Univerzitet u Sarajevu

Elektrotehnički fakultet

Odsjek za računarstvo i informatiku

E-Learning web-aplikacija

Dokumentacija modela aplikacije

Članovi grupe:

Negra Ahmetspahić
Muhamed Opačin
Adnan Šabanović

Sarajevo, juni 2020.

SADRŽAJ

1. Opis teme aplikacije

ELearning je aplikacija sa ugrađenom platformom za učenje. Aplikacija sadrži oblasti koje su podijeljene u kurseve. U zavisnosti od toga da li korisnik pristupa aplikaciji sa fakultetskim e-mailom ili ne, otvara/ne otvara mu se mogućnost da pohađa dodatne kurseve. Moguć je pristup više korisnika. Korisnik također može izabrati oblasti, a i kurseve koje će pohađati. Glavne obaveze aplikacije su da snabdijeva korisnika informativnim sadržajem na razne načine i također vrši provjeravanje usvojenog znanja putem kvizova. Aplikacija također prati i informiše korisnika o napretku u određenim oblastima. Korisnik je također u mogućnosti da ocjenjuje kurs koji je pohađao. Svaki dan korisnik može pristupiti dnevnom izazovu (daily challenge.) Ukoliko korisnik želi pristupiti navedenom izazovu, dobija 3 pitanja iz svakog kursa koji pohađa, te nakon odrađenog izazova može vidjeti rang listu svih korisnika tog kursa koji su pristupili dnevnom izazovu. Primarni cilj aplikacije jeste usvajanje novog znanja, a također i vještina koje korisnik stiče.

1.1. Akteri

Primarni tip aktera

- Posjetilac
- Korisnik
- Korisnik aplikacije bez fakultetskog e-maila
- Korisnik aplikacije sa fakultetskim e-mailom
- Administrator

Drugi tip aktera

- Eksterna aplikacija za provjeravanje, odnosno validaciju e-mail adrese i provjera da li adresa pripada studentu

1.2. Funkcionalnost

- Posjetilac
    - Mogućnost prijavljivanja na sistem
    - Mogućnost kreiranja korisničkog računa
    - Moguć pregled oblasti
    - Moguć pregled kurseva
- Korisnik
    - Mogućnost prijavljivanja na sistem

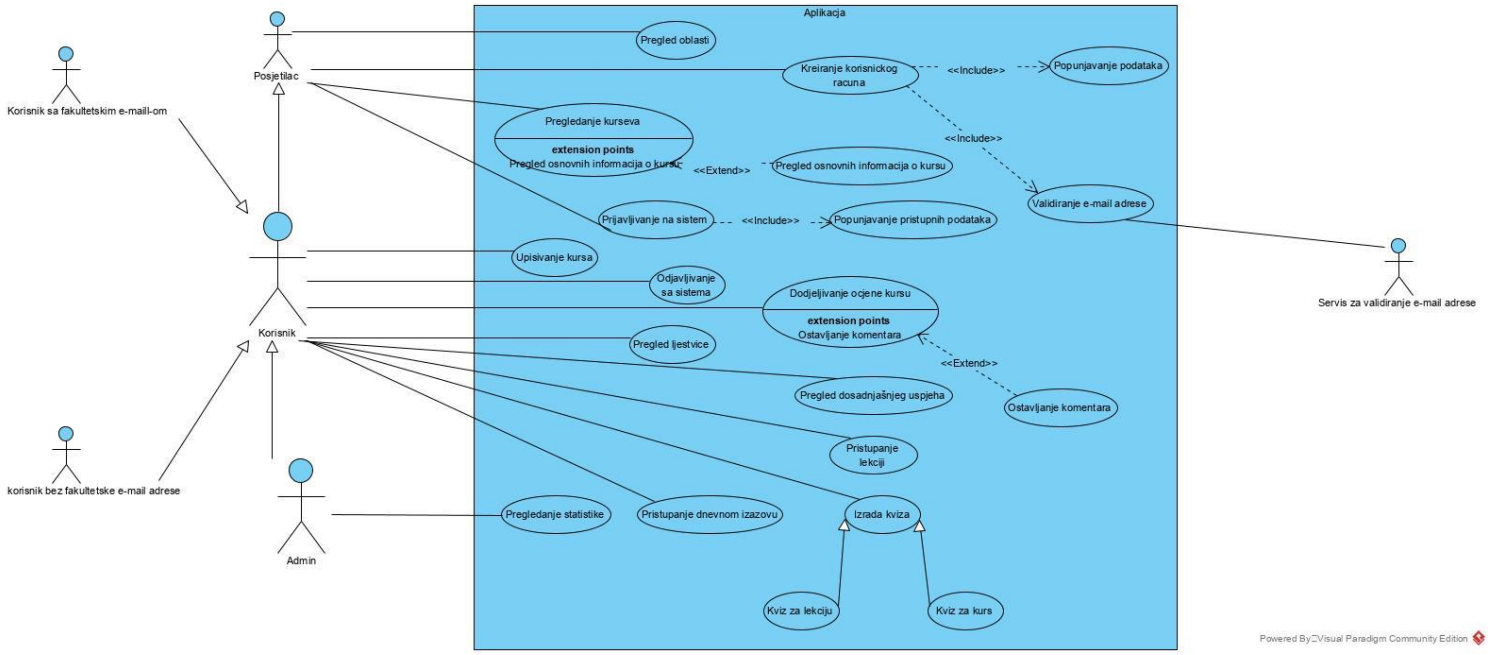
    - Mogućnost kreiranja korisničkog računa
    - Moguć pregled oblasti
    - Moguć pregled kurseva
    - Mogućnost upisivanja kursa
    - Pristupanje lekcijama
    - Izrada kviza za lekciju
    - Izrada kviza za kurs
    - Mogućnost dodjeljivanja ocjene kursu
    - Pristup dnevnom izazovu
    - Mogućnost pregledanja ljestvice kursa kojeg pohađa
    - Ostavljanje komentara na kursu
    - Pregled dosadašnjeg uspjeha
    - Mogućnost odjavljivanja sa sistema
- Korisnik aplikacije bez fakultetskog e-maila
    - Sve funkcionalnosti korisnika
- Korisnik aplikacije sa fakultetskim e-mailom
    - Sve funkcionalnosti korisnika
- Administrator
    - Sve funkcionalnosti korisnika
    - Može pregledati statistiku svih korisnika

1.3. Web servis

Dnevni izazov (daily challenge) je planiran da bude web servis. Ova funkcionalnost služi za određivanje najboljih korisnika koji će se prikazivati na ljestvici.

2. USE CASE dijagram i scenariji

2.1. USE CASE dijagram



Slika 2.1.1. USE CASE dijagram

2.2. Scenariji

Scenarij 1.

Naziv: Pregled oblasti koje su dostupne na sistemu od strane korisnika/posjetioca
Opis: Korisnik/posjetilac pregleda oblasti koje su mu dostupne na platformi putem Web interfejsa kako bi se odlučili koja im oblast najviše odgovara.

Glavni tok: Završava uspješno.
Preduvjeti: Korisnik/posjetilac ima dostupan Web interfejs za pregled.
Posljedice: Korisniku/posjetiocu omogućen pregled oblasti.

Tok događaja:

Korisnik/posjetilac	Sistem ELearn
1. Pristupanje interfejsu za pregled	
	2. Prikaz trenutnih oblasti koje su dostupne na platformi
3. Na osnovu pregleda svih podataka, korisnik/posjetilac pronalazi njemu najzanimljiviju oblast	

Scenarij 2.

Naziv: Odjavljivanje sa sistema od strane korisnika
Opis: Korisnik nakon prijavljivanja na platformu i nakon određenog vremena provedenog na platformi, odjavljuje se sa iste.

Glavni tok: Završava uspješno, korisnik se nakon završetka rada na platformi odjavljuje.
Preduvjeti: Korisnik je prijavljen i ima dostupan Web interfejs za odjavljivanje.
Posljedice: Korisnik je odjavljen.

Tok događaja:

Korisnik/posjetilac	Sistem ELearn
1. Pristupanje interfejsu za odjavljivanje	
	2. Prikaz mogućnosti koje korisnik aplikacije može odabrati
3. Pregledom svih mogućnosti, korisnik odabira odjavljivanje (log out button)	

Scenarij 3.

Naziv: Pregled ljestvice najboljih korisnika za dati kurs.

Glavni tok: Korisnik pritiskom na odgovarajuće dugme otvara rang listu na kojoj može da vidi najbolje rangirane korisnike u tom kursu.
Preduslovi: Prijava na sistem, upisan u kurs.
Posljedice - uspješan završetak: Uspješno se prikaže rang lista.
Posljedice - neuspješan završetak: Rang lista se ne prikaže.

Primarni akteri: Korisnik

Tok događaja:

Korisnik	Sistem ELearn
1. Korisnik pristupa aplikaciji	
2. Korisnik ulazi u kurs u koji je upisan	
3. Korisnik pritišće dugme za ljestvicu	
	4. Prikaz rang ljestvice

Scenarij 4.

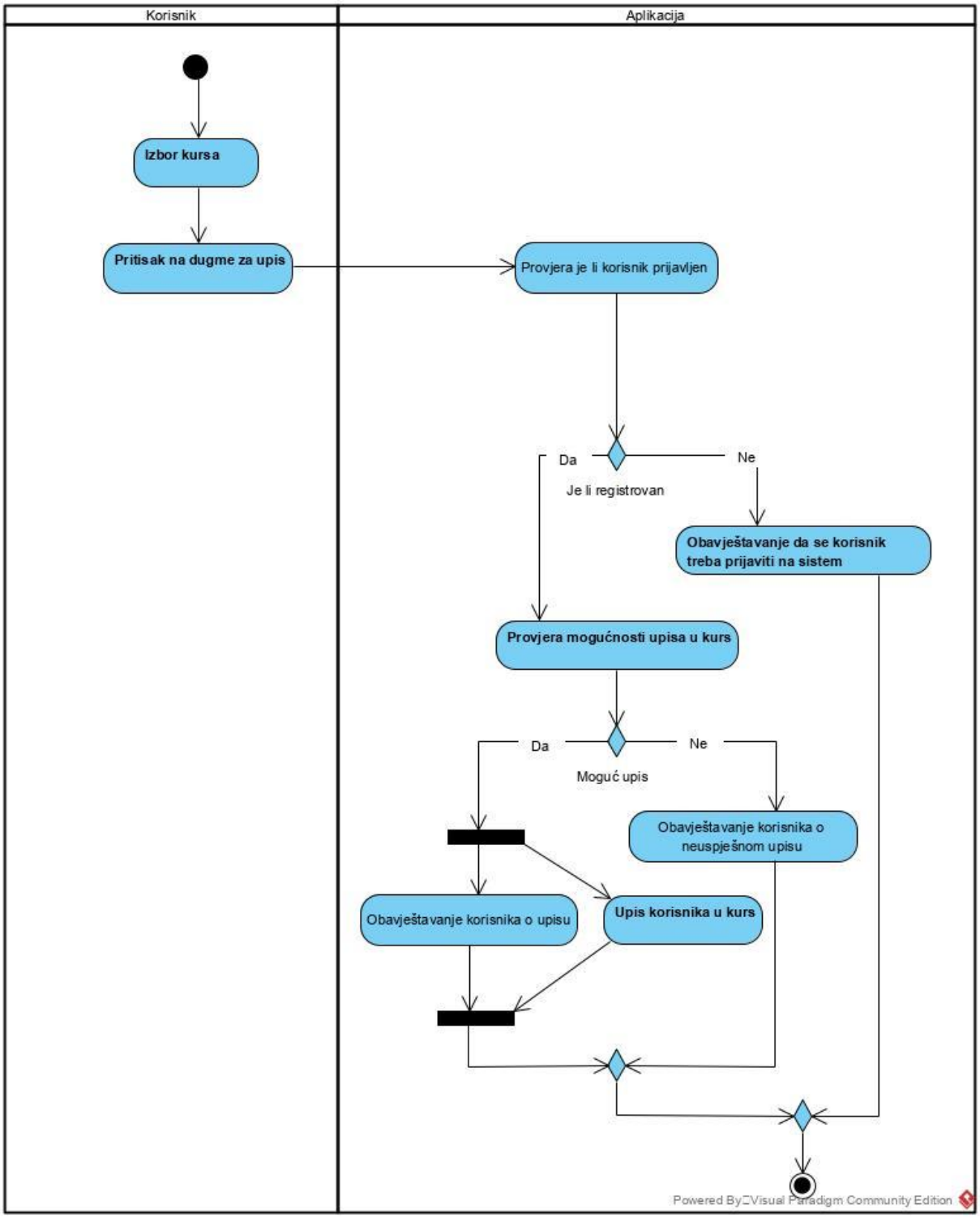
Naziv: Pristup lekcijama unutar kursa od strane korisnika
Opis: Korisnik nakon prijavljivanja na platformu i kursu koji pohađa, odabire lekciju koja je po planu i programu.

