Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Virtualni ormar

Tim: <TG03.1>

Članovi tima Ormarko:

> Andrija Andročec - razvoj poslužiteljske strane
> Lana Benički - razvoj poslužiteljske strane
> Tanja Bertalanić - voditelj tima/razvoj korisničkog sučelja/ dizajner
> Luka Bulić - baza podataka/ispitivanje
> Lovro Đuranec - razvoj poslužiteljske strane
> Marko Vujnović - dokumentacija
> Iva Zubčić - razvoj korisničkog sučelja

Nastavnik: Alan Jović

Opis projektnog zadatka

Dobra organizacija garderobe može biti izazovna, posebno za osobe s velikim brojem odjevnih predmeta. Ponekad želimo podijeliti našu omiljenu odjeću s drugima, a nemamo način da to izvedemo. S obzirom na ove izazove, ova aplikacija trebala bi služiti kao organizacijski alat koji nudi prilagodljivo rješenje za osobe u smislu upravljanja virtualnim ormarima i olakšava dijeljenje artikala.

Glavni cilj aplikacije je omogućiti korisnicima jednostavno i intuitivno kreiranje virtualnog ormara koji uključuje klasifikaciju odjevnih predmeta prema različitim karakteristikama (sezona, otvorenost, ležernost i slično). Korisnici će moći pretraživati ormare prema osobnim preferencijama, stvarati različite odjevne kombinacije temeljem unesenih kriterija i trenutne vremenske prognoze. Korisnici mogu dijeliti svoje odjevne artikle s drugima i pretraživati oglase kako bi našli točno ono što im treba da popune vlastiti ormar. ## Potencijalna korist ovog projekta 1. Organizacijska učinkovitost: Pomaže korisnicima u organizaciji odjeće, štedi vrijeme i smanjuje stres pri odabiru odjevnih kombinacija. 2. Društveni aspekt: Omogućava korisnicima dijeljenje odjevnih artikala s prijateljima, obitelji ili širim krugom korisnika. 3. Olakšva nabavu: Korisnik vidi točno koji mu odjevni artikli nedostaju u ormaru i zna što treba tražiti u oglasima. 4. Korištenje modernih tehnologija: Aplikacija koristi geolokaciju i vremensku prognozu za prilagodbu odabira odjeće u stvarnom vremenu, što dodatno poboljšava korisničko iskustvo. ## Postojeća slična rješenja - Cladwell: Omogućava kreiranje kapsulnih ormara i sugerira odjevne kombinacije. Cladwell ima naglasak na minimalizmu i organiziranju osnovnih odjevnih komada, dok naša aplikacija nudi veći izbor kategorija i prilagodbu strukture ormara. - Stylebook: Aplikacija koja prati što korisnici nose i omogućava detaljno praćenje nošenih artikala. Za razliku od Stylebooka, naš projekt ima opciju dijeljenja artikala i razmjenu s drugim korisnicima. - Pureple: Aplikacija koja koristi umjetnu inteligenciju za preporuku odjevnih kombinacija. Naša aplikacija nudi specifičnije karakteristike artikala, kao što su sezonska prilagođenost i prilagodba specifičnim vremenskim uvjetima. ## Skup korisnika koji bi mogao biti zainteresiran za ostvareno rješenje. - Pojedinci s velikim brojem odjevnih artikala koji žele bolju organizaciju svog ormara. - Ljubitelji mode i stila koji žele svakodnevno eksperimentirati s različitim odjevnim kombinacijama. - Oglašivači u modnoj industriji koji mogu iskoristiti ovu platformu za oglašavanje svojih proizvoda i približiti se korisnicima. ## Mogućnost prilagodbe rješenja - Prilagodba strukture ormara: Korisnici mogu odabrati specifične vrste skladištenja unutar virtualnog ormara (npr. police, ladice). - Personalizacija artikala i kombinacija: Unutar aplikacije korisnici mogu definirati različite kategorije i vrste kombinacija prema svom stilu ili preferencijama. - Korisnički profil s više ormara: Svaki korisnik može imati više ormara s različitim sadržajima, što omogućava korisnicima da organiziraju odjeću po sezoni, vrsti (npr. poslovna odjeća) ili drugim kriterijima. ## Opseg projektnog zadatka 1. Kreiranje i organizacija virtualnih ormara s različitim strukturama. 2. Kategorizaciju i dodavanje artikala prema više značajki. 3. Pretraživanje i dijeljenje artikala u virtualnom

ormaru. 4. Korisničke profile i geolokaciju za praćenje i dijeljenje lokacija ormara. 5. Oglašivačke profile za dodavanje promocijskih materijala. 6. Vremenske prognoze kao dodatnu funkcionalnost za prilagođeni izbor odjeće.

Moguće nadogradnje projektnog zadatka

1. Podrška za više jezika: Kako bi aplikacija bila dostupna međunarodnom tržištu.
2. Dodavanje naprednih filtera: Kao što su materijali, brendovi, posebne oznake za eko i reciklirane proizvode.
3. Optimizacija društvenih značajki: Dodavanje mogućnosti komentiranja ili praćenja ormara drugih korisnika.
4. Integracija umjetne inteligencije za preporuke odjeće: Korisnicima se mogu predložiti artikli prema njihovim stilskim preferencijama.
5. Automatska prepoznavanja slike: U budućnosti bi se mogla uvesti opcija prepoznavanja odjevnih predmeta putem slika, što bi dodatno olakšalo unos.

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-001	Generički korisnici mogu pregledavati i pretraživati artikle označene za dijeljenje prema svim karakteristikama artikla.	Visok	Dokument zahtjeva	Generički korisnik može pretraživati artikle koristeći kategorije, boju, vrstu i slično.
F-002	Generički korisnici mogu pretraživati artikle označene za dijeljenje u koristeći geolokaciju.	Srednji	Dokument zahtjeva	Generički korisnik može pretraživati artikle koristeći geolokaciju.
F-003		Visok		

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
	Generički korisnici mogu detaljno pregledati informacije o artiklu i kontakt informacije vlasnika artikla.		Dokument zahtjeva	Generički korisnik može vidjeti naziv, sliku, kategoriju i kontakt podatke za artikl.
F-004	Neregistrirani korisnici mogu se registrirati i prijaviti u aplikaciju.	Visok	Dokument zahtjeva	Korisnik se može uspješno registrirati i prijaviti putem obrasca za kreiranje virtualnog ormara.
F-005	Registrirani korisnici mogu dodati novi virtualni ormar s prilagodljivom strukturom (polica, ladica, šipka).	Visok	Dokument zahtjeva	Korisnik može dodati ormar s definiranom strukturom i brojem lokacija.
F-006	Registrirani korisnici mogu dodavati artikle u svoje virtualne ormare s definiranim karakteristikama (naziv, slika, kategorija, boja, itd.).	Visok	Dokument zahtjeva	Korisnik može uspješno dodati artikl s kompletnim informacijama.
F-007	Registrirani korisnici mogu pretraživati svoje virtualne ormare prema	Srednji	Dokument zahtjeva	Korisnik može dobiti rezultate pretrage s prikazom rednog broja ormara i

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
	karakteristikama artikla.			lokacije artikla.
F-008	Registrirani korisnici mogu dijeliti artikle s drugim korisnicima, omogućujući pristup informacijama i kontaktu.	Visok	Dokument zahtjeva	Korisnik može označiti artikl kao podijeljen, čime postaje vidljiv svim korisnicima.
F-009	Aplikacija predlaže odjevne kombinacije na temelju zadanih kriterija (boja, aktivnost i slično).	Srednji	Dokument zahtjeva	Korisnik dobiva popis preporučenih artikala s njihovim lokacijama unutar ormara.
F-010	Registrirani korisnici mogu ukloniti ili izmijeniti strukturu virtualnog ormara, uključujući uklanjanje artikala.	Visok	Dokument zahtjeva	Korisnik može uspješno ukloniti ormar, promijeniti njegovu strukturu ili ukloniti artikl iz ormara.
F-011	Na profilima korisnika prikazuje se logo oglašivača koji vodi na galeriju artikala.	Visok	Dokument zahtjeva	Logo oglašivača je klikabilan i vodi na galeriju s artiklima, gdje su prikazani naziv, slika i cijena artikla.
F-012	Oglašivači mogu dodavati i	Visok	Dokument zahtjeva	Oglašivač može uspješno

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
	mijenjati artikle u svojoj galeriji s osnovnim informacijama (slika, naziv, kategorija, cijena).			dodati artikl u galeriju sa slikom i cijenom.

Nefunkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet
NF-001	Korisničko sučelje aplikacije mora omogućiti responzivni prikaz na različitim veličinama zaslona.	Visok
NF-002	Aplikacija mora podržavati minimalno hrvatski i engleski jezik u korisničkom sučelju.	Srednji
NF-003	Sustav treba omogućiti vrijeme odziva kraće od 3 sekunde za sve osnovne funkcije, poput otvaranja ormara ili pretraživanja artikala.	Visok
NF-004		Srednji

ID zahtjeva	Opis	Prioritet
	Sustav mora biti optimiziran za rad do 1000 korisnika koji aplikaciju koriste istovremeno.	
NF-005	Sustav mora koristiti HTTPS protokol za sve mrežne komunikacije kako bi osigurao sigurnost podataka korisnika.	Visok
NF-006	Korisnički podaci, kao što su korisničko ime i e-mail adresa, moraju biti zaštićeni odgovarajućim metodama enkripcije.	Visok
NF-007	Aplikacija mora imati jasnu i jednostavnu strukturu koda, dokumentiranu prema općeprihvaćenim standardima, kako bi se olakšalo održavanje.	Visok

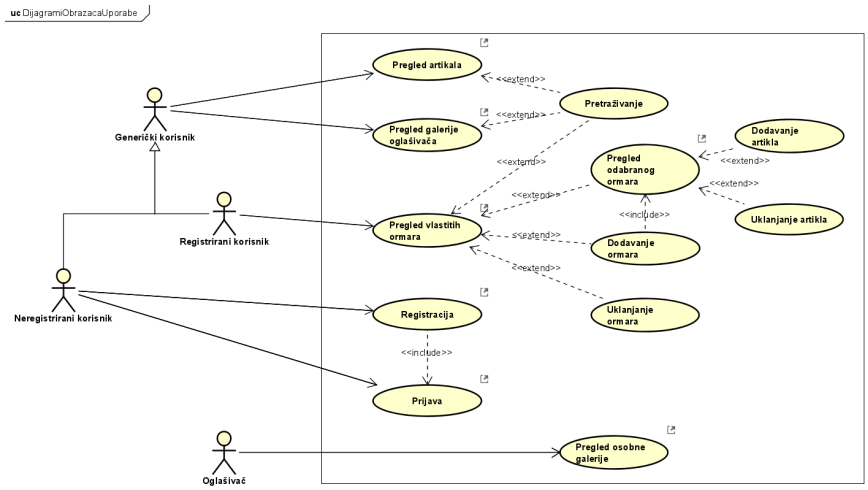
Dionici

1. Korisnici aplikacije
    - Generički korisnik
        - Neregistrirani korisnik
        - Registrirani korisnik
    - Oglašivač
2. Razvojni tim

Obrasci uporabe

Opis obrazaca uporabe

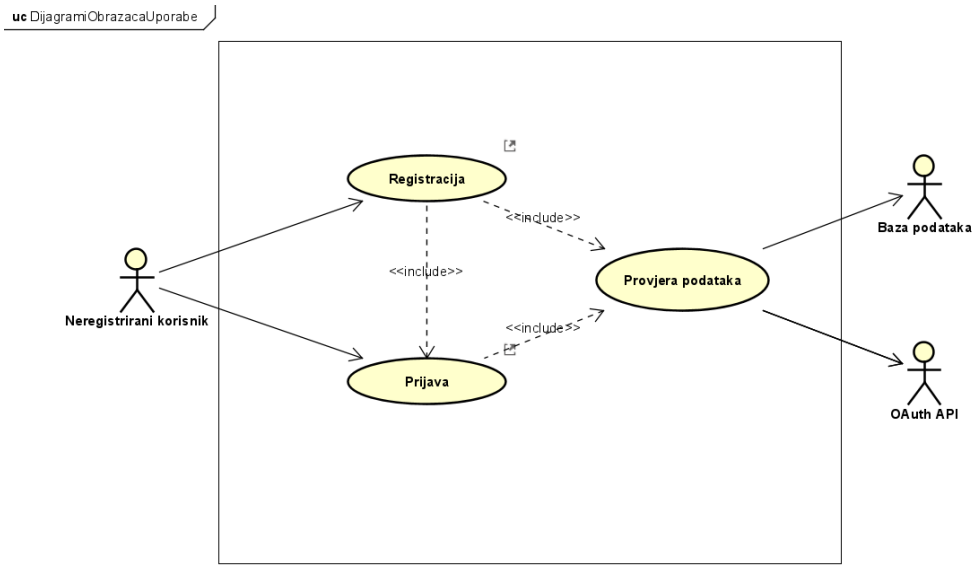
1. Visokorazinski dijagram obrazaca uporabe cijelog sustava



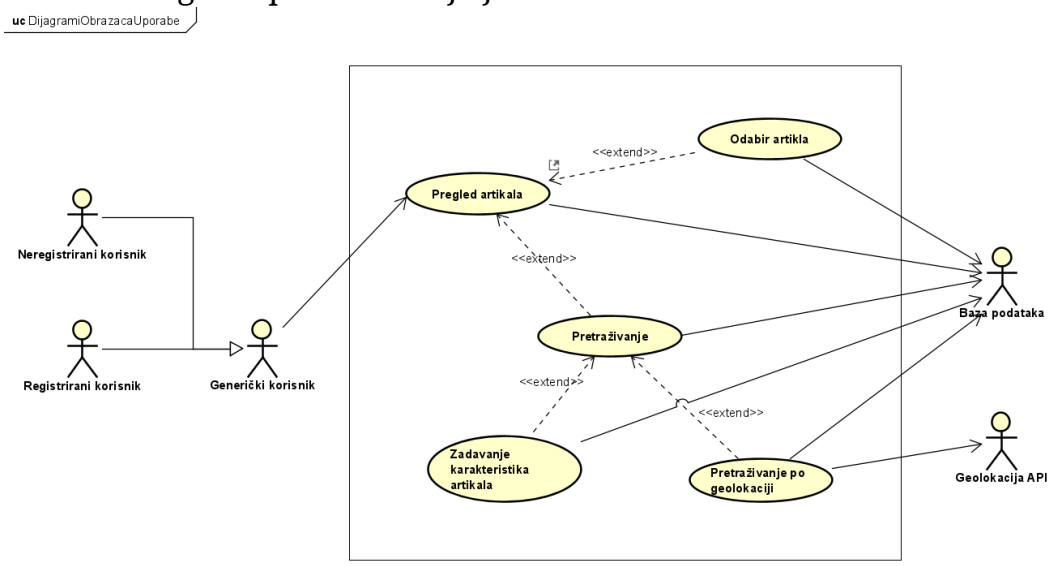
image

2. Dijagram obrazaca uporabe za ključne funkcionalnosti, korisničke role i interakcijom s vanjskim sustavima

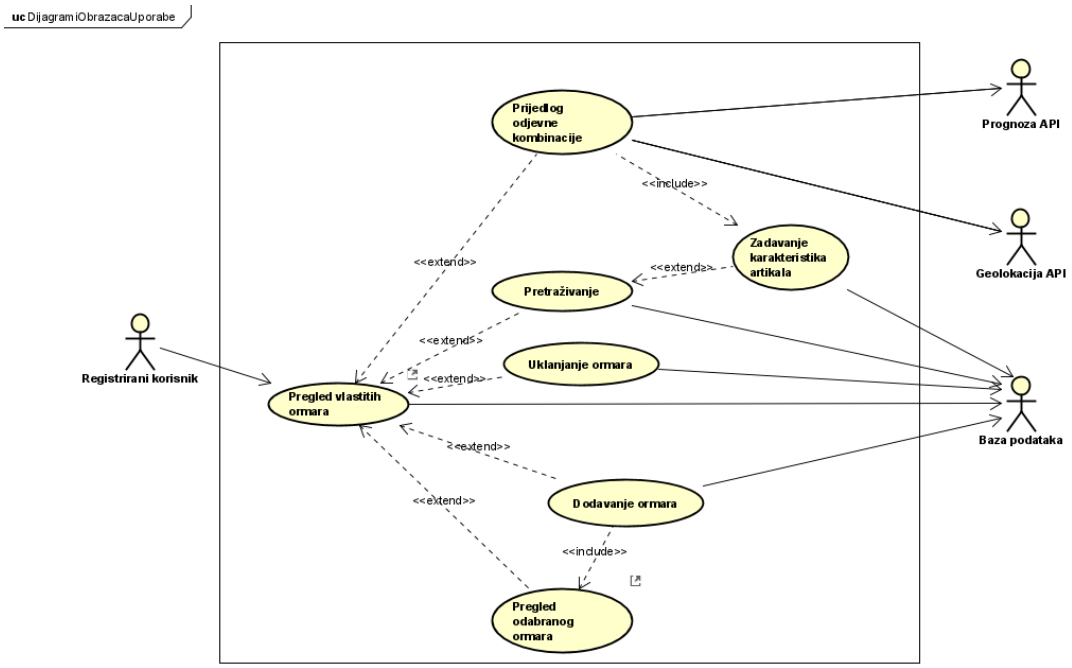
2.1. Prijava i registracija



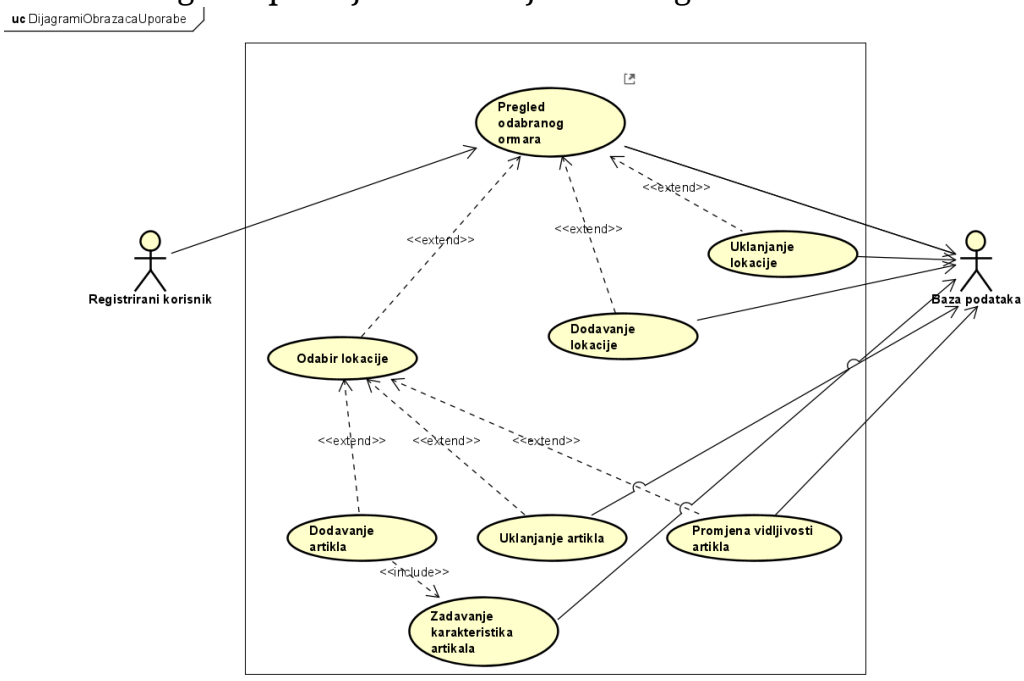
2.2. Pregled i pretraživanje javnih artikala



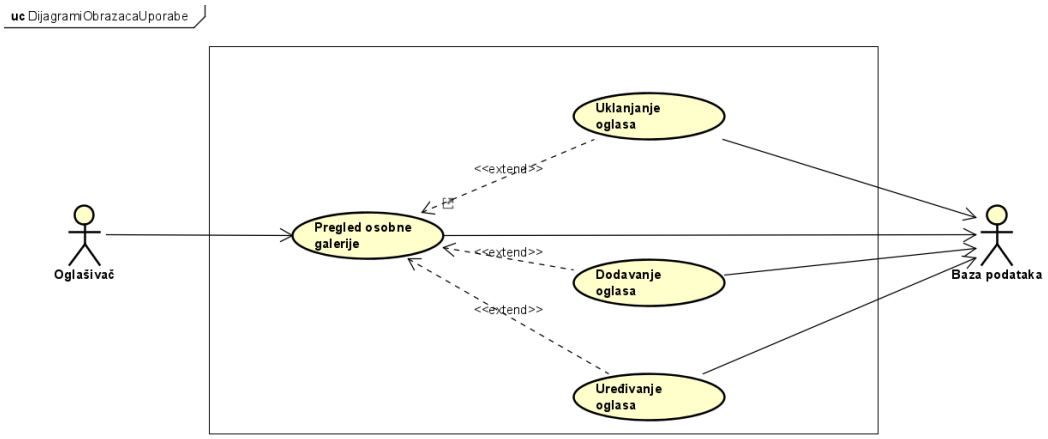
2.3. Pregled vlastitih ormara i prijedlog odjevne kombinacije



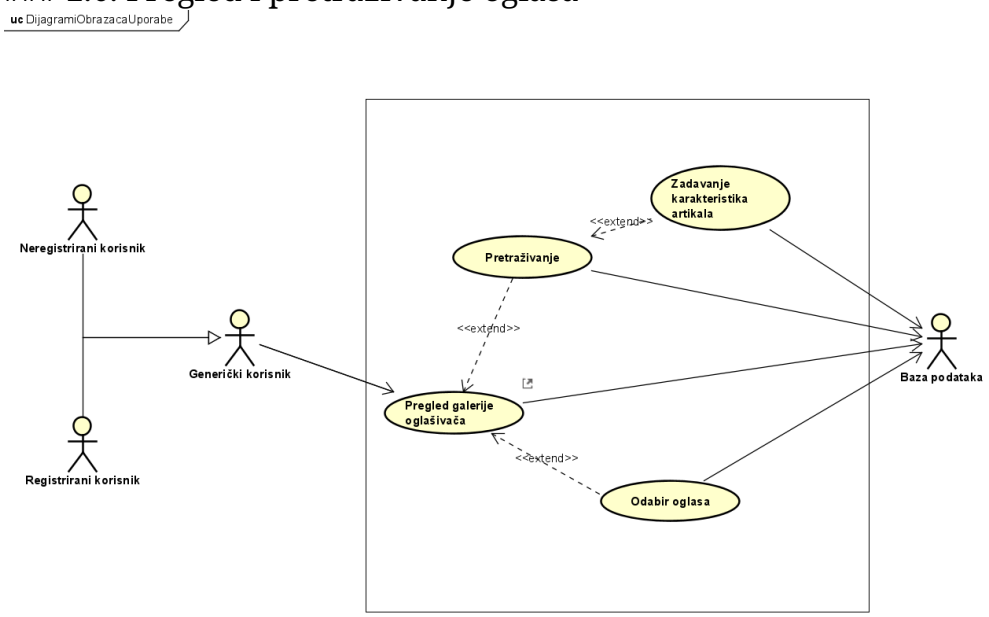
2.4. Pregled i promjene sadržaja vlastitog ormara



2.5. Pregled i promjene osobne galerije oglašivača



2.6. Pregled i pretraživanje oglasa



UC-001 - Registracija

- Glavni sudionik: Neregistrirani korisnik
- Cilj: Stvaranje novog korisničkog računa
- Sudionici: Neregistrirani korisnik, Baza podataka
- Preduvjet: Neregistrirani korisnik stišče gumb “Registracija”

• Opis osnovnog tijeka: > 1. Korisnik upisuje svoje podatke u obrazac > 2. Podaci se provjeravaju s obrascem uporabe “Provjera podataka” > 3. Stvara se novi korisnički račun i korisnik je automatski prijavljen nakon registracije > 4. Zatim se uklanja kartica i vraća korisnika na isto mjesto gdje se nalazio prije registracije

• Opis mogućih odstupanja: > Nastavlja se od koraka 3. > 3. Ako već postoji registrirani korisnik sa istim podacima, registracija je neuspješna i odbija se > 4. Korisnik može probati opet ili izaći iz kartice klikom na gumb “X”

UC-002 - Prijava

- Glavni sudionik: Neregistrirani korisnik
- Cilj: Prijava u postojeći korisnički račun
- Sudionici: Neregistrirani korisnik, Baza podataka
- Preduvjet: Neregistrirani korisnik stišče gumb “Prijava”

• Opis osnovnog tijeka: > 1. Korisnik upisuje svoje podatke u obrazac > 2. Podaci se provjeravaju s obrascem uporabe “Provjera podataka” > 3. Korisnik je prijavljen u svoj korisnički račun > 4. Zatim se uklanja kartica i vraća korisnika na isto mjesto gdje se nalazio prije registracije

• Opis mogućih odstupanja: > Nastavlja se od koraka 3 > 3. Ako ne postoji račun s unesenim podacima, prijava je neuspješna i odbija se > 4. Korisnik može probati opet ili izaći iz kartice klikom na gumb “X”

UC-003 - Provjera podataka

- Glavni sudionik: OAuth API
- Cilj: Provjera podataka za registraciju ili prijavu
- Sudionici: OAuth API, Baza podataka
- Preduvjet: Korisnik se probao prijaviti ili registrirati

• Opis osnovnog tijeka: > 1. Dobiva podatke iz ispunjenog obrasca za registraciju ili prijavu > 2. Ako je unesen obrazac za registraciju i ne postoji već stvoreni korisnički račun s istim podacima, stvara se novi korisnički račun i potvrđuje se registracija > 3. Ako je unese obrazac za prijavu i postoji već stvoreni korisnički račun s istim podacima, potvrđuje se prijava

• Opis mogućih odstupanja: > Nastavlja se od koraka 2 > 2. Ako je unesen obrazac za registraciju i postoji već stvoreni korisnički račun s istim podacima, odbija se registracija > 3. Ako je unesen obrazac za prijavu i ne postoji već stvoreni korisnički račun s istim podacima, odbija se prijava

UC-004 - Pregled artikala

- Glavni sudionik: Generički korisnik
- Cilj: Otvara glavnu karticu s pregledom artikala koji su označeni za dijeljenje
- Sudionici: Generički korisnik, Baza podataka
- Preduvjet: Nema

• Opis osnovnog tijeka: > 1. Ulaskom u aplikaciju prikazuje se kartica s pregledom artikala koji su označeni za dijeljenje

• Opis mogućih odstupanja: > 1. Ako se korisnik nalazi na drugoj kartici onda mora stinsuti ikonu loga “Ormarko” kako bi otvorio pregled artikala

UC-005 - Odabir artikla

- Glavni sudionik: Generički korisnik
- Cilj: Prikazuje detalje odabranog artikla u novoj kartici
- Sudionici: Generički korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda artikala

• Opis osnovnog tijeka: > 1. Korisnik želi pregledati dodatne informacije o artiklu i stišće na artikl > 2. Otvara se kartica s dodatnim informacijama. > 3. Nakon pregleda informacija korisnik mora stisnuti na gumb “X” kako bi izašao iz kartice i vratio se gdje se prije nalazio.

