.1	Plezanje.net	7
3.1.2	Friko.si	7
3.1.3	TheCrag.com	8
3.1.4	ClimbFinder	8
3.2	Načrtovanje aplikacije.....	9
3.2.1	Izbira tehnologije	9
3.2.2	Funkcionalnosti aplikacije	9
3.2.3	Uporabniški vmesnik	9
3.3	Razvoj aplikacije	12
3.3.1	Kreiranje projekta	12
3.3.2	Kreiranje razredov	14
3.3.3	Dostop do podatkov	15
3.3.4	Kreiranje podatkovne baze s pomočjo Entity Framework Core	15
3.3.5	Baza podatkov	16
3.3.6	Odlagališče	17

3.3.7	Izdelava komponent	18
3.3.8	Dodajanje ASP.NET Identity ogrodja v projekt.....	25
3.3.9	ASP.NET Identity roles – Vloge uporabnikov.....	26
3.3.10	Pooblastila	27
4	Navodila za uporabo aplikacije VPlati.....	29
4.1	Anonimni uporabnik	29
4.2	Registriran uporabnik v skupini »user«	29
4.3	Registriran uporabnik v skupini »admin«	33
5	Sklepi.....	35
6	Viri	37

KAZALO SLIK

Slika 1: Primer razor datoteke	3
Slika 2: Primer razdeljene razor datoteke v drevesu projekta	3
Slika 3: Prikaz koncepta Blazor WebAssembly	4
Slika 4: Prikaz koncepta Blazor Server.....	5
Slika 5: Primer portala Plezanje.net	7
Slika 6: Primer portala theCrag	8
Slika 7: Primer dizajna prve strani iz programa	10
Slika 8: Primer dizajna podstrani plezališča.....	10
Slika 9: Primer dizajna podstrani plezališče.....	11
Slika 10: Primer dizajna podstrani smer	11
Slika 11: Primer dizajna podstrani plezalci	12
Slika 12: Primer dizajna podstrani plezalec detajl.....	12
Slika 13: Direktoriji in datoteke v predlogi.....	13
Slika 14: Osnovna aplikacija pognana iz predloge	13
Slika 15: Primer strukture datoteke Models	14
Slika 16: Primer razreda Sektor	14
Slika 17: Primer arhitekture dostopa podatkov	15
Slika 18: Primer nastavitev v Startup.cs datoteki	15
Slika 19: Primer definicij podatkovnih tabel iz datoteke ApplicationDbContext.cs	16
Slika 20: Diagram SQL povezav	17
Slika 21: Primer odlagališča	18
Slika 22: Primer injiciranja odlagališča v razor komponento	18
Slika 23: Primer izgleda komponente PlezaliscaIndex.razor	19
Slika 24: Vsebina PlezalisceIndex.razor datoteke	19
Slika 25: Zgradba direktorija Pages	20
Slika 26: Primer vezave podatkov	21
Slika 27: Primer klica komponente PrikaziOpozorilo	21
Slika 28: Primer atributa parameter nad lastnostjo Opozorilo	21
Slika 29: Primer kaskadnega parametra.....	22

Slika 30: Primer EventCallback in InvokeAsync klic ob izbrisu opozorila	22
Slika 31: Primer obrazca potrdi brisanje	23
Slika 32: Primer uporabe komponente PotrдиComponent	23
Slika 33: Primer uporabe DataAnotations	23
Slika 34: Primer kode za generiranje obrazca	24
Slika 35: Primer uporabe ModalComponent	24
Slika 36: Add Identity Scaffold	25
Slika 37: Razred Plezalec	26
Slika 38: Prijavno okno v našo aplikacijo pred modifikacijami	26
Slika 39: Primer uporabe komponente AuthorizeView	27
Slika 40: Primer z Authorized in NotAuthorized	27
Slika 41: Visual Studio 2019: Primer atributa Authorize	28
Slika 42: Primer dostopa do prijavljenega uporabnika s pomočjo kaskadnega parametra ..	29
Slika 43: Primer registracijskega obrazca	30
Slika 44: Prva stran aplikacije	30
Slika 45: Primer podstrani za urejanje profila uporabnika	31
Slika 46: Prikaz plezališč	31
Slika 47: Primer detajlne podstrani za posamezno plezališče	32
Slika 48: Primer detajlne podstrani smeri	33
Slika 49: Primer komentarjev	33
Slika 50: Dodaj novo plezališče	34
Slika 51: Uredi / Izbriši plezališče	34
Slika 52: Obrazec za dodajanje nove smeri	35

KAZALO TABEL

Tabela 1: Prednosti in slabosti Blazor WebAssembly	5
Tabela 2: Prednosti in slabosti Blazor Server	6

UPORABLJENI SIMBOLI

HTML – Hyper Text Markup Language

CSS – Cascading Style Sheets

API – Application Programming Interface

1 UVOD

Razvoj telekomunikacij in pametne naprave so nam močno olajšale vsakdanje življenje. Dandanes ima večina dostop do spletnih vsebin, zato je porast spletnih aplikacij in razvoj novih tehnologij zelo hiter.

Osnovni gradnik spletnih aplikacij je HTML, standard, ki nam omogoča prikazovanje vsebine na spletu. Za oblikovanje vsebin se uporablja CSS, za interaktivnost spletnih strani skrbi skriptni jezik JavaScript, za prikazovanje HTML strani pa se uporablja brskalnik.

Do nedavnega je za interaktivnost spletnih strani na strani uporabnika skrbel samo programski jezik JavaScript. Leta 2015 je bil uradno predstavljen WebAssembly, ki nam omogoča kompilacijo kode v brskalniku tudi za druge programske jezike. Standard podpirajo vsi najnovejši brskalniki.

V začetku leta 2018 je Microsoft prvič najavil novo ogrodje, ki naj bi podpiralo delovanje na standardu WebAssembly. Ogrodje je dobilo ime Blazor. Aprila 2019 so sporočili, da je javnosti na voljo njegova prva testna različica, uradno pa je tehnologija izšla 19. maja 2020. Blazor nam omogoča programiranje spletnih aplikacij s pomočjo programskega jezika C#. Diplomaska naloga je sprogramirana v tem ogrodju.

Ideja za samo aplikacijo je nastala zaradi moje ljubezni do plezanja in poznavanja problematike podobnih aplikacij, ki že obstajajo na tržišču. Cilj naloge je sprogramirati spletno aplikacijo brez direktne uporabe JavaScript programskega jezika.

2 ORODJA IN TEHNOLOGIJE

V tem poglavju bom predstavil osnovne gradnike, ki sem jih uporabil v aplikaciji. V aplikaciji je zajetih zelo veliko različnih orodij in principov, zato sem se odločil, da bom za lažje razumevanje vsebine te koncepte najprej predstavil.

2.1 HTML

HTML je standard, ki se uporablja za prikazovanje spletnih vsebin v brskalnikih. Njegov razvoj sega v leto 1991, ko je Tim Berners-Lee prvič omenil HTML v zapisih HTML Tags. Uradno je izšel leta 1995 kot HTML 2.0, nato pa so se razvijale nove verzije vse do verzije HTML5, ki se kot standard uporablja še danes. HTML je osnovni gradnik vsake spletne aplikacije ali navadnega spletišča.

2.2 CSS – BOOTSTRAP

CSS so podloge, ki skrbijo za prezentacijo spletnih strani. Z njimi definiramo, kako naj se določeni HTML elementi prikažejo v spletni aplikaciji.

Bootstrap je ogrodje, ki nam pomaga pri dizajnu spletnih strani. Vsebuje predloge za določene HTML elemente, ki jih lahko uporabimo v naših aplikacijah. S pomočjo tega ogrodja lahko tudi tisti, ki nam dizajn ne gre najbolje od rok, lahko ustvarimo aplikacijo modernega videza brez večjih težav.

2.3 C#

C# je objektno usmerjen večnamenski programski jezik. Razvit je bil okoli leta 2000 pri Microsoftu kot nekakšna konkurenca jeziku Java. C# se še vedno razvija, trenutno pa obstaja različica jezika C# 8. Uporaba jezika je zelo vsestranska, saj omogoča razvoj različnih namiznih, spletnih, mobilnih aplikacij, iger itd. S C# programskim jezikom sem se prvič srečal tekom študija pri predmetu Programiranje 1. Pred tem sem že poznal programski jezik Java in glede na izkušnje lahko trdim, da sta oba programska jezika na prvi pogled zelo podobna. S pomočjo C# sem za potrebe študija že izdelal nekaj manjših projektov, kot so igra Vislice, igra Poker in pa spletna aplikacija, ki je razvita s pomočjo tehnologije Razor Pages.

2.4 ASP.NET CORE

ASP.NET Core je novo odprtokodno ogrodje, ki služi za programiranje novih modernih spletnih in mobilnih aplikacij. Ogrodje je bilo dizajnirano za optimizirano delovanje v oblaku ali na lokalnih sistemih ter omogoča razvoj na Windows, Mac in Linux platformah. Menim, da je Microsoft z razvojem ASP.NET Core ogrodja pokazal, da ima željo po večjem deležu uporabe svojih produktov v izdelkih na tržišču in s tem povečati vpliv svojih orodij. Trenutno uradna podprta verzija je .NET Core 3.1, testno pa je na tržišču že .NET 5.

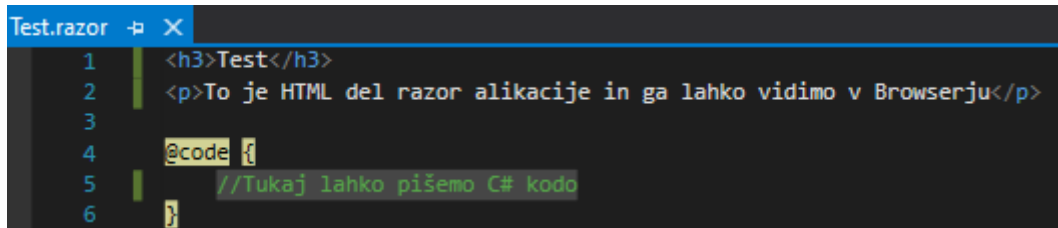
2.5 BLAZOR

Blazor nam omogoča izdelavo interaktivnih aplikacij z uporabo C# programskega jezika. Blazor aplikacije so sestavljene s pomočjo komponent narejenih v C#, HTML in CSS. Posebnost tega ogrodja je, da lahko kodo pišemo v programskem jeziku C# tako na strežniški strani aplikacije kot tudi na uporabnikovi strani aplikacije, kar je bilo pred uvedbo WebAssembly-a v brskalnike nemogoče. Prav tako lahko uporabljamo vse obstoječe .NET knjižnice, ki so že del sistema .NET.

2.5.1 Komponente

Komponente, iz katerih je sestavljen Blazor, so uporabniški vmesniki (na primer obrazec za vnos podatkov ali pa spletna stran). Komponente so del .NET razredov, ki definirajo interaktivno uporabniško izkušnjo, se odzivajo na uporabniško interakcijo, lahko jih gnezdimo in ponovno uporabimo. V Blazor aplikaciji je komponenta sestavljena iz .razor datoteke. Razor je sintaksa, ki združuje HTML in C# kodo.

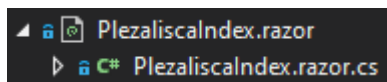
Slika 1: Primer razor datoteke



```
Test.razor 1 <h3>Test</h3>
2 <p>To je HTML del razor aplikacije in ga lahko vidimo v Browserju</p>
3
4 @code {
5     //Tukaj lahko pišemo C# kodo
6 }
```

Programerji ponavadi želimo imeti svojo kodo čim bolj organizirano, zato lahko razor datoteke tudi razdelimo na dva dela, in sicer na HTML in na C# del. Tako dobimo dve datoteki. Na voljo imamo več opcij, sam sem se odločil, da datoteko razbijem na dva dela tako, da C# del kode poimenujem s končnico »razor.cs«. Visual studio 2019 nam potem v drevesu projekta prikaže obe datoteki združeni (glej sliko 2).

