

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Virtualni ormar

Tim: <TG03.2>

Ime tima: Portane

Nastavnik: Vlado Sruk

Asistent: Alan Jović

Opis Projektnog Zadatka: Virtualni Ormar

Uvod i cilj projekta

Projekt "Virtualni Ormar" osmišljen je kako bi korisnicima omogućio bolju organizaciju njihove odjeće i obuće putem digitalnog ormara, čime se smanjuje potreba za ručnim pretraživanjem fizičkih ormara. Cilj aplikacije je olakšati upravljanje velikim količinama odjevnih predmeta, omogućujući korisnicima brzi pregled i kategorizaciju, kao i dijeljenje odjeće s drugim korisnicima. Krajnji cilj je optimizirati organizaciju i uštedjeti vrijeme pri svakodnevnim odabirima odjevnih kombinacija, čime se povećava učinkovitost i zadovoljstvo korisnika.

Problem koji projekt rješava

Mnogi korisnici suočavaju se s izazovima prilikom organizacije ormara, posebno kada imaju veliki broj odjevnih predmeta, također bi htjeli podijeliti svoj ormar sa drugim korisnicima. Na tržištu već postoje aplikacije poput Stylebook, koje pružaju određeni stupanj organizacije ormara i planiranja odjevnih kombinacija. Za razliku od ostalih aplikacija Virtualni Ormar uključuje dodatne funkcionalnosti kao što su prilagodljiva struktura ormara prema korisničkim preferencijama, interaktivni prijedlozi odjevnih kombinacija te mogućnost dijeljenja i pretraživanja artikala među korisnicima.

Funkcionalnosti

Korisnici mogu pregledavati i pretraživati javne articke označene za dijeljenje prema različitim karakteristikama i prema geolokaciji. Za registraciju je potrebno unijeti:

- ime i prezime
- e-mail
- lozinku
- geolokaciju

Korisnici mogu dodavati i upravljati virtualnim ormarima, uključujući mogućnost izmjene strukture ormara, poput broja ladica. Također, mogu pretraživati svoje ormare prema karakteristikama artikla, uz prikaz broja ormara i točne lokacije artikla unutar ormara. Artikli odjeće ili obuće mogu se dodati u virtualni ormar, a svaki artikl mora imati definirane karakteristike.

Oglašivači se mogu registrirati i postaviti vlastitu galeriju artikala s logotipom koji vodi na galeriju. Galerija mora sadržavati slike, nazive, kategorije i cijene artikala. Registrirani korisnici mogu dobiti prijedloge odjevnih kombinacija na temelju zadatih kriterija, poput boje, aktivnosti ili vremenskih uvjeta. Aplikacija se može povezati s vanjskom uslugom vremenske prognoze kako bi automatski prikupila podatke o vremenskim uvjetima.

Postoji ograničenje na broj virtualnih ormara koje korisnici mogu kreirati, dok svaki ormar ima ograničenja u pogledu veličine i broja artikala koje mogu sadržavati komponente ormara.

Uloge korisnika na aplikaciji

Na aplikaciji postoje uloge:

- Neregistrirani korisnik
- Registrirani korisnik
- Oglašivač.

Neregistrirani korisnik

Neregistrirani korisnik ima mogućnost pregleda i pretraživanja artikala označenih za dijeljenje u virtualnim ormarima registriranih korisnika. Pri ulasku u aplikaciju, neregistrirani korisnik može pregledavati dostupne artikle u skladu s karakteristikama koje su postavili registrirani korisnici, kao što su tip odjeće, veličina i boja. Međutim, neregistrirani korisnik ne može dodavati vlastite artikle u ormar, dijeliti artikle ili koristiti dodatne funkcionalnosti koje su dostupne registriranim korisnicima.

Registrirani korisnik

Registrirani korisnik ima mogućnost kreirati i prilagoditi vlastiti virtualni ormar te definirati njegovu strukturu prema osobnim željama. Može dodavati artikle odjeće ili obuće u ormar, uključujući slike, nazive i kategorije. Također, omogućeno mu je pretraživanje vlastitih artikala prema svim karakteristikama. Na temelju unesenih informacija, korisniku će biti ponuđeni prijedlozi za odabir odjevnih kombinacija koje odgovaraju njegovim kriterijima. Registrirani korisnici mogu dijeliti svoje artikle s drugim korisnicima, čineći ih dostupnima na platformi. Također, imaju mogućnost ukloniti virtualni ormar, mijenjati njegovu strukturu ili ukloniti artikle koje više ne žele u svom ormaru.

Oglašivač

Oglašivač se mora registrirati na platformu kako bi mogao upravljati svojim profilom i galerijom artikala. Oglašivač može dodavati, uređivati ili uklanjati artikle, koji moraju sadržavati slike, naziv i kategoriju.

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvatanja
F-001	Pregled i pretraživanje javnih artikala	Visok	Zahtjev dionika	Neregistrirani korisnici mogu pregledavati i pretraživati articlne označene za dijeljenje prema svim karakteristikama i prema blizini trenutne geolokacije.
F-002	Detalji artikla i kontakt informacije	Visok	Zahtjev dionika	Neregistrirani korisnici mogu vidjeti detalje o artiklu i kontakt informacije korisnika koji je podijelio artikl.
F-003	Registracija korisnika	Visok	Zahtjev dionika	Neregistrirani korisnici mogu se registrirati kako bi izradili vlastiti virtualni ormar. Za registraciju potrebno je dati ime, prezime, e-mail i lozinku te geolokaciju.
F-004	Dodavanje ormara i definiranje strukture	Visok	Zahtjev dionika	Registirani korisnici mogu dodati virtualni ormar i prilagoditi njegovu strukturu (broj i vrsta lokacija: polica, ladica, šipka za odjeću).
F-005	Upravljanje virtualnim ormarima	Visok	Zahtjev dionika	Registirani korisnik može ukloniti vlastiti virtualni ormar ili izmjeniti njegovu strukturu (promjeniti broj ladica, polica ili šipki za odjeću) i uklanjati articlne koji se nalaze u ormaru.
F-006	Pretraživanje vlastitih ormara	Visok	Zahtjev dionika	Registirani korisnici mogu pretraživati vlastite ormare prema karakteristikama articla. Pretraživanje prikazuje broj ormara i točnu lokaciju (broj ladice, police ili šipke).
F-007	Dodavanje articla u virtualni ormar	Visok	Zahtjev dionika	Registirani korisnici mogu dodati articla odjeće ili obuće u svoj virtualni ormar.
F-008	Definiranje karakteristika articla	Visok	Zahtjev dionika	Korisnik za svaki artikl definira karakteristike kao što su naziv, slika, opću kategoriju (npr. kaput, jakna, košulja, haljina, cipele, tenisice...), kategoriju godišnjeg doba nošenja (npr. ljeto, proljeće, proljeće i jesen, zima), kategoriju otvorenosti (samo za obuću) (npr. otvoreno, zatvoreno, za kišu, za snijeg), kategoriju ležernosti (radno, za kuću, sportsko, ležerno, svečano), glavnu i sporednu boju articla, opis stanja articla te lokaciju u virtualnom ormaru.
F-009	Registracija oglasivača	Visok	Zahtjev dionika	Oglasivači se mogu registrirati kako bi postavili vlastitu galeriju articla i omogućili prikaz logotipa koji vodi na njihovu galeriju.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvatanja
F-010	Galerija oglašivača	Visok	Zahtjev dionika	Oglašivači mogu definirati galeriju artikala koji moraju sadržavati minimalno sliku, naziv, kategoriju i cijenu.
F-011	Prijedlog odjevne kombinacije	Visok	Zahtjev dionika	Registrirani korisnici mogu dobiti prijedlog odjevne kombinacije (popis artikala i njihova lokacija) na temelju kriterija koje moraju zadati (npr. koje boje, za koju aktivnost, kakvi su vremenski uvjeti).
F-012	Vremenska prognoza za kombinacije	Visok	Zahtjev dionika	Aplikacija se može povezati s vanjskom uslugom vremenske prognoze kako bi automatski dohvaćala podatke o vremenskoj prognozi.
F-013	Dijeljenje artikala	Visok	Zahtjev dionika	Registrirani korisnici mogu označiti artikle za dijeljenje, čime oni postaju javno dostupni za pretraživanje i pregled drugim korisnicima.
F-014	Ograničenje na broj virtualnih ormara	Visok	Zahtjev dionika	Registrirani korisnici mogu kreirati više virtualnih ormara (npr. max 10).
F-015	Ograničenja na strukturu virtualnog ormara	Visok	Zahtjev dionika	Svaki ormar ima ograničenu veličinu (npr. najviše 10 polica, ladica ili šipki za odjeću).
F-016	Ograničenja na veličinu komponenti	Visok	Zahtjev dionika	Svaka komponenta ormara ima maksimalni broj artikala koji su sadržani u njoj (npr. max 10 artikala po polici...)