• Opis mogućih odstupanja: > Nema

UC-006 - Pretraživanje

- Glavni sudionik: Generički korisnik
- Cilj: Prikazuje polje za pretraživanje u koje se unose nazivi objekata koje se želi pretražiti
- Sudionici: Generički korisnik, Baza podataka
- Preduvjet: Korisnik klikne na polje za pretraživanje

• Opis osnovnog tijeka: > 1. Ako pretražuje zadavanjem karakteristika artikala onda odabire karakteristike > 2. Ako pretražuje po geolokaciji onda dobiva objekte u blizini sebe > 3. Nakon unosa za pretraživanje, korisnik mora stisnuti tipku “Enter” ili na gumb s ikonom povećala. > 4. Vraća korisnika na prikaz objekata filtriranih po njegovom izboru

• Opis mogućih odstupanja: > Nema

UC-007 - Zadavanje karakteristika artikala

- Glavni sudionik: Generički korisnik
- Cilj: Dodaje mogućnosti odabira karakteristika artikala
- Sudionici: Generički korisnik, Baza podataka
- Preduvjet: Nema

• Opis osnovnog tijeka: > 1. Korisnik ima mogućnost odabira karakteristika artikala (boja, aktivnost, godišnje doba i slično).

• Opis mogućih odstupanja: > Nema

UC-008 - Pretraživanje po geolokaciji

- Glavni sudionik: Geolokacija API
- Cilj: Dodaje u polje za pretraživanje mogućnost odabira pretraživanja po geolokaciji
- Sudionici: Geolokacija API, Generički korisnik, Baza podataka
- Preduvjet: Korisnik mora pretraživati artikle u kartici pregleda artikala

• Opis osnovnog tijeka: > 1. Korisnik odabire pretraživanje po geolokaciji > 2. U rezultatu pretraživanja nalaze se objekti koji su u korisnikovoj blizini

• Opis mogućih odstupanja: > Nema

UC-009 - Pregled vlastitih ormara

- Glavni sudionik: Registrirani korisnik
- Cilj: Otvara karticu s pregledom vlastitih ormara
- Sudionici: Registrirani korisnik, Baza podataka
- Preduvjet: Nema

• Opis osnovnog tijeka: > 1. Korisnik klikne na gumb “Moji ormari”. > 2. Prikazuje se kartica s pregledom vlastitih ormara

• Opis mogućih odstupanja: > Nema

UC-010 - Dodavanje ormara

- Glavni sudionik: Registrirani korisnik
- Cilj: Dodaje ormar u vlastite ormare i otvara karticu za izmjenu struktutre ormara
- Sudionici: Registrirani korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda vlastitih ormara

• Opis osnovnog tijeka: > 1. Stišće na gumb s ikonom plusa > 2. Korisniku se prikazuje kartica sa sadržajem ormara pomoću obrasca uporabe “Pregled odabranog ormara” > 3. Nakon dodavanja ormara, kartica se uklanja i vraća se na prikaz vlastitih ormara

• Opis mogućih odstupanja: Nastavlja se od koraka 2. > 2. Korisnik odbija dodavanje ormara tako da stišće na gumb “X” > 3. Vraća se na prikaz vlastitih ormara

UC-011 - Pregled odabranog ormara

- Glavni sudionik: Registrirani korisnik
- Cilj: Odabire ormar i prikazuje karticu sa sadržajem ormara
- Sudionici: Registrirani korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregledu vlastitih ormara

• Opis osnovnog tijeka: > 1. Korisnik odabire ormar i stišće na ormar kako bi otovrio sadržaj ormara > 2. Prikazuje se nova kratica za pregled, promjenu strukture i mijenjanje sadržaja odabranog ormara > 3. Vraća korisnika na pregled vlastitih ormara nakon izmjena sadržaja ormara

• Opis mogućih odstupanja: Nastavlja se od koraka 1. > 1. Korisnik dodaje ormar pomoću obrasca “Dodavanje ormara”. > 2. Prikazuje se nova kratica za pregled, promjenu strukture i mijenjanje sadržaja odabranog ormara > 3. Nakon dodavanja ormara, kartica se uklanja i vraća se na prikaz vlastitih ormara

UC-012 - Uklanjanje ormara

- Glavni sudionik: Registrirani korisnik
- Cilj: Uklanja odarani vlastiti ormar
- Sudionici: Registrirani korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda vlastitih ormara

• Opis osnovnog tijeka: > 1. Pronalazi ormar koji želi ukloniti > 2. Stišće na gumb s ikonom koša za smeće > 3. Prikazuje se kartica o potvrdi uklanjanja > 4. Korisnik potvrđuje uklanjanje, kartica se uklanja i ormar se uklanja

• Opis mogućih odstupanja: Nastalja se od koraka 4. > 4. Korisnik odbija uklanjanje, kartica se uklanja i ormar se ne uklanja

UC-013 - Prijedlog odjevne kombinacije

- Glavni sudionik: Registrirani korisnik
- Cilj: Daje prijedlog odjevne kombinacije s obzirom na trenutnu geolokaciju korisnika, prognozu vremena i artikle u vlastitim ormarima korisnika

- Sudionici: Registrirani korisnik, Geolokacija API, Prognoza API
- Preduvjet: Korisnik mora biti u kartici pregleda vlastitih ormara

• Opis osnovnog tijeka: > 1. Korisnik stišće na gumb “Prijedlog kombinacije” i otvara mu se kartica za prijedlog odjevne kombinacije > 2. Dohvaća se geolokacija korisnika pa zatim vremenska prognoza radi boljeg prijedloga > 3. Korisnik bira karakteristike artikala za kombinaciju pomoću obrasca uporabe “Zadavanje karakteristika artikala” i stišće gumb “Predloži” > 4. Na temelju karakteristika artikala slaže se kombinacija i prikazuje korisniku > 5. Korisnik stišće gumb “X” i izlazi iz kartice za prijedlog odjevne kombinacije i vraća se na pregled vlastitih ormara

• Opis mogućih odstupanja: > Nema

UC-014 - Dodavanje lokacije

- Glavni sudionik: Registrirani korisnik
- Cilj: Dodaje lokaciju u odabrani ormar
- Sudionici: Registrirani korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda odabranog ormara

• Opis osnovnog tijeka: > 1. Korisnik stišče na gumb s ikonom plusa > 2. Korisniku se prikazuje kartica za dodavanje lokacije > 3. Korisnik unosi tip lokacije > 4. Vraća se na prikaz odabranog ormara

• Opis mogućih odstupanja: Nastavlja se od koraka 3. > 3. Korisnik odbija dodavanje lokacije tako da stišće na gumb “X” > 4. Vraća se na prikaz odabranog ormara

UC-015 - Uklanjanje lokacije

- Glavni sudionik: Registrirani korisnik
- Cilj: Uklanja lokaciju iz odabranog ormara
- Sudionici: Registrirani korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda odabranog ormara

• Opis osnovnog tijeka: > 1. Pronalazi lokaciju koji želi ukloniti > 2. Stišće na gumb s ikonom koša za smeće > 3. Prikazuje se kartica o potvrdi uklanjanja > 4. Korisnik potvrđuje uklanjanje, kartica se uklanja i lokacija se uklanja

• Opis mogućih odstupanja: Nastalja se od koraka 4. > 4. Korisnik odbija uklanjanje, kartica se uklanja i lokacija se ne uklanja

UC-016 - Odabir lokacije

- Glavni sudionik: Registrirani korisnik
- Cilj: Prikazuje karticu sadržaja odabrane lokacije
- Sudionici: Registrirani korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda odabranog ormara

• Opis osnovnog tijeka: > 1. Stišće na lokaciju koju želi odabrati > 2. Korisniku se prikazuje kartica sa sadržajem lokacije kojeg može mijenjati > 3. Nakon izmjene sadržaja, kartica se uklanja i vraća se na prikaz odabrani ormara

• Opis mogućih odstupanja: > Nema

UC-017 - Dodavanje artikla

- Glavni sudionik: Registrirani korisnik
- Cilj: Dodaje artikl u ormar i otvara karticu za karakteristike artikla
- Sudionici: Registrirani korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda sadržaja lokacije

• Opis osnovnog tijeka: > 1. Stišće na gumb s ikonom plusa > 2. Prikazuje se kartica za karakteristike artikla pomoću obrasca uporabe “Zadavanje karakteristika artikala” > 3. Nakon izmjene karakteristika i unosa podataka, korisnik stišće na gumb “Dodaj”, artikl se dodaje u lokaciju i miče se kartica

• Opis mogućih odstupanja: Nastavlja se od koraka 2. > 2. Korisnik odbija dodavanje artikla tako da stišće na gumb “X” > 3. Vraća se na prikaz sadržaja lokacije

UC-018 - Uklanjanje artikla

- Glavni sudionik: Registrirani korisnik
- Cilj: Uklanja artikl iz odabrane lokacije
- Sudionici: Registrirani korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda sadržaja lokacije

• Opis osnovnog tijeka: > 1. Pronalazi artikl koji želi ukloniti > 2. Stišće na gumb s ikonom koša za smeće > 3. Prikazuje se kartica o potvrdi uklanjanja > 4. Korisnik potvrđuje uklanjanje, kartica se uklanja i artikl se uklanja

• Opis mogućih odstupanja: Nastalja se od koraka 4. > 4. Korisnik odbija uklanjanje, kartica se uklanja i artikl se ne uklanja

UC-019 - Promjena vidljivosti artikla

- Glavni sudionik: Registrirani korisnik
- Cilj: Mijenja vidljivost odabranog artikla
- Sudionici: Registrirani korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u pregledu sadržaja lokacije

• Opis osnovnog tijeka: > 1. Korisnik odabire artikl kojem želi promijeniti vidljivost > 2. Korisnik stišće gumb s ikonom prekriženog oka i artikl postaje vidljiv svim korisnicima

• Opis mogućih odstupanja: Nastavlja se od koraka 2. > 2. Korisnik stišće gumb s ikonom oka i artikl postaje vidlijv samo njemu

UC-020 - Pregled osobne galerije

- Glavni sudionik: Oglašivač
- Cilj: Otvara karticu s pregledom osobne galerije
- Sudionici: Oglašivač, Baza podataka
- Preduvjet: Nema

• Opis osnovnog tijeka: > 1. Korisnik mora stisnuti gumb “Moja galerija” > 2. Prikazuje se pregled osobne galerije

• Opis mogućih odstupanja: > Nema

UC-021 - Uklanjanje oglasa

- Glavni sudionik: Oglašivač
- Cilj: Uklanja oglas iz galerije
- Sudionici: Oglašivač, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda osobne galerije

• Opis osnovnog tijeka: > 1. Pronalazi oglas koji želi ukloniti > 2. Stišće na gumb s ikonom koša za smeće > 3. Prikazuje se kartica o potvrdi uklanjanja > 4. Korisnik potvrđuje uklanjanje, kartica se uklanja i oglas se uklanja

• Opis mogućih odstupanja: Nastalja se od koraka 4. > 4. Korisnik odbija uklanjanje, kartica se uklanja i oglas se ne uklanja

UC-022 - Dodavanje oglasa

- Glavni sudionik: Oglašivač

- Cilj: Dodaje oglas u osobnu galeriju i otvara karticu za odabir artikla
- Sudionici: Oglašivač, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda osobne galerije

• Opis osnovnog tijeka: > 1. Stišće na gumb s ikonom plusa > 2. Prikazuje se kartica za dodavanje oglasa > 3. Nakon postavljanja oglasa, korisnik stišće na gumb “Dodaj”, oglas se dodaje u galeriju i miče se kartica

• Opis mogućih odstupanja: Nastavlja se od koraka 3. > 3. Korisnik odbija dodavanje oglasa i stišće gumb “Odbaci”, oglas se ne dodaje u galeriju i miče se kartica

UC-023 - Uređivanje oglasa

- Glavni sudionik: Oglašivač
- Cilj: Uređuje oglas u osobnoj galeriji
- Sudionici: Oglašivač, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda osobne galerije

• Opis osnovnog tijeka: > 1. Odabire oglas za uređivanje i stišće na oglas > 2. Prikazuje se kartica za uređivanje odabranog oglasa > 3. Nakon uređivanja oglasa, korisnik stišće na gumb “Izmijeni”, oglas se uređuje i miče se kartica

• Opis mogućih odstupanja: Nastavlja se od koraka 3. > 3. Korisnik odbija uređivanje oglasa i stišće gumb “Odbaci”, oglas se ne uređuje i miče se kartica

UC-024 - Pregled galerije oglašivača

- Glavni sudionik: Generički korisnik
- Cilj: Otvara karticu s pregledom galerije oglašivača
- Sudionici: Generički korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda artikala

• Opis osnovnog tijeka: > 1. Korisnik mora stisnuti na logo oglašivača > 2. Prikazuje se pregled galerije oglašivača > 3. Nakon pregleda, vraća korisnika na prikaz pregleda artikala

• Opis mogućih odstupanja: > Nema

UC-025 - Odabir oglasa

- Glavni sudionik: Generički korisnik
- Cilj: Prikazuje detalje odabranog oglasa u novoj kartici
- Sudionici: Generički korisnik, Baza podataka
- Preduvjet: Korisnik mora biti u kartici pregleda galerije oglašivača

• Opis osnovnog tijeka: > 1. Korisnik želi pregledati dodatne informacije o oglasu i stišće na oglas > 2. Otvara se kartica s dodatnim informacijama > 3. Nakon pregleda informacija korisnik mora stisnuti na gumb “X” kako bi izašao iz kartice i vratio se gdje se prije nalazio

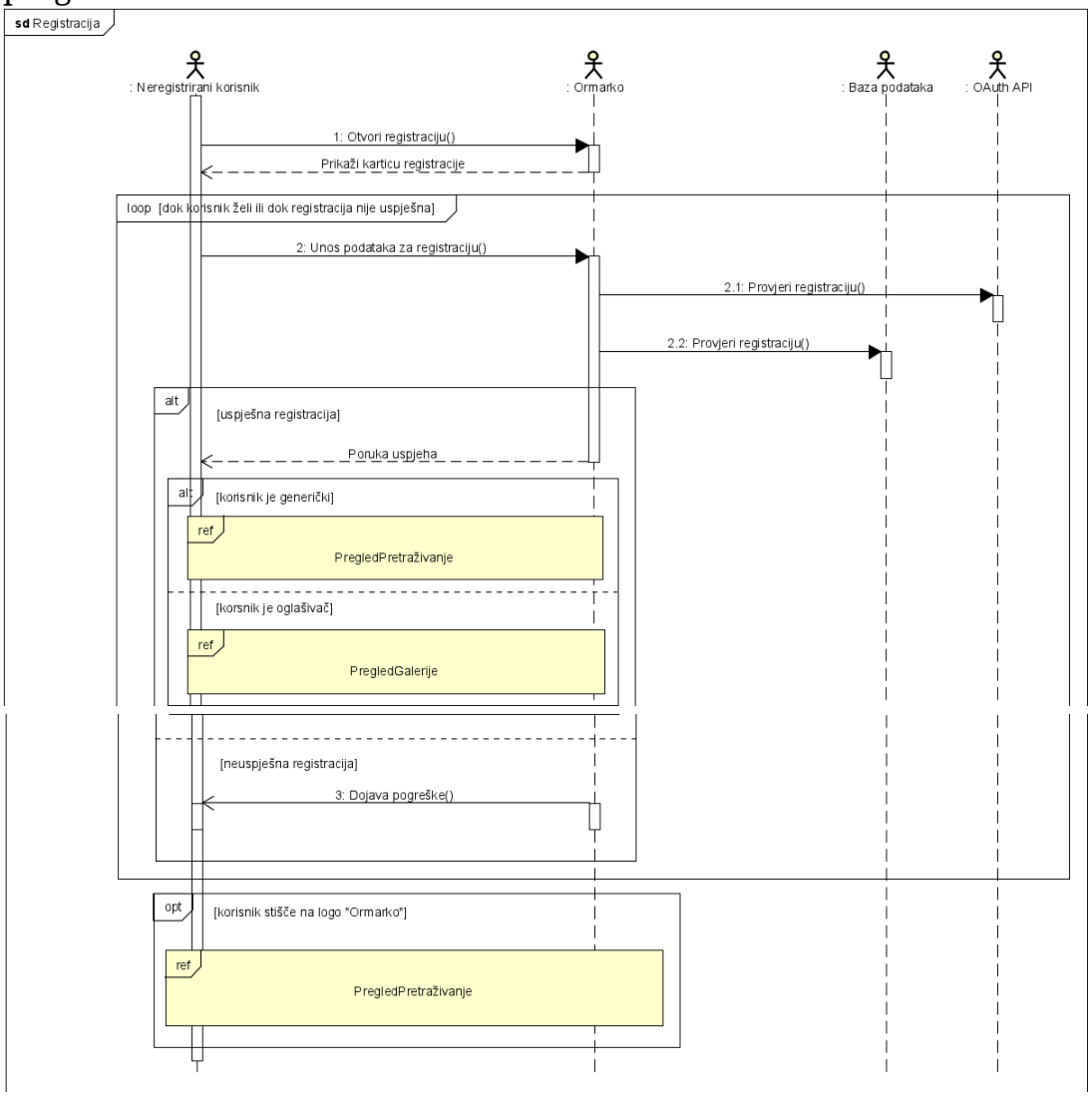
• Opis mogućih odstupanja: > Nema

Sekvencijski dijagrami

Registracija

Otvori se kartica registracije. Kad korisnik unese podatke, oni se provjeravaju pomoću OAuth API-ja i baze podataka. Ako već postoji račun s istim podacima, korisnika se odbija i može pokušati ponovno. Ako ne postoji račun s istim podacima, korisniku se prikazuje poruka uspjeha i u bazi se pamte njegovi podaci. Korisnika se automatski prijavljuje. Ako je korisnik generički vraća se na pregled artikala, ako je korisnik oglašivač prikazuje mu se njegova galerija. Korisnik prije registracije može stisnuti na logo “Ormarko” kako bi se vratio na

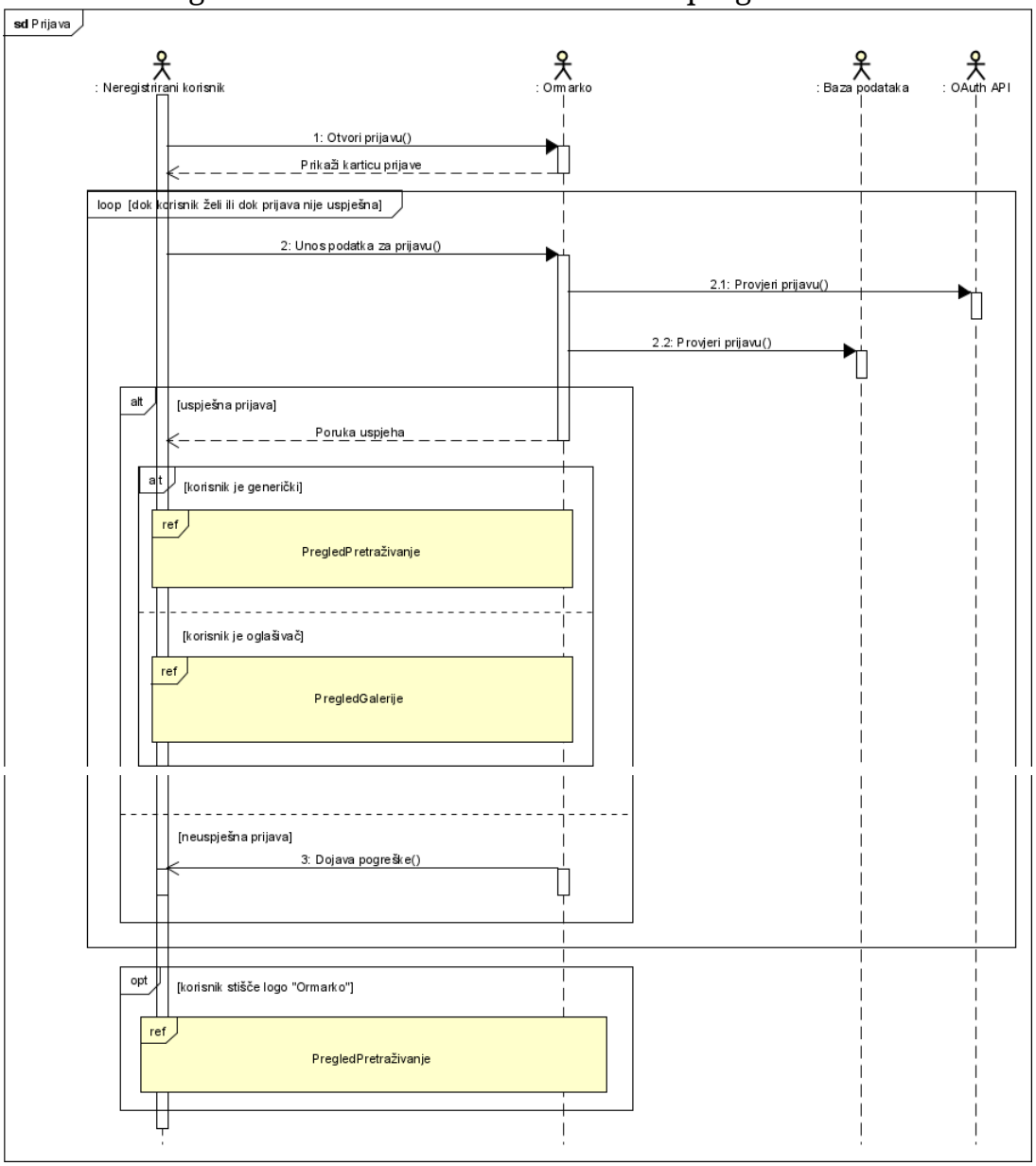
pregled artikala.



Prijava

Otvori se kartica prijave. Kad korisnik unese podatke, oni se provjeravaju pomoću OAuth API-ja i baze podataka. Ako ne postoji račun s unesenim podacima, korisnika se odbija i može pokušati ponovno. Ako postoji račun s unesenim podacima, korisniku se prikazuje poruka uspjeha i u bazi se pamte njegovi podaci. Ako je korisnik generički vraća ga se na pregled artikala, ako je korisnik oglašivač prikazuje mu se njegova galerija. Korisnik prije prijave može

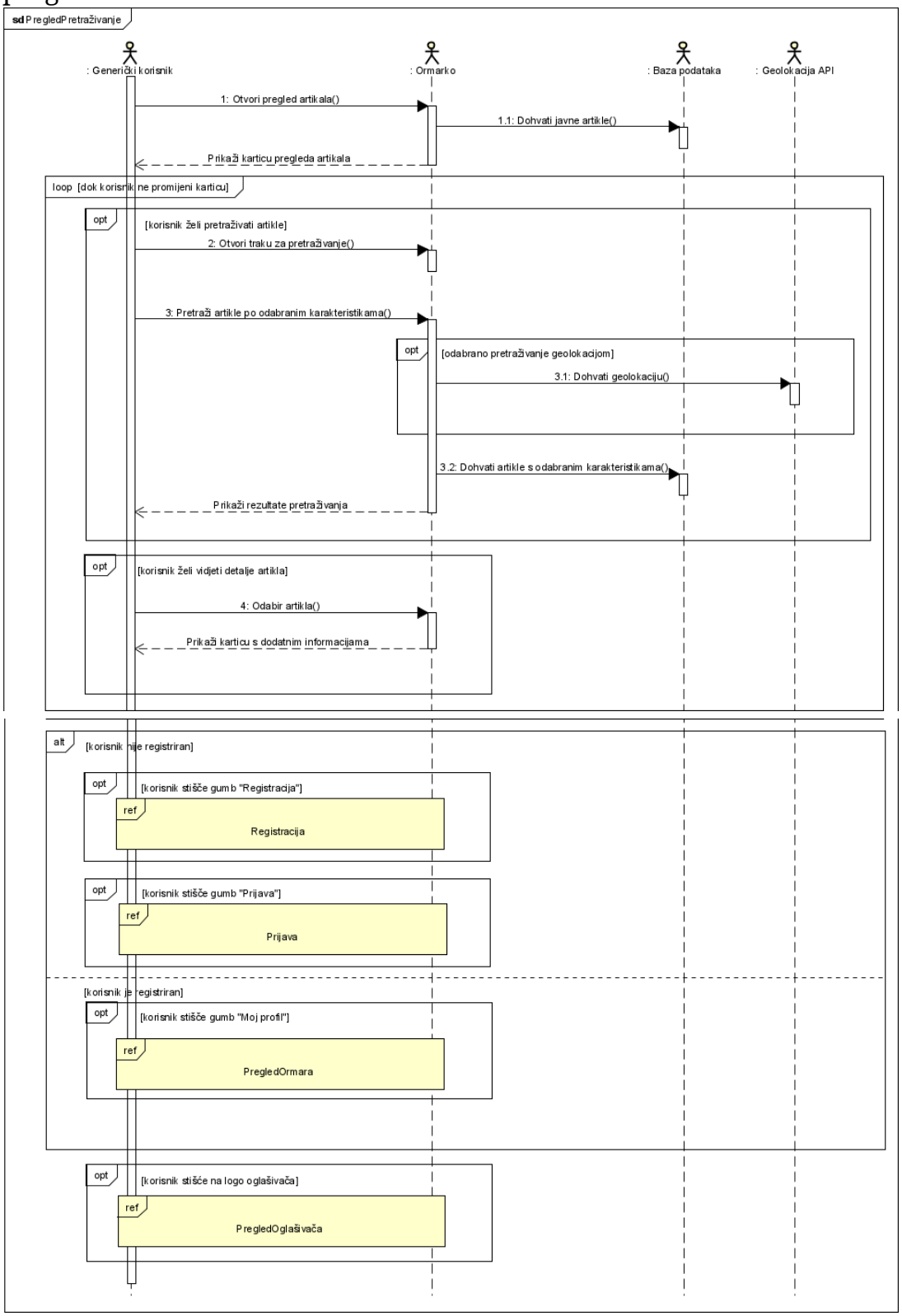
stisnuti na logo “Ormarko” kako bi se vratio na pregled artikala.



Pregled i pretraživanje artikala

Otvori se kartica pregleda artikala. Korisnik može pretraživati artikle prema nazivu, karakteristikama artikala i geolokaciji. Kada stisne na traku za pretraživanje prikazat će se dio u kojem može odabrati filtre za pretraživanje. Ako je odabrao pretraživanje prema geolokaciji, geolokaciju ćemo dobiti slanjem upita na API geolokacije, time možemo filtrirati artikle u blizini korisnika. Nakon toga filtriramo artikle po odabranim karakteristikama i nazivu te prikažemo rezultate korisniku. Korisnik uvijek može snistuni na artikl kako bi saznao više informacija. Ako korisnik nije registriran, može stisnuti gumbe “Registracija” i “Prijava” kako bi promijenio karticu na registraciju ili prijavu. Ako je korisnik registriran, može stisnuti gumb “Moj profil” kako bi promijenio karticu na pregled profila i vlastitih ormara.

