PANEVROPSKI UNIVERZITET APEIRON, BANJA LUKA
FAKULTET INFORMACIONIH TEHNOLOGIJA

Student: Ivan Pavlović
Broj indeksa: 92-20 / RITP-S

Projektovanje informacionih sistema
("Note Me" Application)

Mentor: Prof. dr Branko Latinović

Banja Luka, decembar 2021.

SADRŽAJ

UVOD

Ovaj seminarski rad se radi o mojoj ideji koju već duži niz godina planiram ostvariti i na neki način ovaj predmet me je motivisao da ju konačno ostvarim i postavim na internet.

Ideja aplikacije je da korisnicima pružim što sigurniju i povoljniju kupovinu odjeće i obuće poznatih marki, također u aplikaciji korisnik će moći da kreira nalog (koji će čuvati podatke na serveru) i aplikacija će da sadrži poseban algoritam pretraživanja koji će se ažurirati posebno po svakom korisniku.

Prethodno navedene funkcije bi predstavljale prvu verziju aplikacije (v1.0), i nakon mog usavršavanja sa tehnologijoma planiram aplikaciju postaviti na stepenicu više, tako što bih dodao mnogo novih funkcionalnosti i također želim svoju aplikaciju da bude dostigne globalni nivo.

Za realizovanje ove aplikacije koristiću **U**nified **M**odelling **L**anguage (UML).

U samom početku mog seminarskog rada objasniću šta je to UML i koja je njegova primjena.
Također ću nabrojati osnovne dijagrame koji se nalaze u sklopu UML jezika i objasniti njihovu glavnu primjenu.

Što se tiče praktičnog dijela u detalje ću objasniti funkcionalnosti moje aplikacije i zašto sam koristio određene dijagrame u projektovanju.

1. UML

„UML, skraćenica od Unified Modelling Language je standardizovani jezik za modeliranje koji se sastoji od integrisanih skupova dijagrama, razvijen kako bi pomogli sistemskim i softverskim programerima za specifikaciju, vizuelizaciju, konstruisanje i dokumentovanje softverskih sistema, kao i za poslovno modeliranje i druge sisteme.

UML ima veoma jaku ulogu u razvoju objekto orijentisanog softvera i procesa razvoja softvera“ [1].

„Stvaranje UML-a je prvobitno bilo motivisano željom da se standardizuju različiti notacijski sistemi i pristupi dizajnu softvera“ [2].

„UML koristi uglavnom grafičke zapise da izrazi dizajn softverskih projekata, također pomaže projektnim timovima da komuniciraju, istražuju potencijalne dizajne i validiraju konačni dizajn softvera“ [1].

Sistemski analitičar (eng. System Analyst) je stručnjak za informacione tehnologije (IT) koji se bavi dizajnom, analizom i implementacijom informacionih sistema, koje se prvenstveno zasnuje na UML jeziku.

Slika 1- UML Logo

1.1. Osnovni elementi jezika UML

„Osnovni gradivni elementi jezika UML su: stvari (things), relacije (relationships) i dijagrami (diagrams). Ove elemente ćemo samo nabrojati i formalno kratko definisati, jer cilj ovog uvoda nije da se razumije značenje i način upotrebe ovih elemenata. Oni će biti detaljno objašnjeni kasnije, kako se bude ukazivala potreba za njima. Ovo je samo okvirni uvodni pregled elemenata jezika UML“ [3].

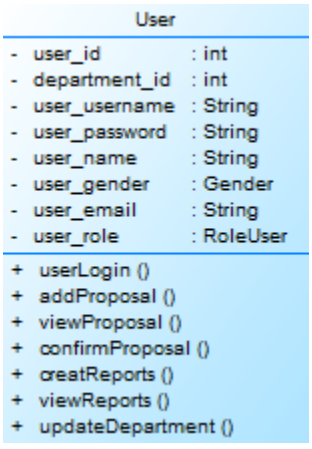
1.1.1. Stvari (things) i njihova podjela

„Postoje sljedeće grupe stvari:

- strukture stvari (structural things);
- stvari ponašanja (behavioral things);
- stvari grupisanja (grouping things);
- stvari označavanja (annotation things);

Strukture stvari (structural things)
Strukture stvari su apstrakcije koje čine strukturni, uglavnom statički dio modela. To su:

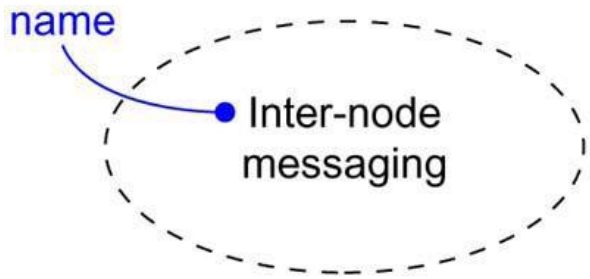
Klasa (class) je opis skupa objekata koji imaju iste atribute, operacije, relacije i sematiku.



Slika 2 - Primjer Klase

Interfejt (interface) je skup operacija koje definišu usluge klase ili komponente.

Kolaboracija (collaboration) je skup uloga i drugih elemenata koji sarađuju da bi ispunili kooperatvno ponašanje složenije od proste sume pojedinačnih ponašanja elemenata.



Slika 3 - Primjer Kolaboracije

Slučaj upotrebe (use case) je opis skupa sekvenci akcija koje sistem izvršava da bi proizveo spolja uočljivo ponašanje bitno za nekog aktera.

Slika 4 - Primjer Slučaja upotrebe

Aktivna klasa (active class) je klasa čiji objekti posjeduju jedan ili više procesa ili niti toka kontrole.

Komponenta (component) je fizički i zamjenljivi dio sistema koji zadovoljava i realizuje skup interfejsa.

Čvor (node) je fizički element koji postoji u vrijeme izvršavanja i predstavlja računarski resurs koji, u principu, posjeduje memoriju i najčešće mogućnost obrade.

Stvari ponašanja (behavioral things)
Stvari ponašanja opisuju ponašanje sistema, tj. čine dinamički dio modela:

Interakcija (interaction) je ponašanje koje uključuje skup poruka koje se razmjenjuju između objekata u određenom kontekstu, da bi se ostvarila određena svrha.

Mašina stanja (state machine) je ponašanje koje definiše sekvencu stanja kroz koje objekti ili interakcija prolazi tokom svog života, kao posljedica događaja, zajedno sa reakcijama na te događaje.

Stvari za grupisanje (grouping things)
Stvari za grupisanje su organizacioni dijelovi jezika. Postoji samo jedna stvar za grupisanje – paket (package). Paket je čisto konceptualna stvar u koju se grupišu svi ostali elementi jezika.

Stvari za označavanje (annotation things) predstavlja komentare (note)“ [3].

1.1.2. Relacije (relations)

„Postoje četiri vrste relacija u jeziku UML:

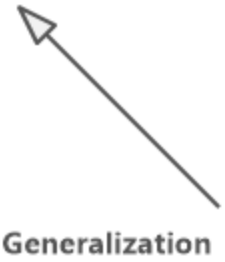
Zavisnost (dependency) je semantička relacija između dvije stvari koja označava da se promjena jedne (nezavisne) stvari odslikava na drugu (zavisnu) stvar.

