Napredne veb tehnologije – Specifikacija predmetnog projekta

Softversko inženjerstvo i informacione tehnologije 2024/2025

1. Uvod

Vaš zadatak je da projektujete i implementirate softversko rešenje koje omogućava monitoring potrošnje električne energije velikog broja domaćinstava. Zamislite idealan scenario u kom svako domaćinstvo (kuća ili stan) ima svoje pametno brojilo koje očitava potrošnju električne energije i periodično automatski javlja centrali potrošnju energije domaćinstva. Vlasnik domaćinstva svaki mesec prima račun putem email-a, a detalje o potrošnji električne energije može da vidi putem platforme dostupne putem interneta koja mu omogućava uvid u potrošnju električne energije domaćinstva tokom dužeg vremenskog perioda. Platforma je aplikacija u oblaku koju može da koristi veliki broj vlasnika različitih domaćinstava i da svako prati potrošnju svog domaćinstva. Službenici koriste platformu kako bi imali uvid u svoj raspored. Administratori te platforme mogu da je koriste kako bi pratili rad celog sistema i kako bi adminsitrirali podatke u sistemu.

2. Uloge korisnika

Platforma razlikuje sledeće vrste korisnika:

- Redovan registrovan korisnik (građanin): može da registruje nova domaćinstva, zatraži pristup podacima o potrošnji električne energije nekog domaćinstva, kao i da nadzire potrošnju električne energije domaćistva;
- Službenik: Službenik radi sa strankama i ima uvid u svoj raspored
- Administrator: ima pravo nadzora i upravljanja sistema u realnom vremenu; **administrator ne može da izvršava nijednu funkcionalnost koja pripada redovnom korisniku, dok redovni korisnik ne može da izvršava nijednu funkcionalnost koja pripada administratoru**;
- Neautentifikovani korisnik: Ima mogućnost registracije i prijave na sistem.

3. Domaćinstvo

Domaćinstvo (kuća ili stan) je potrošač električne energije. Svako domaćinstvo ima brojilo koje samostalno očitava potrošnju električne energije i automatski šalje očitanu vrednost platformi. Za potrebe ovog projekta, svako domaćinstvo ima svoju adresu (ulica i broj) i jedinstveni identifikator. Stoga, kako svako domaćinstvo ima tačno jedno brojilo, a svako brojilo se može

identifikovati kombinacijom ova dva podatka. Smatraćemo da je svako brojilo jednotarifno radi pojednostavljenja obračuna zaduženja domaćinstva za potrošnju električne energije.

Kako ne bismo kupovali brojila koja su skupa i komplikovana za korišćenje da testiramo rad našeg sistema, potrebno je simulirati potrošnju domaćinstva pomoću malih i jednostavnih aplikacija (npr. konzolnih) koje imaju ulogu simulatora domaćinstva (u nastavku ćemo ove aplikacije zvati **simulatori**).

Simulator simulira potrošnju električne energije domaćinstva tako što 1 minut simulacije predstavlja 1 sat potrošnje električne energije. Simulator periodično, 1 u minutu, šalje nasumično generisanu vrednost koja predstavlja utrošenu električnu energiju u prethodnom satu. To bi značilo da za 24 minuta rada simulatora on simulira 1 dan (24 časa) potrošnje domaćinstva. Nasumično generisane vrednosti treba da budu iz nekog smislenog opsega i da što približnije simuliraju ponašanje domaćinstva. U toku dana potrošnja je znatno veća nego npr. noću kada članovi domaćisntva najčešće spavaju.

Prilikom implementacije simulator potrebno je ispuniti sledeće zahteve:

1. Simulator periodično javlja platformi da je online (heartbeat) - u slučaju da se neko duže vreme nije javio da je online (npr. 30 sekundi), platforma treba da smatra da više nije na mreži i to treba da zabeleži u bazi podataka
2. Simulator komunicira sa platformom putem AMQP protokola
3. Jednom u minuti simulator šalje simuliranu potrošnju domaćinstva u prethodnom satu; simulator lokalno pamti kada je poslednji put uspešno poslao simuliranu vrednost
4. Simulator lokalno čuva očitane vrednosti potrošnje električne energije za poslednjih 90 dana (simuliranje slučaja kada se brojilo pokvari usled čega je potrebno da osoba manuelno očita potrošnju)
5. U slučaju da se desi kvar na mreži i da simulator nije u mogućnosti da uspešno isporuči novu očitanu vrednost, potrebno je da zapamti simuliranu vrednost; kada se ponovo uspostavi konekcija sa platformom, simulator bi trebao da pošalje sva merenja koja nisu bila na vreme poslata