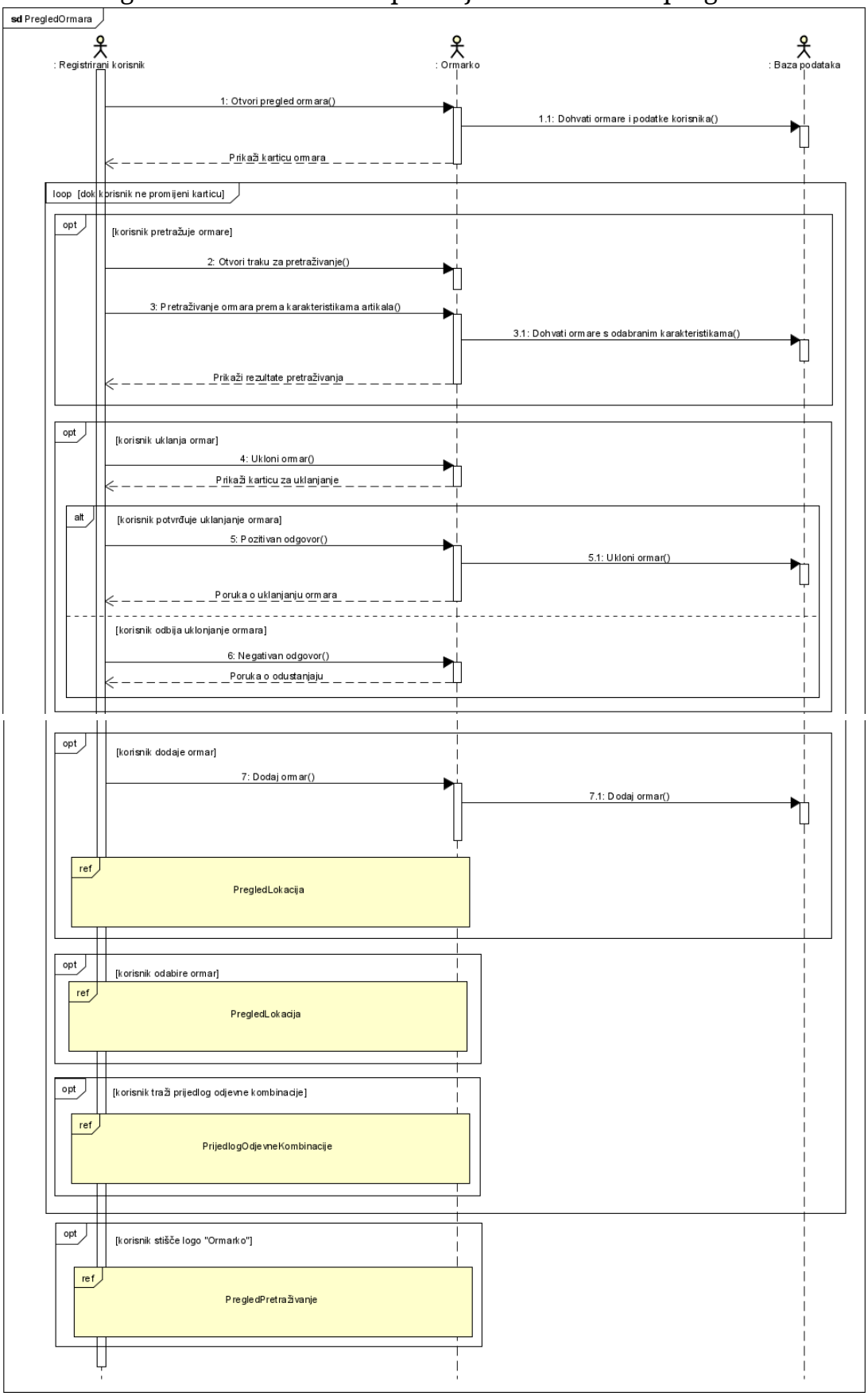
Korisnik može stisnuti logo “Ormarko” kako bi promijenio karticu na pregled artikala.



Pregled vlastitih ormara

Otvori se kartica pregleda vlastitih ormara. Korisniku se prikazuju podaci profila i vlastiti ormari. Korisnik može pretraživati ormare prema karakteristikama artikala koji se nalaze u njima. Korisnik može ukloniti ormar tako da stisne na ikonu koša za smeće na odabranom ormaru, prije nego što se ukloni ormar prikaže se kartica u kojoj korisnik mora potvrditi uklanjanje ormara. Korisnik može dodati ormar tako da stinse ikonu plusa, kada doda ormar odmah se otvara kartica pregleda lokacija ormara u kojoj korisnik može mijenjati strukturu ormara. Korisnik može istu karticu otvoriti tako da stisne na ormar čiji sadržaj želi vidjeti. Korisnik ima mogućnost prijedloga odjevne kombinacije tako da stinse gumb “Prijedlog Kombinacije”, tada se otvara kartica za prijedlog odjevne kombinacije. Korisnik može

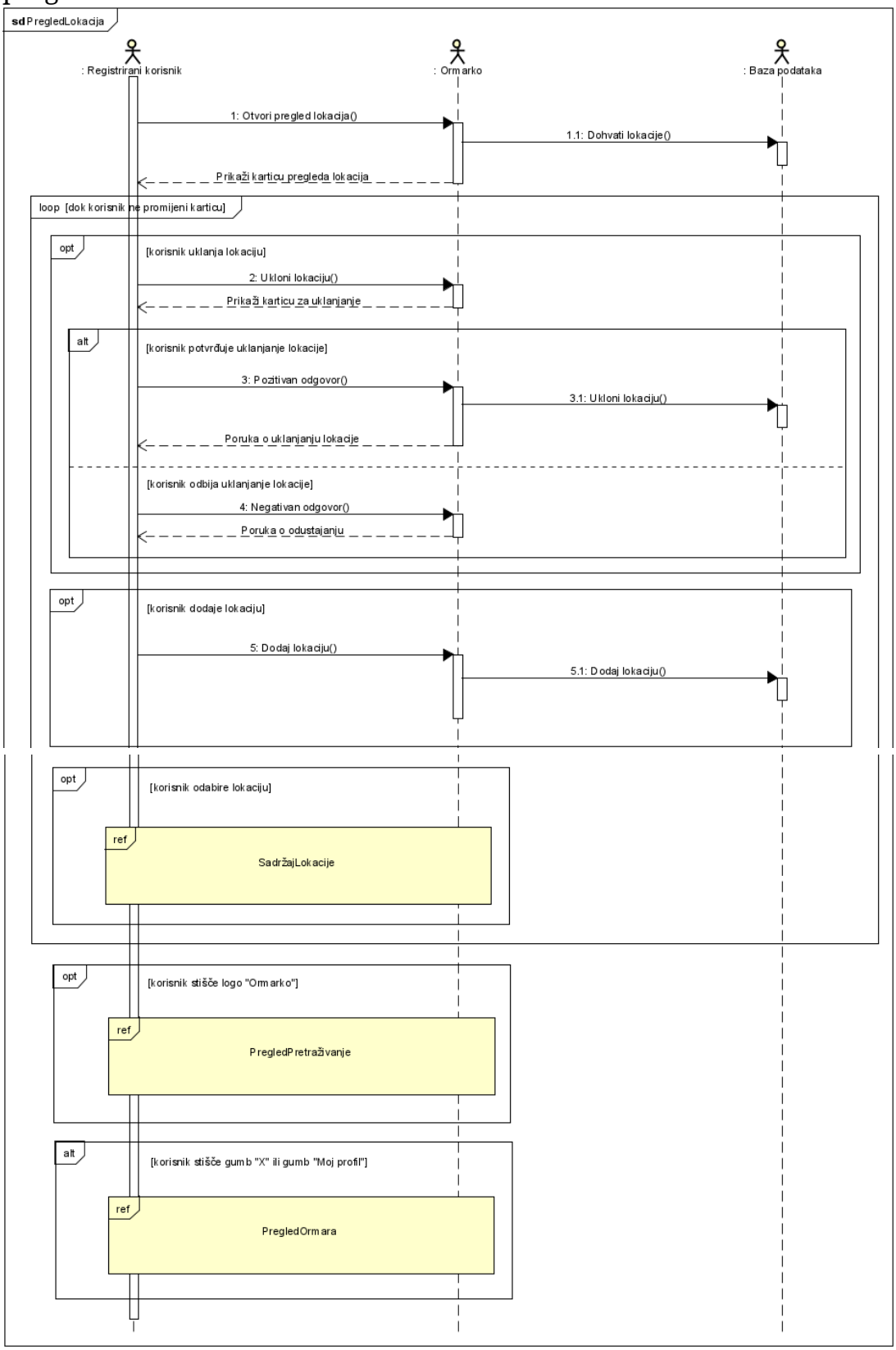
stisnuti logo “Ormarko” kako bi promijenio karticu na pregled artikala.



Pregled lokacija ormara

Otvori se kartica pregleda lokacija ormara. Korisnik može mijenjati strukturu ormara tako da dodaje i uklanja lokacije iz ormara. Korisnik može ukloniti lokaciju tako da stinse ikonu koša za smeće na lokaciji, prije nego što se ukloni lokaciju prikaže se kartica u kojoj korisnik mora potvrditi uklanjanje lokacije. Korisnik može dodati lokaciju tako da stinse ikonu plusa, kada doda lokaciju odmah se otvara kartica pregleda sadržaja lokacije u kojoj korisnik može vidjeti i mijenjati sadržaj lokacije. Korisnik može istu karticu otvoriti tako da stisne na lokaciju čiji sadržaj želi vidjeti. Korisnik može stisnuti gumb “Moj profil” ili gumb “X” kako bi se vratio na pregled vlastitih ormara. Korisnik može stisnuti logo “Ormarko” kako bi promijenio karticu na

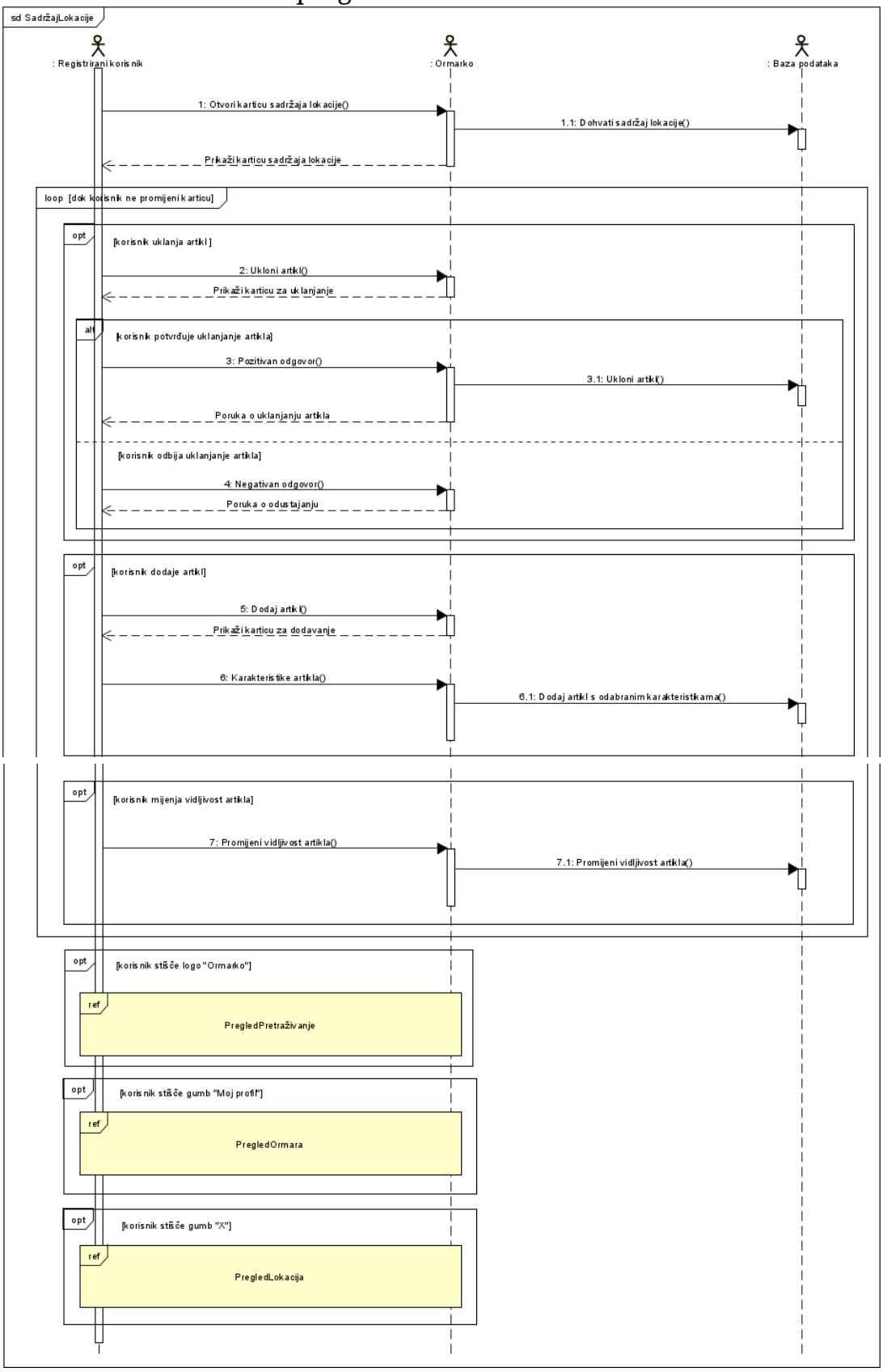
pregled artikala.



Sadržaj lokacije

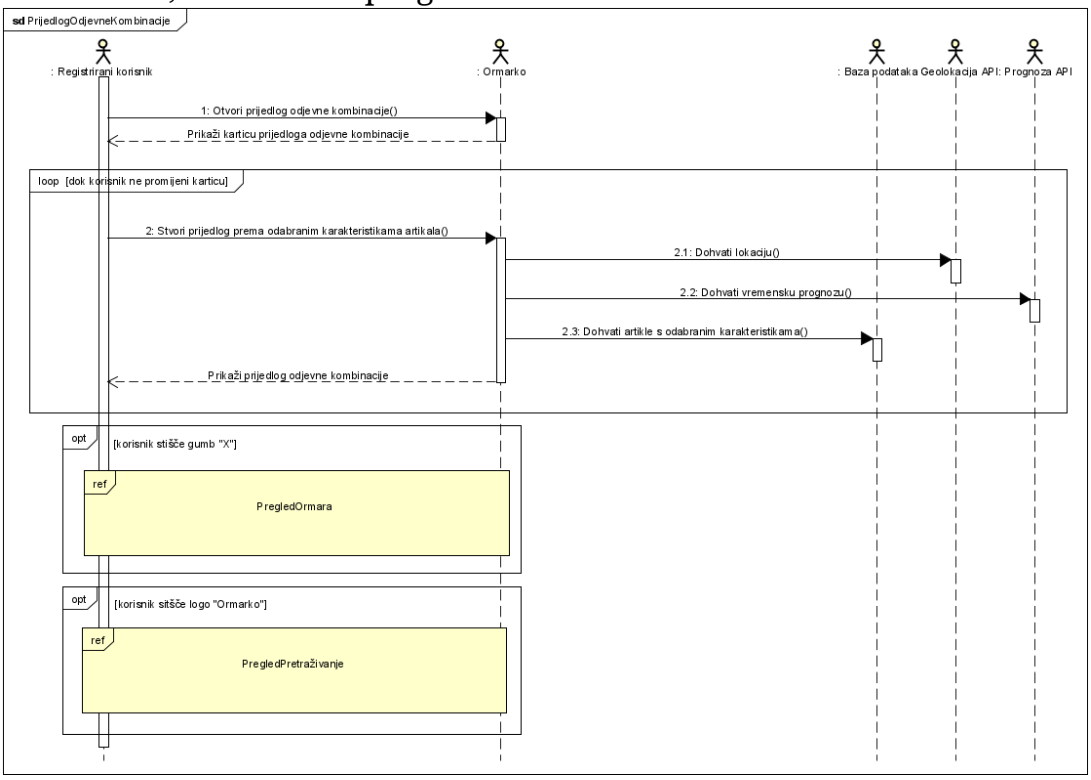
Otvori se kartica pregleda sadržaja lokacije. Korisnik može mijenjati sadržaj lokacije tako da dodaje i uklanja artikle iz lokacije. Korisnik može ukloniti artikl tako da stinse ikonu koša za smeće na artiklu, prije nego što se ukloni artikl prikaže se kartica u kojoj korisnik mora potvrditi uklanjanje artikla. Korisnik može dodati artikl tako da stinse ikonu plusa, kada doda artikl odmah se otvara kartica karakteristika artikla u kojoj korisnik mora unijeti karakteristike i podatke o svom artiklu. Korisnik može vidjeti i mijenjati vidljivost svojih artikala, tako da stinse na ikonu oka. Ako je oko prekiženo onda artikl nije javno vidljiv, ako oko nije prekriženo onda je artikl vidljiv svim korisnicima. Korisnik može stisnuti gumb “Moj profil” kako bi se vratio na pregled vlastitih ormara. Korisnik može stisnuti logo “Ormarko” kako bi promijenio karticu na pregled artikala. Korisnik može stisnuti gumb

“X” kako bi se vratio na pregled ormara.



Prijedlog odjevne kombinacije

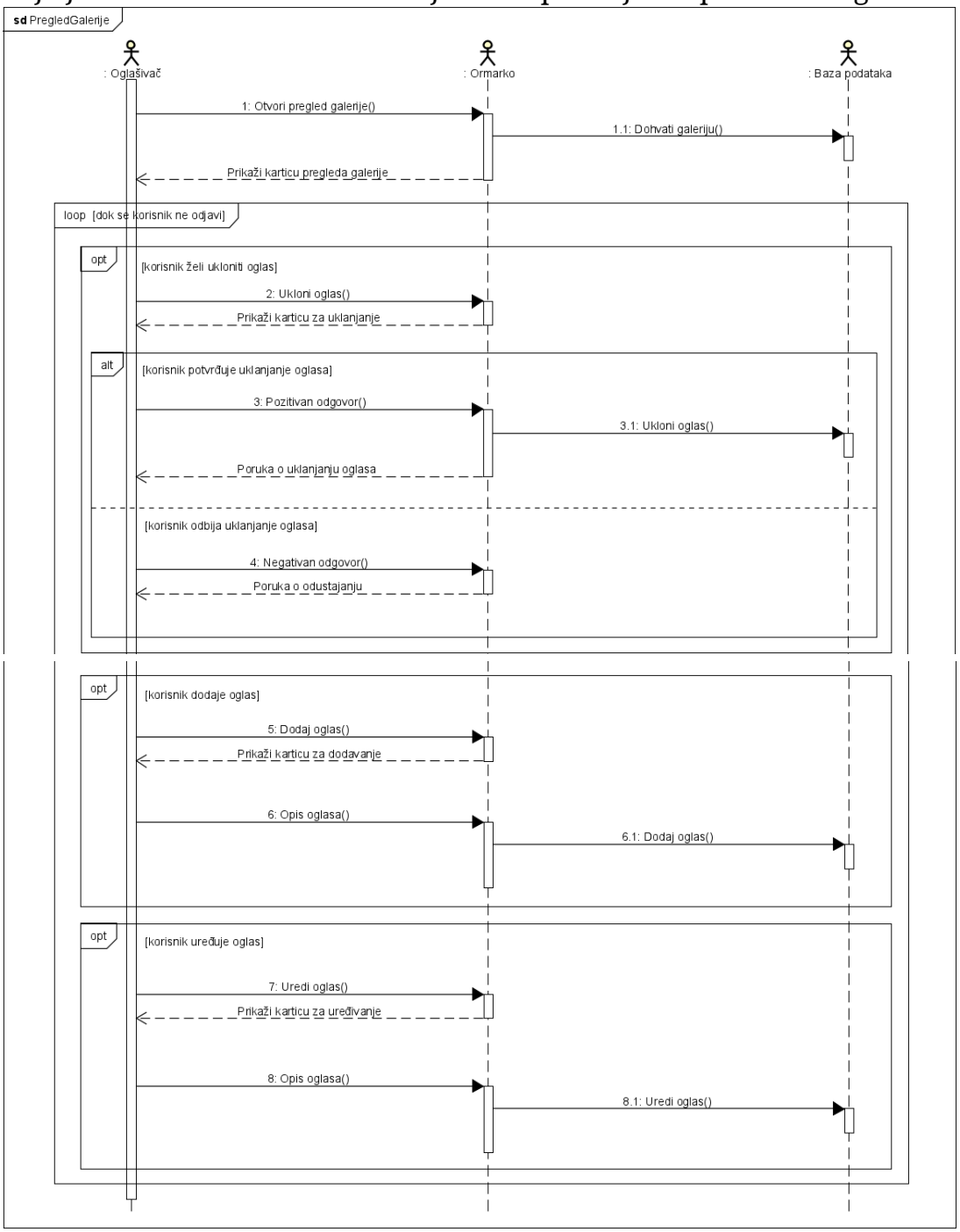
Otvori se kartica prijedloga odjevne kombinacije. Korisnik unosi karakteristike artikala koje želi u odjevnoj kombinaciji. Prije stvaranja kombinacije, preko API-ja geolokacije dobivamo geolokaciju korisnika. Nju koristimo za određivanje vremenske prognoze preko API-ja prognoze. Tu informaciju i odabrane karakteristike artikala koristimo za stvaranje odjevne kombinacije. Korisnik može ponavljati generiranje odjevne kombinacije dok nije zadovoljan. Ako stisne na gumb “X”, vraća ga se na pregled ormara. Ako stisne na logo “Ormarko”, vraća se na pregled artikala.



Pregled galerije

Otvori se kartica pregleda galerije. Oglašivač jedino može biti u ovoj kartici. Ako želi ukloniti oglas, mora dodatno potvrditi u kartici za uklanjanje. Ako želi dodati oglas, mora unijeti podatke o oglasu u karticu za dodavanje. Ako želi urediti oglas, u kartici za uređivanje

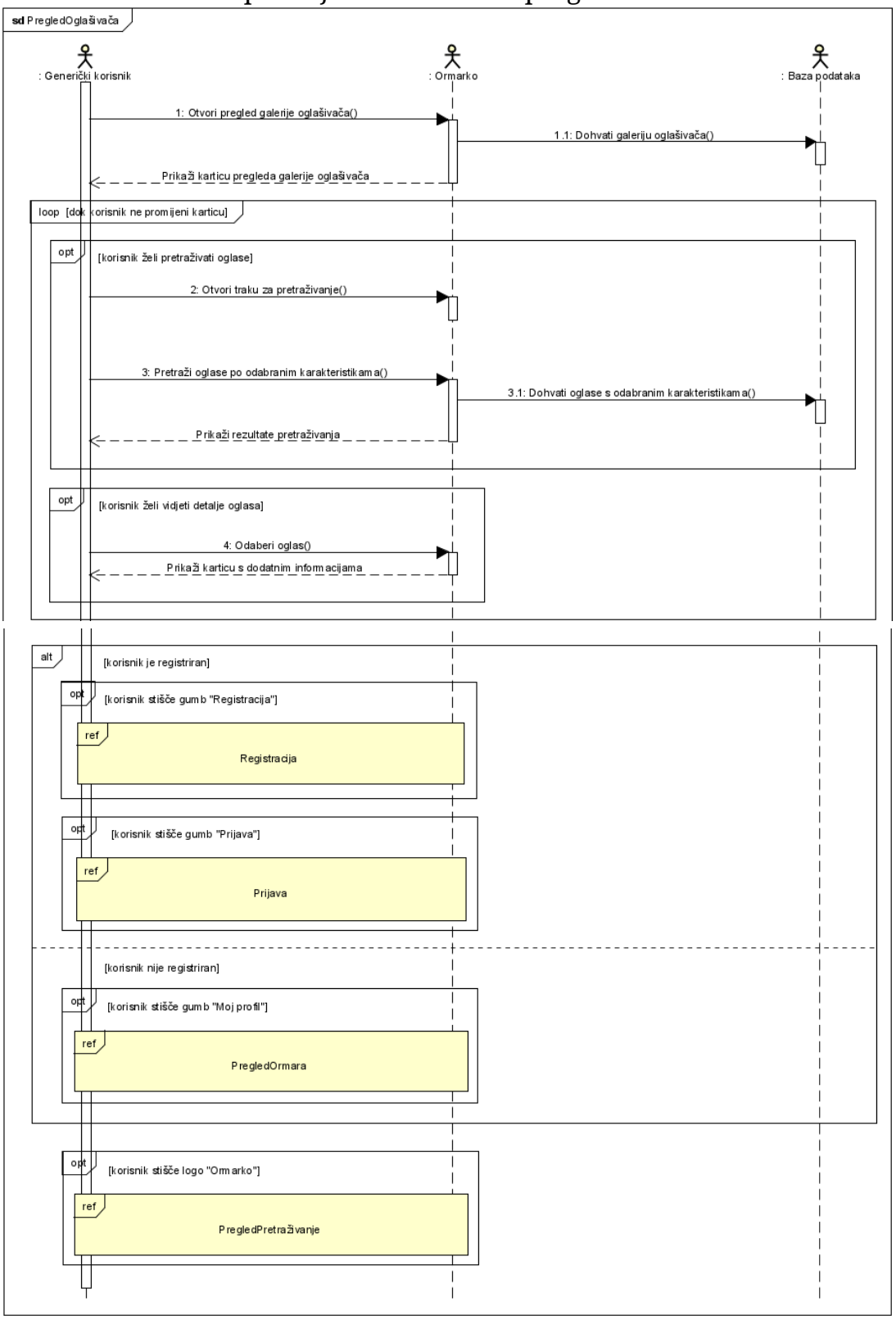
koja je slična kartici za dodavanje može promijeniti podatke o oglasu.



Pregled galerije oglašivača

Otvori se kartica za pregled galerije oglašivača. Korisnik može kliknuti na oglase kako bi vidio dodatne informacije o oglasu. Korisnik može pretraživati oglase po nazivu i karakteristikama artikla. Ako korisnik nije registriran, može stisnuti gumbe “Registracija” i “Prijava” kako bi promijenio karticu na registraciju ili prijavu. Ako je korisnik registriran, može stisnuti gumb “Moj profil” kako bi promijenio karticu na pregled profila i vlastitih ormara. Korisnik može stisnuti logo

“Ormarko” kako bi promijenio karticu na pregled artikala.



Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

ID zahtjeva	ID obrazaca uporabe
F-001	UC-004, UC-006, UC-007
F-002	UC-006, UC-008
F-003	UC-005
F-004	UC-001, UC-002, UC-003
F-005	UC-010, UC-011, UC-014
F-006	UC-016, UC-017
F-007	UC-009, UC-006, UC-007
F-008	UC-019
F-009	UC-013, UC-007
F-010	UC-012, UC-015, UC-016, UC-018
F-011	UC-024, UC-025
F-012	UC-020, UC-021, UC-022, UC-023

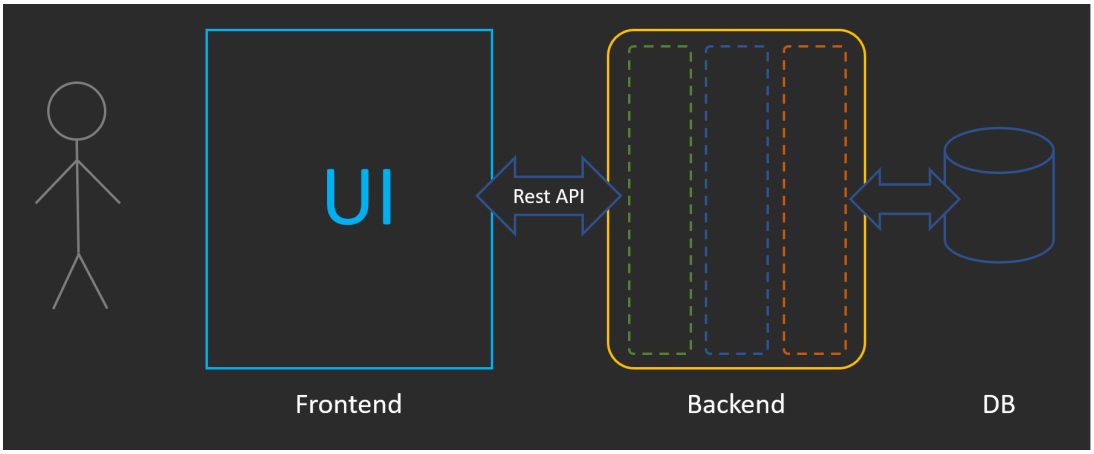
Arhitektura sustava

Opis arhitekture

Stil arhitekture

Za naš sustav odabrali smo klijent-poslužiteljsku arhitekturu s jasnim odvojenjem frontend i backend komponenti. Frontend je implementiran koristeći React biblioteku, dok je backend razvijen u Javi koristeći Spring Boot framework. Ovaj arhitektonski stil odabran je zbog svoje modularnosti i fleksibilnosti. Odvajanje korisničkog sučelja od poslovne logike omogućava neovisni razvoj i održavanje svakog sloja, što ubrzava razvojni proces i olakšava implementaciju novih funkcionalnosti. ### Podsustavi Glavni podsustavi našeg sustava su: * **Frontend aplikacija:** Implementirana u Reactu, odgovorna je za prikaz korisničkog sučelja i interakciju s korisnikom. * **Backend**

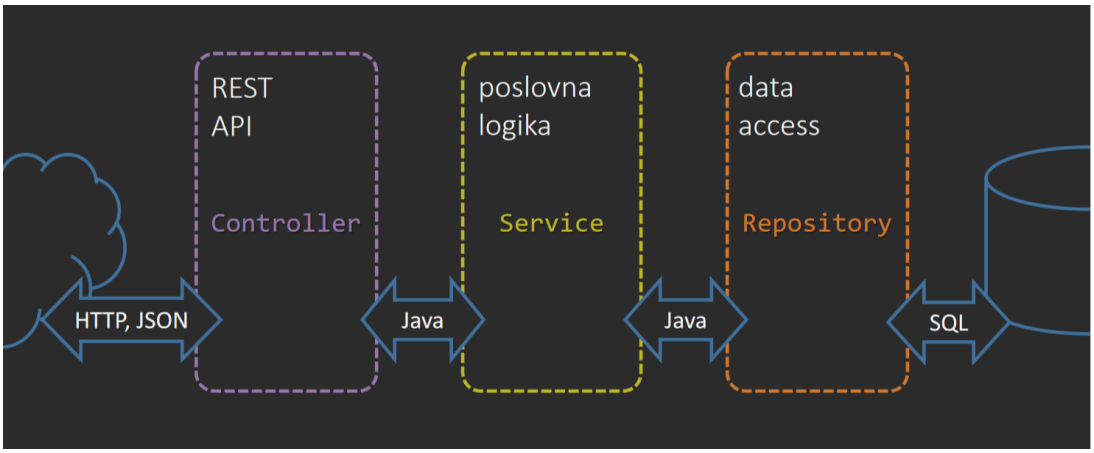
aplikacija: Implementirana u Javi koristeći Spring Boot, zadužena je za poslovnu logiku, upravljanje podacima i sigurnost. * **Baza podataka:** Koristimo PostgreSQL bazu podataka za pohranu svih podataka.



