SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. xxxx

# Informacija o sustavima i uslugama

Ivan Svalina

Zagreb, lipanj 2021.

# Sadržaj

# 1. Uvod

U današnje vrijeme pandemije i Zoom-a običnom čovjeku je posebno potrebna brza i učinkovita internet veza. Naime, mnogi su se ljudi zbog mogućnosti rada na udaljeno odselili u ruralne krajeve ili mjesta gdje ne postoji optička veza ili bilo kakav sličan brzi i efektivan oblik veze. U ovom trenutnom obliku to nije održivo za neke lokalne uprave. Na primjer u ruralnom Maine-u u Sjedinjenim Američkim Državama socijalna radnica Starr Gilmartin se nalazi u području bez optičkog interneta gdje je veza uspostavljena uglavnom preko bakrenih kablova. Zbog toga je na njenoj mreži brzina preuzimanja 80 Mb/s, a brzina učitavanja je 8Mb/s. Aplikacije poput Zooma ili Microsoft Teamsa naprosto ne mogu raditi na tako malim brzinama. Njena situacija nije jedinstvena, takvih sličnih domaćinstava je u SAD-u 21 milijun. Upravo zato mnoge lokalne samouprave u pokušaju privlačenja novih ljudi puno ulažu u svoju mrežnu infrastrukturu[1].

Baš u tu svrhu je napravljen ovaj rad i služi svim ljudima koji imaju takav ili sličan problem kao gospođa Gillmartin. Rad dokumentira aplikaciju u kojoj se mjeri brzina učitavanja, preuzimanja, spremanje podataka i mogućnost usporedbe različitih lokacija. U dokumentaciji postoje različiti grafovi u kojima se pobliže pojašnjava način uporabe aplikacije i opisi različitih dijelova. Svrha rada je ponajprije za svrhe istraživanja ili dijagnostike.

Rad kao cjelina je podijeljen u 7 poglavlja. U prvom je prikazan uvod u rad. Onda u drugom je opisan općeniti opis projektnog zadatka i činjenice koje ga pobliže definiraju(tržište, mjerenje, problemi). Iza toga u trećem je poglavlju opisana specifikacija programske potpore gdje su opisani obrasci uporabe i zahtjevi. I tu se isto tako nalaze sekvencijski dijagrami, koji opisuju tijek radnji kroz vrijeme. Cilj trećeg poglavlja je pojasniti kako rade različite funkcije nakon različitih podražaja. Nadalje, u četvrtom poglavlju detaljno se prikazuje arhitektura aplikacije od MVC arhitekture, Djanga, opisa baza do UML dijagrama stanja, aktivnosti, razreda i komponenti. Cilj ovog poglavlja je da svaki čitatelj ili korisnik vidi na koji način funkcionira aplikacija te da shvati kako komuniciraju različiti dijelovi aplikacije. U petom poglavlju se nalazi implementacija i korisničko sučelje. Ovdje pišu podaci o korištenim tehnikama, načinu ispitivanja, uputama za rad i daje informacije kako koristiti te podatke. Tu se također nalazi i dijagram razmještaja. Iza petog poglavlja slijedi zaključak, gdje je dan rezime rezultata i

zapažanja tijekom izrade ovog rada. I na kraju u 7. poglavlju se nalazi literatura korištena u izradi.

# 2. Opis projektnog zadatka

## 2.1. Mjerenje brzine mreže

Mjerenje brzine mreže nije jednoznačan pojam, već se izražava s nekoliko mjerila brzine mreže. Različiti zahtjevi su posebno osjetljivi na različita mjerila mjerenja brzine. Recimo pretraživanje foto galerije zahtjeva veliku brzinu preuzimanja, dok za igranje igre na mreži treba malo vrijeme povratnog putovanja. AT Tester Methodology [AT08] definira skup od 6 mjerila koje se mjere u svakom pokusu: brzina preuzimanja, brzina učitavanja, vrijeme povratnog putovanja(RTT), jitter, gubitak paketa i dostupnost.

Brzina preuzimanja je najveća brzina kojom se mogu dohvaćati podaci s interneta. Mjeri se preuzimanjem datoteke veće veličine kroz vrijeme, onda napravi prosjek kretanja brzine i na kraju se prosjek izrazi kao brzina preuzimanja. Mjeri se u megabitima po sekundi(Mb/s). S druge strane, brzina učitavanja je mjerilo koje definira brzinu kojom korisnik šalje podatke na mrežu. Na sličan način se mjeri kao i proces mjerenja brzine preuzimanja, samo što se datoteka učitava. Nadalje, vrijeme povratnog putovanja(RTT) je vrijeme potrebno da paket stigne do odredišta i da se vrati. RTT je značajan u sustavima koji imaju dvosmjernu komunikaciju kao što je online kupovina. Nakon toga slijedi jitter. Jitter je vremenska jedinica koja označava end-to-end kašnjenje s jednog paketa na drugi u istom toku podataka. Gubitak podataka predstavlja dio paketa koji ne dostigne odredište. Ovo je primjetljivo posebno u gledanju videa. Na kraju dolazi dostupnost. Bitno je reći da ona označava postotak uspješnih zahtjeva na mreži. Računa se formulom (Dostupnost = (1–F/T)×100%). F označava broj neuspješnih zahtjeva, a T označava ukupan broj zahtjeva[2].

***Table 1: Relevance of Metrics to Various Internet Services***

| Service | Speed | | Delay | | Loss |
|---|---|---|---|---|---|
| | Down | Up | RTT | Jitter | |
| Browse (text) | ++ | - | ++ | - | - |
| Browse (media) | +++ | - | ++ | + | + |
| Download file | +++ | - | - | - | - |
| Transactions | - | - | ++ | + | - |
| Streaming media | +++ | - | ++ | ++ | ++ |
| VOIP | + | + | +++ | +++ | +++ |
| Games | + | + | +++ | ++ | ++ |

*Legend: +++ highly relevant, ++ very relevant, + relevant, - not relevant*

Slika 1. Relevantnost mjerila na različitim internetskim uslugama[2]

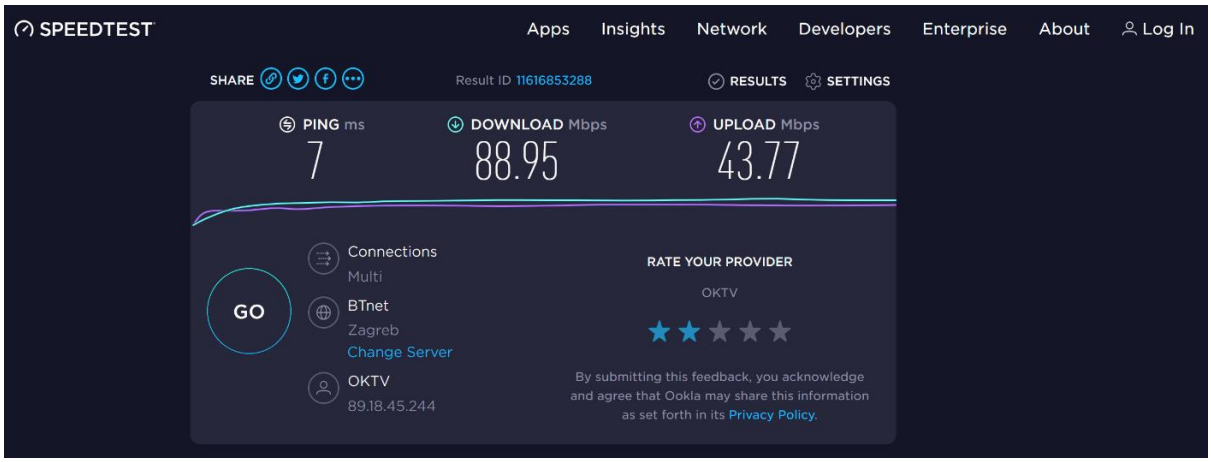
Od svih ovih mjerila, najbitnija su svakako brzina učitavanja i preuzimanja. Ove dvije brzine se mjere tako da se preuzme ili učitava datoteka veće veličine s javno dostupnog servera i ovise jako o putu do servera. Na putu do servera se nalaze razni uređaji, od kućnog modema, usmjerivača, LAN komutatora i samih kabela kojima se provodi promet. Upravo zbog toga se mjerenje brzine učitavanja i preuzimanja može izražavati na različite načina.

Kao prvo, može se izraziti preko kapaciteta. Kapacitet je mjerilo potpunog prometa koji može podnijeti jednu vezu ili staza na mreži. On se izražava preko količine prometa koji veza može podnijeti u određenom vremenskom intervalu. Mjerilo kapaciteta najčešće pomaže u određivanju mrežnih operacija i planiranja oko mreže. Jedan oblik kapaciteta, End-to-end kapacitet je minimalni kapacitet veze. Nakon kapaciteta, brzina se može mjeriti preko dostupne širine pojasa. Najčešće predstavlja najmanje dostupnu širinu pojasa između dva linka na stazi i mjerilo je koje izražava količinu nekorištenog kapaciteta u jednoj vezi ili na stazi. Ovo mjerenje zna nekada biti jako nezgodno te se zbog toga računa oduzimanjem od korištene širine pojasa. Preko dostupne širine pojasa se najčešće određuje je li potrebno nadograditi kapacitet.

