Fakultet elektrotehnike i računarstva

Objektno oblikovanje

**Seminar**

**StudIS**

Željko Baranek, Zlatko Hrastić, Josipa Popović, Matej Peroš

Zagreb, siječanj 2017.

SADRŽAJ

[**1.** **Uvod** 2](#_Toc473755705)

[**2.** **Korisnički zahtjevi** 3](#_Toc473755706)

[**2.1.** **Popis obrazaca uporabe** 3](#_Toc473755707)

[**2.2.** **Razrada obrazaca uporabe** 4](#_Toc473755708)

[**3.** **Objektni model** 12](#_Toc473755709)

[**4.** **Arhitektura sustava** 15](#_Toc473755710)

[**5.** **Izvedba perzistencije** 16](#_Toc473755711)

[**6.** **Izgrađene aplikacije** 18](#_Toc473755712)

[**6.1.** **Desktop aplikacija** 18](#_Toc473755713)

[**6.2.** **Web aplikacija** 24](#_Toc473755714)

[**6.3.** **Mobilna aplikacija i WebAPI2** 28](#_Toc473755715)

[**6.3.1.** **Mobilna aplikacija** 28](#_Toc473755716)

[**6.3.2.** **WebAPI2** 30](#_Toc473755717)

# **Uvod**

Kako bismo olakšali praćenje napretka studenata na predmetima, osmišljeno je programsko rješenje *StudIS*. Ono je, također, osmišljeno tako da pomaže profesorima i asistentima s obavezama vezanim uz studente i predmet koji predaju.

Profesorima će biti olakšano definiranje komponenata predmeta, određivanja bodovnih pragova, unošenje bodova i pregledavanje statistike. Za njih će biti definirano web sučelje putem kojem će moći pristupati sustavu i koristiti njegove servise.

Studentima će biti omogućeno pregledavanje bodova na predmetima koje su upisali i statistika predmeta preko dva sučelja: web i Android.

Administratori imaju vlastito desktop sučelje kojim se služe kako bi dodali nove i izmijenili ili obrisali stare predmete ili korisnike sustava. Osim toga, definiraju koji profesor ima pravo unosa bodova na kojem predmetu.

Aplikacija ostvarena na ovaj način će biti otvorena za daljnju nadogradnju na bilo kojem od tri sučelja te će se moći jednostavno održavati.

# **Korisnički zahtjevi**

## **Popis obrazaca uporabe**

* UC1 - RegistracijaKorisnika
* UC2 - IzmjenaPodatakaStudenata
* UC3 - IzmjenaPodatakaProfesora
* UC4 - IzmjenaPodatakaAdministratora
* UC5 - BrisanjeKorisničkogRačuna
* UC6 - PrijavaKorisnika
* UC7 - StvaranjePredmeta
* UC8 - DefiniranjeStrukturePredmeta
* UC9 - IzmjenaStrukturePredmeta
* UC10 - BrisanjePredmeta
* UC11 - UnosBodova
* UC12 - PridruživanjePredmetaStudentima
* UC13 - PregledavanjePredmeta
* UC14 - PregledavanjeStatistikePredmeta
* UC15 - OdjavaKorisnika