Slika 5- Relacija zavisnosti

Asocijacija (association) je strukturna relacija koja opisuje skup veza, pri čemu veza povezuje instance.

Slika 6 - Relacija asocijacije

Generalizacija (generalization) je relacija generalizacije/specijalizacije kod koje objekte generalizovanog elementa (roditelja – parent) mogu zamijeniti objekti specijalizovanog elementa (potomka – child).



Slika 7 - Relacija generalizacije

Realizacija (realization) je sematička veza u kojoj je jedan element ugovor koji drugi element ispunjava“ [3].

Slika 8- Relacija realizacije

2. DIJAGRAMI

Dijagram je informacija predstavljena pomoću vizuelizacijskih tehnika odnosno pomoću grafičkog interfejsa i njegovih elemenata.

U sklopu UML-a postoje mnogi dijagrami, koji se uglavnom razlikuju po primjeni kojoj su namijenjeni.

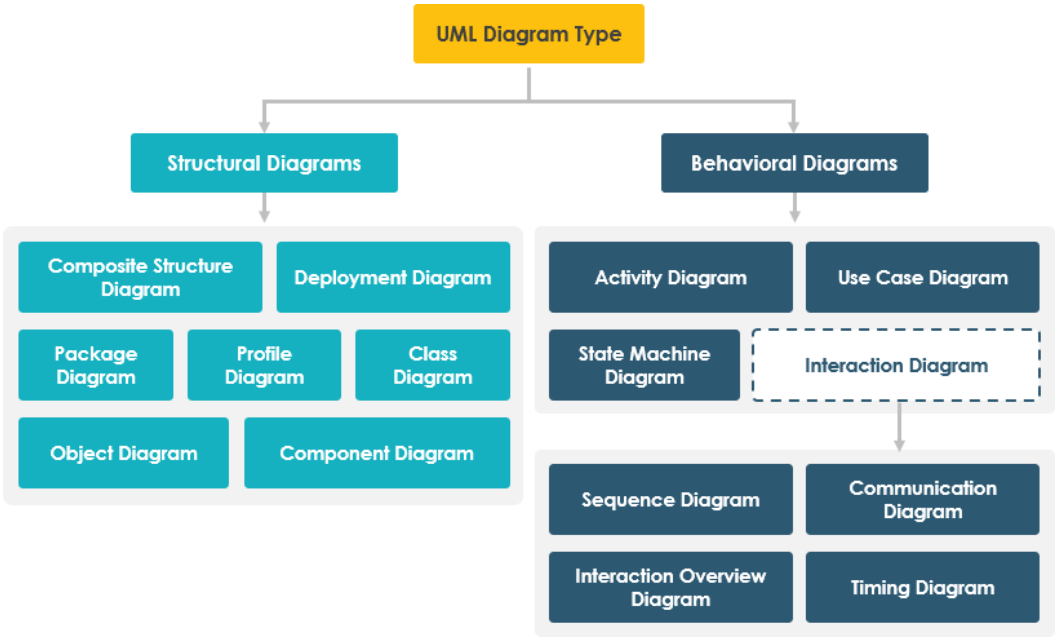
2.1. Podjela dijagrama

„Statičko modeliranje se koristi za specifikaciju strukture objekata, klasa ili komponenti koje postoje u problemskom domenu. Oni se izražavaju korišćenjem klase, objekta ili komponente. Dok se dinamično modeliranje odnosi na predstavljanje interakcija objekata tokom rada. Predstavljen je sekvencom, aktivnošću, kolaboracijom i stanjem. UML dijagrami predstavljaju ova dva aspekta sistema.

Statički pogled naglašava statičku strukturu sistema koristeći objekte, atribute, operacije i relacije. Također sadrži dijagram klase i dijagram komponenti.

Dinamički pogled naglašava dinamičko ponašanje sistema pokazujući saradnju između objekata i izvršenih promjena unutar objekta, također sadrži dijagram sekvence, dijagram aktivnosti i dijagram stanja mašine.

U UML jeziku postoji četrnaest tipova UML dijagrama koji su podijeljeni u dve grupe, od kojih su 7 statičkih i ostalih 7 dinamičkih dijagrama i mogu se prikazati hijerarhijski.



Slika 9 - Tipovi UML dijagrama

2.1.1. Dijagrami ponašanja

UML-ovih sedam dijagrama ponašanja se koriste za vizuelizaciju, specifikaciju, konstruisanje i dokumentovanje dinamičkih aspekata sistema. Pokazuje nam kako se sistem ponaša i kako komunicira samim sobom i drugim entitetima (korisnicima i drugim entitetima). Ono nam pokazuje kako se podaci kreću kroz sistem, kako objekti komuniciraju jedni sa drugima, kako protok vremena utiče na sistem ili koji događaji uzrokuju da sistem promijeni unutrašnja stanja.

Pošto dijagrami ponašanja ilustruju ponašanje sistema, oni se u velikoj mjeri koriste za opisivanje funkcionalnosti softverskih sistema. Kao primjer, dijagrami aktivnosti opisuju poslovne i operativne (eng. step – by – step) aktivnosti komponenti u sistemu.

Drugim riječima, dijagram ponašanja pokazuje kako sistem funkcioniše ‘u pokretu’, odnosno kako sistem komunicira sa spoljnim entitetima, korisnicima i kako reaguje na unos ili događaj i pod kojim ograničenjima radi“ [4].

Behavioral Diagram	Brief Description
Activity Diagram	It is a graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency
Use Case Diagram	It describes a system's functional requirements in terms of use cases that enable you to relate what you need from a system to how the system delivers on those needs.
State Machine Diagram	It shows the discrete behavior of a part of a designed system through finite state transitions.
Sequence Diagram	It shows the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.
Communication Diagram	It shows interactions between objects and/or parts (represented as lifelines) using sequenced messages in a free-form arrangement.
Interaction Overview Diagram	It depicts a control flow with nodes that can contain other interaction diagrams.
Timing Diagram	It shows interactions when the primary purpose of the diagram is to reason about time by focusing on conditions changing within and among lifelines along a linear time axis.

Slika 10 - Dijagrami ponašanja

2.1.2. Dijagrami strukture

„Strukturni dijagrami prikazuju statičku strukturu elemenata u vašem sistemu, tj. kako se jedan objekat odnosi na drugi. Ovaj dijagram nam prikazuje stvari u sistemu: klase, objekte, pakete, fizičke čvorove, komponente i interfejse. Sedam UML-ovih strukturnih dijagrama je grubo organizovano oko glavnih grupa stvari koje ćemo naći u modelovanju našeg sistema.

Pošto dijagrami strukture predstavljaju strukturu, oni se u velikoj mjeri koriste u dokumentovanju softverske arhitekture softverskih sistema. Npr. dijagrami komponenti opisuje kako je softverski sistem podjeljen na komponente i pokazuje zavisnost između ovih komponenti“ [4].