Glavni tok: Korisnik pomoću multimedijalnog sadržaja koji je vezan za tu lekciju usvaja znanje.
Preduvjeti: Korisnik je prijavljen i pohađa kurs.
Posljedice: Korisniku se prikazuje sav sadržaj date lekcije.

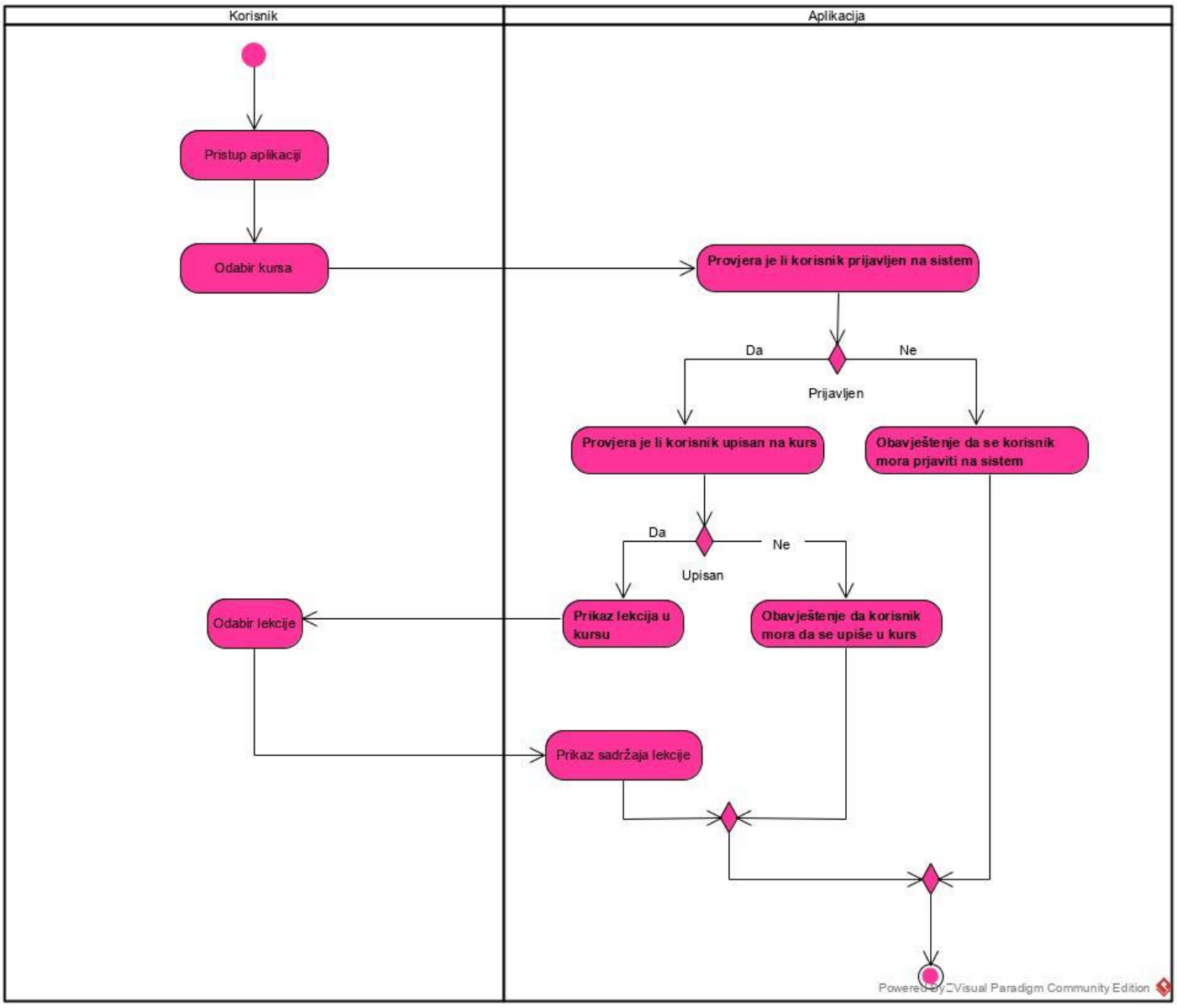
Tok događaja:

Korisnik/posjetilac	Sistem ELearn
1. Pristupanje aplikaciji	
	2. Prikaz svih oblasti i kurseva iz te oblasti koje aplikacija nudi
3. Odabir kursa kojeg korisnik pohađa	
	4. Korisniku se nudi spisak lekcija iz datog kursa
5. Korisniku odabira lekciju koju želi	
	6. Prikaz lekcije sa pratećim multimedijalnim sadržajem

