Programsko inženjerstvo

Ak. god. 2022./2023.

Company Database

Dokumentacija, Rev. *1*

Grupa: *EkipaZaOcevid*

Voditelj: *Petar Hajduk*

Datum predaje: *18. 11. 2022.*

Nastavnik: *Manuela Lukić*

Sadržaj

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Jakov Jakovac	22.10.2022.
0.2	Napisani funkcionalni zahtjevi.	Marko Čengić	09.11.2022.
0.3	Napisan opis projektnog zadatka.	Lovro Čunović	11.11.2022.
0.4	Napravljen opis baze podataka.	Nikola Capan	12.11.2022.
0.5.1	Napisani obrasci uporabe.	Marko Čengić	12.11.2022.
0.5.2	Dorada obrazaca uporabe.	Jakov Jakovac	14.11.2022.
0.6	Napisan opis arhitekture sustava.	Lovro Čunović	16.11.2022.
0.7.1	Napravljeni dijagrami obrazaca uporabe.	Marko Čengić	17.11.2022.
0.7.2	Napravljeni sekvencijski dijagrami.	Marko Čengić	17.11.2022.
0.8	Napravljeni dijagrami razreda.	Nikola Capan	17.11.2022
0.9	Priprema dokumenta za prvu potpunu verziju.	Jakov Jakovac	17.11.2022

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	**Opis promjene/dodatka**	**Autori**	**Datum**
0.10.1	Dorada arhitekture sustava temeljem povratnih informacija.	Lovro Čunović	18.11.2022
0.10.2	Promjena aktora funkcionalnih zahtjeva temeljem povratnih informacija.	Marko Čengić	18.11.2022
0.10.3	Dorada tablica baze podataka temeljem povratnih informacija.	Jakov Jakovac	18.11.2022
0.10.4	Dorada dijagrama baze podataka temeljem povratnih informacija.	Nikola Capan	18.11.2022
0.11	Dorada opisa projektnog zadatka.	Jakov Jakovac	18.11.2022
0.12	Ispravci grešaka i posljednje pripreme dokumenta za prvu potpunu verziju.	Jakov Jakovac	18.11.2022
1.0	Verzija samo s bitnim dijelovima za 1. ciklus.	Jakov Jakovac	18.11.2022.

2. Opis projektnog zadatka

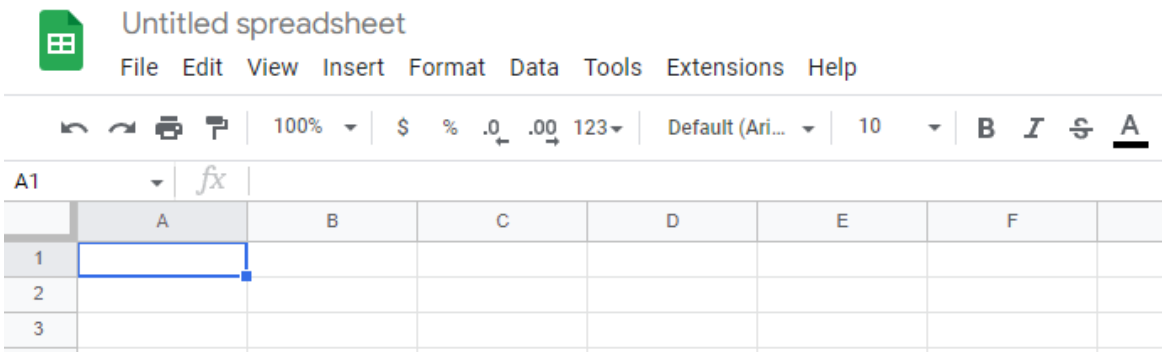
2.1 Opis problematike

Porastom broja neprofitnih organizacija iz godine u godinu, raste i potreba za kvalitetnim aplikacijama koje bi takvim organizacijama omogućile nesmetan rad i funkcioniranje. Kako takve organizacije nisu u mogućnosti samostalno postići zadovoljavajuću razinu prihoda, to im postaje najveći problem te primorane su na suradnju s kompanijama sponzorima.

Odaziv sponzora je često vrlo malen te je stoga potrebno kontaktirati velik broj kompanija. Praćenje statusa suradnji (uspješne / neuspješne) i osoba zaduženih za kontaktiranje sponzora vrlo brzo postaje kompleksno i neefikasno, a također dolazi i do mogućnosti curenja informacija. Raste potreba za aplikacijom koja će omogućiti evidenciju navedenih stavki te korisnicima dodijeliti razinu ovlasti kako bi umanjili mogućnost curenja informacija.

2.2 Postojeća rješenja

Najraširenije rješenje koje danas koriste gotovi svi jesu tablice. Bilo to u MS Excelu za jednokorisnički ili Google sheets 2.1 za višekorisnički rad, tablice su jedan od najboljih načina prikaza velikog broja podataka. Nažalost, ova rješenja imaju gore navedene probleme te stoga nisu pogodna za upotrebu na razini veće organizacije.



Slika 2.1: Prikaz alata Google sheets

2.3 Opseg projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije *"Company Database"* koja će služiti za evidenciju statusa suradnje kompanija na projektima i jednostavan uvid od strane ovlaštenih korisnika.

Korisnici će se u web aplikaciju prijavljivati koristeći Google račun koji mora biti registriran od strane drugih korisnika putem web aplikacije. Svaki je korisnik definiran sljedećim parametrima:

- ime
- prezime
- korisničko ime (nadimak)
- opis
- email za login u aplikaciju
- email za obavijesti
- najviša razina ovlasti koju posjeduje (ako ima više razina ovlasti)

Razlikovat ćemo 5 razina ovlasti, a to su redom od najviše prema najnižoj:

- Administrator
- Moderator
- Fundraising (FR) responsible
- Fundraising (FR) team member
- Observer

Svaki korisnik određene razine ovlasti ima sve mogućnosti svoje i svih nižih razina ovlasti sustava. Također ima mogućnost (kroz određene akcije) dodjeljivanja

i oduzimanja razina ovlasti korisnicima nižih razina od svoje.

Iznimke su administrator koji može dodijeliti i oduzeti i ostalim administratorima sustava te FR team member koji ne može dodijeliti niti oduzeti ovlasti jer se observer automatski dodjeljuje svim korisnicima koja nemaju višu razinu ovlasti.

Projekt je obilježen nazivom, kategorijom, tipom (interni ili eksterni), datumom početka i kraja, FR responsible-om zaduženim za navedeni projekt, popisom FR team member-a, FR ciljem (željeni prihod od projekta), prvim i drugom „pingom“ (timestamp) te popisom kompanija s kojima se surađuje na tom projektu.

Kompanija je obilježena nazivom, područjem kojim se bavi (npr. IT), ABC kategorizacijom, mjesecom planiranja budžeta, državom, poštanskim brojem, gradom, adresom, linkom na web stranicu, kratkim opisom, boolean varijablom *„kontaktirati“* (koja označava treba li tu tvrtku u budućnosti kontaktirati) te popisom zaposlenika.

Popis kompanija moguće je uzlazno i silazno sortirati prema nazivu, području, mjesecu planiranja budžeta i linku na web stranicu te pretraživati po nazivu.