Structural Diagram	Brief Description
Composite Structure Diagram	It shows the internal structure of a classifier, classifier interactions with the environment through ports, or behavior of a collaboration.
Deployment Diagram	It shows a set of nodes and their relationships that illustrates the static deployment view of an architecture.
Package Diagram	It groups related UML elements into a collection of logically related UML structure.
Profile Diagram	
Class Diagram	It shows a set of classes, interfaces, and collaborations and their relationships, typically, found in modeling object-oriented systems.
Object Diagram	It shows a set of objects and their relationships, which is the static snapshots of instances of the things found in class diagrams.
Component Diagram	It shows a set of components and their relationships that illustrates the static implementation view of a system.

Slika 11 - Dijagrami strukture

2.2. Korišteni dijagrami

Dijagrami koje sam odabrao za projektovanje aplikacije su:

- Dijagram aktivnosti
- Dijagram sekvence
- Dijagram slučaja
- Dijagram entitetske relacije – ER

2.2.1. Dijagram slučaja

“Dijagrami slučaja nazivaju se (eng. Use Case), a koriste se za opisivanje niza akcija (slučajeva korištenja) koje neki sistem treba ili može obaviti u suradnji s jednim ili više vanjskih korisnika ili aktera koji sudjeluju sistemu. Slučaj upotrebe predstavlja specifikaciju skupa akcija koje izvršava sistem radi dobrobit jednog ili više aktera u sistemu.

Jedan od glavnih ciljeva UML-a je podrška poslovnom modeliranju sistema, ali UML specifikacija ne daje zapise specifične za poslovne potrebe. Dijagram slučaja je uveden u Rational Unified Process (RUP) s ciljem predstavljanja poslovnih funkcija u poslovnim modeliranjima.

Dijelovi Use Case dijagrama su:

- Use Cases (sekvence akcija koje su karakteristične za upotrebu sistema, a predstavljaju funkcije koje sistem izvršava)
- Actor (akteri se nalaze izvan sistema)
- Interactions (aktivnosti međusobne komunikacije aktera)

Smatra se da su USE CASE dijagrami važni jer obavljaju kompletan razvoj softvera pa se zbog takvog razvoja koji je zasnovan na UML-u kaže da je usmjeravan Use Case-om, odnosno slučajevima upotrebe.

Slika 12 - Prikaz aktera i slučaja upotrebe u Use Case dijagramu

Primjer:

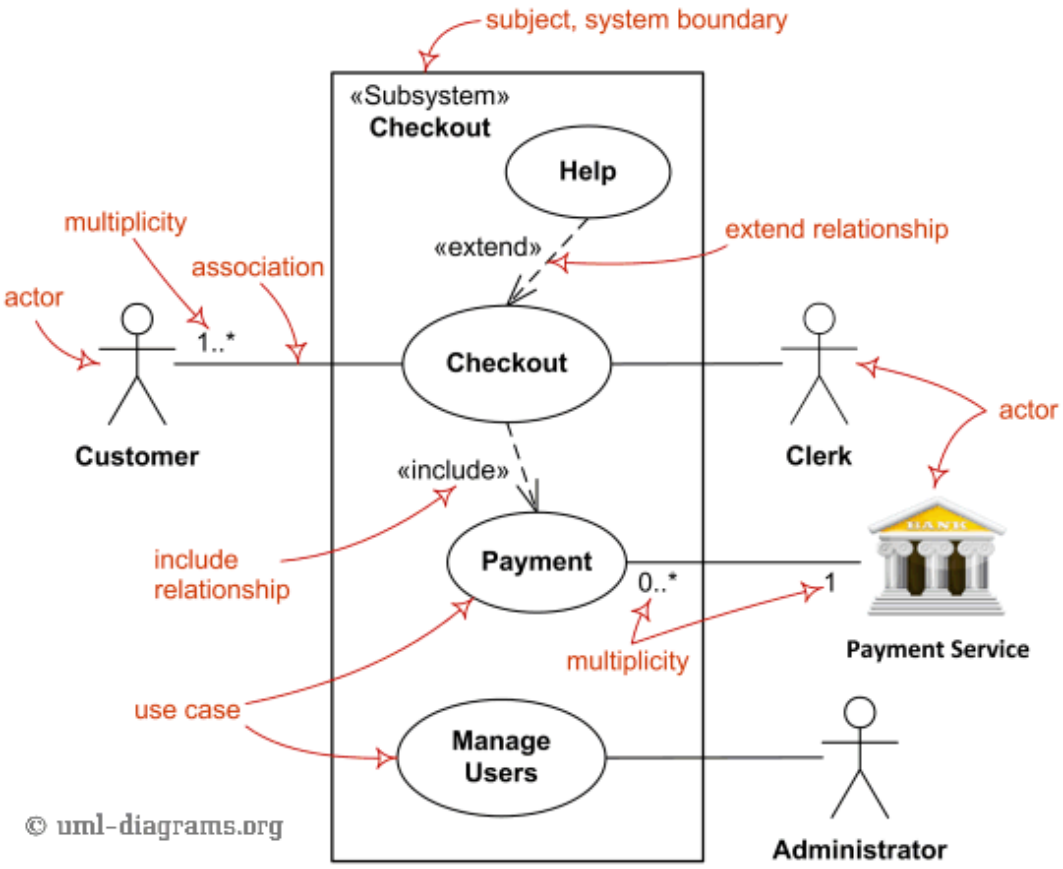
Glavni elementi Use Case dijagrama prikazujem na slici 13:

- Actor (Kupac)
- Help (Pomoć)
- Checkout (Naplata)
- Payment (Plaćanje)
- Manage Users (Upravljanje korisnicima)

Kupac bira proizvod, kontaktira prodavača, nabavlja informacije o kupovini i plaćanju, te na kraju postiže dogovor narudžbe o kupnji s prodavačem. Akter određuje ulogu vanjskog entiteta koji surađuje s predmetom (npr. razmjena podataka), ljudskim korisnikom projektovanog sistema ili nekim drugim sistemom. Kada je vanjski entitet u interakciji sa predmetom tada dolazi do izražaja uloga aktera koji predstavlja fizičku koja može imati nekoliko različitih uloga. Svi sudionici moraju imati nazive pretpodstavljenoj ulozim, npr. akteri koji imaju različite akterske uloge, mogu imati nazive poput ovih: kupac (customer), web klijent, student, sistem plaćanja (payment service), administrator. Akteri se specijalizuju pomoću generalizacije. Svaki Use Case predstavlja jedinicu korisnih funkcionalnosti koje subjekti pružaju akterima. Asocijacija između aktera i slučaja upotrebe pokazuje komunikaciju između aktera i slučaja upotrebe.

U elipsama su prikazani slučajevi upotrebe zahtijeva aktora (npr. plaćanje – Payment) te su povezane vezama uključivanja (include) jednog osnovnog slučaja s drugim "uključenim" slučajom proširivanjem slučaja upotrebe osnovnim ponašanjem proširujućeg slučaja upotrebe (Extend), a generalizacijom kao kod klasa jedan slučaj upotrebe može biti specijaliziran na više slučajeva s

dodatnim osobinama. Za upravljanje korisnicima projekta i resursa (Manage Uses) te administracijom (Administrator) najvažnija je log aktivnost slučaja upotrebe koja predstavlja zapis svih aktivnosti između navedenih slučajeva upotrebe. Mnogostrukost se odnosi na to da UML omogućuje korištenje mnoštva u jednom, odnosno predstavlja udruženja između aktera i slučajeva upotrebe. Kada aktor ima asocijaciju na slučaj upotrebe s mnoštvom to znači da akter može biti uključen u više slučajeva korištenja istog tipa" [5].