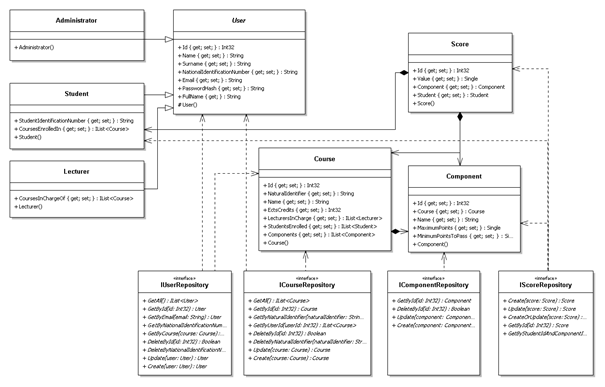
## **Razrada obrazaca uporabe**

* UC1 - RegistracijaKorisnika
  + **Glavni sudionik:** administrator
  + **Cilj:** registracija korisnika u sustav u svrhu korištenja istog
  + **Sudionici:** baza podataka, class library, desktop sučelje
  + **Preduvjet:** račun korisnika ne smije već postojati u bazi podataka
  + **Rezultat:** račun korisnika je zabilježen u bazi podataka
  + **Željeni scenarij:** 
    1. Administrator popunjava sve potrebne podatke za registraciju korisnika
    2. Novonastali račun se sprema u bazu podataka
    3. Sustav potvrđuje uspješnu registraciju
    4. Sustav šalje podatke registriranom korisniku na email adresu
  + **Mogući scenarij:** 
    1. Administrator unosi podatke koji već postoje u bazi podataka, a moraju biti jedinstveni
    2. Sustav dojavljuje pogrešku i traži ponavljanje postupka
* UC2 – IzmjenaPodatakaStudenata
  + **Glavni sudionik**: administrator
  + **Cilj**: izmijeniti osobne podatke
  + **Sudionici**: baza podataka, class library, desktop sučelje
  + **Preduvjeti**: administrator je prijavljen u sustav
  + **Rezultat**: administrator je izmijenio osobne podatke studenta
  + **Željeni scenarij**:
    1. Administratoru se prikažu osobni podaci studenta s mogućnošću promjene
    2. Administrator mijenja neke od podataka te inicira pohranjivanje promjena
    3. Sustav vrši provjeru ispravnosti unesenih podataka
    4. Ako su uneseni podaci ispravni, pohranjuju se u sustav i administratoru se prikazuje poruka o uspjehu
  + **Mogući drugi scenariji**:
    1. Neki od unesenih podataka su neispravni
    2. Administratoru se prikazuje poruka o grešci i od njega se traži da unese ispravne podatke
* UC3 – IzmjenaPodatakaProfesora
  + **Glavni sudionik**: administrator
  + **Cilj**: izmijeniti osobne podatke
  + **Sudionici**: baza podataka, class library, desktop sučelje
  + **Preduvjeti**: administrator je prijavljen u sustav
  + **Rezultat**: administrator je izmijenio osobne podatke profesora
  + **Željeni scenarij**:
    1. Administratoru se prikažu osobni podaci s mogućnošću promjene
    2. Administrator mijenja neke od podataka te inicira pohranjivanje promjena
    3. Sustav vrši provjeru ispravnosti unesenih podataka
    4. Ako su uneseni podaci ispravni, pohranjuju se u sustav i administratoru se prikazuje poruka o uspjehu
  + **Mogući drugi scenariji**:
    1. Neki od unesenih podataka su neispravni
    2. Administratoru se prikazuje poruka o grešci i od njega se traži da unese ispravne podatke
* UC4 – IzmjenaPodatakaAdministratora
  + **Glavni sudionik**: administrator
  + **Cilj**: izmijeniti osobne podatke
  + **Sudionici**: baza podataka, class library, desktop sučelje
  + **Preduvjeti**: administrator je prijavljen u sustav
  + **Rezultat**: administrator je izmijenio svoje osobne podatke
  + **Željeni scenarij**:
    1. Administratoru se prikažu njegovi osobni podaci s mogućnošću promjene
    2. Administrator mijenja neke od podataka te inicira pohranjivanje promjena
    3. Sustav vrši provjeru ispravnosti unesenih podataka
    4. Ako su uneseni podaci ispravni, pohranjuju se u sustav i administratoru se prikazuje poruka o uspjehu
  + **Mogući drugi scenariji**:
    1. Neki od unesenih podataka su neispravni
    2. Administratoru se prikazuje poruka o grešci i od njega se traži da unese ispravne podatke
* UC5 – BrisanjeKorisničkogRačuna
  + **Glavni sudionik**: administrator
  + **Cilj**: izbrisati korisnički račun
  + **Sudionici**: baza podataka, class library, desktop sučelje
  + **Preduvjeti**: administrator je prijavljen u sustav, postoji korisnički račun
  + **Rezultat**: korisnički račun više ne postoji
  + **Željeni scenarij**:
    1. Administrator unosi svoju lozinku (sigurnosna mjera)
    2. Administrator inicira brisanje korisničkog računa
    3. Ako je lozinka ispravna, korisnički se račun zauvijek briše
  + **Mogući drugi scenarij:**
    1. Administrator nije unio ispravnu lozinku
    2. U tom slučaju prikazuje mu se odgovarajuća poruka o grešci i od njega se traži da unese ispravnu lozinku
* UC6 – PrijavaKorisnika
  + **Glavni sudionik**: korisnik
  + **Cilj**: prijava u sustav
  + **Sudionici**: baza podataka, class library, desktop, web ili mobilno sučelje
  + **Preduvjeti**: korisnik je registriran
  + **Rezultat**: korisnik je prijavljen u sustav i sada su mu dostupne sve njegove mogućnosti
  + **Željeni scenarij**:
    1. Korisnik unosi svoju email adresu i lozinku
    2. Sustav provjerava ispravnost unesenih podataka
    3. Ako uneseni podaci odgovaraju podacima korisničkog računa, korisnika se prijavljuje u sustav
  + **Mogući drugi scenariji**:
    1. Uneseni su neispravni podaci
    2. Korisniku se prikazuje odgovarajuća poruka o grešci i vraća ga se na prijavni obrazac
* UC7 - StvaranjePredmeta
  + **Glavni sudionik:** administrator
  + **Cilj:** stvaranje i pohranjivanje predmeta u bazu podataka
  + **Sudionici:** baza podataka, class library, desktop sučelje
  + **Preduvjet:** predmet ne smije postojati u bazi podataka
  + **Rezultat:** novi predmet je pohranjen u bazi podataka
  + **Željeni scenarij:**
    1. Administrator popunjava sve bitne informacije vezane uz predmet
    2. Novi predmet se sprema u bazu podataka
    3. Sustav potvrđuje uspješno kreiranje predmeta
  + **Mogući scenarij:**
    1. Administrator unosi podatke koji već postoje u bazi podataka, a nužno je da su jedinstveni
    2. Sustav dojavljuje pogrešku i traži ponavljanje postupka
* UC8 - DefiniranjeStrukturePredmeta
  + **Glavni sudionik:** profesor
  + **Cilj:** definirati sve komponente predmeta te odrediti maksimume i pragove
  + **Sudionici:** baza, web sučelje, class library
  + **Preduvjet:** predmet mora postojati u bazi podataka
  + **Rezultat:** predmet s definiranom strukturom
  + **Željeni scenarij:**
    1. Profesor određuje strukturu predmeta (komponente) i unosi ju u sustav
    2. Dodavanje pragova i maksimuma bodova po komponenti
    3. Pohrana u bazu podataka te potvrda sustava
    4. Sustav potvrđuje uspješno obavljenu akciju
* UC9 – IzmjenaStrukturePredmeta
  + **Glavni sudionik**: profesor
  + **Cilj**: izmijeniti određene ili sve komponente predmeta, maksimalne bodove i pragove
  + **Sudionici**: baza, web sučelje, class library
  + **Preduvjeti**: profesor je prijavljen u sustav, predmet mora postojati u bazi podataka
  + **Rezultat**: komponente predmeta su uspješno izmijenjene
  + **Željeni scenarij**:
    1. Profesoru se prikažu podaci i komponente predmeta
    2. Profesor mijenja neke od podataka te inicira pohranjivanje promjena
    3. Sustav vrši provjeru ispravnosti unesenih podataka
    4. Ako su uneseni podaci ispravni, pohranjuju se u sustav i profesoru se prikazuje poruka o uspjehu
  + **Mogući drugi scenariji**:
    1. Neki od unesenih podataka su neispravni (negativne vrijednosti pragova ili maksimuma bodova) ili neispunjeni
    2. Profesoru se prikazuje poruka o grešci i od njega se traži da unese ispravne podatke
* UC10 – BrisanjePredmeta
  + **Glavni sudionik**: administrator
  + **Cilj**: izbrisati predmet
  + **Sudionici**: baza podataka, class library, desktop sučelje
  + **Preduvjeti**: administrator je prijavljen u sustav, predmet postoji
  + **Rezultat**: predmet više ne postoji
  + **Željeni scenarij**:
    1. Administrator unosi svoju lozinku (sigurnosna mjera)
    2. Administrator inicira brisanje predmeta
    3. Ako je lozinka ispravna, predmet se račun zauvijek briše
  + **Mogući drugi scenariji**:
    1. Administrator nije unio ispravnu lozinku
    2. U tom slučaju prikazuje mu se odgovarajuća poruka o grešci i od njega se traži da unese ispravnu lozinku
* UC11 - UnosBodova
  + **Glavni sudionik:** profesor
  + **Cilj:** unijeti bodove svim studentima
  + **Sudionici:** baza podataka, class library, web sučelje
  + **Preduvjet:** predmet mora postojati i imati upisane studente
  + **Rezultat:** studentima su upisani bodovi iz predmeta
  + **Željeni scenarij:** 
    1. Profesor odabire predmet kojem želi upisati bodove svih studenata
    2. Profesor dobije listu studenata i upisuje njihove bodove
    3. Sustav potvrđuje pohranu unesenih bodova
  + **Mogući scenarij:**
    1. Profesor upisuje više bodova nego što je definirano kao maksimum
    2. Sustav odbija takav unos
* UC12 - PridruživanjePredmetaStudentima
  + **Glavni sudionik:** administrator
  + **Cilj:** pridružiti predmet studentu
  + **Sudionici:** baza podataka, class library, desktop sučelje
  + **Preduvjet:** predmet mora postojati i student mora postojati, administrator je prijavljen u sustav
  + **Rezultat:** studentu je pridružen predmet
  + **Željeni scenarij:** 
    1. Administrator odabire predmet kojem želi dodati studente
    2. Administrator dobije listu svih studenata i pridružuje ih predmetu
    3. Sustav potvrđuje pohranu unesenih studenata
* UC13 - PregledavanjePredmeta
  + **Glavni sudionik:** administrator
  + **Cilj:** pregled upisanih predmeta
  + **Sudionici:** baza podataka, class library, desktop sučelje
  + **Preduvjet:** administrator je prijavljen u sustav
  + **Rezultat:** pregledana lista predmeta
  + **Željeni scenarij:** 
    1. Administrator pregledava upisane predmeta
* UC14 - PregledavanjeStatistikePredmeta
  + **Glavni sudionik:** profesor
  + **Cilj:** pregled statistike predmeta
  + **Sudionici:** baza podataka, class library, web sučelje
  + **Preduvjet:** sudionik mora biti zaduženi profesor na predmetu
  + **Rezultat:** pregledana statistika
  + **Željeni scenarij:**
    1. Profesor pregledava statistiku
* UC15 - OdjavaKorisnika
  + **Glavni sudionik:** korisnik
  + **Cilj:** odjava korisnika iz sustava
  + **Sudionici:** baza podataka, class library, desktop, web ili mobilno sučelje
  + **Preduvjet:** korisnik mora biti prijavljen
  + **Rezultat:** odjava korisnika
  + **Željeni scenarij:**
    1. Korisnik odabire opciju odjavljivanja
    2. Korisnik se uspješno odjavljuje
  + **Mogući scenarij:**
    1. Sustav nije uspješno odjavio korisnika
    2. U tom slučaju prikazuje mu se odgovarajuća poruka o grešci

# **Objektni model**

* 1. **Specifikacija domenskih modela**

Na prikazanom dijagramu se nalazi osnovni model podataka iz *StudIS* aplikacije. Uz ove razrede, aplikacija sadrži i razrede servisa u kojima se nalazi poslovna logika potreba za baratanje objektima razreda domene.