Na kraju je također bitno spomenuti prijenosni kapacitet koji je mjerilo koje se najčešće koristi kao mjerilo određivanja brzine preuzimanja ili učitavanja. On je mjerilo količine podataka koja se može prenijeti na jednoj mrežnoj stazi s transportnim protokolima koji izbjegavaju zagušenja, kao što je TCP. Prijenosni kapacitet je određen brojem različitih TCP tokova koji se natječu, dinamikom, postavkama TCP stoga na svakom kraju itd. Također je kod njega posebno to što on ne mjeri uvijek zaglavlje paketa i pakete podataka koji se šalju. Poseban je isto tako po tome što se može definirati kao ostvariva propusnost samo jedne transportne veze, naravno uz to da je transportni protokol dobro postavljen[3].

## 2.2. Tržište i potreba

Postoje različiti testovi koji omogućuju mjerenje. Ovdje će se spomenuti samo neki od najbitnijih. Kao prvo bi trebalo spomenuti Speedtest/Ookla test od Ookla Net Metricsa. Uz test Ookla ima besplatnu web stranicu speedtest.net, koja je posredno korištena za izradu praktičnog dijela ovog rada. Speedtest/Ookla test je korišten od 2006.-e godine, pokrenut je 1.4 milijarde puta i upravlja se sa sto web stranica diljem svijeta. Nadalje ovaj test daje razne opcije korisniku[3].

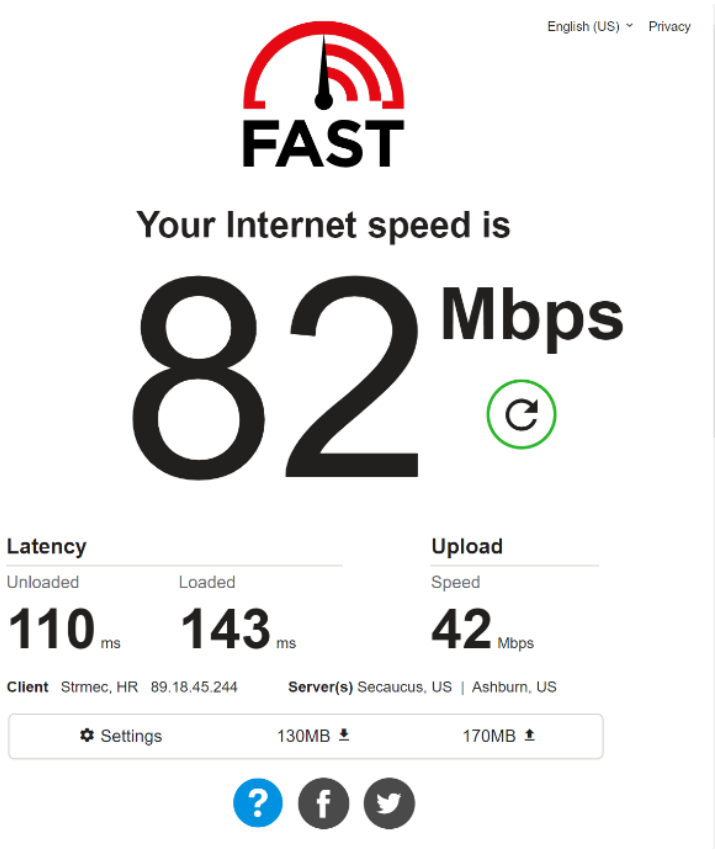



Slika 2. Speedtest/Ookla

Test daje opcije prikazivanja: pinga(ms), brzine preuzimanja(Mb/s), brzine učitavanja(Mb/s), grafičkog prikaza, vremena, računala servera i klijenta. Osim osnovnih funkcija, ovaj test pruža pregled povijesti mjerenja kroz grafički i pisani prikaz, mogućnost dijeljenja, drukčijeg načini prikazi vremena, softvera i brzine, prikaz na različitim jezicima, prikaz na različitim operacijskim sustavima i razne druge mogućnosti[6].

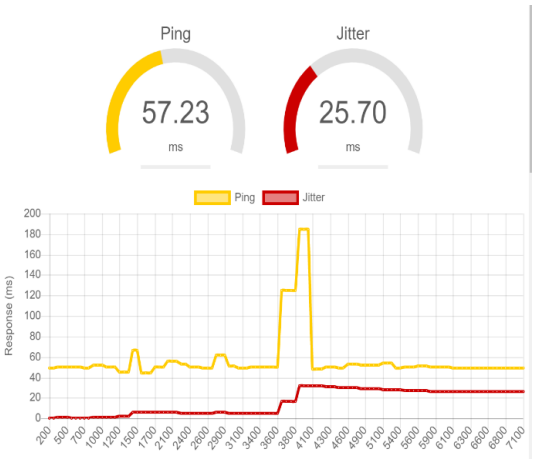
Također kod testa preuzimanja na Ookli se koristi čak na 8 paralelnih HTTP dretvi, onda se svaki uzorak dijeli na 8 jednakih komada(od kojih se najbržih 10% i najsporijih 30% izbacuju) i na kraju se ovi ostali komadi računaju u prosjek. Slično vrijedi i za test učitavanja. Kod testa učitavanja POST metodom se učitavaju podaci na serversku stranu, test iskoristi 8 paralelnih HTTP dretvi i sortira dijelove po brzini od kojih najbržih pola se koristi za računanje prosjeka brzine učitavanja. Još je bitno napomenuti da Ookla upotrebljava više TCP veza i da ih koristi za to da se izbjegnu zagušenja na mjerenju[3].

Drugi bitan test je fast.com. Ovaj test se ističe zbog toga što ga podupire Netflix i služi ujedno omogućavanju što bolje usluge gledanja Netflixovog sadržaja. Zato se ovaj test posebno koncentrira na mjerenje brzine učitavanja. Test daje prikaz brzine učitavanja, latencije, klijenta računala, server računala i u najvećoj kućici brzine preuzimanja. Nadalje, daje mogućnost prikaza pinga i ostalih stvari klikom na „Show more info“. Fast.com izvodi više preuzimanja i učitavanja s Netflixovih servera(u ovom slučaju u Seacaucus, NJ i Ashburn, VA) i mjeri maksimalnu brzinu koju određena mrežna veza može podržati. Test je globalno dostupan[4].

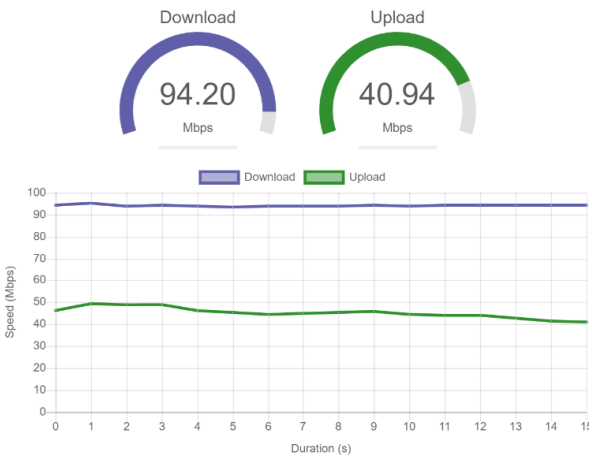



Slika 3. fast.com

Na kraju bi trebalo spomenuti češki speedtest.cesnet.cz/. U njemu se na pritisak gumba Start mjeri ping, jitter, brzina preuzimanja i brzina učitavanja te prikazuje rezultate grafički. Kod speedtest.cesnet.cz se šalje više zahtjeva unutar intervala od 100ms. Ping se mjeri kao vrijeme potrebno za odgovor kod HTTP zahtjeva. Vrijeme se dobiva preko JavaScript Performance API-ja. Jitter se računa preko standardne devijacije svih vrijednosti pinga naspram trenutnog mjerenja pinga. Brzina učitavanja i preuzimanja se mjeri preko prijenosnih blokova podataka preko HTTP zahtjeva. Također se mjeri i cijela širina pojasa, što omogućuje grafički prikaz. Ova web stranica prikazuje dva grafa. Na prvom grafu su parametri vrijeme odgovora(ms) i trajanje(ms) i služi prikazu pinga i jittera, dok se na drugom grafu izražavaju brzina preuzimanja i brzina učitavanja s parametrima brzina(Mbps) i trajanja(s)[5].