Slika 1. Skica podsustava ### Preslikavanje na radnu platformu Arhitektura je implementirana koristeći hibridni pristup: * **Lokalni razvoj:** Tijekom razvoja, frontend aplikacija pokreće se lokalno na localhost:5173, a backend na localhost:8080. * **Cloud infrastruktura:** Baza podataka hostirana je na AWS poslužitelju, što omogućava veću dostupnost i skalabilnost te olakšava buduću migraciju cijelog sustava u cloud okruženje.

Ovaj pristup omogućuje razvijateljima brz i jednostavan razvoj te testiranje aplikacije, dok istovremeno koristi prednosti cloud usluga za upravljanje podacima. ### Spremišta podataka Koristimo relacijsku bazu podataka PostgreSQL za pohranu podataka. PostgreSQL je odabran zbog svoje pouzdanosti, performansi i naprednih mogućnosti upravljanja podacima. Upravljanje bazom podataka obavlja se putem PgAdmin sustava. ### Mrežni protokoli Za komunikaciju između komponenti sustava koristimo: * **HTTP protokol:** Za slanje REST API zahtjeva između frontend i backend aplikacija. * **CORS (Cross-Origin Resource Sharing):** Omogućava frontend aplikaciji na localhost:5173 pristup resursima na backendu localhost:8080 ### Globalni upravljački tok Podaci i informacije cirkuliraju kroz sustav zahvaljujući Cross-Origin Resource Sharing (CORS) koji omogućuje frontend aplikaciji da pristupi resursima na backend aplikaciji. Frontend šalje HTTP zahtjeve, koje backend obrađuje. Isključivo backend komunicira sa bazom podataka.
U nastavku je opisan tok podataka na primjeru registracije korisnika: > **Backend:** Kontroler RegistrationController koristi @PostMapping na ruti /api/signup/user za kreiranje novog korisnika > Lozinka korisnika kodira se pomoću PasswordEncoder prije spremanja u bazu podataka > Prilikom registracije, korisnik se automatski preusmjerava na profile

stranicu
> **Frontend:** U Register.jsx komponenti koristi se handleSubmit funkcija koja šalje POST zahtjev na /api/signup/user s podacima za registraciju > U slučaju uspješne registracije, korisnik se preusmjerava na stranicu profila, a u slučaju greške prikazuje se poruka o neuspjehu registracije ### Sklopovskoprogramski zahtjevi * **Backend** > Java 23
> Maven za upravljanje ovisnostima
> Springboot framevork * **Frontend:** > Node.js i npm za pokretanje React aplikacije * **Baza podataka:** > PostgreSQL server
> Pristup AWS infrastrukturi za bazu podataka ## Obrazloženje odabira arhitekture Pri odabiru arhitekture vodili smo se sljedećim principima: * **Visoka kohezija**: Svaka komponenta sustava ima jasno definirane odgovornosti. Frontend upravlja korisničkim sučeljem, backend poslovnom logikom, a baza podataka pohranom podataka. * **Niska povezanost**: Odvajanje frontend i backend slojeva smanjuje međuovisnost, omogućavajući neovisni razvoj i održavanje. * **Fleksibilnost**: Modularna arhitektura omogućava lako proširenje funkcionalnosti i prilagodbu sustava budućim zahtjevima. * **Sigurnost**: Korištenje Spring Security frameworka za autentifikaciju i autorizaciju korisnika povećava sigurnost sustava. ## Organizacija sustava na visokoj razini ### Klijent-poslužitelj **Klijent** odnosno Frontend React aplikacija pruža korisničko sučelje i šalje zahtjeve backendu putem HTTP protokola. Poslužitelj odnosno Backend Spring Boot aplikacija obrađuje zahtjeve klijensta te upravlja poslovnom logikom i komunikacijom s bazom podataka ### Baza podataka Koristimo **PostgreSQL** bazu podataka koja je povezana s backend aplikacijom putem JPA (Java Persistence API) i Spring Data JPA ### Grafičko sučelje Korisničko sučelje je responzivna web aplikacija. React frontend aplikacija pruža interaktivno i responzivno korisničko sučelje. ## Organizacija aplikacije ### Frontend i backend slojevi Na razini frontenda, komponente su organizirane po funkcionalnosti te on komunicira s backendom putem REST API pozova. Backend ima Controller-Service- Repository stil. Sloj Repository je najniži sloj i komunicira izravno s bazom podataka. Svaki repositorij bavi se svojom odredenom tablicom odnosno entitetom u bazi podataka i niti jednom drugom. Sloj Service je srednji sloj koji se bavi poslovnom logikom te uslugama. Sadrži funkcije poput pronalaženja elemenata u tablici po posebnim kriterijima. Sloj Controller bavi se zahtjevima klijenata odnosno HTTP zahtjevima. Također je prisutan i sloj model koji sadrži entitete koji predstavljaju podatke.



Slika 2. Skica CSR stila

Baza podataka

Baza podataka na ovom projektu implementirana je koristeći PostgreSQL programski jezik te PgAdmin sustav. Pristup bazi organiziran je preko AWS poslužitelja. Baza podataka sastoji se od šest tablica, od kojih su u skupinama povezane četiri i dvije. Prva skupina od četiri tablice odnosi se na registrirane korisnike te njihove ormare, lokacije u ormarima i artikle. Druga skupina od dvije tablice odnosi se na registrirane oglašivače i njihove artikle.

Opis tablica

users (Korisnici)

Atribut	Tip podatka	Opis varijable
username	VARCHAR(20)	Primarni ključ
pass	VARCHAR(200)	Ne-null vrijednost
city	VARCHAR(20)	Ne-null vrijednost
country	VARCHAR(20)	Ne-null vrijednost
e_mail	VARCHAR(50)	Ne-null vrijednost, Jedinstvena vrijednost
googleOauth	BOOL	Zadana vrijednost “FALSE”

marketers (Oglašivači)

Atribut	Tip podatka	Opis varijable
username	VARCHAR(20)	Primarni ključ
pass	VARCHAR(200)	Ne-null vrijednost
e_mail	VARCHAR(50)	Ne-null vrijednost, Jedinstvena vrijednost
logo	BYTEA	Ne-null vrijednost, Jedinstvena vrijednost

closets (Ormari)

Atribut	Tip podatka	Opis varijable
closet_id	SERIAL	Primarni ključ
closet_owner	VARCHAR(200)	Strani ključ, Ne-null vrijednost

locations (Lokacije)

Atribut	Tip podatka	Opis varijable
location_id	SERIAL	Primarni ključ
closet_id	INT	Strani ključ, Ne-null vrijednost
location_type	VARCHAR(20)	Ne-null vrijednost

articles_user (Artikli korisnika)

Atribut	Tip podatka	Opis varijable
article_id	SERIAL	Primarni ključ
location_id	INT	Strani ključ, Ne-null vrijednost
sharing	BOOL	Zadana vrijednost “FALSE”
title	VARCHAR(50)	Ne-null vrijednost
img	BYTEA	Ne-null vrijednost
category	VARCHAR(20)	Ne-null vrijednost
season	VARCHAR(20)	Ne-null vrijednost
openness	VARCHAR(20)	/
how_casual	VARCHAR(20)	Ne-null vrijednost
main_color	VARCHAR(20)	Ne-null vrijednost

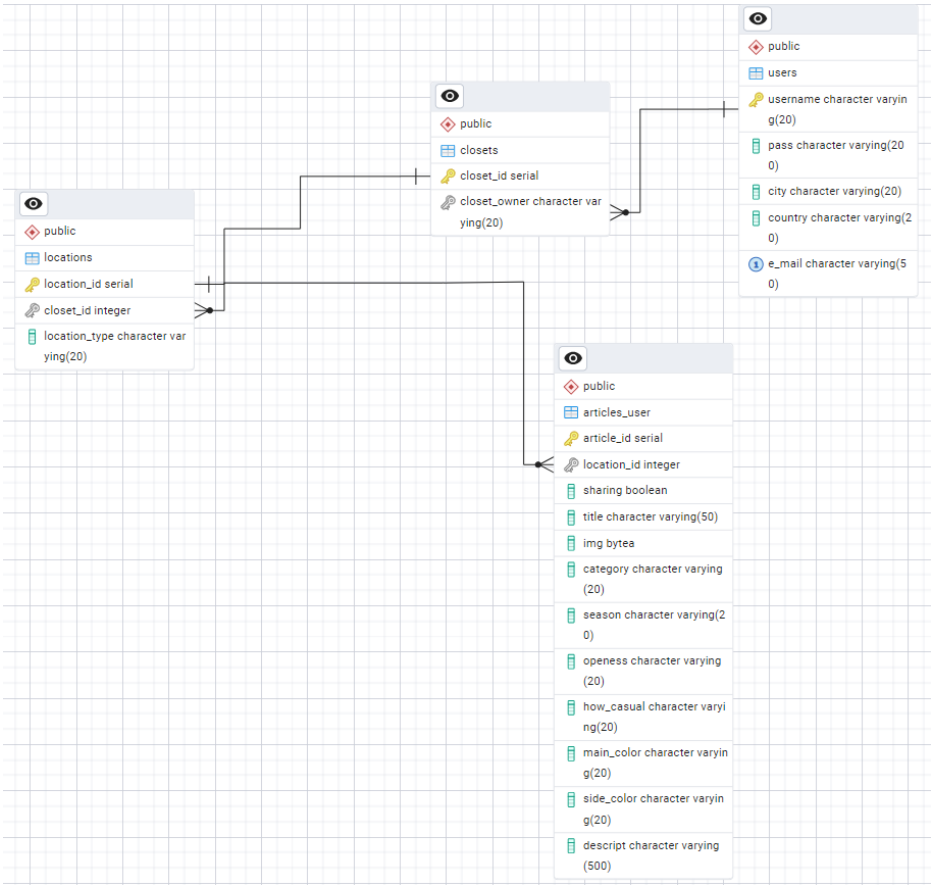
Atribut	Tip podatka	Opis varijable
side_color	VARCHAR(20)	Ne-null vrijednost
descript	VARCHAR(500)	/

articles_marketing (Artikli oglašivača)

Atribut	Tip podatka	Opis varijable
article_id	SERIAL	Primarni ključ
article_marketer	VARCHAR(20)	Strani ključ, Ne-null vrijednost
title	VARCHAR(50)	Ne-null vrijednost
category	VARCHAR(20)	Ne-null vrijednost
img	BYTEA	Ne-null vrijednost
price	REAL	Ne-null vrijednost

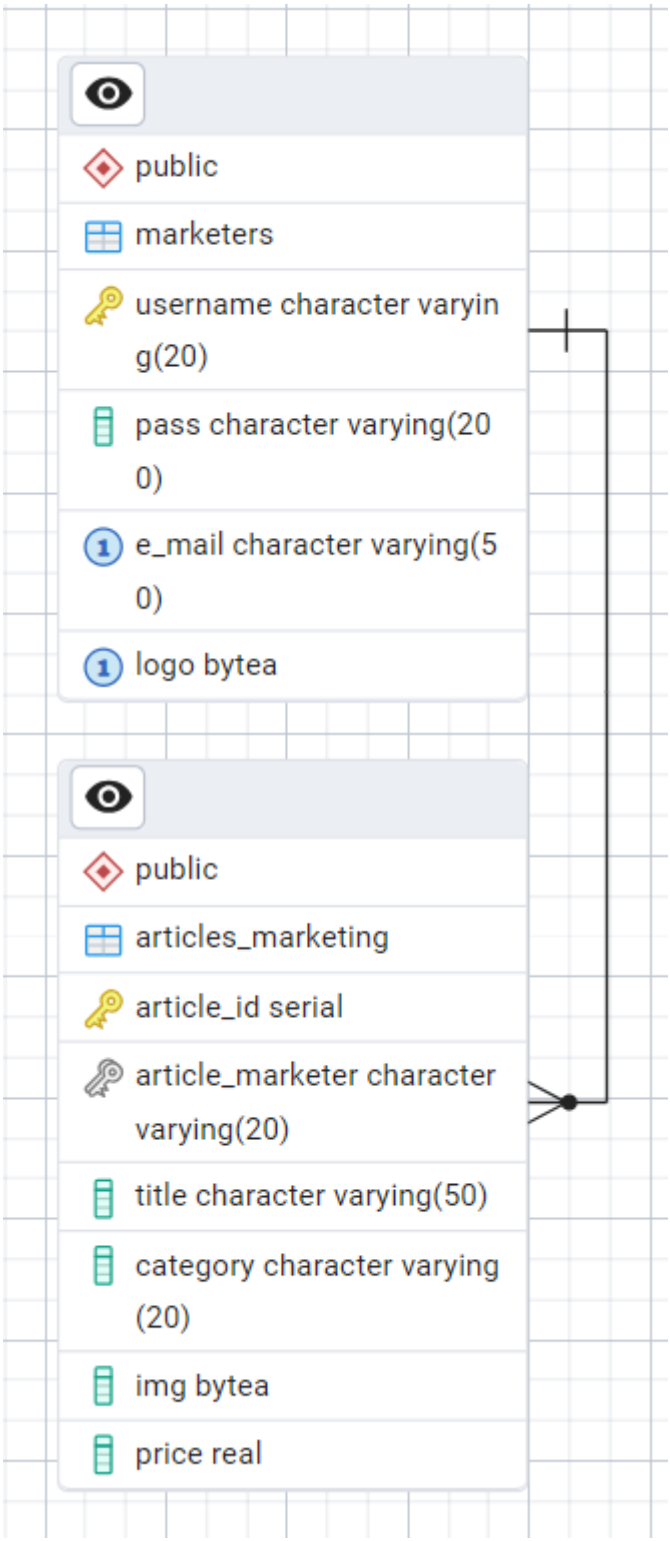
Prilikom brisanja zavisnih entiteta (na temelju stranog ključa) dolazi do kaskadnog brisanja entiteta (users -> closets -> locations -> articles_user i marketers -> articles_marketing).

Dijagram baze podataka



Schema 1

Prvi dio sheme baze podataka prikazuje odnose između tablica “users”, “closets”, “locations” i “articles_user”. Artikli su slabi entitet u odnosu na lokacije u ormaru, koje su slabi entitet u odnosu na ormare, koji su slabi entitet u odnosu na korisnika. Korisnik je regularni entitet. Veze korisnik-ormari, ormari-lokacije i lokacije-artikli su oblika 1-0..N.



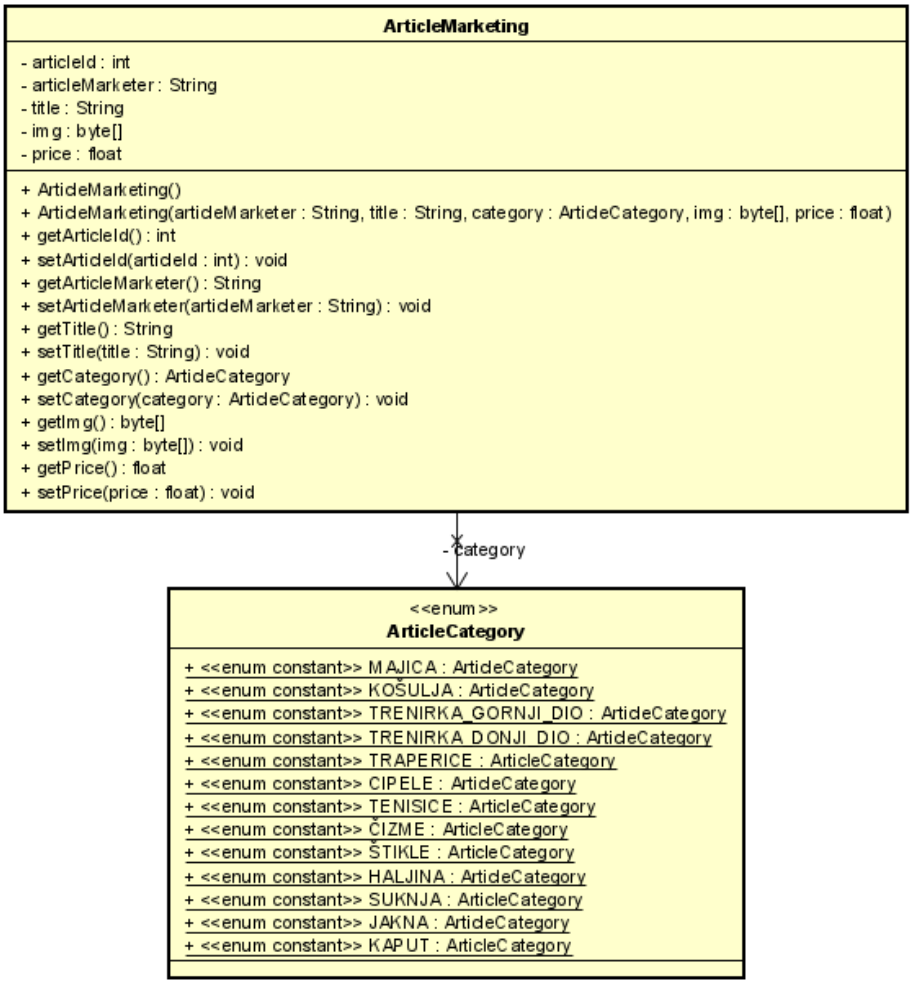
Schema 2

Drugi dio sheme baze podataka prikazuje odnose između tablica “marketers” i “articles_marketing”. Artikli su slabi entitet u odnosu na oglašivače, a oglašivači su regularni entitet. Veza oglašivači-artikli je oblika 1-0..N.

Dijagram razreda

Klase

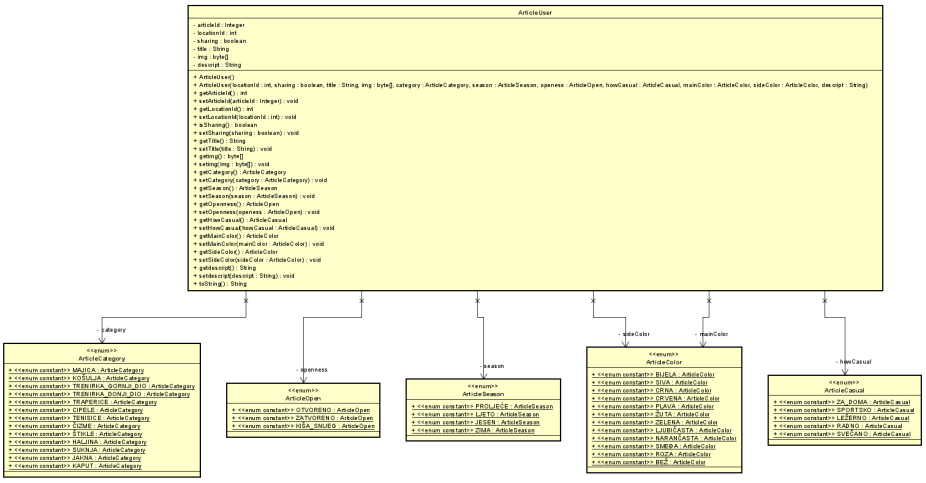
Article Marketing Class



image

Klasa koja prikazuje artikl oglašivača. Artikl oglašivača ima privatne atribute: articleId, articleMarketer, title, img, price i category. Postoje dva konstruktora, jedan deafultni i jedan parametrizirani. U category se postavlja jedna vrijednost iz enumeracije Article Category. Postoje getteri i setteri za sve navedene atribute.

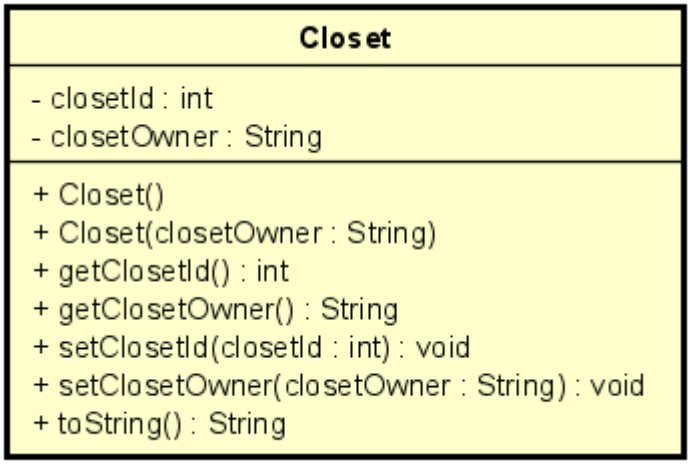
Article User Klasa



image

Klasa koja prikazuje artikl korisnika. Artikl korisnika ima privatne atribute: articleId, locationId, sharing, title, img, descript, category, season, howCasual, mainColor, sideColor i openness. Postoje dva konstruktora, jedan deafultni i jedan parametrizirani. U category se postavlja jedna vrijednost iz enumeracije Article Category. U season se postavlja jedna vrijednost iz enumeracije Article Sesaon. U howCasual se postavlja jedna vrijednost iz enumeracije Article Casual. U mainColor i sideColor se postavlja jedna vrijednost iz enumeracije Article Color. U openness se postavlja jedna vrijednost iz enumeracije Article Open. Postoje getteri i setteri za sve navedene atribute.

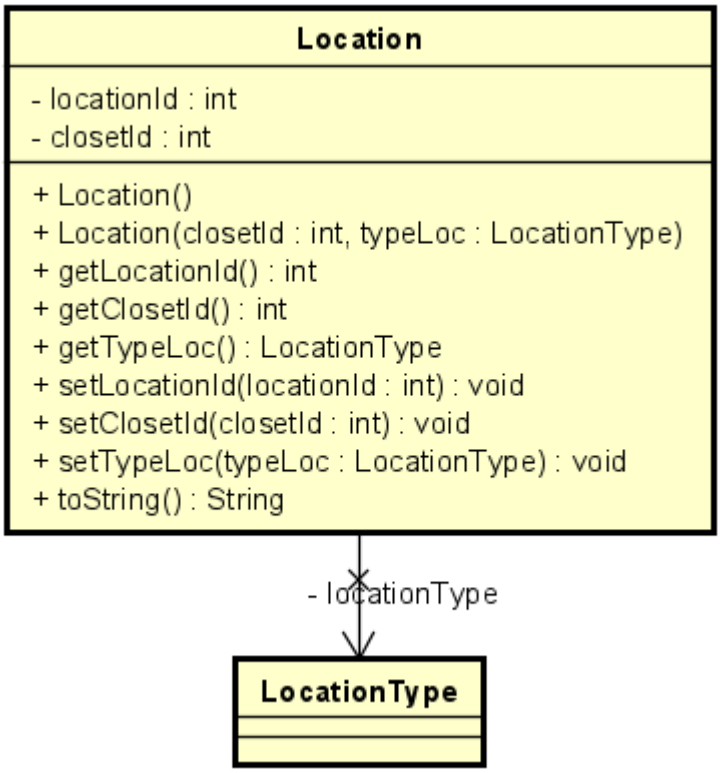
Closet Klasa



image

Klasa koja prikazuje ormar. Ormar ima privatne atribute: closetId i closetOwner. Postoje dva konstruktora, jedan deafultni i jedan parametrizirani, i toString metoda. Postoje getteri i setteri za sve navedene atribute.

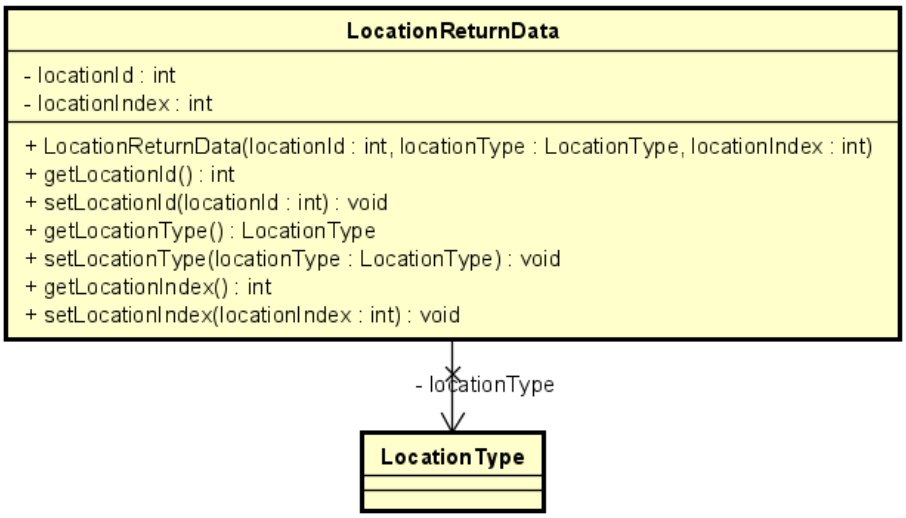
Location Klasa



image

Klasa koja prikazuje lokaciju u ormaru. Lokacija ima privatne atribute: locationId, closetId i typeLoc. Postoje dva konstruktora, jedan deafultni i jedan parametrizirani, i toString metoda. U typeLoc se postavlja jedna vrijednost iz enumeracije Location Type. Postoje getteri i setteri za sve navedene atribute.