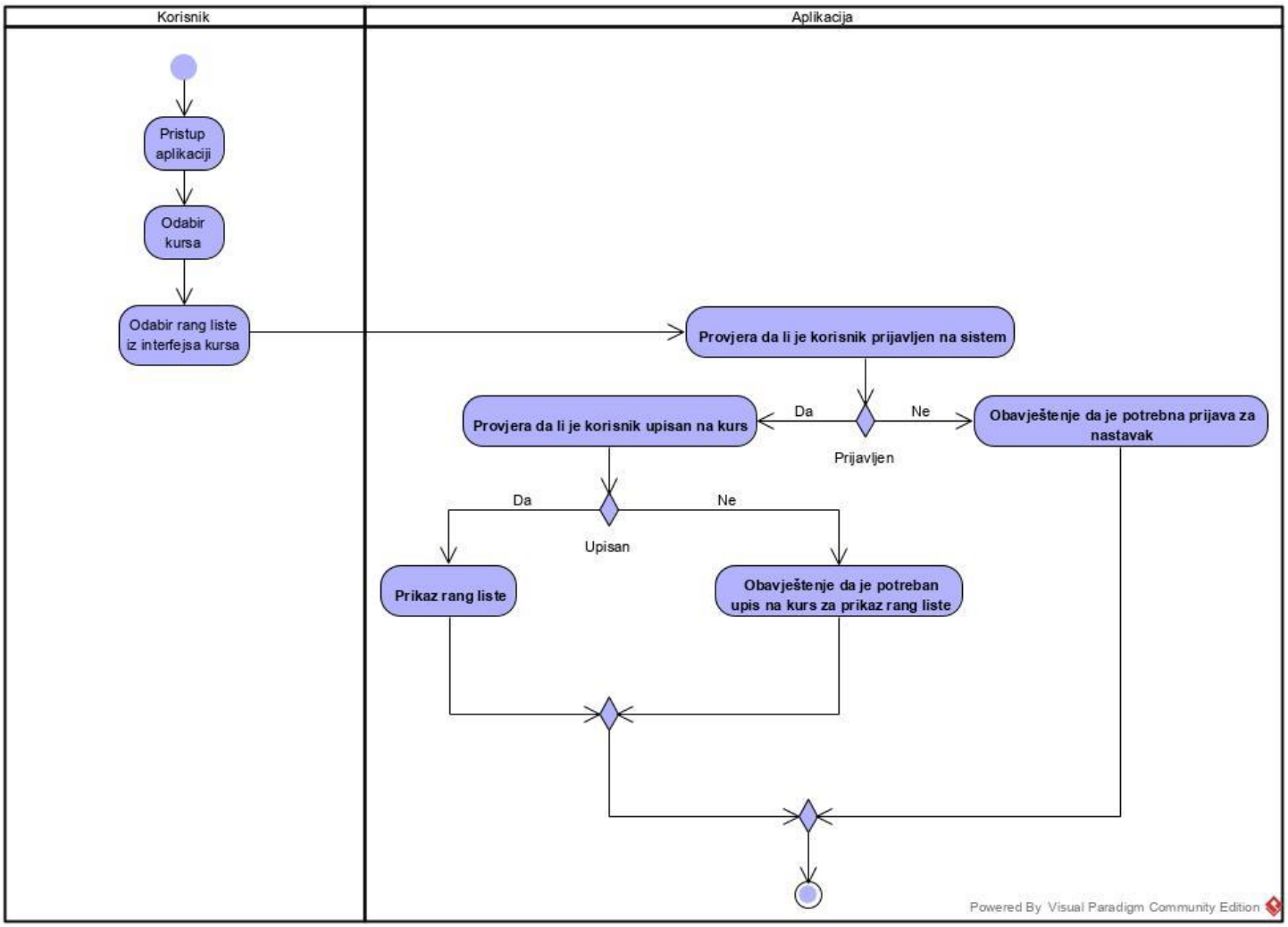
3. Dijagram aktivnosti



Slika 3.1. Dijagram aktivnosti 1



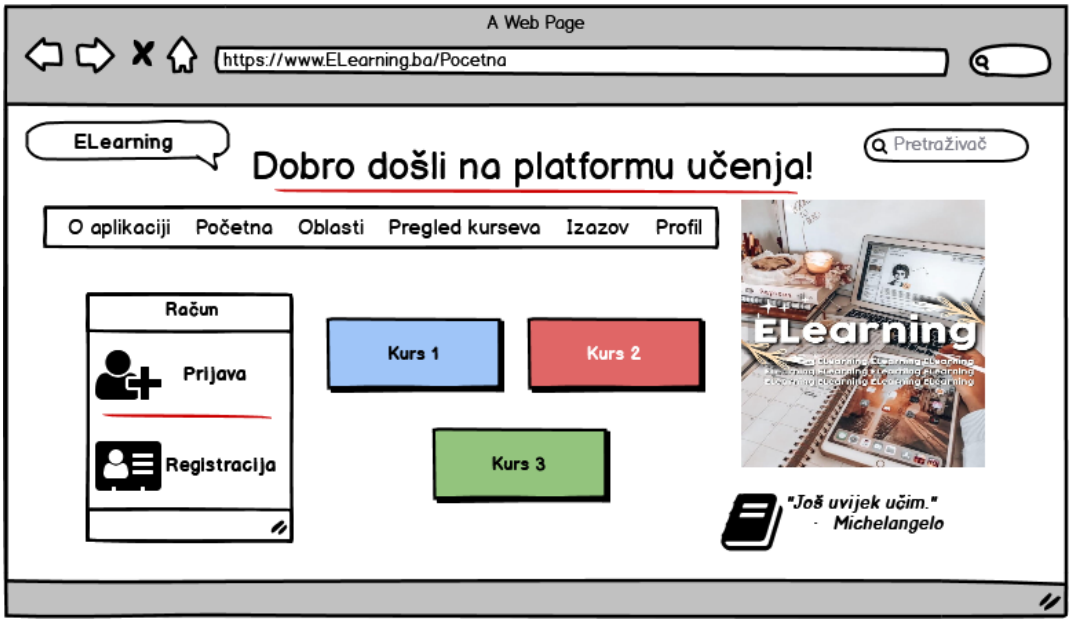
Slika 3.2. Dijagram aktivnosti 2



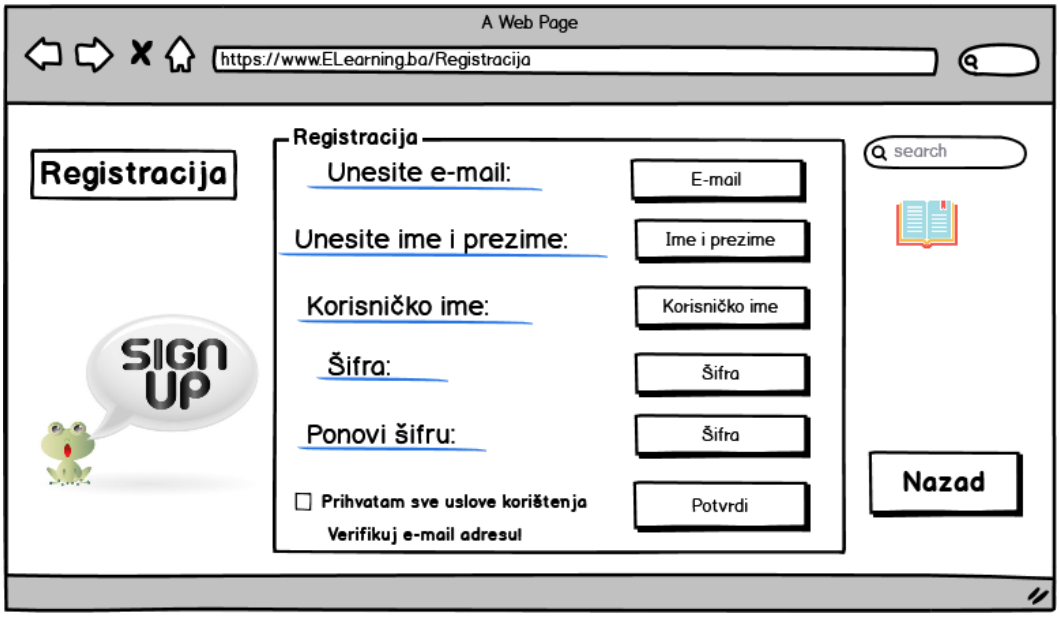
Slika 3.3. Dijagram aktivnosti 3

4. Prototip formi sistema

Kao što se može vidjeti na slici 4.1., na početnoj stranici se nalazi nešto više o aplikaciji, Početna, oblasti itd, te račun korisnika. Ukoliko posjetilac ima svoj korisnički račun, on se prijavljuje sa svojim korisničkim imenom i šifrom. Ukoliko nema, može kliknuti na tipku za registraciju na početnoj ili na dugme na ovoj stranici, što se može vidjeti na slikama ispod. Korisnik se registruje sa nekoliko informacija koje treba ispuniti. Prihvatanjem svih uslova i popunjavanjem navedenih informacija, posjetilac postaje korisnik naše aplikacije. Klikom „O aplikaciji“ koji se nalazi na početnoj stranici, posjetilac i korisnik imaju mogućnost da vide informacije o našoj stranici, kao što su osnivači, opis i mreže putem kojih može pratiti kretaore aplikacije, što se vidi na slici 4.4.

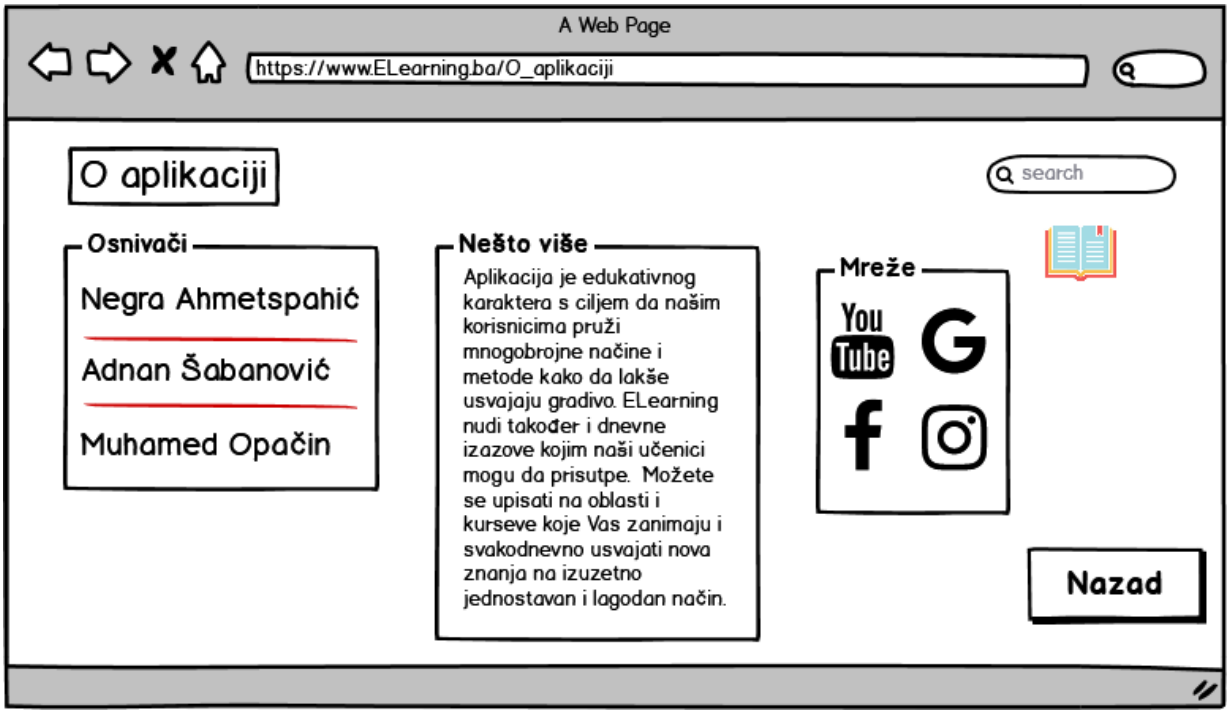


Slika 4.1. Početna stranica



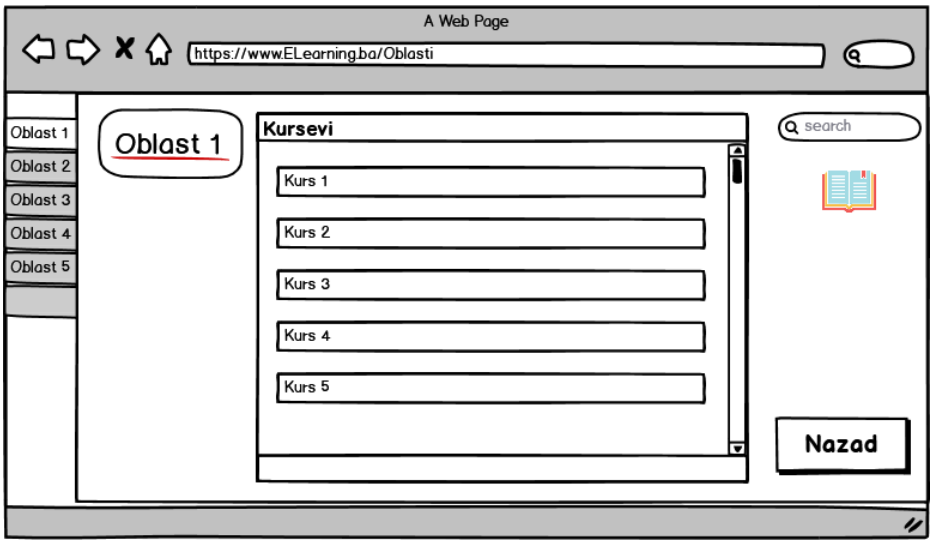
Slika 4.2. Regsitracija

Slika 4.3. Prijava

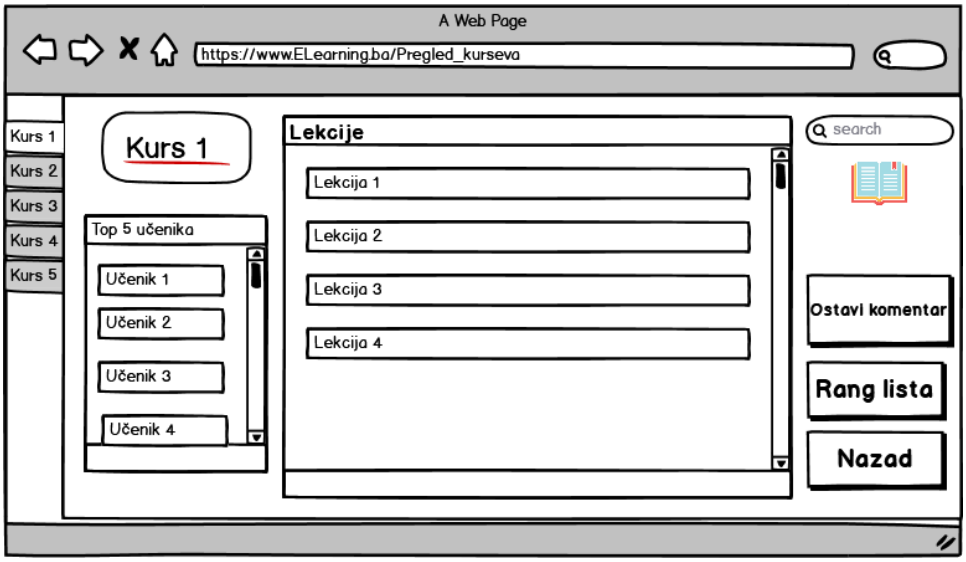


Slika 4.4. O aplikaciji

Ova aplikacija je zamišljena da se sastoji od više oblasti, a te oblasti se sastoje od kurseva. Kursevi se sastoje od lekcija kao što se iz priložene slike ispod može vidjeti, a također, postoji i prozorčić sa top 5 korisnika svakog kursa.