Slika 4. Ping i jitter




Slika 5. Brzina preuzimanja i učitavanja

## 2.3. Problemi u procesu mjerenja

Kod procesa mjerenja brzine mreže se vrlo često naiđe na neke probleme. U radu češkog sveučilišta u Hradec Kralovu se ističu 3 problema koja su uzrokovana podacima koji se prenose na mreži. Prvi navedeni problem je veličina slike. Općenito slike imaju veliku veličinu datoteke, a pogotovo ako su visoke kvalitete ili ako se preuzimaju na mobilnim uređajima. Zato se taj problem najčešće rješava s optimizacijom stope kompresije, izbjegavanja prevelikog broja slika ili preko toga da se na stranicama slike veće veličine učitavaju tek kad se dođe do njih. Još je jedan od problema dizajn koji se razlikuje od uređaja do uređaja. Problem dizajna se najčešće rješava preko skrivanja elemenata koji su preveliki ili korištenjem takozvanih fluidnih slika. Na kraju treba spomenuti probleme izazvane JavaScriptom. Jako je zanimljiv podatak da se u testiranju Alexa Top 100 URL-ova došlo do zaključka da se s ukidanjem JavaScripta dođe do povećanja brzina od 31% prosječno. Naime, ovo je slučaj iz razloga što neki programeri nisu dovoljno obučeni za JavaScript i onda ne znaju pravilno napisati kod za web stranicu. Sličan je slučaj u istraživanju bio i za radne okvire kao što su Bootstrap i Foundation [7].

S druge strane problemi u procesu mjerenja mogu nastati i zbog loše infrastrukturne podrške. Unatoč tome što vlasnici usluga interneta često tvrde da je određena mreža određene brzine, najčešće u stvarnosti to nije slučaj. Vrlo se često dogodi da je brzina manja nego što je oglašena kod vlasnika usluge. To se događa kada više korisnika dijeli dostupan kapacitet na mreži ili da infrastruktura(kabeli, usmjerivači, računala..) nije optimizirana za rad na većim brzinama. Naziv za taj problem je zagušenje. S druge stane može doći do slučajeva gdje brzina mreže može biti veća od oglašene i iz više razloga mogu nastati. Prvi razlog je to što se može dogoditi da u mjerenju jedan paket odskoči brzinom ili da se dva paketa pošalju uzastopce i onda se izmjeri da je ukupna brzina mreže veća nego što zapravo jest. Onda isto tako se mogu dogoditi anomalije s netrivijalnim algoritmima protokola nižeg ranga. Kao zadnji razlog može se dogoditi pojava poznata s imenom Powerboost. Powerboost je pojava koja pojačava stopu slanja podataka do čak i više od 110% oglašene brzine. Ovo je često slučaj kod pristupnih mreža koje maknu ograničenja brzine u određenim vremenskim periodima [3].

## 2.4. Ciljevi rada

Cilj ovog završnog rada je razvijanje programske podrške u svrhu stvaranja sustava za mjerenje i pohranjivanje podataka o pojedinim mrežama „Speedtest“. Ovaj sustav daje svakom korisniku mogućnost praćenja podataka o odabranoj mreži i prethodnim mjerenjima. Sustav daje mogućnosti:

- mjerenja brzine preuzimanja
- mjerenja brzine učitavanja
- mjerenja pinga

Prikaz brzine se kod brzine preuzimanja i brzine učitavanja grafički prikaže u obliku polukruga brzina u odnosu na cijeli kapacitet. Nakon mjerenja se svi zabilježeni podaci bilježe u bazu podataka(PostgresSQL). Tamo se unose podaci o:

- rednom broju mjerenja
- mreži na kojoj je mjerenje izvršeno
- datumu i vremenu
- brzini preuzimanja
- brzini učitavanja
- pingu

Svi su ovi podaci vidljivi na stranici History, gdje se prikazuju sva mjerenja koja je obavio korisnik. Također postoji u History-ju mogućnost da se pregledaju podaci za brzinu učitavanja i brzinu preuzimanja u Kb/s ili MB/S ili Mb/s, brisanja podataka i filtracije po datumu i vremenu.

# 3. Specifikacija programske potpore

Specifikacija programske potpore aplikacije mjerenja brzine mreže se može podijeliti na funkcionalne, nefunkcionalne i ostale zahtjeve. Zahtjeve postavlja korisnik razvojnom timu i po tim zahtjevima razvojni tim odlučuje jesu li zahtjevi provedivi u aplikaciji. Funkcionalni zahtjevi su zahtjevi koje razvojni tim može provesti, dok su nefunkcionalni oni koje razvojni tim nije u stanju provesti. Zahtjevi se dijele na popis dionika, aktore i njihove zahtjeve, obrasce uporabe i sekvencijske dijagrame. Za obrasce uporabe također postoje dijagrami obrazaca uporabe.

## 3.1. Funkcionalni zahtjevi

**Dionici:**

1. Administrator
2. Korisnik
3. Razvojni tim

**Aktori i njihovi funkcionalni zahtjevi:**

1. Korisnik može:
   (a)Napraviti mjerenje
   (b)Odabrati opciju pregleda povijesti mjerenja
      i. Pregled svojih već obavljenih mjerenja kroz povijest
      ii. Pregledati svoja mjerenja u drugim mjernim jedinicama
      iii. Filtrirati po datumu i vremenu
2. Administrator može:

   (a)Napraviti mjerenje

   (b)Odabrati opciju pregleda povijesti mjerenja

      i. Pregled svih već obavljenih mjerenja kroz povijest

      ii. Pregledati mjerenja u drugim mjernim jedinicama

      iii. Filtrirati po datumu i vremenu

iv. Brisanje podataka

### 3.1.1. Obrasci uporabe

Obrasci uporabe opisuju sve moguće interakcije sustava. Sastoje se od obrasca uporabe, aktora i granice sustava. Obrazac uporabe je akcija koju sustav ili drugi entitet izvodi u interakciji s aktorima sustava. Aktor je koherentan skup uloga koje imaju korisnici u interakciji s obrascima uporabe, a granica sustava predstavlja granicu između fizikalnog sustava i različitih aktora koji su u interakciji s fizikalnim sustavom[8].

**UC1-Mjerenje**

- **Glavni sudionik**: Korisnik
- **Cilj:** Prikaz podataka o brzini učitavanja, brzini preuzimanja i pingu i spremanje u bazu
- **Sudionici:** Baza podataka
- **Preduvjet:** Pristup stranici
- **Opis osnovnog tijeka:**
    1. Korisnik pritisne gumb „Start“
    2. Prikazuju se na sučelju podaci za zadano mjerenje
    3. Podaci se smještaju u bazu podataka
- **Opis mogućih odstupanja:**

    2.a Sustav nije u mogućnosti izmjeriti brzinu

        1. Sustav javlja poruku o greški

**UC2-Filtracija po datumu i vremenu**

- **Glavni sudionik:** Korisnik
- **Cilj:** Filtriranje podataka po određenim vremenskim periodima
- **Sudionici:** Baza podataka
- **Preduvjet:** Pristup stranici
- **Opis osnovnog tijeka:**

  1.Korisnik pritisne na gumb „Filtriraj“

  2. Prikaže mu se opcija odabira određenog vremenskog perioda na traci

  3. Odabere vremenski period

4. Na stranici „History“ se ažuriraju podaci u određenom vremenskom periodu

- **Opis mogućih odstupanja:**

  2.a Korisnik odustaje od odabira

  1.Vrati se u stanje prije odabira na „Filtriraj“

  4.a Ne postoje podaci za taj vremenski period

  1. Ispiši poruku „Nema mjerenja za taj vremenski period“

**UC3-Prikaži brzinu s drugom mjernom jedinicom**

- **Glavni sudionik:** Korisnik
- **Cilj:** Prikaz u drugoj mjernoj jedinici od prikazane(Kb/s ili Mb/s)
- **Sudionici:** Baza podataka
- **Preduvjet:** Pristup stranici
- **Opis osnovnog tijeka:**
  1. Korisnik pritisne gumb „Mjerna jedinica“
  2. Prikažu mu mjerne jedinice koje može odabrati
  3. Klikne na određenu mjernu jedinicu
  4. Promijeni se mjerna jedinica na prikazu u stranici „History“

**UC4-Izbriši zapis o mjerenju**

- **Glavni sudionik**: Administrator
- **Cilj:** Brisanje zapisa o pojedinom mjerenju
- **Sudionici:** Baza podataka
- **Preduvjet:** Pristup stranici
- **Opis osnovnog tijeka:**
  1. Administrator pritisne gumb „Obriši“
  2. Odabere koje zapise želi obrisati
  3. Pritisne na „Potvrdi da želiš obrisati“