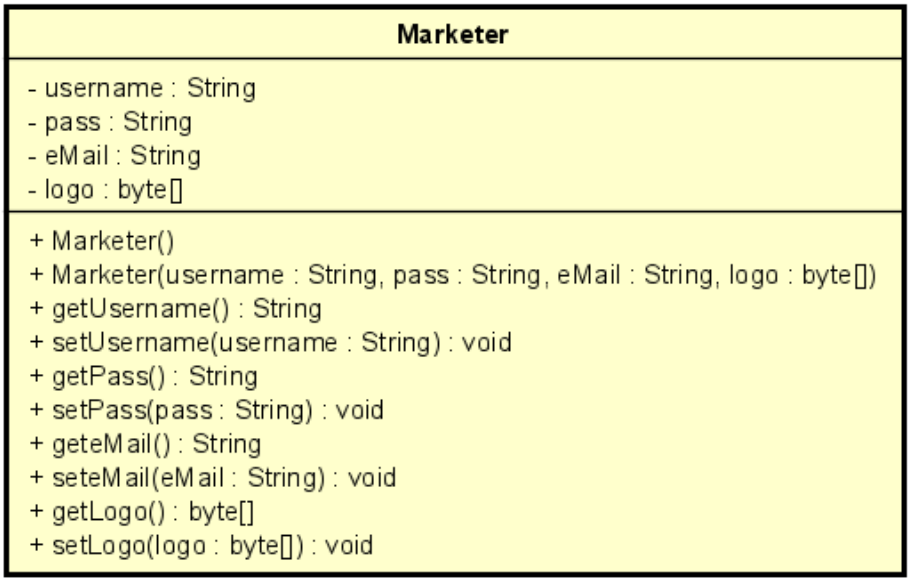
Location Return Data Klasa



image

Klasa koja prikazuje podatke o artiklima u lokaciji. Location Return Data ima privatne atribute: locationId, locationType i locationIndex. Postoji jedan parametriziran konstruktor. Postoje getteri i setteri za sve navedene atribute.

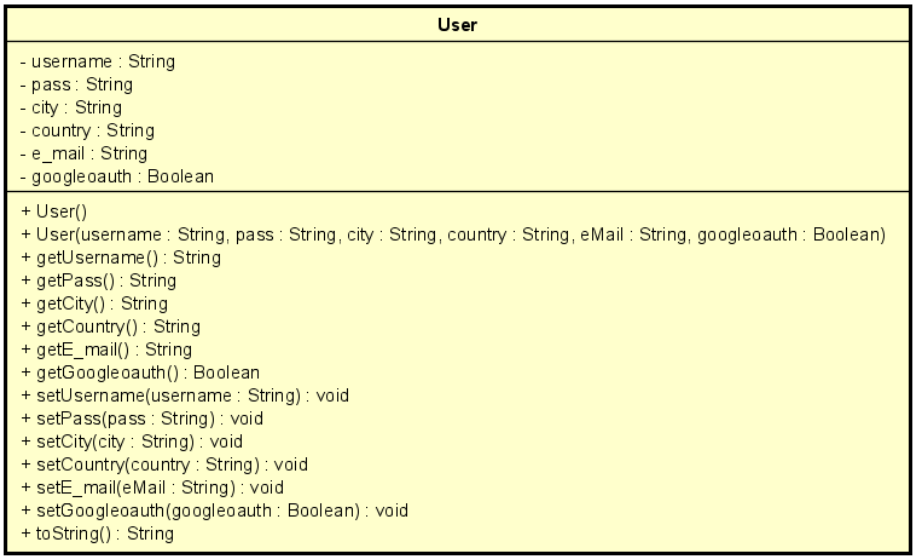
Marketer Klasa



image

Klasa koja prikazuje oglašivača. Oglašivač ima privatne atribute: username, pass i e_mail. Postoje dva konstruktora, jedan deafultni i jedan parametrizirani. Postoje getteri i setteri za sve navedene atribute.

User Klasa

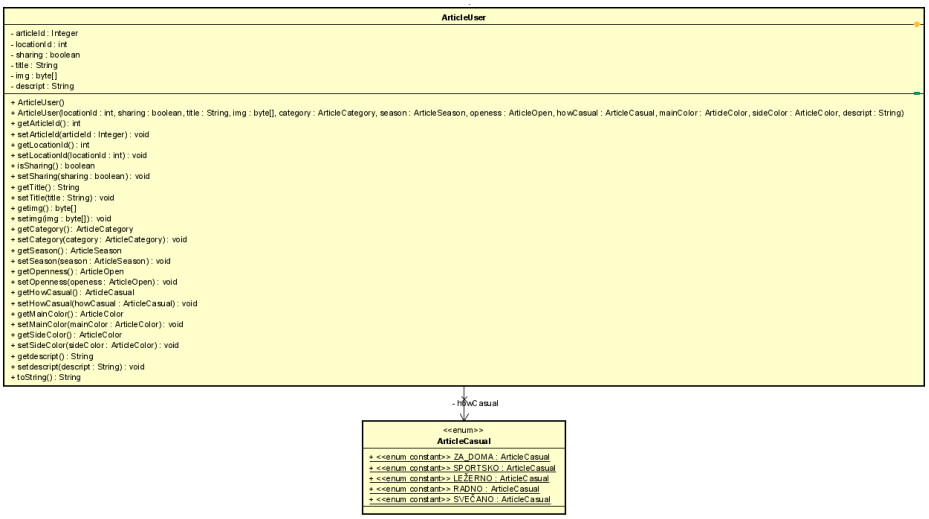


image

Klasa koja prikazuje korisnika. Korisnik ima privatne atribute: username, pass, city, country i e_mail. Postoje dva konstruktora, jedan deafultni i jedan parametrizirani, i toString metoda. Postoje getteri i setteri za sve navedene atribute.

Enumeracije

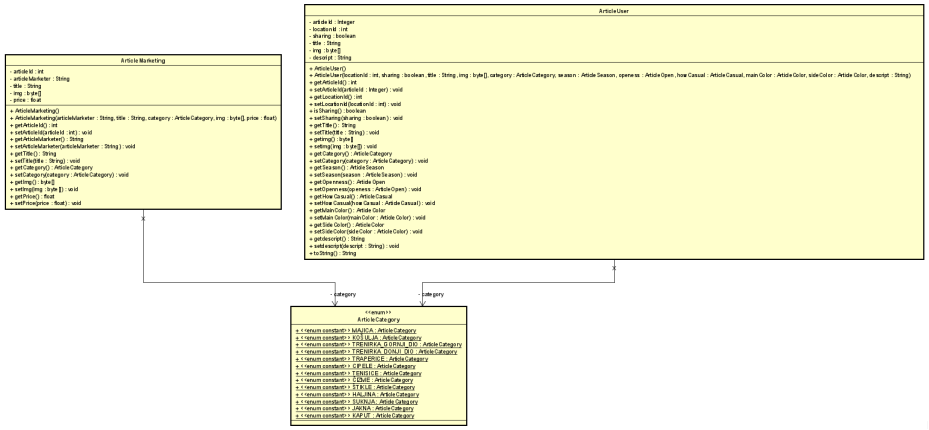
Article Casual Enumeracija



image

Enumeracija koja prikazuje ležernost artikla odjeće.

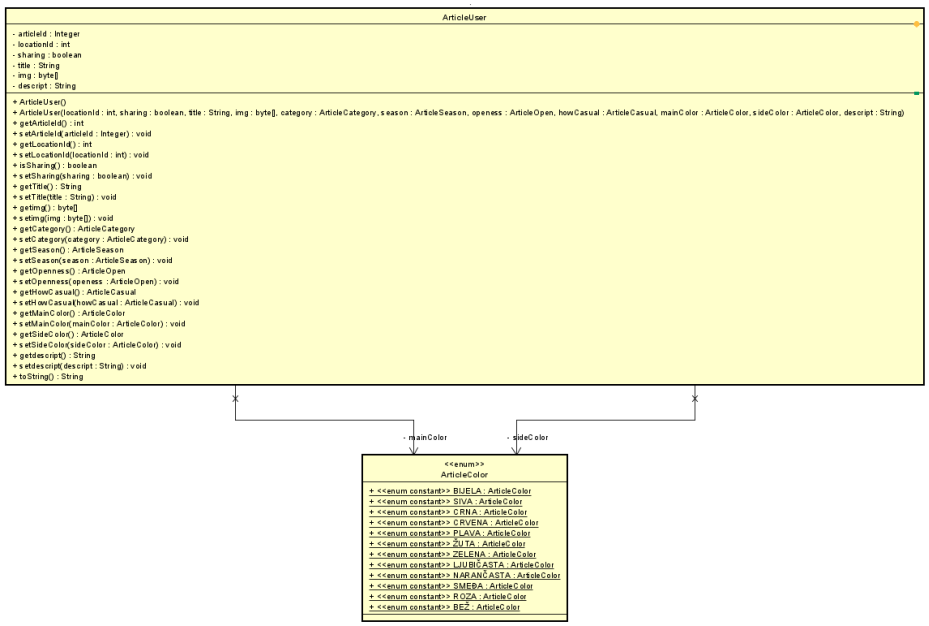
Article Category Enumeracija



image

Enumeracija koja prikazuje kategoriju artikla odjeće.

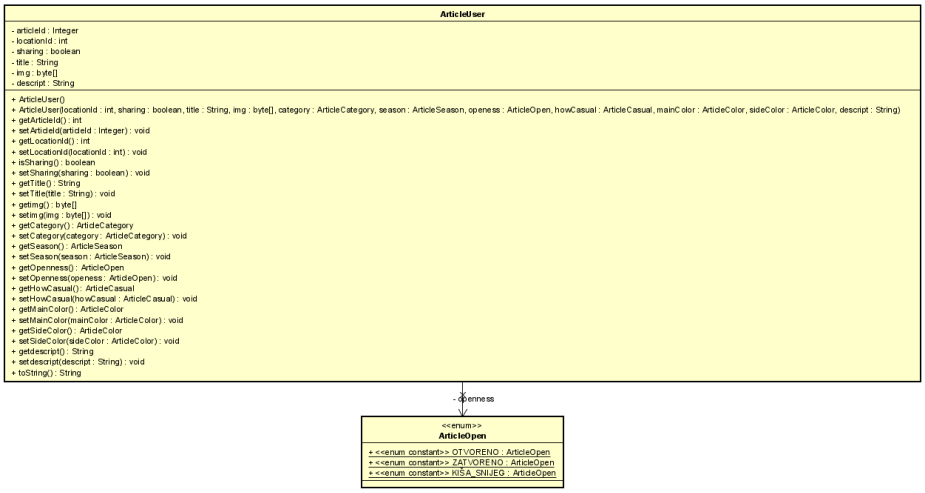
Article Color Enumeracija



image

Enumeracija koja prikazuje boju artikla odjeće.

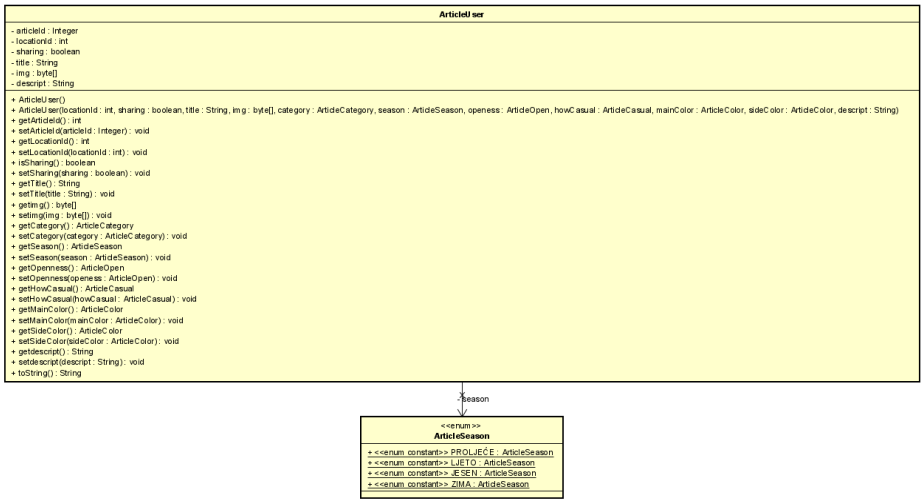
Article Open Enumeracija



image

Enumeracija koja prikazuje otvorenost artikla odjeće.

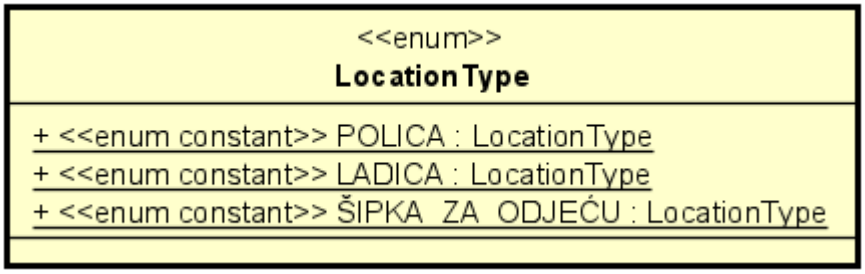
Article Season Enumeracija



image

Enumeracija koja prikazuje godišnje doba za koje je artikl odjeće predviđen.

Location Type Enumeracija

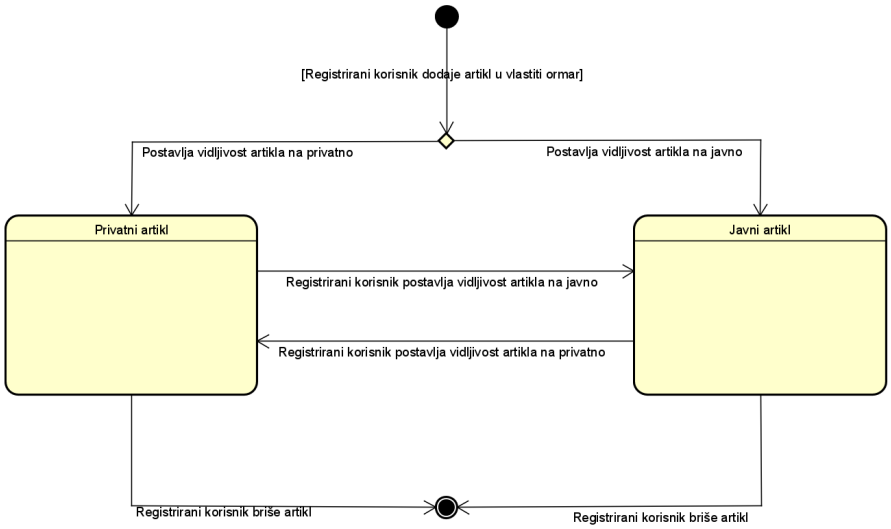


image

Enumeracija koja prikazuje tip lokacije u ormaru.

Dijagram stanja

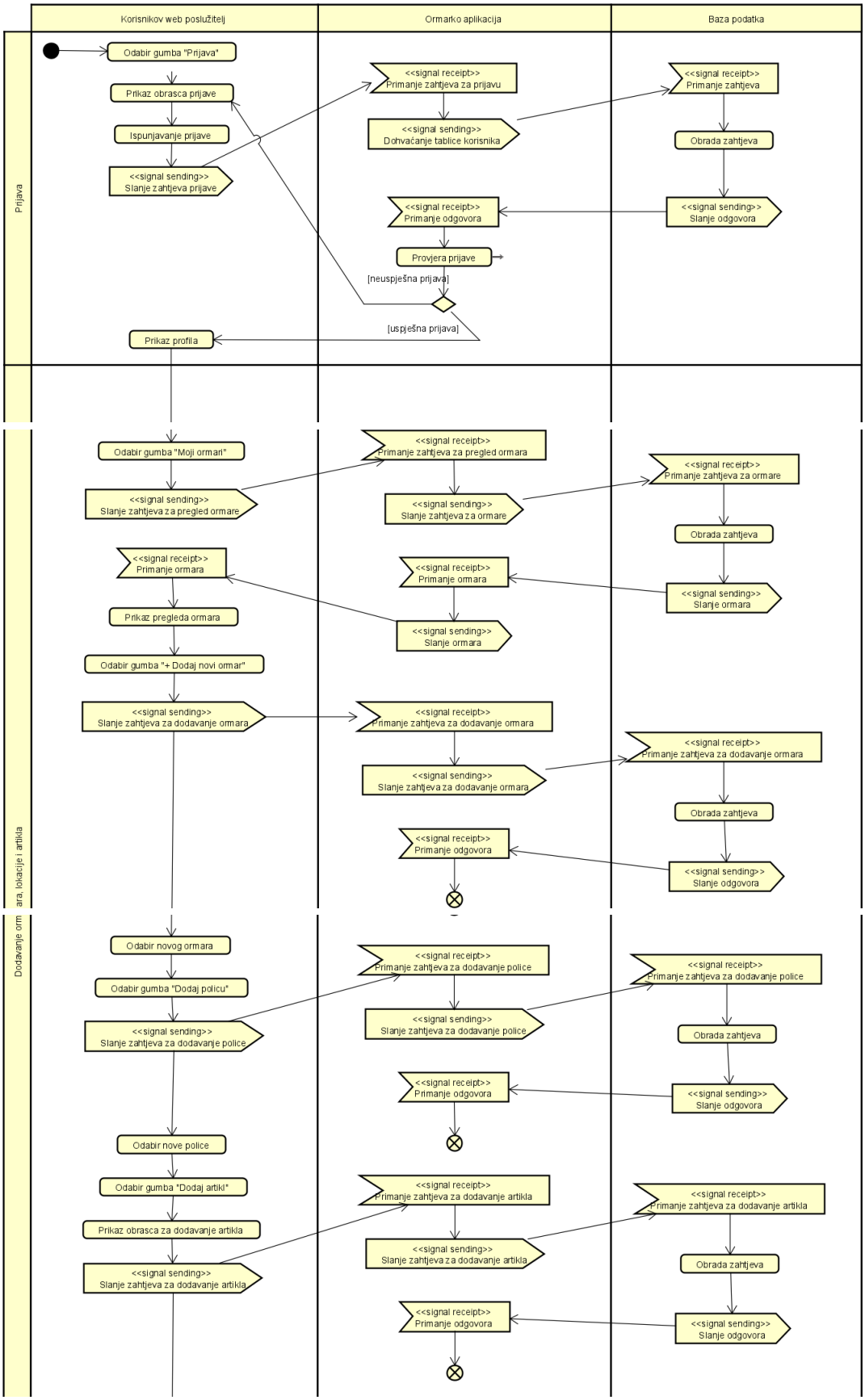
Dijagram predstavlja funkcionalnost promijene vidljivosti artikla. Kada registrirani korisnik dodaje artikl u vlastiti ormar, tada se artikl stvara u bazi podataka i mora mu postaviti vidljivost na privatno ili javno. Korisnik zatim može proizvoljno mijenjati vidljivost artikla između privatno i javno. Kada korisnik briše artikl tada se i objekt artikla briše iz baze podataka.

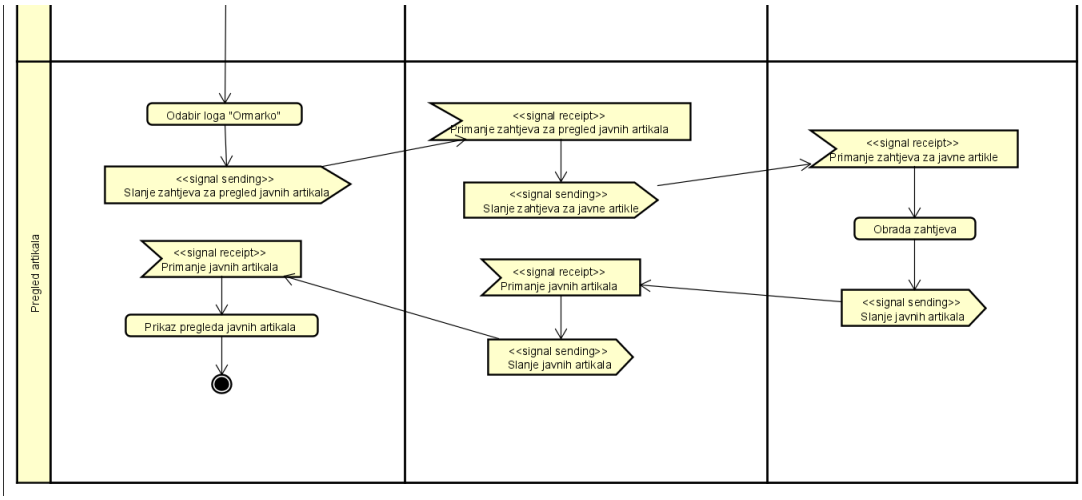


image

Dijagram aktivnosti

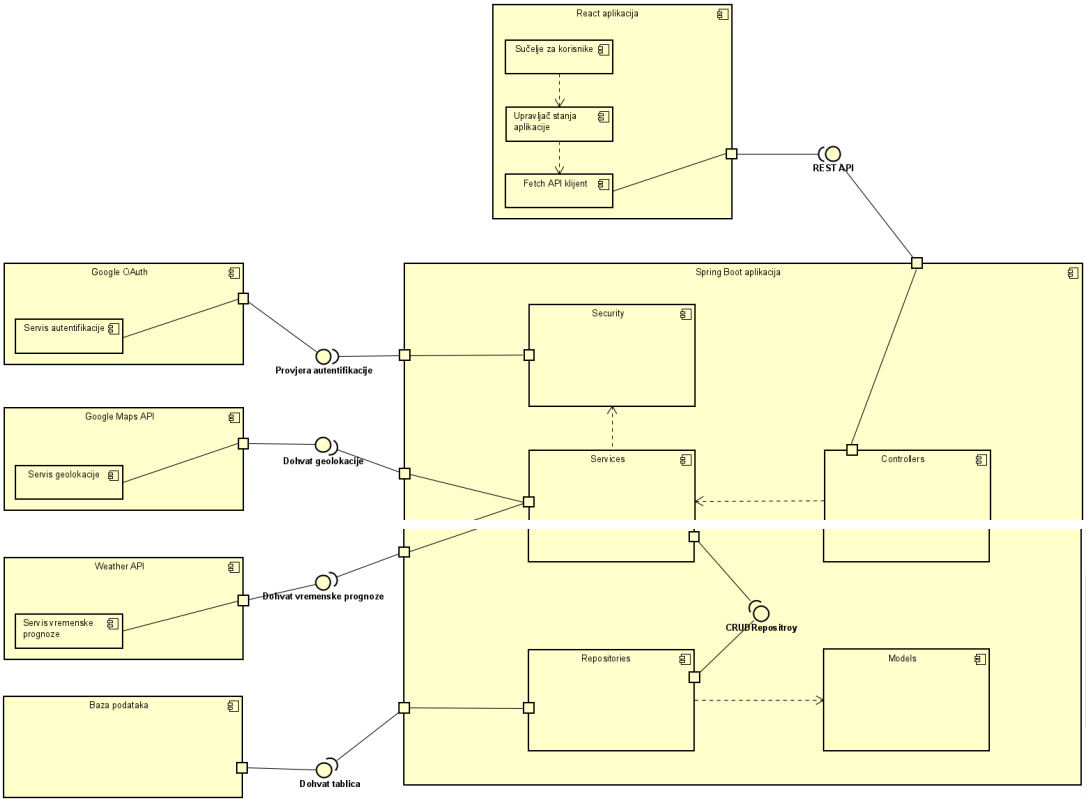
Dijagram predstavlja moguću aktivnost korisnika. Aktori dijagrama su: korisnikov web preglednik (korisnik), Ormarko aplikacija i baza podataka. Dijagram prikazuje prijavu korisnika i moguću neuspješnu prijavu. Zatim korisnik slijedno dodaje vlastiti ormar, policu i artikl. Sve promjene su zabilježene u bazi podataka. Na kraju korisnik pregledava javne artikle, ako pri dodavanju svog artikla odabrao kućicu “Dijeli”, njegov artikl bi trebao biti javno vidljiv.


Korisnikov web poslužitelj
Ormarko aplikacija
Baza podatka
Prijava
Odabir gumba "Prijava"
Prikaz obrasca prijave
Ispunjavanje prijave
<<signal sending>>
Slanje zahtjeva prijave
<<signal receipt>>
Primanje zahtjeva za prijavu
<<signal sending>>
Dohvaćanje tablice korisnika
<<signal receipt>>
Primanje zahtjeva
Obrada zahtjeva
<<signal sending>>
Slanje odgovora
<<signal receipt>>
Primanje odgovora
Provjera prijave
[neuspješna prijava]
[uspješna prijava]
Prikaz profila
Odabir gumba "Moji ormari"
<<signal sending>>
Slanje zahtjeva za pregled ormara
<<signal receipt>>
Primanje zahtjeva za pregled ormara
<<signal sending>>
Slanje zahtjeva za ormare
<<signal receipt>>
Primanje zahtjeva za ormare
Obrada zahtjeva
<<signal sending>>
Slanje ormara
<<signal receipt>>
Primanje ormara
Prikaz pregleda ormara
Odabir gumba "+ Dodaj novi ormar"
<<signal sending>>
Slanje zahtjeva za dodavanje ormara
<<signal receipt>>
Primanje zahtjeva za dodavanje ormara
<<signal sending>>
Slanje zahtjeva za dodavanje ormara
<<signal receipt>>
Primanje zahtjeva za dodavanje ormara
Obrada zahtjeva
<<signal sending>>
Slanje odgovora
<<signal receipt>>
Primanje odgovora
Dodavanje orm ara, lokacije i artikla
Odabir novog ormara
Odabir gumba "Dodaj policu"
<<signal sending>>
Slanje zahtjeva za dodavanje police
<<signal receipt>>
Primanje zahtjeva za dodavanje police
<<signal sending>>
Slanje zahtjeva za dodavanje police
<<signal receipt>>
Primanje zahtjeva za dodavanje police
Obrada zahtjeva
<<signal sending>>
Slanje odgovora
<<signal receipt>>
Primanje odgovora
Odabir nove police
Odabir gumba "Dodaj artikl"
Prikaz obrasca za dodavanje artikla
<<signal sending>>
Slanje zahtjeva za dodavanje artikla
<<signal receipt>>
Primanje zahtjeva za dodavanje artikla
<<signal sending>>
Slanje zahtjeva za dodavanje artikla
<<signal receipt>>
Primanje zahtjeva za dodavanje artikla
Obrada zahtjeva
<<signal sending>>
Slanje odgovora
<<signal receipt>>
Primanje odgovora



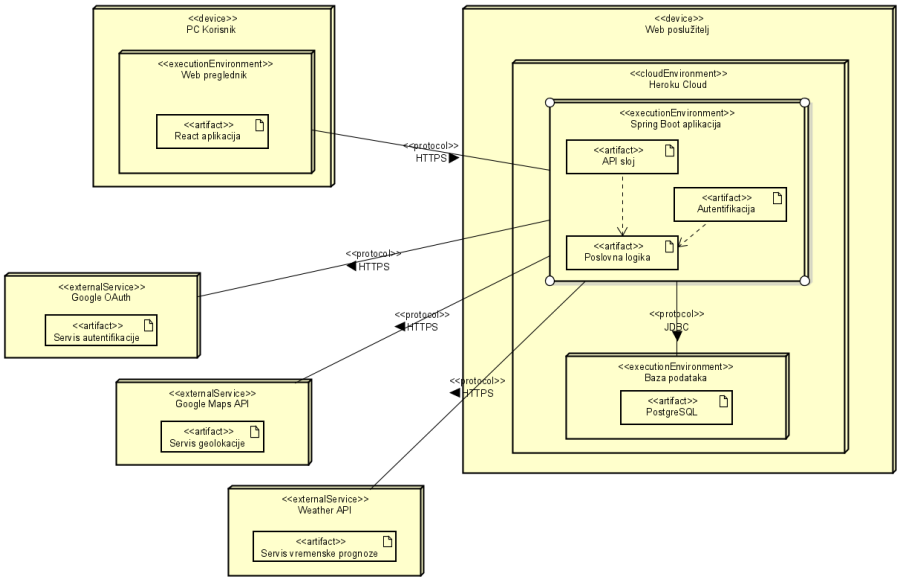
Dijagram komponenata

Dijagram predstavlja arhitekturu sustava pomoću komponenti i sučelja. Sustav se sastoji od React aplikacije, Spring Boot aplikacije, baze podataka i vanjskih servisa. React aplikacija je povezana s Spring Boot aplikacijom pomoću REST API. Baza podataka je povezana s Spring Boot aplikacijom pomoću JDBC za dohvaćanje podatka. Vanjski servisi su povezani s Spring Boot aplikacijom pomoću svog API. Spring Boot aplikacija se sastoji od kontrolera, servisa, zaštite, repozitorija i modela.