Slika 13 - Elementi Use Case dijagrama (primjer)

2.2.2. Dijagram aktivnosti

“Dijagram aktivnosti je dijagram koji ističe tok kontrole od aktivnosti do aktivnosti. Najčešće se koristi za prikaz tokova u sistemu, sa alternativnim putanjama. Veoma je sličan klasičnim blok dijagramima, s tim što se na njemu prikazuju i paralelni tokovi. Njegove osnovne primjene su modelovanje tokova poslovanja i modelovanje operacija.

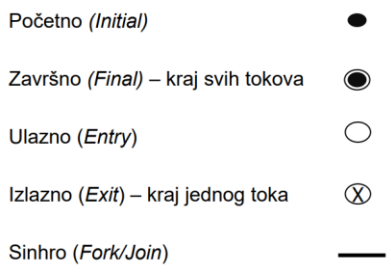
AKTIVNOST je obrada koju je moguće dekomponovati na jedinice – elementarne jedinice podaktivnosti ili pojedinačne akcije.
AKCIJA je osnovna jedinica specifikacije ponašanja.

Akcije mogu biti:

- Pokretanje aktivnosti (pozivanje operacije ili slanje signala)
- Čitanje (vraćanje informacije)
- Upis podataka (promjena stanja)
- Kreiranje ili uništavanje objekata
- Izračun – izvršavanje primitivnih (aritmetičkih) operacija i funkcija
- Izvršenje akcije koja dalje inicira izvršenje neke druge aktivnosti

Dijagrami aktivnosti predstavljaju tehniku za opis proceduralne logike i poslovnih procesa. Sa tog aspekta su slični staroj tehnici flowchart, ali glavna razlika je što DA podržava modelovanje paralelnog izvršavanja aktivnosti. DA sadrže: akcije, objekte, kontrolne čvorove, psuedo stanja, grane.

Psuedo stanja su specijalni markeri koji usmjeravaju tok saobraćaja u dijagramu.



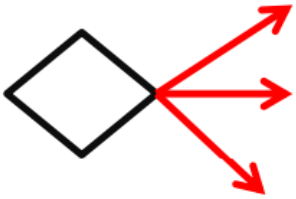
Slika 14 - Psuedo stanja u dijagramu

Grananje specifikuje alternativne putanje toka zavisno od uslova grananja.

Za granjanje i spajanje koristimo simbol:

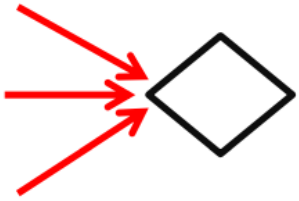
Slika 15 - Simbol za spajanje (grananje)

Više grana može izlaziti iz simbola grananja (decision):



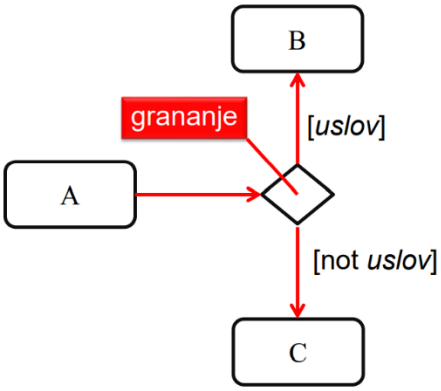
Slika 16 - donošenje odluke u grananju

Više grana može ulaziti u simbol grananja (merge):



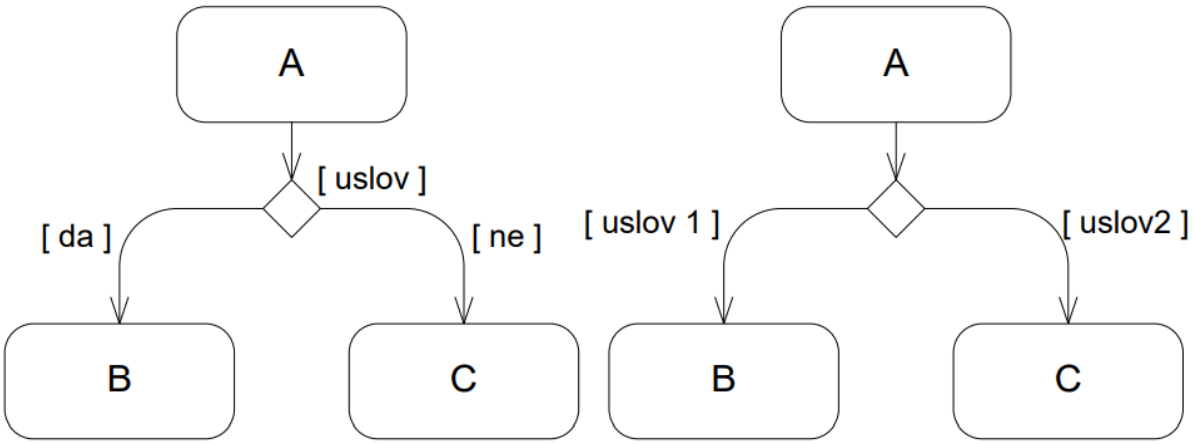
Slika 17 - stapanje u grananju

Uslov se piše između uglastih zagrada na grani. Else grana se pojavljuje ako nije ispunjen nijedan uslov. Primjer:



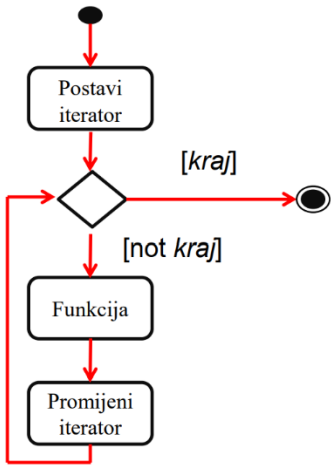
Slika 18 - Else uslov u grananju

Odlučivanje – OR grananje – Decision



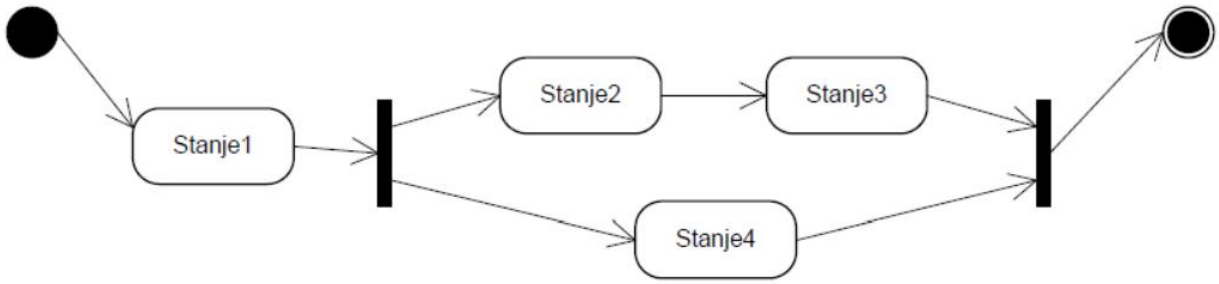
Slika 19 – OR grananje

Iteracija grananjem:



Slika 20 - Iteracija u grananju

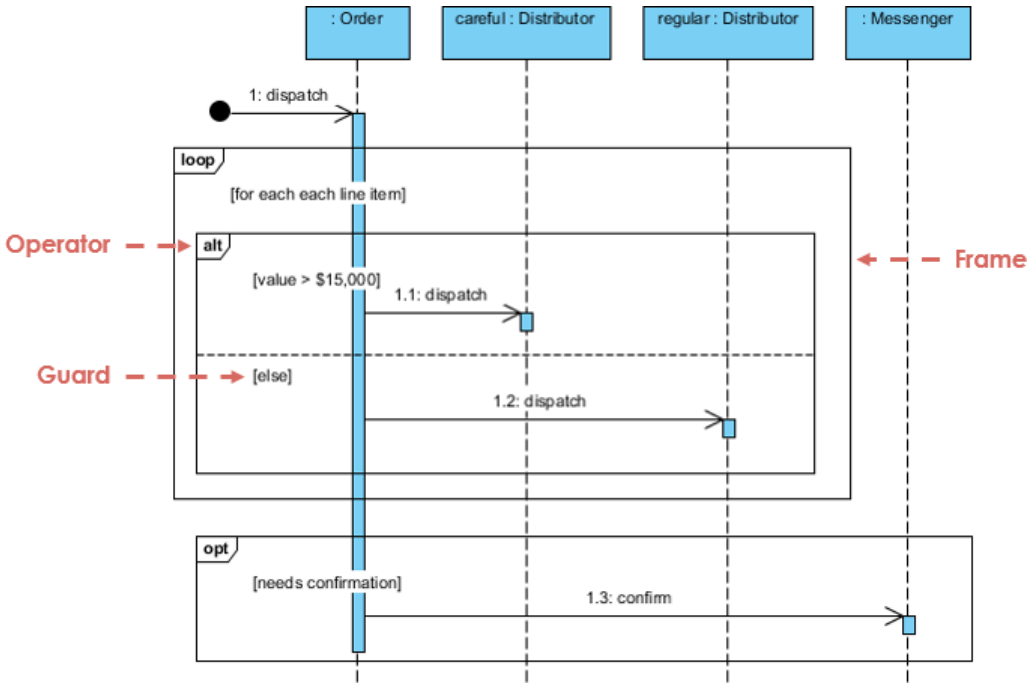
Račvanje i skupljanje (fork and join) koje se najčešće koristi pri paralelnom odvijanju nekih aktivnosti. Pritom se dijagram stanja račva od nekog stanja u niz drugih stanja da bi se nakon prolaska kroz ta paralelna stanja ponovo skupio. Sastanak (randezvous) ima više prelaza koji ulaze izlaze. Jednom kad su svi ulazni prijelazi aktivirani pokreću se svi izlazni prijelazi, svaki u zasebnoj dretvi“ [6].



Slika 21 - Čvor račvanja i skupljanja

2.2.3. Dijagram Sekvence

Ove dijagrame najčešće koristimo za neku interakciju, kojoj je glavni cilj razmjena poruka tokom izvršavanja. Primarni fokus ovog dijagrama je opis interakcije koja se odnosi na ponavljanje poruka koje su razmjenjene, zajedno sa pojavom određenih specifikacija tokom izvršavanja.



Slika 22 - Primjer dijagrama sekvence

Osnovni elementi dijagrama sekvence su: objekti (predstavljeni pravougaonikom), poruke (prikazane strelicama), vrijeme (prikazano vertikalnom isprekidanom linijom – linija života).

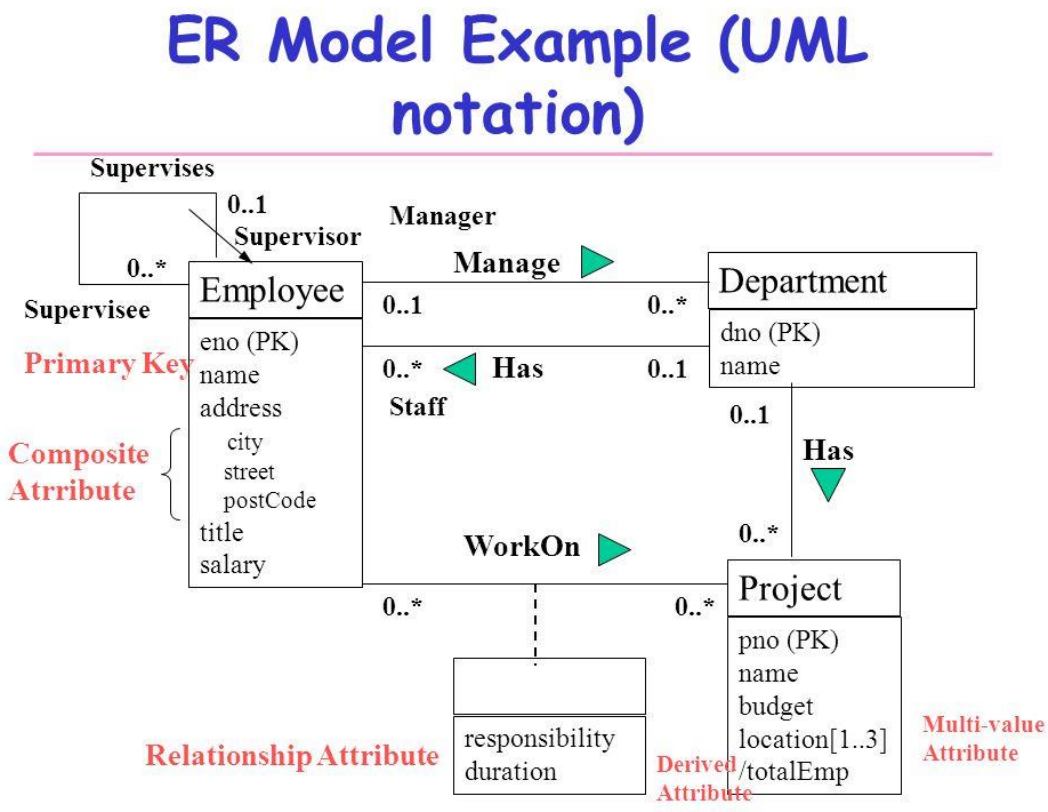
2.2.4. Dijagram entitetske relacije

ER dijagram predstavlja vezu između entitetskih zajednica unutar baze podataka.

U ovom slučaju entitet predstavlja objekat, odnosno komponentu koju čine podaci. Entitetska zajednica (skup) je kolekcija sličnih entiteta. Entiteti se sastoje od atributa koji definišu njihova svojstva. Definisanjem entiteta, njihovih atributa i pokazivajući povezanost između njih, ER dijagram pokazuje logičku strukturu baze podataka. Ovi dijagrami se najčešće koriste za skiciranje dizajna baze podataka.