Zaposlenik kao entitet pripada kompaniji, a definiran je imenom, prezimenom, email adresom, brojem mobitela, ulogom u kompaniji (npr. CEO) te kratkim opisom.

Suradnja predstavlja poveznicu između projekta i kompanije, a obilježena je odgovornom osobom (koja je za nju zadužena), kontaktiranom osobom u kompaniji, kategorijom (financijska, materijalna ili akademska suradnja), vlastitim statusom (kontaktirano, ping, dopis, sastanak, uspješno ili neuspješno), komentarom (tj. sažetkom suradnje) te vrijednošću suradnje (koje se sumiraju i popunjavaju FR cilj projekta).

Aplikacija, sa svim navedenim funkcionalnostima i specifičnostima, bit će izvedena kao web aplikacija kojoj korisnici pristupaju pomoću Google autentifikatora. Bit će jednostavna za korištenje zahvaljujući preglednom i intuitivnom sučelju. Sustav će podržavati rad više korisnika u stvarnom vremenu, frontend će biti ostvaren u React-u, a backend će koristiti relacijsku bazu podataka i Spring boot.

2.4 Mogućnost prilagodbe rješenja

Kako bi što više ljudi moglo znati za projekt, organizatori nerijetko kontaktiraju razne medije kako zainteresirani ne bi propustili priliku. Slična aplikacija bi se

također mogla koristiti za komunikaciju s medijima. Entitet Tvrtka bi bilo bi potrebno izmijeniti u Medij zajedno s odgovarajućim atributima, dok bi u entitetu Suradnja bilo potrebno izmijeniti samo atribute.

Ovakva bi aplikacija bila od koristi ne samo neprofitnim organizacijama, već i svima koji organiziraju nešto što treba publicitet.

2.5 Mogućnost nadogradnje rješenja

Aplikacija bi se dodatno mogla nadograditi proširenjem komentara suradnje, primarno u svrhu prenošenja znanja od iskusniji osoba. Te bi osobe kroz komentar na suradnju mogli dati svoj input vezano uz neku temu. Tada bi i korisnik koji je odgovoran za suradnju dobio te obavijesti na naslovnicu.

Nadalje, aplikacija bi mogla, osim tabličnog, podržavati više načina prikaza podataka o korisnicima, projektima, tvrtkama i suradnjama. Možda bi ih se moglo grupirati temeljem nekih postojećih ili novih atributa te onda to iskoristiti.

Također bi bilo korisno uvesti dodatne mogućnosti filtriranja svih tablica i drugih načina prikaza podataka.

Ako se projekti neke organizacije ponavljaju iz godine u godinu, bilo bi korisno uvesti neki način kopiranja podataka iz tablice prošlogodišnjeg projekta. Korisno bi bilo dodati i mogućnost izvoza i uvoz podataka iz i u tablice u JSON i CSV formatima.

Kako bi se još više smanjila mogućnost curenja informacija, mogao bi se uvesti tzv. soft lock korisnika koji bi očuvao podatke o korisniku, ali mu onemogućio da se ulogira (npr. ako korisnik više nije dio organizacije).

Aplikaciju bi također bilo moguće spojiti s nekakvim internim sustavom za upravljanjem članovima organizacije.

Što se tiče UX-a, tu uvijek ima mjesta za napredak. Dodavanje animacija tokom prijelaza od jedne stranice na drugu ili važnije tokom dohvata podataka iz baze kako bi korisnik znao da se njegov zahtjev procesuira. Moglo bi se uvesti neki način prikaza uspješnosti odrade nekog zadatka (npr. dodavanje podatka u bazu).

Mogao poboljšati i sam UI da aplikacija izgleda profesionalnije i bude ugodnija za korištenje, ali i kako bi aplikaciju mogle koristiti osobe s posebnim potrebama.

3. Arhitektura i dizajn sustava

Arhitekturu možemo podijeliti na 3 podsustava:

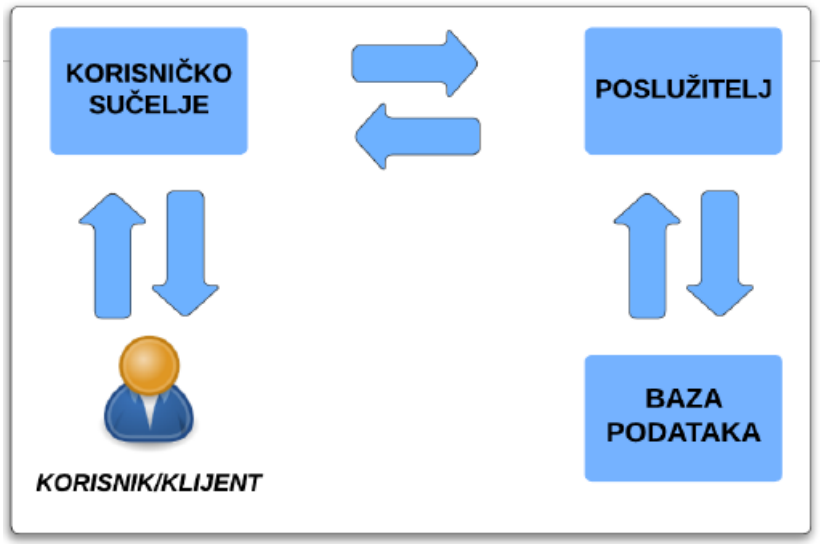
- Korisničko sučelje (frontend)
- Poslužitelj (backend)
- Baza podataka

Web preglednik (*Internetski preglednik, Web browser, Internet browser*) je program koji korisniku omogućuje pregled web-stranica i multimedijalnih sadržaja vezanih uz njih. Svaki internetski preglednik je interpreter (program koji u realnom vremenu izvršava izvorni kod napisan u nekom programskom jeziku, umjesto da ga, prije izvršavanja cijelog prevede u strojni jezik, što inače radi jezični prevoditelj). Dakle, stranica je pisana u kodu koji preglednik interpretira u čitljiv sadržaj. Korisnici putem web preglednika šalju zahtjeve poslužitelju.

Web poslužitelj je temelj svake Web aplikacije jer on služi za komunikaciju klijenta s aplikacijom. Takva vrsta komunikacije odvija se uz pomoć HTTP (*Hyper Text Transfer Protocol*) protokola koji predstavlja glavnu i najčešću metodu prijenosa informacija na Webu. Osnovna namjena ovog protokola je omogućavanje objavljivanja i prezentacije HTML dokumenata, tj. Web stranica. Poslužitelj nam dakle služi za prosljeđivanje zahtjeva Web aplikaciji te je zadužen za njezino pokretanje.

Od ostalih protokola koji služe za komunikaciju i prijenos informacija i podataka između klijenta i poslužitelja treba napomenuti: IP (*Internet Protocol*) koji služi za prijenos podataka u blokovima (paketi, datagrami), komunikacijski protokol ARP (*Address Resolution Protocol*) te ICMP (*Internet Control Message Protocol*) koji je ugrađen u svaki IP modul kako bi omogućio računalima slanje kontrolnih poruka o greškama. ICMP prijavljuje greške, ali nije zadužen za njihovo ispravljanje.