Slika 3.1. 1. Model podataka

Sva navedena sučelja iz ovog modela izvedena su u infrastrukturnom sloju aplikacije. Nadalje, razred 'User' i svi njegovi izvedeni razredi su entiteti zato što modeliraju stvarne osobe. Razredi 'Course', 'Component' i 'Score' također su entiteti jer posjeduju vlastiti identitet koji je bitan za rad aplikacije. Svi navedeni razredi također su i agregati.

Prva klasa, 'Course', modelira jedan kolegij na fakultetu. Ona sadrži niz potrebnih atributa koji su potrebni kako bi se predmet mogao vjerodostojno prikazati u sustavu. Posebno su istaknute reference na sve komponente koje pripadaju predmetu, na profesore zadužene za predmet i na studente upisane na njega.

Posljednji razred 'Component' modelira komponentu ocjenjivanja određenog predmeta. On predstavlja dio predmeta kao što su međuispit, završni ispit ili laboratorijske vježbe. Sadrži maksimalan broj bodova koji je moguće ostvariti na komponenti, minimalan broj bodova potreban za prolaz (prag) i ime komponente. Također, posjeduje referencu na predmet kojem pripada.

Razred 'Score' predstavlja broj bodova koje je student ostvario po određenoj komponenti. Razlog iz kojeg je ovaj razred napravljen kao 'entity' objekt a ne kao 'value-type' objekt je zato što postoji mogućnost da će u daljnjoj nadogradnji sustava svako bodovanje morati imati svoj identitet.

Repozitorije u našem modelu podataka predstavljaju sučelja 'IUserRepositroy', 'ICourseRepository', 'IComponentRepository' i 'IScoreRepository'.

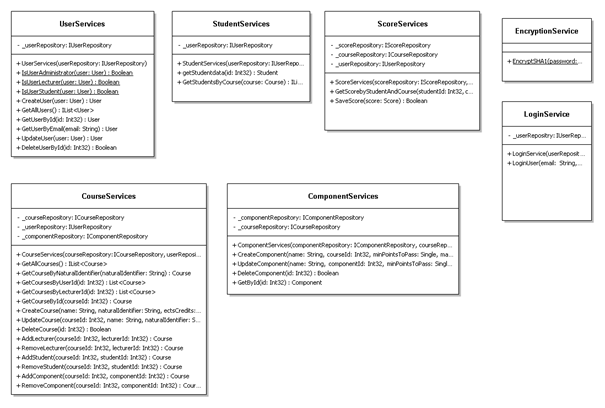
* 1. **Specifikacija servisa**

Sva domenska logika 'StudIS' aplikacije izvedena je putem servisa. Oni su zaduženi za perzistenciju određenih podataka i za provedbu pravila definiranih u domeni. Razredi 'UserServices', 'StudentServices', 'ScoreServices', 'CourseServices' i 'ComponentServices' u konstruktoru primaju sučelja prema repozitorijima. Te veze na dijagramu nisu ucrtane zbog preglednosti.

Svaki razred/servis ima intuitivno ime i sadrži metode koje se odnose na manipulaciju pripadajućim domenskim modelom. Jedine iznimke su razredi 'LoginService' i 'EncryptionService.'

Razred 'EncryptionService' je servis za enkripciju lozinki. Napravljen je odvojeno od servisa za prijavu kako bi se omogućilo korištenje različitih metoda za enkripciju ako se aplikacija bude koristila kod različitih klijenata.

Iako bi se servis za prijavu mogao izvesti i unutar 'UserServices' razreda, odvojen je u poseban servis jer je logički različitih od ostalih metoda korisničkog servisa.



Slika 3.1. 2. Servisi aplikacije

# **Arhitektura sustava**

Arhitektura cjelokupnog sustava napravljena je u slojevima kako bi se aplikacija lakše održavala i kako bi se promjene u jednom dijelu što manje propagirale na druge dijelove. Zbog ovih zahtjeva, arhitektura je napravljena maksimalno strogo slojevita, to jest maksimizirana je komunikacija slojeva samo sa slojem ispod sebe. Zbog ovakvog izbora arhitekture dio servisa samo prosljeđuje podatke infrastrukturnom sloju, no ovakav dizajn važan je kako bi se minimizirale dodatne promjene u slučaju refaktoriranja repozitorija.

Arhitekturu sustava najbolje opisuje Marinescuov peteroslojni model.

* Prezentacijski sloj
  + Pogledi (engl. *View*) iz *MVC* Web aplikacije
  + *Android* dio mobilne aplikacije
  + *WinForms* iz desktop aplikacije
* Aplikacijski sloj
  + Upravljačke klase (engl. *Controllers*) iz *MVC* Web aplikacije
  + Upravljačke klase *WebAPI* - ja
  + Upravljačke klase *WinForms* aplikacije
* Servisi
  + Posebni projekti i pripadajuće klase označene riječima *Service* ili *Services*
* Domena
  + Svi domenski specificirani modeli
  + Modeli (engl. *Models*) iz *MVC* aplikacije
* Sloj perzistencije
  + Implementacije sučelja repozitorija

# **Izvedba perzistencije**

S obzirom na zahtjeve domene koja je dosta generičke naravi, uloga očuvanja perzistencije dodijeljena je relacijskoj SQL bazi podataka. Kako bi se olakšala komunikacija sustava s bazom podataka, koristio se *Object-relational mapper* (ORM) *NHibernate*. Referenca na *NHibernate* postoji jedino iz projekta 'StudIS.DAL' i koristi se u implementaciji 'NHibernateService'.

'NHibernateService' iz 'StudIS.DAL' implementira perzistenciju u *MSSQL* *Express* bazu podataka u datoteci, a 'NHibernateService2' iz 'SQLiteConfig' u *in-memory SQLite* bazu podataka koja se koristi za testiranje.

*NHibernate* zahtijeva podešavanje koje može biti izvedeno na dva načina: pisanjem XML konfiguracije ili korištenje *FluentNHibernate* dodatka koji generira XML konfiguraciju. U našem projektu je odabran *FluentNHibernate* zbog čitkijeg koda koji je lakše održavati. Konfiguracija podrazumijeva definiranje *ConnectionString*-a za bazu podataka u koju želimo perzistirati podatke te dodavanja razreda koji definiraju mapiranja podataka.

Mapiranje podataka podrazumijeva uparivanje atributa razreda iz modela s atributima tablice u relacijskoj bazi podataka. Osim toga, moguće je postaviti i integritetska ograničenja kao što su *Unique*, *Primary* *Key*, *Foreign* *key* i dozvoljeni rasponi vrijednosti te mapirati međusobne veze (asocijacije) među entitetima - *one-to-one*, *one-to-many* i *many-to-many*.Osim toga, *NHibernate* pruža kvalitetno rješenje problema mapiranja hijerarhije razreda u tablice - što je iskorišteno prilikom mapiranja razreda 'Student', 'Lecturer' i 'Administrator' koji svi nasljeđuju apstraktni razred 'User'.

Odabrana je strategija korištenja iste tablice za sva tri razreda modela (entiteta) uz diskriminatorski atribut - *Table per class hierarchy*. Ona je pogodna zato što se podrazredi razlikuju u malo atributa (npr. 'Student' ima JMBAG dok 'Lecturer' i 'Administrator' nemaju).

namespace StudIS.DAL.Mappings

{

public class StudentMap : SubclassMap<Student>

{

public StudentMap()

{

DiscriminatorValue("student");

Map(x => x.StudentIdentificationNumber);

HasManyToMany(x => x.CoursesEnrolledIn);

}

}

}

Kod 5.1. Primjer mapiranja

S obzirom da tijekom razvoja može doći do promjene modela i mapiranja, iste se promjene moraju odraziti i u tablicama baze podataka. Kako se to ne bi moralo obavljati ručno, koristi se *NHibernate Schema Export* funkcionalnost koja gradi novu ili promijenjenu strukturu baze podataka (tablice, atribute, ograničenja…).