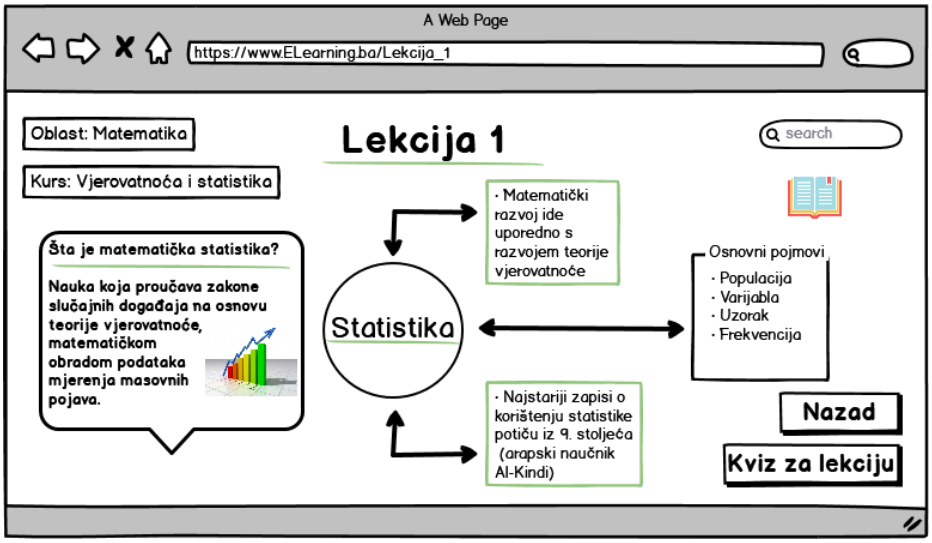


Slika 4.5. Oblasti



Slika 4.6. Kursevi

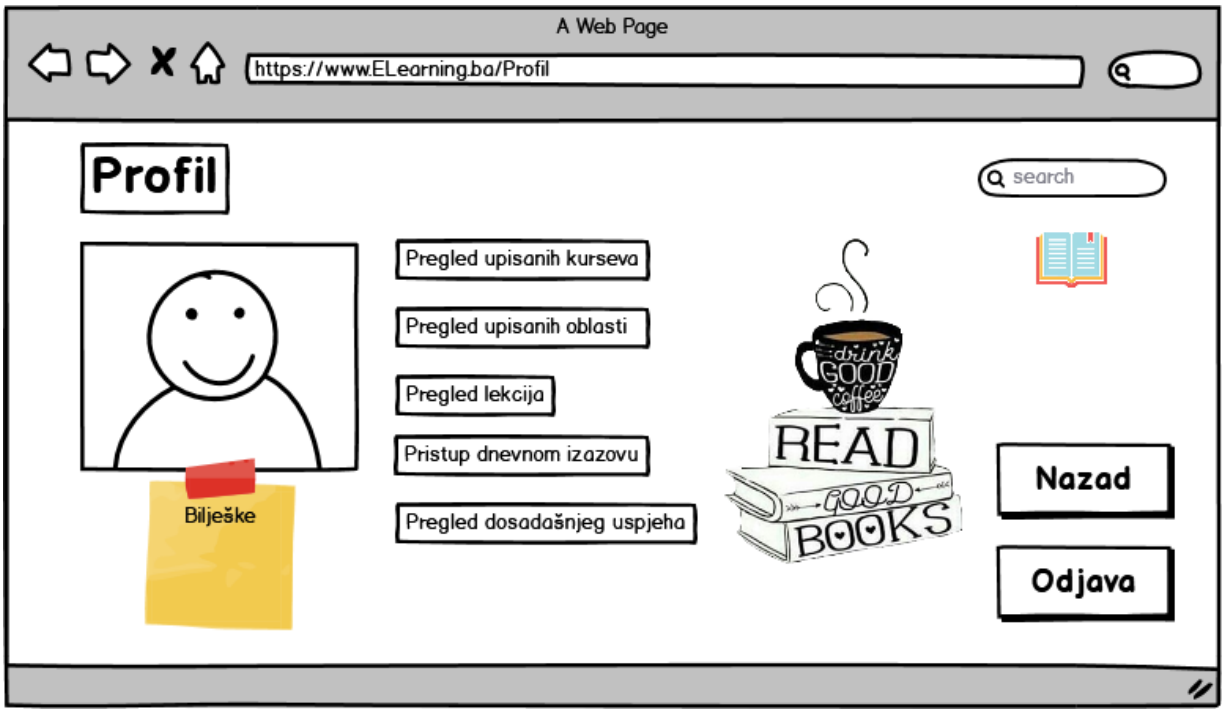
Na narednoj slici može se vidjeti prikaz lekcije. Aplikacija će sadržati kviz za lekciju.



Slika 4.7. Lekcija

Slika 4.8. Kviz za lekciju

Korisnik će na svom profilu imati pregled upisanih kurseva i oblasti, pregled lekcija, pristup dnevnom izazovu te pregled dosadašnjeg uspjeha. Klikom na dugme pristup dnevnom izazovu, pristupa se kvizu iz svih kurseva koji se pohađaju.



Slika 4.9. Pregled profila

Slika 4.10. Dnevni izazov

5. DIZAJN KLASA I SOLID PRINCIPI

5.1. Dizajn klasa

Klasa	**Korisnik**
Atributi	id: int ime: String prezime: String email: String kursevi: List<Kurs>
Metode	upisiKurs(Kurs k): void ostaviOcjenu(Kurs k, int o): void ostaviKomentar(Kurs k, String s): void zapocniKviz(Kurs k): Kviz zapocniKviz(Lekcija l): Kviz odjava(): void
Opis	Atributi klase predstavljaju osnovne informacije o registrovanom korisniku, kao što su lične informacije poput imena i prezimena,te kursevi koje pohađaju. Korisnici mogu upisati novi kurs, ocijeniti i komentarisati kurs koji pohađaju, te pristupiti kvizu, bilo to iz pojedinačne lekcije nekog kursa ili cijelog kursa što smo kroz metode implementirali. Također, korisnik se može odjaviti sa svog računa. Ova klasa je apstraktna.
Naslijeđene klase	Nije naslijeđena klasa.

Klasa	**Posjetilac**
Atributi	guestId: int
Metode	prijaviSe(): void registracija(): void
Opis	Posjetilac je klasa koja modelira posjetioca koji nema nikakvu značajnu funkciju, osim u pregledu aplikacije. Dodijeljen mu je id – broj, samo kako bismo razlikovali posjetioce.
Naslijeđene klase	Nije naslijeđena.

Klasa	**KorisnikSaFakultetskimMailom**
Atributi	
Metode	
Opis	Kako je klasa naslijeđena, ona ima sve atribute i metode kao i klasa Korisnik. Posjetilac pri registraciji upisuje svoj e-mail, koji se provjerava u klasi „KlasaZaVerifikacijuFakultetskogMaila“, te ako klasa utvrdi da e-mail koji je posjetilac napisao jeste fakultetski, kreira se objekat tipa KorisnikSaFakultetskimMailom.
Naslijeđene klase	Ova klasa je naslijeđena iz klase Korisnik.

Klasa	**KorisnikBezFakultetskogMaila**
Atributi	
Metode	
Opis	Klasa je naslijeđena, te ona ima sve atribute i metode kao i klasa Korisnik. Posjetilac pri registraciji upisuje svoj e-mail, koji se provjerava u klasi „KlasaZaVerifikacijuFakultetskogMaila“, te ako klasa utvrdi da e-mail koji je posjetilac napisao NIJE fakultetski, kreira se objekat tipa KorisnikBezFakultetskogMaila.
Naslijeđene klase	Ova klasa je naslijeđena iz klase Korisnik.

Klasa	**Administrator**
Atributi	
Metode	prikaziStatistiku(): void
Opis	Administrator predstavlja osobu koja ima sve funkcionalnosti kao i klasa „Korisnik“ , ali pored toga može pregledati statistiku svih korisnika aplikacije.

Naslijeđene klase	Klasa „Administrator “ je naslijeđena klasa iz klase „Korisnik“

Klasa	**Kurs**
Atributi	naziv: String lekcije: List<Lekcija> upisani: List<Korisnik>
Metode	upisiKurs(Korisnik k): void dodajLekciju(Lekcija l): void dajLjestvicu(): Ljestvica
Opis	Klasa koja modelira skup lekcija. Atribut „naziv“ predstavlja naziv kursa, „lekcije“ – skup lekcija koji dati kurs sadrži, „upisani“ – skup svih korisnika koji su upisani na dati kurs.
Naslijeđene klase	Nije naslijeđena.

Klasa	**Lekcija**
Atributi	naziv: String put: String
Metode	prikaz(): void
Opis	Lekcija predstavlja klasu koja omogućava pregled lekcije. Atribut „naziv“ predstavlja naziv lekcije kojoj korisnik pristupa. Kako je lekcija jedna datoteka, potreban nam je pristupni put do te datoteke, što i sam atribut „put“ predstavlja. Metoda „prikaz“ prikazuje datu lekciju.
Naslijeđene klase	Nije naslijeđena.

Klasa	**Oblast**
Atributi	naziv: String kursevi: List<Kurs>
Metode	dodajKurs(Kurs k): void

Opis	Klasa koja modelira oblast kojoj korisnik pristupa. Služi da bi se aplikacija prilagodila korisnikovom iskustvu, te kako bi korisnik lakše pronašao kurseve njegovog interesa. Atributi klase su naziv oblasti, te svi kursevi koji pripadaju toj oblasti.
Naslijeđene klase	Nije naslijeđena klasa.

Klasa	**Kviz**
Atributi	pitanja: List<Pitanje>
Metode	predaj(): int dajBrojTacnoOdgovorenih(): int
Opis	Klasa koja modelira vid provjere znanja stečenog pohađanjem kursa/lekcije na aplikaciji. Atributi klase uključuju listu pitanja koja se nalaze u kvizu. Metoda „predaj“ podrazumijeva završetak kviza, te kroz „dajBrojTacnoOdgovorenih“ korisnik ima uviđaj koliko je tačnih odgovora imao.
Naslijeđene klase	Nije naslijeđena klasa.

Klasa	**Pitanje**
Atributi	pitanje: String odgovori: List<String> tacanOdgovorIndeks: int trenutniOdgovor: int
Metode	provjeri(): bool
Opis	Klasa koja modelira pitanja koja se pojavljuju u određenom kvizu. Atributi klase podrazumijevaju postavku – atribut „naziv“, niz ponuđenih odgovora (ako ih ima) – atribut „odgovori“, te indeks tačnog odgovora i trenutnog odgovora korisnika kroz atribute „tacanOdgovorIndeks“ i „trenutniOdgovor“. Metoda „provjeri“ poredi odgovor koji je tačan tj. njegov indeks i odgovor koji je korisnik stavio.

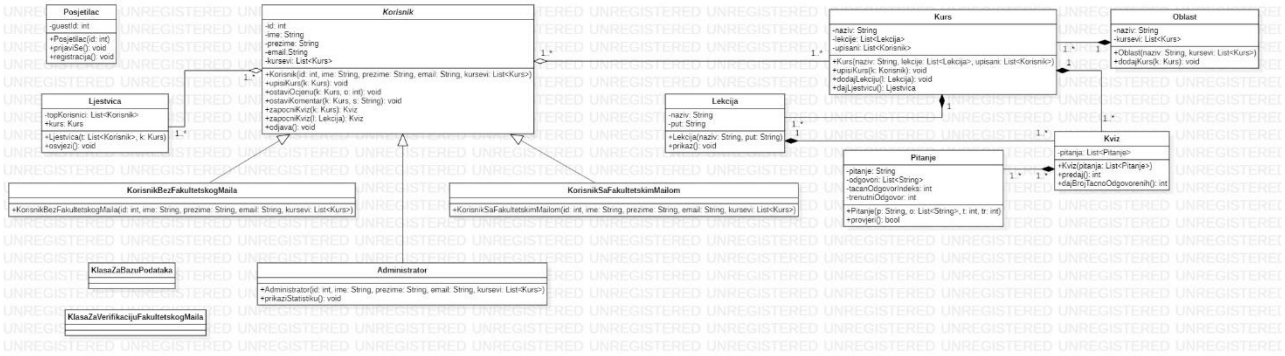
	Vraća true/false u ovisnosti je li tačno/netačno.
Naslijeđene klase	Nije naslijeđena klasa.

Klasa	**Ljestvica**
Atributi	topKorisnici: List<Korisnik> kurs: Kurs
Metode	osvjezi(): void
Opis	Klasa „Ljestvica“ predstavlja najbolje rangirane korisnike sortirajući ih u opadajući redoslijed. Atribut „topKorisnici“ predstavlja listu najuspješnijih korisnika, „kurs“ će predstavljati za koji kurs želimo rang ljestvicu s obzirom da svaki kurs ima listu najuspješnijih takmičara. Metoda „osvjezi“ zapravo ažurira podatke ukoliko je neki korisnik popravio svoj rang.
Naslijeđene klase	Nije naslijeđena.