4. Funkcionalni zahtevi

4.1. Inicijalno stanje sistema

Kada se softversko rešenje pokrene, potrebno je automatski (prilikom prvog podizanja aplikacije) generisati superadmin nalog (sa username-om **admin**). Predefinisa lozinka superadmina treba da bude nasumično generisana tekstualna vrednost koja sadrži veći broj karaktera. Predefinisanu lozinku u izvornom obliku sačivati u tekstualnu datoteku, a lokaciju ove datoteke navesti u uputstvu za pokretanje aplikacije. Nakon prve prijave, potrebno je zahtevati od superadmina promenu predefinisane lozinke i ne dozvoliti mu pristup aplikaciji dok to ne

uradi. Superadmin može da dodaje druge admine u sistem i da radi sve što i oni mogu da rade u sistemu. Kada superadmin doda novog admina, nalog novododatog admina biva automatski aktivan. Obični admini ne mogu da dodaju druge admine.

4.2. Registracija i prijava korisnika

Dizajnirati početnu stranu koju vidi neprijavljeni korisnik sa linkovima na stranice za registraciju i prijavu. Omogućiti registraciju i prijavu novih građana na sistem. Nakon što korisnik popuni formu za registraciju (u sklopu koje **upload-uje svoju profilnu sliku**) i uspešno se registruje, njegov nalog nije aktivan, već sistem šalje email na unetu adresu korisnika sa linkom za aktivaciju naloga. Nakon što se nalog aktivira, korisnik može da se prijavi. Prijavljeni korisnik u bilo kom trenutku može da se odjavi sa sistema.

4.3. Komunikacija simulatora sa platformom

Implementirati inicijalnu verziju simulatora i tom prilikom ispuniti zahteve 1, 2 i 3 iz poglavlja 3. Kada simulator ostvari konekciju sa platformom, on periodično javlja da je na mreži. U slučaju da duže vreme simulator ne javi da je na mreži, platforma treba automatski da označi da nije na mreži. Komunikacija simulatora i platforme se izvodi putem AMQP protokola sa posrednikom (broker). Demonstrirati pokretanjem 3 različite instance simulatora koji nezavisno jedan od drugog komuniciraju sa platformom.

4.4. Pregled dostupnosti brojila domaćinstava (simulatora)

Administrator može da pretraži sva domaćinstva u sistemu. Implementirati efikasan mehanizam pretrage domaćinstava bez vlasnika u sistemu. Potrebno je istražiti kako popularne aplikacije za rad sa nekretninama (npr. katastar ili aplikacije za prodaju i izdavanje nekretnina) ovo rade i po uzoru na njih implementirati najefikasnije rešenje za pronalazak domaćinstva. Izborom domaćinstva može da vidi sve njegove detalje. Za izabrano domaćinstvo može da vidi grafike dostupnosti. Korisnik može da izabere vremenski period u kom želi da vidi dostupnost simulatora (npr. poslednji 3h, poslednjih 6h, poslednih 12h, poslednja 24h, poslednjih nedelju dana, mesec dana, 3 meseca, godinu dana ili da izabere datume od-do pri čemu razlika između ta dva datuma ne sme da bude veća od godinu dana). Za izabrani vremenski period, korisnik vidi koliko je procentualno i vremenski simulator bio na mreži i koliko nije bio na mreži (koliko je bio online, a koliko offline). U slučaju da je u pitanju kraći vremenski period (npr. poslednih 6h), prikazati dostupnost simulatora unutar svakog sata, dok ako je izabran duži vremenski period (npr. 1 mesec), prikazati dostupnost simulatora unutar svakog dana ili nedelje. Dostupnost simulatora prikazati pomoću različitih grafika (bar chart, plot-ovi itd.). U slučaju da je izabran period od poslednjih 3 sata, omogućiti prikaz u realnom vremenu. Dobavljanje podataka radi prikaza akcija u realnom vremenu ostvariti pomoću WebSocket protokola.