Perzistencija podataka se odvija u razredima repozitorijima koji su izvedeni iz sučelja repozitorija definiranih u modelu domene. Svaki repozitorij u svojem konstruktoru prima referencu na *INHibernateService* tj. u slučaju naših aplikacija svi repozitoriji primaju instancu *NHibernateService* razreda. Kod stvaranja repozitorija za svaki repozitorij stvara se nova sesija i tako se osigurava perzistencija unutar repozitorija čak i ako podaci još nisu zapisani u bazu podataka. Na ovaj način izbjeglo se otvaranje sesije kod svakog poziva bilo koje metode repozitorija.

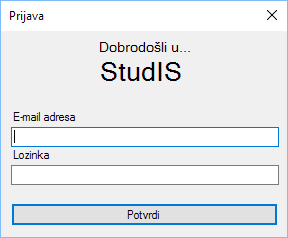
Ovakav pristup je dobar kod *MVC* i *WebAPI* aplikacija jer se repozitoriji injektiraju u kontrolere i stvaraju se kada stvaraju i kontroleri. Kod desktop aplikacije potrebno je voditi više računa kod stvaranja repozitorija i otvaranja novih sesija.

# **Izgrađene aplikacije**

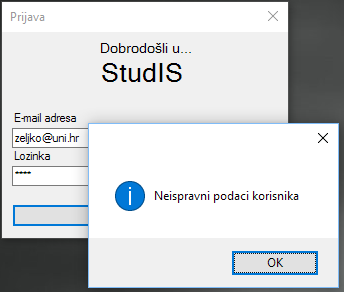
## **Desktop aplikacija**

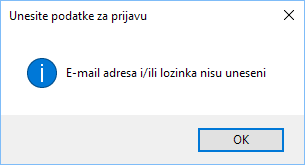
Desktop aplikacija pruža administratorima sustava sučelje putem *Windows Formi*.

Početni zaslon je obrazac za prijavu u sustav (*LoginForm*) koji od korisnika traži unos e-mail adrese i lozinke.



Slika 6.1. 1. Prijava korisnika

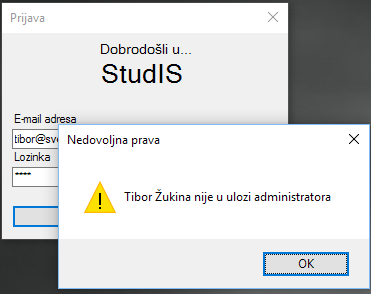
Korisnika se obavještava ukoliko ne unese vrijednost u neko od polja ili ako podaci ne pripadaju niti jednom korisničkom računu.



Slika 6.1. 2. Podaci nisu uneseni

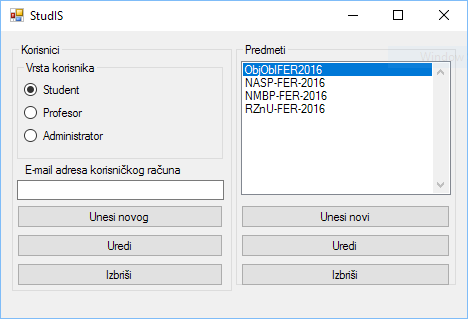
Slika 6.1. 3. Neispravni podaci korisnika

S druge strane, ukoliko su podaci korisnika valjani, ali on nije u ulozi administratora (nego je, primjerice, student ili predavač), dobiva se upozorenje.



Slika 6.1. 4. Nedovoljna prava za prijavu

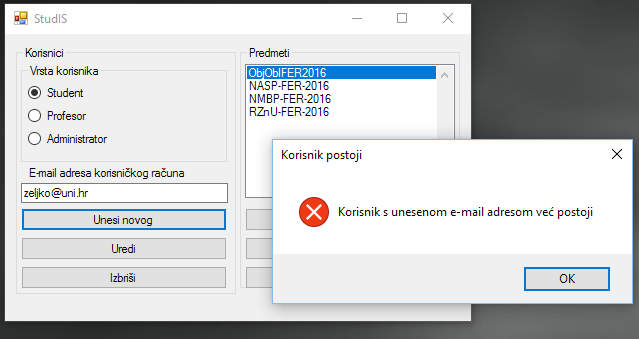
Po uspješnoj prijavi, otvara se glavni prozor aplikacije.



Slika 6.1. 5. Uspješna prijava

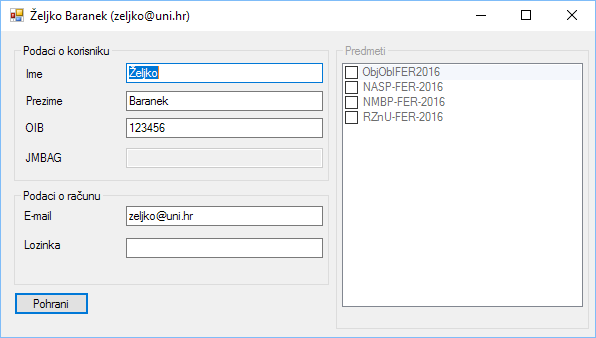
S lijeve strane nalaze se kontrole za CRUD operacije s korisnicima, a s desne za predmete.

Obzirom da se pretpostavlja veći broj korisnika, njih se dohvaća tako što se prvo unese e-mail adresa korisnika s kojim se želi raditi. Ako se želi stvoriti novog korisnika, odabire se vrsta korisnika koji će biti stvoren, unesi njegova buduća e-mail adresa korisničkog računa te odabire gumb “Unesi novog”.

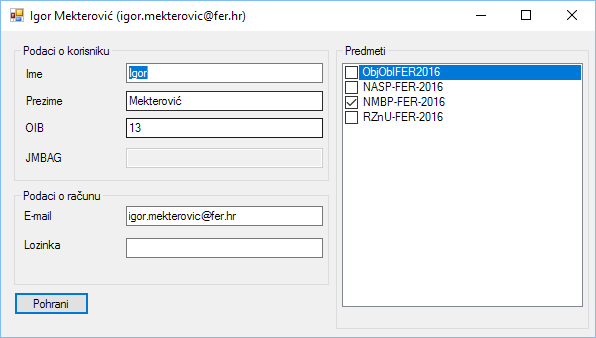


Slika 6.1. 6. Korisnik već postoji

Ukoliko se žele urediti podaci postojećeg korisnika, postupak je isti, osim što se pritišće gumb “Uredi”.



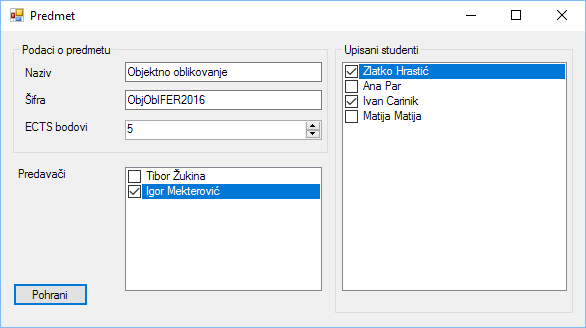
Slika 6.1. 7. Uređivanje administratora



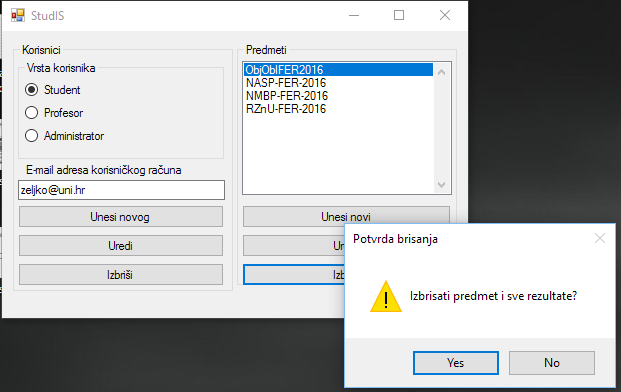
Slika 6.1. 8. Uređivanje profesora

Obrazac za stvaranje novog korisnika je isti kao obrazac za uređivanje postojećeg korisnika.

