SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Seminar

**Sustav za upravljanje životnim ciljevima**

u okviru predmeta Objektno oblikovanje

Grupa: <[o.o]>

Tijan Tomislav Radoš

Mislav Kučanda

Ivan Rittinger

Zagreb, siječanj 2017

Sadržaj

[1. Opis projektnog zadatka 1](#_Toc473713574)

[2. Opis obrazaca uporabe 3](#_Toc473713575)

[2.1. Aktori sustava 3](#_Toc473713576)

[2.2. Obrasci uporabe 4](#_Toc473713577)

[3. Arhitektura i dizajn sustava 8](#_Toc473713578)

[3.1. Svrha, opći prioriteti i skica sustava 8](#_Toc473713579)

[3.2. Poslužitelj 10](#_Toc473713580)

[4. Opis Web aplikacije 11](#_Toc473713581)

[5. Mobilna aplikacija 15](#_Toc473713582)

[6. Class dijagram 20](#_Toc473713583)

[7. Specifikacija O/R mapiranja aplikacije 21](#_Toc473713584)

# Opis projektnog zadatka

Primarni cilj projekta „*The Bucket List*“ je, kao što i sam naziv kaže, kreiranje virtualne *Bucket-liste*, odnosno liste vlastitih želja i nedovršenih postignuća kojima osoba teži te ih želi ostvariti do kraja svog života. Međutim, kako je svatko individua, te su svačije želje individualne, tako i mi dajemo korisnicima mogućnost praćenja tuđih želja i ciljeva radi motivacije i inspiracije za uspješno dovršavanje pojedinih, eventualnih opisivanja iskustva, dodavanja slika i slično. Motivacija stoji u tome da svatko u svom privatnom životu ima barem nekoliko stvari koje zasigurno želi postići dok može, te iskoristiti svaki trenutak u životu da načini sebe sretnijim.

Životni ciljevi su grupirani unutar liste želja (*Bucket List*) pri čemu se svaki životni cilj može nalaziti u jednom od dva stanja: 'u izvođenju' i 'ostvaren'. Korisnici mogu o svakom životnom cilju voditi blog, na način da opisuju kako su krenuli, kako su napredovali, te na posljetku kako su ga i ostvarili . Primjerice, korisniku kojemu je životni cilj smršaviti određen broj kilograma može pisati o svojim iskustvima vezanim za tjelovježbu, način prehrane i dr. Na taj način korisnik opisuje svoj napredak te ostaje motiviran sve do ostvarenja svojeg cilja.

Sustav omogućuje međusobno povezivanje ljudi s istim životnim ciljevima kako bi mogli međusobno dijeliti iskustva i biti si podrška. Tako korisnik može pratiti napredak ljudi sa sličnim ciljevima što ga motivira za ostvarenje vlastitih. Ideja sustava je i pružiti inspiraciju, motivaciju, te pozitivnu energiju, te navesti korisnike da ostanu aktivni tako da vide što raznovrsniji izbor tuđih životnih ciljeva i na taj način otkriju lude i genijalne ideje za sebe. Ako korisnika primjerice zanima kamo otići na medeni mjesec, može pronaći inspiraciju u tuđim životnim ciljevima iste kategorije, što će mu dati mnoštvo zanimljivih ideja i korisnih savjeta.

Svaki korisnik ima svoj korisnički profil koji se sastoji od korisničkog imena, lozinke, profilne slike, životnog moto-a i kratkog opisa. Korisnik može kreirati životne ciljeve koji spadaju u pojedinu kategoriju i koji se sastoje od naziva, slike, stanja ('u izvođenju' i 'ostvaren') i opisa. Za svaki životni cilj, korisnik može mijenjati sliku te svaku komponentu u što spada i opis te na taj način može održavati blog o tome kako je tijek izvođenja izgledao (popraćen datumima). Na koncu, korisnik može postaviti određeni cilj kao „ostvaren“. Korisnik može pratiti druge korisnike na svojoj naslovnici te dodati njihove ciljeve na listu svojih, što je idealno za one koji možda nemaju mnogo vlastitih ideja, ali su lako motivirani tuđim. Ako korisnik nije prijavljen u sustav, on nema pristup pregledavanju tuđih ideja, ali se može registrirati i krenuti u akciju.

Implementirane su mobilna i web aplikacija nad istim skupom podataka, odnosno nad istom bazom podataka (*MsSQL*) te su omogućene iste funkcionalnosti. Za implementaciju *MVC* web aplikacije korišten je programski jezik *C#* te razvojna okolina *Visual Studio 2015*. Aplikacije koriste zajednički sloj za pristup podacima pri čemu je za O/R mapiranje korišten *Fluent NHibernate*

# Opis obrazaca uporabe

## Aktori sustava

*Neregistrirani korisnik*, inicijator

* korisnik koji nije odradio registraciju u sustav
* ne može pregledavati životne ciljeve
* ne može pratiti ostale korisnike
* može se registrirati