4.5. Podnošenje zahteva za registraciju nekretnine sa domaćinstvima

Običan korisnik (osoba koji gradi kuću ili stambenu zgradu) može da preda zahtev za registraciju nekretnine. Pored osnovnih informacija (poput adrese, grada, broja spratova itd.), jedne ili više slika nekretnine (koje se **upload-uju**), mora da navede lokaciju nekretnine tako što je označi na klikom na mapi (koristiti bilo koje dostupno rešenje: npr. OpenLayers map, Leaflet, Modest maps, Polymaps itd.). Za izbor grada ponuditi izbor jednog od gradova iz predefinisanog spiska gradova koji se nalazi u bazi podataka. Takođe, potrebno je da priloži jednu ili više slika i/ili jedan ili više PDF dokumenata kojima dokazuje da ima pravo da registruje nekretninu. Nakon popunjavanja ovih podataka, korisnik dodaje jedno ili više domaćinstava unutar novokreirane nekretnine pri čemu za svako od njih navodi sprat (prizemlje, prvi, drugi itd.), kvadraturu i jedinstveni identifikator domaćinstva unutar nekretnine (npr. broj stana unutar zgrade). Kada se kreira zahtev, on je na čekanju da ga administrator odobri ili odbije. Takođe, korisnik može da vidi spisak svih svojih zahteva sa njihovim statusom i vremenom podnošenja i zaključivanja, koji može filtrirati i sortirati po različitim parametrima. Jedan korisnik može da kreira više zahteva.

4.6. Obrada zahteva za registraciju nekretnine

Administrator ima pregled svih zahteva za registraciju nekretnina na čekanju i može da ih prihvati ili odbije uz obrazloženje. Jednom kada je zahtev odbijen ili prihvaćen, njegov status se ne može menjati. Ako administrator prihvati, nekretnina i sva domaćinstva koja joj pripadaju bivaju dostupna i sistem korisniku čiji je zahtev prihvaćen šalje email sa potvrdom. Ako administrator odbije, mora da navede obrazloženje zašto odbija i korisniku čiji je zahtev odbijen šalje email sa informacijom da je zahtev odbijen uz obrazloženje zašto. Stranica sa pregledom svih zahteva je početna stranica administratora nakon što se prijavi na sistem.

4.7. Podnošenje zahteva za potvrdu vlasništva domaćinstva

Običan korisnik može da zatraži pristup podacima nekog domaćinstva tako što će potvrditi da je njegov vlasnik. Implementirati efikasan mehanizam pretrage domaćinstava bez vlasnika u sistemu. Potrebno je istražiti kako popularne aplikacije za rad sa nekretninama (npr. katastar ili aplikacije za prodaju i izdavanje nekretnina) ovo rade i po uzoru na njih implementirati najefikasnije rešenje za pronalazak domaćinstva. Kada korisnik pronađe domaćinstvo, potrebno je da priloži jednu ili više slika i/ili jedan ili više PDF dokumenata kojima dokazuje vlasništvo nad domaćinstvom. Kada se kreira zahtev, on je na čekanju da ga administrator odobri ili odbije. Takođe, korisnik može da vidi spisak svih svojih zahteva sa njihovim statusom i vremenom podnošenja i zaključivanja, koji može filtrirati i sortirati po različitim parametrima. Jedan korisnik može da kreira više zahteva.

4.8. Obrada zahteva za potvrdu vlasništva domaćinstva

Administrator ima pregled svih zahteva za potvrdu vlasništva domaćinstva na čekanju i može da ih prihvati ili odbije uz obrazloženje. Jednom kada je zahtev odbijen ili prihvaćen, njegov status se ne može menjati. Ako administrator prihvati, korisnik koje je podneo zahtev dobija mogućnost da upravlja domaćinstvom i sistem korisniku čiji je zahtev prihvaćen šalje email sa potvrdom. Ako administrator odbije, mora da navede obrazloženje zašto odbija i korisniku čiji je zahtev odbijen šalje email sa informacijom da je zahtev odbijen uz obrazloženje.

4.9. Davanje pristupa podacima domaćinstva

Vlasnik domaćinstva može da dodeli pristup podacima domaćinstva drugim korisnicima. Potrebno je implementirati efikasan mehanizam pretrage korisnika u sistemu. Jednom kada vlasnik pronađe traženog korisnika, može mu dodeliti pristup. Potrebno je istražiti kako popularne aplikacije sa velikim brojem korisnika (poput društvenih mreža) ovo rade i po uzoru na njih implementirati najefikasnije rešenje. Osoba koja je dobila pristup domaćinstvu ima pristup svim funkcionalnostima domaćinstva (plaćanje računa, uvid u račune, potrošnju itd.), osim dodeli prava drugim korisnicima.