Klijent koristi korisničko sučelje, tj. Web aplikaciju, prikazanu u pregledniku, te preko nje šalje zahtjeve poslužitelju. Nakon obrade podataka, poslužitelj vraća odgovor aplikaciji koja ga potom prikazuje korisniku.



Slika 3.1: Prikaz arhitekture sustava

Programski i korisnički jezici koje smo koristili su: HTML, CSS i JavaScript za Frontend te Java za Backend. Radni okviri koje smo odabrali su React (Frontend) i Spring Boot (Backend). Od razvojnih okruženja koristili smo IntelliJ u okviru izrade Backenda te Visual Studio Code za izradu Frontenda.

U okviru Backenda korišten je REST (*Representational State Transfer*) API (*Application programming Interface*). REST je softverska arhitektura koja nameće uvjete o tome kako API treba raditi, a API definira pravila koja morate slijediti za komunikaciju s drugim softverskim sustavima.

REST API je sučelje koje dva računalna sustava koriste za sigurnu razmjenu informacija putem interneta. Većina Web aplikacija mora komunicirati s drugim internim aplikacijama i aplikacijama trećih strana kako bi izvršile razne zadatke, a REST API podržava ovu razmjenu informacija jer slijedi sigurne, pouzdane i učinkovite softverske komunikacijske standarde.

3.1 Baza podataka

Podaci u aplikaciji će se spremati u relacijsku bazu podataka koristeći Postgres kao DBMS. Baza će se sastojati od sljedećih entiteta:

- Korisnik
- Mjesto
- Tvrtka
- Zaposlenik
- Projekt
- ProjectFRTeamMembers
- Suradnja

3.1.1 Opis tablica

Korisnik		
IDKorisnik	INT	ID korisnika
Ime	VARCHAR(40)	Ime korisnika
Prezime	VARCHAR(40)	Prezime korisnika
Nadimak	VARCHAR(40)	Nadimak korisnika
LoginEmail	VARCHAR(60)	Email adresa pomoću kojeg se korisnik logira
NotificationEmail	VARCHAR(60)	Email adresa na koju korisnik prima obavijesti
MaxRazinaOvlasti	INT	Maksimalna razina ovlasti koju korisnik posjeduje (ENUM)
Opis	VARCHAR(480)	Opis korisnika

Mjesto		
PBr	INT	Poštanski broj mjesta
Naziv	VARCHAR(120)	Naziv mjesta

Tvrtka		
IDTvrtka	INT	Id tvrtke
Naziv	VARCHAR(60)	Naziv tvrtke
Podrucje	VARCHAR(40)	Područje kojim se tvrtka bavi
ABCKategorizacija	CHAR	Kategorija tvrtke (ENUM: A, B ili C)
MjesecPlaniranjaBudzeta	INT	Mjesec u kojem tvrtka planira budžet
Drzava	VARCHAR(60)	Država u kojoj tvrtka posluje
PBr	INT	Poštanski broj mjesta tvrtke
Adresa	VARCHAR(120)	Adresa tvrtke
WebURL	VARCHAR(60)	Adresa web stranice tvrtke
Kontaktirati	BOOLEAN	Treba li ubuduće tu tvrtku kontaktirati

Zaposlenik		
IDZaposlenik	INT	Id zaposlenika
IDTvrtka	INT	Id tvrtke za koju zaposlenik radi
Ime	VARCHAR(40)	Ime zaposlenika
Prezime	VARCHAR(40)	Prezime zaposlenika
Email	VARCHAR(60)	Email adresa zaposlenika
Tel	VARCHAR(20)	Broj telefona zaposlenika

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

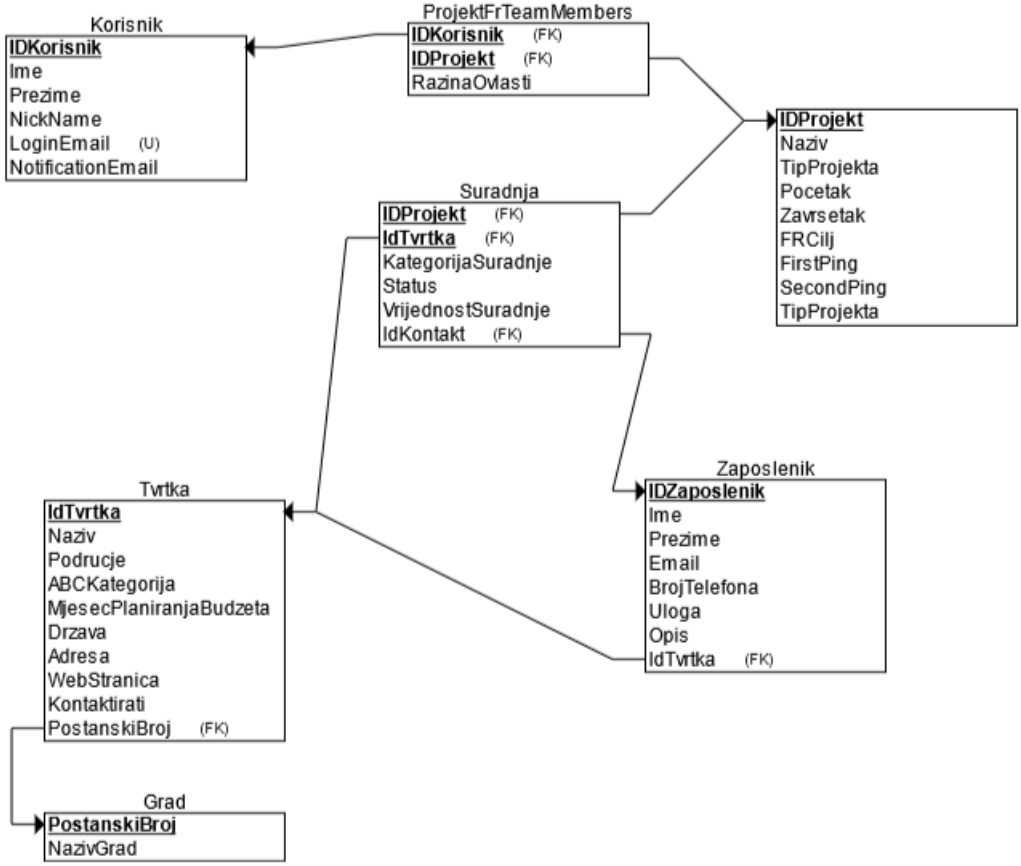
Zaposlenik		
Uloga	VARCHAR(40)	Pozicija zaposlenika u tvrtki (npr. PR)
Opis	VARCHAR(60)	Opis zaposlenika (npr. glavni kontakt)

Projekt		
IDProjekt	INT	ID projekta
Naziv	VARCHAR(40)	Naziv projekta
Kategorija	INT	Kategorija u koju spada projekt
Tip	INT	Tip projekta (ENUM: eksterni ili interni)
ProjektPocetak	TIMESTAMP	Datum početka projekta
ProjektZavrsetak	TIMESTAMP	Datum završetka projekta
IDFRResp	INT	ID korisnika odgovornog za odnose s tvrtkama projekta
FRCilj	INT	Količina novca koja se želi prikupiti za projekt
PrviFRPing	TIMESTAMP	Datum prvog ”ping”-a
DrugiFRPing	TIMESTAMP	Datum drugog ”ping”-a