Ostali zahtjevi

Skalabilnost

ID zahtjeva	Opis	Prioritet
NF-1.1	Aplikacija mora podržavati istovremeni rad velikog broja korisnika (minimum 100).	Visok
NF-1.2	Sustav podržava velik broj korisnika bez pada performansi.	Visok
NF-1.3	Baza podataka podržava velik broj artikala.	Visok

Pristupačnost i prilagodljivost

ID zahtjeva	Opis	Prioritet
NF-2.1	Web aplikacija prilagođena je za mobilne uređaje (responzivna).	Visok
NF-2.2	Aplikacija mora raditi na različitim preglednicima (Chrome, Safari, Firefox) i operativnim sustavima (iOS, Android, Windows)	Visok
NF-2.3	Sučelje mora biti intuitivno i lako za korištenje, omogućujući korisnicima jednostavan pristup svim funkcionalnostima aplikacije.	Visok

Zahtjevi za održavanje

ID zahtjeva	Opis	Prioritet
NF-3.1	Sustav treba biti oblikovan tako da omogućuje jednostavno održavanje.	Visok
NF-3.2	Sustav treba imati dovoljnu dokumentaciju.	Visok
NF-3.3	Sustav treba biti popraćen "Priručnikom za rad" koji opisuje pravilnu upotrebu sustava.	Visok
NF-3.4	Sustav treba imati "Plan implementacije" za pravilno postavljanje sustava.	Visok

Pouzdanost i performanse

ID zahtjeva	Opis	Prioritet
NF-4.1	Sustav mora biti dostupan korisnicima u svakom trenutku, uz minimalno vrijeme zastoja.	Visok
NF-4.2	Aplikacija treba omogućiti brzo učitavanje stranica čak i kod velikog broja artikala.	Visok

Sigurnost

ID zahtjeva	Opis	Prioritet
NF-5.1	Svi podaci između klijenta i servera trebaju biti preneseni putem sigurnih protokola (kao što je HTTPS) kako bi se zaštitili podaci od presretanja.	Visok
NF-5.2	Korisnici moraju imati mogućnost da odaberu koje informacije će dijeliti s drugima, uključujući dijeljenje artikala i geolokacije.	Visok
NF-5.3	Osobni podaci korisnika (kao što su ime, e-pošta i geolokacija) moraju biti zaštićeni u skladu s GDPR-om i sigurnosnim standardima.	Visok
NF-5.4	Pristup virtualnim ormarima, uređivanje i dijeljenje artikala treba biti ograničeno na ovlaštene korisnike putem sigurne autentifikacije i autorizacije.	Visok

Dionici

Dionik	Opis
Neregistrirani korisnik	Posjetitelji aplikacije koji mogu pregledavati i pretraživati artikle koji su označeni za dijeljenje. Imaju ograničen pristup dok se ne registriraju (F-001, F-002, F-003).
Registrirani korisnik	Korisnici koji su kreirali profil u aplikaciji. Mogu dodavati, uređivati i dijeliti artikle u svojim virtualnim ormarima te pretraživati javne i vlastite artikle (F-004, F-005, F-006, F-007, F-008, F-011, F-013).
Oglašivači	Kompanije ili pojedinci koji se oglašavaju u aplikaciji. Oni mogu postaviti galeriju svojih artikala i prikazati klikabilni logo koji vodi do njihovih proizvoda (F-009, F-010).
Razvojni tim	Osobe odgovorne za razvoj, održavanje i nadogradnju aplikacije. Oni implementiraju funkcionalne i nefunkcionalne zahtjeve.
Dizajnerski tim	Tim zadužen za korisničko iskustvo i vizualni dizajn aplikacije, osiguravajući intuitivno i responzivno sučelje za korisnike.
Naručitelji	Osobe koje su postavili zahtjev za izradom aplikacije te njezin opis.

Aktori

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:

- pregledavati i pretraživati javne artikle (F-001)
- vidjeti detalje o artiklu i kontakt-informacije o njegovom vlasniku (F-002)
- registrirati se za što je potrebno ime, prezime, e-mail i lozinka te geolokacija (F-003)

2. Registrirani korisnik (inicijator) može:

- prijaviti se u sustav putem e-mail adrese i lozinke
- kreirati virtualni ormar po željenoj strukturi (F-004, F-014), upravljati ormarima (brisati, mijenjati strukturu) (F-005, F-015)
- dodavati artikle u ormar te definirati njihovih karakteristika (F-007, F-008)
- dijeljenje artikala s drugim korisnicima (F-013)
- zahtjev izrade odjevne kombinacije po danim kriterijima (F-011)

3. Oglašivač (inicijator) može:

- prijaviti ili registrirati se u sustav (F-009) te definirati svoju galeriju s artiklima koja će se prikazivati drugim korisnicima (F-010)
- dodavati artukle u galeriju uz obaveznu prilaganje slike, naziva, kategorije i cijene artikla (F-010)