4.10. Pregled potrošnje električne energije domaćinstva

Omogućiti vlasniku domaćinstva praćenje potrošnje električne energije. Implementirati prikaz potrošnje domaćinstva po mesecima u vidu bar chart-a. Korisnik vidi 12 podataka za 12 meseci i može da menja kojih 12 meseci je prikazano (kreće se napred/nazad u prošlost/budućnost) Klikom na neki mesec može videti sličan bar chart za sve dane u izabranom mesecu. Korisnik može da izabere vremenski period u kom želi da vidi pregled potrošnje (npr. poslednji 1h, poslednjih 6h, poslednih 12h, poslednja 24h, poslednjih nedelju dana, mesec dana, 3 meseca, godinu dana ili da izabere datume od-do pri čemu razlika između ta dva datuma ne sme da bude veća od godinu dana). U slučaju da je izabran period od poslednjih sat vremena, omogućiti prikaz u realnom vremenu. Dobavljanje podataka radi prikaza akcija u realnom vremenu ostvariti pomoću WebSocket protokola.

4.11. Monitoring potrošnje električne energije

Omogućiti administratoru da prati potrošnju električne energije. Administrator ima spisak svih domaćinstava u sistemu (nastavak na 4.4) i za svaku može da vidi koliko električne energije je potrošeno u različitim danima, mesecima i godinama Takođe, vidi i spisak svih gradova u kojima se nalaze registrovane nekretnine i za svaki grad može da vidi koliko su sve nekretnine koje pripadaju tom gradu zajedno potrošile u istim periodima. Korisnik može da izabere vremenski period u kom želi da vidi pregled potrošnje (npr. poslednji 1h, poslednjih 6h, poslednih 12h, poslednja 24h, poslednjih nedelju dana, mesec dana, 3 meseca, godinu dana ili da izabere datume od-do pri čemu razlika između ta dva datuma ne sme da bude veća od godinu dana). U slučaju da je izabran period od poslednjih sat vremena, omogućiti prikaz u realnom vremenu.

Dobavljanje podataka radi prikaza akcija u realnom vremenu ostvariti pomoću WebSocket protokola.

4.12. Administracija službenika

Omogućiti administratoru da administrira službenike. Administrator može da registruje nove službenike (pri čemu upload-uje profilnu sliku službenika). Predefinisana lozinka službenika je njegov JMBG. Administrator može da vidi spisak svih službenika. Potrebno je implementirati efikasan mehanizam pretrage službenika u sistemu. Potrebno je istražiti kako popularne aplikacije sa velikim brojem korisnika (poput društvenih mreža) ovo rade i po uzoru na njih implementirati najefikasnije rešenje. Klikom na izabranog službenika može da vidi njegov profil. Administrator može da suspenduje službenika čime ovaj gubi pravo da se prijavi na sistem, a svi termini koji su zakazani bivaju otkazani.

4.13. Upravljanje rasporedom

Službenicima je radno vreme od ponedeljka do petka od 8h do 16h, sa pauzom od 12h do 12:30h. Službenici rade sa strankama u terminima od po pola sata (od 8h do 8:30h, od 8:30h do 9h itd.). Svaki termin je slobodan dok se ne zakaže. Službenik može da zakaže sastanak privatni sastanak koji može da zauzme jedan ili više vezanih slobodnih termina. Potrebno je implementirati efikasan mehanizam zakazivanja sastanaka te bi trebalo istražiti kako popularne aplikacije sa kalendarima i sastancima ovo rade i po uzoru na njih implementirati najefikasnije rešenje. Službenik ne može istovremeno imati dva sastanka u jednom terminu niti se dva sastanka mogu preklapati. Prilikom implementacije potrebno je adekvatno rešiti konfliktnu situaciju kada više korisnika aplikacije istovremeno pokuša da rezerviše isti termin.

4.14. Zakazivanje termina kod službenika

Omogućiti običnom korisniku da zakaže sastanak sa službenikom. Korisnik izborom službenika vidi njegov raspored slobodnih termina. Omogućiti korisniku da zakaže sastanak u tačno jednom slobodnom terminu (od 30 minuta) kod službenika. Potrebno je implementirati efikasan mehanizam zakazivanja sastanaka te bi trebalo istražiti kako popularne aplikacije sa kalendarima i sastancima ovo rade i po uzoru na njih implementirati najefikasnije rešenje. Službenik ne može istovremeno imati dva sastanka u jednom terminu. Prilikom implementacije potrebno je adekvatno rešiti konfliktnu situaciju kada više korisnika aplikacije istovremeno pokuša da rezerviše isti termin.