Slika 2: Primer razdeljene razor datoteke v drevesu projekta

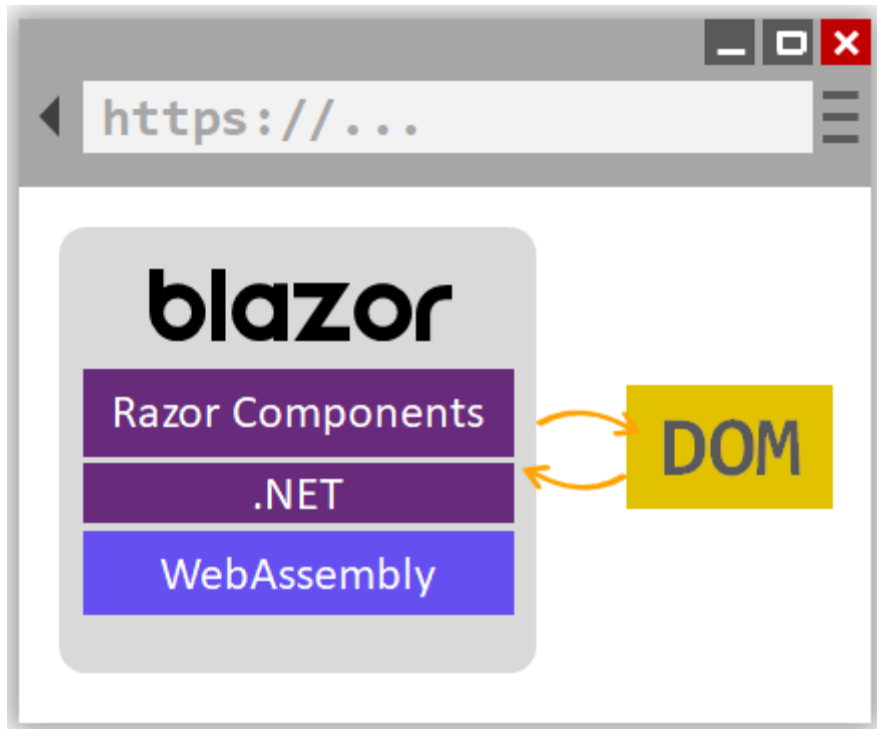


C# del datoteke mora podedovati iz razreda `ComponentBase`, HTML del datoteke pa potrebuje referenco na C# del datoteke.

2.5.2 Blazor WebAssembly

Blazor ogrodje ima možnost dveh različnih pristopov. Eden od teh je pristop s pomočjo `WebAssembly`, kar pomeni, da aplikacijo razvijamo za samostojno delovanje v brskalniku. Vse datoteke iz aplikacije se prenesejo v brskalnik, ki to kodo prevede in nam aplikacijo predvaja lokalno. Aplikacija se izvaja na strani uporabnika.

Slika 3: Prikaz koncepta Blazor WebAssembly

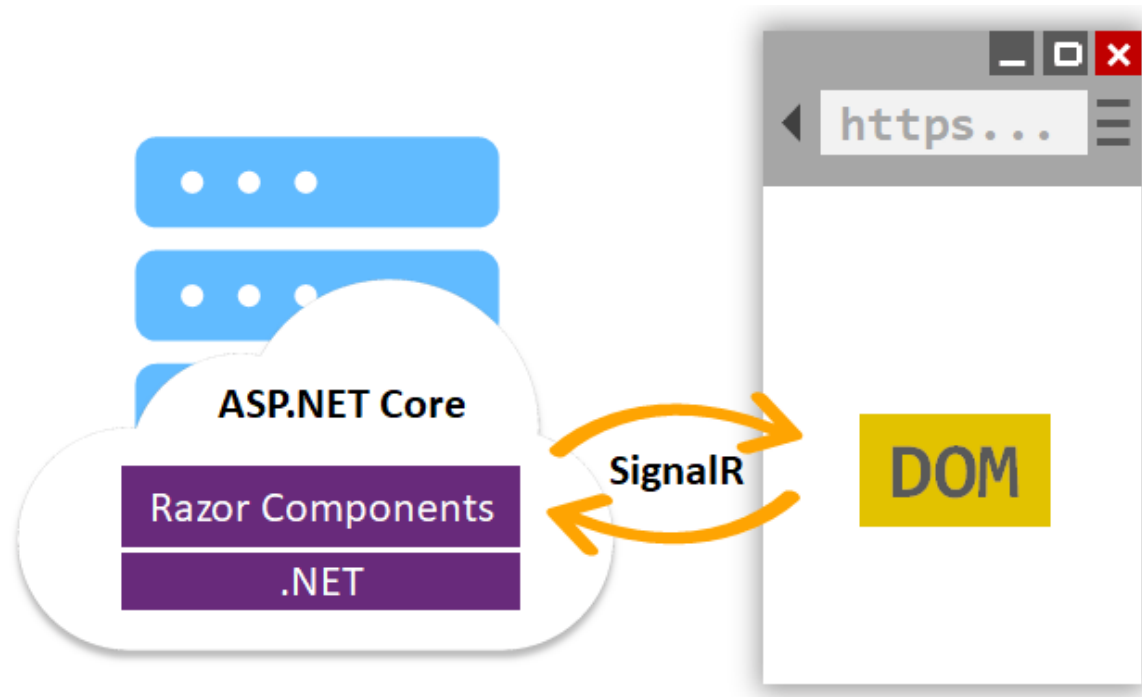


Dokumentacija Microsoft Blazor: <https://docs.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-3.1>

2.5.3 Blazor Server

Druga možnost uporabe Blazor ogrodja je tako imenovana Blazor Server, ki omogoča gostovanje Razor komponent na strežniku ASP.NET Core aplikacije. Uporabniški vmesnik se posodablja s pomočjo SignalR povezav.

Slika 4: Prikaz koncepta Blazor Server



Dokumentacija Microsoft Blazor: <https://docs.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-3.1>

SignalR povezava je odprtokodna knjižnica, napisana v ASP.NET, ki olajšuje delovanje spletnih aplikacij v realnem času. To nam omogoča, da se koda, ki se izvaja na strežniku, do uporabnika prenese takoj.

2.5.4 Primerjava Blazor WebAssembly and Blazor Server

Prednosti in slabosti obeh možnosti bom predstavil v tabelah.

Tabela 1: Prednosti in slabosti Blazor WebAssembly

Blazor WebAssembly	
Prednosti	Slabosti
Ne potrebujemo strežnika.	Prva povezava traja dlje, ker se mora celotna aplikacija naložiti na uporabnikov računalnik.
Aplikacija uporablja vire uporabnikovega računalnika.	Omejenost glede na kapacitete brskalnika.
Lahko gostuje na našem lastnem strežniku.	Uporabnik potrebuje zmogljiv računalnik.
Ne potrebujemo ASP.NET Core web stržnika.	

Tabela 2: Prednosti in slabosti Blazor Server

Blazor Server	
Prednosti	Slabosti
Povezava do aplikacije je hitrejša.	Potrebujemo ASP.NET Core strežnik.
Omejena je le na vire strežnika.	Potrebujemo aktivno povezavo do strežnika, sicer aplikacija ne deluje.
Uporabnik potrebuje le brskalnik.	Lahko pride do zakasnitev zaradi oddaljenosti uporabnika od strežnika.
Aplikacija je bolj varna od WebAssembly rešitve.	Omejena možnost razširitev.

Kot lahko vidimo iz primerjav, imata obe rešitvi določene prednosti in slabosti. Sam sem se odločil za rešitev Blazor Server. V primeru, da bi želel razviti aplikacijo na podlagi Blazor WebAssembly, bi potreboval še več drugih znanj, kar pa bi mogoče na koncu zasenčilo samo idejo diplomske naloge, ki je uporaba C# programskega jezika za izgradnjo cele aplikacije.

2.6 ENTITY FRAMEWORK CORE

Entity Framework Core je odprtokodna tehnologija za dostop do baze podatkov. S pomočjo te tehnologije lahko preslikamo naše objekte iz aplikacije direktno v neko bazo podatkov ali obratno. Podpira vse moderne baze podatkov.

2.7 IDENTITY FRAMEWORK

Identity Framework je API, ki nam omogoča funkcionalnosti prijave v uporabniški vmesnik aplikacije. Ogrodje upravlja z uporabnikovimi gesli, podatki itd. S pomočjo tega ogrodja lahko v aplikaciji naredimo različne profile uporabnikov.

2.8 VISUAL STUDIO 2019

Visual studio 2019 je eden glavnih pripomočkov, ki sem ga uporabljal za izdelavo aplikacije. Gre za urejevalnik kode, s pomočjo katerega zgradimo celotno aplikacijo. Prav tako vsebuje že prednarejene komponente različnih ogrodij. Aplikacija je za osebno uporabo brezplačna.

3 NAČRT IN RAZVOJ APLIKACIJE

3.1 ANALIZA OBSTOJEČIH REŠITEV

V primeru, da razvijamo aplikacijo, ki že obstaja na tržišču, je vedno najbolje, da na začetku analiziramo obstoječe rešitve. S tem dobimo dragocene informacije in se pri našem načrtovanju lažje odločamo, katere elemente je smiselno integrirati v našo aplikacijo iz obstoječih rešitev oz. kaj le tem manjka, da bo naša aplikacija na koncu še boljša.

3.1.1 Plezanje.net

Plezanje.net je spletna stran, ki je med slovenskimi plezalci najbolj uporabljena. Stran je relativno preprosta za uporabo, ponuja nam mnogo funkcionalnosti, kot so ocenjevanje in komentiranje smeri, dodajanje novih plezališč, sektorjev in smeri, izdajanje opozoril. Vsebuje večino javno znanih plezališč iz Slovenije, prav tako je vnesenih tudi veliko plezališč, ki se nahajajo v tujini. Kot prednost te aplikacije vidim dolgo zgodovino, kar posledično pomeni veliko bazo podatkov in veliko število uporabnikov, slabost pa vidim v tem, da je vizualno in aplikativno že malo zastarela in bi bila potrebna konkretne prenove. Prav tako določene funkcionalnosti spletne aplikacije več ne delujejo.

Slika 5: Primer portala Plezanje.net

Pregled težavnosti

4a ... 4c+10265a ... 5c+1686a ... 6b+476c ... 7a+247b ... 7c+168a ... 9b

Smeri

Prikaži ...

Ime	Težavnost	Dolžina	Avtor/prvi vzpon	Druge informacije
F - Stara Čreta (leva stena)				
Najbolj hude alpske bejbe	6b	13 m	Klemen Peterlin(2017)/Klemen Peterlin ...	
Prvi koraki	7a	14 m	Sašo Ocirk(2014)	[1 komentar]
Čista jeba	7a+			
Družinska sreča	7a/			
Polhova poč	6b			
Luska direkt	6b			
Malič - Čreta	6b			
Ne podirajte mojih kostanjev	7a	15 m	Albin Simonič(2009)	
Spominska knjiga	6b+	15 m	Albin Simonič(2009)	
Bori šumijo	6c+/7a	15 m	Albin Simonič(2009)	
Lepi časi	6b	15 m	Albin Simonič(2009)	[1 komentar]
Prehitri lokanci	6a+	15 m	Albin Simonič(2009)	[1 komentar]
G - Stara Čreta (srednja stena)				
Sidro	6c+	10 m	Sašo Ocirk(2016)	
Pikica	6b+/c	10 m	Albin Simonič(2009)	
Leteče luske	5c	12 m	Albin Simonič(2009)	
Skrite luknje	6a+	12 m	Albin Simonič(2009)	
Bukva direkt	6b	12 m	Albin Simonič(2009)	
Potres	6b	12 m	Albin Simonič(2009)	
Garaško poletje	6b+	12 m	Albin Simonič(2009)	
Mikjeva	7a	10 m	Sašo Ocirk(2016)	

Komentarji

V luknji kakšne pol metra levo od linije svedrov sem danes srečal sovo. Če gnezdi ne vem. Drugače pa lepa smer, meni veliko težja od Družinske sreče.

Žiga Rozman, 9.4.2017 20:48:19

Plezanje.net: <https://www.plezanje.net/climbing/db/showCrag.asp?crag=1211>

3.1.2 Friko.si

Novejša spletna aplikacija, ki pa ni namenjena orientaciji plezalcev po slovenskih plezališčih. Vsebuje le nekaj zapisov plezališč, vendar pa nam ne omogoča interakcije.

Menim, da je stran zamišljena kot neke vrste plezalni portal, kjer so predstavljene novice iz sveta plezanja oziroma alpinizma. Stran sem omenil predvsem zaradi tega, ker mi je všeč njihov dizajn vodiča po plezališčih – podoben nadgrajen dizajn bom tudi sam implementiral v svojo aplikacijo.

3.1.3 TheCrag.com

Tuja spletna aplikacija, ki vsebuje večino elementov, ki bi jih tudi sam želel imeti v lastni aplikaciji. Vsebuje vse, kar ima slovenski portal Plezanje.net, prav tako pa omogoča vodenje lastne evidence preplezanih smeri in prikazovanje različnih statistik, ki jih hranijo v svojih bazah podatkov. Slabost tega portala je predvsem v tem, da ni tako popularen med slovenskimi plezalcami, kar lahko ugotovimo že po hitrem pregledu baze uporabnikov, posledično se v njihovi bazi ne nahajajo vsa slovenska plezališča.