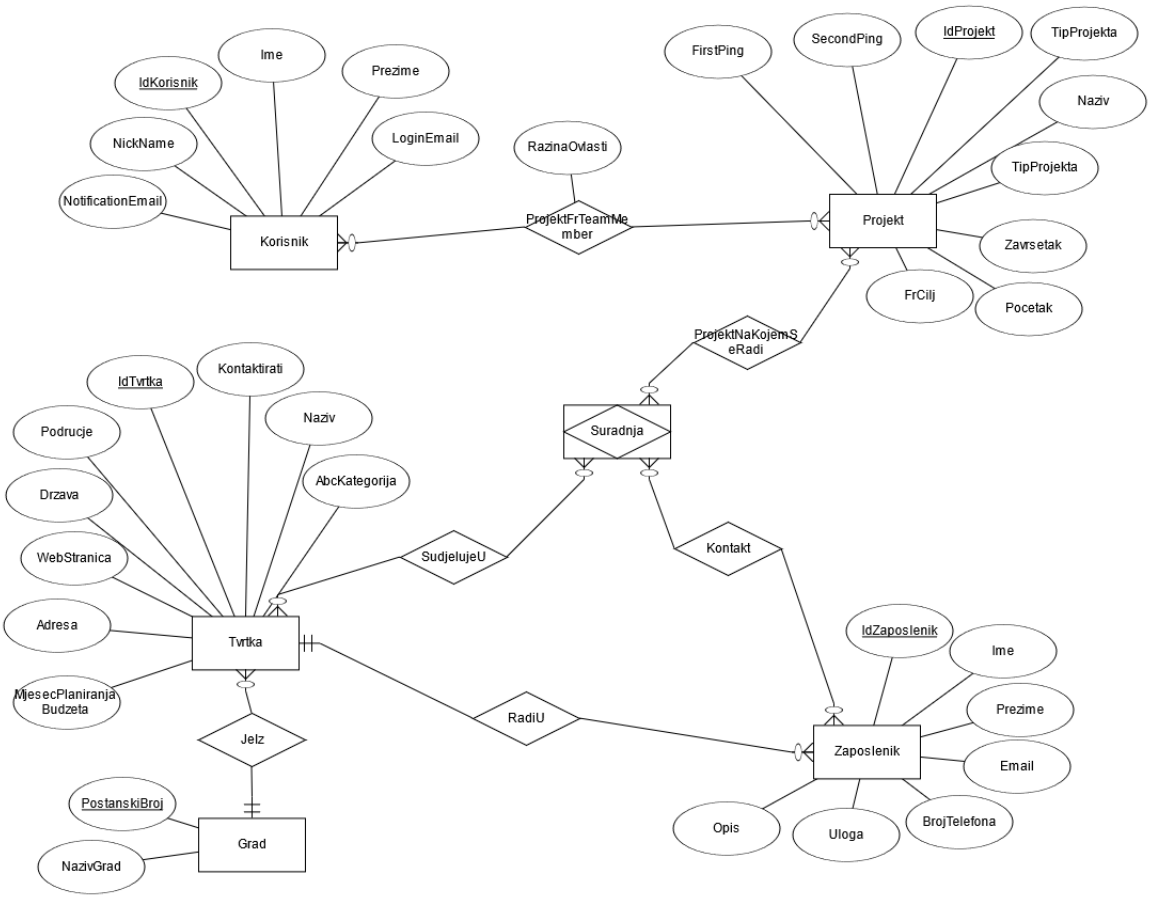
ProjectFRMembers		
IDKorisnik	INT	Id korisnika koji je FR team member projekta
IDProjekt	INT	Id projekta

Suradnja		
IDProjekt	INT	Id projekta
IDTvrtka	INT	Id tvrtke
IDOdgovoran	INT	Id korisnika odgovornog za suradnju s tvrtkom
IDKontakt	INT	Id kontakt osobe u tvrtci
Kategorija	VARCHAR(20)	Kategorija suradnje (ENUM: financijska, materijalna ili akademska)
Status	VARCHAR(20)	Kontaktirano, ping, dopis, sastanak, uspješno ili neuspješno
Komentar	VARCHAR(480)	Komentar na suradnju
Vrijednost	INT	Ostvarena novčana vrijednost suradnje