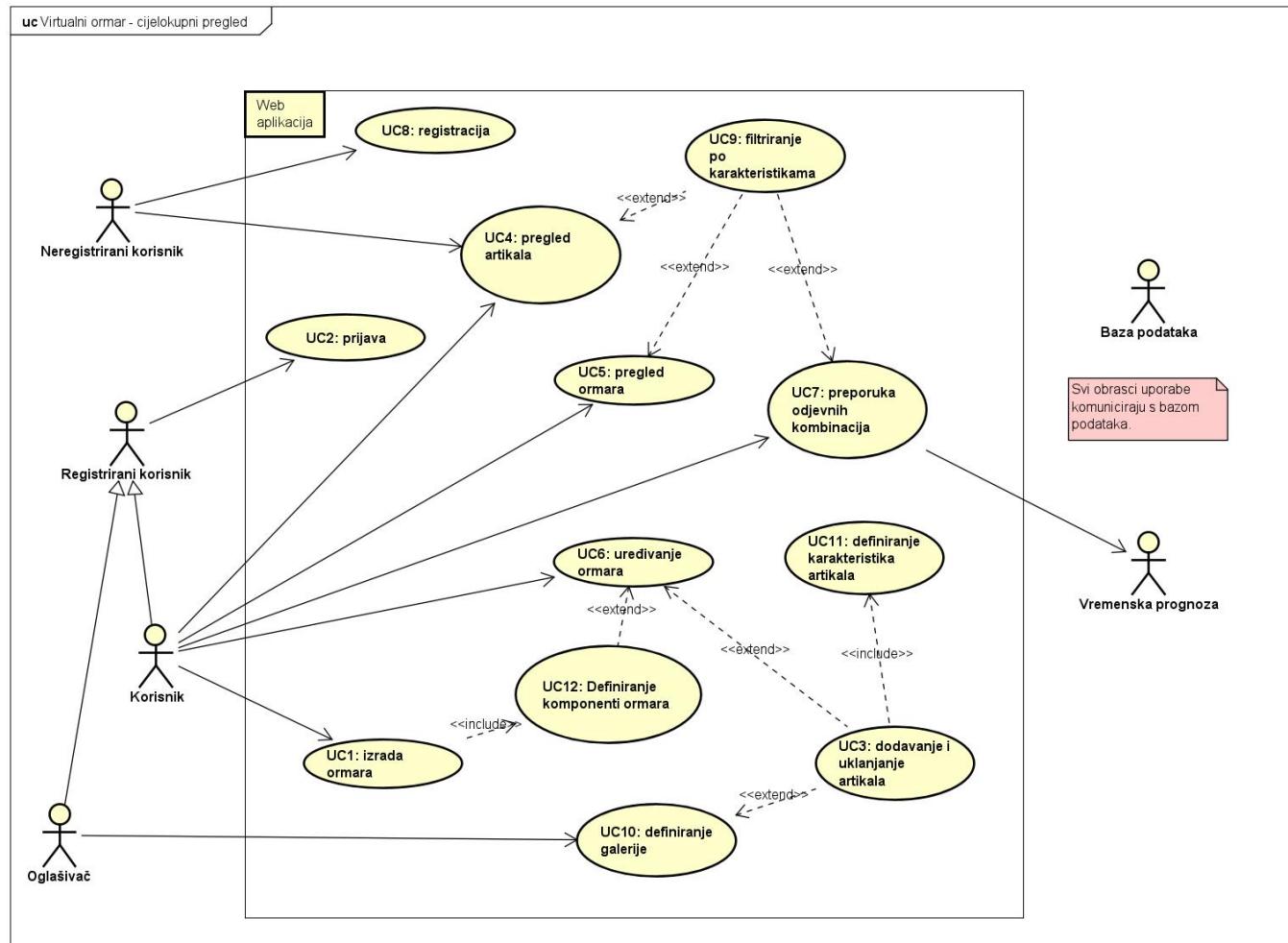
4. Baza podataka (sudionik):

- pohranjuje sve podatke o korisnicima i njihovim ormarima
- pohranjuje sve artikle i informacije o njihovim lokacijama
- pohranjuje dodatne statičke podatke ključne za rad sustava