4.15. Definisanje cenovnika

Administrator ima mogućnost da definiše novi cenovnik. Prilikom definisanja cenovnika, navode se:

- cena jednog kWh za sve 3 zone
- cena obračunske snage
- vrednost pdv-a.

Ostale stavke koje su deo obračuna računa za električnu energiju koje inače koristimo ćemo zanemariti za potrebe projekta radi pojednostavljenja obračuna. Kada se kreira novi cenovnik, on ne važi odmah već njegovo važenje počinje tek od sledećeg meseca (npr. ako se kreira novi cenovnik 20. juna, on sa važenjem kreće tek od 1. jula i nove cene će biti vidljive na julskim računima).

4.16. Izdavanje računa

Administrator ima mogućnost da pokrene izdavanje računa za mesec za koji računi još uvek nisu izdati. Kada izabere mesec za koji se izdaju računi, potrebno je generisati račune kao PDF dokumente čiji će izgled dizajniran biti po uzoru na račune za električnu energiju koji svakodnevno koristimo. U sklopu računa potrebno je ugraditi QR kod čijim skeniranjem se otvara stranica za plaćanje računa. Račune u PDF formatu je potrebno poslati svim vlasnicima nekretnina putem email-a.
Prilikom formiranja cene računa, potrebno je voditi računa o tarifama. Potrošena električna energija se izražava u kWh. Ukupan broj utrošenih kWh je raspoređen po zonama (zelena - do 350 kWh , plava – od 351 do 1600 kWh i crvena od 1601 kWh). Ako je domaćinstvo npr. potrošilo 1830 kWh, prvih 350 kWh će biti obračunato po zelenoj zoni, narednih 1250 Kwh (1600 kWh - 350 kWh) će biti obračunato po plavoj zoni i preostalih 230 kWh (1830 kWh - 1600 kWh) će biti obračunato po crvenoj zoni. Na izračunati obračun potrošnje energiju po tarifama, potrebno je dodati rezultat množenja obračunske snage (uzeti kao konstantu 7 kW/kWh) i cene obračunske snage. Na konačan rezultat uračunati pdv. Za primer obračuna potrošnje od 1830 kWh, konačni obračun bi izgledao:

obracun_bez_pdv = 7 * cena_obračunske_naknade + (350 * cena_kwh_zelene_zone + 1250 * cena_kwh_plave zone_zone + 230 * cena_kwh_plave zone_zone)

konačni_obračun = obracun_bez_pdv + obracun_bez_pdv * vrednost_pdv-a

4.17. Plaćanje računa

Vlasnici domaćinstava mogu da vide sve spisak svih računa i da ih sortiraju i filtriraju po većem broju parametara. Za svaki račun moguće je videti njegov status (da li je izvršeno plaćanje ili nije) kao i koji cenovnik je primenjivan prilikom njegovog izdavanja. U slučaju da račun nije plaćen, moguće ga je platiti. Dizajnirati stranicu za plaćanje po uzoru na popularne e-prodavnice. Korisnik unosi podatke o kartici i vrši plaćanje. Pretpostaviti da je plaćanje uvek uspešno, tj. da na računu uvek ima dovoljno sredstava. Nakon plaćanja, korisnik na email dobija PDF dokument sa popunjenom uplatnicom po uzoru na uplatnice koje svakodnevno koristimo.

4.18. Ponašanje simulatora prilikom prekida komunikacije

Unaprediti simulator ispunjenjem zahteva 4 i 5 iz poglavlja 3. Simulator treba da bude otporan na prekid komunikacije sa platformom. To podrazumeva nastavak simuliranja potrošnje domaćinstva čak i kad slanje očitanih vrednosti nije moguće. Potrebno je da simulator čuva

očitane vrednosti lokalno u periodu od 90 dana te da ih nakon isteka tog perioda briše. Potrebno je istražiti na koje načine uređaji sa ograničenim hardverskim mogućnostima mogu čuvati podatke. Nakon što simulator uspostavi konekciju, šalje platformi sva merenja koja ranije nisu uspešno dostavljena.