Dva osnovna razloga za pravljenje dijagrama za bazu podataka:

- Ako dizajniramo novu bazu ili trebamo dokumentovani trenutnu
- Ako već imamo bazu podataka koju želimo dokumentovati, možemo napraviti dijagram koristeći podatke iz naše baze



Slika 23 - ER dijagram (primjer)

3. APLIKACIJA

U ovom dijelu seminarskog rada opisaću osnovne funkcionalnosti aplikacije, pomoću dijagrama koje sam prethodno naveo i također ovaj dio je većinski praktični.

3.1. O aplikaciji

Za ovu aplikaciju već duže vrijeme imam velike ambicije da ju ostvarim. Nakon prikupljenog znanja kojeg sam savladao, napokon mogu ostvariti svoju aplikaciju na pravilan način.

Odlučio sam se za ovu aplikaciju jer i sam gubim važne podatke veoma često, a aplikacije koje postoje za bilješke smatram da nisu toliko dobro napravljene.

3.2. Osnovne funkcije aplikacije

Osnovne funkcije koje su krucijalne za moju aplikaciju su:

- Pravljenje naloga (sign-up) ovaj korak se preskače ukoliko imate već profil
- Prijavljivanje naloga (sign-in)
- Kreiranje bilješki (create notes)
- Izmjene bilješki (edit notes)
- Brisanje bilješki (delete notes)

3.2.1. Pravljenje naloga

Nakon ulaska na aplikaciju, ukoliko je naš korisnik prvi put dobiće opciju za pravljenje naloga (en. Sign-Up) u kojoj će unjeti osnovne podatke kao što su: ime, prezime, datum rođenja, broj telefona, e-mail adresu i na kraju korisnik je obavezan da ispuni jednostavnu captchu da potvrdi da je stvarna ličnost.

3.2.2. Prijavljivanje naloga

Nakon kreiranja naloga korisnik će se moći prijaviti pomoću tog naloga na aplikaciji i to će ga odvesti na glavni prozor aplikacije.

3.2.3. Kreiranje bilješki

Nakon same prijave u samom startu korisnik će moći kreirati bilješke. U procesu kreiranja bilješke korisnik će kreirati naslov po volji i njen sadržaj, a isto tako aplikacija će podržavati sve Google fontove i također korisnik će unutar sadržaja imati pristup opcijama kao što su: bold, italic i underline.

U novijim verzijama bi implementovao mogućnost čuvanja podataka na serveru ili na samom uređaju korisnika i premium verziju koja bi sve sačuvane podatke enkriptovala radi veće sigurnosti.

3.2.4. Izmjene bilješki

Ovu opcijom smatram čak i jednom od najbitnijih radi potrebe za ažuriranjem podataka ili gramatičkim greškama koje ne primjetimo tokom kucanja.

3.2.5. Brisanje bilješki

Pošto imamo opciju za izmjene i kreiranje, isto tako moramo imati opciju za brisanje podataka. U budućim verzijama želim da nakon brisanja određene bilješke da se ona ne izbriše sa uređaja zauvijek već da se skladišti u privremenom otpadu i naravno ako korisnik želi da zauvijek izbriše bilješku i ta opcija će ostati na raspolaganju.

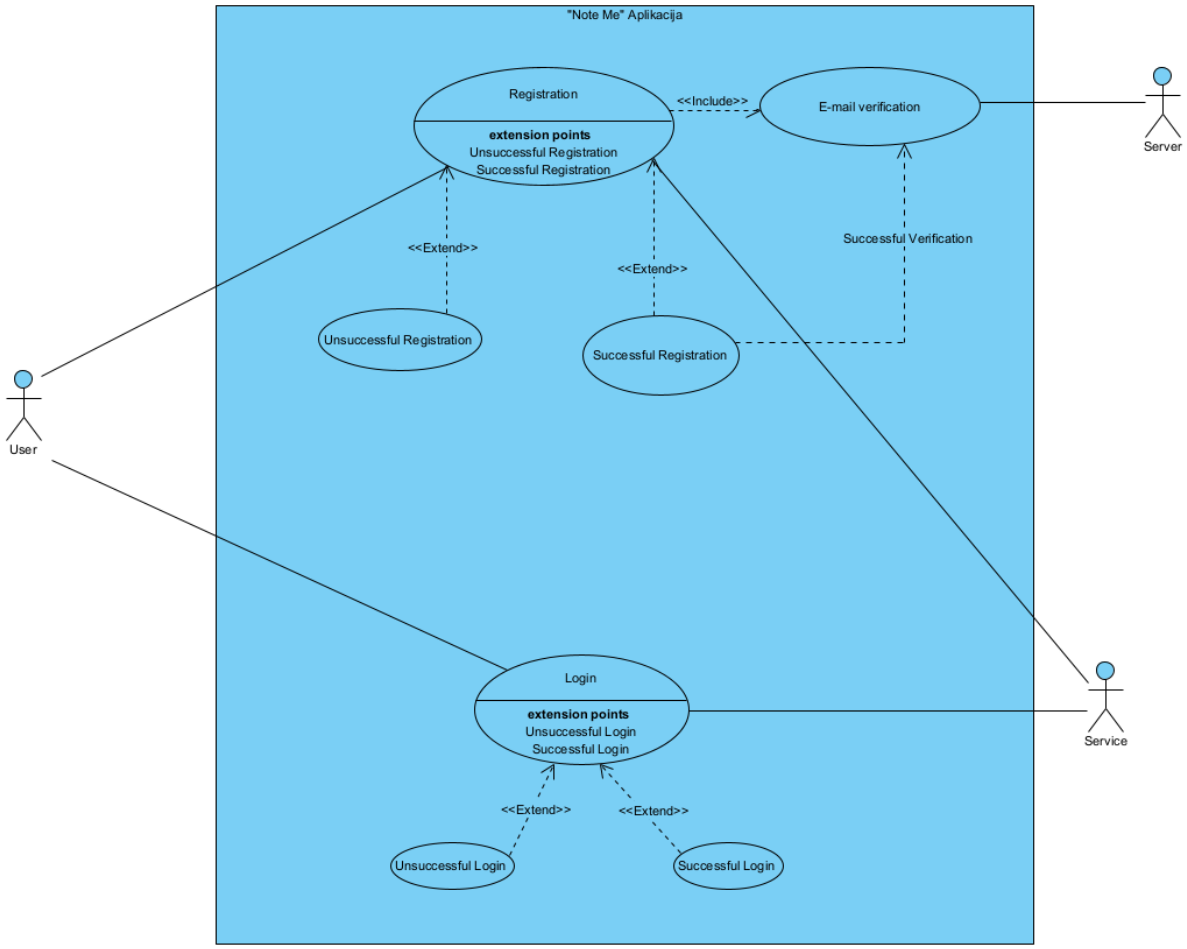
3.3. Dijagrami u aplikaciji

Ova sekcija je posvećena dijagramima sa kojima sam projektovao aplikaciju pomoću UML jezika i programa Visual Paradigm for UML.

Slika 24 - Visual Paradigm Logo

3.3.1. Dijagrami slučaja

Kroz ovaj dijagram želim da prikažem aplikaciju prije ulaska na glavni prozor i za glavni prozor zasebno.