Slika 6: Primer portala theCrag



Thecrag: <https://www.thecrag.com/climbing/slovenia>

3.1.4 ClimbFinder

Mobilna aplikacija, ki je kopija slovenskega vodnika v digitalni obliki. Menim, da tehnologija ni najbolj priljubljena tema med plezalcami, saj je večina naravnana na življenje v naravi, zato dosti plezalcev za raziskovanje slovenskih plezališč še vedno uporablja knjige. Založba Sidarta se je odločila, da izda mobilno aplikacijo, ki vsebuje vsa uradna plezališča. Aplikacijo je potrebno kupiti, kar je zagotovo velik minus, saj niso vsi za to pripravljeni plačati, osebno pa se mi zdi zelo uporabna, saj danes vedno hodimo okoli s pametno napravo v žepu. Prednost te aplikacije je zagotovo vizualna predstava vseh smeri v plezališčih, prav tako pa vsebuje vse GPS koordinate parkirišč v bližini plezališč. Kot slabost aplikacije je

potrebno omeniti, da aplikacija ni interaktivna, tako da so vse ocene smeri povzete iz predlaganih ocen postavljavca. Ocene smeri so sicer dinamične, saj ima vsak posameznik pravico postaviti svojo oceno v primeru, da smer prepleza, tako da je to podatek, ki ga ne moremo dobro predstaviti v statični aplikaciji. Prav tako ne moremo ostalih plezalcev opozoriti na nevarnosti, ki nad nami pretijo v plezališčih (kače, razmajan kamen itd.).

3.2 NAČRTOVANJE APLIKACIJE

3.2.1 Izbira tehnologije

Izbira tehnologije sem določil že ob izbiri teme za diplomsko nalogo. Za ogrodje Blazor Server sem se odločil, ker je ena najnovejših tehnologij za razvoj spletnih aplikacij. Za komercialno produkcijo takšne aplikacije bi bilo mogoče boljše kakšno drugo JavaScript ogrodje, ampak menim, da bo aplikacija vseeno zelo zanimiva in uporabna.

3.2.2 Funkcionalnosti aplikacije

Glede na analizo obstoječih rešitev želim v svojo aplikacijo integrirati naslednje funkcionalnosti:

- Registracija uporabnikov
- Profil uporabnikov
- Prikaz plezališč
- Vnos novega plezališča, ki ima različne sektorje, v katerih se nahajajo različne smeri
- Komentiranje smeri
- Ocenjevanje smeri
- Izdajanje opozoril na nivoju plezališč
- Različni nivoji uporabe glede na pravice uporabnikov (Administrator, Registriran uporabnik, Anonimen uporabnik)
- Hranjenje podatkov v podatkovni bazi

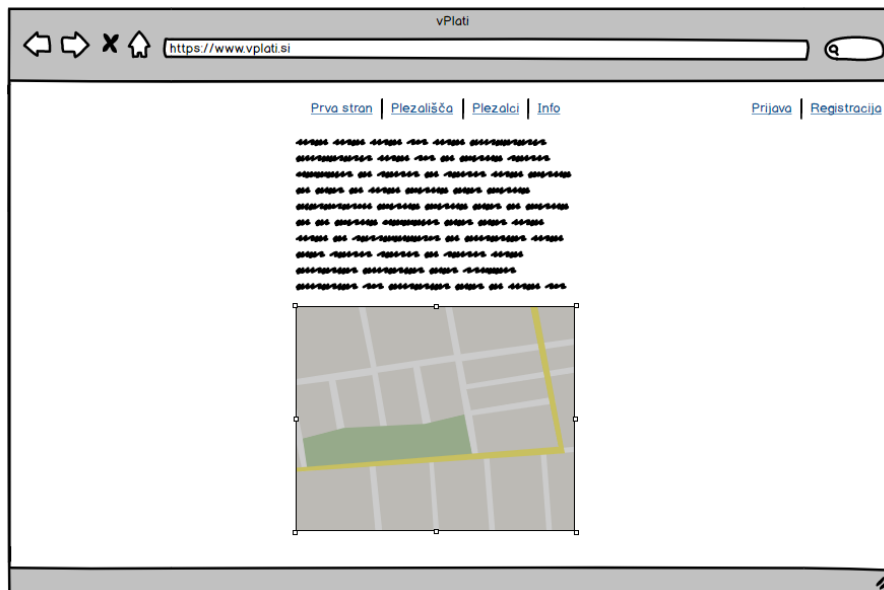
3.2.3 Uporabniški vmesnik

Pred začetkom izdelave aplikacije sem si pripravil nek okvir uporabniškega vmesnika, ki ga bom skušal implementirati v svojo aplikacijo. Vmesnik sem pripravil s pomočjo programa Balsamiq. S tem sem nekako določil smernice, ki se jih bom skušal držati pri izdelavi aplikacije.

3.2.3.1 Prva stran

Primer prve strani. Prva stran bo vsebovala neko naslovno sliko in nekaj besedila o portalu.

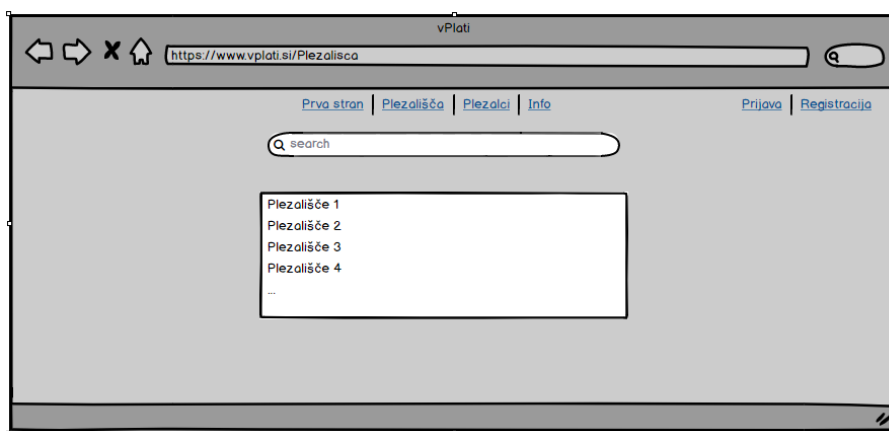
Slika 7: Primer dizajna prve strani iz programa



3.2.3.2 Podstran plezališča

Primer uporabniškega vmesnika za plezališča, kjer bo možnost iskanja in pa izbire plezališča s seznama.

Slika 8: Primer dizajna podstrani plezališča



3.2.3.3 Podstran plezališče

Primer vmesnika za posamezno plezališče. Na vrhu se nahaja ime plezališča, v grafikonu je seznam ocen smeri v plezališču, sledijo pa opozorila in seznam sektorjev s posameznimi smermi. S klikom na posamezno smer se nam odpre pogled posamezne smeri.

Slika 9: Primer dizajna podstrani plezališče

https://www.vplati.si/Plezalisca

Prva stran | Plezališča | Plezalci | Info | Prijava | Registracija

Plezališče 1

Opozorilo: Lorem Ipsum is simply dummy.

Sektor 1:

- Smer 1 Ocena: 5b
- Smer 2 Ocena: 6a
- Smer 3 Ocena: 4a

Sektor 2:

- Smer 1 Ocena: 5b
- Smer 2 Ocena: 6a
- Smer 3 Ocena: 4a

3.2.3.4 Podstran smer

Vmesnik za posamezno smer. Vsebuje opozorilo, izdano za posamezno smer, osnovno statistiko smeri, predlagane ocene in komentarje.

Slika 10: Primer dizajna podstrani smer

https://www.vplati.si/Plezalisca/Sektor/Smer

Prva stran | Plezališča | Plezalci | Info | Prijava | Registracija

Smer

Opozorilo: Lorem Ipsum is simply dummy te

Težavnost: 5b
Dolžina: 15m
Zabeleženi vzponi: 12
Avtor: Ime Avtorja Smeri

Ocene:

5b	Ime ocenjevalca	Datum
5a	Ime ocenjevalca	Datum
5c	Ime ocenjevalca	Datum
5b	Ime ocenjevalca	Datum

Komentarji:

Ta smer je res težka.
Ime komentatorja Datum

Komentarji:

Ta smer je res težka.
Ime komentatorja Datum

3.2.3.5 Podstran plezalci

Primer vmesnika plezalcev. Plezalce je mogoče iskati po vzdevku, imenu ali priimku. Prav tako bomo imeli možnost filtriranja uporabnikov glede na določene attribute (preplezane smeri, najtežje preplezana smer, ...).

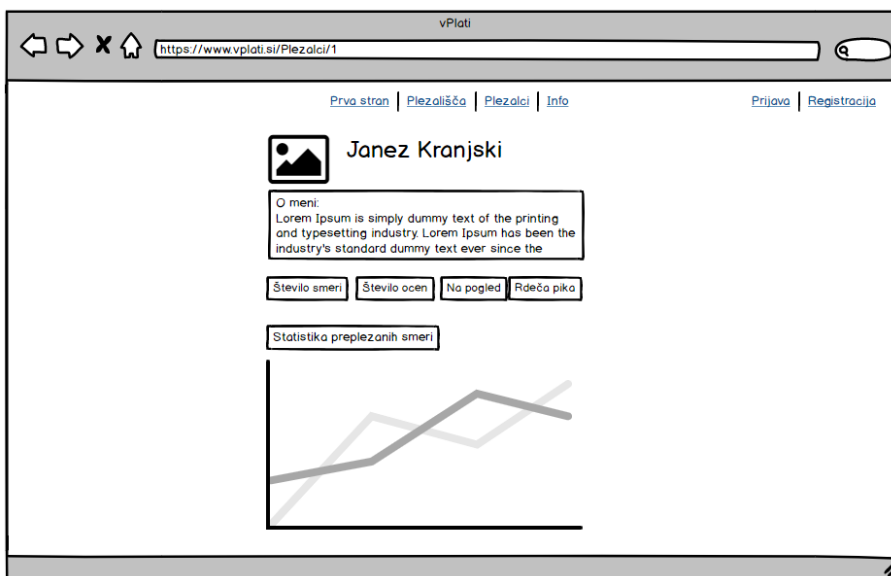
Slika 11: Primer dizajna podstrani plezalci



3.2.3.6 Podstran plezalec

Primer strani registriranega uporabnika vsebuje podatke o plezalcu, ki jih bo vnesel ob registraciji profila. Prav tako se bodo izpisale in izrisale različne statistične karakteristike plezalca.

Slika 12: Primer dizajna podstrani plezalec detajl



3.3 RAZVOJ APLIKACIJE

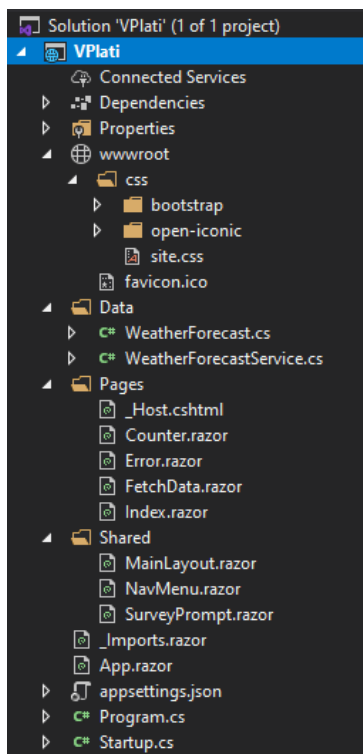
Tukaj se bom osredotočil na delo, ki je bilo potrebno za razvoj aplikacije. Z ogrođjem Blazor se še nisem srečal, tako da sem med razvojem aplikacije včasih prišel v slepo ulico in potreboval veliko časa ter vložene energije, da sem določene izzive tudi obvladal.