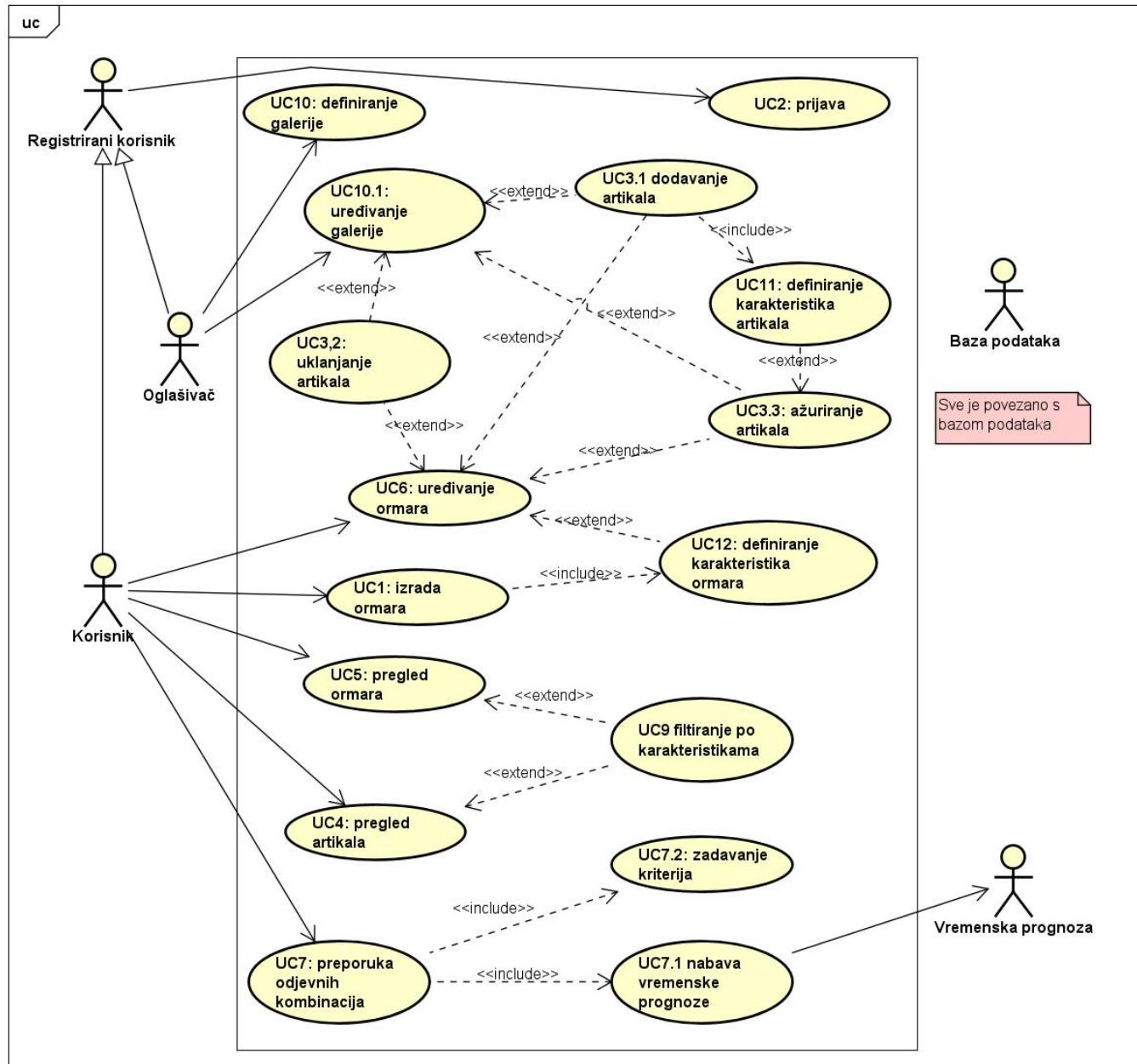
Obrasci uporabe

Opis obrazaca uporabe

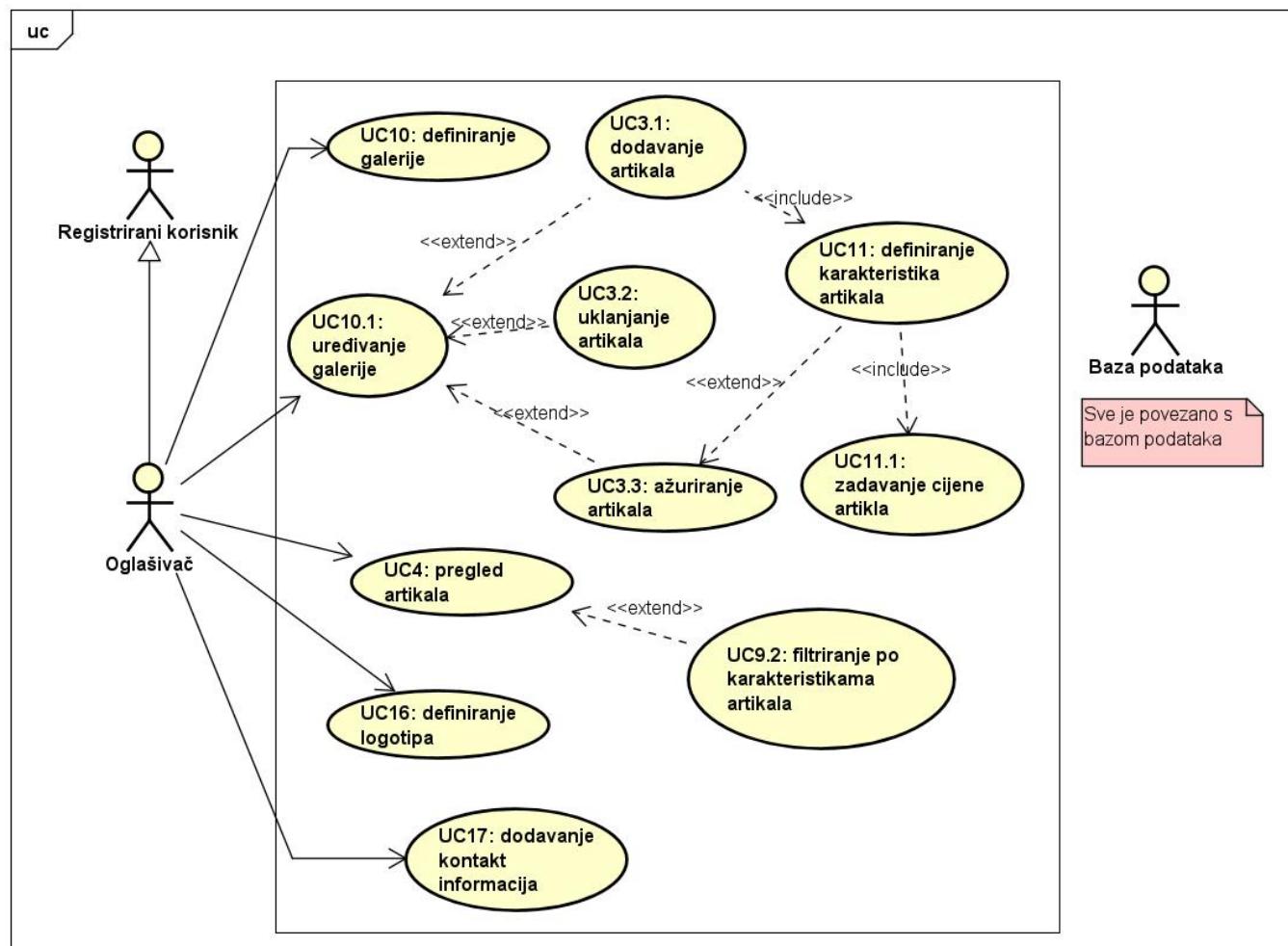
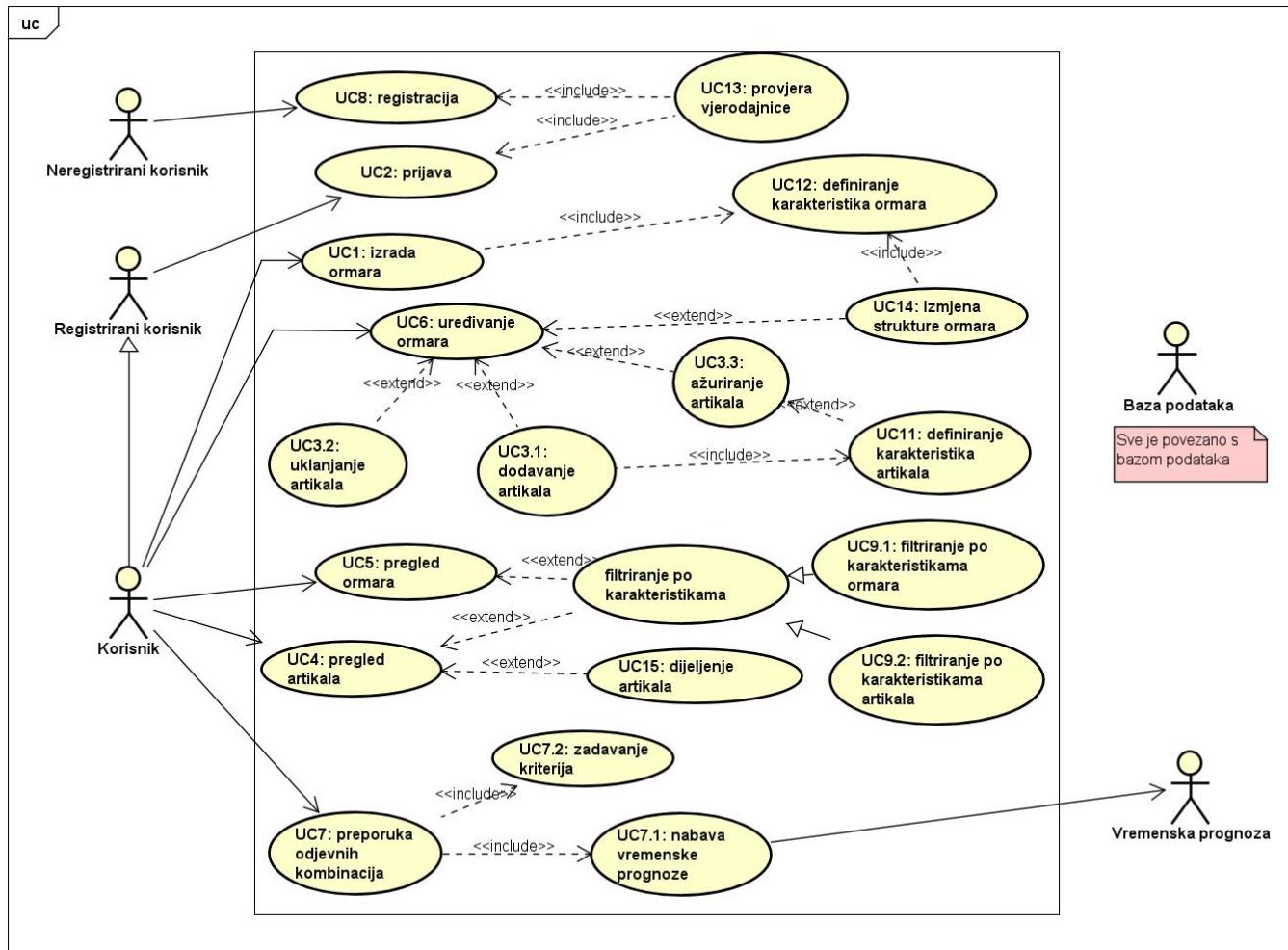
1. Visokorazinski dijagram obrazaca uporabe cijelog sustava