**Dijagram obrazaca uporabe**

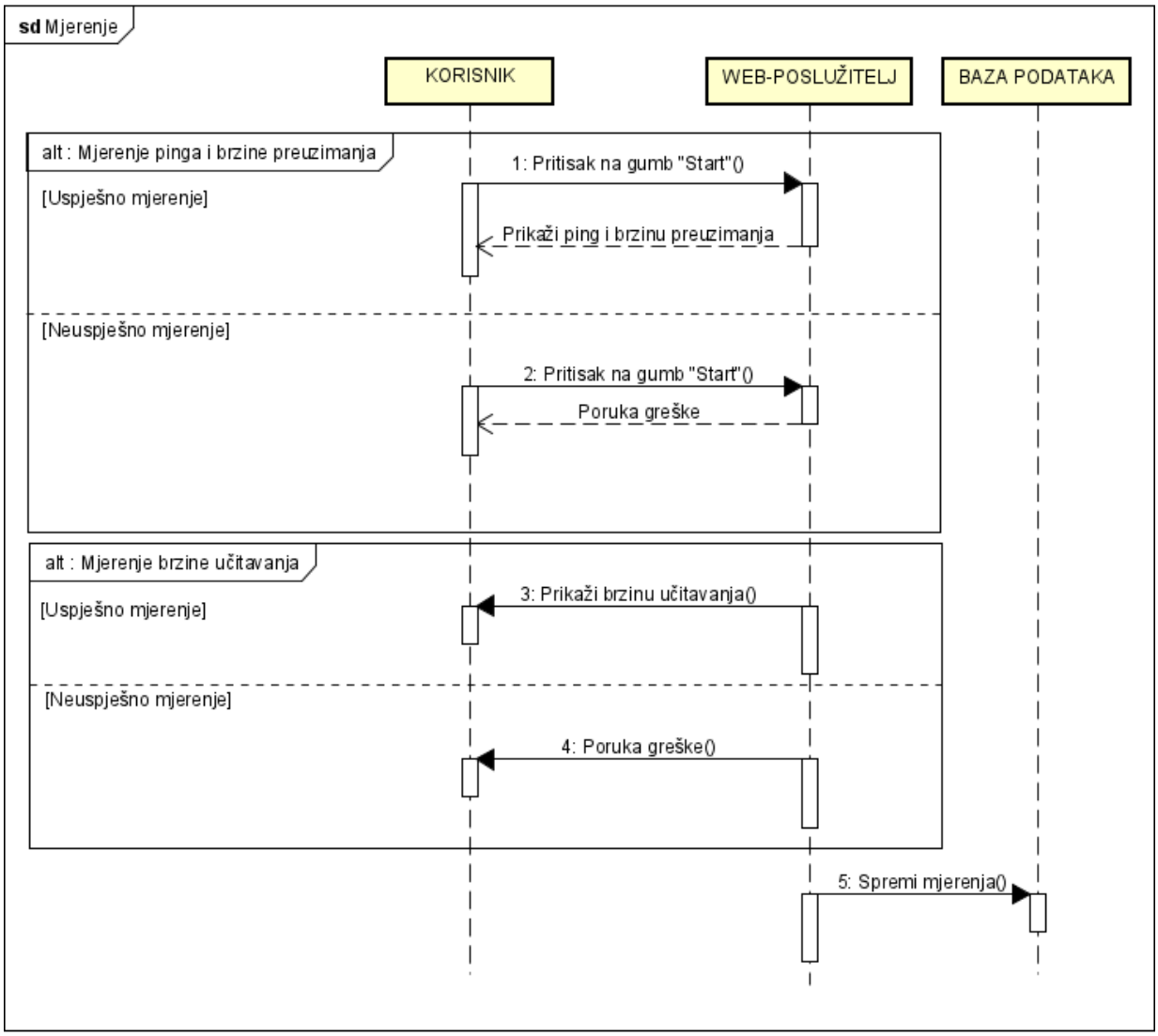
Dijagram obrazaca uporabe pokazuje obrasce uporabe, aktore i njihove odnose. Predstavlja detalje unutar dijagrama obrazaca uporabe s tekstom ili dodatnim dijagramima interakcije između sustava[8]. U ovome konkretnom slučaju s lijeve strane

se nalaze aktivni aktori korisnik i administrator i postoji generalizacija između administratora i korisnika.  Među obrascima uporabe se nalaze mjerenje, filtracija po datumu i vremenu, prikaz brzine drugom mjernom jedinicom i brisanje zapisa o mjerenju. Kao pasivni aktor baza podataka izvršava zahtjeve koje joj pošalju aktivni aktori slijeva.

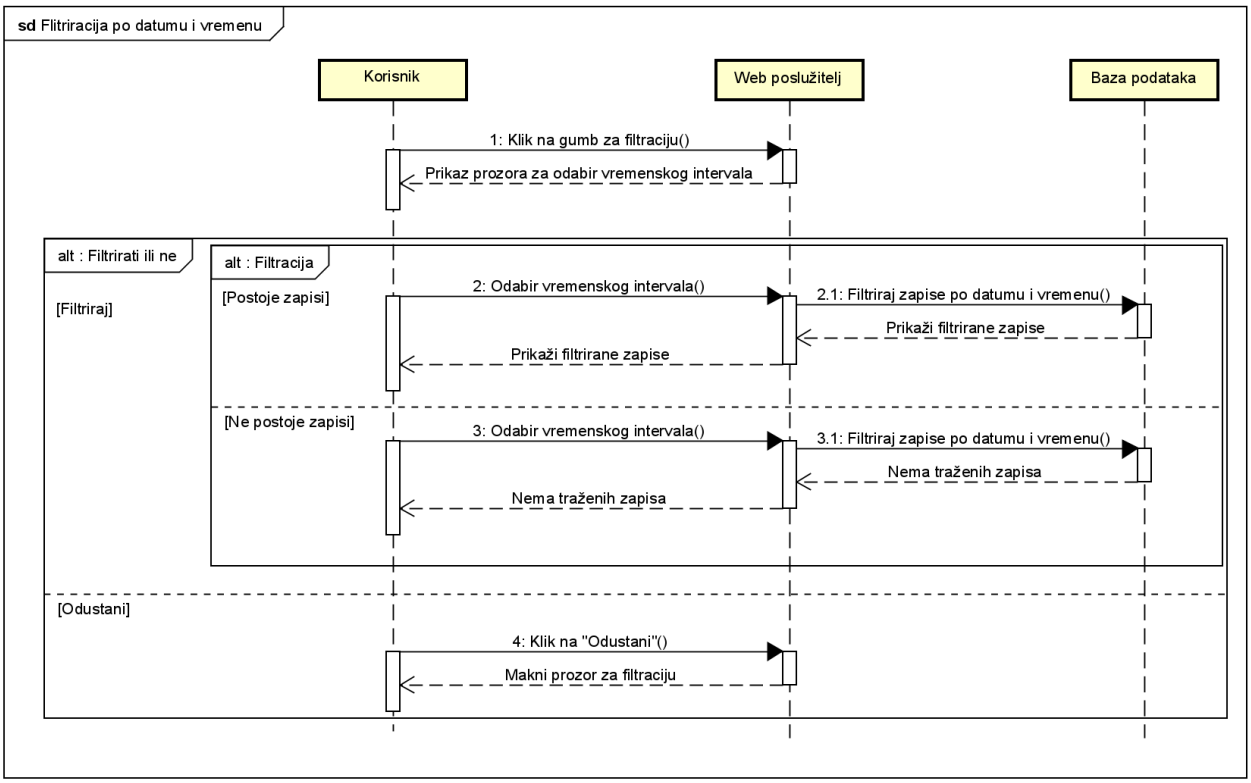

Slika 6. Dijagram obrazaca uporabe ove aplikacije

### 3.1.2. Sekvencijski dijagrami

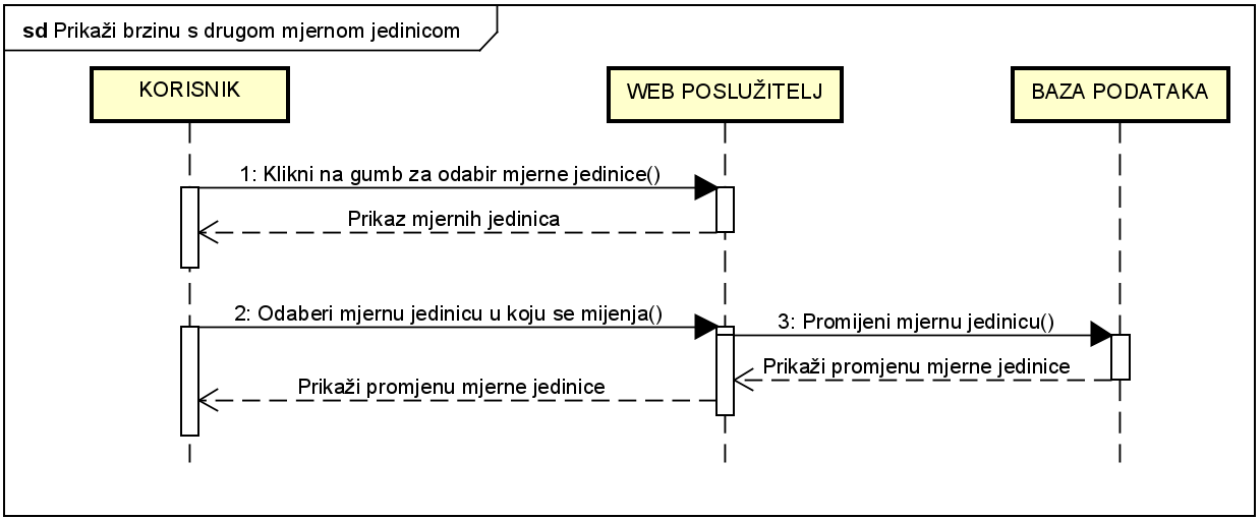
Sekvencijski dijagram je oblik dijagrama interakcije koji pokazuje objekte kao životne crte koje idu prema dnu stranice. Interakcije u vremenu se prikazuju kao poruke korištenjem strelica životne crte početnog objekta koji započinje interakciju. Ovi su dijagrami odlični u prikazu objekata koji su u međusobnoj komunikaciji i daju tijek komunikacije kroz vrijeme[9].