*Registrirani korisnik*, inicijator

* korisnik koji je odradio registraciju u sustav
* može se prijaviti u sustav

*Prijavljeni korisnik*, inicijator

* korisnik koji je registriran i prijavljen u sustav
* može pratiti druge korisnike
* može kreirati životne ciljeve i mijenjati podatke o istim
* može dodavati tuđe životne ciljeve na svoju listu
* može mijenjati svoje korisničke podatke

*Baza podataka*, sudionik

* sprema podatke o svim korisnicima, životnim ciljevima, člancima i komentarima

*Poslužitelj*, sudionik

* pruža uslugu te obrađuje zahtjeve
* omogućuje pristup bazi podataka na zahtjev korisnika
* pruža web usluge

## Obrasci uporabe

* UC1 – *RegistracijaKorisnika*
* **glavni sudionik:** neregistrirani korisnik
* **cilj:** uspješna registracija u sustav preko obrasca
* **sudionici:** poslužitelj, baza podataka
* **rezultat:** korisnik je prijavljen u sustav, može upravljati životnim ciljevima te pratiti druge korisnike
* **željeni scenarij:**
  1. neregistrirani korisnik odabire poveznicu za registraciju i ispunjava web/mobilni-obrazac sa svojim podacima
  2. poslužitelj provjerava ispravnost unesenih podataka te postoji li već u bazi podataka korisnik sa istim podacima
  3. korisnik se uspješno registrirao u sustav
* **mogući drugi scenarij:** korisnik se nije registrirao u sustav zbog neispravno unesenih podataka ili zbog postojećeg korisnika u bazi podataka sa istim podacima
* UC2 – *PrijavaKorisnika*
* **glavni sudionik:** registrirani korisnik
* **cilj:** uspješna prijava u sustav
* **sudionici:** poslužitelj, baza podataka
* **preduvjet:** glavni sudionik je registriran u sustavu
* **rezultat:** korisnik je prijavljen u sustav
* **željeni scenarij:**
  1. neprijavljeni korisnik odabire poveznicu za prijavu te unosi korisničko ime i lozinku u za to namijenjena polja
  2. poslužitelj provjerava ispravnost unesenih podataka
  3. korisnik se uspješno prijavio u sustav
* **mogući drugi scenarij:** korisnik se nije uspio prijaviti u sustav zbog neispravno unesenih podataka
* UC3 – *KreiranjeŽivotnogCilja*
* **glavni sudionik:** prijavljeni korisnik
* **cilj:** uspješno kreiranje životnog cilja
* **sudionici:** poslužitelj, baza podataka
* **preduvjet:** glavni sudionik je prijavljen u sustav
* **rezultat:** životni cilj je uspješno unesen u bazu podataka te se pojavljuje na korisnikovom zidu koji predstavlja njegovu „*Bucket Listu*“-u
* **željeni scenarij:**
  1. korisnik odabire poveznicu za kreiranje životnog cilja te odabire naziv, opis, sliku i stanje životnog cilja
  2. poslužitelj unosi podatke u bazu podataka
  3. unos životnog cilja je uspješno obavljen, prikazuje se u aplikaciji na korisnikovom profilu te naslovnici svih ostalih
* **mogući drugi scenarij:** kreiranje životnog cilja nije uspjelo zbog nepopunjavanja obaveznih podataka te je potrebno ponoviti postupak
* UC4 – *DodavanjeŽivotnogCilja*
* **glavni sudionik:** prijavljeni korisnik
* **cilj:** uspješno dodavanje tuđeg životnog cilja na svoju listu
* **sudionici:** poslužitelj, baza podataka
* **preduvjet:** glavni sudionik je prijavljen u sustav
* **rezultat:** cilj je uspješno unesen u bazu podataka te se pojavljuje na zidu korisnika te na naslovnici svim korisnicima
* **željeni scenarij:**
  1. korisnik odabire gumb za dodavanje tuđeg životnog cilja na listu svojih
  2. poslužitelj unosi podatke u bazu podataka
  3. unos životnog cilja je uspješno provedeno, prikazuje se u aplikaciji na korisnikovom profilu
* **mogući drugi scenarij:** baza podataka trenutno nije u mogućnosti pružiti uslugu te je potrebno ponoviti postupak
* UC5 – *AžuriranjeŽivotnogCilja*
* **glavni sudionik:** prijavljeni korisnik
* **cilj:** uspješno uređivanje nekog životnog cilja
* **sudionici:** poslužitelj, baza podataka
* **preduvjet:** glavni sudionik je prijavljen u sustav te životni cilj postoji
* **rezultat:** životni cilj je uspješno ažuriran
* **željeni scenarij:**
  1. korisnik odabire životni cilj koji želi ažurirati, unosi promjene (primjerice, dodaje novi datum i korak u izvršavanju tog cilja ili općenito prilagođava tuđi dodani cilj sebi)
  2. poslužitelj ažurira podatke u bazi podataka
  3. promijene se pojavljuje zajedno sa pripadajućim životnim ciljem
* **mogući drugi scenarij:** ažuriranje podataka nije uspjelo
* UC6 – *IzvršavanjeŽivotnogCilja*
* **glavni sudionik:** prijavljeni korisnik
* **cilj:** uspješno postavio status nekog životnog cilja na „izvršen“
* **sudionici:** poslužitelj, baza podataka
* **preduvjet:** glavni sudionik je prijavljen u sustav te životni cilj postoji
* **rezultat:** životni cilj je uspješno izvršen
* **željeni scenarij:**
  1. korisnik odabire životni cilj kojem želi promijeniti status te odabire opciju „Postavi izvršeno“ ili „Ostvari“
  2. poslužitelj ažurira podatke u bazi podataka
  3. promijene se pojavljuje zajedno sa pripadajućim životnim ciljem
* **mogući drugi scenarij:** ažuriranje podataka nije uspjelo
* UC7 – *IzmjenaKorisničkihPodataka*
* **glavni sudionik:** prijavljeni korisnik
* **cilj:** promjena vlastitih korisničkih podataka
* **sudionici:** poslužitelj, baza podataka
* **preduvjet:** glavni sudionik je prijavljen u sustav te je odabrao opciju za promjenu podataka
* **rezultat:** korisnik je promijenio svoje korisničke podatke
* **željeni scenarij:**
  1. glavni sudionik odabire gumb za promjenu vlastitih podataka
  2. poslužitelj u bazu podataka unosi promijene
  3. prikazuje se obavijest o uspješno provedenim promjenama
* **mogući drugi scenarij:** ažuriranje podataka nije uspjelo