5. Nefunkcionalni zahtevi

5.1. Serverske platforme

Za realizaciju projekta može se izabrati serverska platforma po želji. Neke od platformi mogu biti:

- Java + Spring Boot (koristi se na vežbama)
- Java + Play framework
- Java + Spark framework
- Python + Django
- Ruby on Rails
- .NET
- …

5.2. Klijentske tehnologije

Za realizaciju klijentske aplikacije može se izabrati jedan od sledećih frontend tehnologija:

- Angular
- Vue
- React

5.3. Slanje email-a

Za slanje email-a nije obezbeđen poseban servis. Možete koristiti sopstveni email nalog. Email-ovi koje šalje vaša platforma treba da imaju jasno definisan template koji izgledom podseća na izgled vaše platforme.

5.4. API dizajn

API glavne aplikacije treba da bude dizajniran i dokumentovan u skladu sa OpenAPI specifikacijom.

5.5. Perzistencija podataka

Sve podatke koji se kreiraju u toku rada platforme potrebno je trajno perzistirati. Podatke koje generiše simulator (poput očitvanja potrošnje električne energije i promene stanja/statusa) potrebno je čuvati u time series bazi podataka po izboru (npr. InfluxDb). Ostale podatke (poput

informacija o korisnicima, domaćinstvima itd.) potrebno je čuvati u relacionoj bazi podataka po izboru (npr. MySQL, MariaDB, PostgreSQL itd.).

5.6. Serviranje statičkog sadržaja

Iako mnoge serverske platforme imaju mogućnost da serviraju statički sadržaj i sadržaj koji korisnici upload-uju (poput slika), takav način serviranja sadržaja nije preporučljiv. Umesto toga, tehnologije poput nginx-a mogu da se koriste kao reverse proxy i da bi servirali statički sadržaj. Česta upotreba npr. nginx-a je serviranje statičkog sadržaja koji čini korisnički interfejs aplikacije (html, css, js itd.). U tom slučaju, ako je klijentska aplikacija izrađena pomoću nekog radnog okvira i/ili skupa biblioteka (npr. Angular, React, Vue), ona se pretvori u skup statičkog sadržaja (tj. izgradi za produkciju - *production build*) koji nginx servira.

5.7. Reverse proxy

Kada imate softversko rešenje koje ima više komponenti ili resursa koji su dostupni na različitim rutama, a pri čemu je potrebno definisati način pristupa, dobra praksa je koristiti reverse proxy (npr. nginx, Apache Traffic Server, HAProxy itd.). Reverse proxy može da reši i neke druge probleme sa kojima se možete susresti prilikom izrade projektnog zadatka.

5.8. Čuvanje korisničkih datoteka

Potrebno je obezbediti sigurno čuvanje datoteka koje korisnici upload-uju kao što su slike i dokumenti. Kako ove datoteke mogu biti izuzetno velike, ne bi trebalo da se njihovim serviranjem bavi serverska aplikacija. Stoga, potrebno je izabrati tehnologiju čiji će zadatak biti da omogući prikaz i preuzimanje ovih datoteka. Takođe, potrebno je obezbediti proveru prava pristupa ovim datotekama, tj. da ne može svako da preuzme svaku od datoteka već da se reguliše pravo pristupa u skladu sa specifikacijom funkcionalnih zahteva.

5.9. Prikaz grafikona

Za prikaz grafikona na korisničkom interfejsu mogu se koristiti third party biblioteke.

5.10. Testiranje performansi sistema

Potrebno je izvršiti testiranje opterećenja sistema (*load testing*) za česte scenarije korišćenja. Svaki student treba da izabere 10 različitih scenarija korišćenja koje je implementirao u sklopu funkcionalnosti koje mu pripadaju (jedan scenario bi npr. bio da se korisnik prijavi u sistem i podnese zahtev za registraciju nove nekretnine). Za svaki od scenarija treba da se provere performanse sistema i ispita njihova promena sa povećanjem količine podataka sa kojim aplikacija rukuje i povećanjem broja korisnika koji istovremeno pokušavaju da izvrše isti scenario (npr. veliki broj korisnika istovremeno pokuša da se prijavi na sistem i kreira zahtev za registraciju nove nekretnine). Potrebno je istražiti koji alati se mogu koristiti za ovaj vid testiranja (JMeter, Locust, Gatling itd.) i koristiti jedan i/ili više njih kako bi se izvršilo potpuno testiranje

izabranih scenarija korišćenja. Testirati ponašanje aplikacija počev od slučaja kada je istovremeno koriti mali broj korisnika (10-ak) do slučaja kada je koristi veliki broj korisnika (nekoliko hiljada).