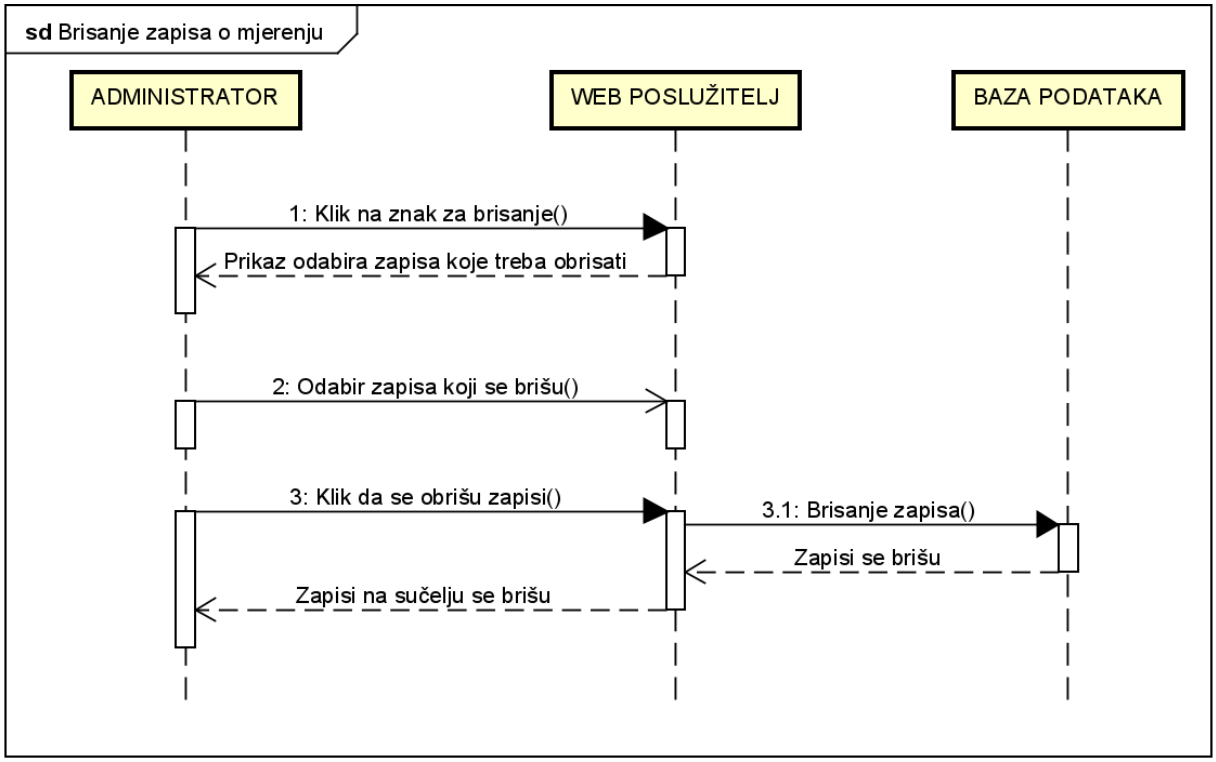

Slika 7. Sekvencijski dijagram operacije mjerenja




Slika 8. Sekvencijski dijagram operacije filtracije zapisa po datumu i vremenu




Slika 9. Sekvencijski dijagram prikaza brzine u drugoj mjernoj jedinici(Kb/s ili Mb/s)




Slika 10. Sekvencijski dijagram brisanja zapisa o mjerenju

## 3.2 Nefunkcionalni zahtjevi

- Ne postoje grafove prikaza mjerenja na stranici „Test“

## 3.3 Ostali zahtjevi

- Sustav omogućuje višestruko korištenje
- Konekcija s bazom podataka dobro zaštićena i brza i otporna na vanjske greške
- Sustav napravljen s objektno orijentiranim jezicima
- Korisničko sučelje podržava hrvatsku abecedu
- Pristup omogućen iz preglednika s HTTPS protokolom
- Brzo i efikasno izvođenje komponenti

# 4. Arhitektura i dizajn sustava

Arhitektura u ovoj aplikaciji se dijeli na 3 podsustava:

- Klijent
- Baza podataka
- Server

Klijent ima funkciju da postavlja zahtjeve serveru i preko servera dobiva informacije iz baze podataka. On komunicira preko sučelja koje je napisano u HTML-u, Bootstrap-u i CSS-u te prilikom svake promjene koju inicira klijent se preko koda napisanog u jQuery-ju i JavaScriptu prenosi zahtjev serveru. Server je simuliran na platformi kroz radni okvir Django gdje postoje sve potrebne datoteke za tu svrhu. Isto Django ima template-e koji služe kao platforma za frontend kodove.

Baza podataka u kojoj je napisana aplikacija je PostgreSQL. Ona je povezana s Djangom kroz modele koji simuliraju bazu i kroz koje se vrši komunikacija i unos u bazu. U bazi se koristi koncept Model-View-Controller i tablice s relacijama OneToOne.

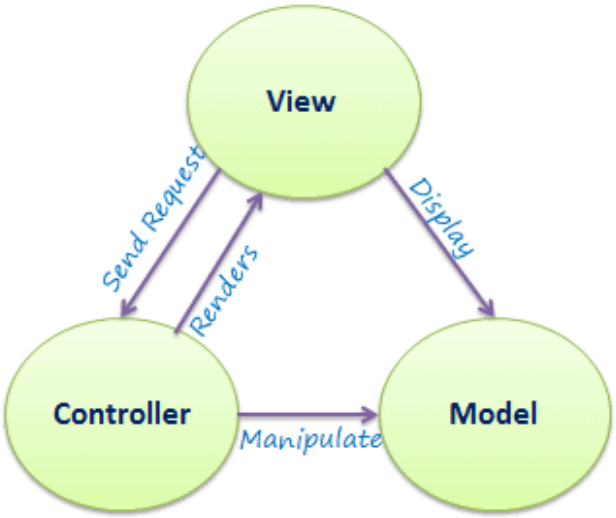
Server je računalo ili sustav koji omogućuje pristup resursima, podacima, uslugama i programima na drugim računalima(tzv. klijentima) kroz neku mrežu. Ovo u teoriji znači da bilo koje računalo se može smatrati serverom ako dijeli resurse računalima klijentima[11]. Nadalje, server vrši komunikaciju s bazom i drži vremensku sesiju. Server filtrira zahtjeve i gleda koji je uspješan, a koji nije.

## 4.1. Django

Django je radni okvir visokog levela za programski jezik Python koji omogućuje brzo razvijanje sigurnih i održivih web stranica. Jako je popularan i jako je korišten u svijetu. Među ostalim koristio se na razvijanju web stranica Instagrama, Mozille, National Geographica, Pinteresta itd. Ono što izdvaja Django je grupiranost i organizacija pri raspolaganju s podacima. Čim HTTP zahtjev stigne unutar Djanga, odmah se preusmjeri dalje u datoteke spremne za obradu tog zahtjeva. Bitne datoteke su models.py, views.py, .html file, urls.py itd. Više o ovome u idućim poglavljima[16].

## 4.2. MVC(Model-View-Controller arhitektura)

MVC arhitektura služi u svrhu dijeljenja odgovornosti za što efikasniju suradnju. Sastoji se od tri dijela: modela, kontrolera i pogleda[12].




Slika 11. MVC arhitektura u Djangu[13]

Model je dio sustava koji je glavni u postavljanju svih zadataka vezanih za podatke, od validacije, stanja sesije i kontrole. Također zadužen je za enkapsulaciju metoda za pristup podacima i stvara razrede biblioteka za višestruku uporabu. Mogućnosti sustava modela daju mu opciju kontroliranja struktura podataka i samih podataka u bazi. Što se konkretno tiče Djanga, sustav koristi XML datoteke u svrhu spremanja datoteka između migracija. U Djangu je predstavljen kao models.py datoteka, koja ima direktnu vezu s bazom podataka

S druge strane pogled je zadužen za sučelje, što predstavlja sve forme, gumbe, grafičke elemente i sve ostale HTML,CSS ili JavaScript elemente. Pogled postoji u MVC arhitekturi iz razloga što je prije bilo jako komplicirano razdvojiti logičku stranu aplikacije i frontend dio. Tako je omogućen lakši proces kodiranja i lakša dijagnostika problema. Pogled se kod Djanga može vidjeti u dijelu koji se naziva „Templates“ i koristi se u kombinaciji s Jinja template engine-om.