# Arhitektura i dizajn sustava

## Svrha, opći prioriteti i skica sustava

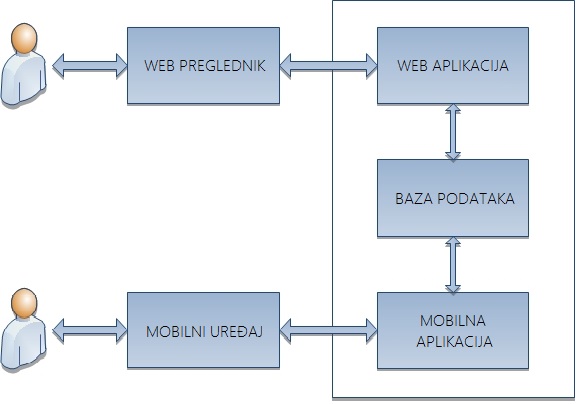
Dobar izbor arhitekture ključan je za uspješan projekt jer nam omogućuje najbolje zadovoljavanje traženih zahtjeva sustava. Upravo zbog toga, razmatranje različitih arhitektura te njihovih pozitivnih i negativnih strana, važan je korak kod oblikovanja sustava kako bi se postigla njegova fleksibilnost, mogućnost nadogradnje i jeftino održavanje.

S obzirom da je naš projekt usko vezan uz Internet jer sustav mora biti što dostupniji korisnicima u svakom trenutku, odlučili smo implementirati sustav u obliku web i mobilne aplikacije. Arhitektura web aplikacije će biti objektno-orijentirana MVC aplikacija. Razlog odabira objektno-orijentiranog pristupa je mogućnost jednostavnog modeliranja, jednostavnost održavanja i nadogradnje sustava te odvajanja sustava na razrede kako bi postigli višu razinu apstrakcije i preglednosti.

Kao dio Web aplikacije, pružamo uslugu i mobilnoj aplikaciji pomoću Web API 2 kontrolera. Na taj način koristimo isti backend dio za mobilnu i web aplikaciju te sustav ostaje konzistentan i perzistentan.

Arhitektura opisanog sustava sastoji se od sljedećih podsustava:

* Web aplikacija (+ Web API)
* Mobilna aplikacija
* Baza podataka



Slika 1 - arhitektura aplikacije

## Poslužitelj

Cjelokupan sustav se nalazi na poslužitelju, te mu se može pristupiti pozivom na link:

<http://ec2-34-249-76-120.eu-west-1.compute.amazonaws.com/>

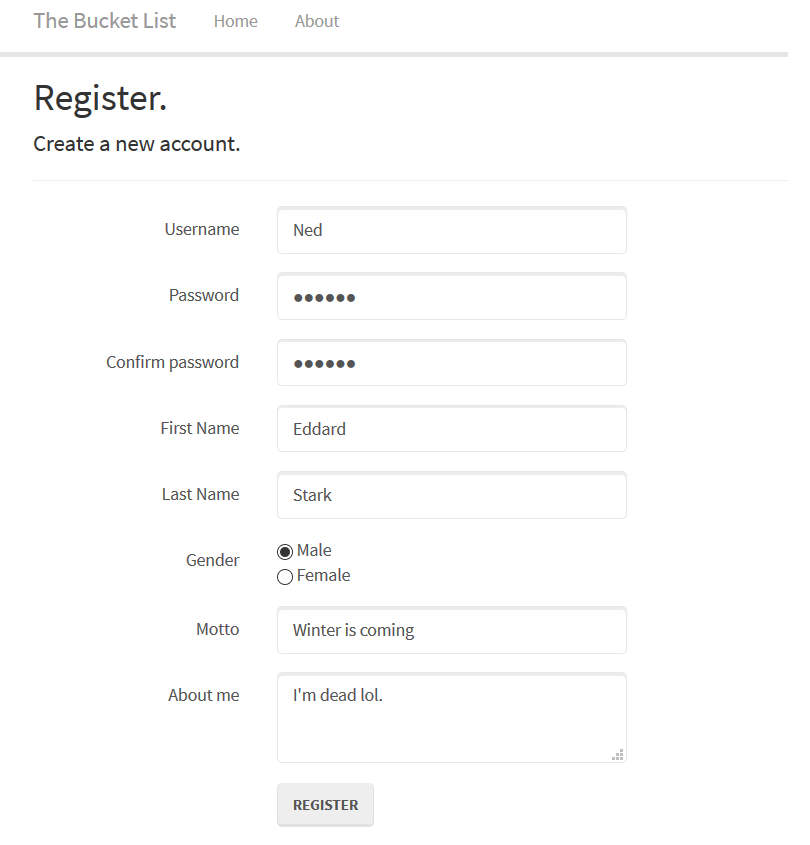
Na taj način pružamo uslugu konstantno te web i mobilna aplikacija funkcioniraju uredno.

Ukoliko je potrebno pristupiti web aplikaciji, pozivom na navedeni URL u bilo kojem pretraživaču se može ostvariti željena akcija. Sustav je aktivan te MVC radi kontinuirano po uzoru na navedeno u predavanjima.

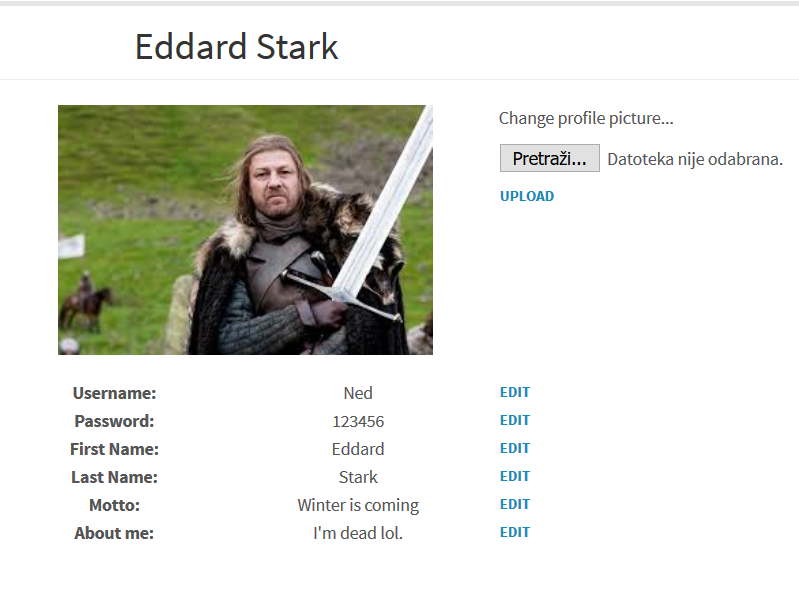
Baza podataka se također nalazi na zasebnom poslužitelju te njoj pristupaju funkcije konfiguracije FluentNHibernate. Također, baza podataka je izvedena unutar sučelja MS SQL Server.