Potrebno je da svaki student sastavi izveštaj (u slobodnoj formi) kako je izvršio testiranje svog dela sistema. Dokument može da sadrži tekst, slike i grafikone koji dokumentuju proces testiranja. U slučaju da prilikom testiranja uočite da određeni delovi sistema imaju loše performanse, potrebno je uložiti napore u unapređenje tih delova sistema i to bi trebalo takođe dokumentovati u izveštaju. Konačnu verziju izveštaja konvertovati u **PDF dokument** i okačiti na udaljeni git repozotorijum.

Napomena: Savetuje se da platformu (sa svim njenim komponentama) pokrenete na jednom računaru, a da na drugom računaru (ili više njih) pokrenete alate za testiranje performansi. Računari bi trebalo da se nalaze u istoj mreži kako bi mogli međusobno komunicirati. Na ovaj način ćete platformi dozvoliti da koristi što više resursa koje ima računar na kom je platforma pokrenuta, dok će drugi računar (ili više njih) služiti za pokretanje što više aplikacija koje će komunicirati sa platformom.

5.11. Kontrola verzija i kontinualni rad

Neophodno je koristiti **Git** za kontrolu verzija i repozitorijum mora biti na **GitLab**-u. Asistentskom nalogu Vanja-Mijatov (vanja.mijatov@uns.ac.rs) dati pristup repozitorijumu i ulogu **Maintainer**-a. Takođe, potrebno je u **README.md** napisati koje tehnologije su korišćene, ko je učestvovao u izradi rešenja i navesti tačno uputstvo za pokretanje projekta i priložiti skriptu za popunjavanje baze testnim podacima.

Napomena: Na GitLab-u će se kroz commit-ove pratiti kontinualni rad. Projekti koji budu kačeni na GitLab u svega nekoliko komitova neće biti podložni ocenjivanju. Nije neophodno raditi na projektu prema nekoj strogoj metodologiji (npr. Scrum) niti voditi tablu sa zadacima (kanban board ili nešto slično), ali je neophodno da se nove verzije koda šalju redovno u toku semestra na udaljeni repozitorijum.

6. Ocenjivanje

6.1. Bodovanje - opšte

Projekat ukupno nosi **50 bodova**. Minimalan broj bodova koji je potreban za polaganje projekta je **26**. Projekat se ocenjuje u 3 etape:

1. Prva kontrolna tačka - nosi 5 bodova i izvodi se u terminu vežbi sredinom novembra
2. Druga kontrolna tačka - nosi 10 bodova i izvodi se u terminu vežbi sredinom decembra
3. Finalna odbrana - nosi 35 bodova i izvodi se u 3 roka:
    a. Januarsko-februarski rok

b. Junski rok
c. Septembarski rok

U sklopu odbrane projekta u istom terminu održaće se i usmeni ispit koji takođe nosi **50 bodova**.

6.2. Podela zadataka u timu

Student 1

4.1, 4,2, 4.7, 4.8, 4.10, 4.14

Student 2

4.3, 4,4, 4.12, 4.13, 4.15, 4.16

Student 3

4.5, 4,6, 4.9, 4.11, 4.17, 4.18

6.3. Prva kontrolna tačka

Prva kontrolna tačka će se održati u terminu vežbi u nedelji od 18.11. do 22.11. i vrednuje se sa 5 bodova. Za ovu kontrolnu tačku potrebno je implementirati sledeće funkcionalnosti:

- Student 1 - 4.1. i 4.2.
- Student 2 - 4.3. i 4.4.
- Student 3 - 4.5. i 4.6.

6.4. Druga kontrolna tačka

Druga kontrolna tačka će se održati u terminu vežbi u nedelji od 16.12. do 20.12. i vrednuje se sa 10 bodova. Za ovu kontrolnu tačku potrebno je implementirati sledeće funkcionalnosti:

- Student 1 - 4.7. i 4.8.
- Student 2 - 4.12. i 4.13.
- Student 3 - 4.11. i 4.18.