3.3.1 Kreiranje projekta

V programu Visual Studio 2019 imamo že vnaprej pripravljene predloge za začetek našega projekta. Izbrana predloga je Blazor app – Blazor Server. Seveda ne smemo pozabiti tudi na

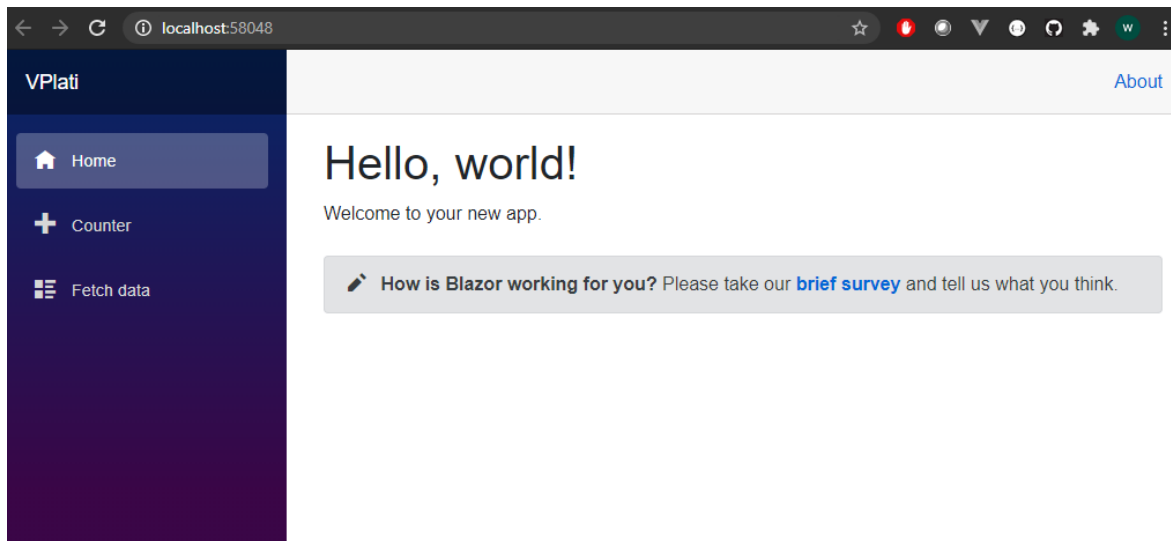
ime projekta. Odločil sem se, da bom aplikacijo poimenoval VPlati. Plata je izraz, s katerim se v plezalnem žargonu opiše neprevisno gladko steno.

Slika 13: Direktoriji in datoteke v predlogi



Če program takoj poženemo, se nam v brskalniku odpre predloga, ki jo je naredila Microsoftova ekipa. V njej že imamo navigacijski meni in postavitev osnovnih spletnih elementov. Seveda lahko po želji vse komponente dodatno spreminjamo.

Slika 14: Osnovna aplikacija pognana iz predloge

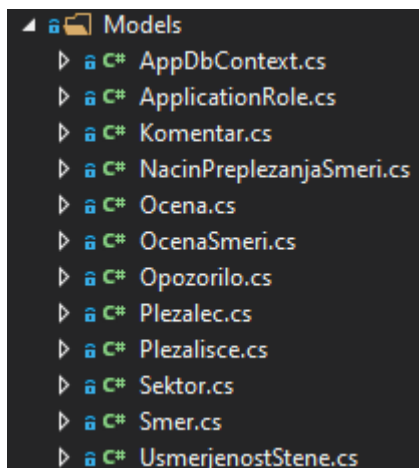


Kot lahko vidimo, je samo ogrodje aplikacije praktično že pripravljeno, tako da lahko svojo pozornost preusmerimo v izgradnjo razredov, ki jih bomo v aplikaciji uporabili.

3.3.2 Kreiranje razredov

Razredi, ki jih bomo uporabili v projektu, so zelo pomembni, saj so direktno povezani z hrambo podatkov in uporabo Entity Frameworka. Vse razrede sem kreiral v datoteki z imenom Models.

Slika 15: Primer strukture datoteke Models



Kot lahko vidimo, je v datoteki veliko različnih razredov, ki po večini predstavljajo naše objekte, ki smo si jih zamislili za samo aplikacijo. Ob samem začetku razvoja aplikacije še nisem imel narejenih vseh razredov, kot sem jih imel ob koncu, saj sem tekom razvoja ugotovil, da potrebujem dodatne razrede ali pa samo lastnosti znotraj razredov, zato sem jih seveda primerno prilagodil glede na potrebe.

Posebni razred, ki se ne nanaša direktno na naše objekte, je AppDbContext.cs. Ta razred je potreben za delovanje Entity Framework Core ogrodja. V njem imamo definirana pravila, ki se nanašajo na delo z bazo podatkov. Pri kreiranju razredov, ki jih bomo uporabili za generiranje naše baze podatkov, moramo biti pozorni, saj moramo že na tem mestu vedeti, kako bodo objekti med sabo povezani, zato moramo pravilno definirati vse lastnosti. Prav tako moramo v definiciji razredov definirati, katere lastnosti so ob vnosu v bazo podatkov obvezne ali pa kako so le te omejene glede na različne lastnosti podatkov.

Slika 16: Primer razreda Sektor

```
22 references
public class Sektor
{
    7 references
    public int Id { get; set; }
    [Required(ErrorMessage = "Ime sektorja je obvezen podatek!")]
    [MinLength(1, ErrorMessage = "Ime sektorja mora vsebovati vsaj 1 znak!")]
    12 references
    public string ImeSektorja { get; set; }
    5 references
    public List<Smer> Smeri { get; set; }
    8 references
    public int PlezalisceId { get; set; }
    4 references
    public Plezalisce Plezalisce { get; set; }
}
```


3.3.3 Dostop do podatkov

Zelo pomembna tema pri izdelavi spletne aplikacije je način dostopa do podatkov, ki jih želimo hraniti v neki bazi. Seveda imamo na izbiro več različnih pristopov dostopa do podatkov, moramo pa vedeti, na kakšen način bo naša aplikacija delovala, ter poznati prednosti in slabosti različnih pristopov. Eden od pristopov bi lahko bil direktni dostop do podatkov preko Blazor Server sloja. Ta je najlažji za implementacijo, prav tako pa zadostuje našim potrebam.

Slika 17: Primer arhitekture dostopa podatkov



Pragimtech: <https://www.pragimtech.com/blog/blazor/blazor-data-access-strategies/>

3.3.4 Kreiranje podatkovne baze s pomočjo Entity Framework Core

Najprej moramo v projekt naložiti zadnjo verzijo ogrodja, ki jo lahko prenesemo iz NuGet Package Manager orodja znotraj Visual Studia 2019. Nato moramo v datoteki Startup.cs, ki je del predloge, nastaviti določene parametre, ki določajo, katero bazo podatkov bomo uporabljali, ter kako se imenuje naš razred, ki ga potrebujemo za delo z podatki.

Slika 18: Primer nastavitve v Startup.cs datoteki

```
services.AddDbContext<AppDbContext>(options =>  
options.UseSqlServer(Configuration.GetConnectionString("DbConnection")));
```

Ker pred začetkom razvoja aplikacije nismo imeli obstoječe baze podatkov, je logična rešitev za kreiranje baze podatkov v našo aplikacijo s pomočjo pristopa Code-First, kar pomeni, da se bo baza podatkov naredila na podlagi .NET razredov. V AppDbContext.cs datoteki imamo tako nastavljene vse podatkovne tabele, ki jih bomo uporabljali za našo bazo podatkov.

Slika 19: Primer definicij podatkovnih tabel iz datoteke AppDbContext.cs

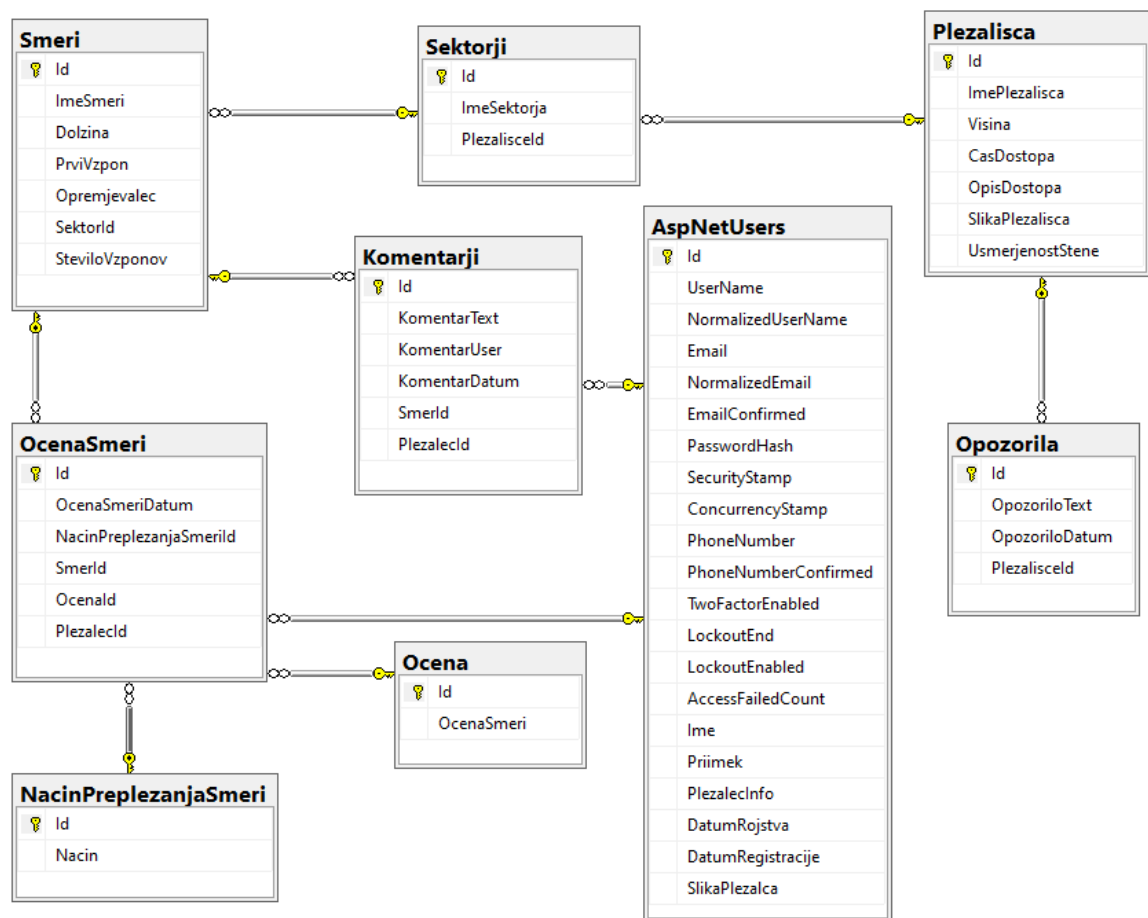
```
7 references
public DbSet<Plezalisce> Plezalisca { get; set; }
3 references
public DbSet<Sektor> Sektorji { get; set; }
4 references
public DbSet<Smer> Smeri { get; set; }
3 references
public DbSet<Komentar> Komentarji { get; set; }
4 references
public DbSet<Opozorilo> Opozorila { get; set; }
1 reference
public DbSet<Ocena> Ocena { get; set; }
2 references
public DbSet<OcenaSmeri> OcenaSmeri { get; set; }
1 reference
public DbSet<NacinPreplezanjaSmeri> NacinPreplezanjaSmeri { get; set; }
```

Konfiguracija nam vzame kar nekaj časa, medtem ko je sama izgradnja tabel, ko imamo vse skupaj pripravljeno, zelo hitra. V konzolo najprej vpišemo ukaz Add-Migration, ki nam zgradi vso potrebno kodo za izgradnjo baze podatkov, nato pa z ukazom Update-Database generiramo podatkovno bazo.

3.3.5 Baza podatkov

Za bazo podatkov sem izbral MSSQL, ki je že integrirana v sam Visual Studio 2019. To lahko z manjšimi popravki v konfiguraciji prenesemo na drug strežnik. Sama logika podatkovne baze je bila ustvarjena s pomočjo ogrodja Entity Framework Core. Poskrbeti sem moral samo za to, da so bili vsi razredi pravilno konfigurirani, da so nato relacije med tabelami jasne.

Slika 20: Diagram SQL povezav



3.3.6 Odlagališče

Odlagališče je abstraktni način dostopa do nivoja podatkovne baze. Ta tehnika je uporabna predvsem zato, ker nas dejansko ne zanima, na kakšen način pridemo do podatkov. V primeru, da se naša baza podatkov spremeni iz SQL v neko drugo podatkovno bazo, na ta način ni potrebne dodatne implementacije na nivoju klicev, ki jih to odlagališče uporablja. Takšnemu sistemu se reče tudi »loosely coupled system« – ohlapno povezan sistem. To tehniko sem se srečal tudi pri drugih projektih in moram priznati, da je njena izvedba res dobra praksa programiranja, zato sem jo uporabil tudi v tem projektu. Poleg tega pa je koda tudi lažje berljiva in na splošno bolj čista.

Slika 21: Primer odlagališča

```
public interface IVPlatiRepository
{
    2 references
    Task<IEnumerable<Plezalisce>> GetPlezalisca();
    1 reference
    Task<IEnumerable<Plezalisce>> Search(string ime);
    2 references
    Task<Plezalisce> GetPlezalisce(int plezalisceId);
    1 reference
    Task<Plezalisce> AddPlezalisce(Plezalisce plezalisce);
    1 reference
    Task<Plezalisce> UpdatePlezalisce(Plezalisce plezalisce);
    1 reference
    Task<Plezalisce> DeletePlezalisce(int plezalisceId);
    2 references
    Task<IEnumerable<Ocena>> GetOcena();
}
```

Do odlagališča lahko nato dostopamo iz komponent na takšen način, da injiciramo odlagališče v komponento. To naredimo v razor datotekah s pomočjo atributa Inject.