Ovisno o vrsti korisnika, pojedina polja su siva i onemogućena (primjerice, JMBAG za administratore i predavače te predmeti za administratore, dok su za studente dostupna oba polja, za predavače samo predmeti).

Pri radu s predmetima, odabir predmeta za rad vrši se iz polja s popisom svih predmeta (prikazani su prirodni identifikatori) te se pritiskom na gumb za uređivanje ili brisanje inicira željena radnja.

Slika 6.1. 9.Stvaranje i uređivanje predmeta

Prilikom pokušaja brisanja (bilo korisnika, bilo predmeta), administratora se traži eksplicitna potvrda. Zbog konfiguracije NHibernate-a, pokušaj brisanja podatka na koji postoje strani ključevi rezultirat će bacanjem iznimke. Ona će biti uhvaćena u korisničkoj aplikaciji i bit će prikazana poruka o neuspješnoj radnji brisanja.

Slika 6.1. 10. Brisanje predmeta

S obzirom na pretjeranu složenost klasičnog oblikovnog obrasca *MVC* i *Observer* za ovakvu aplikaciju, desktop aplikacija oblikovana je na način koji najviše podsjeća na '*Passive View'*.

U projektu 'StudIS.Desktop' postoji folder *Controllers* u kojem su smještene upravljačke klase. Oni u konstruktoru primaju servise, a servisima su prethodno u njihovom konstruktoru poslane implementacije repozitorija.

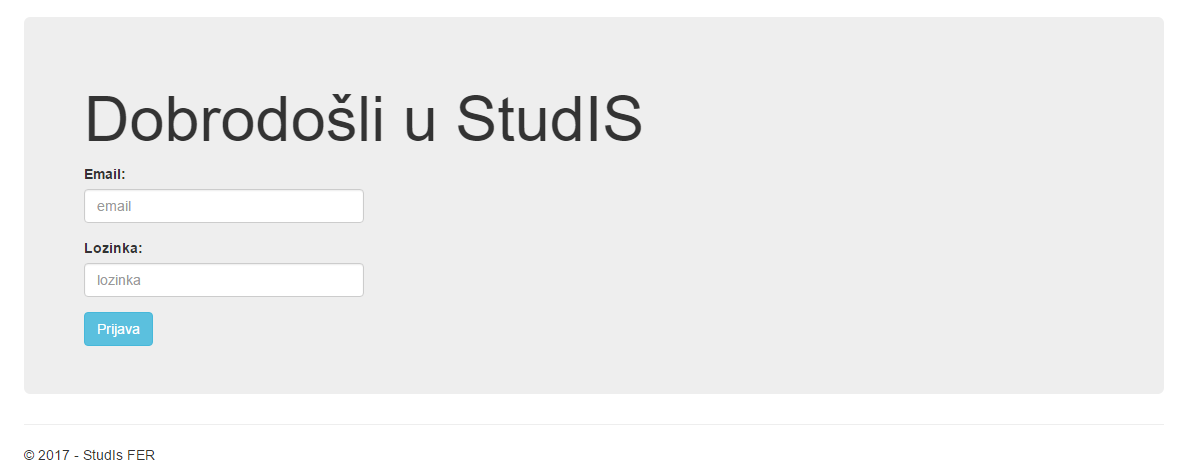
Problemi s ovakvim pristupom je taj, da ukoliko dođe do iznimke u sloju servisa ili repozitorija, dolazi i do zatvaranja sesije s bazom podataka. Zbog toga daljnje ponašanje aplikacije nije moguće predvidjeti.

## **Web aplikacija**

Web aplikacija je izrađena pomoću *ASP.NET MVC* radnog okvira (engl. *Framework*). Upravo taj okvir je izabran zbog implementiranog oblikovnog obrasca Model - Pogled - Upravljač (engl. *Model - View - Controller*) već na razini arhitekture. *MVC* obrazac je jedan od temeljnih obrazaca koji rezultiraju dobrim dizajnom sustava i općenito je poželjan za uporabu kada ga je god moguće primijeniti. Posljedica korištenja tog konkretnog radnog okvira je da se razvojni programer obvezuje poštivati obrazac, ali mu se to uvelike olakšava kroz niz ugrađenih mogućnosti.

Za generiranje pogleda su korišteni modeli koji nisu isti kao oni domenski zbog potrebe za kombinacijom raznih podataka. Upravljanje stranicama se izvodi s kontrolerima koji koriste servise za rad s podacima.

Nakon pokretanja aplikacije, kako bi se nastavilo s radom, potrebno je prijaviti se u sustav.

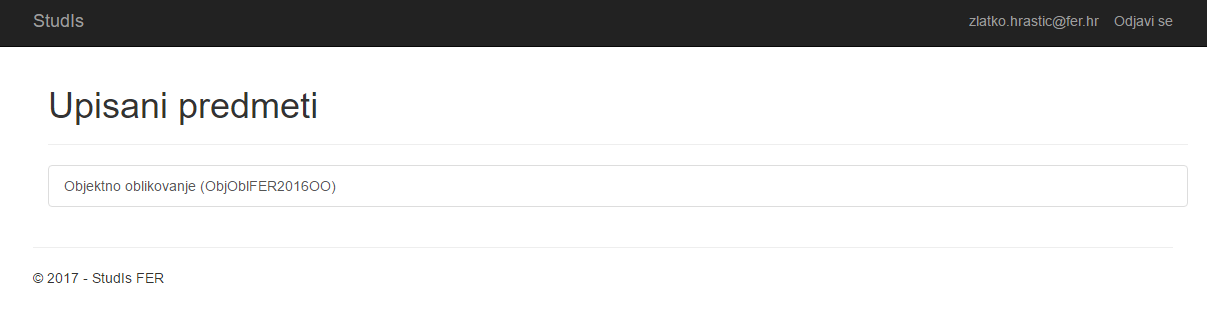


Slika 6.2. 1. Prijava u sustav

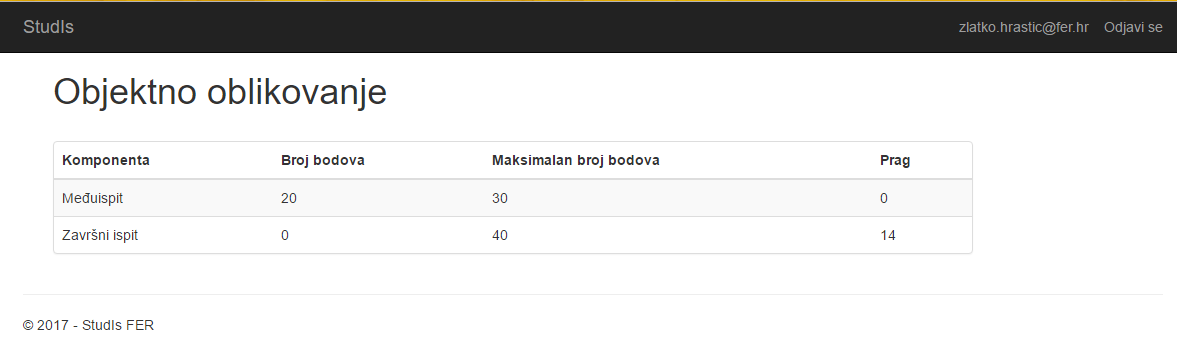
Ukoliko su uneseni podaci krivi sustav ne dopušta nastavak rada već zahtjeva ponovni unos. Postoje dvije skupine korisnika koje mogu koristiti ovo sučelje: studenti i profesori.

Studenti imaju samo nekoliko opcija, dok profesori posjeduju mnogo više.

Nakon prijave u studentski račun prikazuje se naslovna stranica na kojoj je popis svih upisanih predmeta. Pritiskom na bilo koji od predmeta otvara se stranica na kojoj su detalji komponenta i bodova po komponentama.