Dijagram razmještaja

Specifikacijski dijagram razmještaja prikazuje specifikacije sustava. Aplikacija Ormarko je postavljena na Heroku poslužitelj u oblaku. Za backend aplikacije se koristi Java Spring Boot u kojem je napisana glavna logika aplikacije. Frontend dio aplikacije koji se posluži korisniku na njegov web preglednik je pisan u React-u. Backend dio aplikacije koristi i vanjske servise Google Oauth, Google Maps i Weather za dodante funkcionalnosti aplikacije. Komunikacija s frontendom i vanjskim servisima se provodi HTTPS protokolom. Komunikacija s bazom podataka se provodi JDBC protokolom.



image

Ispitivanje programskog rješenja

Ispitivanje na razini komponenti provelo se koristeći JUnit biblioteku za izradu testnih klasa u programskom jeziku Java. Ispitivanje komponenti uključivalo je ispitivanje stanja i metoda klasa nužnih za ostvarivanje temeljnih funkcionalnosti. Ispitivanje na razini sustava bilo je provedeno koristeći SeleniumIDE tehnologiju. Ispitni slučajevi izrađeni su na temelju funkcionalnih zahtjeva definiranih na početku projekta.

Ispitivanje komponenti

Za ispitivanje na razini komponenti, koristeći tehnologiju JUnit, izrađeni su i provedeni slijedeći ispitni slučajevi:

Naziv	Klasa	Ispitivana metoda/ stanje	Ulaz	Očekivani
TC-01	Korisnik	Postava i dohvat unutarnjih varijabli klase	setUsername(“lbulic”), setPass(“Ormarko123”), setCity(“Zagreb”), setCountry(“Croatia”), setE_mail(“luka.bulic0302@gmail.com”), setGoogleoauth(false)	getUsernar getPass()=“ getCity()=“ getCountry getE_mail(getGooglec
TC-02	Korisnik	Ispis objekta klase	User(“lbulic”, “Ormarko123”, “Zagreb”, “Croatia”, “luka.bulic0302@gmail.com”, false)	“User{user pass=‘Orm country=‘C eMail=‘luk googleoaut
TC-03	Oglašivač	Postava i dohvat unutarnjih varijabli klase	setUsername(“lbulic”), setPass(“Ormarko123”), seteMail(“luka.bulic0302@gmail.com”)	getUsernar getPass()=“ geteMail()=
TC-04	Ormar	Postava i dohvat unutarnjih varijabli klase	setClosetId(1), setClosetOwner(“lbulic”)	getClosetId getClosetO
TC-05	Ormar	Ispis objekta klase	Closet(“lbulic”) + setClosetId(1)	“Closet{clo
TC-06	Lokacija	Postava i dohvat unutarnjih varijabli klase	setLocationId(1), setClosetId(1), setTypeLoc(LocationType.POLICA)	getLocatio getTypeLo
TC-07	Lokacija	Ispis objekta klase	Location(1, LocationType.POLICA) + setLocationId(1)	“Location{ typeLoc=P
TC-08		Postava i dohvat	setArticleId(1), setLocationId(2), setSharing(true), setTitle(“T-shirt”),	getArticleI getSharing

Naziv	Klasa	Ispitivana metoda/ stanje	Ulaz	Očekivani
	Artikl korisnika	unutarnjih varijabli klase	setCategory(ArticleCategory.MAJICA), setSeason(ArticleSeason.LJETO), setOpenness(ArticleOpen.OTVORENO), setHowCasual(ArticleCasual.LEŽERNO), setMainColor(ArticleColor.ZELENA), setSideColor(ArticleColor.CRVENA), setdescript(“T-shirt for casual wearing.”)	getCategor getSeason(getOpenne getHowCas getMainCo getSideCol getdescript wearing.”
TC-09	Artikl korisnika	Ispis objekta klase	ArticleUser(1, true, “T-shirt”, new byte[] {1,2,3,4}, ArticleCategory.MAJICA, ArticleSeason.PROLJEĆE, ArticleOpen.OTVORENO, ArticleCasual.LEŽERNO, ArticleColor.CRNA, ArticleColor.BIJELA, “Another T-shirt.”) + setArticleId(2)	“ArticleUse descript=‘A
TC-10	Artikl oglašivača	Postava i dohvat unutarnjih varijabli klase	setArticleId(1), setArticleMarketer(“lbulic”), setTitle(“T-shirt”), setCategory(ArticleCategory.MAJICA), setPrice(3.22f)	getArticleI getArticleM getTitle()=‘ getCategor getPrice()=

Ispitivanje sustava

Za ispitivanje na razini sustava, koristeći tehnologiju SeleniumIDE, izrađeni su i provedeni slijedeći ispitni slučajevi:

Naziv	Funkcionalni zahtjev	Opis	Ulaz	Očekivani izlaz	I
TS-01	F-004	Registracija novog korisnika	Korisničko ime: lbulic; Grad: Zagreb; Država: Hrvatska; E-mail: luka.bulic0302@gmail.com; Lozinka: Ormarko123	Uspješna registracija i prikaz profilne stranice	U I I s
TS-02	F-004	Prijava korisnika s ispravnim podatcima	Korisničko ime: lbulic; Lozinka: Ormarko123	Uspješna prijava i prikaz profilne stranice	U i s

Naziv	Funkcionalni zahtjev	Opis	Ulaz	Očekivani izlaz
TS-03	F-004	Prijava korisnika s neispravnim podatcima	Korisničko ime: lbulic; Lozinka: Ormarko1245	Upozorenje o neispravnim prijavnim podatcima
TS-04 (v1)	F-004	Prijava korisnika preko Google računa	Google račun: luka.bulic0302@gmail.com	Uspješna prijava i prikaz profilne stranice
TS-05	F-004	Registracija novog korisnika s neispravnim podatcima	Korisničko ime: [PRAZNO]; Grad: Zagreb; Država: Hrvatska; E-mail: luka.bulic0302@gmail.com; Lozinka: Ormarko123	Upozorenje o neisupnjenom obaveznom polju
TS-06 (v1)	F-004	Registracija novog korisnika s podatcima koji su već u upotrebi	Korisničko ime: lbulic; Grad: Zagreb; Država: Hrvatska; E-mail: luka.bulic0302@gmail.com; Lozinka: Ormarko123	Upozorenje o podatcima koji su već u upotrebi
TS-07	F-001	Pretraživanje dijeljenih artikla s jednim filterom kao neregistrirani korisnik	Filter: MAJICA	Prikaz artikla “Summer T-Shirt”
TS-08 (v1)	F-001	Pretraživanje dijeljenih artikla s dva filtera kao neregistrirani korisnik	Filteri: MAJICA + LJETO	Prikaz artikla “Summer T-Shirt”
TS-09	F-001	Pretraživanje dijeljenih artikla bez označenih filtera kao	Filteri: /	Prikaz poruke “No products found matching your filters.”

Naziv	Funkcionalni zahtjev	Opis	Ulaz	Očekivani izlaz
		neregistrirani korisnik		
TS-10	F-005	Dodavanje novog ormara kao registrirani korisnik	Odabir “Dodaj novi ormar”	Prikaz novog stvorenog ormara “Ormar #1”
TS-11	F-003	Pregled detalja dijeljenog artikla kao neregistrirani korisnik	Odabir dijeljenog artikla “Majica”	Prikaz kontakta korisnika, slike artikla i detalja o artiklu
TS-12	F-010	Uklanjanje ormara kao registrirani korisnik	Odabir opcije “X” kod “Ormar #1”, te odabir opcije da smo sigurni da želimo izbrisati ormar	Prikaz popisa ormara s porukom “Nema dodanih ormara”
TS-13	F-005	Dodavanje nove lokacije u ormaru kao registrirani korisnik	Odabir “Dodaj ladicu”	Prikaz nove stvorene ladice “Ladica 1”
TS-14	F-006	Dodavanje novog artikla u lokaciji kao registrirani korisnik	Odabir “Dodaj artikl”, zatim unos: Naslov “T-shirt”, Upload slika, Dijeli DA, Kategorija MAJICA, Godišnje doba PROLJEĆE, Ležernost ZA_DOMA, Glavna boja BIJELA, Sporedna boja SIVA, Opis /	Prikaz novog stvorenog artikla “T-shirt” u lokaciji
TS-15	F-008	Dodavanje novog dijeljenog artikla u lokaciji kao registrirani korisnik te pregled na	Odabir “Dodaj artikl”, zatim unos: Naslov “Short pants”, Upload slika, Dijeli DA, Kategorija TRENIRKA_DONJI_DIO, Godišnje doba PROLJEĆE, Ležernost ZA_DOMA,	Prikaz novog stvorenog artikla “Short pants” na početnoj stranici s ostalim

Naziv	Funkcionalni zahtjev	Opis	Ulaz	Očekivani izlaz
		početnoj stranici	Glavna boja BIJELA, Sporedna boja SIVA, Opis /	dijeljenim artiklima

Korištene tehnologije i alati

1. Programski jezici

Javascript 16.13

JavaScript je korišten za razvoj klijentskog dijela aplikacije. Omogućuje interaktivno korisničko sučelje i dinamičke funkcionalnosti. Asinkrono programiranje pomoću funkcija async/await omogućava da aplikacije ne blokiraju izvršavanje dok čekaju vanjske resurse (poput učitavanja podataka ili API poziva koje smo često koristili). Javascript je odabran za razvoj aplikacije jer je popularan jezik s velikom zajednicom podrške te jer je njegovo korištenje standard u razvoju klijentskog dijela aplikacije. Više o navedenoj tehnologiji možete pronaći na sljedećem linku:
https://developer.mozilla.org/en-US/docs/Web/JavaScript

CSS

CSS je korišten za stiliziranje korisničkog sučelja aplikacije. CSS pomaže u definiranju izgleda elemenata poput boja, fontova i rasporeda, omogućava kontrolu nad izgledom web stranica i aplikacija, poboljšava performanse, te pruža fleksibilnost i responzivnost dizajna. Pomoću njega, stranice su dizajnirane da budu responzivne i vizualno privlačne. CSS je korišten jer omogućuje odvajanje dizajna od strukture sadržaja, što olakšava održavanje i unaprjeđenje. Više o navedenoj tehnologiji možete pronaći na sljedećem linku:
https://developer.mozilla.org/en-US/docs/Web/CSS

Java 21

Java je korištena za razvoj poslužiteljske strane aplikacije. Omogućuje statičko programiranje u objektno orijentiranoj paradigmi, koja je zbog svoje karakteristične modularnosti izrazito pogodna za razvoj srednje

složenih aplikacija. Brzina izvođenja programskoga koda je puno bolja u odnosu na dinamičke jezike, a zbog prevođenja koda putem JVM-a, Java je fleksibilan jezik za izvođenje na bilo kojem operacijskom sustavu. Radi tih osobitosti Java je popularan jezik s velikom zajednicom podrške te je standardni odabir za razvoj poslužiteljskog dijela aplikacije. Više o navedenoj tehnologiji možete pronaći na sljedećem linku:
https://www.java.com

2. Radni okviri i biblioteke

Spring Boot 3.4.2

Spring Boot je korišten za razvoj poslužiteljskog dijela aplikacije. Ovaj Java okvir omogućuje brzi razvoj aplikacija zahvaljujući svom modularnom i lako proširivom dizajnu. Spring Boot nudi unaprijed konfigurirane postavke i omogućuje integraciju različitih modula Spring ekosustava, poput Spring Data, Spring Security i Spring MVC, čime se olakšava gradnja i upravljanje složenim aplikacijama. Spring Boot koristi ugrađeni web poslužitelj, kao što su Tomcat ili Jetty, što eliminira potrebu za ručnim postavljanjem vanjskih aplikacijskih poslužitelja. Aplikacije razvijene pomoću Spring Boota lako se pokreću kao samostalne Java aplikacije. Jedan od ključnih aspekata Spring Boota je rad s RESTful API-jem, što omogućuje razmjenu podataka između pozadinskog i korisničkog sučelja. Korištenjem anotacija u opisu metoda i klasa, olakšana je izrada programske podrške. Dodatno, Spring Boot upravlja složenim procesima, poput sigurnosti, autentifikacije i pristupa bazi podataka, time štedeći na vremenu potrošenom na izgradnju aplikacije. Spring Boot je odabran jer omogućuje izradu robusnih, skalabilnih i lako održivih pouzdanih, poslužiteljskih aplikacija uz minimalno postavljanje i konfiguraciju.
Više o navedenoj tehnologiji možete pronaći na sljedećem linku:
https://spring.io/

React 18.3.1

React je korišten za razvoj korisničkog sučelja aplikacije. Ova JavaScript biblioteka omogućuje modularnost kroz komponente koje se mogu ponovno koristiti. To svojstvo čini kod lakšim za održavanje i organiziranijim. Svaka komponenta može imati svoj vlastiti status (state) i prikaz (render), čime se olakšava upravljanje složenim sučeljima i ponovna upotreba koda. React koristi virtualni DOM, što poboljšava rad prilikom ažuriranja korisničkog sučelja. Prvo se ažurira

virtualna verziju DOM-a, uspoređuje se s prethodnim stanjem i zatim se primjenjuju samo promjene na stvarni DOM. U kombinaciji s Reactom, koristili smo Node.js i React Router za upravljanje navigacijom. React je korišten jer omogućuje razvoj brzih, responzivnih i interaktivnih korisničkih sučelja. Više o navedenoj tehnologiji možete pronaći na sljedećem linku: https://react.dev/

Node.js 22.12.0

Node.js je JavaScript runtime okruženje koje omogućuje izvršavanje JavaScript koda izvan preglednika. U aplikaciji je korišten za pokretanje razvojnih alata, kao što su build skripte i dependency manager (npm). Node.js je korišten jer omogućuje učinkovito rukovanje ovisnostima i skriptama. Više o navedenoj tehnologiji možete pronaći na sljedećem linku: https://nodejs.org/en

3. Baza podataka

PostgreSQL i pgAdmin 4 8.13

PostgreSQL inačica je programskog jezika SQL koja omogućava izradu i rukovođenje baze podataka kroz pgAdmin sustav. U sustavu se radi spajanje na lokalnu ili udaljenu bazu podataka, a manipulacija istom odvija se preko SQL naredbi (upita). Na ovom projektu, postgreSQL i pgAdmin sustav koristili su se za izradu tablica baze podataka, postavljanje uvjeta na pojedinačne atribute te povezivanje tablica u jedinstvenu bazu. Više o navedenoj tehnologiji možete pronaći na sljedećim linkovima: https://www.postgresql.org/

4. Razvojni alati

Git

Git je korišten za verzioniranje i suradnju na kodu, što je omogućilo praćenje promjena, vraćanje na prethodne verzije te lakšu suradnju više razvojnih timova. Distribuirani pristup ovom alatu jamči sigurnost i fleksibilnost u razvoju. Više o navedenoj tehnologiji možete pronaći na poveznici: https://git-scm.com/

IntelliJ IDEA 2024.3.2

IntelliJ IDEA je integrirano razvojno okruženje korišteno za pisanje, testiranje i debugiranje koda na klijenstkoj i poslužiteljskoj strani. Podržava mnoge programske jezike poput Jave i JavaScripta koje smo koristili u razvoju aplikacije te nudi integraciju s alatima za izgradnju poput Mavena koji smo koristili u razvoju aplikacije. Njegove napredne značajke poput inteligentnog predlaganja koda, ugrađene podrške za Spring Boot i alati za verzioniranje putem Git-a značajno su ubrzale razvoj. IntelliJ je korišten jer omogućuje produktivnost kroz alate koji olakšavaju upravljanje projektima i rješavanje pogrešaka. Više o navedenoj tehnologiji možete pronaći na sljedećem linku: https://www.jetbrains.com/idea/

5. Alati za ispitivanje

JUnit 5.1

JUnit je biblioteka u programskom jeziku Java koja omogućuje testiranje web aplikacije na razini komponenti. Biblioteka funkcionira tako da se za danu klasu koju želimo ispitati napravi testna klasa, u kojoj se izradom testova pozivaju metode ispitivane klase i procjenjuje jesu li rezultati u skladu s našim očekivanjima. U ovom projektu, JUnit biblioteka koristila se za ispitivanje 6 klasa definiranih u dijagramu razreda, preko kojih su ostvarene osnovne funkcionalnosti web-aplikacije (korisnik, oglašivač, ormar, lokacija, artikl korisnika, artikl oglašivača). Više o navedenoj tehnologiji možete pronaći na sljedećem linku: https://www.vogella.com/tutorials/JUnit/article.html

SeleniumIDE 3.17.2

SeleniumIDE je ekstenzija internetskog preglednika koja omogućuje testiranje web aplikacije na razini sustava. Ekstenzija funkcionira tako da se izradom novog projekta mogu izrađivati različiti ispitni slučajevi koji se zatim izvršavaju u posebnom web prozoru. Također je moguće i u posebnom web prozoru pokrenuti snimanje, čime SeleniumIDE bilježi svaki korak koji je napravljen u toj sekvenci korištenja aplikacije. Tom se funkcionalnošću omogućuje brzo i učinkovito ponovno korištenje ispitnih slučajeva. U ovom projektu, SeleniumIDE korišten je za ispitivanje naše aplikacije na sustavnoj razini, čime je pokrivena većina funkcionalnih zahtjeva definirana na početku projekta. Više informacija o navedenoj tehnologiji možete pronaći na sljedećem linku: https://www.selenium.dev/selenium-ide/

6. Cloud platforma

AWS-RDS

AWS-RDS (*Amazon Relational Database Service*) je internetska usluga za izradu i održavanje relacijske baze podataka. Kroz AWS-RDS uslugu, na bazu podataka je moguće se spojiti udaljeno, kroz definirana svojstva kao što su naziv, korisničko ime, šifra i port. Ovakav način spajanja je nužan za funkcionalnost web aplikacije. U ovom projektu, na AWS-RDS usluzi stvorena je baza podataka na koju se spojilo preko sustava pgAdmin. Nakon izrade relacijske sheme, na bazu se spajalo iz Spring Boot tehnologije radi ostvarenja funkcionalnosti pohrane i dohvata podataka. Više informacija o navedenoj tehnologiji možete pronaći na sljedećem linku: https://aws.amazon.com/rds/

7. Komunikacija

Komunikacija između članova tima se odvijala putem Whatsapp-a i Discorda. Whatsapp je bio primarni alat za brzu i neposrednu komunikaciju. Korišten je za svakodnevne dogovore, kratke poruke, hitne upite i brzo rješavanje manjih problema. Na Discordu smo održavali online sastanke, odrađivali podjelu rada na aplikaciji, usklađivali prioritete te izvještavali o napretku i problemima u razvoju. Za organizaciju i praćenje zadataka koristili smo posebne kanale na Discordu, gdje smo redovito izvještavali o napretku rada, prijavljivali izazove i razmjenjivali ideje za rješenja problema. Ova platforma je također omogućila bolju suradnju jer smo imali zajedničke prostore za raspravu o većim funkcionalnostima aplikacije i tehničkim izazovima.

Upute za puštanje u pogon

1. Instalacija

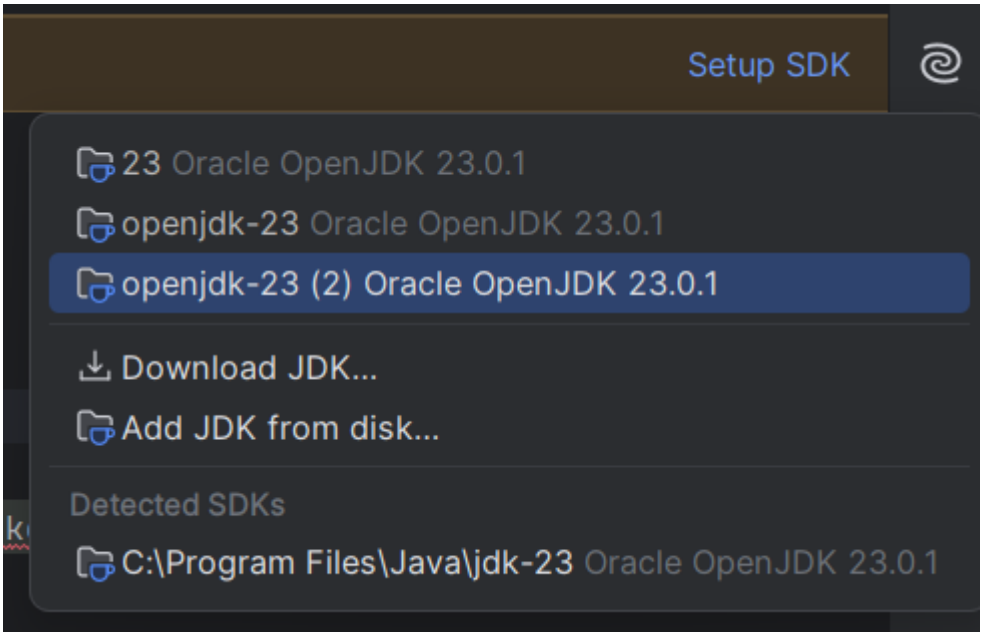
Preduvjeti za instalaciju su sljedeći softveri: * **Node.js**: verzija 22.12.0 * **NPM**: verzija 8 ili novije (dolazi s Node.js) * **Git**: verzija 2.30 ili novija * **Java**: verzija 21 ili 23 * **Maven**: verzija 3.9.9

Također se preporučuje korištenje InteliJ IDEA 2024.3.2.

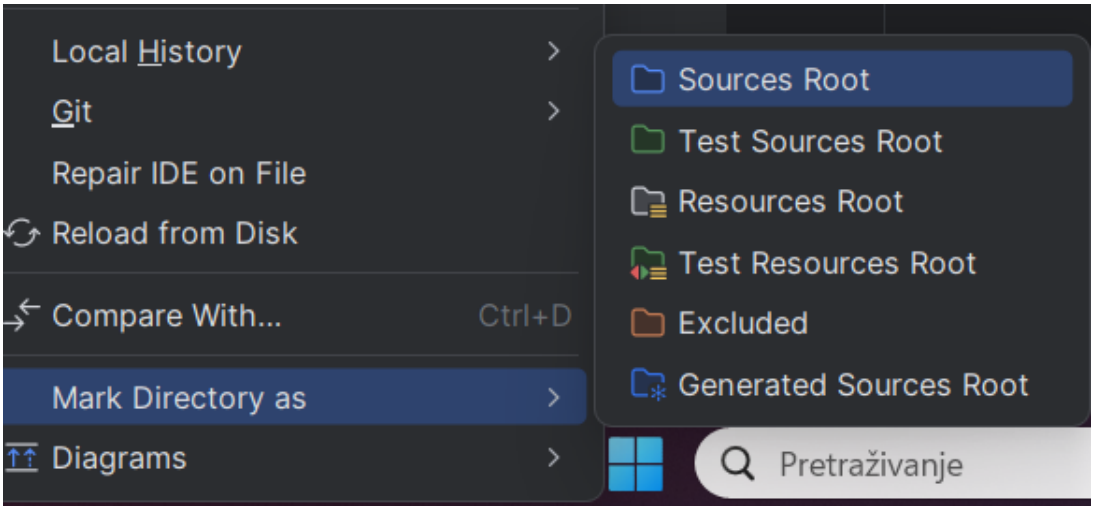
Klonirajte git repozitorij
> git clone https://github.com/vujnovicmarko/Ormarko >
> cd Ormarko

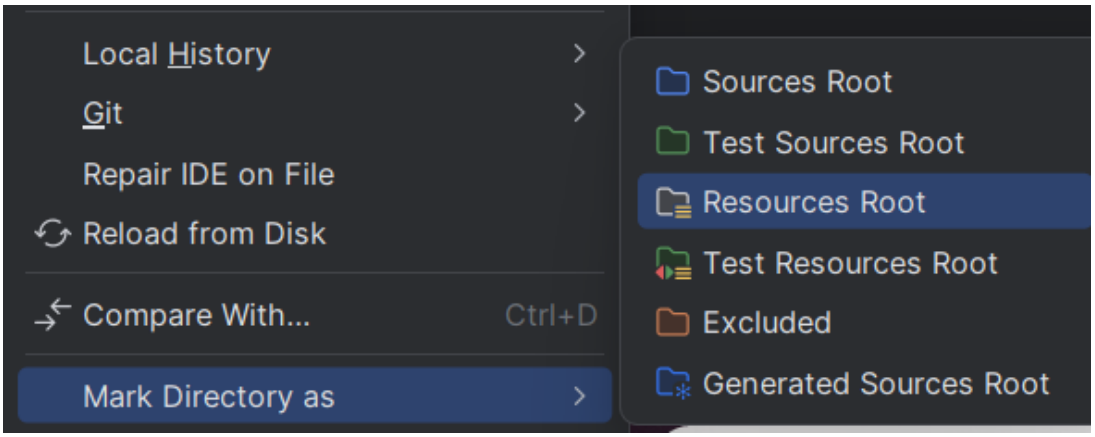
2. Postavke

Preporučuje se koristiti InteliJ IDEA za otvaranje aplikacije. U IntelliJ IDEA otvoriti direktorij Ormarko kao projekt. Ukoliko nije automatski postavljeno, u postavkama projekta namjestiti Java SDK koji se koristi na računalu.

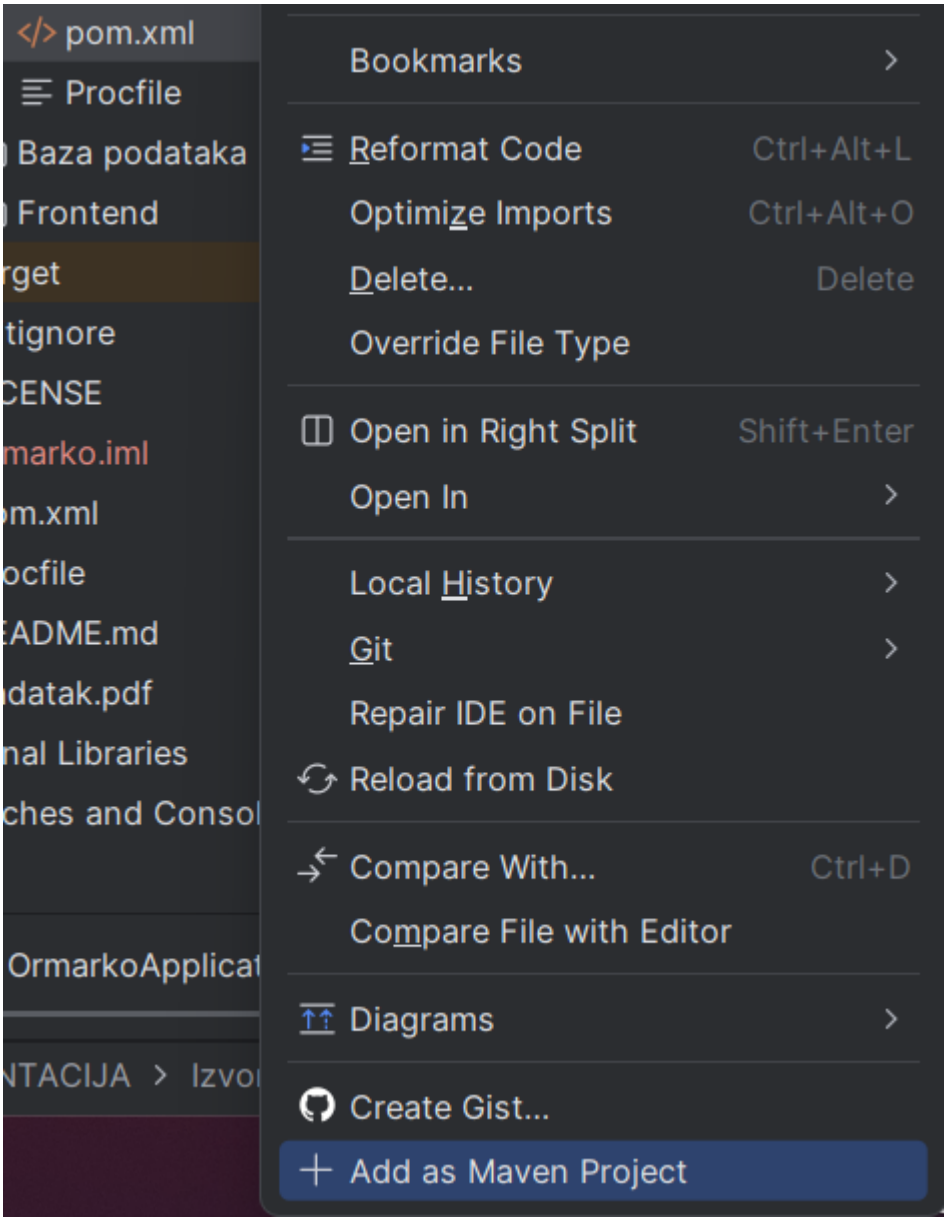


Navigirati u direktorij: Ormarko. Potrebno je odabrati direktorij java te desnim klikom ga označiti kao “Sources Root”, a direktorij resources kao “Resources folder”.