Kontroler je zadužen za procesiranje događaja koje inicira ili korisnik ili sustav i služi da bi prihvaćao zahtjeve, pripremao podatke za odgovor i formatirao način odgovora. Također vrši komunikaciju s modelom da bi dohvatio potrebne podatke iz tablice u bazi i generirao pogled. Baš zato je poznat  drugim imenom kao akcija. U Djangu se kontroler nalazi u view datoteci

koja je  u komunikaciji s frontendom(html datotekom) preko urls datoteke koja služi kao putokaz prema traženoj datoteci. S druge strane kad view datoteka želi komunicirati s bazom podataka, onda to radi preko ugrađenih paketa u Django. Ti paketi se upotrebljavaju u kombinaciji s models.py datotekom[12].

## 4.3. Baza podataka

Kako se za ovu aplikaciju prikuplja puno podataka, potrebno je naći neku dobru bazu podataka koja može podnijeti opterećenja na aplikaciju. U ovoj aplikaciji se koristi baza podataka PostgreSQL. PostgreSQL je moćan otvoren sustav baza podataka koji koristi jezik SQL i može koristiti najsloženije poslove. Poznata je po tome što ima reputaciju jake arhitekture, integriteta, robusnosti i predanosti open source poslovanju. Štoviše, omogućuje korisniku definiranje svojih tipova podataka, pisanje svojih funkcija i smanjuje upotrebu loših ideja u arhitekturi[14].  U Djangu postoje automatski generirane baze podataka s tablicama auth_user, django_session i ostalim izvedenim tablicama[15]. One su sve vidljive u bazi PostgreSQL gdje za ovu aplikaciju postoje tablice: Auth_user, Django_session i Measurement

### 4.3.1. Opis tablica

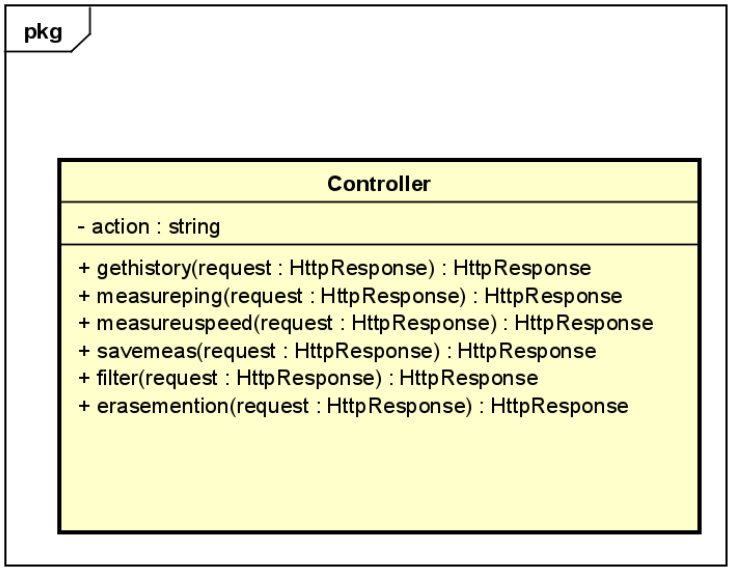
| **Measurements** | | |
|---|---|---|
| Number (PK) | INT | Redni broj mjerenja |
| Network | VARCHAR | Ime mreže |
| Date&time | TIMESTAMP | Datum i vrijeme mjerenja |
| Download speed | DOUBLE | Izmjerena brzina preuzimanja |
| Upload speed | DOUBLE | Izmjerena brzina učitavanja |
| Ping | INT | Izmjereni ping |

### 4.3.2. Dijagram baze podataka

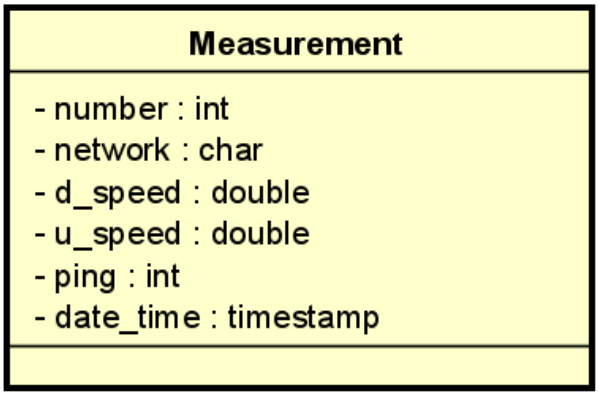
UML Dijagram baze podataka je grafički prikaz baza podatka i njihovih međusobnih odnosa. U ovom slučaju dijagram baze podataka je jednostavan i ekvivalentan opisu tablice „Measurements“ u prošlom dijelu.

## 4.4. Dijagram razreda

UML Dijagram razreda je strukturni dijagram koji prikazuje razrede u objektno orijentiranom sustavu, njegove atribute i metode te veze između razreda koji međusobno komuniciraju ili se nasljeđuju[17]. Njegovi osnovni elementi su razredi i relacije. Razred je tip podataka koji se sastoji od atributa(osnovna obilježja razreda) i operacije(metode kojima se ostvaruju odgovornosti nekog razreda). Relacije su veze između razreda koje mogu biti: pridruživanje, generalizacija, ovisnost i realizacija[17].