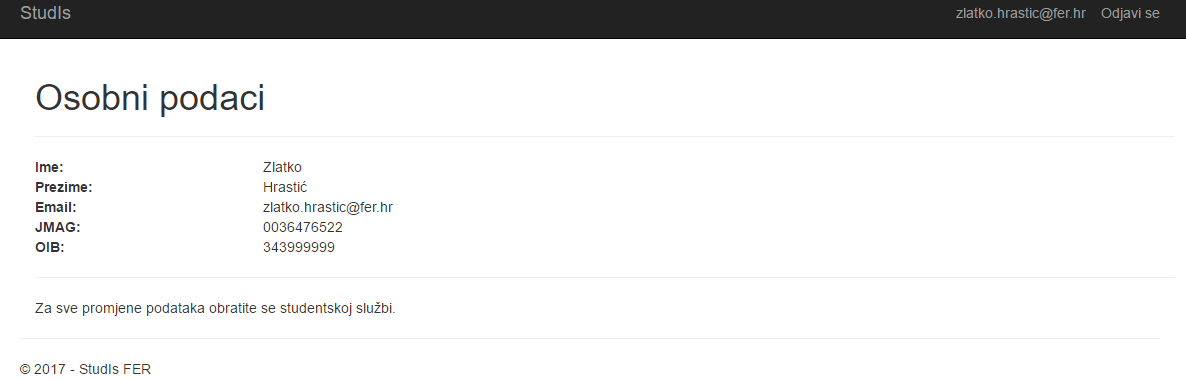


Slika 6.2. 2. Upisani predmeti



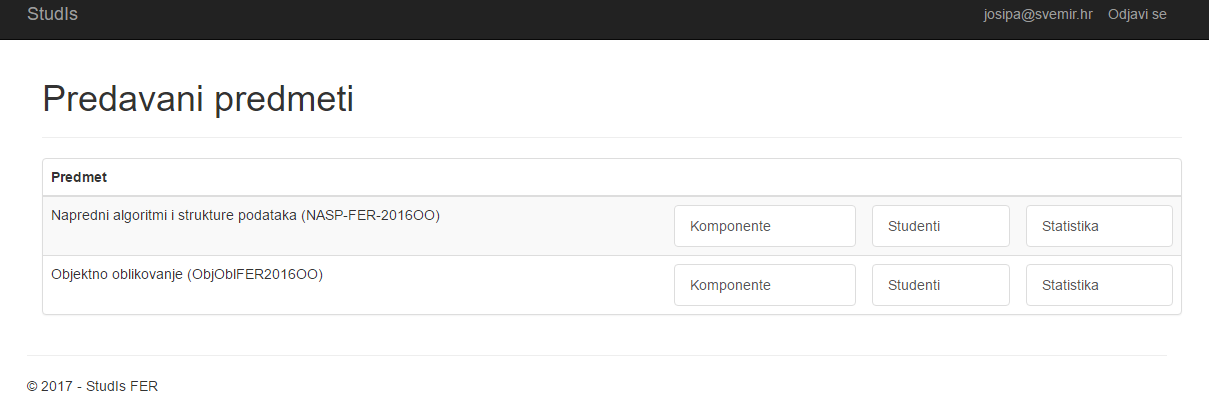
Slika 6.2. 3. Detalji predmeta

Alatna traka koja je prisutna neovisno o stranici koja se prikazuje omogućava neometanu navigaciju. Pritiskom na 'StudIS' vraćamo se na početnu stranicu, dok se pritiskom na vlastiti e – mail stranica preusmjerava na prikaz osobnih podataka. Pritiskom na tipku 'Odjavi se' student se odjavljuje iz sustava.



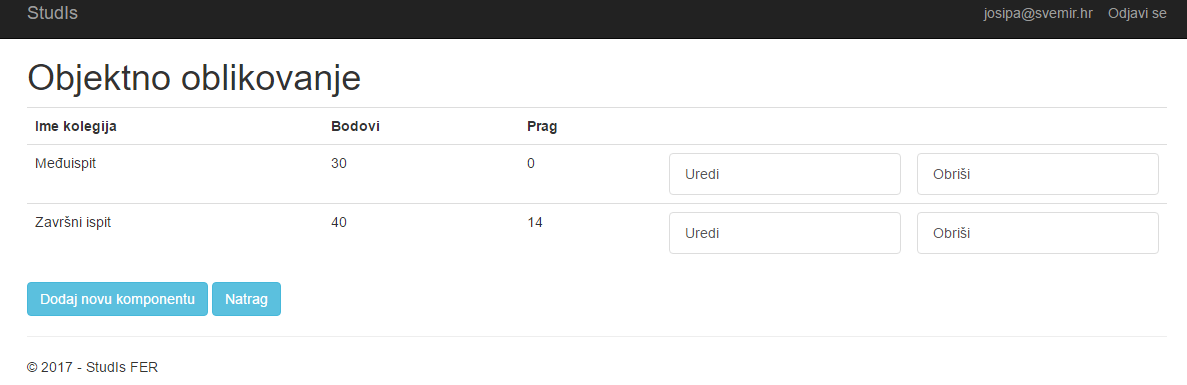
Slika 6.2. 4. Osobni podaci studenta

Prilikom prijave na račun profesora prikazuje se stranica sa svim predmetima na kojim isti sudjeluje.



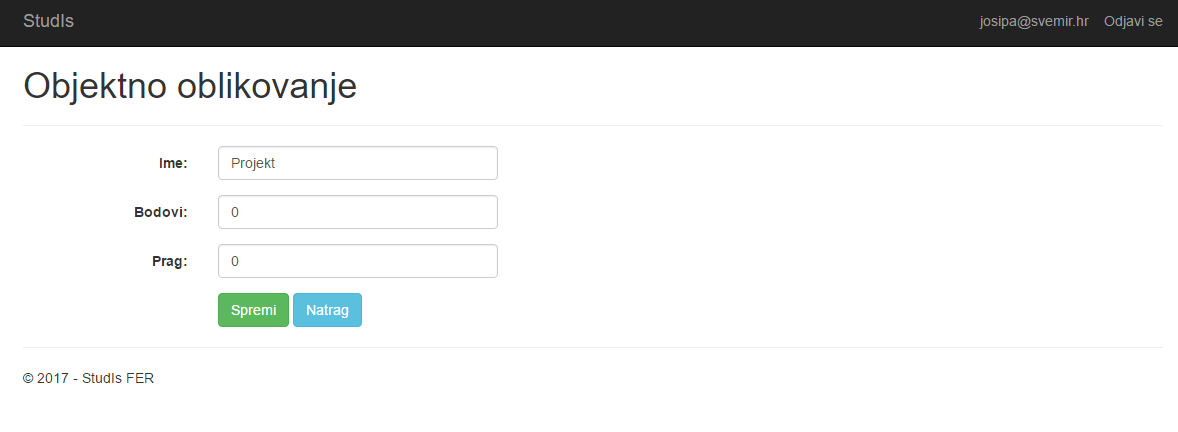
Slika 6.2. 5. Predavani predmeti

Pritiskom na tipku 'Komponente' stranica se preusmjerava na popis komponenata za pojedini predmet. Tu su ponuđene opcije za uređivanje ili brisanje postojećih komponenata te za dodavanje novih. Također je dostupan gumb 'Natrag' koji omogućava povratak na početnu stranicu.