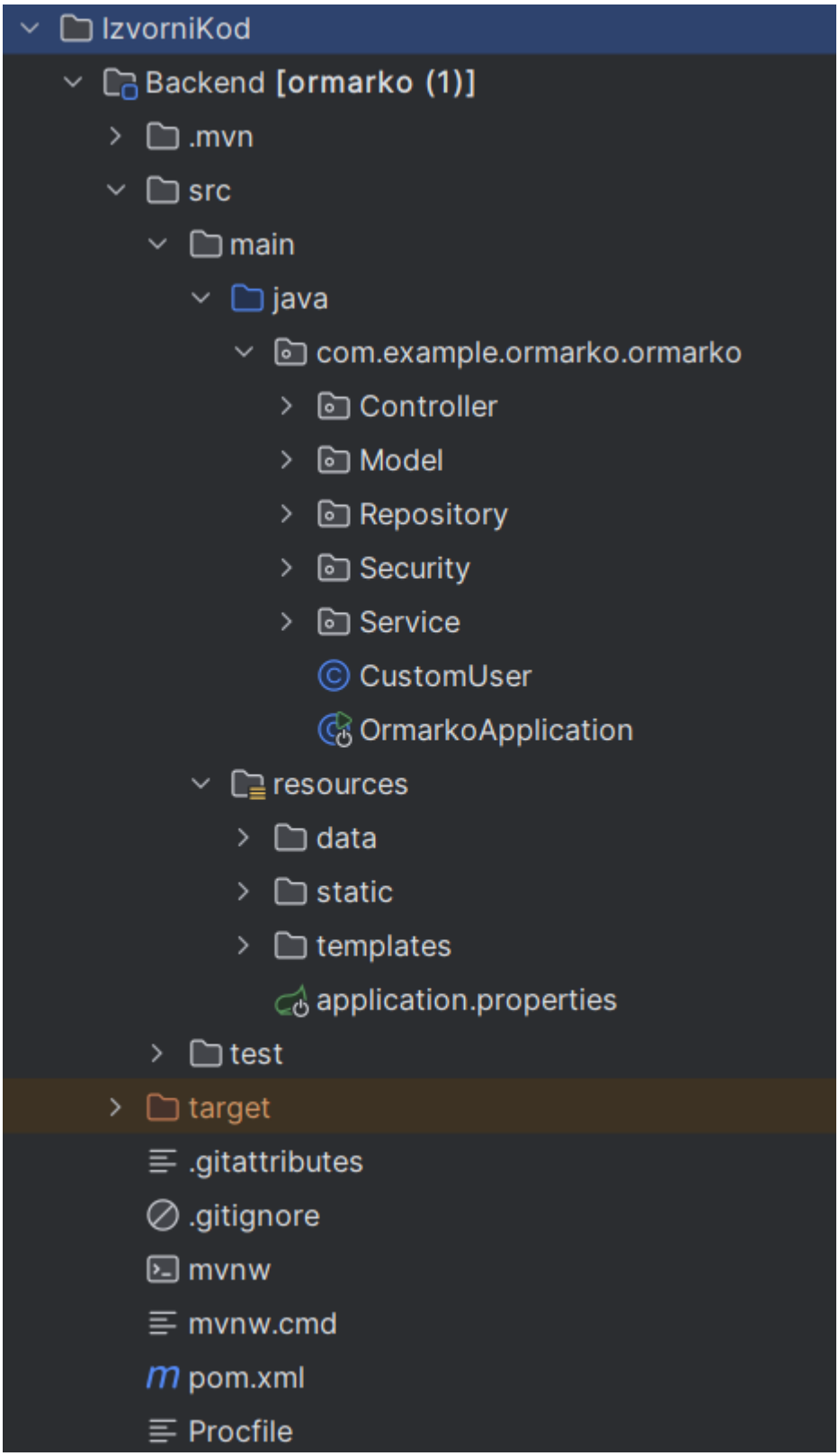




Osigurati da se datoteka pom.xml nalazi u direktoriju Backend te da se pokreće kao Maven project.



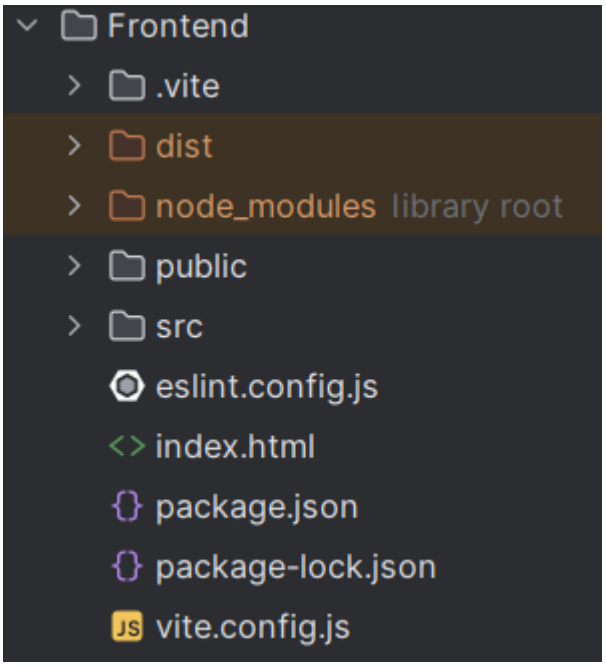
Ovako bi trebao izgledati zadani direktorij:



U terminalu Frontend direktorija pokrenuti naredbu

```
npm install
```

Ovako bi trebao izgledati zadani direktorij:



Namjestiti vite.config.js datoteku. Primjer:

```
'/api': { target: 'http://localhost:8080', changeOrigin:

'/oauth2': { target: 'http://localhost:8080', changeOrigi
```

Bazu podataka nije potrebno dodatno namještati. Pristup njoj je organiziran preko AWS poslužitelja(Cloud platforma).

3. Pokretanje aplikacije

Frontend i Backend se mogu pokrenuti svaki zasebno ili zajedno.
Razvojno okruženje

Frontend se pokreće sljedećom naredbom u direktoriju Frontend > npm run dev

Pokretanjem naredbe, otvara se korisničko sučelje u zadanom web pregledniku na adresi http://localhost:5173/.

Backend se pokreće gumbom Run u OrmarkoApplication.java u razvojnom okruženju ili naredbom > ./mvnw clean spring-boot:run

Na adresi http://localhost:5173/ su tada frontend i backend pokrenuti zajedno.

Produkcijsko okruženje i prevođenje aplikacije

Aplikacija se prevodi sljedećom naredbom u direktoriju Frontend > npm run build

Potrebno je sadržaje novostvorenog dist direktorija kopirati u direktorij static:

.
Aplikacija se pokreće gumbom Run u OrmarkoApplication.java u razvojnom okruženju ili naredbom
> ./mvnw clean spring-boot:run

Adresom http://localhost:8080/ se pristupa pokrenutoj aplikaciji.

4. Upute za administratore

- Ažuriranje aplikacije
  Aplikacija se ažurira sljedećom naredbom u terminalu repozitorija
  > git pull
- Pregled errora i logova
  Errore i logove je moguće pregledati i dijagnosticirati pomoću naredbe Inspect u web pregledniku ili u razvojnom okruženju.

5. Upute za postavljanje na Heroku platformu (Cloud Deploy)

Ovaj vodič objašnjava kako postaviti aplikaciju na Heroku platformu.

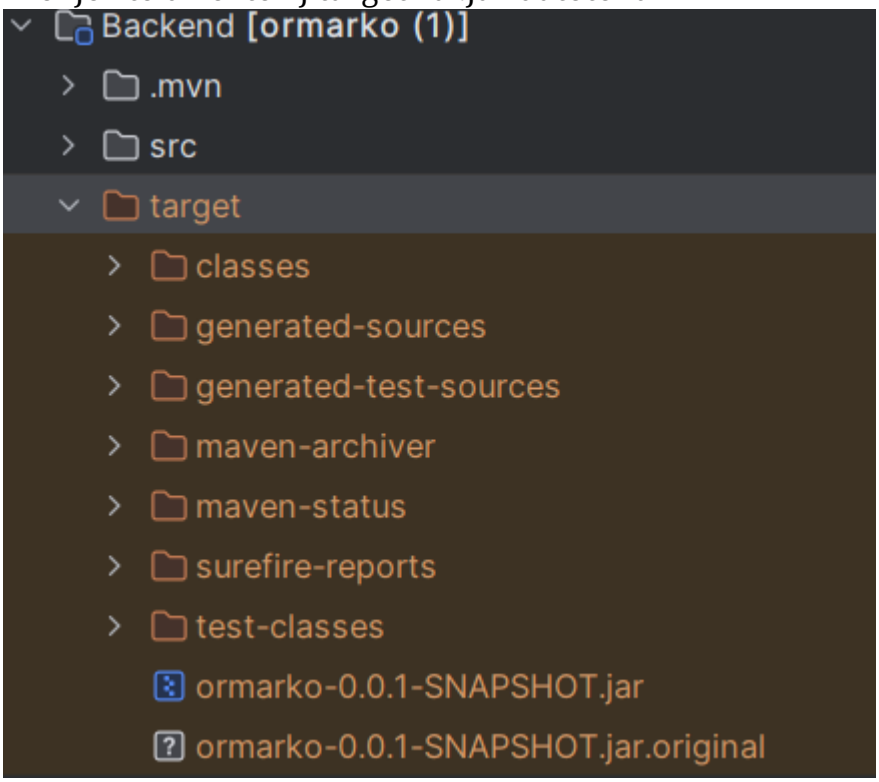
1. Instalirati Heroku CLI

Ako već nemate instaliran Heroku CLI, preuzmite ga i instalirajte s Heroku CLI službene stranice https://devcenter.heroku.com/articles/heroku-cli
Provjerite instalaciju pokretanjem naredbe:
> heroku –version

2. Priprema aplikacije za deployment

- napravite build aplikacije pomoću Maven-a sljedećom naredbom > mvn clean package

- Provjerite direktorij target za .jar datoteku



- Napravite Procfile
  Napravite datoteku Procfile u istom direktoriju gdje se nalazi pom.xml
  Primjer sadržaja Procfile datoteke: > web: java -jar target/ormarko-0.0.1-SNAPSHOT.jar

3. Postaviti aplikaciju na Heroku

- Inicijaliziranje Git repozitorija > git init
  > git add .
  > git commit -m “Prepare backend for Heroku deployment”

- Prijava na heroku > heroku login

- Stvaranje Heroku aplikacije > heroku create *imeaplikacije*

Ovo će generirati URL https://*imeaplikacije*.herokuapp.com gdje će aplikacija biti dostupna * Dodati buildpack > heroku buildpacks:set heroku/java

- Povežite lokalni repozitorij s Heroku aplikacijom > heroku git:remote -a ormarko-backend

- Deploy na Heroku > git push heroku main

4. Provjera aplikacije

- Logovi se mogu pratiti pomoću naredbe: > heroku logs –tail
- Otvaranje aplikacije u pregledniku > heroku open

Opis pristupa aplikaciji na javnom poslužitelju

Aplikacija je puštena u pogon pomoću platforme Heroku. Web stranici se može pristupiti putem linka: https://ormarkodeploy-c46f4289b2cf.herokuapp.com/

Zadatak ovog projekta bio je razviti web aplikaciju koja igra ulogu virtualnog ormara. Korištenje naše aplikacije omogućilo bi virtualnu organizaciju odjevnih predmeta za svakog korisnika, kao i pregled predmeta dijeljenih od drugih korisnika ili preko oglasa. Projekt se odvijao u dva ciklusa. U prvom ciklusu je bio formiran tim, podijelila su se zaduženja među članovima te je izrađena baza podataka i prototip aplikacije s jednom osnovnom funkcionalnošću. Drugi ciklus uključivao je implementaciju ostalih funkcionalnosti i ispitivanje aplikacije na razini komponenti i sustava. Dokumentacija je bila redovito ažurirana kako se projekt odvijao.

Komunikacija među članovima tima odvijala se preko mreža Whatsapp i Discord, kao i preko određenih funkcionalnosti GitHub-a kao što su ‘Issues’. Glavna poteškoća na koju smo naišli bila je deployment aplikacije na kraju prvog ciklusa, zbog tehničkih nejasnoća i vremenskog ograničenja rokom predaje. Na kraju je za prvi ciklus deployment bio odrađen za dio aplikacije, a puni deployment bio je učinjen među prvim aktivnostima drugog ciklusa.

Za ostvarivanje ciljeva ovog projekta ključne su bile tehničke vještine vezane uz tehnologije PostgreSQL, Spring Boot, React, SeleniumIDE i Heroku sustav za deployment. Također, za pravilno i slijedno provođenje projektnih aktivnosti bila su važna znanja kolegija “Programsko inženjerstvo” kao što su analiza zahtjeva, dizajn arhitekture sustava, izrada UML dijagrama, provođenje ispitivanja na više razina uz prikladne testne slučajeve, itd.

Budući smjerovi ovog projekta uključivali bi osmišljavanje novih funkcionalnosti i implementaciju istih, uz daljnje osvježavanje dokumentacije. Također, mogla bi se provesti razmatranja računalne učinkovitosti postojećeg sustava te uočiti potencijalna mjesta za optimizaciju.

Sve su funkcionalnosti uspješno implementirane.

Rev.	Opis promjena/datoteka	Autori	Datum
0.1	Postavljanje Wikija	Tanja	20.10.2024.
0.2	Stvaranje predloška	Marko	31.10.2024.
0.3	Opis baze podataka	Luka	04.11.2024.
0.4	Dodao dijagrame obrazaca uporabe	Marko	06.11.2024.
0.5	Nadopunjen dnevnik sastanaka i plan rada	Luka	06.11.2024.
0.6	Dodao sekvencijske dijagrame	Marko	08.11.2024.
0.7	Dodao opise sekvencijskih dijagrama	Marko	09.11.2024.
0.8	Opis arhitekture sustava	Tanja	10.11.2024.
0.9	Dodao dijagrame razreda	Marko	12.11.2024.
0.10	Napisan izvještaj o radu za 1. predaju	Luka	15.11.2024.
0.11	Uređen popis literature	Tanja	15.11.2024.
0.12	Izvještaj rada članova projekta	Tanja	15.11.2024.
1.1	Unos slučajeva ispitivanja sustava, ažuriranje dnevnika sastanaka i tjednog plana	Luka	13.1.2025.
1.2	Unos slučajeva ispitivanja sustava i komponenti	Luka	19.1.2025.
1.3	Dodao dijagrame stanja, aktivnosti, komponenti i razmještaja	Marko	20.1.2025.
1.4	Ažuriranje dnevnika sastanaka, napisan zaključak, naveden dio korištenih tehnologija	Luka	20.1.2025.
1.5	Upute za puštanje u pogon i ažuriranje predloška	Tanja	22.1.2025.

Kontinuirano osvježavanje 1. Programsko inženjerstvo, FER ZEMRIS 2. Astah User Guide 3. Modeliranje programske potpore UML-dijagramima, FER ZEMRIS 4. React Tutorial for Beginners 5. PgAdmin 4 documentation 8.13 6. W3Schools SQL tutorial 7. Creating a React + Spring Boot Web App, Coders Campus 8. Learn the basics of git in

under 10 minutes 9. Git - the simple guide 10. React website 11. React Website Tutorial - Beginner React JS Project Fully Responsive 12. Building web applications in Java with Spring Boot 3 – Youtube Tutorial 13. Selenium testing 14. JUnit 15. Heroku

Dnevnik sastajanja

1. sastanak

- 14.10.2024.
- Prisustvovali: L. Benički, T. Bertalanić, L. Bulić, A. Andročec, M. Vujnović, I. Zubčić, L. Đuranec
- Teme sastanka: > * diskusija projektnog zadatka > * podjela odgovornosti u timu

1. sastanak

- 21.10.2024.
- Prisustvovali: L. Benički, T. Bertalanić, L. Bulić, A. Andročec, I. Zubčić, L. Đuranec
- Teme sastanka: > * prezentacija funkcijskih i nefunkcijskih zahtjeva projekta > * prezentacija tehnologija u kojima će projekt biti ostvaren

1. sastanak

- 28.10.2024.
- Prisustvovali: L. Benički, T. Bertalanić, L. Bulić, A. Andročec, M. Vujnović, L. Đuranec
- Teme sastanka: > * prezentacija izrađenih dijagrama obrazaca uporabe > * prezentacija izrađenih sekvencijskih dijagrama

1. sastanak

- 04.11.2024.
- Prisustvovali: L. Benički, T. Bertalanić, L. Bulić, A. Andročec, I. Zubčić, L. Đuranec
- Teme sastanka: > * prezentacija baze podataka > * prezentacija osnovne backend arhitekture

1. sastanak

- 11.11.2024.

- Prisustvovali: L. Benički, T. Bertalanić, L. Bulić, A. Andročec, M. Vujnović, I. Zubčić, L. Đuranec
- Teme sastanka: > * demonstracija osnovnih funkcionalnosti aplikacije

1. sastanak

- 16.12.2024.
- Prisustvovali: L. Benički, T. Bertalanić, L. Bulić, A. Andročec, M. Vujnović, I. Zubčić, L. Đuranec
- Teme sastanka: > * dogovor za daljnju implementaciju funkcionalnosti aplikacije

1. sastanak

- 9.1.2025.
- Prisustvovali: L. Benički, T. Bertalanić, L. Bulić, A. Andročec, M. Vujnović, I. Zubčić, L. Đuranec
- Teme sastanka: > * demonstracija rezultata ispitivanja sustava

1. sastanak

- 13.1.2025.
- Prisustvovali: L. Benički, T. Bertalanić, L. Bulić, A. Andročec, M. Vujnović, I. Zubčić, L. Đuranec
- Teme sastanka: > * demonstracija novih implementiranih funkcionalnosti registracije korisnika i pretraživanja artikla

1. sastanak

- 20.1.2025.
- Prisustvovali: L. Benički, T. Bertalanić, L. Bulić, M. Vujnović, I. Zubčić, L. Đuranec
- Teme sastanka: > * finalna demonstracija pred drugu predaju i dogovor oko preostalih obaveza

Plan rada

Tjedan	Zadatci
1.	Podjela zaduženja unutar tima i dogovor oko korištene tehnologije
2.	

Tjedan	Zadatci
	Proučavanje funkcijskih i nefukncijskih zahtjeva, izrada use-case i sekvencijskih dijagrama
3.	Izrada sheme baze podataka, postavljanje baze podataka na AWS server, izrada osnovne arhitekture poslužitelja te jednostavnog korisničkog sučelja
4.	Implementacija funkcija za prijavu i registraciju te drugih temeljnih funkcionalnosti
5.	Finalizacija i predaja prve revizije projekta i projektne dokumentacije
6.-7.	Uspješno proveden deployment prve vezije. Konceptualna razrada ispitnih slučajeva na razini komponenti i na razini sustava.
8.	Implementacija funkcionalnosti registracije oglašivača i pretraživanja artikla
9.	Implementacija funkcionalnosti stvaranja ormara i lokacija te dodavanja artikla, ispitivanje komponenti i sustava.
10.	Dovršavanje završne verzije projekta i deployment, dovršavanje dokumentacije.

Tablica aktivnosti

Stavka	Osoba - Br. sati
Upravljanje projektom	Tanja Bertalanić - 10h
Opis projektnog zadatka	Marko Vujnović - 1h
Funkcionalni zahtjevi	Tanja Bertalanić - 1h, Lovro Đuranec - 1h, Marko Vujnović - 1h
Opis pojedinih obrazaca	Marko Vujnović - 12h
Dijagram obrazaca	Marko Vujnović - 12h

Stavka	Osoba - Br. sati
Sekvencijski dijagrami	Marko Vujnović - 10h
Opis ostalih zahtjeva	Marko Vujnović - 1h
Arhitektura i dizajn sustava	Andrija Andročec - 40h, Lana Benički - 42h , Tanja Bertalanić - 40h, Iva Zubčić - 35h, Lovro Đuranec - 40h
Baza podataka	Luka Bulić - 18h
Dijagram razreda	Marko Vujnović - 5h
Dijagram stanja	Marko Vujnović - 5h
Dijagram aktivnosti	Marko Vujnović - 3h
Dijagram komponenti	Marko Vujnović - 2h
Korištene tehnologije i alati	Andrija Andročec - 5h, Lana Benički - 5h, Lovro Đuranec - 1h, Tanja Bertalanić - 3h
Ispitivanje programskog rješenja	Luka Bulić - 2h (BP), Andrija Andročec - 5h, Lana Benički - 9h, Tanja Bertalanić - 4h, Lovro Đuranec - 8h; Luka Bulić - 16h (ispitivanje komponenti i sustava), Iva Zubčić - 4h
Dijagram razmještaja	Marko Vujnović - 2h
Upute za puštanje u pogon	Tanja Bertalanić - 3h
Dnevnik sastajanja	Luka Bulić - 3h
Zaključak i budući rad	Luka Bulić - 1h
Popis literature	Luka Bulić - 15min, Lana Benički - 5min, Tanja Bertalanić - 10min

Izvještaj rada članova projekta - 1. predaja

Kao voditelj projekta, tražila sam svakog člana tima da mi ukratko opiše kako su pridonjeli projektu u prvom ciklusu. Sav njihov rad i napredak dokumetirani su na githubu putem commitova, issuesa i wikija. > Tanja Bertalanić, voditelj tima

Luka Bulić

Baza podataka

Moje glavno zaduženje prilikom prvog ciklusa ovog projekta bila je izrada baze podataka, postavljanje baze podataka na poslužitelj, ispitivanje baze podataka nakon postavljanja te pomoć pri spajanju baze podataka s backend timom. Za početak, bilo je potrebno razmotriti tehnologije u kojima će baza podataka biti realizirana. Tu su dolazili u obzir MySQL i PostgreSQL, ali je odabran PostgreSQL zbog prethodnog iskustva na FER-ovim kolegijima. Slijedeći korak bio je detaljno čitanje projektnog zadatka, iz kojeg su određene potrebne relacije, atributi i enumeracije koje će biti potrebno implementirati. Određeno je 6 nužnih tablica - korisnici, ormari, lokacije, artikli korisnika, oglašivači i artikli oglašivača. Atributi navedenih određeni su kao oni definirani projektnim zadatkom te dodatni atributi nužni za ostvarivanje funkcionalnosti aplikacije (primarni i strani ključevi, opcija dijeljenja artikla, itd.). Nakon određivanja samih relacija, bilo je nužno odrediti i prikladan način njihovog povezivanja, tj. napraviti shemu. Shema je podijeljena u dva funkcionalna dijela. Prvi obuhvaća korisnike, ormare, lokacije te artikle korisnika. Relacije u ovom dijelu baze međusobno su povezane vezama 1-0..n. Primarni ključ korisnika je korisničko ime, dok ostale relacije imaju serijski primarni ključ (pri stvaranju novog entiteta primarni ključ je cijeli broj koji se poveća za 1). Slično je napravljeno i za drugi dio baze koji obuhvaća oglašivače i artikle oglašivača. Nakon što je dovršena konceptualna shema, prema njoj je napisan izvorni kod u SQL jeziku koji je pokrenut u PgAdmin programu kako bi se baza generirala. Provedeno je i otklanjanje sintaksnih grešaka u inicijalnom kodu nakon prvog pokretanja. Inicijalna verzija baze stvorena je na ‘localhost’ poslužitelju. Ona je ispitana dodavanjem vrijednosti u pojedine tablice i praćenjem jesu li te vrijednosti dobro povezane. Ovo je izvedeno SQL upitima za dodavanje i označavanje vrijednosti. Nakon ovog ispitivanja otklonjene su još eventualne semantičke pogreške.

Deployment baze podataka

Postavljanje baze podataka na AWS poslužitelj bilo je nužno kako bi se svim članovima tima (i kasnije samoj web aplikaciji) omogućio pristup. Prvi korak postupka bio je postavljanje AWS računa. Slijedeći korak bio je otvaranje nove baze podataka na AWS-RDS usluzi, sa svim postavkama s kojima je bio otvoren prototip u vidu lokalne baze (PostgreSQL...). Nakon završetka inicijalizacije, baza je bila aktivna na

AWS-u. Koristeći podatke zadane u inicijalizaciji (URL, port, korisničko ime i lozinka), napravljeno je spajanje na AWS-RDS kroz PgAdmin. Zatim je ponovno pokrenuta SQL skripta sa svim definiranim relacijama kako bi se ispitani prototip realizirao na poslužiteljskoj usluzi. Isti su podatci predani i ostalim članovima tima kako bi se bazi moglo pristupiti kroz backend. Završni rad na bazi tijekom ciklusa podrazumjevao je izmjene oko enumeracija i definicija tipova određenih atributa u dogovoru s timom na backend-u kako bi se omogućila lakša kompatibilnost.

Dokumentacija

Glede rada na dokumentaciji, moje je primarno zaduženje bilo pisanje dijela arhitekture sustava koji se tiče baze podataka. Uvodni dio opisao je korištene tehnologije za kreaciju, ispitivanje i postavljanja baze podataka na poslužitelj. Idući je zadatak bio detaljan opis tablica i atributa (tip podatka, dodatne definicije poput primarni ključ, strani ključ, ne-null vrijednost, jedinstvena vrijednost). Konačno, zadatak je bio izraditi sve relacijske sheme baze podataka u vidu grafičkih prikaza te pružati uz njih tekstulani opis. Ostala zaduženja glede dokumnetacije bila su vođenje dnevnika sastanaka i tjednog plana rada na projektu, što je bilo ažurirano sukladno zadatcima koji su nam svaki tjedan bili zadani na laboratorijskim vježbama.