Pošto još ne znamo kako radi integracija sa bazom podataka i višemrežnim aplikacijama, ne znamo sve potrebene atribute i metode. To ćemo nadograditi u skladu sa predavanjima i tutorijalima koje budu te teme obrađivali, a također i opis kako sve treba funkcionisati. Za sada znamo samo da je potrebno, pa ćemo ovdje navesti dvije tabele.

Klasa	**KlasaZaBazuPodataka**
Atributi	
Metode	
Opis	
Naslijeđene klase	

Klasa	**KlasaZaVerifikacijuFakultetskogMaila**
Atributi	
Metode	
Opis	
Naslijeđene klase	



Slika 5.1.1. Dijagram klasa

5.2. SOLID principi

Princip S

„ Korisnik“ i naslijeđene klase iz nje imaju mogućnost upisivanja u kurs,

pokretanja kvizova i ostavljanja recenzija, što bi se moglo riješiti odgovarajućim interfejsima, te i naslijeđene klase ne zadovoljavaju ovaj princip. Klasa „Kurs“ isto tako upisuje korisnike u kurs, dodaje lekcije i vrši pokretanje kvizova.

Dakle, klasa „Kurs“ i naslijeđene klase iz nje narušavaju ovaj princip.

Princip O

Jedina klasa koja bi mogla kršiti princip O jeste klasa „Ljestvica“ jer jedina ima atribut tipa druge klase, ali pošto taj atribut služi samo za identifikaciju kursa, a ne sa radom sa tim kursom, ona ne krši princip O.

Princip L

Princip L nije ispravno implementiran u klasi „Administrator“ s obzirom da je ona naslijeđena iz klase „Korisnik“, a ne može se zamijeniti sa baznom klasom. Kod ostalih naslijeđenih klasa je zadovoljen princip.

Princip I

Trenutno nemamo nikave interfejse pa nemamo interfejs koji može da krši ovaj princip.

Princip D

Što se tiče principa D, jedina nasljeđivanja koja imamo su iz klasa „Korisnik“ i

„Kviz“ koje su planirane da budu omotač koji sadrži potrebne atribute i funkcionalnosti, te naš sistem se pridržava principa D.

6. PRIMJENA PATERNA

6.1. Strukturalni paterni

Adapter patern

Osnovna namjena ovog paterna je da omogući širu upotrebu već postojećih klasa. Ovaj patern se koristi kada ne želimo mijenjati već postojeću klasu, ali potreban nam je drugačiji interfejs. Konkretno u našem primjeru, ovaj patern nigdje ne možemo primijeniti s obzirom da nema potrebe za ikakvom konverzijom jedne klase u neku drugu. Svaka je precizno definisana i ima svoju ulogu.

Bridge patern

Bridge patern se koristi da omogući odvajanje implementacije i apstrakcije neke klase kako bi ta klasa mogla posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije.

U našoj aplikaciji Bridge patern nema potrebe da se koristi s obzirom da kod naslijeđenih klasa iz „Korisnik“, tj. kod „KorisnikSaFakultetskimMailom“, „KorisnikBezFakultetskogMaila“ i „Administrator“ nema potrebe za ovom vrstom paterna jer se svi korisnici upisuju na kurs, ostavljaju ocjenu, započinju kviz na isti način.

Composite patern

Koristi se kada svi objekti imaju različite implementacije nekih metoda, ali im je svima potrebno pristupiti na isti način, te se na taj način pojednostavljuje njihova implementacija. Composite patern služi za kreiranje hijerarhije objekata. Vrlo slično Bridge paternu, ovaj šablon nema potrebe da se koristi s obzirom da svi korisnici obavljaju metode na identičan način.

Decorator patern

Osnovna namjena ovog paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja postojećim objektima. Umjesto definisanja velikog broja izvedenih klasa, dovoljno je omogućiti različito dekoriranje objekata. Kod nas, pošto korisnik nema opciju da uređuje objekte, nema potrebe za ovim paternom.

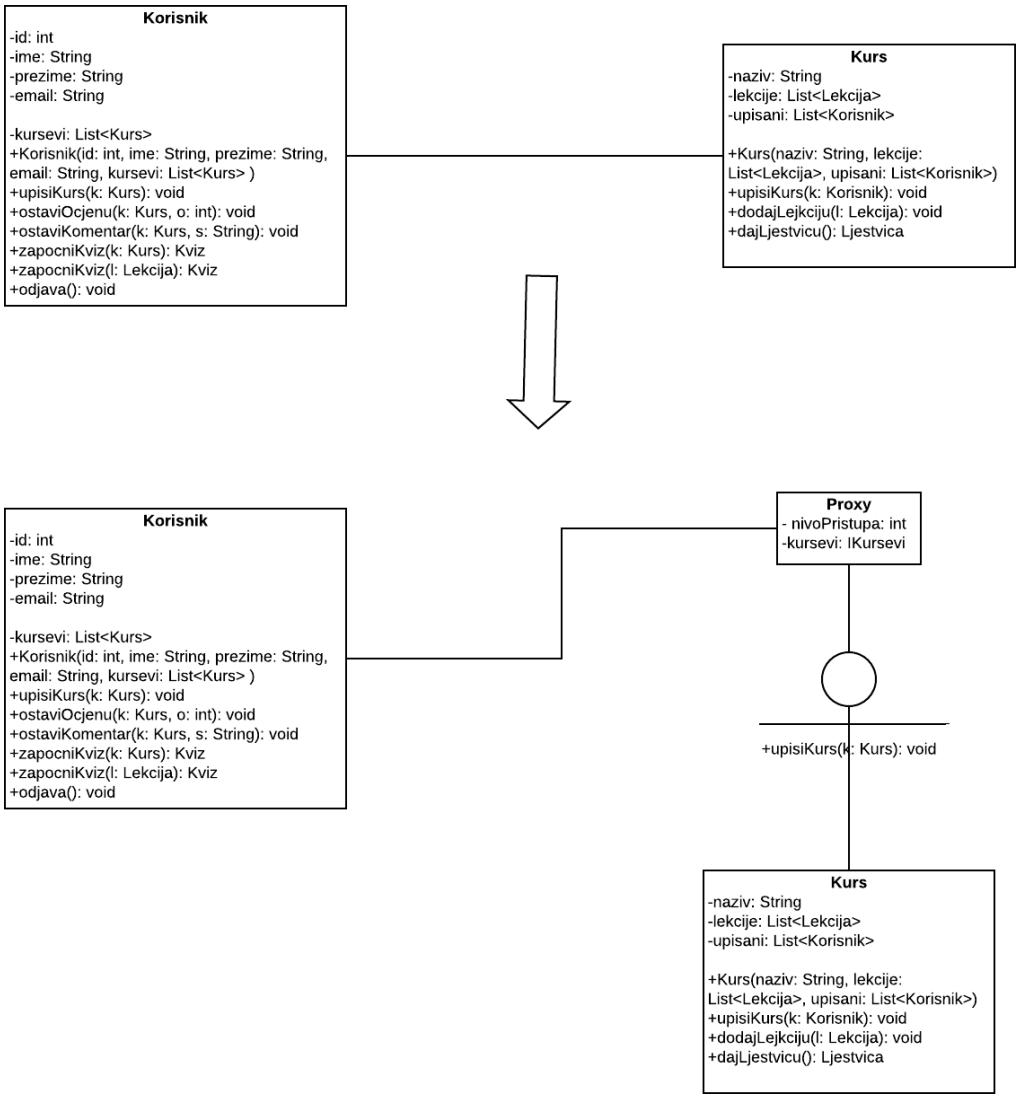
Facade patern

Ovaj patern se koristi kada sistem ima više podsistema pri čemu su apstrakcije i implementacije usko povezane. Facade patern služi kako bi pojednostavio klijentima korištenje kompleksnih sistema. Klijenti vide samo krajnji izgled objekta. Pošto su sve funkcionalnosti koje klijent radi zasebne, i svaka ima svoju svrhu, fasadu ovdje nije potrebno postavljati.

Proxy patern

Proxy patern služi za dodatno osiguravanje objekata od pogrešne ili zlonamjerne upotrebe. Namjena ovog paterna je da omogući pristup i kontrolu pristupa stvarnim objektima. Obično je mali javni surogat objekat. U našoj aplikaciji, prije svega je potrebno provjeriti da li neki korisnik ima pristup kursu, i ako nema taj kurs mu se treba onemogućiti, tako da se ovaj patern može primijeniti.

Na slici smo prikazali klase „Korisnik“ i „Kurs“, kao i njihovu vezu prije i poslije upotrebe proxy paterna:



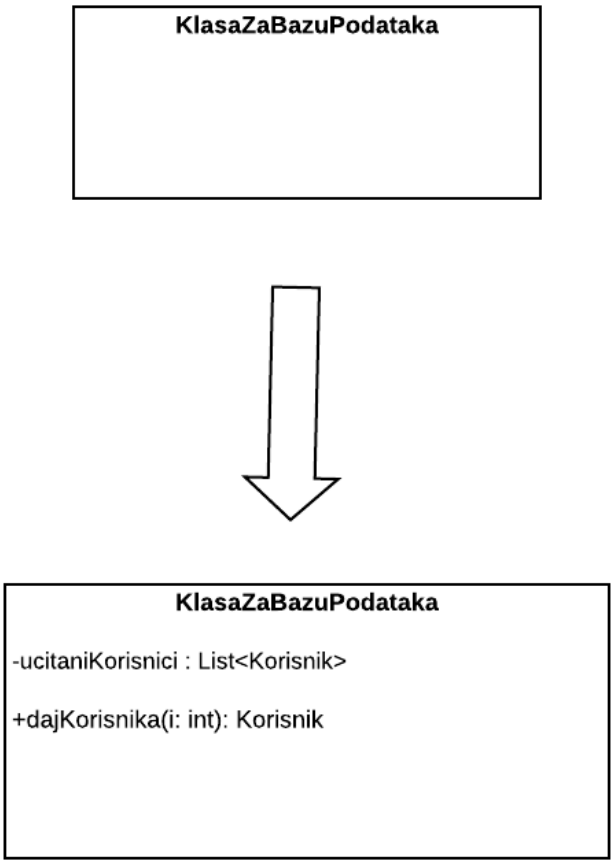
Slika 6.1.1. Implementacija proxya

Flyweight patern

Flyweight patern se koristi kako bi omogućio da više različitih objekata dijele isto glavno stanje, a imaju različito sporedno stanje.

U bazi podataka smo dodali listu: ucitaniKorisnici: List<Korisnik> kao atribut, a dajKorisnika(i: int): Korisnik je dodana metoda. Ta metoda provjerava da li je korisnik ranije

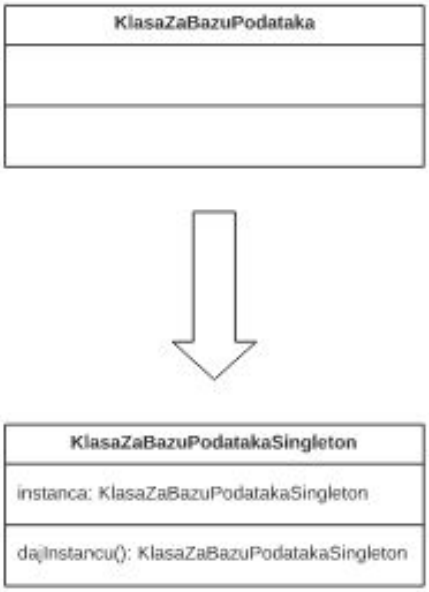
učitan iz baze podataka. Ako jeste učitan, tada postoji objekat tog korisnika i onda daje taj objekat, a ako nije, onda ga učitava, stavlja u listu i vraća kao rezultat.