3.1.2 Dijagram baze podataka



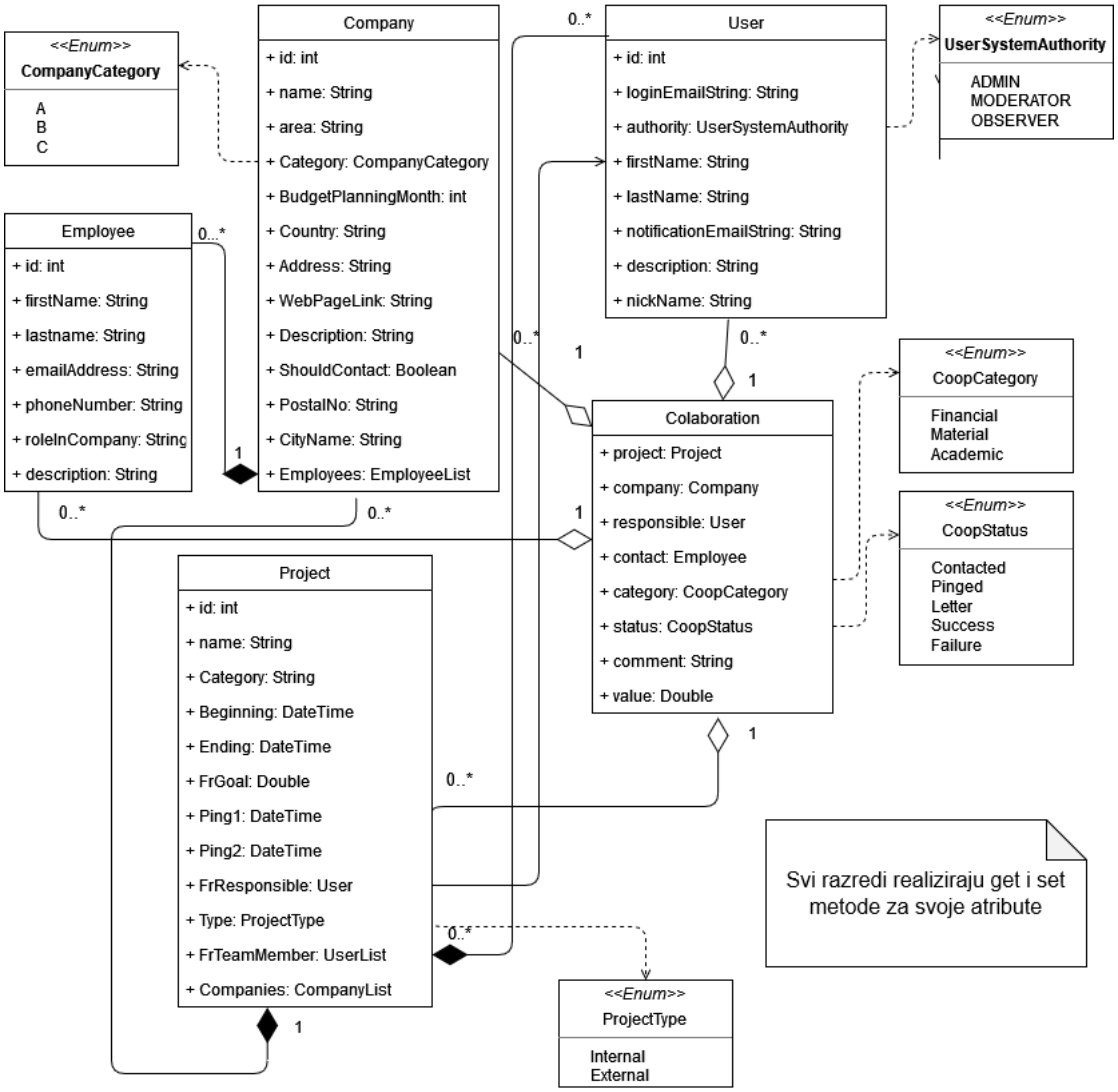
Slika 3.2: Dijagram baze podataka



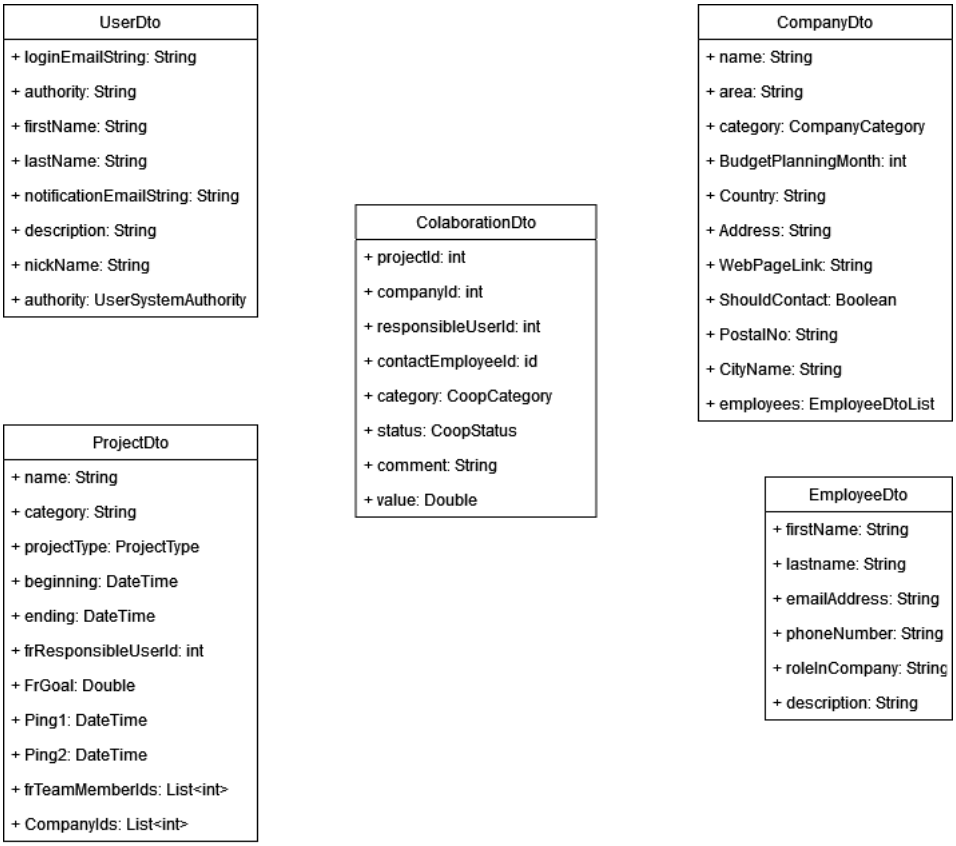
Slika 3.3: Entity-relationship dijagram

3.2 Dijagram razreda

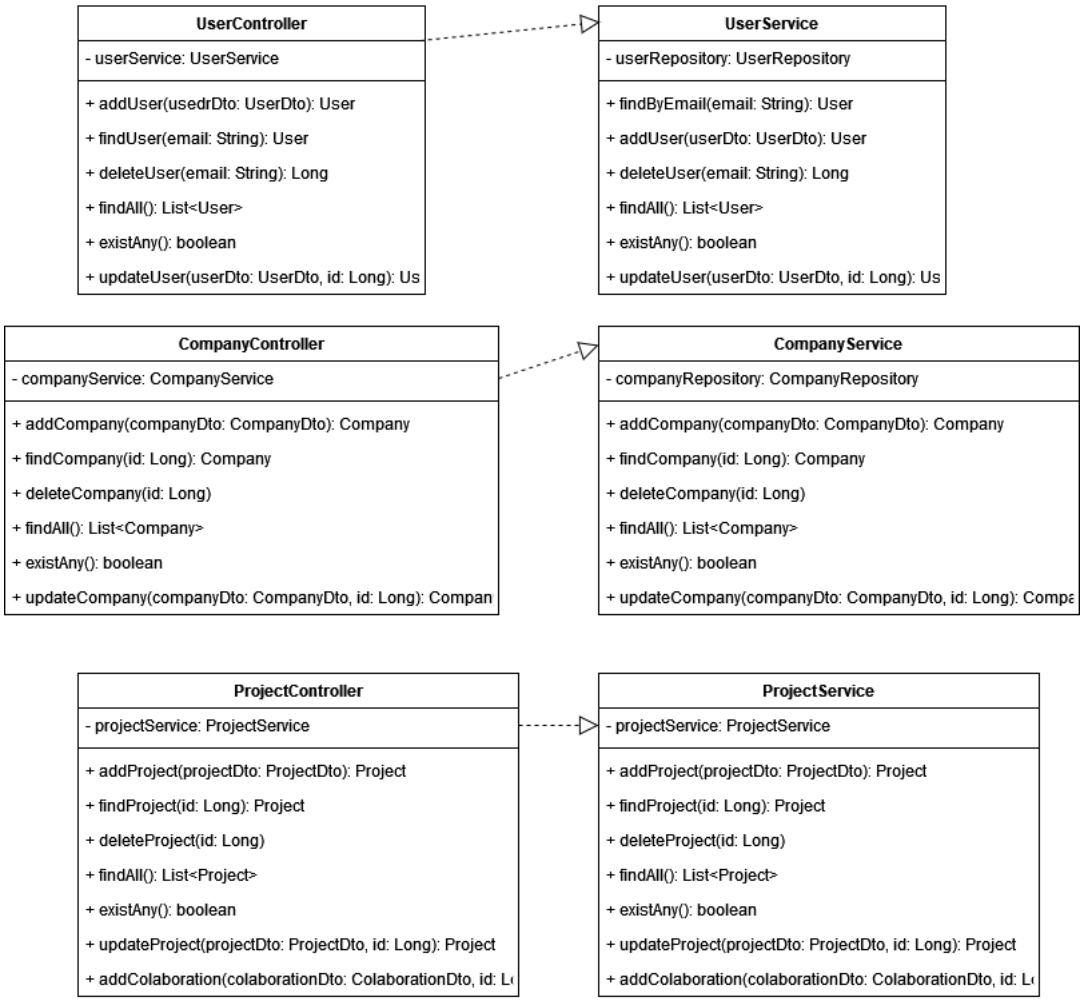
Na sljedećim slikama prikazani su UML dijagrami razreda koji se koriste u aplikaciji. Podijeljeni su u 3 dijela. Prvi dijagram prikazuje klase kojima smo modelirali entitete aplikacije, drugi prikazuje DTO-ove *(Data transfer object)*, a treći prikazuje kontrolere. Može se uočiti da postoje veze između klasa iz različitih dijagrama (npr. UserDto sadrži UserSystemAuthority), ali su one izostavljene radi preglednosti.