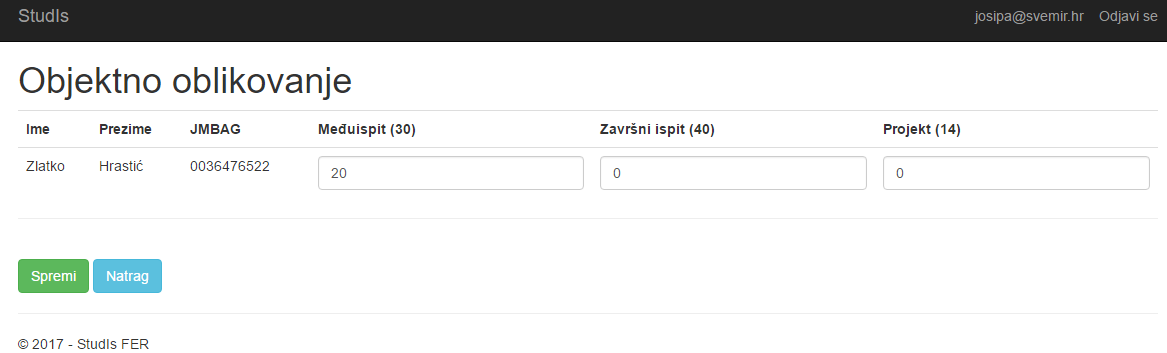
Slika 6.2. 6. Sve komponente kolegija

Ukoliko je odabrana opcija za brisanje pokazuje se pogled za potvrdu brisanja. Bitno je napomenuti da se predmet ne može pobrisati ukoliko postoje studenti kojima su upisani bodovi za nju. Prilikom uređivanja i stvaranja novog predmeta nije moguće ostaviti ime, bodove ni prag praznim. Svaki od tih pogleda ima gumb za povratak na komponente kolegija.



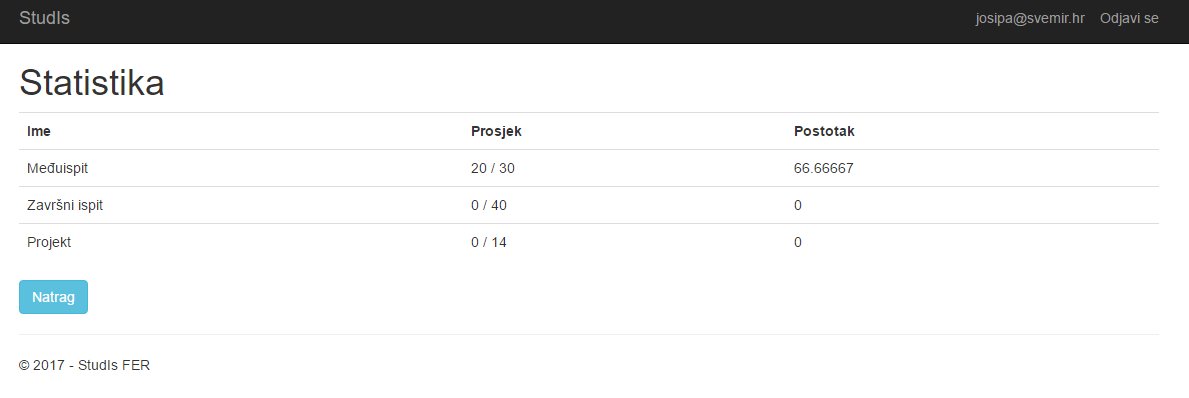
Slika 6.2. 7. Dodavanje komponente

Odabirom opcije 'Studenti' se prikaže pogled s popisom svih studenata na kolegiju i njihovim bodovima po komponentima. Svi bodovi za sve studente se mogu odjednom promijeniti. Na taj način se jako brzo može unositi velika količina bodova odjednom.



Slika 6.2. 8. Studenti

Pritiskom na gumb 'Statistika' dolazi se do statistike za pojedini predmet po komponentama.



Slika 6.2. 9. Statistika

Alatna traka omogućava navigaciju isto kao i kod studentskog računa.

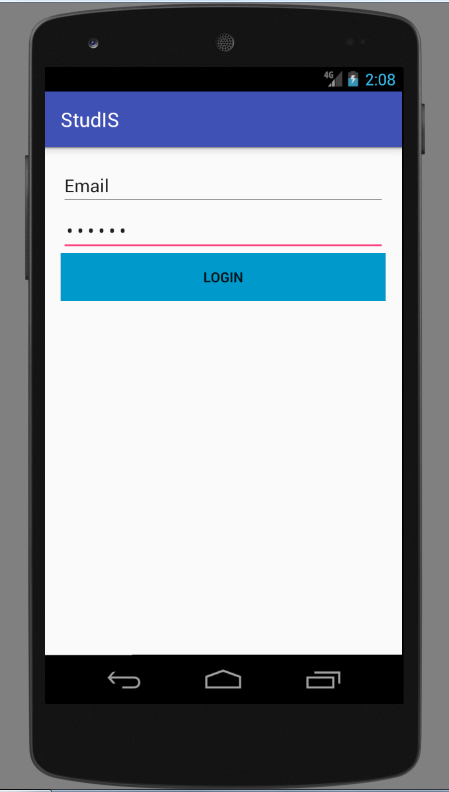
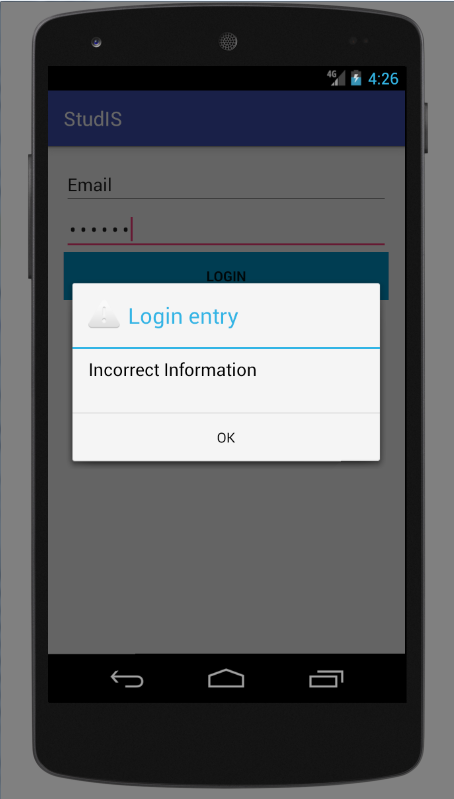
## **Mobilna aplikacija i WebAPI2**

### **Mobilna aplikacija**

Mobilna aplikacija je izgrađena pomoću *Android studio 2.2* okruženja. Aplikacija je namijenjena pregledu studentskih podataka i bodova na upisanim predmetima pisana u *Android 4.1* *Jellybeans* verziji za 96% uređaja koji koriste upravo tu verziju.

Aplikacija se sastoji od dva dijela: *Activity* i *Request*. *Request* služi za slanje i dohvaćanje informacija o studentu preko *WebAPI2* sučelja, pomoću *HTTP* knjižnice *Volley*. Svi poslani podaci su u *JSON* formatu. *Activity* su srž aplikacije, te prikazuju dohvaćene informacije i omogućuju studentu da navigira sučeljem po svojoj potrebi.

Pri pokretanju aplikacije student ima prikazan ekran za prijavu, kojeg kontrolira 'Login Activity'. Upisom vlastitog e – maila i šifre pokreće se 'LoginRequest' koji upisane podatke šalje *WebAPI2* sučelju preko URL-a za prijavu. Lozinka se nikada ne šalje u originalnom obliku već se kriptira. Ukoliko su upisane informacije točne prikazuje se pogled za upisane predmete, pokrene se 'CoursesActivity' koji šalje dobivene informacije o studentu.