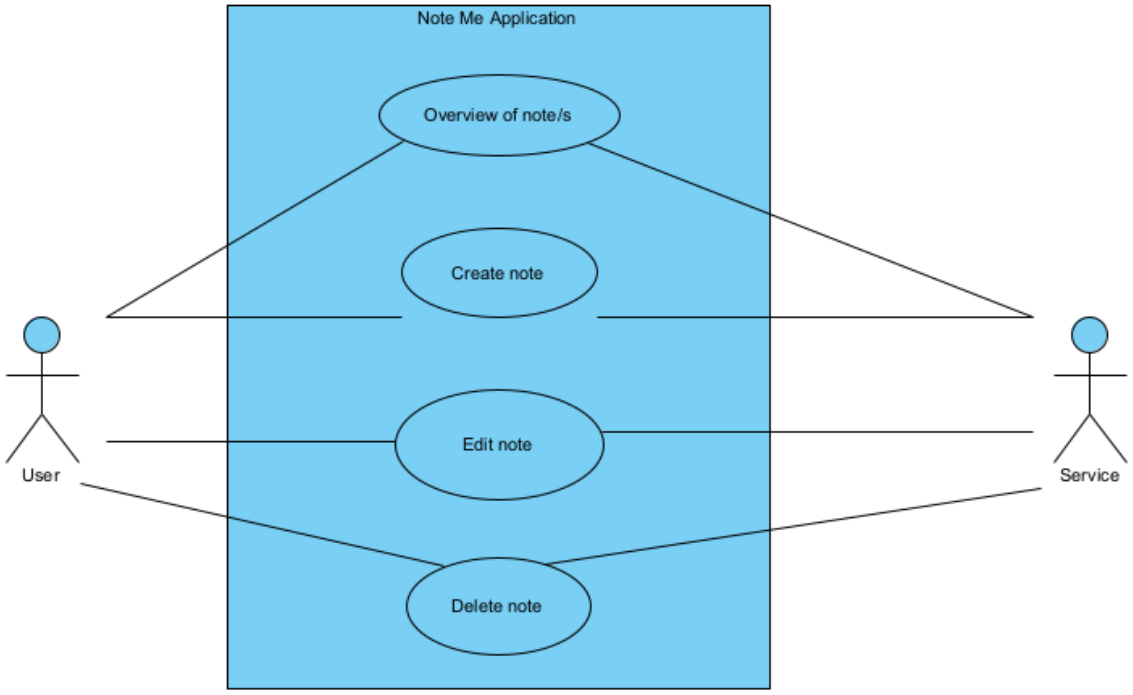
Slika 25 - Prvi dijagram slučaja

Kod registracije imamo tri slučaja: Neuspješne registracije, uspješne registracije i verifikacije elektronske pošte.
U slučaju da korisnik ne ispuni polja za unos korektno, desiće se slučaj neuspješne verifikacije i korisnik će imati mogućnost ponovne registracije.

Kada korisnik unese korektne informacije, desiće se slučaj uspješne registracije, potom automatski korisniku se šalje potvrda verifikacije putem email-a i za to je zadužen slučaj email verifikacije.
Prijava sadrži dva sadržaja: neuspješne prijave i uspješne prijave.

Ako korisnik ne unese korektne podatke, korisnik će imati mogućnost da ponovi unos podataka i za to je zadužen slučaj neuspješna prijava.
Drugi slučaj je ako korisnik unese korektne podatke, korisnik je dalje prosljedjen na glavni prozor aplikacije i za taj dio je zadužen slučaj uspješna prijava.

Kroz ovaj dijagram želim da prikažem glavni dio aplikacije.

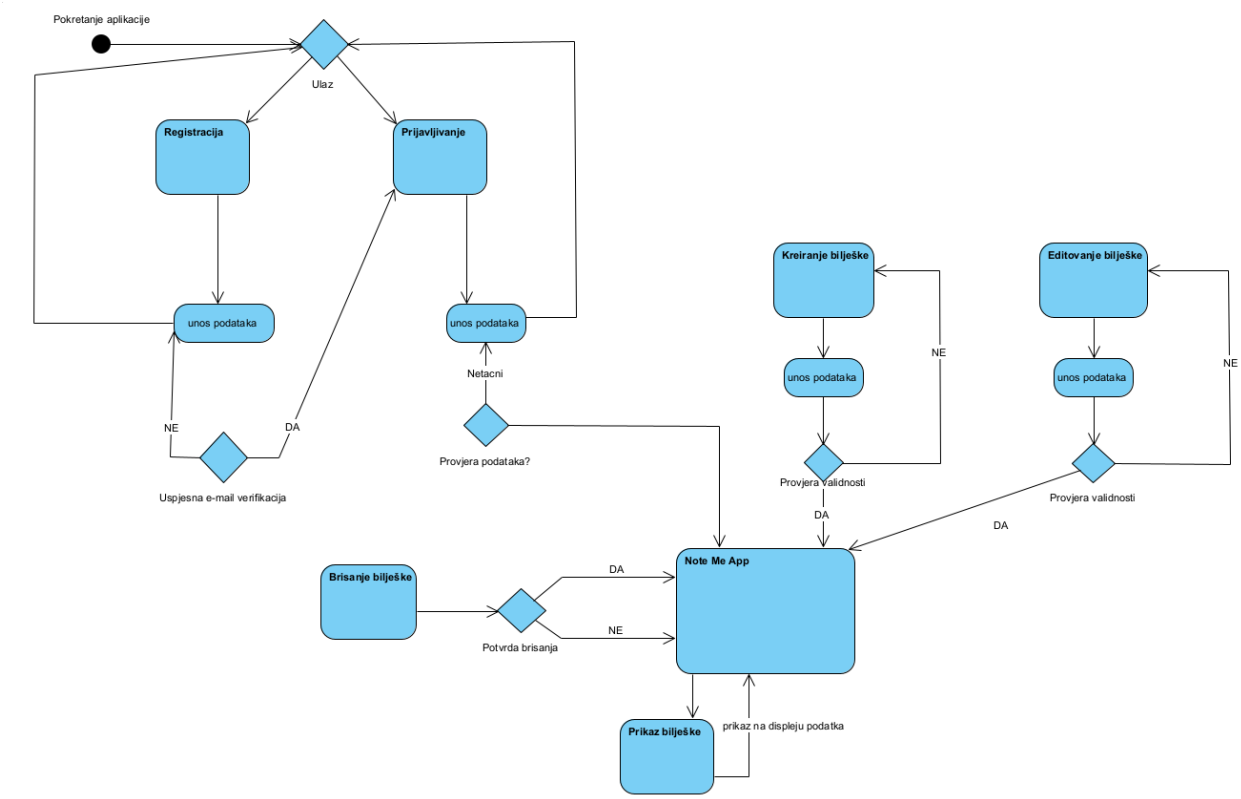


Slika 26 - Drugi dijagram slučaja

U drugom dijagramu imamo slučajeve: pregled bilješke, kreiranje bilješke, editovanje bilješke i brisanje bilješki i da navedem da su ovo osnovne funkcije aplikacije.

3.3.2. Dijagram aktivnosti

Pomoću dijagrama aktivnosti objasniću primjenu i karakteristike aplikacije.



Slika 27 - Dijagram aktivnosti

Crna tačka u gornjem lijevo uglu prethodne fotografije označava početnu tačku tj. početak ulaska u samu aplikaciju.