Slika 3.4: UML dijagram razreda



Slika 3.5: UML dijagram Data Transfer Object-a



Slika 3.6: UML dijagram kontrolera i servisa

4. Implementacija i korisničko sučelje

4.1 Korištene tehnologije i alati

dio 2. revizije

Detaljno navesti sve tehnologije i alate koji su primijenjeni pri izradi dokumentacije i aplikacije. Ukratko ih opisati, te navesti njihovo značenje i mjesto primjene. Za svaki navedeni alat i tehnologiju je potrebno ***navesti internet poveznicu*** *gdje se mogu preuzeti ili više saznati o njima.*

4.2 Ispitivanje programskog rješenja

4.2.1 Ispitivanje komponenti

Svi endpointovi na backend-u su ručno testirani, a ispitivanje jedinica (engl. unit testing) proveli smo nad osnovnim funkcionalnostima servisnog sloja koristeći biblioteku junit. Prilikom testiranja pojedinog razreda, servise i repozitorije čije metode taj razred poziva smo *mock-ali* koristeći biblioteku mockito, pa smo samo u svakoj test metodi zadali što će pojedina pozvana metoda vraćati.

1. UserServiceTests

U UserServiceTests testnoj klasi ispitali smo funkcionalnost dodavanja novog korisnika u bazu podataka.

Naredbom *Mockito.when(userRepository.save(Mockito.any())).thenReturn(test);* smo rekli mockanom userRepository-ju kako da se ponaša prilikom poziva save metode.

```
@Test
public void addUserTest() {
    UserDTO userDTO = new UserDTO(
            loginEmailString: "peroperic@gmail.com",
            AUTHORITY.ADMIN,
            firstName: "pero",
            lastName: "peric",
            notificationEmailString: "nest@nesto.com",
            description: "opis",
            nickname: "perica"
    );
    AppUser test = new AppUser(
            loginEmailString: "peroperic@gmail.com",
            AUTHORITY.ADMIN,
            firstName: "pero",
            lastName: "peric",
            notificationEmailString: "nest@nesto.com",
            description: "opis",
            nickname: "perica"
    );

    Mockito.when(userRepository.save(Mockito.any())).thenReturn(test);

    AppUser user = userService.addUser(userDTO);

    assertThat(user).isSameAs(test);
}
```

Slika 4.1: Add user test

```
Tests passed: 1 of 1 test – 855 ms
Starting Gradle Daemon...
Gradle Daemon started in 8 s 370 ms

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test
BUILD SUCCESSFUL in 23s
4 actionable tasks: 1 executed, 3 up-to-date
12:02:29 AM: Execution finished ':test --tests "com.example.backend.UserServiceTests.addUserTest"'.
```

Slika 4.2: Add user test result

2. **ProjectServiceTests**

U ProjectServiceTests klasi smo ispitali funkcionalnosti dohvata projekta iz baze podataka te dodavanje FR team membera na projekt.

```
@Test
public void findById_ReturnExpected(){
    Project project = mockProject();
    Mockito.when(projectRepository.findById(Mockito.any())).thenReturn(Optional.of(project));
    Project result = projectService.findById(Long.valueOf( l: 1));
    assertThat(result).isSameAs(project);
}
```

Slika 4.3: Find project by Id test

```
Tests passed: 1 of 1 test – 877 ms

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test
BUILD SUCCESSFUL in 3s
4 actionable tasks: 1 executed, 3 up-to-date
12:06:50 AM: Execution finished ':test --tests "com.example.backend.ProjectServiceTests.findById_ReturnExpected"'.
```

Slika 4.4: Find project by Id test result

```java
@Test
public void addFrTeamMember_AddsTeamMember(){
    Project project = mockProject();
    AppUser user = mockUser();
    Mockito.when(projectRepository.findById(Mockito.any())).thenReturn(Optional.of(project));
    Mockito.when(userRepository.findById(Mockito.any())).thenReturn(Optional.of(user));
    projectService.addFrTeamMember(Long.valueOf( l: 1), Long.valueOf( l: 1));
    assertThat(project.getFrteammembers()).contains(user);
    assertThat(project.getFrteammembers().size()).isEqualTo( expected: 1);
}
```

Slika 4.5: Add FR team member to project test

```
Tests passed: 1 of 1 test – 821 ms

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test
BUILD SUCCESSFUL in 2s
4 actionable tasks: 1 executed, 3 up-to-date
12:08:54 AM: Execution finished ':test --tests "com.example.backend.ProjectServiceTests.addFrTeamMember_AddsTeamMember"'.
```

Slika 4.6: Add FR team member result

3. CompanyServiceTests

U CompanyServiceTests klasi smo ispitali sljedeće funkcionalnosti:

Kreiranje nove kompanije

Dohvaćanje svih kompanij

Bacanje EntityNotFoundException-a prilikom pokušaja brisanja nepostojeće kompanije

Bacanje AuthenticationException-a prilikom pokušaja dohvata podataka od kompanije od strane Usera koji je samo Observer

```java
@Test
public void createCompany_CreatesCompany() throws AuthenticationException
{
    CompanyDto companyDto = new CompanyDto(
            name: "name",
            domain: "domain",
            abcCategory: 'a',
            budgetPlanningMonth: 2,
            country: "Croatia",
            zipCode: 10000,
            address: "Address",
            webUrl: "www.example.com",
            contactInFuture: false);
    Company company = new Company(
            name: "name",
            domain: "domain",
            abcCategory: 'a',
            budgetPlanningMonth: 2,
            country: "Croatia",
            zipCode: 10000,
            address: "Address",
            webUrl: "www.example.com",
            contactInFuture: false);

    Mockito.when(companyRepository.save(Mockito.any())).thenReturn(company);
    Company result = companyService.createCompany(mockUser(AUTHORITY.ADMIN), companyDto);
    assertThat(result).isSameAs(company);
}
```

Slika 4.7: Create company test

```
Tests passed: 1 of 1 test – 805 ms

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test
BUILD SUCCESSFUL in 2s
4 actionable tasks: 1 executed, 3 up-to-date
12:16:18 AM: Execution finished ':test --tests "com.example.backend.CompanyServiceTests.createCompany_CreatesCompany"'.
```