2. dijagram obrazaca uporabe za ključne funkcionalnosti



3. dijagram obrazaca uporabe za korisničke role



4. dijagram obrazaca uporabe za osnovne poslovne procese

5. dijagram obrazaca uporabe za kritične sustave i integracije

UC 1 - izrada ormara

- Glavni korisnik: Registrirani korisnik
- Cilj: Izrada virtualnog ormara
- Sudionici: registrirani korisnik, baza podataka
- Preduvjet: Korisnik prijavljen u sustav
- Opis osnovnog tijeka:

1. Korisnik započinje izradu ormara.
2. Korisnik definira strukturu ormara (UC 12).
3. Korisnik potvrđuje izradu ormara.
4. Ormar se zapisuje u bazu podataka.
5. Kreirani se ormar prikazuje korisniku.

- Opis mogućih odstupanja:

- 1.) Korisnik nije prijavljen u sustav.
- 2.) Greška u bazi podataka.
3. 2.) Korisnik zadao neispravan broj polica, ladica ili šipki.

UC 2 - prijava

- Glavni korisnik: Registrirani korisnik, oglašivač.
- Cilj: Prijava u web aplikaciju
- Sudionici: registrirani korisnik, oglašivač, baza podataka
- Preduvjet: Korisnik registriran u sustav
- Opis osnovnog tijeka:

1. Korisnik započinje prijavu unosom e-mail adrese i lozinke.
2. Sustav provjerava ispravnost podataka i njihovu prisutnost u bazi podataka.
3. Korisnik je prijavljen u sustav.

- Opis mogućih odstupanja:

- 2.) Korisnik nije registriran u sustavu.
- 1.) Korisnik zadao neispravne podatke za prijavu.
- 1.) Korisnik zaboravio lozinku

UC 3 - dodavanje i uklanjanje artikala

- Glavni korisnik: Registrirani korisnik, oglašivač.
- Cilj: Dodavanje novih ili uklanjanje postojeć artikala.
- Sudionici: registrirani korisnik, oglašivač, baza podataka
- Preduvjet: Korisnik prijavljen, moguće definiranje karakteristika artikala.
- Opis osnovnog tijeka:

1. Korisnik traži uklanjanje artikala iz ormara/galerije ili dodavanje artikala u ormar/galeriju.

2. Korisnik bira artikl za uklanjanje / definira karakteristike artikla za dodavanje.
3. Sustav ažurira bazu podataka
4. Promjena se prikazuje korisniku.

- Opis mogućih odstupanja:

1. 1.) Korisnik nije prijavljen u sustavu.
2. 2.) Korisnik zadao neispravnu karakteristiku artikla
3. 3.) Greška u bazi podataka.

UC 4 - pregled artikala

- Glavni korisnik: svi korisnici
- Cilj: Korisnici imaju omogućen pregled artikala koji su drugi korisnici podijelili.
- Sudionici: svi korisnici, baza podataka
- Preduvjet: -
- Opis osnovnog tijeka:

1. Pri ulazu u aplikaciju korisnicima se prikazuju podijeljeni artikli

- Opis mogućih odstupanja:

UC 5 - pregled ormara

- Glavni korisnik: Registrirani korisnik, neregistrirani korisnik
- Cilj: Korisniku je omogućen pregled svih njegovih ormara ili ormara koji su podijeljeni
- Sudionici: registrirani korisnik, neregistrirani korisnik, baza podataka
- Preduvjet: -
- Opis osnovnog tijeka:

1. Korisnik pristupa svojim ili s njime podijeljenim ormarima.
2. Filtrira sadržaj po karakteristikama artikala i ormara.
3. Podaci se dohvataju iz baze podataka.
4. Rezultat se prikazuje korisniku preko web aplikacije.

- Opis mogućih odstupanja:

1. 3.) Greška u bazi podataka.

UC 6 - uređivanje ormara

- Glavni korisnik: Registrirani korisnik.
- Cilj: Izmjena strukture virtualnog ormara (naziv, broj polica i ladica) te dijeljenje s drugim korisnicima.
- Sudionici: registrirani korisnik, baza podataka
- Preduvjet: Korisnik prijavljen, moguća izrada ormara, postoji barem jedan ormar.
- Opis osnovnog tijeka:

1. Korisnik traži promjenu strukture ormara ili njegovo brisanje.
2. Korisnik mijenja strukturu ormara (UC12) ili njegovo ime.
3. Korisnik bira hoće li učiniti svoj ormar javno dostupnim.
4. Korisnik bira hoće li dodati ili ukloniti artikle u svoj ormar (UC 3).

5. Sustav ažurira bazu podataka
6. Promjena se prikazuje korisniku.

- Opis mogućih odstupanja:

1. 1.) Korisnik nije prijavljen u sustavu.
2. 2.) Novi raspored neispravan.
3. 4.) Greška u bazi podataka.

UC 7 - preporuka odjevnih kombinacija

- Glavni korisnik: Registrirani korisnik.
- Cilj: Korisniku se, na temelju zadanih kriterija, preporuča odjevna kombinacija (popis artikala i njihovih lokacija).
- Sudionici: Registrirani korisnik, baza podataka, vanjski servis za vremensku prognozu.
- Preduvjet: Korisnik prijavljen, definiran barem jedan ormar.
- Opis osnovnog tijeka:

1. Korisnik traži izradu odjevne kombinacije i unosi kriterije (boja, vremenska prognoza, namjena...)
2. Sustav dohvaća podatke iz baze podataka i slaže prijedlog.
3. Prijedlog se prikazuje korisniku.

- Opis mogućih odstupanja:

1. 1.) Korisnik nije prijavljen u sustavu.
2. 1.) Korisnik nema definiranih ormara ili su prazni.
3. 2.) Greška u bazi podataka.
4. 2.) Prijedlog ne zadovoljava kriterije.