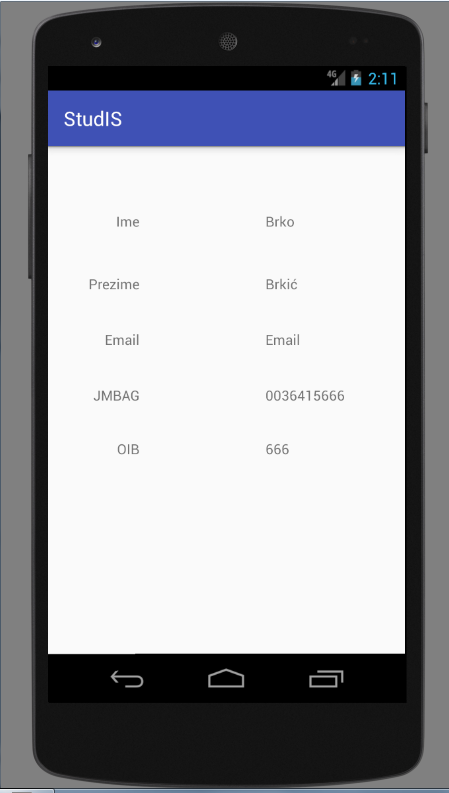
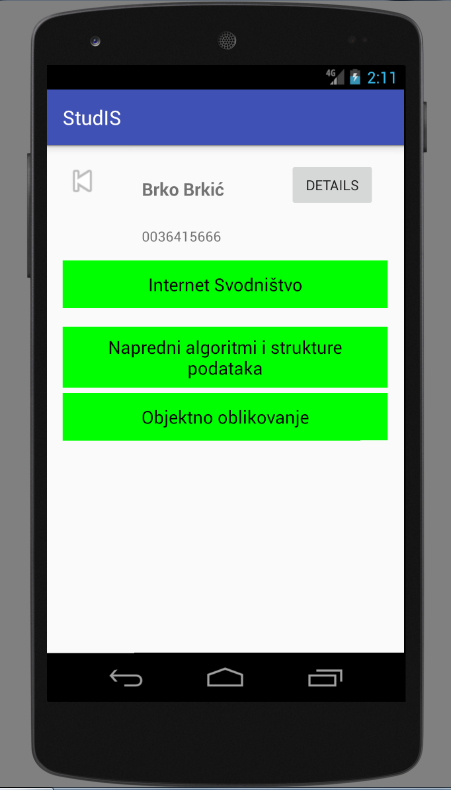
Slika 6.3.1. 1. Prijava

Slika 6.3.1. 2. Neuspjela prijava

'CoursesActivity' prilikom pokretanja koristi *CoursesRequest* da dohvati listu predmeta upisanog studenta preko dobivenog identifikatora putem URL-a *http://{host}/api/StudentData/GetCoursesByStudentId/{StudentId}.*

Nakon toga student može dohvatiti svoje detalje, vratiti se na početnu stranicu ili vidjeti bodove upisanih predmeta. Klikom na dobivenu dugmad aplikacija se postavlja na odgovarajuće pogled i pokreće 'ScoreActivity' ili 'UserDataActivity'.

'UserDataActivity' prilikom pokretanja pokreće *UserRequest* koji pošalje GET upit na [*http://{host}/api/StudentData/GetStudentData/{Id}*](http://10.0.2.2:42271/api/StudentData/GetStudentData/%7BId), i prikaže dobivene podatke na ekranu.



Slika 6.3.1. 3. Uspješan login

Slika 6.3.1. 4. Detalji korisnika

'ScoreActivity' prilikom pokretanja pokreće *ScoreRequest* koji pošalje GET upit na *http://{host}/api/StudentData/GetScoreData/{studentId}/{courseId},* i prikaže dobivene bodove na ekranu.



Slika 6.3.1. 3. Prikaz bodova iz kolegija

### **WebAPI2**

Pristupna točka za mobilnu aplikaciju u našem sustavu izgrađena je korištenjem *Microsoftove WebAPI2* tehnologije. Sve akcije koje obavlja ovo sučelje su akcije čitanja s obzirom na to da mobilna aplikacija služi za lakši pregled podataka, ali ne i za njihovo mijenjanje. Mobilna aplikacija namijenjena je isključivo za studente i zbog toga *WebAPI2* sučelje ima implementiranu logiku samo za pregledavanje podataka vezanih za studente.

Sučelje aplikacije sastoji se od 2 kontrolera: 'LoginControlle' i 'StudentController'. 'LoginController' je kontroler koji u sebi sadrži samo metodu za prijavljivanje korisnika. On prima korisnički e – mail i kriptiranu lozinku, a vraća podatke o studentu koji se pokušao prijaviti u sustav.

U slučaju netočnih podataka za prijavu ili u slučaju da se pokušao prijaviti korisnik koji nije student, kontroler neće na izlazu dati nikakve podatke. Ovaj kontroler je usko povezan sa 'LoginService' razredom jer koristi njegove usluge kako bi provjerio podatke korisnika.

'StudentDataController' je upravljački razred zadužen za izvršavanje akcija kojima se traže podaci o prijavljenom studentu. Sadrži 3 metode.

Prva metoda je 'List<SimpleCourseModel> GetCoursesByStudentId(int studentId)'. Ona dohvaća podatke o predmetima na koje je upisan student s pripadajućim identifikatorom. Ova metoda usko je povezana sa 'CourseServices' razredom jer od njega dobiva podatke. Metoda ne vraća *Course* iz modela domene podataka nego vraća 'SimpleCourseModel'. To je je *ViewModel* razred koji se koristi kako bi se izlaz sučelja izolirao od domenskih modela i kako promjena u modelu domene ne bi nužno izazvala promjenu na izlazu iz metode. Metodi se pristupa preko sljedećeg *URL*-a: : *http://{host}/api/StudentData/GetScoreData/{studentId}/{courseId}.*

Druga metoda je 'ScoredCourse GetScoreData(int studentId,int courseId)'. Ona dohvaća podatke o bodovanju studenta u pojedinom predmetu. Metoda kao parametre prima identifikator studenta i identifikator predmeta te vraća bodove za sve komponente definirane u traženom predmetu. Metoda vraća instancu 'ScoredCourse' razreda koji je *ViewModel* razred. On se sastoji od imena predmeta i liste 'SimpleScore' objekata. 'SimpleScore' je također *ViewModel* razred, a predstavlja pojednostavljenje 'Score' razreda iz modela domene. Metodi se pristupa putem URL-a: *http://{host}/api/StudentData/GetScoreData/{studentId}/{courseId}.*

Treća metoda *WebAPI2* sučelja je metoda 'SimpleStudentModel GetStudentData(int id)'. Ova metoda vraća osobne podatke studenta s traženim identifikatorom. Metoda je usko vezana uz 'StudentServices' razred s obzirom na to da koristi njegove usluge kako bi dohvatila podatke. Ako metoda primi nepostojeći identifikator ili identifikator korisnika koji nije student, metoda neće vratiti nikakve podatke. Inače metoda vraća 'SimpleStudentModel' model podatka koji je pojednostavljenje 'Student' razreda iz modela podataka. Metodi se pristupa preko URL-a *http://{host}/api/ StudentData/GetStudentData/{id}.*

Kako *WebAPI2* kontroleri ne bi bili spregnuti s implementacijama repozitorija koristi se obrazac injekcije ovisnosti. Injekcija repozitorija u kontrolere obavlja se uz pomoć knjižnice *Ninject*. *Ninject* omogućava povezivanje razreda i sučelja repozitorija na način da kada kontroler prima referencu na određeno sučelje *Ninject* mu prosljeđuje povezani razred. Ovako povezivanje obavlja se samo jednom i to pri pokretanju 'StudIs.WebAPI' projekta. Ovakvim pristupom omogućava se lagana promjena repozitorija u budućim nadogradnjama i omogućava se lakše testiranje kontrolera korištenjem *Mockova*.

*WebAPI2* sučelje je vrlo pogodno za korištenje za pisanje sučelja prema vanjskim aplikacijama. Omogućava da se svi modeli koje vraćaju metode kontrolera serijaliziraju u *JSON* ili *XML* formatu. Ova karakteristika je u slučaju naših aplikacija vrlo korisna s obzirom na to da je *WebAPI2* sučelje pisano u *C#*, a *Android* aplikacija je pisana u jeziku *Java*. S obzirom na to da *WebAPI2* sučelje ne koristi binarnu serijalizaciju, klijenti nisu ograničeni na korištenje *.NET Frameworka* i jezika *C#*. Zbog toga se na *WebAPI2* sučelje aplikacije mogu vrlo lako spojiti i drugi klijenti koji žele koristiti *StudIS* aplikaciju.

# 