# Opis Web aplikacije

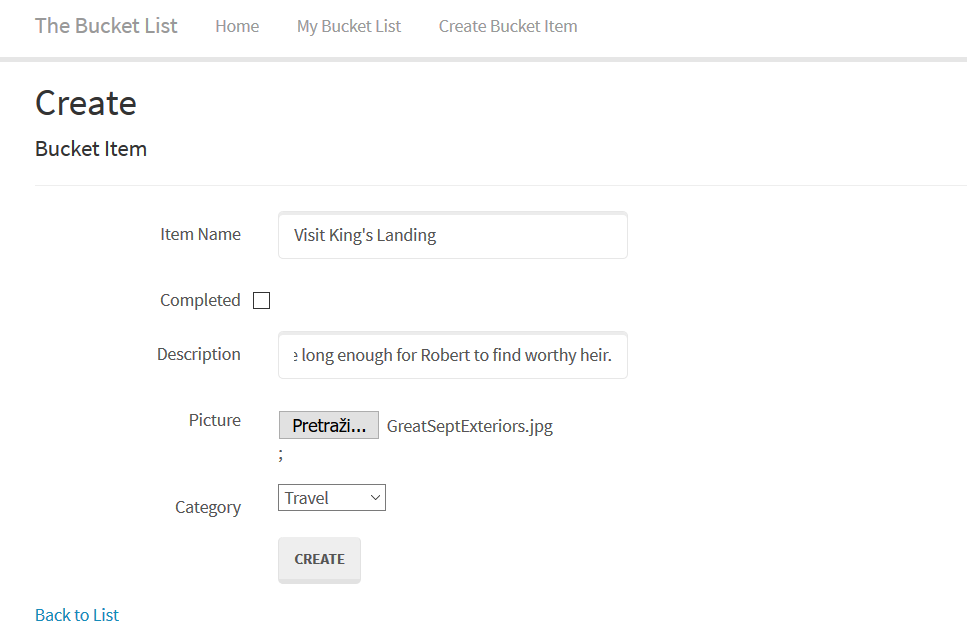
Web aplikacija je kronološki prva razvijena i to po standardnom ASP.NET MVC obrascu. Modeli domene služe kao proširenje osnovnih entiteta koje koristi Nhibernate. Osim modela domene napravljeno je i nekoliko view-modela koji služe za agregiranje nekih entiteta i mogućnosti koje je potrebno zajedno posluživati u view. Kontroleri postoje u dvije kategorije: MVC kontroleri i web API kontroleri. Prvi služe za web aplikaciju a drugi za mobilnu aplikaciju. Kontroleri upravljaju osnovnim funkcionalnostima i služe kao međukorak između modela domene i korisničkog sučelja. Ostvaruju osnovne CRUD metode i neke „manje“ metode za koje je zbog malih jedinica posla bilo smatrano da je bolje da se izvode pojedinačno. Kontroleri su ti koji koriste Nhibernate-ov session kako bi komunicirali sa bazom prilikom obrade podataka. Kao rezultate MVC kontroleri vraćaju ActionResult tipa View() ili RedirectToAction(). View-ovi u MVC-u su oblika cshtml i funkcionalno su jednostavni. Oni primaju viewmodele i rade manje obrade primljenih podataka () prije nego ih prikažu korisniku. Također oni i primaju podatke od korisnika najčešće uz pomoć HTML helpera. Cijela je aplikacija radi lakše integracije sa mobilnom objavljena (privremeno) na javnom serveru i to na adresi : <http://ec2-34-249-76-120.eu-west-1.compute.amazonaws.com>.



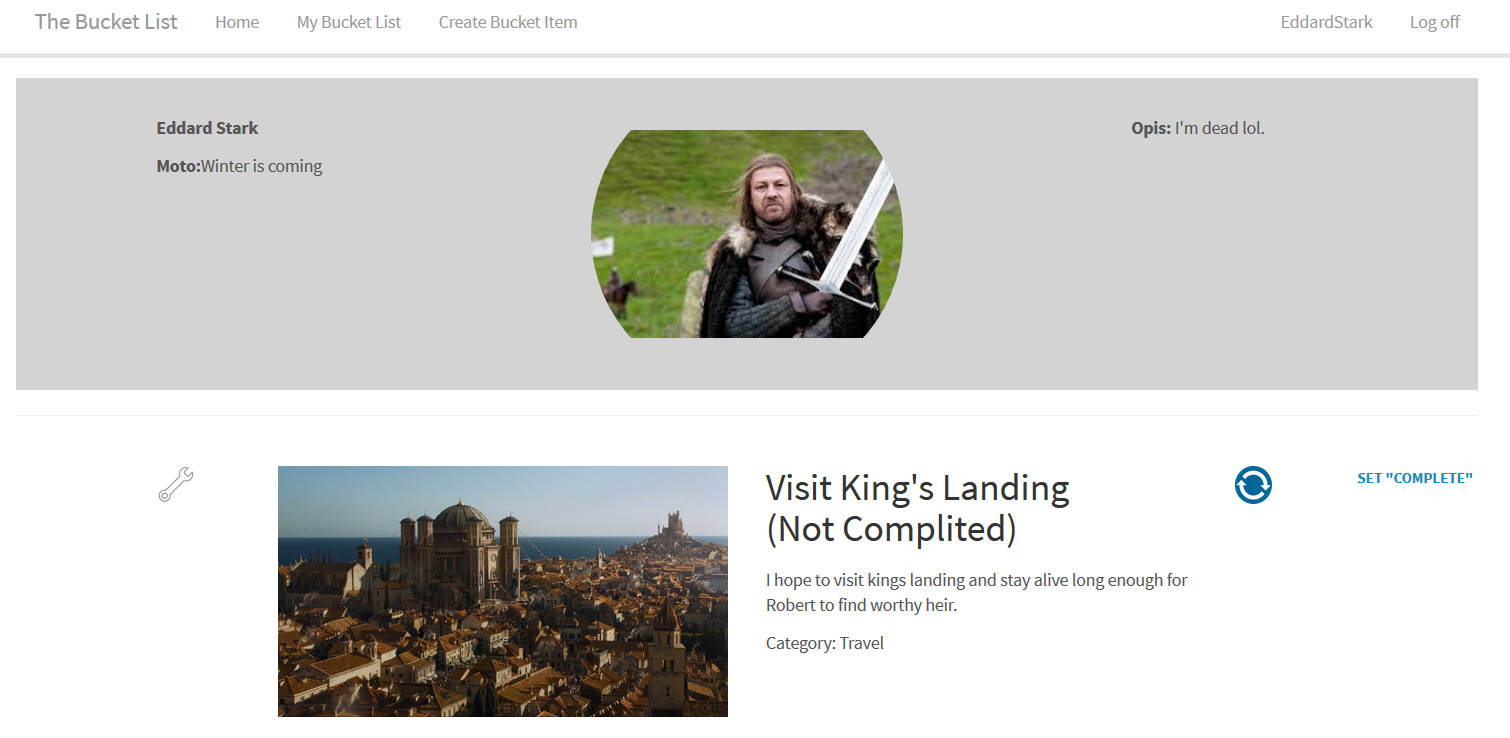
Potencijalni korisnik može se registrirati sa ovom formom.