6.5. Finalna odbrana

Implementacija svih funkcionalnosti je preduslov za uspešno polaganje projekta. Studenti koji ne implementiraju sve funkcionalnosti ne mogu uspešno položiti projekat. Da biste osvojili sve bodove za implementaciju neke funkcionalnosti nije bitno samo da li radi funkcionalnost, već i kako radi. Kako aplikacija treba da podrži rad sa velikim brojem korisnika, nekretnina i simulatora, potrebno je da za svaku funkcionalnost nađete optimalno rešenje. To bi značilo da, ako je očekivan sporiji rad sistema ili njegovih delova zbog dobavljanja ili obrade veće količine podataka, koristeći neke od metoda obrade i keširanja podataka ubrzate rad sistema. Keširanje podataka se često radi na strani serverske aplikacije (obično uz korišćenje baze za keširanje

poput redis-a). Zatim, u slučaju prikazivanja veće količine podataka u vidu tabela ili grafika, potrebno je primeniti dobru praksu koja podrazumeva paginaciju, downsampling i slične pristupe. U tu svrhu dozvoljeno je koristiti gotova rešenja (aplikacije, biblioteke, dodatne baze podataka).

Drugi aspekt koji će se ocenjivati je priprema aplikacije za pokretanje u produkciji. Potrebno je kreirati statičke datoteke na osnovu klijentske aplikacije i servirati ih korišćenjem neke tehnologije za serviranje statičkog sadržaja koja to efikasno radi. Zatim, potrebno je podesiti reverse proxy po izboru. Dalje, potrebno je rešiti problem čuvanja i dobavljanja datoteka koje upload-uju korisnici. Takođe, neophodno je pojednostaviti konfiguraciju aplikacije te identifikovati minimalan set koraka i podešavanja koje bi bilo potrebno uraditi kako bi se rešenje pokrenulo u oblaku.

Korisničko iskustvo (UX) je takođe značajna stavka prilikom ocenjivanja projekta. Bitno je da klijentska aplikacija se intuitivno koristi i da akcije koje zahtevaju dobavljanje ili obradu veće količine podataka ne učine pretraživač da se zamrzne ili ugasi. Email-ovi koje šalje vaša platforma treba da imaju jasno definisan template koji izgledom podseća na izgled vaše platforme.

Na finalnoj odbrani će biti provereno ispunjavanje svih funkcionalnih i nefunkcionalnih zahteva sa akcentom na zahteve iz sekcija od 5.3. do 5.11. pri čemu je potrebno posebnu pažnju posvetiti zahtevu 5.10. Testiranje performansi sistema je izuzetan bitan aspekt u sklopu dizajniranja softvera koje će koristiti veliki broj korisnika te ovaj zahtev koji nosi najveći deo ocene.

7. Akademska čestitost

Akademska čestitost podrazumeva samostalno pisanje radova (teksta, programskog koda i slično) uz striktno poštovanje tuđih autorskih prava. Ovaj pojam i obaveza su sastavni deo Zakona o visokom obrazovanju:

http://www.parlament.gov.rs/upload/archive/files/lat/pdf/predlozi_zakona/3048-14Lat.pdf

Više o ovoj temi možete pronaći, na primer, na sledećim linkovima:

- https://en.wikipedia.org/wiki/Academic_honor_code
- https://www.akademsko-pisanje.uniri.hr/akademska-cestitost/

U okviru ovog predmeta to podrazumeva da studenti samostalno pišu sopstveni rad. Pomoć drugih studenata u obliku direktno preuzetih delova teksta ili programa nije dozvoljena. **Kako se ovaj projekat radi u timu, odgovornost za nepridržavanje**

principa akademske čestitosti snose svi članovi tima.

Sistemi za otkrivanje plagijarizma su vremenom postali prilično efektivni. Više informacija se može naći ovde:

https://theory.stanford.edu/~aiken/moss/

ili recimo ovde:

https://github.com/genchang1234/How-to-cheat-in-computer-science-101

Kako će se na ovom predmetu prilikom analize projekata pristupati originalnim udaljenim repozitorijumima, treba imati u vidu da se na njima vidi kompletna istorija izmena u toku rada na projektu. Okolnosti u kojima se na repozitorijumu pojavljuje nešto neregularno odmah dovode u sumnju dati studentski tim.

Zajedničko učenje studenata ili rad na projektu je lepa praksa, ali to ne podrazumeva da se do rezultata koji se ocenjuje dolazi zajednički.