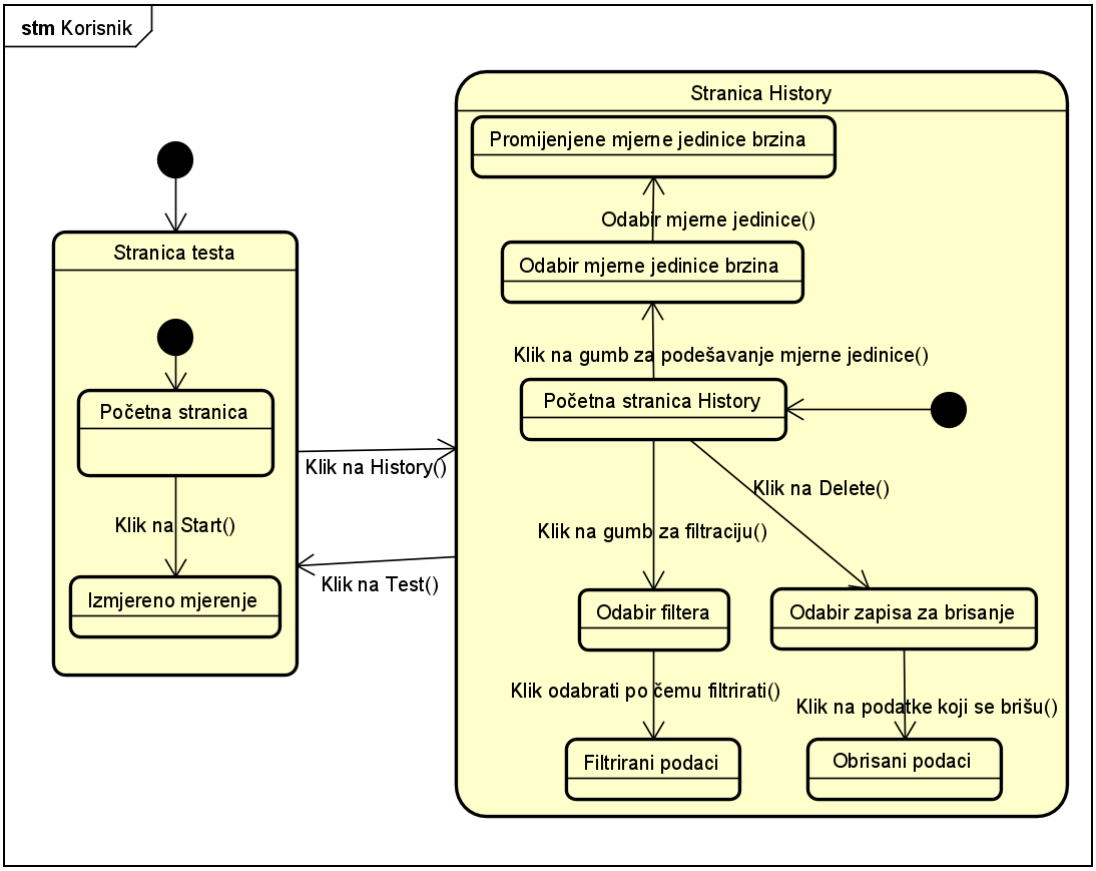

Slika 12. Dijagram razreda-controllers




Slika 13. Dijagram razreda-models

## 4.5. Dijagram stanja

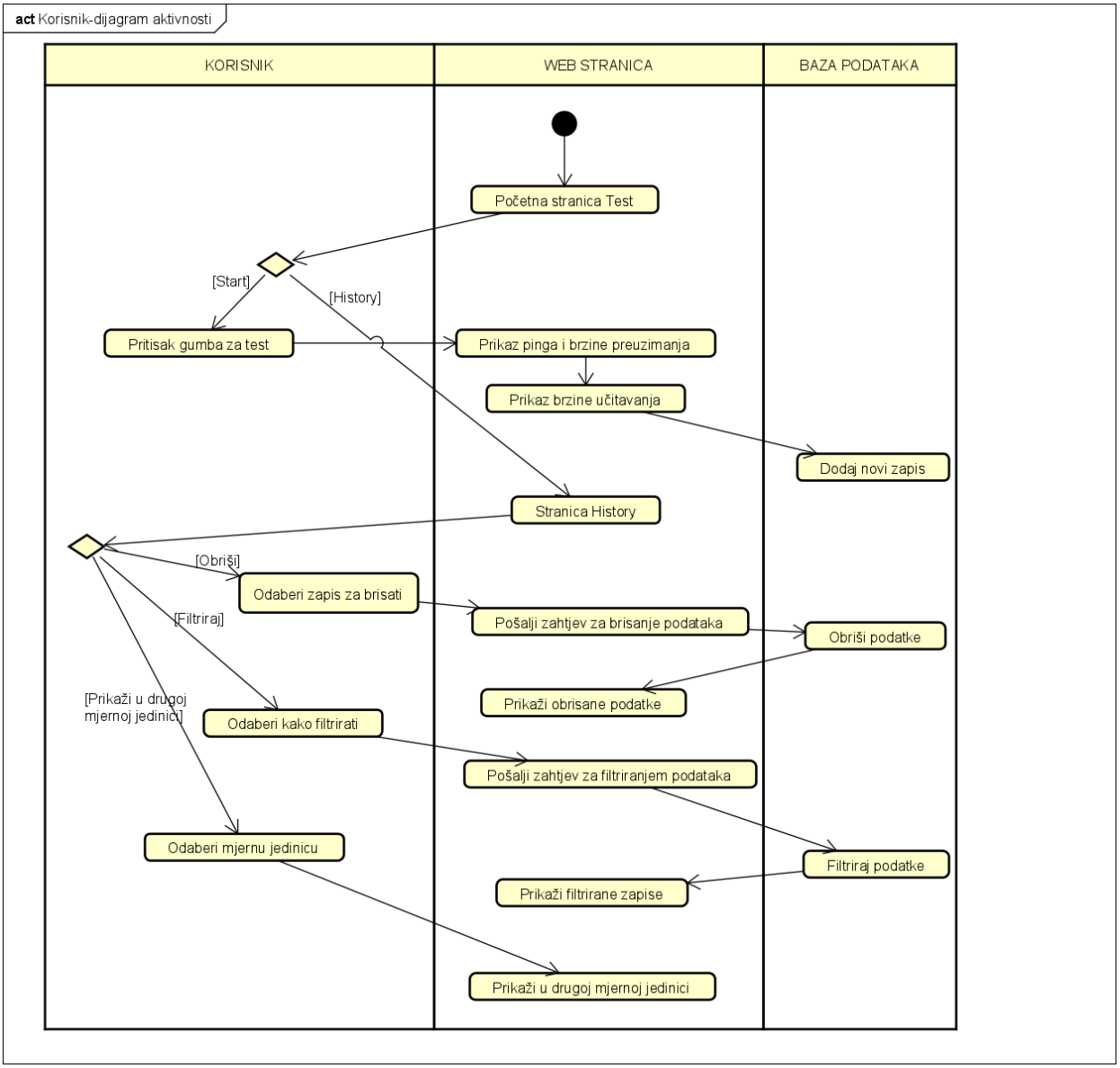
UML Dijagram stanja opisuje dinamičko ponašanje dijela sustava u jedinici vremena te prikazuje stanje objekata i prijelaze iz jednog stanja u drugom s temeljem u događajima. Sastoji se od stanja i prijelaza. Stanje objekta je definirano skupom invarijanti i može automatski izvoditi određene radnje. Ove se radnje opisuju s akcijama: entry, do, interni prijelazi i exit. S druge strane prijelazi su dozvoljene promjene stanja iz trenutnog u novo stanje koji su potaknuti događajima uz zadovoljenje nekog uvjeta. Nadalje, dijagram stanja omogućuje i opcije uvjetnih grananja, raznih pseudostanja, složenih stanja itd[18].




Slika 14. Dijagram stanja za korisnika

## 4.6. Dijagram aktivnosti

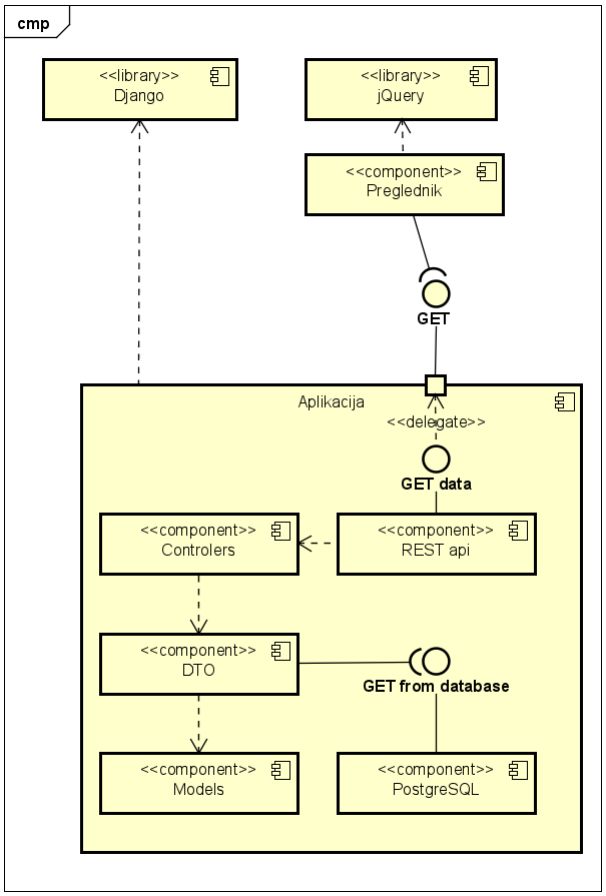
Dijagram aktivnosti je UML dijagram koji se grafički modelira ponašanja nizom akcija. Sastoji se od čvorova, veza i particija(plivaće staze)[18]. Čvor je fizički resurs koji se koristi u sustavu i ima procesni kapacitet i memoriju. Veze su spojevi između čvorova, a particija aktivnosti je stupac dijagrama aktivnosti koji se koristi u svrhu prikazivanja područja odgovornosti svakog sudionika[19].




Slika 15. dijagram aktivnosti za korisnika

## 4.7. Dijagram komponenti

UML dijagram komponenti je strukturni, statički dijagram koji omogućuje vizualizaciju organizacije i međuovisnosti između implementacijskih komponenti te odnos programske potpore i okoline. Sastoji se od komponenti, sučelja i poveznica. Komponente su zasebna enkapsulirana cjelina programske potpore s dobro definiranim sučeljem. Sučelje je imenovan skup javno vidljivih atributa i apstraktnih operacija i mogu biti ponuđena ili zahtijevana. I na kraju poveznice mogu biti spojnice, delegacije ili ovisnosti[18].




Slika 16. Dijagram komponenti

# 5. Implementacija i korisničko sučelje

U svrhu implementacije aplikacije i korisničkog sučelja je potrebno opisati sve korištene tehnologije i alate, ispitivanja, dijagram razmještaja i upute za pokretanje. Kod korištenih tehnologija i alata pišu svi alati i tehnologije za komunikaciju, razvoj backend, razvoj frontenda, razvojnog okruženja, baze podataka, izrade UML grafova i frontend-backend komunikacije. Kod ispitivanja je opisano ispitivanje komponenti i ispitivanje sustava. Onda slijedi dijagram razmještaja, gdje je opisana struktura topologije sustava. I na kraju je opisan način puštanja sustava u pogon

## 5.1. Korištene tehnologije i alati

- Komunikacija s mentorom
    - Outlook office[1]
- Backend
    - Django[2]
    - Python 3.7.9[3]
    - Speedtest -cli
- Frontend
    - HTML[4]
    - Bootstrap[5]
    - JavaScript[6]
    - CSS
- Frontend-Backend komunikacija
    - jQuery[7]
    - Ajax[8]

[1] https://outlook.office.com/mail/
[2] https://www.djangoproject.com /
[3] https://www.python.org/downloads/release/python-379/
[4] https://html.com/
[5] https://getbootstrap.com/
[6] https://www.javascript.com/
[7] https://jquery.com
[8] https://api.jquery.com/jquery.ajax/

- Razvojno okruženje
    - Microsoft Visual Studio[9]
- Izrada UML grafova
    - Astah professional[10]

- Baza podataka
    - PostgreSQL[11]
    - pgAdmin4[12]