Jednom kada je stvoren novi korisnik, on se može prijaviti u aplikaciju i urediti svoj profil (npr dodati sliku).



Prijavljeni korisnik također može dodati novi Bucket item (životni cilj).



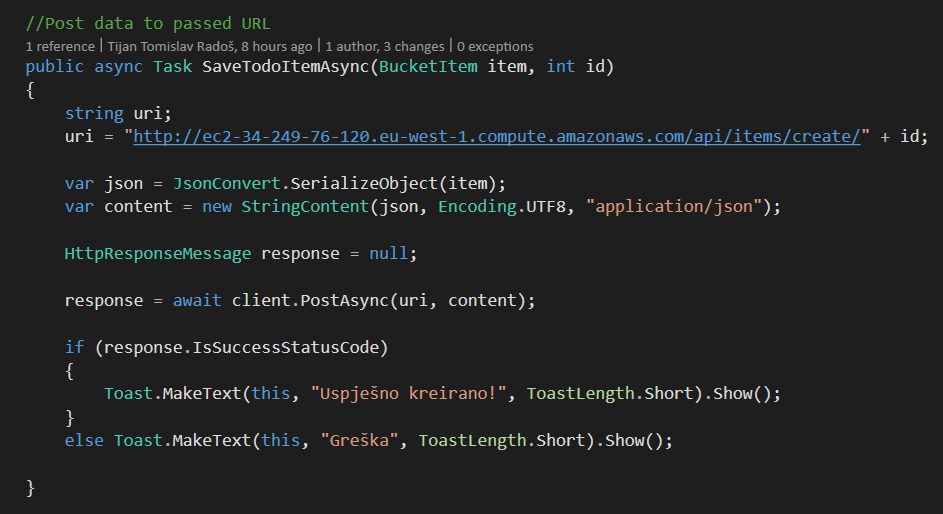
Također, može pregledati svoje ciljeve na svom profilu.

Neke mogućnosti nisu u potpunosti ostvarene u web aplikaciji ali jesu ostvarene u mobilnoj pa će biti prikazane u poglavlju o mobilnoj aplikaciji.

# Mobilna aplikacija

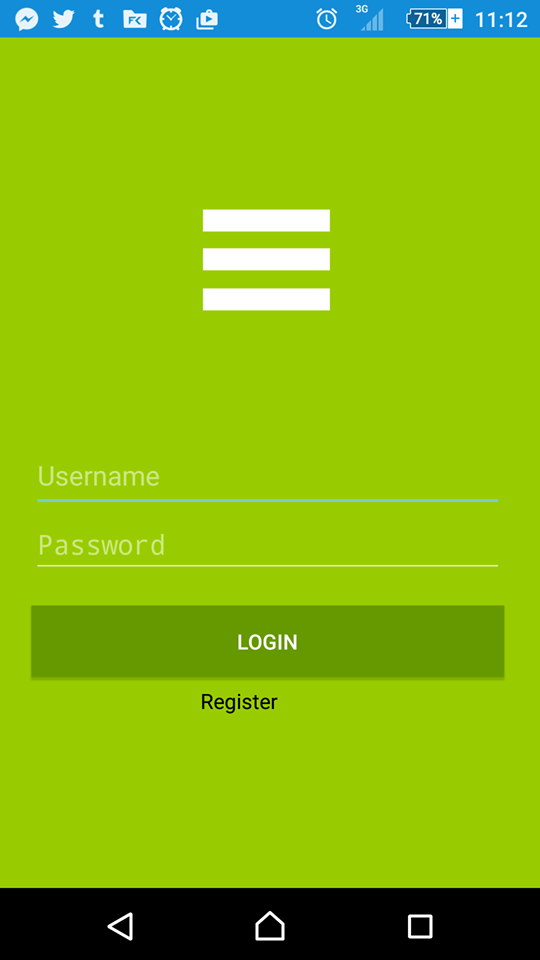
Za izradu mobilne aplikacije korišten je software Xamarin. Mobilna se aplikacija sastoji od nekoliko activity klasa i layouta koji zajedno reagiraju na korisnikove zahtjeve i shodno tomu, prikazuju rezultate na ekranu. Activity klase se povezuju sa zajedničkim backend dijelom aplikacije tako da pozivaju web api kontrolere koji funkcioniraju gotovo isto kao MVC kontroleri za web aplikaciju samo što u ovom slučaju oni direktno primaju i šalju podatke u JSON formatu.