UC 8 - registracija

- Glavni korisnik: Neregistrirani korisnik.
- Cilj: Korisniku se, na temelju danog imena, prezimena, e-mail adrese i geolokacije, izrađuje korisnički račun.
- Sudionici: Neregistrirani korisnik, baza podataka.
- Preduvjet: -
- Opis osnovnog tijeka:

1. Korisnik daje tražene podatke.
2. Sustav provjerava ispravnost formata.
3. Sustav provjerava jedinstvenost podataka u bazi podataka (e-mail adresa nije već iskorištena).
4. Podaci se pohranjuju u bazu podataka.
5. Korisnik dobiva povratnu informaciju.

- Opis mogućih odstupanja:

1. 2.) Format podataka nije ispravan.
2. 3.) Postoji korisnički račun sa istim podacima.
3. 3.) Greška u bazi podataka.

UC 9 - filtriranje po karakteristikama

- Glavni korisnik: Svi korisnici
- Cilj: Korisniku može filtrirati pretragu tako što definira karakteristike artikala/ormara koji ga zanimaju.
- Sudionici: Svi korisnici, baza podataka.
- Preduvjet: -
- Opis osnovnog tijeka:

1. Korisnik zadaje karakteristike.
2. Sustav dohvaća podatke o artiklima i ormarima koji zadovoljavaju kriterije iz baze podataka.
3. Artikli i ormari se prikazuju korisniku.

- Opis mogućih odstupanja:

1. 2.) Greška u bazi podataka.

UC 10 - definiranje galerije

- Glavni korisnik: Ovlašivač.
- Cilj: Ovlašivač može definirati galeriju svojih artikala koji će se oglašavati drugim korisnicima.
- Sudionici: Ovlašivač, baza podataka.
- Preduvjet: Ovlašivač prijavljen u sustav.
- Opis osnovnog tijeka:

1. Ovlašivač započinje izradu vlastite galerije.
2. Daje informacije o vlastitim artiklima (sliku artikla, naziv, opću kategoriju i cijenu).
3. Podaci se spremaju u bazu podataka.
4. Kreirana se galerija prikazuje putem web aplikacije.

- Opis mogućih odstupanja:

1. 1.) Ovlašivač nije prijavljen u sustav.
2. 2.) Greška u danim podacima.
3. 3.) Greška u bazi podataka.

UC 11 - definiranje karakteristika artikala

- Glavni korisnik: Registriran korisnik, ovlašivač.
- Cilj: Korisnici mogu mijenjati karakteristike artikala koje su spremili u svoj ormar/galeriju.
- Sudionici: Registriran korisnik, ovlašivač, baza podataka.
- Preduvjet: Korisnik prijavljen u sustav.
- Opis osnovnog tijeka:

1. Korisnik odabire artikl kojem želi izmjeniti karakteristike.
2. Korisnik unosi nove podatke o artiklu.
3. Korisnik bira hoće li svoj artikl podijeliti s drugima (učiniti javno dostupnim).
4. Novi se podaci spremaju u bazu podataka.
5. Aplikacija prikazuje izmjenjeni artikl korisniku.

- Opis mogućih odstupanja:

1. 1.) Korisnik nije prijavljen u sustav.
2. 2.) Greška u danim podacima.
3. 4.) Greška u bazi podataka.

UC 12 - definiranje komponenti ormara

- Glavni korisnik: Registrirani korisnik
- Cilj: Korisnici mogu definirati strukturu svojih virtualnih ormara
- Sudionici: Registriran korisnik, baza podataka.
- Preduvjet: Korisnik prijavljen u sustav, postoji barem jedan virtualni ormar.
- Opis osnovnog tijeka:

1. Korisnik odabire virtualni ormar kojem želi izmjeniti strukturu.
2. Korisnik unosi nove karakteristike ormara (broj polica, ladica i šipki).
3. Novi se podaci spremaju u bazu podataka.

- Opis mogućih odstupanja:

1. 1.) Korisnik nije prijavljen u sustav, ne postoji barem jedan ormar.
2. 2.) Greška u podacima od korisnika.
3. 3.) Greška u bazi podataka.

UC 13 - provjera vjerodajnjice

- Glavni korisnik: Registrirani/neregistrirani korisnik
- Cilj: Provjera ispravnosti e-mail adrese i lozinke
- Sudionici: Registriran korisnik, neregistrirani korisnik, baza podataka.
- Preduvjet: -
- Opis osnovnog tijeka:

1. Korisnik unosi svoje podatke za prijavu/registraciju.
2. Sustav provjerava ispravnost danih podataka.
3. Sustav dojavljuje korisniku poruku o ispravnosti podataka.

- Opis mogućih odstupanja:

1. 2.) Dani podaci su neispravni (pogrešna lozinka/e-mail adresa)
2. 2.) Greška u bazi podataka.

UC 14 - izmjena strukture ormara

- Glavni korisnik: Registrirani korisnik
- Cilj: Izmjeniti strukturu postojećeg virtualnog ormara.
- Sudionici: Registriran korisnik, baza podataka.
- Preduvjet: postoji barem jedan ormar
- Opis osnovnog tijeka:

1. Korisnik odabire ormar kojem želi izmjeniti strukturu.
2. Korisnik odabire novu strukturu ormara (UC12).
3. Izmjene se prenose u bazu podataka.

- Opis mogućih odstupanja:

- 1.) Korisnik nema niti jedan virtualni ormari
- 2.) Korisnik nije prijavljen u sustav.
- 3.) Greška u bazi podataka.

UC 15 - dijeljenje artikala

- Glavni korisnik: Registrirani korisnik
- Cilj: Registrirani korisnik može označiti artikl koji će podijeliti s drugim korisnicima.
- Sudionici: Registriran korisnik, baza podataka.
- Preduvjet: postoji barem jedan artikl
- Opis osnovnog tijeka:

1. Korisnik odabire artikl koji želi podijeliti.
2. Izmjene se prenose u bazu podataka.
3. Podijeljeni artikli prikazuju se drugim korisnicima.

- Opis mogućih odstupanja:

- 1.) Korisnik nema niti jedan artikl
- 2.) Korisnik nije prijavljen u sustav.
- 3.) Greška u bazi podataka.

UC 16 - definiranje logotipa

- Glavni korisnik: Ovlašivač
- Cilj: Ovlašivač može definirati svoj logotip koji, kada korisnik na njega klikne, prikazuje galeriju i kontakt informacije ovlašivača.
- Sudionici: Ovlašivač, baza podataka.
- Preduvjet: -
- Opis osnovnog tijeka:

1. Ovlašivač unosi logotip.
2. Izmjene se prenose u bazu podataka.
3. Logotip se prikazuje korisnicima.

- Opis mogućih odstupanja:

- 1.) Ovlašivač nije prijavljen u sustav.
- 2.) Greška u bazi podataka.

UC 17 - dodavanje kontakt informacija

- Glavni korisnik: Ovlašivač
- Cilj: Ovlašivač može dodati svoje kontakt informacije (e-mail, telefonski broj, poveznice na vanjske stranice...)
- Sudionici: Ovlašivač, baza podataka.
- Preduvjet: -
- Opis osnovnog tijeka:

1. Ovlašivač unosi svoje kontrakt informacije
2. Izmjene se prenose u bazu podataka.
3. Kontakt informacije prikazuju se korisnicima.