Slika 6.1.2. Implementacija flyweighta

6.2. Kreacijski paterni

Singleton patern služi kako bi se neka klasa mogla instancirati samo jednom. Na ovaj način može se omogućiti i instantacija klase tek onda kada se to prvi put traži. Osim toga, osigurava se i globalni pristup jedinstvenoj instanci te iste klase Što se tiče singleton paterna idealna klasa na kojoj se može primijeniti je "KlasaZaBazuPodataka" jer nam treba samo jedna takva klasa, odnosno stvaranje više instanci te klase je suvišno, jer jednom kad izvršimo konekciju sa bazom, ne moramo to više raditi. Sljedeća slika prikazuje kako bi se implementirao ovaj patern u našem projektu:



Slika 6.2.1. Singleton patern

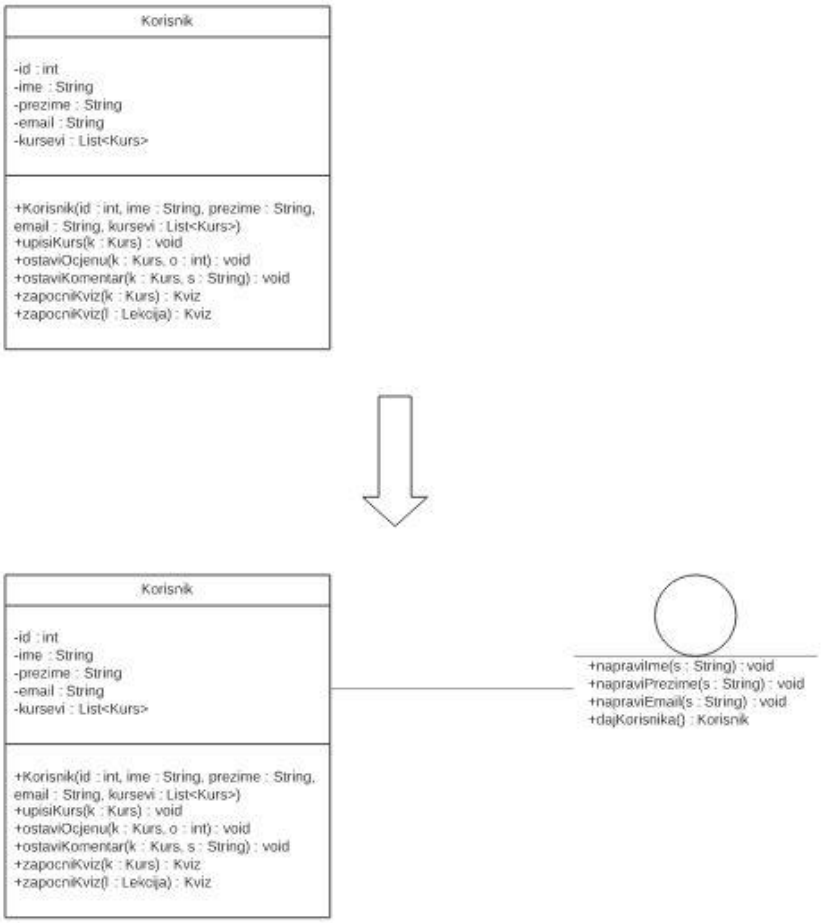
Jednostavno rečeno **Prototype patern** omogućava smanjenje kompleksnosti kreiranja novog objekta tako što se uvodi operacija kloniranja.
U našem trenutnom stanju dijagrama klasa, ne dolazi do potrebe da se išta treba klonirati, odnosno ne treba da dođe do potrebe da se prave identične duboke kopije drugih objekata. Eventualno u slučaju da se uvede neka dodatna funkcionalnost, kao što je na primjer ponavljanje kviza sa istim pitanjima. U tom slučaju bi se mogao klonirati početni kviz i na toj kopiji izvršiti premještanje pitanja i odgovora u pitanjima.

Factory method patern služi za omogućavanje instanciranje različitih vrsta podklasa koristeći factory metodu koja odlučuje koja će se podklasa instancirati i koja programska logika izvršiti.
Jedina klasa koja je upitna kada dolazi do ovog paterna je klasa "Korisnik", jer se jedino u njoj dešavanasljeđivanje. Jedine klase koje koriste korisnika su ljestvica i kurs, ali one imaju identičnu implementaciju za sve klase. Možda bi se moglo diskutovati da provjeravanje pristupa nekom kursu može da se riješi ovim paternom, ali to je posao proxy paterna. Eventualno da program treba da različito funkcioniše za različite klase, na primjer da obični korisnici mogu da uzmu određen broj kurseva dostupnih samo fakultetskim korisnicima pa da se nekako treba računati broj 'premium' kurseva koje mogu uzeti.

Abstract Factory patern se koristi ukoliko postoji više tipova istih objekata te različite klase koriste različite podtipove, te klase postaju fabrike za kreiranje objekata zadanog podtipa bez potrebe za specificiranjem pojedinačnih objekata.
Opet i ovaj problem zbog nedostatka uslova za provjeru vrsta instanci nije potreban. Promjena u dijagramu kada bi mogao biti primjenjen ovaj patern je da klasa "Kviz" ima dvije podklase,

za obični kviz i za ispit, pri čemu studenti mogu samo raditi ispite, a obični korisnici samo obične kvizove.

Builder patern služi za apstrakciju procesa konstrukcije objekta, kako bi se kao rezultat mogle dobiti različite specifikacije objekta koristeći isti proces konstrukcije.
Ovaj patern bi se mogao koristiti kod klase "Korisnik" koja ima konstruktor koji prima veći broj parametara, a primjenom ovog paterna ne mora da obavi dodjeljivanje u konstruktoru odjednom.
Još jedna olakšavajuća stvar je što olakšava situacije kada npr. korisnik unese pogrešnu e-mail adresu, primjenom ovog paterna ne mora ponovo da definiše ime, prezime i slično, već samo definiše ponovo e-mail. Ovo također olakšava postupak dodavanja novih atributa ovoj istoj klasi.
Sljedeća slika prikazuje kako bi se implementirao ovaj patern u našem projektu:



Slika 6.2.2. Builder

6.3. Paterni ponašanja

Strategy patern

Strategy patern izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Pogodan je kada postoje različiti primjenjivi algoritmi za neki problem. Kada bi se različite metode klase

„Korisnik“ drugačije odvijale za sve podklase, tada bi se ovaj patern mogao iskoristiti. Kako to nije slučaj kod nas, i kako nema drugih nasljeđivanja primjena ovog paterna bi bila suvišna.

State patern

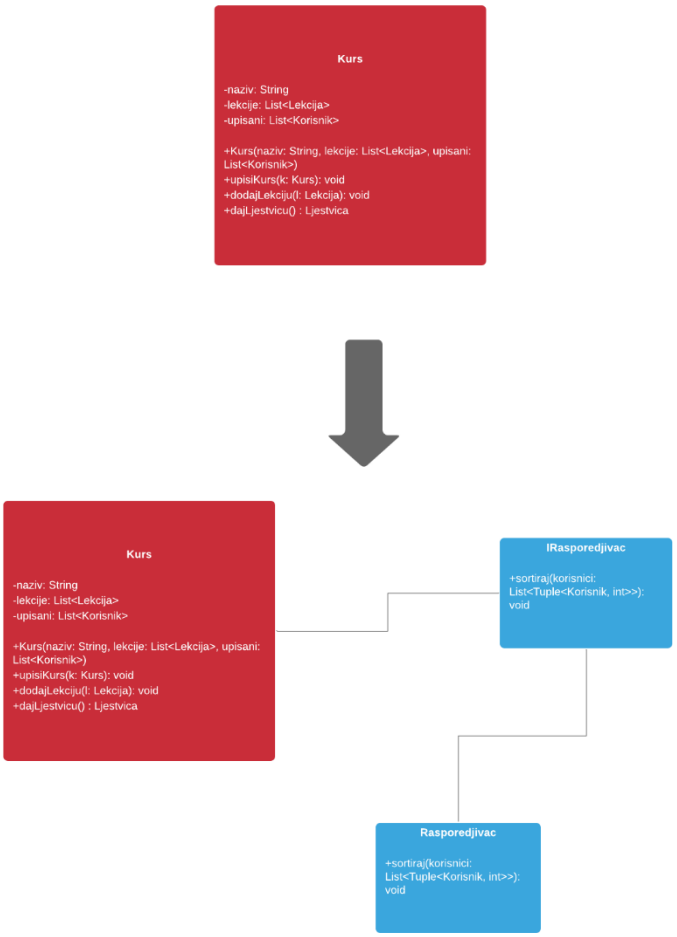
State patern je dinamička verzija Strategy paterna, što bi značilo da mijenja način ponašanja na osnovu trenutnog stanja. U našem projektnom zadatku se ovaj patern ne primjenjuje s obzirom da nemamo situaciju da objekat mijenja način ponašanja, tj. nema promjene stanja. Kada se posjetilac uloguje, on ne mijenja stanje u neko drugo. Također, isto vrijedi i prilikom registracije i prijave.

Observer patern

Observer patern uspostavlja relaciju između objekata. Kada jedan objekat promijeni stanje, drugi zainteresirani objekti se obavještavaju. Ovaj patern smo mogli primijeniti na naš projektni zadatak, ali zbog potreba tutorijala nismo. Slučaj u kojem bi se mogao primijeniti je ukoliko bi se korisnikovo stanje na ljestvici promijenilo, on bi bio obaviješten.

Template Method patern

Ovaj patern omogućava izdvajanje određenih koraka algoritama u odvojene podklase. U našem primjeru konkretno, mi smo primijenili ovaj patern koristeći metodu „dajLjestvicu(): Ljestvica“ kao TemplateMethod. Operacija koja se primjenjuje u Iraporedjivac-u i Rasporedjivac-u je „sortiraj(korisnici: List<Tuple<Korisnik, int>>): void“, gdje argument funkcije koja sortira predstavlja listu uređenih parova koji predstavljaju korisnika zajedno sa njegovim poenima. Na slici ispod vidimo stanje prije i poslije upotrebe ovog paterna:

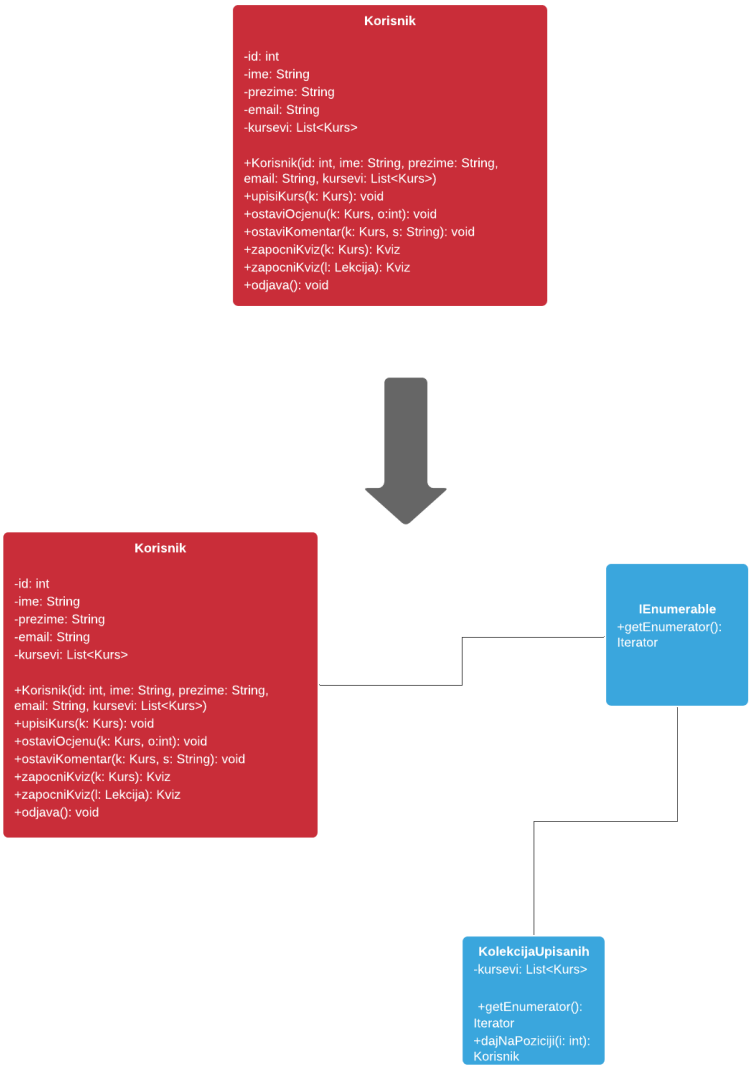


Slika 6.3.1. Primjena Template Method paterna na projektni zadatak

Iterator patern

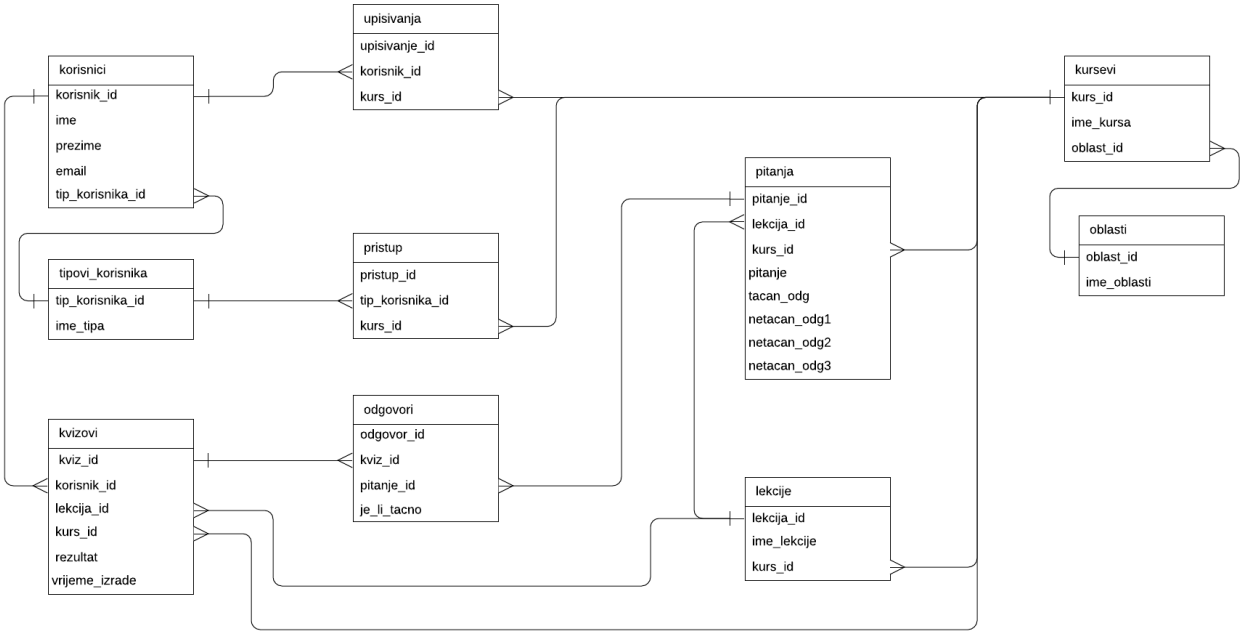
Uloga ovog paterna je da omogući sekvencijalni pristup elementima kolekcije bez poznavanja kako je kolekcija strukturirana. Interfejs koji smo dodali je IEnumerable koji sadrži metodu „getEnumerator(): Iterator“, a kolekcija „KolekcijaUpisanih“ nasljeđuje tu klasu. Naime, primarni cilj primjene ovog paterna jeste da imamo iterativno idemo kroz upisane kurseve jednog korisnika.

Na slici smo prikazali primjenu ovog paterna:

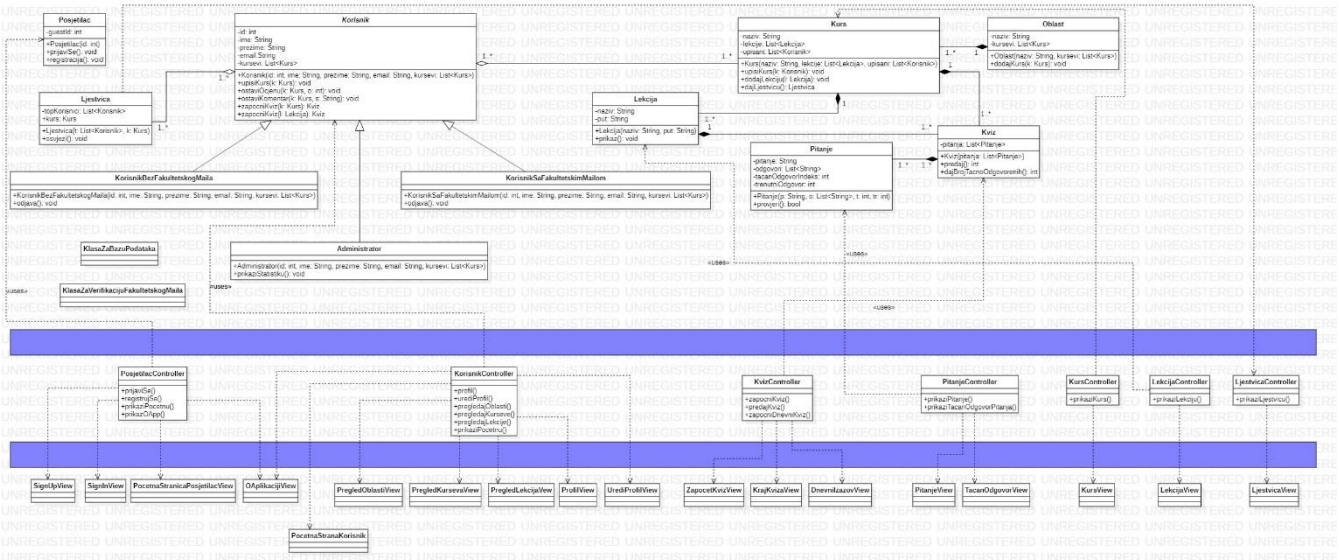


Slika 6.3.2. Iterator patern

7. ERD i MVC dijagrami

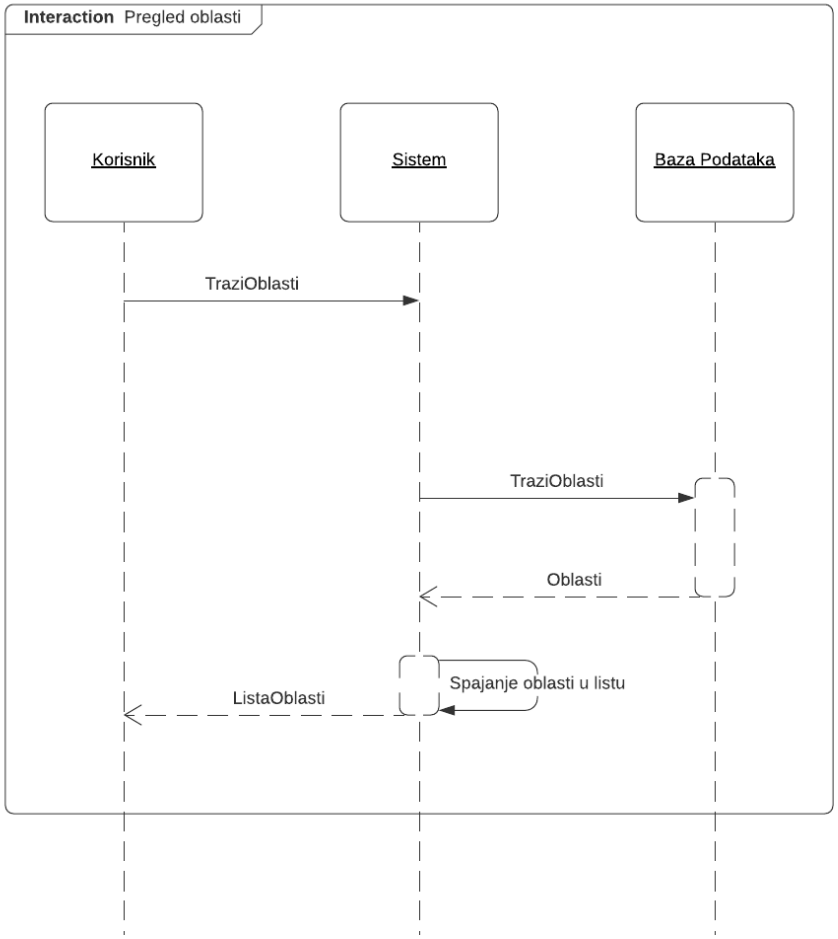


Slika 7.1. ERD

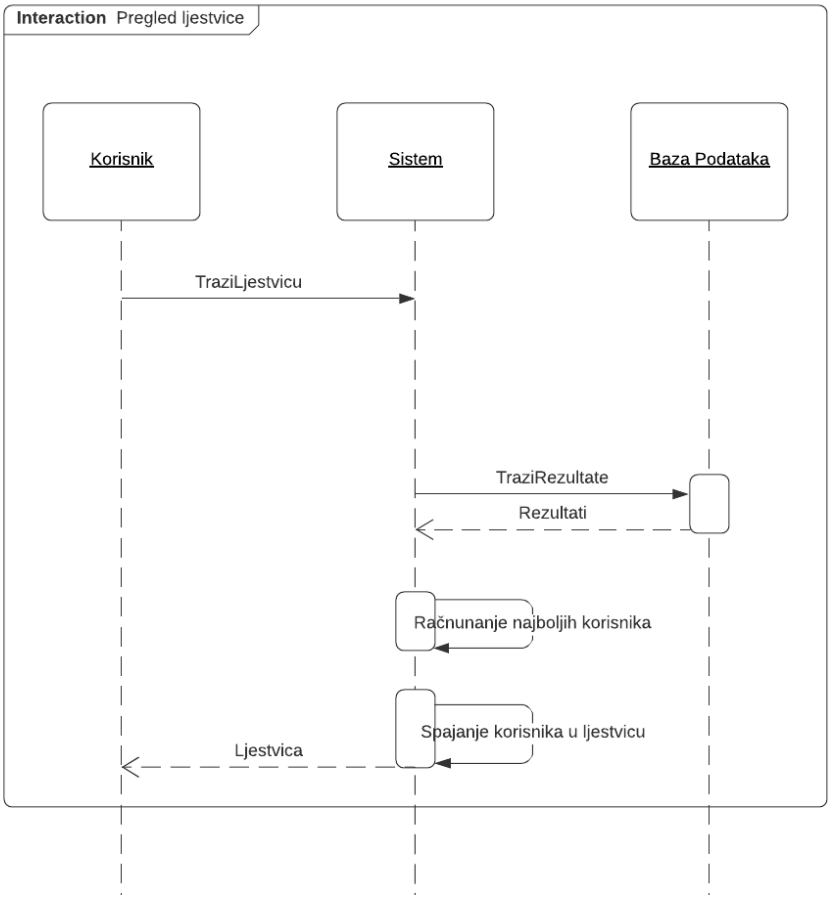


Slika 7.2. MVC

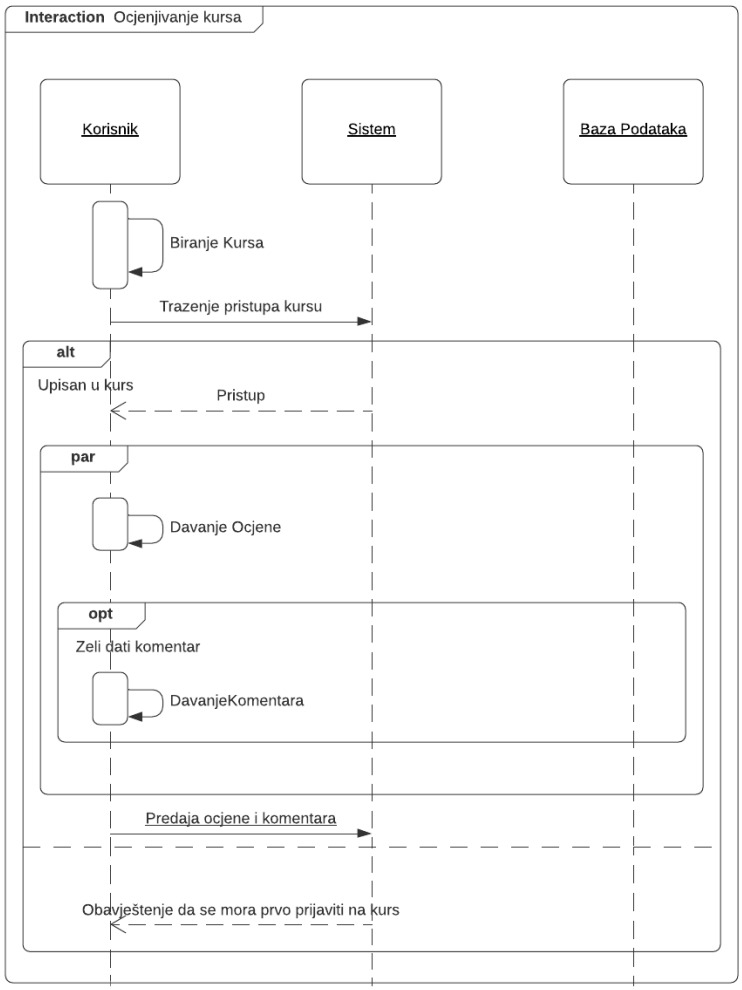
8. DIJAGRAM SLOŽENE STRUKTURE, KOMPONENTI I SEKVENCI



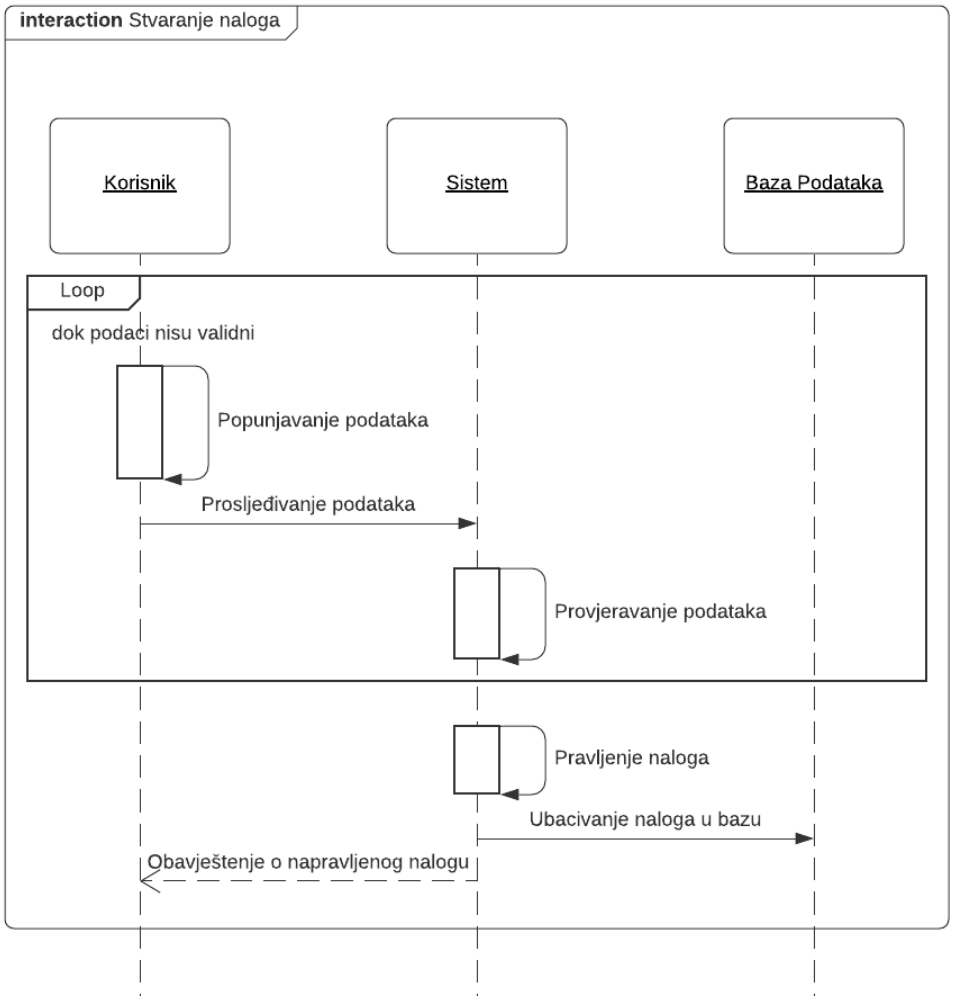
Slika 8.1. Dijagram sekvenci 1



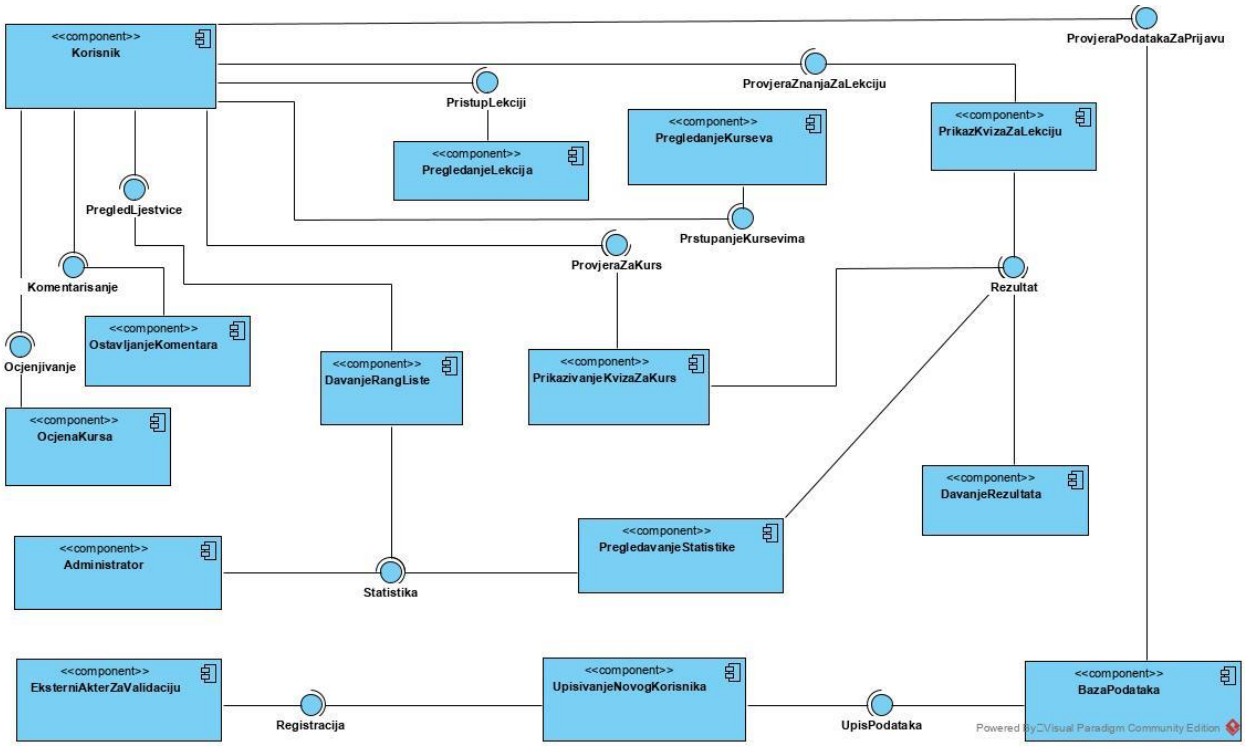
Slika 8.2. Dijagram sekvenci 2



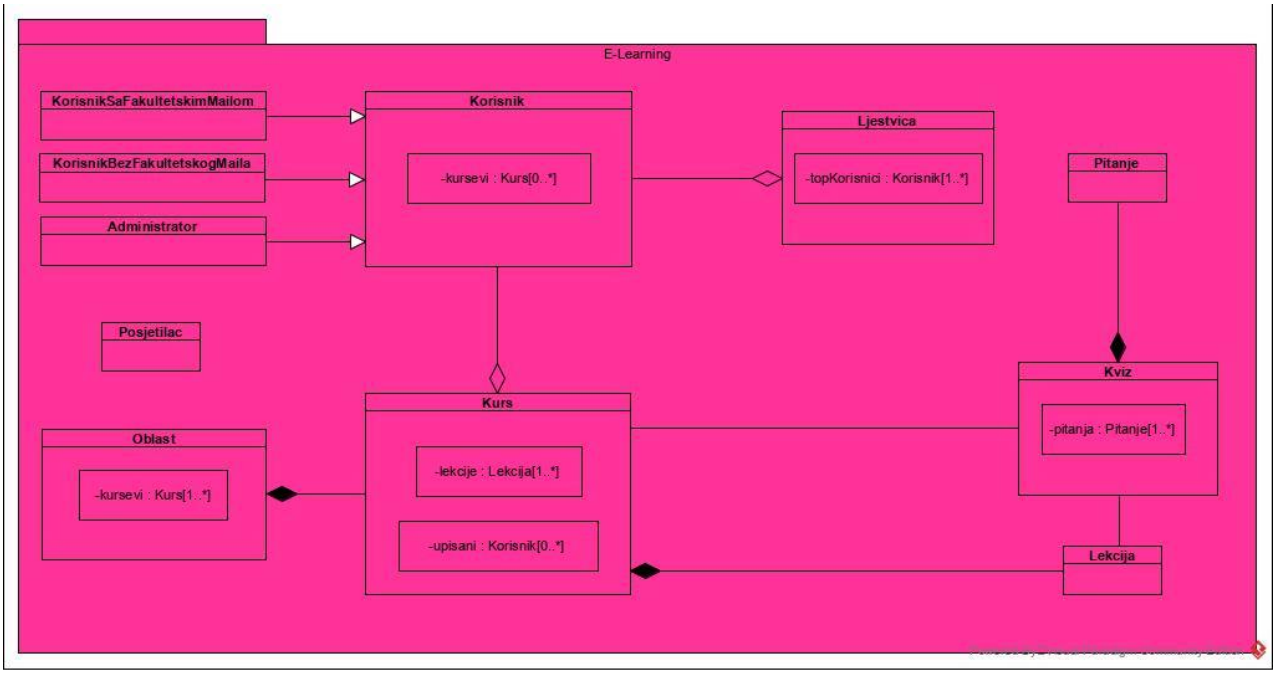
Slika 8.3. Dijagram sekvenci 3



Slika 8.4. Dijagram sekvenci 4



Slika 8.5. Dijagram komponenti



Slika 8.6. Dijagram složene strukture