Slika 4.8: Create company test result

```java
@Test
public void getAllCompanies_ReturnsCompanies() throws AuthenticationException
{
    List<Company> companies = List.of(
        new Company(Long.valueOf( l 1), name: "name", domain: "domain", abcCategory: 'a', budgetPlanningMonth: 3, country: "Croatia", zipCode: 10000, address: "Address", webUrl: "example.com", contactInFuture: false),
        new Company(Long.valueOf( l 2), name: "name", domain: "domain", abcCategory: 'a', budgetPlanningMonth: 3, country: "Croatia", zipCode: 10000, address: "Address", webUrl: "example.com", contactInFuture: false),
        new Company(Long.valueOf( l 3), name: "name", domain: "domain", abcCategory: 'a', budgetPlanningMonth: 3, country: "Croatia", zipCode: 10000, address: "Address", webUrl: "example.com", contactInFuture: false));
    Mockito.when(companyRepository.findAll()).thenReturn(companies);

    List<Company> result = companyService.getAllCompanies(mockUser(AUTHORITY.ADMIN));
    assertThat(result).isSameAs(companies);
}
```

Slika 4.9: Get all companies test

```
Tests passed: 1 of 1 test – 852 ms

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava
> Task :processTestResources NO-SOURCE
> Task :testClasses
> Task :test
BUILD SUCCESSFUL in 5s
4 actionable tasks: 2 executed, 2 up-to-date
12:12:22 AM: Execution finished ':test --tests "com.example.backend.CompanyServiceTests.getAllCompanies_ReturnsCompanies"'.
```

Slika 4.10: Get all companies test result

```
@Test
public void deleteCompany_WhenCompanyDoesntExist_ThrowsEntityNotFoundException(){
    Mockito.when(companyRepository.findById(Mockito.any())).thenReturn( value: Optional.empty());
    Assertions.assertThrows(EntityNotFoundException.class, () -> {
        companyService.deleteCompany(mockUser(AUTHORITY.ADMIN), Long.valueOf( l: 1));
    });
}
```

Slika 4.11: Delete company which does not exist test

```
Tests passed: 1 of 1 test – 784 ms

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test
BUILD SUCCESSFUL in 2s
4 actionable tasks: 1 executed, 3 up-to-date
12:17:25 AM: Execution finished ':test --tests "com.example.backend.CompanyServiceTests.deleteCompany_WhenCompanyDoesntExist_ThrowsEntityNotFoundException"'.
```

Slika 4.12: Delete company which does not exist tes result

```
@Test
public void getCompany_IfUserIsObserver_ThrowsAuthenticationException() throws AuthenticationException
{
    Assertions.assertThrows(AuthenticationException.class, () -> {
        companyService.getCompany(mockUser(AUTHORITY.OBSERVER), Long.valueOf( l: 1));
    });
}
```

Slika 4.13: Get company info by observer test

```
Tests passed: 1 of 1 test – 745 ms

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test
BUILD SUCCESSFUL in 2s
4 actionable tasks: 1 executed, 3 up-to-date
12:14:50 AM: Execution finished ':test --tests "com.example.backend.CompanyServiceTests.getCompany_IfUserIsObserver_ThrowsAuthenticationException"'.
```

Slika 4.14: Get company info by observer test result

4. CollaborationServiceTests

U CollaborationServiceTests testnoj klasi smo ispitali funkcionalnost dodavanja suradnje u bazu podataka.

```java
@Test
public void addCollaboration_AddsCollaboration(){
    Project project = mockProject();
    Company company = mockCompany();
    Contact contact = mockContact();
    AppUser responsible = mockUser();

    Mockito.when(projectRepository.findById(Mockito.any())).thenReturn(Optional.of(project));
    Mockito.when(companyRepository.findById(Mockito.any())).thenReturn(Optional.of(company));
    Mockito.when(contactRepository.findById(Mockito.any())).thenReturn(Optional.of(contact));
    Mockito.when(userRepository.findById(Mockito.any())).thenReturn(Optional.of(responsible));

    Collaboration collaboration = new Collaboration(
            new CollaborationId(project, company),
            contact,
            responsible,
             priority: false,
            Category.ACADEMIC,
            Status.LETTER,
             comment: "Comment",
            Long.valueOf( l: 2000)
    );
    Mockito.when(collaborationsRepository.save(Mockito.any())).thenReturn(collaboration);
    CollaborationDTO collaborationDTO = new CollaborationDTO(
            Long.valueOf( l: 1),
            Long.valueOf( l: 1),
            Long.valueOf( l: 1),
             priority: false,
            Category.ACADEMIC,
            Status.LETTER,
             comment: "Comment",
            Long.valueOf( l: 2000)
    );

    Collaboration result = collaborationsService.addCollaboration(Long.valueOf( l: 1), collaborationDTO);
    assertThat(result).isSameAs(collaboration);
}
```

Slika 4.15: Add collaboration test

```
✔ Tests passed: 1 of 1 test – 1 sec 51 ms

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test
BUILD SUCCESSFUL in 3s
4 actionable tasks: 1 executed, 3 up-to-date
12:26:22 AM: Execution finished ':test --tests "com.example.backend.CollaborationsServiceTests.addCollaboration_AddsCollaboration"'.
```

Slika 4.16: Add collaboration test result

4.2.2 Ispitivanje sustava

Potrebno je provesti i opisati ispitivanje sustava koristeći radni okvir Selenium[1]*. Razraditi* ***minimalno 4 ispitna slučaja*** *u kojima će se ispitati redovni slučajevi, rubni uvjeti te poziv funkcionalnosti koja nije implementirana/izaziva pogrešku kako bi se vidjelo na koji način sustav reagira kada nešto nije u potpunosti ostvareno. Ispitni slučaj se treba sastojati od ulaza (npr. korisničko ime i lozinka), očekivanog izlaza ili rezultata, koraka ispitivanja i dobivenog izlaza ili rezultata.*

[1] https://www.seleniumhq.org/

Izradu ispitnih slučajeva pomoću radnog okvira Selenium moguće je provesti pomoću jednog od sljedeća dva alata:

- *dodatak za preglednik **Selenium IDE** - snimanje korisnikovih akcija radi automatskog ponavljanja ispita*
- ***Selenium WebDriver** - podrška za pisanje ispita u jezicima Java, C#, PHP koristeći posebno programsko sučelje.*

Detalji o korištenju alata Selenium bit će prikazani na posebnom predavanju tijekom semestra.