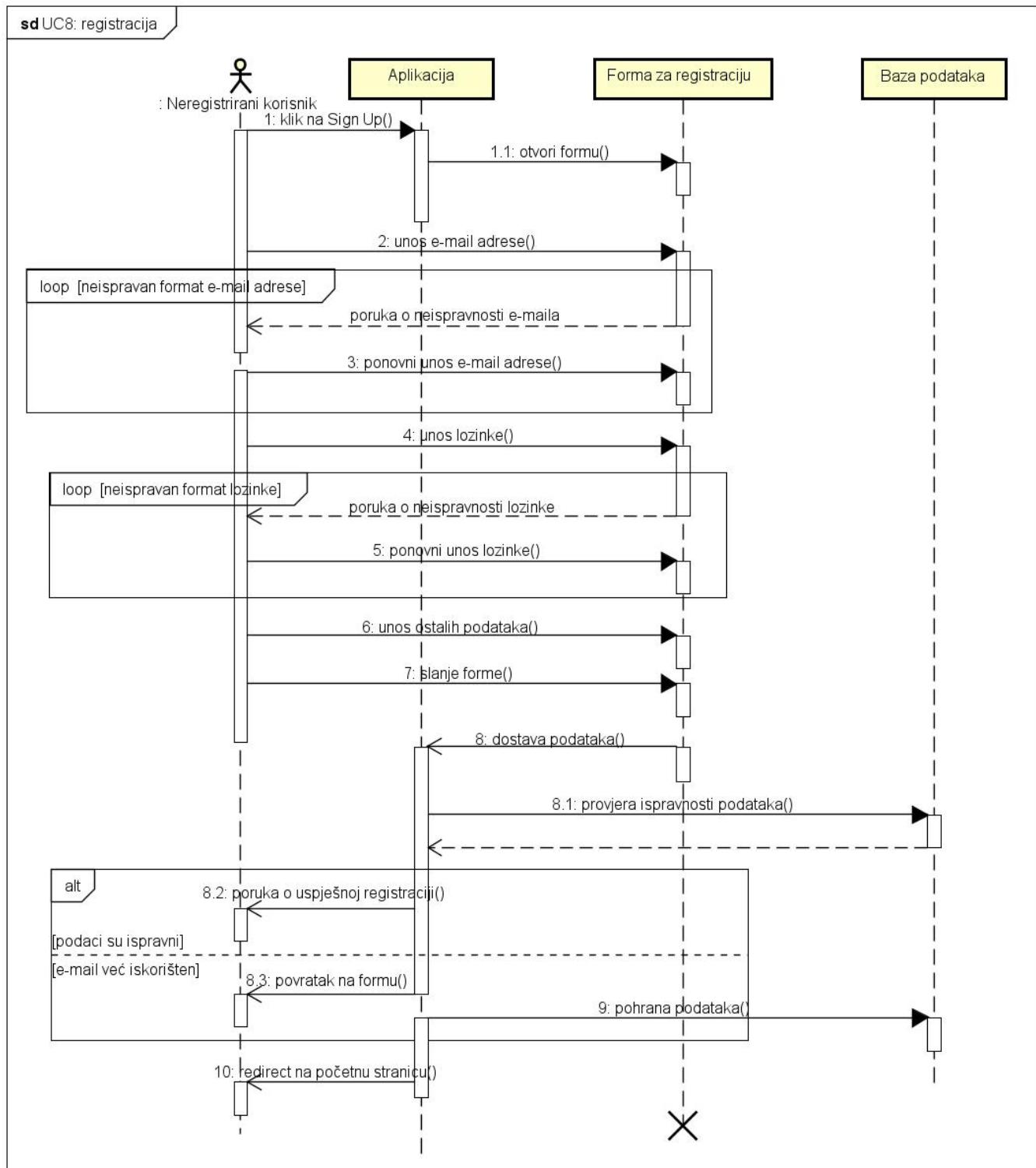
- Opis mogućih odstupanja:

1. 1.) Ovlašivač nije prijavljen u sustav.
2. 2.) Greška u bazi podataka.