## 5.2. Ispitivanje programskog rješenja

### 5.2.1. Ispitivanje komponenti

U radnom okviru Django postoje jedinični testovi koji koriste Pythonove standardne biblioteke modula unittest. Ovaj test definira testove koristeći pristup temeljen na klasama i simulira se tako da ne ostane trajno pohranjen u računalu. Funkcionira na način da pronađe svaku datoteku čije ime počinje s test i da ga provjeri. Na kraju treba u cmd-u pokrenuti test s naredbom „./manage.py test“[20]. Razlog ispitivanja komponenti je to što se treba utvrditi jesu li komponente nezavisne jedna od druge i jesu li komponente funkcionalne same od sebe. Traži isto tako da se svaka metoda samostalno testira

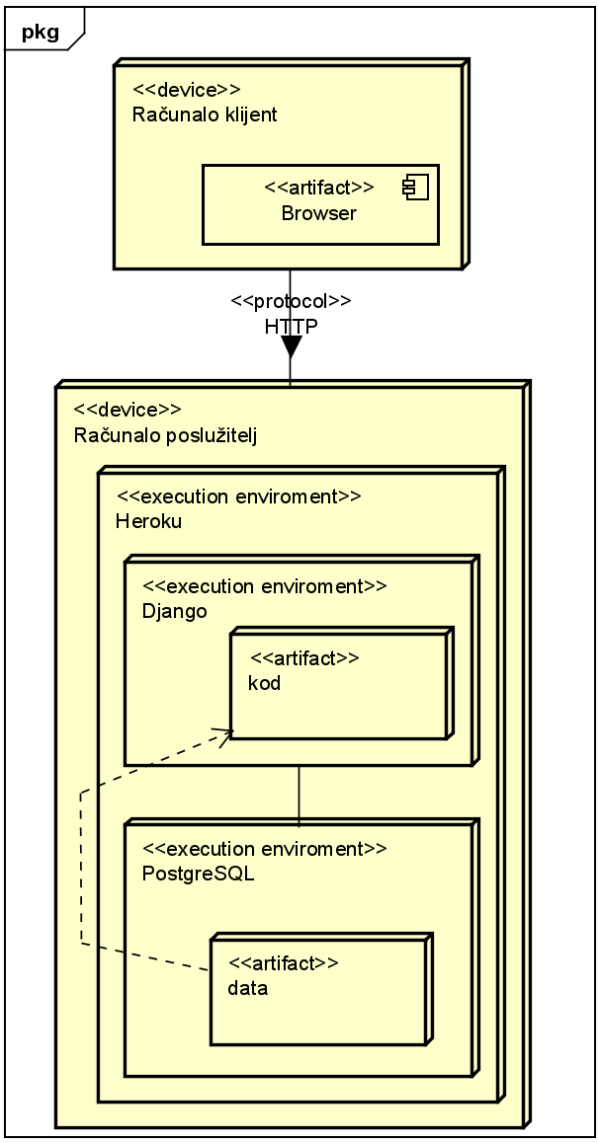
### 5.2.2. Ispitivanje sustava

Ispitivanje sustava testira funkcionalnosti sustava prema zadanim funkcionalnim zahtjevima nakon što se testiraju komponente pojedinačno. Ispitivanje sustava ispituje komunikaciju između pojedinačnih komponenti. Pojava greške u ovom koraku pokazuje da se radi se o velikoj sustavnoj grešci.

[9] https://visualstudio.microsoft.com
[10] https://astah.net/products/astah-professional/
[11] https://www.postgresql.org
[12] https://www.pgadmin.org

## 5.3. Dijagram razmještaja

UML dijagram razmještaja je strukturni statički UML dijagram koji služi opisivanju topologije sustava i usredotočen je na odnos sklopovskih i programskih dijelova. Dijagram razmještaja se može podijeliti na vrste: specifikacijski dijagram razmještaja, dijagram razmještaja instanci, implementacijski dijagram i dijagram mrežne arhitekture. Isto tako se sastoji od osnovnih elemenata: čvorovi(uređaji i okolina izvođenja), artefakti(implementirani moduli i podaci) i spojevi(komunikacijski putevi). Čvorovi mogu biti uređaji i okoline izvođenja[18].




Slika 17. dijagram razmještaja

## 5.4. Upute za puštanje u pogon

Za puštanje u pogon je zadužena Heroku platforma koja služi za hostanje softvera. Za puštanje u pogon ove aplikacije je potrebno konfigurirati Django okruženje za Heroku. Za ovo je prvo potrebna Procfile datoteka koja se nalazi u korijenu repozitorija. Nekada je potrebno uz Procfile datoteku instalirati i datoteku Gunicorn server za Python aplikacije. Nakon toga treba napraviti promjene u datoteci settings.py unutar Djanga. Ovo se radi skidanjem django-heroku paketa koji automatski konfigurira postavke u Djangu da bi Django mogao raditi s Heroku-om. Iza toga je potrebno izrazom „import django_heroku“ uvesti django-heroku paket u settings.py i na kraju dodati metodu za aktivaciju Heroku-a u settings.py[21].

# 6. Zaključak

U završnom radu je obrađena problematika testiranja, pohrane i integracije podataka o mreži kroz razne aspekte koji ih pobliže opisuju. Detaljno su analizirani podaci o svim podacima vezanim uz mjerenje te su se na temelju opažanja izveli zaključci. Iz zaključaka je u sklopu rada razvijena aplikacija. Aplikacija daje razne mogućnosti koje omogućuju što jednostavnu uporabu i korištenje i omogućuje da se na svakom računalu, koje koristi aplikaciju, izmjere statistike mreže. U tu svrhu su odabrane tehnike i arhitektura programa. Kroz rad s tim tehnikama i arhitekturom može se zaključiti da se daljnji razvoj lako može ostvariti i da je moguća budućnost aplikacije svijetla.

Sve u svemu, rad na aplikaciji je bio dvostran. S jedne strane se prikupe nova znanja i razna iskustva s raznim procesima, dok s druge strane je oduzima dosta vremena u učenju, pretraživanju i primjeni.

# 7. Literatura

## 7.1. Literatura za pisani dio rada


1. https://journalistsresource.org/economics/rural-broadband-coronavirus/
2. https://www.researchgate.net/publication/252362569_Comparison_of_AT-Tester_with_Other_Popular_Testers_for_Quality_of_Service_Experience_QoSE_of_an_Internet_Connection1
3. https://www.researchgate.net/publication/228280854_Understanding_Broadband_Speed_Measurements
4. https://fast.com/
5. https://speedtest.cesnet.cz/
6. https://www.speedtest.net/
7. https://www.researchgate.net/publication/300349759_Loading_Speed_of_Modern_Websites_and_Reliability_of_Online_Speed_Test_Services
8. https://www.zemris.fer.hr/predmeti/opp/Notes/Procesi_zahtjeva.ppt
9. Oblikovanje programske potpore FER, 03 prezentacija
10. https://towardsdatascience.com/working-structure-of-django-mtv-architecture-a741c8c64082
11. https://www.paessler.com/it-explained/server
12. https://www.researchgate.net/publication/275540078_Designing_an_MVC_Model_for_Rapid_Web_Application_Development
13. https://www.tutorialsteacher.com/mvc/mvc-architecture
14. https://www.postgresql.org/about/
15. https://docs.djangoproject.com/en/3.2/
16. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction
17. Programsko inženjerstvo FER,06 prezentacija
18. Programsko inženjerstvo FER,07 prezentacija
19. https://highered.mheducation.com/sites/0077110005/student_view0/glossary.html
20. https://docs.djangoproject.com/en/3.2/topics/testing/overview/
21. https://devcenter.heroku.com/articles/django-app-configuration

## 7.2. Literatura za praktični dio rada


1. https://www.youtube.com/watch?v=FnUkVcQ_3CQ&t=53s
2. https://www.youtube.com/watch?v=DjIFK51Mo74
3. https://www.youtube.com/watch?v=OTmQOjsl0eg&t=992s
4. https://www.youtube.com/watch?v=UkokhawLKDU
5. https://game-icons.net/1x1/delapouite/hamburger-menu.html
6. https://www.speedcheck.org/
7. https://www.highspeedinternet.com/tools/speed-test
8. https://www.geeksforgeeks.org/test-internet-speed-using-python/
9. https://stackoverflow.com/questions/48043089/javascript-check-if-html-p-tag-is-empty
10. https://www.programcreek.com/python/example/117126/speedtest.Speedtest
11. https://stackoverflow.com/questions/36787908/how-to-check-if-date-is-in-this-week-in-javascript
12. https://www.geeksforgeeks.org/how-to-pass-data-to-javascript-in-django-framework/
13. https://linuxtut.com/en/367599a00bcd709de512/
14. https://www.codegrepper.com/code-examples/python/python+datetime+get+date+one+week+from+today
15. www.speedtest.net
16. https://getbootstrap.com/
17. https://www.w3schools.com/css/css_dropdowns.asp