To se provodi na ovakav način, kontroleri se referenciraju preko uri-a:



Slijede neke funkcionalnosti koje su ostvarene u mobilnoj aplikaciji.

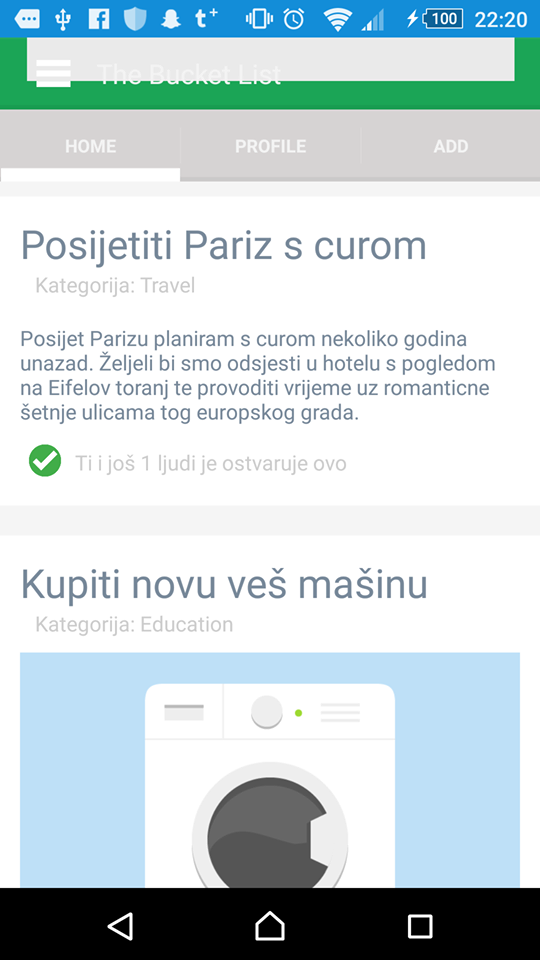
Iz ovog ekrana se registrirani korisnik može prijaviti ili neregistrirani korisnik može odabrati mogućnost da se registrira:



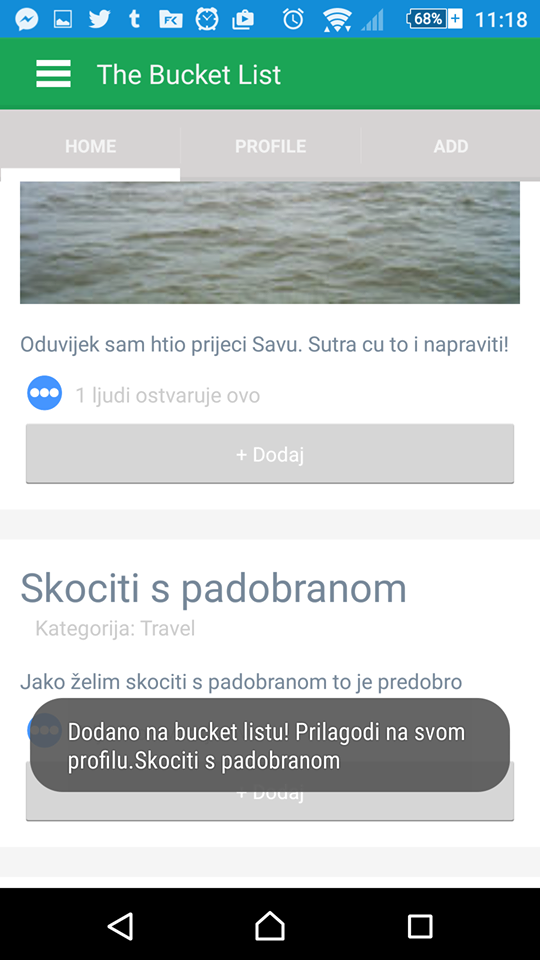
Prijavljeni korisnik može na svom profilu pregledavati koje ciljeve ima, mijenjati im ostvarenost i općenito ih uređivati