Slika 22: Primer injiciranja odlagališča v razor komponento

```
[Inject]
0 references
public IVPlatiRepository DataRepository { get; set; }
```

3.3.7 Izdelava komponent

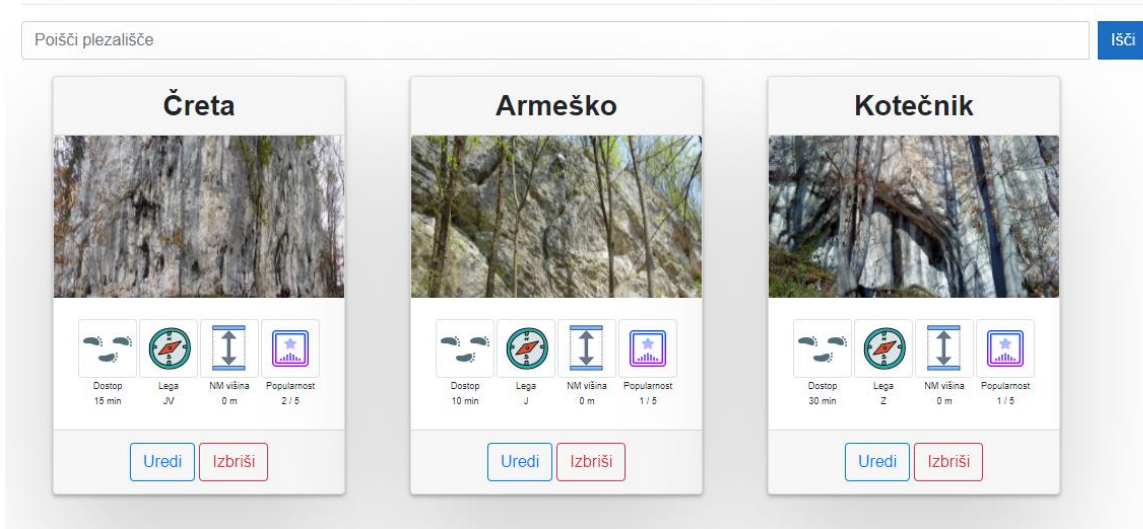
Samo programiranje se je začelo na tem mestu. Praktično vse, kar je bilo potrebno urediti do te točke, so bile samo priprave, da smo lahko začeli v našo aplikacijo implementirati razne funkcionalnosti. Implementiral sem jih eno za drugo in po vsaki implementaciji tudi preveril, če deluje tako, kot sem si zamislil. Prva komponenta, ki sem jo ustvaril, je bila prikaz vseh plezališč.

3.3.7.1 Komponente z direktivo @page

Prva komponenta, ki sem jo razvil, je bila PlezaliscaIndex. Namen komponente je prikaz vseh plezališč. Za sam dizajn sem se odločil, da uporabim kartico, ki vsebuje ime plezališča, sliko plezališča in štiri karakteristike plezališča: dostop, lega, nadmorska višina ter popularnost plezališča. Pod samo kartico sem tudi dodal dva gumba, ki nam omogočata dodatne funkcionalnosti naše aplikacije. Poleg tega sem želel v komponento dodati tudi iskalnik in nekaj filtrov, ki nam pomagajo, da lažje poiščemo želeno plezališče, v primeru, da imamo v bazi veliko plezališč. Tako lahko našo iskano plezališče poiščemo v iskalniku ali pa razporedimo plezališča po vrstnem redu glede na ime, število smeri v plezališču, popularnost ali pa čas dostopa do plezališča.

Slika 23: Primer izgleda komponente PlezaliscaIndex.razor

Plezališča



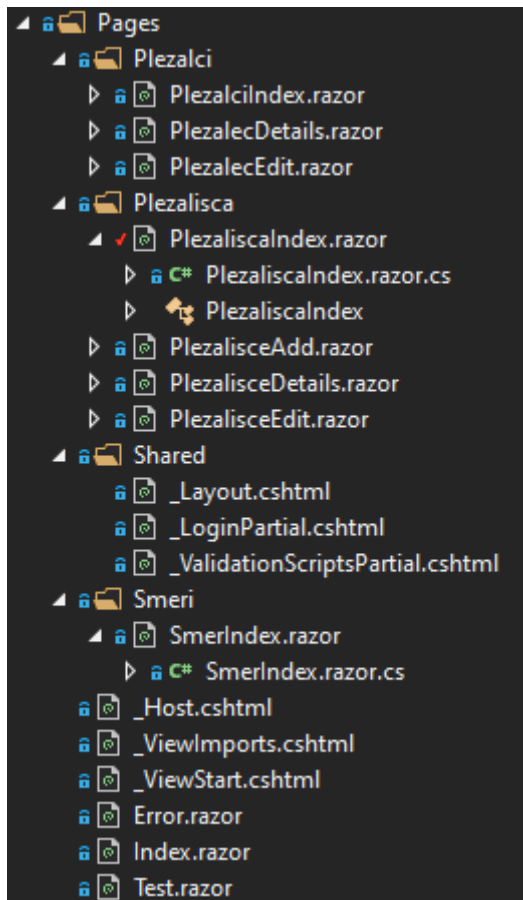
Komponenta `PlezaliscaIndex.razor` je prikazovalna komponenta, kar pomeni, da moramo v njeni glavi določiti naslov, na katerega želimo, da nas aplikacija popelje. Ta ukaz direktiva se zabeleži z znakom `@page` in za njim v narekovajih napišemo usmerjevalno pot, na katero naj nas brskalnik usmeri v primeru, da pride do zahteve po prikazu te komponente (glej sliko 24).

Slika 24: Vsebina `PlezaliscaIndex.razor` datoteke

```
PlezaliscaIndex.razor* X
1 @page "/plezalisca"
2 @inherits PlezaliscaIndexBase
3
4 <div class="container">
5   <h2 class="font-weight-bold">Plezališča</h2>
6   <hr>
7   <Iskanje OnSearch_Klik="Search_Klik" LabelText="Poišči plezališče"/>
8   <if (Plezalisca == null)>
9     {
10      
11    }
12  else
13  {
14    <div>
15      <div class="card-deck text-center m-auto">
16        <foreach (var plezalisce in Plezalisca)>
17          {
18            <PlezalisceCard Plezalisce="plezalisce" OnIzbrisiPlezalisce_Klik="OnInitializedAsync" />
19          }
20        </div>
21      </div>
22    }
23  }
24 </div>
```

Vse komponente, ki vsebujejo ta ukaz na vrhu komponente, se nahajajo v direktoriju `/Pages`. Ker je v aplikaciji kar nekaj podstrani, sem zaradi lažje organiziranosti naredil dodatne direktorije, v katere sem glede na objekte shranjeval komponente.

Slika 25: Zgradba direktorija Pages



3.3.7.2 Parameter poti

Naslednja komponenta, ki sem jo razvil, je bila komponenta `PlezalisceDetails`. Njen namen je prikazati podatke o določenem plezališču. Tudi ta komponenta bo imela direktivo `@page`, kar pomeni, da bo samostojna komponenta, ampak ker bomo imeli v aplikaciji več plezališč, moramo nekako določiti, kako bomo vedeli, katero plezališče moramo uporabniku prikazati glede na njegovo izbiro. To naredimo s pomočjo parametra poti, ki ga podamo na koncu direktive `@page »/plezalisca/{id}«`. S tem povemo, da komponenta pričakuje, da ji bomo pri zahtevku poslali tudi identifikacijski podatek `id`, ki bo odločal o tem, katero plezališče bomo uporabniku prikazali. V primeru, da pri zahtevku ne dodamo identifikacijske številke, nas brskalnik opozori, da ta naslov ne obstaja.

3.3.7.3 Ravnanje z dogodki

Znotraj komponente `PlezalisceDetails` imamo prikazan tudi primer ravnanja z dogodki. V primeru, da želimo gumb ali kakšen drug HTML element sprogramirati, da se odzove na klik z miško, mu lahko podamo ukaz `@onclick=»Nek Ukaz«` in znotraj narekovajev podamo funkcijo, ki naj se izvede. S pomočjo ravnanja z dogodki, lahko naredimo aplikacijo zelo interaktivno. Ob kliku lahko spremenimo ostale HTML elemente ali izvedemo osvežitev v naši podatkovni bazi.

3.3.7.4 Vezava podatkov

Če želimo podatke izmenjavati med HTML delom Blazor komponente in C# delom, moramo vedeti, na kakšen način se podatki lahko povezujejo med sabo. To naredimo tako, da v C# strani kode z lastnostjo predstavimo želen podatek, na HTML strani komponente pa lahko ta podatek izpišemo s pomočjo znaka @ (glej sliko 26).

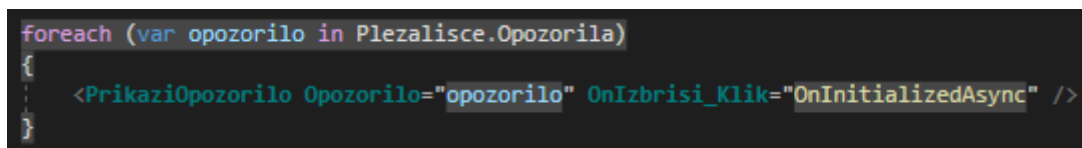
Slika 26: Primer vezave podatkov



3.3.7.5 Komponenta otrok in parametri

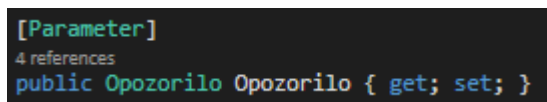
V primeru, da želimo v eni od komponent, ki vsebuje @page direktivo, prikazati nek seznam drugih elementov (v našem primeru želimo v komponenti PlezalisceDetails prikazati vse sektorje in smeri v tem plezališču), lahko znotraj te komponente pokličemo novo komponento. Tako dobimo dve komponenti, ki imata relacijo starš–otrok. V našem projektu sem vse komponente, ki nimajo @page direktive, ustvaril v Models direktoriju. Prav tako so znotraj narejeni dodatni direktoriji za lepšo organiziranost.

Slika 27: Primer klica komponente PrikaziOpozorilo



Če želimo na komponenti otrok prikazovati podatke, jih moramo na nek način posredovati. Kot lahko vidimo iz slike 27, to lahko naredimo enostavno tako, da podamo parameter h klicu komponente – v našem primeru Opozorilo. V komponenti otrok moramo nujno pred ta podatek oz. lastnost podati atribut [Parameter] (glej sliko 28).

Slika 28: Primer atributa parameter nad lastnostjo Opozorilo



Drugi način posredovanja parametrov med komponentami je s pomočjo kaskadnih parametrov. Na ta način lahko pošljemo podatek do vseh komponent, tudi če imamo kasnejše komponente še večkrat ugnezdene. Takšen primer je v naši aplikaciji v komponenti PlezalisceDetails, kjer kaskadno vrednost podamo v komponento PrikaziSmer.

Slika 29: Primer kaskadnega parametra

```
<tbody>
  @var st = 1;
  @foreach (var smer in sektor.Smeri)
  {
    <CascadingValue Value="@smer">
      <PrikaziSmer PrijavljenPlezalec="PrijavljenPlezalec" Ocene="@Ocene" Counter="@st" OnSprememba_Smer_Klik="OnInitializedAsync" />
      @st++;
    </CascadingValue>
  }
</tbody>
```

Vrednost kasneje s pomočjo atributa kličemo znotraj komponente PrikaziSmer. Dodatno pa imamo v tej komponenti še eno komponento UrediSmer, ki prav tako potrebuje podatke o smeri, tako da je preko te kaskadne komponente podatek dostopen obema komponentama.

3.3.7.6 Povratni klic dogodka

V primeru, da se kontekst komponente otrok spremeni, moramo to nekako sporočiti starševski komponenti, saj sicer za to novo stanje ne ve, posledično pa se zato vsebina v našem brskalniku ne osveži. Ta klic imenujemo povratni klic dogodka. V komponenti otrok moramo z atributom Parameter določiti EventCallback lastnost, ki jo moramo nato priklicati z ukazom InvokeAsync(), ko se stanje v komponenti otrok dejansko spremeni (glej sliko 30). Ta klic moramo nato tudi podati kot parameter v klicu komponente otrok – primer je viden na sliki 24.