Andrija Andročec

Arhitektura aplikacije

Tijekom prvog ciklusa izrade projekta, temeljito sam proučavao potencijalne tehnologije za izradu aplikacije te njihovu kompatibilnost. Naposljetku je odabrana kombinacija Reacta za klijentski te Springa za poslužiteljski dio aplikacije. Cjelokupna aplikacija sastoji se od dvije manje aplikacije (klijentske i poslužiteljske) koje međusobno komuniciraju HTTP zahtjevima, koja je ostvarena *reverse proxy* metodom. Prilikom komunikacije *reverse proxy* metodom korisnik akcijama na klijentskoj aplikaciji šalje zahtjeve poslužiteljskoj aplikaciji koja ih obrađuje, dohvaća potrebne podatke iz baze podataka, obrađuje dohvaćene podatke te ih prosljeđuje natrag klijentskoj aplikaciji koja ih prikazuje korisniku. Nadalje, kako bi se osigurala preglednost, organiziranost te skalabilnost aplikacije, poslužiteljska strana je dodatno podijeljena na tri osnovne cjeline: Controller, Service i Repository. Repository predstavlja software koji je korišten za spajanje na bazu podataka te dohvat podataka iz nje putem

SQL upita. Service preuzima podatke od Repositoryja te ih obrađuje i provodi poslovnu logiku nad njima prije nego što ih prosljeđuje zadnjoj fazi poslužitelja, Controlleru. Controller služi kao posrednik između klijentske i poslužiteljske aplikacije, obrađuje HTTP zahtjeve upućene poslužitelju te vraća obrađene podatke nazad na klijentski dio.

Spajanje poslužiteljskog dijela na bazu podataka

U svrhu spajanje poslužiteljskog dijela na bazu podataka napravljeni su klase (modeli) u programskom jeziku Javi, koji predstavljaju entitete i njihove relacije zapisane u bazi podataka. Za svaki od entiteta kreiran je zaseban Repository u svrhu ostvarenja CRUD funkcionalnosti prilikom pristupa bazi podataka, a programsko rješenje ostvareno je pomoću Spring Data JPA paketa. Zatim su nadodane Service klase koje posjeduju instancu pripadnog Repositoryja kao člansku varijablu putem koje dobivaju podatke iz Repositoryja (tj. baze podataka) te iste obrađuju. Naposljetku, u svrhu ispitivanja navedenih programa i njihovih funkcionalnosti, napravljeni su Controlleri za obradu zahtjeva za određenim podacima. No kako nam te funkcionalnosti nije bio cilj ostvariti u prvom ciklusu rada na projektu, ti Controlleri još nisu dovršeni.

Lana Benički

Povezivanje frontend i backend

Tijekom prvog ciklusa projekta, moje glavno zaduženje bilo je osigurati pravilno povezivanje frontend-a i backend-a. Krenula sam s proučavanjem tehnologija Spring Boot-a i React-a, a zatim s analizom implementacije koja je do tada bila napravljena na backendu i frontendu. Implementirala sam logiku unutar API ruta koja omogućava prijenos podataka s frontenda na backend i obrnuto, osiguravajući pritom da podaci budu sigurno pohranjeni i prikazani. Prilikom povezivanja bilo je potrebno prilagoditi sigurnosne postavke kako bi se omogućio pristup resursima s localhost:5173, gdje se frontend aplikacija razvija. Kako bih riješila ovaj problem, koristila sam CORS koji omogućava komunikaciju između localhost:8080 (backend) i localhost:5173 (frontend). Postavila sam dozvole za potrebne HTTP metode, kao i odgovarajuća zaglavlja. Prvo sam povezala implementacije za registraciju i prijavu bez većih problema. Nakon što su backend i frontend za prikaz artikala za neregistriranog korisnika bili dodani, spojila sam i njih. Dodala sam i prikazivanje

Google prijave na frontend te uredila prikaz Login.css kako bi se Google prijava uklopila u login formu. Na kraju sam radila na prikazu podataka o profilu na posebnoj stranici nakon što je korisnik ulogiran. Morala sam dodati neke funkcionalnosti na backendu kako bih to mogla implementirati. Dodala sam provjeru na backendu je li korisnik prijavljen putem Google računa ili putem standardne autentifikacije, te se na temelju toga prikazuju različiti podaci (google mail ili podaci koje korisnik unosi pri registraciji). Dodala sam podršku korisničkim ulogama na backendu, što je omogućilo bolju integraciju sa Spring Security. Dodala sam automatsko prijavljivanje korisnika nakon uspješne registracije te SecurityContext za novoautentificiranog korisnika koji je spremljen u sesiju putem HttpSession, što je riješilo problem prikaza podataka prethodnog korisnika nakon registracije. Implementirala sam dohvat podataka o korisniku na frontend te ažurirala stil profilne stranice. Dodala sam funkcionalnost na backend-u za odjavu korisnika koja briše korisničke podatke iz frontend aplikacije te resetira sesiju.

Dokumentacija

Poslala sam voditeljici tima dokument s opisom aplikacije i arhitekture sustava za dio koji sam ja radila, kako bi to mogla uključiti u dokumentaciju projekta.

Tanja Bertalanić

Vođenje tima

Početkom prvog ciklusa uspostavili smo zajedničke komunikacijske kanale koje sam kasnije moderirala i koristila za vođenje tima. Uspostavili smo discord kanal, napravili whatsapp grupu te sam, radi boljeg praćenja napretka našeg projekta, napravila Notion stranicu. Stranica na Notionu sadrži kalendar u koje sam upisivala važne datume i rokove te dio “Tasks” gdje sam pratila koji su zadatci krenuli s radom, koji još nisu te koji su završeni. Nakon svake laboratorijske vježbe, okupila sam tim za sastanak te smo razgovarali o daljnjem razvoju naše web aplikacije i tehničkim detaljima. Većina komunikacije odvijala se virtualno te nije bilo potrebe za duljim sastancima uživo. Svaki član tima je redovito izvršavao svoje obaveze te izvještavao o svome napretku i radu na projektu.

Frontend

Osim mog zaduženja kao voditelj tima, moj zadatak je također i razvoj poslužiteljske strane odnosno frontenda. Krenula sam s proučavanjem tehnologije React-a. Kolega Vujnović je postavio početnu React aplikaciju na kojoj sam kasnije razvijala frontend. Napravila sam forme za prijavu i registraciju. Pomoću react-router-dom omogućila sam preusmjeravanje na stranici. Organizirala sam i rasporedila komponente web aplikacije u odgovarajuće direktorije te sam vodila računa o kvaliteti i konzistentnosti koda na frontendu. Dodala sam funkcionalnost odjave te dodala prikaz modalnog prozora za artikle koji se nalaze na početnoj stranici. Surađivala sam s kolegicom Benički na povezivanju backenda i frontenda tako što bih dio posla, za određene funkcionalnosti, odradila na frontendu te bi to ona u konačnici objedinila sa backendom i nadogradila.

Git

Na kraju prvog ciklusa sam spojila razvojne grane(dev i devdoc) u glavnu granu(main).

Dokumentacija

Početkom prvog ciklusa sam napravila prvi template za wiki. Rapisala sam prvu verziju funkcionalnih i nefunkcionalnih zahtjeva. Detaljno sam opisala arhitekturu sustava. Ažurirala popis literature.

Iva Zubčić

Tijekom prvog ciklusa projekta, moje zaduženje bilo je poboljšati izgled korisničkog sučelja te omogućiti deploy aplikacije na produkcijsko okruženje. Prvo sam uredila CSS datoteke frontenda, stilizirala sam stranice za prijavu, registraciju. Pokušala sam deployati projekt pomoću platforme Render. Pripremila sam Docker konfiguracije za backend i frontend aplikaciju te postavila Nginx za posluživanje statičkih datoteka frontenda. Kroz GitHub Actions workflow, automatizirala sam proces deploya kako bi projekt bio spreman za kontinuiranu integraciju i dostavu. Iako su Docker i workflow konfiguracije bile ispravno postavljene, deploy na Renderu nije bio uspješan te smo na kraju odlučili umjesto Rendera koristiti Heroku.

Lovro Đuranec

Uspostavljanje Spring Security-a

U okviru projekta, moja glavna odgovornost bila je uspostavljanje Spring Security-a kao osnovnog sustava za autentifikaciju i autorizaciju korisnika. Za početak, proučena je dokumentacija Spring Security-a i primijenjeni su osnovni sigurnosni principi za aplikaciju. Prvi korak bio je konfiguriranje osnovne Spring Security postavke u aplikaciji. To je uključivalo implementaciju klase SecurityConfiguration u kojoj su definirana pravila za pristup različitim endpointima aplikacije. Klasa SecurityConfiguration konfigurira autentifikaciju putem korisničkog imena i lozinke, kao i definiranje URL putova koji su dozvoljeni korisnicima bez autentifikacije, poput login stranice. Pomoću ove konfiguracije omogućeno je korisnicima da se prijavljuju putem osnovnog obrasca (username/password) te da imaju pristup zaštićenim stranicama nakon uspješne prijave.

Inicijalna implementacija login/register sustava

Nakon uspostavljanja osnovnog sigurnosnog sustava, pristupilo se razvoju funkcionalnosti za prijavu i registraciju korisnika. Implementirani su temelji za formu za registraciju korisnika, koja omogućava unos korisničkog imena, email adrese i lozinke. Za registraciju korisnika, stvorena je posebna kontrolerska klasa koja obrađuje zahtjeve za registraciju i dodaje novog korisnika u bazu podataka. Korisničke podatke spremali smo pomoću Spring Data JPA-a u entitete User koji su se povezivali s tablicom u bazi podataka. Za prijavu korisnika, implementirali smo Spring Security obrazac za prijavu koji omogućuje unos korisničkog imena i lozinke. Nakon uspješne prijave, korisnik bi bio preusmjeren na profilnu stranicu.

Google OAuth2 login

Nakon što je osnovni sustav za prijavu i registraciju bio postavljen, implementirali smo mogućnost prijave putem Google OAuth2 servisa. Ovaj korak bio je ključan za olakšavanje prijave korisnicima koji već imaju Google račun, omogućujući im prijavu u aplikaciju bez potrebe za kreiranjem novog korisničkog računa. Postavljanje OAuth2 autentifikacije u Spring Boot aplikaciji zahtijevalo je integraciju s Google-ovim OAuth2 API-jem. Konfigurirali smo application.properties za pristup Google OAuth2 servisima, uključujući potrebne ključeve i tajne. Za prijavu putem Google-a, koristili smo Spring Security OAuth2

podršku. Implementirali smo CustomOAuth2UserService klasu koja preuzima podatke o korisniku nakon uspješne autentifikacije putem Google-a. Nakon toga, korisnici koji su se prijavili putem Google računa bili su preusmjereni na stranicu za profil, a sustav je automatski registrirao korisnika u bazi podataka ako je to bio njihov prvi put prijave. Za registraciju korisnika nakon prijave putem OAuth2, koristili smo postavke za povezivanje s postojećim korisnicima ili kreiranje novih korisničkih računa.

Pokušaj deployanja aplikacije putem Herokua

Pokušali smo deployati aplikaciju na platformu Heroku koristeći Github. Kako platforma nije prepoznavala naše promjene ispravno, odlučili smo pokušati drugim načinom. Pokušali smo implementirati deployment putem lokalnog terminala, koristeći Heroku CLI za ručno slanje aplikacije na platformu. Nažalost, ni ovaj pristup nije uspio zbog tehničkih poteškoća i različitih problema s build konfiguracijom koji su spriječili aplikaciju da ispravno funkcionira u Heroku okruženju.

Marko Vujnović

GitHub Wiki Dokumentacija

Održavao sam dokumentaciju projekta u GitHub Wikiju. Redovito sam ažurirao dokumente, dodavao nove informacije i osiguravao da cijeli tim ima pristup najnovijim verzijama dokumentacije. GitHub Wiki omogućio je praćenje promjena i održavanje ažuriranih podataka o razvoju sustava.

Opis Projekta i Analiza Zahtjeva

Radio sam i na opisu projekta i analizi zahtjeva. Tekst zadatka kojeg smo dobili je dobro obrađen pa nisam imao previše posla oko opisa projekta i analize zahtjeva.

Funkcijski Zahtjevi

Bio sam odgovoran za pisanje funkcijskih i nefunkcijskih zahtjeva, koji definiraju što sustav treba omogućiti korisnicima. Za svaki zahtjev sam detaljno opisao željenu funkcionalnost, uključujući sve potrebne uvjete za ispravan rad sustava, kao i način interakcije s korisnicima i drugim sustavima.

UML Dijagrami

Izradio sam sve potrebne UML dijagrame, uključujući dijagrame obrazaca uporabe, sekvencijske dijagrame i dijagrame razreda. Dijagrame sam crtao u programu Astah. Sve dijagrame sam crtao ručno. Dijagrame razreda sam i generirao iz java koda klasa i enumeracija.

Opis Obrazaca Uporabe

Za svaki obrazac uporabe koji je bio definiran, napisao sam detaljan opis koji objašnjava kako korisnici ili drugi sustavi komuniciraju s našim sustavom. Ovi opisi uključuju osnovne tokove aktivnosti, uvjete koji moraju biti ispunjeni za uspješan dovršetak svakog obrazca.

Izvještaj rada članova projekta - 2. predaja

Kao voditelj projekta, tražila sam svakog člana tima da mi ukratko opiše kako su pridonjeli projektu u drugom ciklusu. Sav njihov rad i napredak dokumetirani su na githubu putem commitova, issuesa i wikija. > Tanja Bertalanić, voditelj tima

Luka Bulić

Ispitivanje programskog rješenja

U ovom ciklusu projekta moje je glavno zaduženje bilo ispitivanje programskog rješenja. Ovo je podrazumijevalo ispitivanje na razini komponenti i ispitivanje na razini sustava. Ispitivanje na razini komponenti napravio sam koristeći Java biblioteku JUnit, koja pruža učinkovitu izradu ponovljivih ispita za ispitivanje pojedinačnih klasa. Ispitivane klase su uključivale klase nužne za ostvarivanje funkcionalnosti sustava, a to su korisnik, oglašivač, ormar, lokacija, artikl korisnika i artikl oglašivača. Napravljeno je ukupno 8 testnih slučajeva za navedene klase koji ispituju metode zaslužne za promjenu unutarnjih vrijednosti objekata te metode za ispis objekata. Što se tiče ispitivanja na razini sustava, tu sam koristio SeleniumIDE tehnologiju koja omogućuje ispitivanje aplikacije puštene u pogon u web

pregledniku. Provedeno je ukupno 15 ispitivanja koja pokrivaju većinu funkcionalnih zahtjeva projekta, te su prikazani uspješni i neuspješni ispiti koji su nam pomogli u popravku grešaka.

Baza podataka

U ovom ciklusu nastavio sam biti zadužen za vođenje baze podataka, ali je u ovoj fazi projekta tu bilo minimalno posla. Glavni zadatci su mi bili eventualni ispravci nekih atributa i čišćenje baze od entiteta napravljenih za ispitivanje.

Dokumentacija

Što se tiče dokumentacije, bio sam zadužen za kompletno dokumentiranje ispitivanja na razini komponenti i ispitivanja na razini sustava, koje sam prikazao u tabličnom obliku. Za testiranje komponenti jasno je navedena ispitivana klasa i metoda, ulaz, očekivani izlaz, stvarni izlaz i uspjeh testa, a za testiranje sustava navedeni su funkcionalni zahtjev, opis testa, ulaz, očekivani izlaz, stvarni izlaz i uspjeh testa. Kao i u 1. ciklusu, bio sam zadužemn za vođenje dnevnika aktivnosti grupe u vidu evidentiranja sastanaka i tjednog plana rada. Od ostalih dijelova dokumentacije, moje je zaduženje bilo napisati zaključak projekta, ključne izazove i rješenja, te prikazati dijagrame promjena.

Andrija Andročec

Implementacija funkcionalnosti za registrirane i neregistrirane korisnike s poslužiteljske strane

Moje glavno zaduženje u drugom ciklusu predaje bila je implementacija funkcionalnosti za registrirane i neregistrirane korisnike s poslužiteljske strane. Korištenjem prethodno ostvarenog spajanja poslužiteljskog dijela aplikacije na bazu ostvario sam API-je za pretraživanje artikala, dodavanje i brisanje ormara i lokacija, dodavanje i brisanje artikala te pretraživanje ormara.

Povezivanje frontenda i backenda

U suradnji s kolegama s klijentske strane radio sam na povezivanju klijentske i poslužiteljske strane, iz perspektive poslužitelja.

Lana Benički

Implementacija funkcionalnosti za oglašivače s poslužiteljske strane

U drugom ciklusu projekta bila sam zadužena za implementaciju funkcionalnosti vezanih za oglašivače na backendu. Kreirala sam modele, servise i repozitorije za upravljanje oglašivačima te prilagodila sustav za autentikaciju kako bi podržavao prijavu i registraciju oglašivača. Implementirala sam validaciju podataka pri registraciji kako bi se spriječilo korištenje postojećih e-mail adresa i korisničkih imena, uz prikaz odgovarajućih upozorenja. Dalje sam radila na funkcionalnostima za dodavanje, prikaz i brisanje artikala koje oglašivači mogu upravljati na svojim profilima. Omogućila sam prikaz svih oglašivača i njihovih artikala registriranim i neregistriranim korisnicima. Na kraju sam implementirala mogućnost dodavanja i prikaza slika logo-a oglašivača te slika artikala.

Povezivanje frontenda i backenda

U suradnji s Ivom Zubčić povezala sam frontend i backend za sve funkcionalnosti vezane za oglašivače.

Frontend

Unijela sam manje stilističke promjene na frontendu, poput dodavanja gumba za navigaciju kroz aplikaciju i dijelova obrazaca za upload slika.

Tanja Bertalanić

Deployment

U ovom ciklusu bila sam zadužena za deployment aplikacije koja nije bila deployana za prvu predaju. Odlučili smo deployati pomoću platforme Heroku. Aplikaciju sam puštala u pogon slijedno kada su se napravile veće promjene na frontendu i backendu.

Frontend

Nastavila sam s implementacijom ostalih funkcionalnosti što uključuje: pretraživanje artikala, dodavanje i brisanje ormara i lokacija, dodavanje i brisanje artikala, pretraživanje ormara, prijedlozi odjevnih kombinacija. CSS-om sam uredila napravljene funkcionalnosti. Omogućila sam responzivnost aplikacije.

Povezivanje frontenda i backenda

Osim samog izgleda korisničkog sučelja, bavila sam se povezivanjem frontenda i backenda prvenstveno sa strane frontenda. U suradnji s kolegama koji su radili na backendu sam omogućila dohvaćanje podataka s backenda na frontend za ranije navedene funkcionalnosti.

Dokumentacija

Opisala neke od korištenih tehnologija te napisala upute za puštanje aplikacije u pogon. Dodala sam ažurirani predložak za Wiki na Github Wiki.

Iva Zubčić

Tijekom drugog ciklusa rada na projektu, bila sam odgovorna za daljni razvoj frontend dijela aplikacije. Implementirala sam korisničko sučelje za profil oglašivača, uključujući funkcionalnosti poput prijave i registracije oglašivača te njihovog ormara za upravljanje sadržajem. Doradila sam headera na stranicama za registraciju i prijavu. Dodala sam novu listu oglašivača, koja omogućava prikaz svih profila oglašivača na jednom mjestu, te zasebne profile oglašivača s njihovim osnovnim informacijama. Svaka stranica je stilizirana korištenjem CSS-a.

Lovro Đuranec

Geolokacija i pretraživanje

Implementirao sam sustav dohvaćanja i spremanja korisničke lokacije preko preglednika i Google Geolocation API-ja. Funkcionalnost omogućuje dovoljno precizno praćenje trenutačne lokacije registriranih i neregistriranih korisnika, koja se koristi za filtriranje rezultata pretraživanja prema geografskoj blizini.

Integracija vremenske prognoze i Predlaganje odjevnih kombinacija

Uz geolokaciju, integrirao sam API za dohvaćanje vremenske prognoze na temelju lokacije korisnika. Podaci o vremenskim uvjetima (kiša, snijeg, temperatura) koriste se za funkcionalnost prijedloga odjevnih kombinacija. Na temelju tih podataka i dostupnih artikala korisniku, sustav predlaže primjer odjevne kombinacije za trenutačne vremenske uvjete.

Marko Vujnović

UML Dijagrami

Tijekom drugog ciklusa bio sam odgovoran za izradu svih UML dijagrama potrebnih za detaljan prikaz strukture i ponašanja sustava. Ova aktivnost uključivala je izradu dijagrama razreda, stanja, aktivnosti, komponenti i razmještaja. Svaki od dijagrama izrađen je ručno koristeći program Astah, koji omogućuje modeliranje i jednostavno upravljanje složenim sustavima.

Dijagrame razreda sam trebao generirati iz Java koda koji se koristi za modele na backendu.

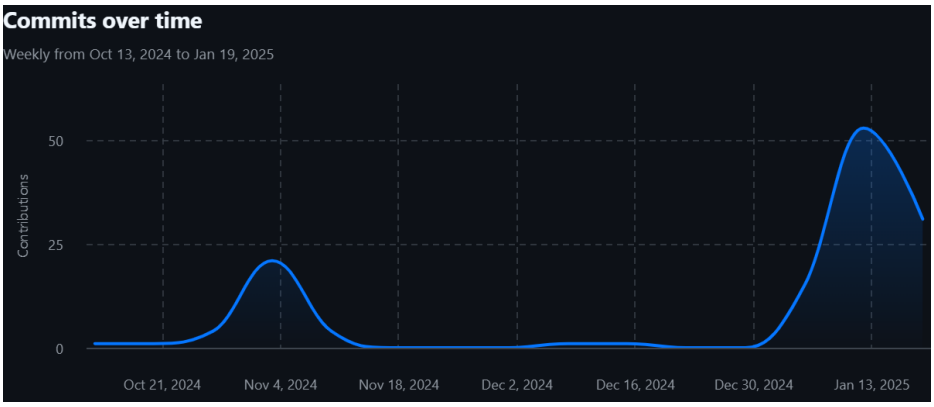
Dijagrami stanja korišteni su za prikaz ponašanja vidljivosti artikla unutar ormara i kako se tijekom nekih aktivnosti vidljivost mijenja.

Dijagrami aktivnosti prikazivali su tijek rada i procese unutar sustava i uvjetne odluke. Na ovaj način osigurana je jasna vizualizacija sekvenci koraka potrebnih za realizaciju osnovnih funkcionalnosti aplikacije.

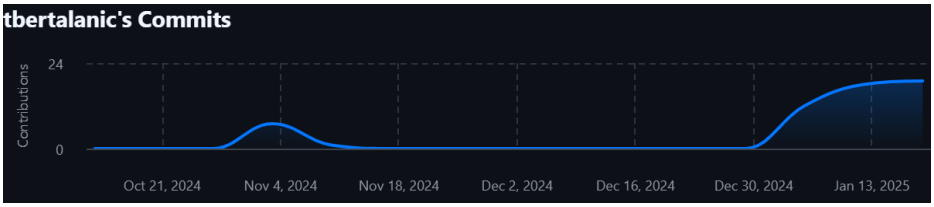
Dijagrami komponenti pružili su detaljan uvid u modularnu podjelu sustava, prikazujući kako su različiti dijelovi sustava organizirani i međusobno povezani.

S druge strane, dijagrami razmještaja omogućili su prikaz fizičke arhitekture sustava, uključujući protokole i uređaje na kojima se komponente sustava izvršavaju.

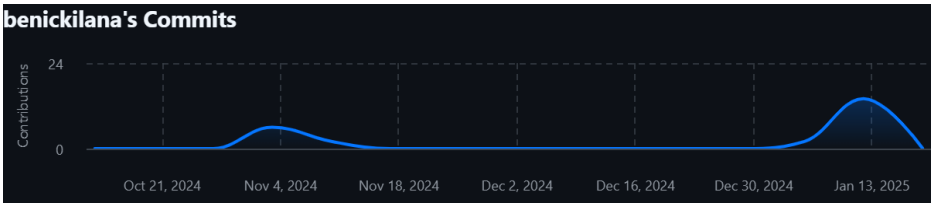
Dijagram pregleda promjena



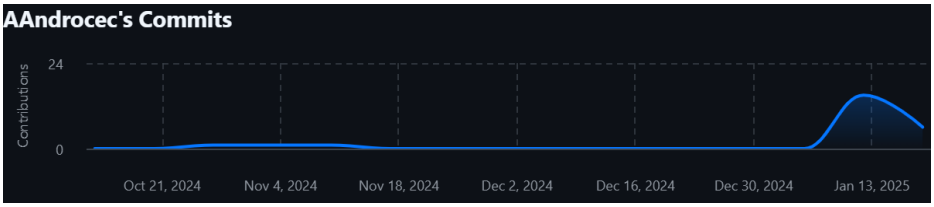
Commits over time



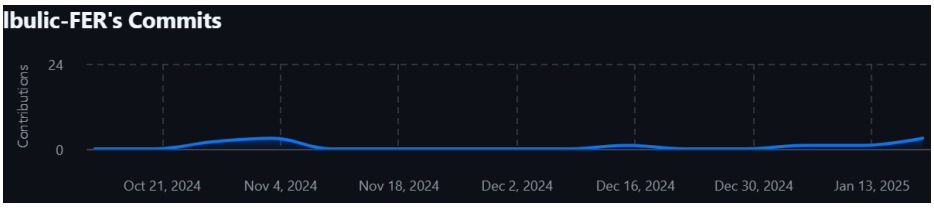
tbertalanic’s Commits



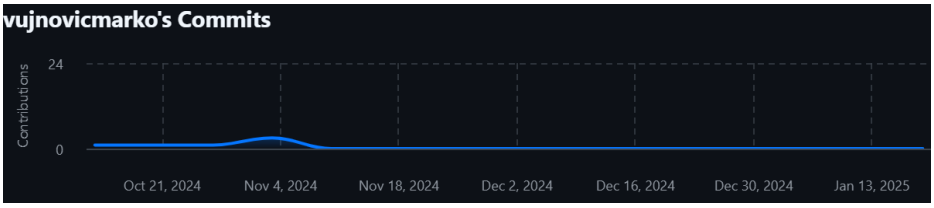
benickilana’s Commits



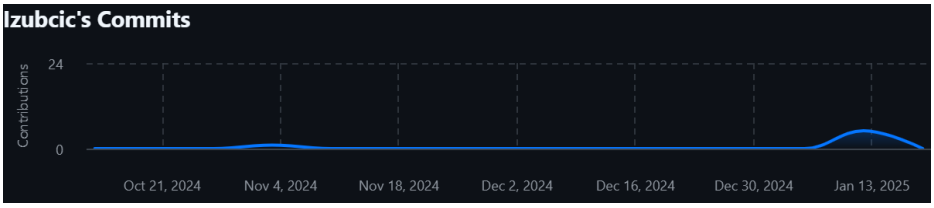
AAndrocec’s Commits



lbulic-FER’s Commits



vujnovicmarko’s Commits



Izubcic’s Commits

Kjučni izazovi i rješenja

- Ključni izazovi ovog projekta svodili su se na upoznavanje s novim tehnologijama i alatima te učinkovitom uspostavom komunikacije. S obzirom na više različitih aspekata projekta, od kojih je svaki zahtijevao primjenu 1-2 tehnologije, bilo je potrebno uložiti puno vremena u savladavanje njihove uporabe. Tu ubrajamo izradu programskog rješenja (Spring Boot, React, pgAdmin), ispitivanje (JUnit, Selenium) te vođenje projekta (Git, UML). Također, tim od sedam članova zahtijevao je visoku razinu koordinacije i učestalu komunikaciju.

- Problem savladavanja novih tehnologija riješen je učinkovitom podjelom posla među članovima. Na taj način je svatko imao dio opterećenja, čime se uštedilo cijelokupno vrijeme. Za učinkovitu komunikaciju uspostavili smo Discord server sa specijaliziranim kanalima za pojedine aspekte projekta (general, frontend, backend i dokumentacija). Također, tjedni sastanci koji su pružali feedback od strane nastavnika i priliku za sastajanje uživo bili su od pomoći.

Ormarko