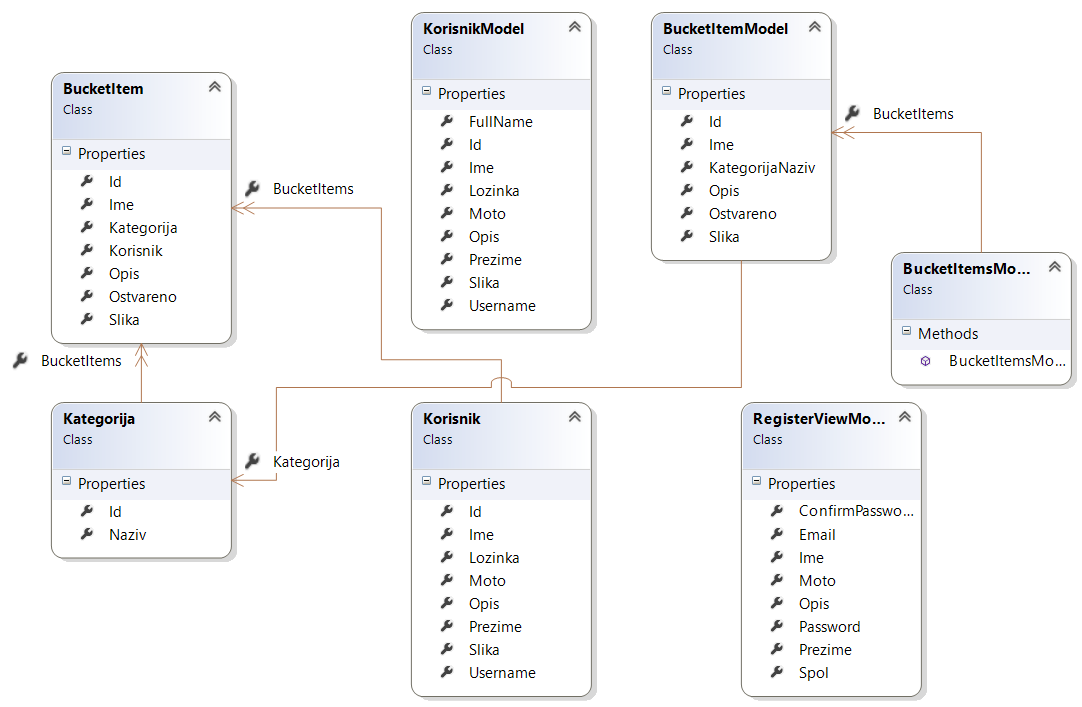
Također, može pregledavati ciljeve drugih korisnika i ciljeve koje dijeli sa drugima.



Kada u ovom prozoru naiđe na cilj koji on nema, a sviđa mu se i želi ga dodati na svoju listu, može to učiniti pritiskom na gumb “+ Dodaj”. To će dodati taj cilj na njegovu listu i omogučiti mu da ga uredi kako bi ga prilagodio sebi.

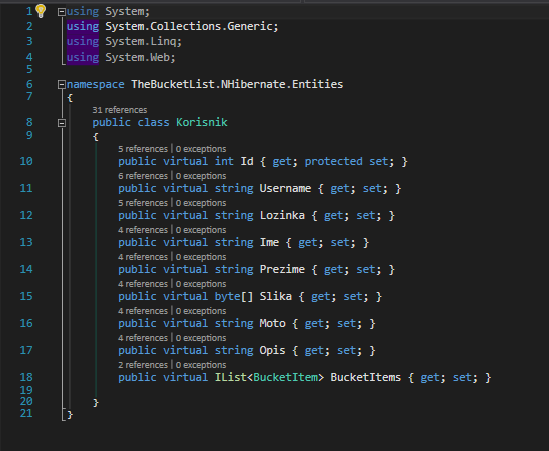


# Class dijagram



# Specifikacija *O/R mapiranja* aplikacije

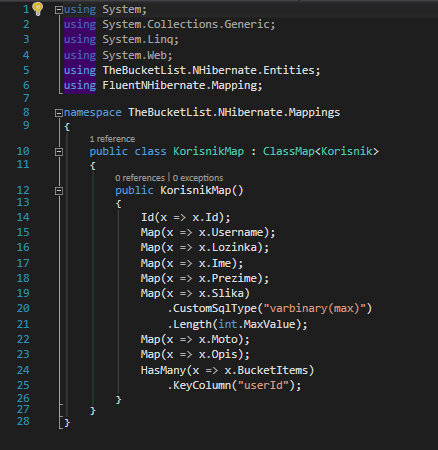
Povezivanje s bazom podataka ostvarili smo pomoću *Fluent NHibernate-a,* čije su glavne komponente vidljive u primjeru entiteta korisnika za kojeg je u nastavku dana i klasa mapiranja te primjer samog dohvaćanja korisnika sa specificiranim ID-em.



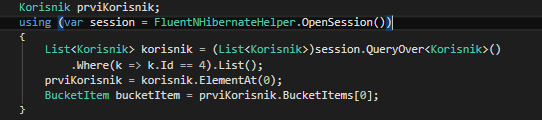
Slika 2 - Klasa entiteta korisnik

*Backend* dio aplikacije sastoji se od entiteta koji odgovaraju relacijama u bazi podataka. Ti entiteti su implementirani analogno primjeru sa slike 2.

Za svaku klasu pojedinačnog entiteta definirana je klasa *EntitetMap.cs* koja nasljeđuje klasu *ClassMap<T>* te određuje O/R mapiranje. Konkretno, pridjeljujemo određenim atributima klase entiteta vrijednosti iz baze podataka. Primjer takve klase dan je na slici 3.



Slika 3 - O/R mapiranje za korisnika



Slika 4 - primjer dohvaćanja korisnika sa određenim ID-jem

Na primjeru sa slike 4 može se vidjeti dohvaćanje korisnika s ID-jem jednakim četiri iz baze podataka korištenjem prethodno navedenih klasa.