Slika 30: Primer EventCallback in InvokeAsync klic ob izbrisu opozorila

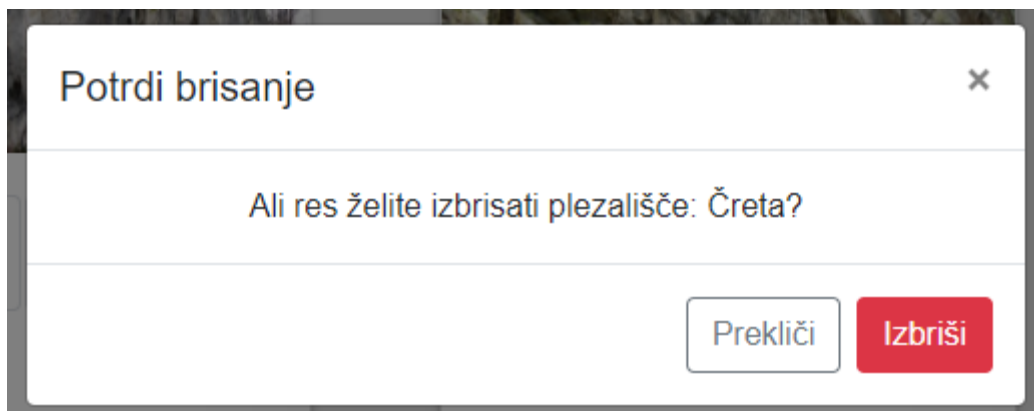
```
[Parameter]
1 reference
public EventCallback OnIzbrisi_Klik { get; set; }

1 reference
protected async Task IzbrisiOpozorilo_Klik()
{
    await DataRepository.DeleteOpozorilo(Opozorilo.Id);
    await OnIzbrisi_Klik.InvokeAsync(Opozorilo.Id);
}
```

3.3.7.7 Izdelava univerzalne komponente za večkratno uporabo

Idealen primer takšne komponente je potrditveno okno v primeru izbrisa določenega podatka iz aplikacije. V naši aplikaciji je takšnih gumbov kar nekaj, tako da bi bilo izdelovanje komponent za vsak primer posebej res odvečno. Zato sem naredil komponento, ki jo lahko uporabljam pri vseh možnostih za izbris. Takšno komponento sem naredil v posebni knjižnici za razor komponente. Odkril sem namreč hrošča, ki mi ni omogočal normalnega delovanja baze podatkov, brez da sem komponenti PotrdiComponent in ModalComponent predstavil v to knjižnico. Obe komponenti uporabljam v aplikaciji večkrat, predstavil bom samo PotrdiComponent, saj bom komponento ModalComponent razložil na drugem primeru.

Slika 31: Primer obrazca potrdi brisanje



Kot lahko vidimo iz slike 30, je sama oblika obrazca vedno enaka. Prav tako želimo vedno obdržati obliko gumbov. Ob različni uporabi komponente moramo vedno podati nov tekst v sredini obrazca. To dosežemo tako, da komponenti ob klicu dodamo parameter `PotrdiOknoText`, ki nosi vsebino sporočila. Obrazec odpremo s pomočjo direktive `@ref`, ki nam na ekranu izriše obrazec. Nato moramo s pomočjo povratnega klica dogodka starševski komponenti sporočiti, ali je uporabnik potrdil izbris določenega elementa.

Slika 32: Primer uporabe komponente `PotrdiComponent`

```
<button type="button" class="btn btn-outline-danger" onclick="IzbrišiKlik">Izbriši Plezališče</button>
<PotrdiComponent @ref="PotrdiDelete" OnPotrdiSpremenba="IzbrišiPlezalisce_Klik" PotrdiOknoText="@($"Ali res želite izbrisati plezališče: {Plezalisce.ImePlezalisca})?" />
```

3.3.7.8 Izdelava obrazcev in njihova validacija

Blazor podpira validacijo obrazcev s pomočjo imenskega prostora `DataAnnotations`. Naše lastnosti v razredih lahko opremimo z različnimi atributi, ki nam kasneje pomagajo tako pri validaciji obrazcev kot tudi pri kreiranju SQL podatkovne baze. Za lastnost lahko zahtevamo, da je obvezna, lahko jo omejimo glede na dolžino besedila, interval med dvema določenima številkama itd. Prav tako lahko s posebnimi parametri določimo izpis napake, ki se pojavi v primeru, da se ne držimo pravila pri generiranju te lastnosti.

Slika 33: Primer uporabe `DataAnnotations`

```
41 references
public class Plezalisce
{
    21 references
    public int Id { get; set; }
    [Required(ErrorMessage = "Ime plezališča je obvezen podatek.")]
    [MinLength(3)]
    24 references
    public string ImePlezalisca { get; set; }

    [Required(ErrorMessage = "Vnesi nadmorsko višino plezališča v metrih.")]
    [Range(260, 3000, ErrorMessage = "Vnesite nadmorsko višino plezališča, med 260 m - 3000 m.")]
    9 references
    public int Visina { get; set; }

    [Required(ErrorMessage = "Vnesi čas dostopa v minutah.")]
    [Range(1, 120, ErrorMessage = "Vnesite čas dostopa v minutah med 1 in 120 minut.")]
    12 references
    public int CasDostopa { get; set; }
```

Za generiranje obrazcev in tudi njihovo validacijo ima Blazor integrirane komponente za generiranje obrazcev. Trenutno so na voljo komponente `InputText`, ki se uporablja za vpisovanje kratkega besedila, `InputTextArea`, ki je primeren za vpisovanje daljšega besedila, `InputNumber` za vnos števil, `InputDate` za vnos datuma ter `InputSelect` za izbiro elementa iz seznama elementov. Naštete komponente moramo uporabljati znotraj komponente `EditForm`, ki nam pove, da se tukaj začne naš obrazec. V komponento `EditForm` moramo podati objekt, v katerega želimo vpisati podatke, in pa akcijo, ki se bo izvedla, ko obrazec uspešno izpolnimo in potrdimo vnos.

Slika 34: Primer kode za generiranje obrazca

```
<EditForm Model="NovKomentar" OnValidSubmit="() => ConfirmationChange(true)">
  <DataAnnotationsValidator />

  <InputTextArea id="komentarSmeri" class="form-control" placeholder="Tukaj vpišite novo opozorilo" @bind-Value="@NovKomentar.KomentarText" />
  <ValidationMessage For="@((())=>NovKomentar.KomentarText)" />

  <div class="modal-footer">
    <button type="button" class="btn btn-outline-secondary" data-dismiss="modal" @onclick="() => ConfirmationChange(false)">
      Prekliči
    </button>

    <button class="btn btn-primary" type="submit">
      Shrani
    </button>
  </div>
</EditForm>
```

3.3.7.9 Predloge za komponente

Izgradnjo bolj kompleksnih komponent lahko dosežemo s pomočjo predlog. Na ta način lahko dosežemo, da so komponente bolj uporabne. Za generiranje komponent s pomočjo predlog moramo uporabiti parameter `RenderFragment`. Ta nam označi del komponente, kjer bomo vnesli neko drugo vsebino znotraj že obstoječe komponente. Na ta način sem v svojem projektu napisal kodo, ki jo uporabljamo v različnih obrazcih v aplikaciji. Komponenta je mišljena za večkratno uporabo, zato sem jo podobno kot `PotrdiComponent` shranil v Razor knjižnico `VPlatiComponents`. Komponenta se imenuje `ModalComponent`. V njej imamo definiran `RenderFragment ModalBody`, s katerim določimo, kje se bo spremenila vsebina te komponente. Ta fragment moramo nato v praksi uporabiti znotraj naše komponente (glej sliko 35).

Slika 35: Primer uporabe `ModalComponent`

```
<ModalComponent ButtonStyle="btn-outline-primary" ButtonText="Komentar" ModalHeader="Vpišite Nov Komentar:" OnPotrdi="Add" OnOdprlo="Add" OnZaprlo="Close" @ref="ModalComponent">
  <ModalBody>
    <EditForm Model="NovKomentar" OnValidSubmit="() => ConfirmationChange(true)">
      <DataAnnotationsValidator />

      <InputTextArea id="komentarSmeri" class="form-control" placeholder="Tukaj vpišite novo opozorilo" @bind-Value="@NovKomentar.KomentarText" />
      <ValidationMessage For="@((())=>NovKomentar.KomentarText)" />

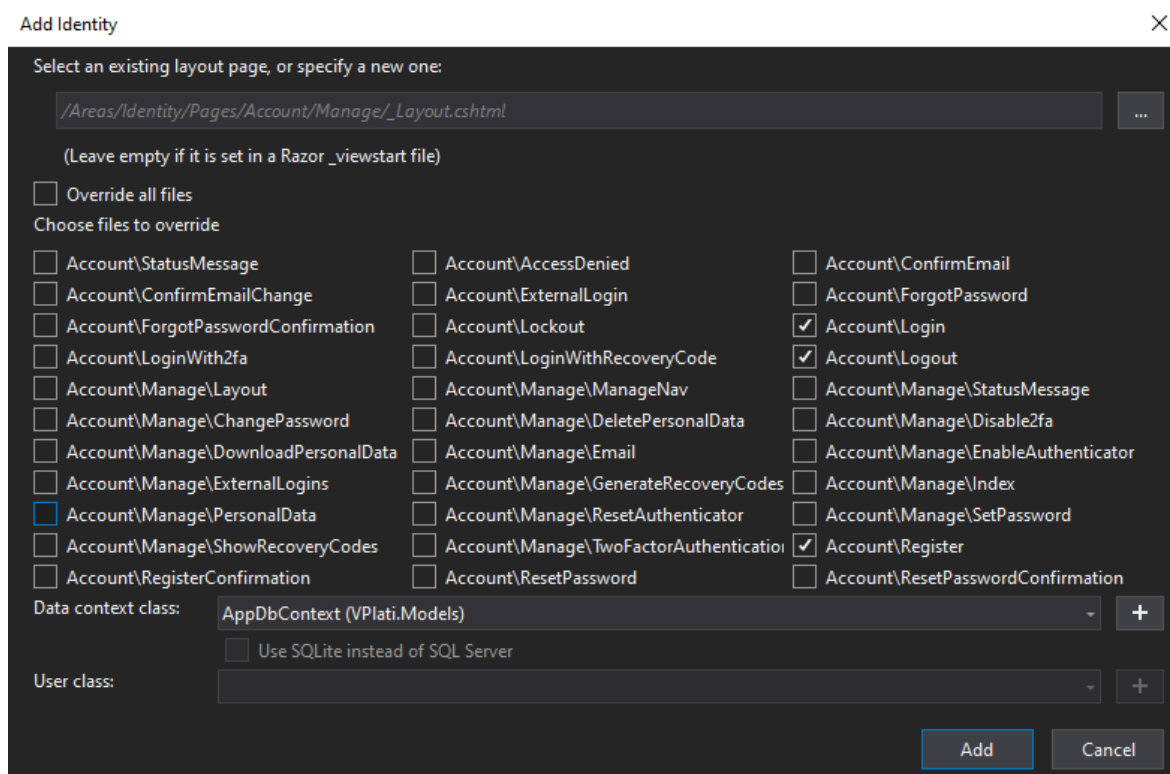
      <div class="modal-footer">
        <button type="button" class="btn btn-outline-secondary" data-dismiss="modal" @onclick="() => ConfirmationChange(false)">
          Prekliči
        </button>

        <button class="btn btn-primary" type="submit">
          Shrani
        </button>
      </div>
    </EditForm>
  </ModalBody>
</ModalComponent>
```

3.3.8 Dodajanje ASP.NET Identity ogrodja v projekt

Ker naš projekt že bazira na podatkovni bazi in ker želimo na isto podatkovno bazo vezati tudi podatke o uporabnikih, da bomo lahko kasneje v bazo dodali tudi profile plezalcev, moramo najprej popraviti razred `AppDbContext.cs`, ki mora po novem podedovati lastnosti razreda `IdentityDbContext`. Na ta način povemo, da bomo uporabljali ASP.NET Core Identity `EntityFrameworkCore`. V primeru, da ne razširimo `IdentityUser` razreda, smo omejeni samo na nekatere podatke, ki jih ta razred poseduje. S pomočjo Visual Studia 2019 dodamo v projekt nov »IdentityScaffolder«. To je neke vrste predloga, ki je že vnaprej pripravljena za vstavljanje v različne projekte. Predloga bazira na tehnologiji Razor Pages in je kompatibilna z Blazor tehnologijo, kljub temu da se logika ogrodja kar močno razlikuje.

Slika 36: Add Identity Scaffolder



Z vnosom predloge v naš projekt smo spremenili strukturo naše podatkovne baze, tako da moramo narediti novo migracijo in popravke v naši podatkovni bazi s pomočjo ukazov `Add-Migration` in `Update-Database`.