4.3 Dijagram razmještaja

dio 2. revizije

*Potrebno je umetnuti **specifikacijski** dijagram razmještaja i opisati ga. Moguće je umjesto specifikacijskog dijagrama razmještaja umetnuti dijagram razmještaja instanci, pod uvjetom da taj dijagram bolje opisuje neki važniji dio sustava.*

4.4 Upute za puštanje u pogon

dio 2. revizije

U ovom poglavlju potrebno je dati upute za puštanje u pogon (engl. deployment) ostvarene aplikacije. Na primjer, za web aplikacije, opisati postupak kojim se od izvornog kôda dolazi do potpuno postavljene baze podataka i poslužitelja koji odgovara na upite korisnika. Za mobilnu aplikaciju, postupak kojim se aplikacija izgradi, te postavi na neku od trgovina. Za stolnu (engl. desktop) aplikaciju, postupak kojim se aplikacija instalira na računalo. Ukoliko mobilne i stolne aplikacije komuniciraju s poslužiteljem i/ili bazom podataka, opisati i postupak njihovog postavljanja. Pri izradi uputa preporučuje se ***naglasiti korake instalacije uporabom natuknica*** *te koristiti što je više moguće* ***slike ekrana*** *(engl. screenshots) kako bi upute bile jasne i jednostavne za slijediti.*

Dovršenu aplikaciju potrebno je pokrenuti na javno dostupnom poslužitelju. Studentima se preporuča korištenje neke od sljedećih besplatnih usluga: Amazon AWS, Microsoft Azure ili Heroku. Mobilne aplikacije trebaju biti objavljene na F-Droid, Google Play ili Amazon App trgovini.

Kako se aplikacija sastoji od dva distinktivna dijela, backend-a i frontend-a, svaki dio funkcionira samo za sebe te se zasebno pušta u pogon.

4.4.1 Backend

Backend se sastoji od koda u obliku gradle projekta u rađenom u Spring boot okruženju. Kako Spring boot samostalno radi tablice i ažurira relacije u bazi podataka, potrebno je samo na poslužitelju na kojem će se nalaziti kompajliran pokrenut kod stvoriti bazu podataka s nazivom *cdb*.

Za početak je potrebno otvoriti virtualni stroj s ubuntu serverom na poslužitelju po izboru nakon čega je potrebno unesti sljedeće naredbe koristeći command prompt kako bi se ulogirali i postavili server:

- ssh root@ipv4AdresaServera
- sudo apt update
- sudo apt upgrade
- sudo apt install fail2ban
- sudo systemctl restart fail2ban.service

Sada kada je server postavljen, potrebno je instalirati postgresql i podignuti bazu naziva cdb:

- sudo apt-get install postgresql postgresql-contrib
- sudo systemctl start postgresql.service
- sudo systemctl enable postgresql.service
- sudo -u postgres psql
- \password postgres
- CREATE DATABASE cdb
- \c cdb
- \q

Nakon što je baza podataka na serveru stvorena, potrebno je kompajlirati kod za backend aplikacije te ga tako kompajliranog prebaciti na server. Sljedeće naredbe potrebno je izvršiti u matičnog folderu koda preuzetnog na vlastito računalo:

- cd Backend
- gradle bootJar
- scp build/libs/backend-0.0.1-SNAPSHOT.jar root@ipv4AdresaServera:/var/www

Nakon što je kompajlirani kod na serveru, potrebno je instalirati javu kako bi ga mogli pokrenuti:

- sudo apt install openjdk-17-jdk

Kako bi osigurali da aplikacija radi i kada na serveru ne postoji ulogirani korisnik, potrebno je kompajliran kod pokrenuti kao servis:

- cd /usr/lib/systemd/system
- nano runSpringServer.service
- *Sljedeći kod upiši u novootvoreni text editor:*
- [Unit]
- Description=webserver Daemon
- [Service]
- ExecStart=/usr/bin/java -jar /var/www/backend-0.0.1-SNAPSHOT.jar
- User=root
- [Install]
- WantedBy=multi-user.target
- *Zatvori text editor koristeći kombinacije tipki CTRL+S pa CTRL+X.*
- sudo systemctl start runSpringServer.service
- sudo systemctl enable runSpringServer.service

4.4.2 Frontend

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE
5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Astah Community, http://astah.net/editions/uml-new

Indeks slika i dijagrama

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak
   - Datum: 17. listopada 2022.
   - Prisustvovali: P. Hajduk., M. Čengić, N. Capan, J. Jakovac
   - Teme sastanka:
     - Predstavljanje nove teme za prijedlog
     - Rasprava o novoj temi za prijedlog
     - Rasprava o mogućoj podjeli rada
     - Teambuilding
2. sastanak
   - Datum: 14. studenoga 2022.
   - Prisustvovali: P. Hajduk., M. Čengić, N. Capan, J. Jakovac, M. Balog, I. Baričević, L. Čunović
   - Teme sastanka:
     - Rasprava o implementacijskim detaljima
     - Rasprava o daljnjim koracima
3. sastanak
   - Datum: 04. prosinca 2022.
   - Prisustvovali: P. Hajduk., M. Čengić, N. Capan, J. Jakovac, M. Balog, I. Baričević
   - Teme sastanka:
     - Rekapitulacija detalja napravljenog
     - Rasprava o bazičnim stavkama funkcionalnosti aplikacije

Tablica aktivnosti

	Petar Hajduk	Jakov Jakovac	Matej Balog	Ivor Baričević	Marko Čengić	Nikola Capan	Lovro Čunović
Upravljanje projektom	0	18	0	0	2	0	0
Opis projektnog zadatka	0	10	0	0	0	0	2
Funkcionalni zahtjevi	0	1	0	0	2	0	0
Opis pojedinih obrazaca	0	3	0	0	8	0	0
Dijagram obrazaca	0	0	0	0	4	0	0
Sekvencijski dijagrami	0	0	0	0	4	0	0
Opis ostalih zahtjeva	0	0	0	0	1	0	0
Arhitektura i dizajn sustava	0	3	0	0	0	0	2
Baza podataka	2	2	0	0	0	7	0
Dijagram razreda	0	0	0	0	0	8	0
Dijagram stanja	0	0	0	0	0	0	0
Dijagram aktivnosti	0	0	0	0	0	0	0
Dijagram komponenti	0	0	0	0	0	0	0
Korištene tehnologije i alati	0	1	0	0	0	0	0
Ispitivanje programskog rješenja	0	2	0	0	0	0	0
Dijagram razmještaja	0	0	0	0	0	0	0
Upute za puštanje u pogon	0	1	0	0	0	0	0
Dnevnik sastajanja	0	1	0	0	0	0	0
Zaključak i budući rad	0	0	0	0	0	0	0
Popis literature	0	0	0	0	0	0	0

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Petar Hajduk	Jakov Jakovac	Matej Balog	Ivor Baričević	Marko Čengić	Nikola Capan	Lovro Čunović
Dizajn aplikacije	0	13	1	1	0	0	0
Rad na responzivnosti aplikacije	0	0	0	0	0	0	0
Implementacija Login/Logout	0	7	0	0	0	0	0
Implementacija Razina ovlasti	0	0	0	0	0	0	0
Izrada headera	0	2	0	0	0	0	0
Izrada naslovnice	0	1	0	0	0	0	0
Izrada users stranice	0	1	4	3	0	0	0
Izrada user forme	0	4	0	2	0	0	0
Izrada delete forme	0	0	0	1	0	0	0
Izrada projects stranice	0	0	0	0	0	0	0
Izrada project forme	0	0	0	0	0	0	0
Izrada companies stranice	0	0	0	0	0	0	0
Izrada company forme	0	0	0	0	0	0	0
Izrada collaborations komponente	0	0	0	0	0	0	0
Izrada collaboration forme	0	0	0	0	0	0	0
Izrada spring backenda	2	1	0	2	0	0	0
Deploy aplikacije	0	6	0	0	0	0	0