Nakon uspješnog pokretanja aplikacije, korisniku će biti ponuđene opcije: Prijavljivanje i Registracija. Ako naš korisnik ima već postojeći nalog na našoj aplikaciji iće direktno na prijavljivanje, a u slučaju da želi novi nalog može ga kreirati ponovo na registraciji.

U slučaju da korisnik neuspješno unese podatke imaće mogućnost da ponovi proces prijavljivanja, a u suprotnome biće prosljeđen ka glavnom prozoru aplikacije.

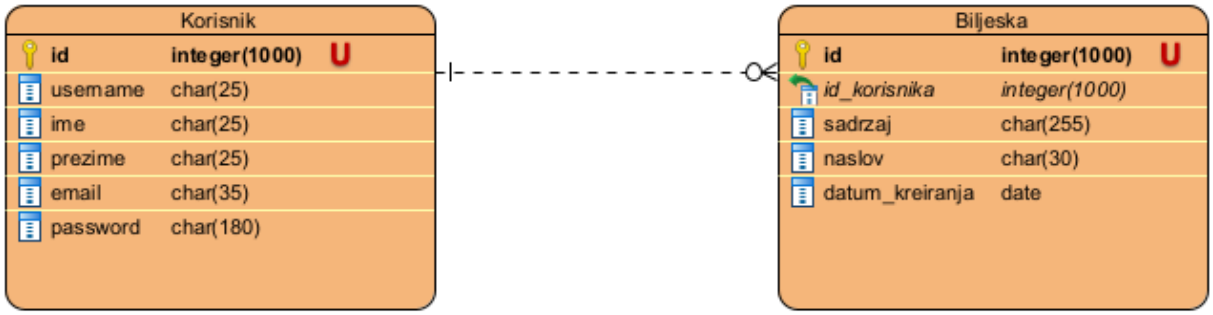
Jedina razlika sa kojom se susrećemo sa registracijom i prijavljivanjem je ta što registracija zahtjeva dodatnu verifikaciju elektronske pošte tzv. e-maila.

U dijelu gdje imamo kreiranje bilješki i modifikovanje bilješki, imaćemo provjeru da li su podaci korisnika validni, ako nisu proces će biti ponovljen sve do trenutka kada korisnik unese validne podatke.

Radi ličnih grešaka u pojmu slučajnih brisanja podataka, odlučio sam da nakon selektovanja bilješke za brisanje korisniku će se prikazati potvrda za brisanje (DA\NE).

3.3.3. ER Dijagram

Pomoću ovog dijagrama imamo mogućnost da opišemo bazu podataka, nažalost prva verzija baza podataka će biti vrlo jednostavna i u ovom dijagramu koristio sam samo dve tabele pod nazivom korisnik i bilješka koje su povezane primarnim ključem.



Slika 28 - ER dijagram

U prvoj tabeli Korisnik navedeno je:

- id (primarni ključ, jedinstven) - integer (1000)
- username – char (25)
- ime – char (25)
- prezime – char (25)
- email – char (35)
- password – char (180)

U drugoj tabeli Biljeska navedeno je:

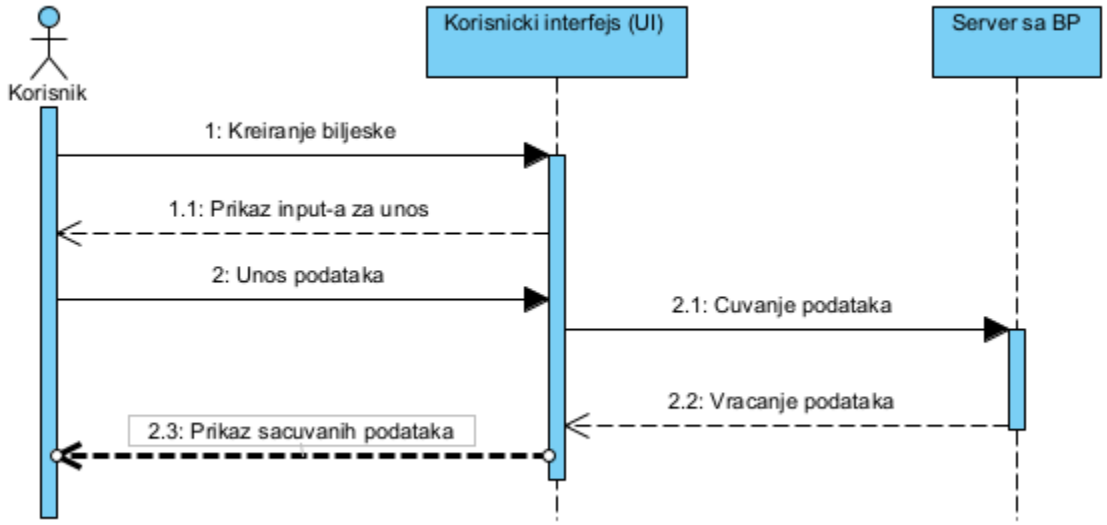
- id (primarni ključ, jedinstven) – integer (1000)
- id_korisnika (strani ključ, jedinstven) – integer (1000)

- sadrzaj – char (255)
- naslov – char (30)
- datum_kreiranja – date

Bitno za navesti je to da može postojati samo jedan korisnik sa istom email adresom i istim id-om i jedan korisnik može imati više bilješki u bazi podataka.

3.3.4. Dijagram sekvence

Kroz ovaj dijagram prikazan je korisnik, korisnički interfejs i server koji hostuje bazu podataka i također objašnjavam proces kreiranja bilješke kroz vrijeme.



Slika 29 - Dijagram sekvence

ZAKLJUČAK

Ovaj predmet zajedno sa seminarskim radom me je na neki način pokrenuo da razmišljam van svojih granica za koje sam i sam mislio da nikada neću dostići. Također ovaj projekat mi je duže vrijeme stajao u malom mozgu ali nikad nisam imao motivaciju da ga ostvarim i sad kako je već praktično gotov planiram ostvarivati i osmišljati mnogo kompleksnije i naravno korisnije aplikacije.

Naime, pored motivacije koje sam stekao izradom ovog seminarskog rada, mogu reći da sam stekao mnogo znanja što se tiče UML jezika i nadam se da u budućnosti ću ih koristiti.

4. POPIS SLIKA

5. CITATNI IZVORI

[1] Visual Paradigm, »Unified Modeling Language,« 2021. [Mrežno]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/.

[2] »Wikipedia - Unified Modeling Language,« 2021. [Mrežno]. Available: https://en.wikipedia.org/wiki/Unified_Modeling_Language.

[3] B. Latinović, Informacioni sistemi, Banja Luka: Panevropski univerzitet APEIRON, 2006, pp. 272-282.

[4] »Visual Paradigm,« 2021. [Mrežno]. Available: https://www.visual-paradigm.com/tw/guide/uml-unified-modeling-language/behavior-vs-structural-diagram/.

[5] »UML Use Case Diagrams,« [Mrežno]. Available: https://www.uml-diagrams.org/use-case-diagrams.html.

[6] T. Lethbridge, Object-Oriented Software Engineering: Practical Software Development using UML and Java, Second Edition by Timothy Lethbridge, Mc Graw Hill, 2005.