Ker želimo razred uporabiti za generiranje profilov uporabnikov portala, ga moramo razširiti. Tako ustvarimo nov razred `Plezalec`, ki podeduje lastnosti razreda `IdentityUser`. V razred dodamo lastnosti, ki jih bomo potrebovali za generiranje vseh naših funkcionalnosti. V našem primeru smo želeli dodati podatke o imenu, priimku, informacijah o plezalcu in pa datumu rojstva. V naslednjem koraku moramo popraviti vse reference na `IdentityUser` razred in ponovno posodobiti našo bazo podatkov, ker smo v tabelo `AspNetUsers` dodali nove lastnosti.

Slika 37: Razred Plezalec

```
using System.Threading.Tasks;

namespace VPlati.Models
{
    0 references
    public class Plezalec:IdentityUser
    {
        0 references
        public string Ime { get; set; }
        0 references
        public string Priimek { get; set; }
        0 references
        public string PlezalecInfo { get; set; }
        0 references
        public DateTime DatumRojstva { get; set; }
    }
}
```

Sedaj nam ostane testiranje implementiranega vmesnika. Kot smo že omenili, je predloga narejena v Razor Pages tehnologiji, tako da moramo malo prilagoditi izgled registracijskega vmesnika in pa prijavnega okna. Modifikacija je potrebna tudi, če želimo vnesti dodatne lastnosti, ki smo jih implementirali v razred Plezalec. Blazor v trenutni različici trenutno ne ponuja rešitve za upravljanje z uporabniki, je pa Microsoft že objavil, da se bo to po vsej verjetnosti spremenilo z izdajo .NET5.

Slika 38: Prijavno okno v našo aplikacijo pred modifikacijami

3.3.9 ASP.NET Identity roles – Vloge uporabnikov

Za določanje pravic na nivoju nekih skupin se lahko uporabljajo Identity roles. V aplikaciji sem nastavil, da se ob vsakem kreiranju novega uporabnika le temu avtomatsko dodeli

pravica z imenom »user«. Druga skupina uporabnikov je »admin« in jo moramo uporabnikom določiti ročno.

3.3.10 Pooblastila

Zadnji korak, ki sem ga naredil pri kreiranju aplikacije, je bil dodeljevanje pooblastil različnim skupinam uporabnikom. Blazor nam to omogoča s pomočjo komponente `AuthorizeView`.

Slika 39: Primer uporabe komponente `AuthorizeView`

```
<AuthorizeView>
    @if (PlezalecSeNiOcenilSmer(context.User.Identity.Name))
    {
        <div class="text-center">
            <OcenisMer Smer="Smer" OnVnesi="OnInitializedAsync" />
        </div>
    }
</AuthorizeView>
```

V primeru, da želimo uporabniku, ki ni pooblaščen za dostop do nekega konteksta, pokazati nekaj drugega, lahko to naredimo s pomočjo komponente `Authorized` oziroma `NotAuthorized`.

Slika 40: Primer z `Authorized` in `NotAuthorized`

```
<AuthorizeView>
    <Authorized>
        @if (PlezalecSeNiOcenilSmer(context.User.Identity.Name))
        {
            <div>
                <OcenisMer Smer="Smer" OnVnesi="OnInitializedAsync" />
            </div>
        }
    </Authorized>
    <NotAuthorized>
        Do te vsebine nimate dostopa!
    </NotAuthorized>
</AuthorizeView>
```

Omejitev določenim uporabnikom lahko postavimo tudi na nivoju celotne komponente ali pogleda razor datotek. To storimo s pomočjo atributa `Authorize`, ki ga zapišemo v vrh razor datoteke. V primeru, da določimo takšno omejitev, moramo tudi popraviti Blazor Router parameter v `App.razor` datoteki na `AuthorizeView`, v nasprotnem primeru aplikacija ne bo delovala. Določimo lahko tudi, kaj naj se uporabnikom prikaže v primeru, da skušajo dostopati do takšne podstrani in za to nimajo pravic.

Slika 41: Visual Studio 2019: Primer atributa Authorize

```
@page "/addplezalisce"
@inherits PlezalisceAddBase
@attribute [Authorize(Roles = "admin")]

<div class="container">
    <EditForm Model="@Plezalisce" OnValidSubmit="AddPlezalisce">
        <DataAnnotationsValidator />
        <h2 class="font-weight-bold">Dodaj novo plezališče:</h2>
        <hr />
        <div class="form-group row">
            <label for="imePlezalisca" class="col-sm-2 col-form-label">
                Ime plezališča:
```

V primeru, da želimo pooblastila določiti glede na različne tipe uporabnikov (v našem primeru »user« in »admin«) moramo to določiti s pomočjo parametrov, ki jih podamo atributom ali komponentam (glej sliko 41).

Poleg vseh navedenih pristopov naš program včasih tudi zanima, kateri uporabnik je trenutno povezan z aplikacijo in kakšne so njegove lastnosti. V naši aplikaciji nas recimo zanima, ali je registriran uporabnik že preplezal neko smer oz. če je za njo že podal oceno. Seveda ne želimo, da uporabnik poda več ocen za določeno smer, saj bi na tak lahko način manipuliral z oceno smeri. Do uporabnika, ki je trenutno prijavljen v podatkovno bazo, lahko dostopamo na dva različna načina. V primeru, da želimo do podatkov dostopati v HTML delu razor datoteke, lahko znotraj AuthorizeView komponente uporabimo parameter context, ki poseduje podatke o prijavljenem uporabniku, kot je razvidno s slik 39 in 40. Kadar pa želimo dostopati do uporabnika preko C# dela komponente, moramo uporabiti kaskadni parameter. Kaskadni parameter je eden od načinov transporta podatkov iz starševske komponente do otroka. S pomočjo tega parametra lahko dostopamo do stanja pooblastil v aplikaciji in nato preko UserManager naložimo podatke v naš objekt Plezalec (glej sliko 42).

Slika 42: Primer dostopa do prijavljenega uporabnika s pomočjo kaskadnega parametra

```
[CascadingParameter]
1 reference
private Task<AuthenticationState> AuthenticationStateTask { get; set; }

[Inject]
1 reference
public NavigationManager NavigationManager { get; set; }
[Inject]
1 reference
IFileReaderService FileReaderService { get; set; }
[Inject]
2 references
IWebHostEnvironment WebHostEnvironment { get; set; }

23 references
public Plezalec Plezalec { get; set; } = new Plezalec();
2 references
public ElementReference InputReference { get; set; }
2 references
public string ImageContent { get; set; } = string.Empty;
3 references
public string Notification { get; set; } = "SkrijElement";

15 references
protected async override Task OnInitializedAsync()
{
    var authState = await AuthenticationStateTask;
    var currentUser = authState.User;
    Plezalec = await UserManager.GetUserAsync(currentUser);
}
```

4 NAVODILA ZA UPORABO APLIKACIJE VPLATI

4.1 ANONIMNI UPORABNIK

Aplikacija je namenjena trem različnim tipom uporabnikov. V primeru, da oseba ni registriran uporabnik, ima omejen dostop do interakcije s samo aplikacijo. Omogočeni so vsi pogledi, nima pa možnosti komentiranja, izdajanja opozoril in ocenjevanja določene smeri.

4.2 REGISTRIRAN UPORABNIK V SKUPINI »USER«

Vsak uporabnik ima možnost, da se v aplikacijo registrira. Do registracijskega obrazca se dostopa preko zavihka, ki se nahaja v zgornjem navigacijskem meniju. Večina registracijskih podatkov je obvezna, kar nas aplikacija tudi opozori v primeru, da določene podatke v obrazcu izpustimo.

Slika 43: Primer registracijskega obrazca

Registracija

Ustvari nov uporabniški račun

Ime

Ime je obvezen podatek.

Priimek

Priimek je obvezen podatek.

Uporabniško ime

Uporabniško ime je obvezen podatek.

Email

Email je obvezen podatek.

V primeru uspešne registracije se avtomatsko prijavimo v aplikacijo, kjer se nam pokaže tudi prva stran. V navigacijskem meniju, kjer se je nahajala registracija, imamo sedaj izpisana pozdrav in pa naše uporabniško ime. Aplikacija nam seveda omogoča tudi odjavo.

Na prvi strani aplikacije lahko vidimo zadnjih 5 ocenjenih smeri in pa zadnjih 5 izdanih opozoril v plezališčih.

Slika 44: Prva stran aplikacije

VPlati

Prva stran

Plezališča

Plezalci

Pozdravljen, kekec

Odjava

Dobrodošli v aplikaciji VPlati

Stran je namenjena vsem plezalcem, ljubiteljem vertikalne in na splošno vsem privržencem športov, ki se odvijajo v naravi. V naši aplikaciji se nahajajo podatki o slovenskih plezališčih in registriranih uporabnikih, ki svoje podvige veselo vpisujejo v aplikacijo. Cilj aplikacije je olajšati plezalcem pot v naravno plezališča, jih opozarjati o nevarnostih in pomagati z nasveti v težkih smereh. Količina podatkov v aplikaciji je omejena, ampak se bo z večjim številom uporabnikov tudi baza podatkov pridno povečevala.

Za dostop do naprednih storitev, kot so komentiranje in ocenjevanje smeri, ali pa izdajanje opozoril in pa označevanje preplezanih strani, se je potrebno registrirati. V primeru, da ste postavljalci smeri ali pa da želite pomagati pri povečevanju baze podatkov o plezališčih nam pišite na e-mail naslov info@vplati.si in z veseljem vam bomo nadgradili vaš uporabniški račun v administratorskega, s katerim boste lahko dodajali v aplikacijo nova plezališča, nove sektorje in nove smeri.

Zadnjih 5 ocenjenih smeri

#	Plezališče	Smer	Plezalec	Ocena	Nacin	Dolžina	Datum
1	Čreta	test12	kekec	5c	Na Pogled	23 m	11/08/2020
2	Čreta	test	kekec	4c	Na Pogled	22 m	11/08/2020
3	Čreta	test12	w4rcic	4a	Na Pogled	23 m	10/08/2020
4	Čreta	test	w4rcic	5c	Na Pogled	22 m	10/08/2020
5	Čreta	test	test2	5c	Flash	22 m	09/08/2020

Zadnjih 5 izdanih opozoril

#	Plezališče	Opozorilo	Datum
1	Čreta	Pazi kače v sektorju A. Plezaj na lastno odgovornost!	10/08/2020

V primeru, da kliknemo na pozdrav oz. naše uporabniško ime, se nam odpre podstran, kjer lahko popravimo svoje uporabniške podatke. Prav tako lahko v aplikacijo dodamo svojo sliko.

Slika 45: Primer podstrani za urejanje profila uporabnika

V levem navigacijskem meniju imamo možnosti izbire med tremi podmeniji. Poleg prve strani imamo še Plezališča in Plezalce. V primeru, da odpremo zavihek plezališča, se nam prikažejo vsa plezališča, ki se nahajajo v naši podatkovni bazi. Tukaj lahko plezališče poiščemo s pomočjo iskalnika, lahko pa uredimo vrstni red plezališč po različnih parametrih.

Slika 46: Prikaz plezališč

S klikom na sliko od plezališč se nam prikaže detajlni pogled plezališča. V tem pogledu imamo sedaj prvo interakcijo, ki anonimnim uporabnikom ni omogočena, in sicer izdajo ter brisanje opozorila za plezališče. Pod opozorilom imamo gumb Dodaj Novo Opozorilo, ki odpre obrazec, preko katerega lahko dodamo informacije o opozorilu. V primeru, da

opozorilo več ni aktualno, lahko preko ikone, ki prikazuje koš za smeti, to opozorilo tudi izbrišemo. Aplikacija nas pred izbrisom nekega podatka vedno dodatno opozori, če želimo izbris res narediti. Spodaj imamo podatke o plezališčih in številih smeri glede na težavnosti. Potem pa imamo izpisane sektorje plezališč po tabelah. Znotraj vsake tabele se nahajajo smeri v plezališču.

Slika 47: Primer detaljne podstrani za posamezno plezališče

Opozorila:

Ni aktivnih opozoril.

Opis dostopa

Zelo lepo plezališče v dolini trente. V bližini se tudi nahaja več balvanskih problemov. Sparkirajte za galerijo na parkirišču, nato se podate čez mostiček in po prelepem travniku pridete do ogromnega balvana, kjer se nahaja plezališče. Ob plezališču nas vedno pričaka zelo prijazen domačin Rajko, koji ima zelo rad korenje.

Težavnost smeri

4a ... 4c	5a ... 5c	6a ... 6c+	7a ... 7c+	8a ... 8c+	9a ... 9c
3	1	1	4	0	0

Sektorji plezališča

A	Ime smeri	Težavnost	Dolžina
1	Smrekca	4c	9 m
2	Muc Mac	5a	9 m
3	Tire e tas	6a+	12 m
4	Ploha	5b	9 m

Vsaka vrstica tabele, vključno z imenom smeri, deluje kot povezava na podstran detaljni pogled smer. Na tej podstrani imamo najprej osnovne podatke o smeri, pod njo pa (v primeru, da še nismo ocenili smeri) možnost ocenjevanja smeri. V primeru, da kliknemo na gumb Oцени Smer, se nam odpre obrazec, kjer izberemo oceno in pa način, na katerega smo smer preplezali. V primeru, da smo smer že ocenili, tega ne moremo ponoviti. Pod gumbom Oцени Smer se nato nahaja tabela, kjer so zabeležene vse ocene, ki so jih podali uporabniki. Težavnost smeri se izračuna glede na povprečno vrednost vseh ocen in je interaktivna.

Slika 48: Primer detaljne podstrani smeri

Hrastov list

Osnovni podatki:

Težavnost	5a
Dolžina Smeri	15 m
Zabeleženih Vzponov	1
Opremljevalec	Vili Guček
Leto postavitve	2013

Oceni Smer

Ocene:

#	Ocena	Plezalec	Način	Datum Vnosa
1	5a	w4rcic	Na Pogled	15/08/2020

Komentarji:

Hrastov list nima komentarja


Nazaj

Komentar


Na dnu strani imamo še možnost komentiranja posamezne smeri, kjer lahko ostalim uporabnikom sporočimo, kako nas je določena smer v plezališču navdušila ali pa razočarala. V primeru, da smo komentar napisali, imamo tudi možnost, da ga pobrišemo.

Slika 49: Primer komentarjev

Komentarji:



w4rcic Objavljen dne: 13/08/2020
dfgsh dfh g



test Objavljen dne: 09/08/2020
To je noc komentar

Nazaj

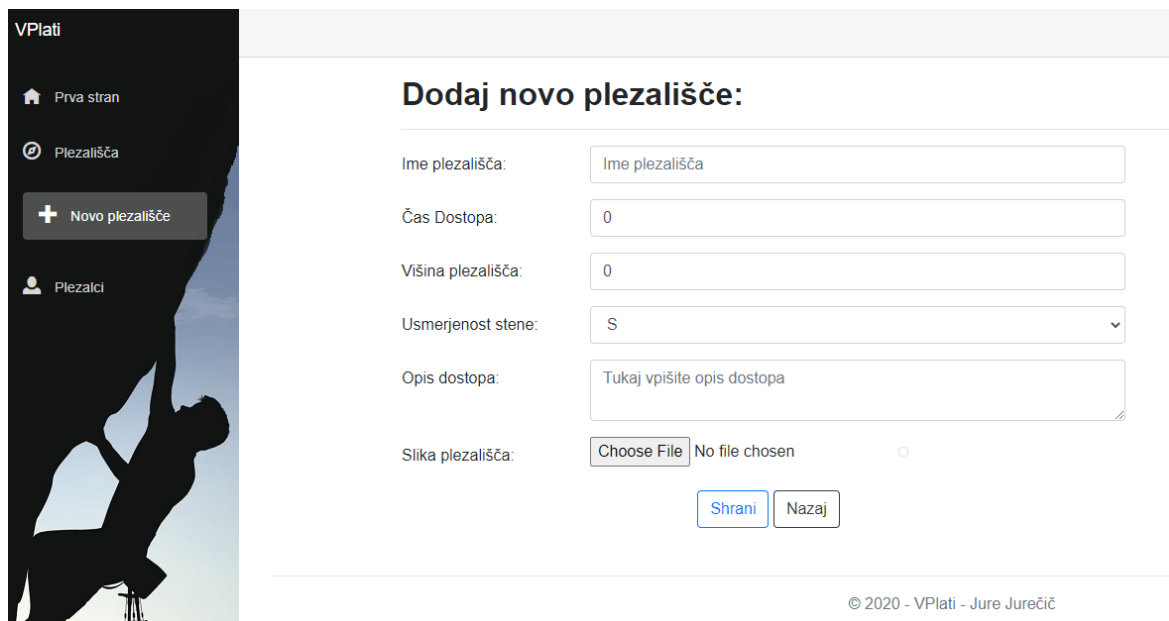
Komentar

V aplikaciji lahko dostopamo tudi do določenih podatkov ostalih uporabnikov aplikacije. V primeru, da v navigacijskem meniju levo izberemo možnost Plezalci, se nam odprejo vsi registrirani uporabniki. V primeru, da kliknemo na kartico, se nam odpre detaljna podstran plezalec, kjer imamo določene podatke o plezalcu (njegove dosežke, zadnjih pet ocen itd.).

4.3 REGISTRIRAN UPORABNIK V SKUPINI »ADMIN«

Uporabniki v skupini »admin« poleg vseh pooblastil, ki jih imajo navadni registrirani uporabniki, posedujejo še dodatna pooblastila, s katerimi lahko vplivajo na bazo naših podatkov. V navigacijskem meniju levo imajo dodatno možnost in sicer izbiro Dodaj novo plezališče.

Slika 50: Dodaj novo plezališče



VPlati

Prva stran

Plezališča

+ Novo plezališče

Plezalci

Dodaj novo plezališče:

Ime plezališča:

Čas Dostopa:

Višina plezališča:

Usmerjenost stene:

Opis dostopa:

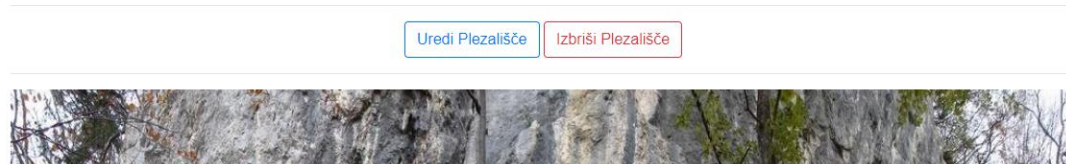
Slika plezališča: No file chosen ☐

© 2020 - VPlati - Jure Jurečič

Administratorji aplikacije torej lahko dodajo vsebine plezališč. Admin lahko med drugim tudi ureja podatke o plezališči in smeri. Pod imenom plezališča se nam izpišeta tudi možnosti uredi in izbriši. Prav tako pa imamo v detajlnem pogledu smeri ter v seznamu vseh smeri v detajlnem pogledu plezališča možnost urejanja podatkov o smeri.

Slika 51: Uredi / Izbriši plezališče

Čreta



V urejevalniku podatkov o plezališču lahko dodajamo in brišemo nove sektorje. Nov sektor lahko dodamo tudi v detajlnem pogledu plezališče. V detajlnem pogledu smeri pa lahko uredimo osnovne podatke o smeri, prav tako pa lahko poleg svojih brišemo tudi neprimerne komentarje navadnih registriranih uporabnikov.

Slika 52: Obrazec za dodajanje nove smeri

Opis dostopa

Plezališče nima podanega opisa plezališča.

Vnesite podatke o novi smeri:

Ime nove smeri:

Ocena smeri:

Ocena smeri:

Dolžina smeri:

Opremljevalec:

Datum Prvega Vzpona:

Komentar:

Prekliči Shrani

Dodaj Novo Smer

5 SKLEPI

Izdelava aplikacije se je izkazala za kar obsežen projekt. Od same ideje do planiranja in nato cele izgradnje projekta je bilo kar veliko dela. Najtežji del pri celem projektu je bilo razumevanje novih konceptov v ogrodju Blazor. Velikokrat sem se tudi znašel v slepi ulici, brez rešitev za razvoj določenih modulov, ki jih nisem želel implementirati s pomočjo programskega jezika JavaScript. Na koncu mi je uspelo tudi s pomočjo navdušencev nad novo tehnologijo Blazor, ki so zelo aktivni na različnih spletnih kanalih.

Aplikacija še ni čisto pripravljena na produkcijsko okolje, definitivno bi moral razširiti funkcionalnosti in dodati nekaj malega tudi na vizualnem aspektu aplikacije. Imam dodatne ideje, kako bi lahko aplikacijo nadgradil z zemljevidi, prav tako bi v primeru velike količine podatkov potreboval možnost prikazovanja iskanih zadetkov na podstraneh, implementacijo določenih filtrov pri iskanju plezališč itd. Mislim, da bi se z vsemi temi novimi moduli in obsegom aplikacije odmaknil od samega koncepta z delom Blazor ogrodja.

Samo ogrodje mi je po tem, ko sem začel koncept razumevati, postalo zelo všeč. Moram pa dodati, da sem opazil tudi kar nekaj pomanjkljivosti. Določeni moduli, ki se uporabljajo za izgradnjo podobnih aplikacij, v Blazorju še niso podprta. Eden od takšnih primerov je recimo

dodajanje slike v obrazec. To sem moral implementirati ročno preko modula, ki ga je razvil eden od uporabnikov Blazor ogrodja, kar mi je vzelo zelo veliko časa, ker obstoječa verzija več ni bila podprta z zadnjo verzijo, ki sem jo uporabljal v svojem projektu. Drug takšen primer je tudi upravljanje z uporabniškimi profili, ki se nanaša na tehnologijo Razor Pages in se močno razlikuje od Blazor ogrodja.

Menim, da je Microsoft vse moči razvoja usmeril v izdajo .NET 5.0 verzije, ki naj bi izšla novembra 2020. Za to verzijo napovedujejo, da bodo rešili določene težave, ki jih ima Blazor ogrodje, tako da se je že veselim in upam, da bo Blazor postal še boljše orodje za izdelavo spletnih aplikacij in dobra alternativa ogrodjem, ki so zgrajena na JavaScript programskem jeziku.

6 VIRI

- ASP.NET Core – Dostopno 12. julij, 2020, <https://aspnetcore.readthedocs.io/en/stable/intro.html>
- ASP.NET Core – Identity setup – Dostopno 4. avgust, 2020, <https://www.pragimtech.com/blog/blazor/asp.net-core-identity-setup-in-blazor-application/>
- ASP.NET Core Blazor authentication and authorization – Dostopno 13. avgust, 2020, <https://docs.microsoft.com/en-us/aspnet/core/blazor/security/?view=aspnetcore-3.1>
- Blazor – Dostopno 12. julij, 2020, <https://devblogs.microsoft.com/aspnet/blazor-now-in-official-preview/>
- Blazor component parameters – Dostopno 11. avgust, 2020, <https://www.pragimtech.com/blog/blazor/blazor-component-parameters/>
- Blazor data access – Dostopno 16. julij, 2020, <https://www.pragimtech.com/blog/blazor/blazor-data-access-strategies/>
- Blazor data binding – Dostopno 11. avgust, 2020, <https://www.pragimtech.com/blog/blazor/blazor-data-binding/>
- Blazor EventCallback – Dostopno, 11. avgust, 2020, <https://www.pragimtech.com/blog/blazor/blazor-eventcallback/>
- Blazor event handling – Dostopno, 11. avgust, 2020, <https://www.pragimtech.com/blog/blazor/blazor-event-handling/>
- Blazor forms and validation – Dostopno, 13. avgust, 2020, <https://docs.microsoft.com/en-us/aspnet/core/blazor/forms-validation?view=aspnetcore-3.1>
- Blazor route parameters – Dostopno, 11. avgust, 2020, <https://www.pragimtech.com/blog/blazor/blazor-route-parameters/>
- Blazor templated components – Dostopno, 11. avgust, 2020, <https://docs.microsoft.com/en-us/aspnet/core/blazor/components/templated-components?view=aspnetcore-3.1>
- C# – Dostopno 12. julij, 2020, [https://en.wikipedia.org/wiki/C_Sharp_\(programming_language\)](https://en.wikipedia.org/wiki/C_Sharp_(programming_language))
- CSS – Dostopno 12. Julij, 2020, <https://sl.wikipedia.org/wiki/CSS>
- DataAnnotations in Depth – Dostopno, 13. avgust, 2020, <https://www.c-sharpcorner.com/article/dataannotations-in-depth/>
- Entity Framework Core – Dostopno 11. avgust, 2020, <https://docs.microsoft.com/en-us/ef/core/>
- HMTL – Dostopno 12. julij, 2020, <https://en.wikipedia.org/wiki/HTML>

- Introducing .NET5, Dostopno 13. avgust, 2020, <https://devblogs.microsoft.com/dotnet/introducing-net-5/>
- Introduction to Identity on ASP.NET core – Dostopno 11. avgust, 2020, <https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-3.1&tabs=visual-studio>
- Repository Pattern – Dostopno 16. julij, 2020, <https://www.pragimtech.com/blog/blazor/rest-api-repository-pattern/>
- RESTful API – Dostopno 16. julij, 2020, <https://www.pragimtech.com/blog/blazor/what-are-restful-apis/>
- WebAssembly – Dostopno 12. julij, 2020, <https://en.wikipedia.org/wiki/WebAssembly>
- WebAssembly – Dostopno 12. julij, 2020, <https://webassembly.org/>