Sekvencijski dijagrami

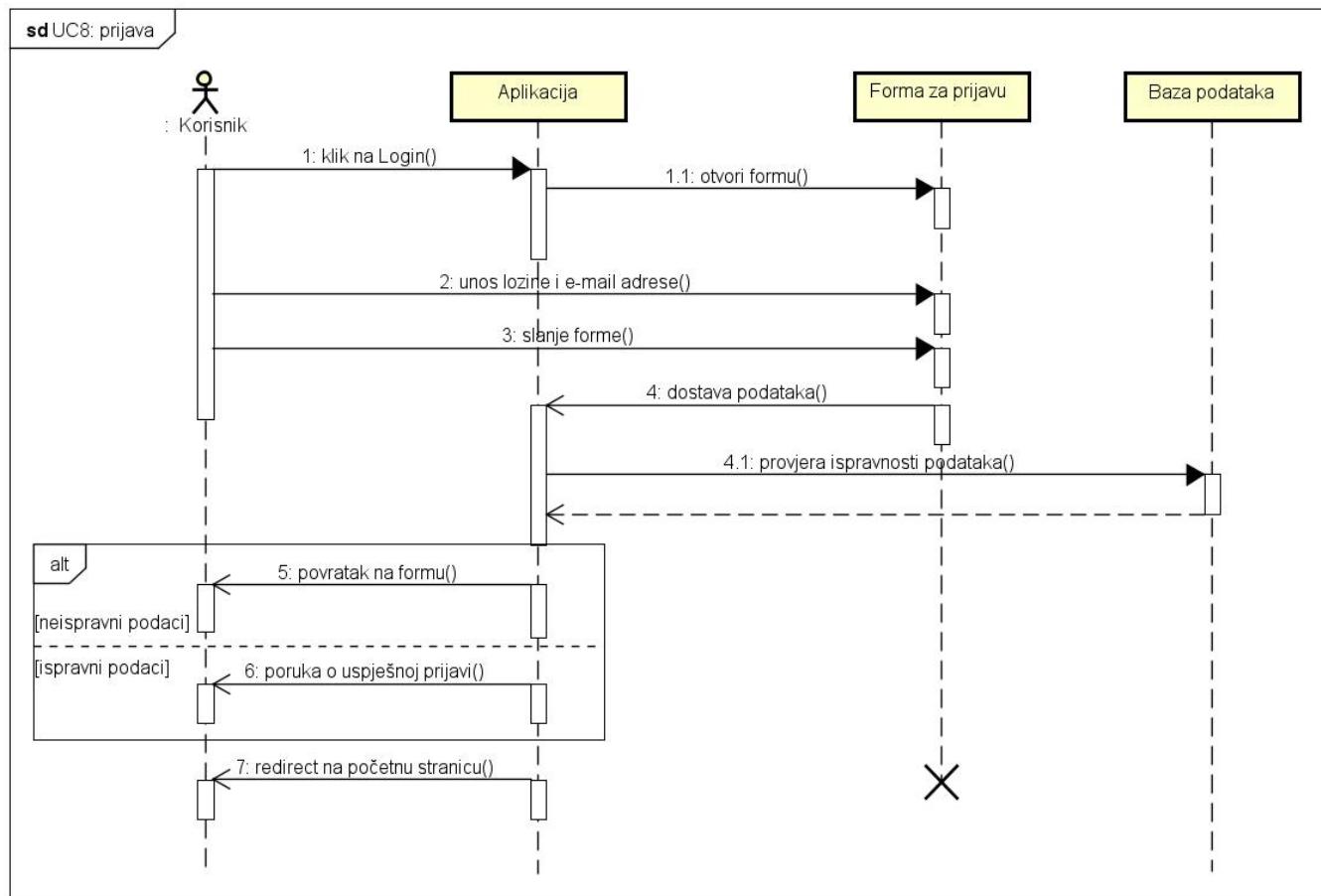
UC 8 - registracija

Neregistrirani korisnik započinje registraciju klikom na gumb *Registrate*. Nakon toga odlazi na stranicu sa formom za registraciju. Korisnik mora upisati svoje ime, prezime, e-mail adresu te lozinku. Sva su polja obavezna. E-mail adresa mora zadovoljavati formu *ime@domena* te biti jedinstvena u sustavu (niti jedan drugi korisnik nema istu adresu). Lozinka također mora poštivati definirani format (zadan u nefunkcionalnim zahtjevima). Korisnik pokušava upisati tražene podatke sve dok ne uspije zadovoljiti format. Nakon toga se jedinstvenost e-mail adrese provjerava u bazi podataka te se, ovisno o rezultatu provjere, korisnik preusmjerava na početnu stranicu aplikacije ili ponovno na formu za registraciju sa prigodnom porukom koja objašnjava do koje je greške došlo.



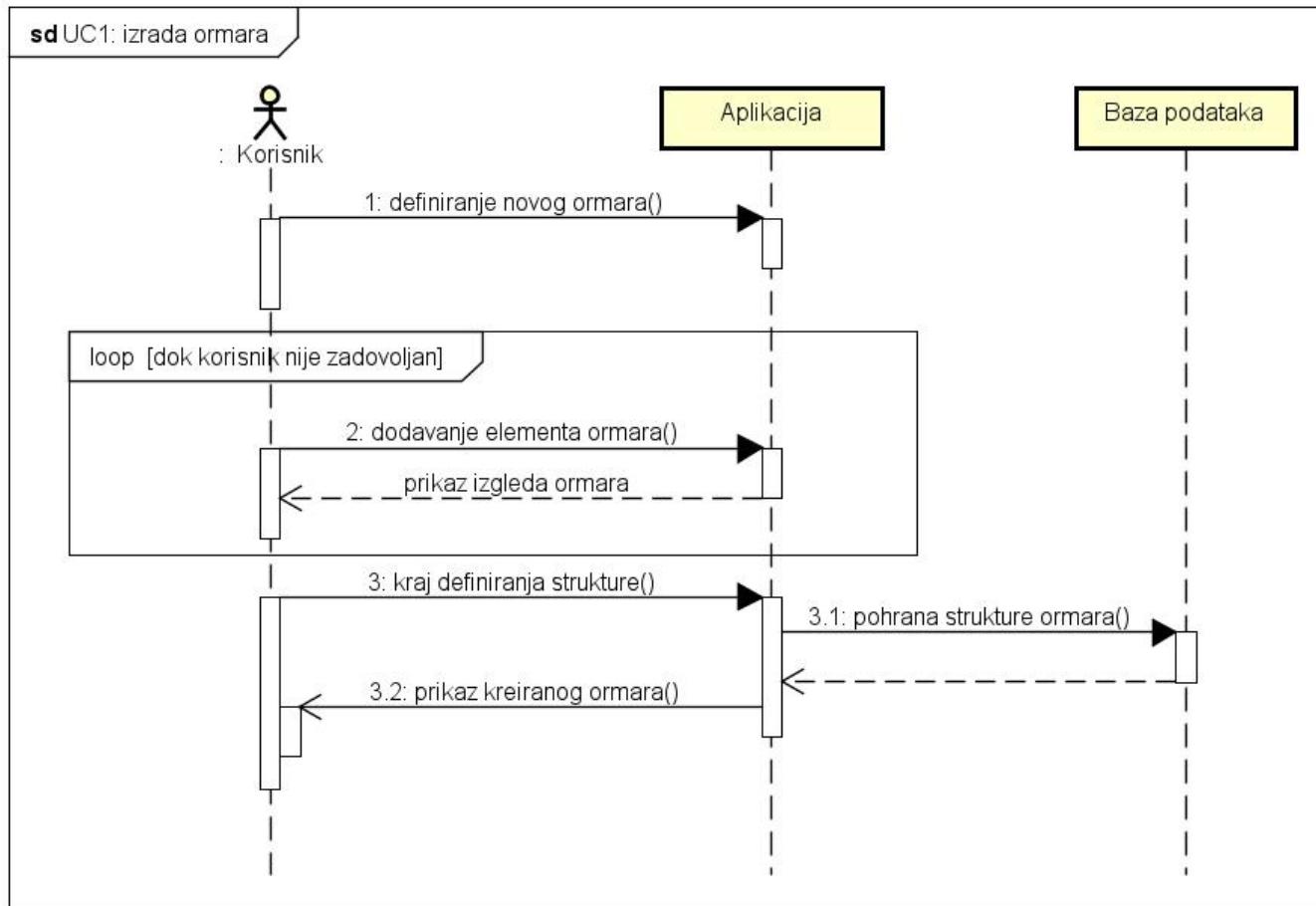
UC 2 - prijava

Registrirani korisnik započinje prijavu klikom na gumb *Log In*. Nakon toga odlazi na stranicu sa formom za prijavu. Korisnik mora unijeti svoju e-mail adresu te lozinku. Sva su polja obavezna. E-mail i lozinka se nakon toga pretražuju u bazi podataka i, ukoliko je potvrđena njihova ispravnost, korisnik se preusmjerava na početnu stranicu.



UC 1 - izrada ormara

Korisnik započinje izradu novog ormara za što je potrebno ime ormara, broj polica, ladica i šipki te geolokacija ormara. Kada je korisnik zadovoljan definiranom strukturu ona se pohranjuje u bazu podataka te se korisniku prikazuje kreirani ormari.



Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

Arhitektura sustava

Opis arhitekture

Backend

Aplikacija koristi Spring Boot (3.3.5) framework u jeziku Java (21) za backend.

Arhitektura backenda modelirana je klasičnim Spring MVC principom koji se sastoji od sljedećih slojeva (komponenti):

Model

Model predstavlja podatke koje aplikacija koristi i manipulira. Ovi podaci se mapiraju na entitete baze podataka. Definirani su pomoću Java klasa anotiranih s JPA (Java Persistence API) anotacijama.

Repositorij

Repositorij je sloj koji se koristi za komunikaciju s bazom podataka. Spring Data JPA olakšava kreiranje repositorija pomoći jednostavnog sučelja koje nasljeđuje JpaRepository.

Servis

Servisni sloj obavlja poslovnu logiku aplikacije. On koristi repozitorij za pristup podacima i obavlja složene operacije nad modelima.

Fasada

Fasada je dizajnerski obrazac koji pojednostavljuje pristup složenim sistemima i služi kao jedinstvena ulazna točka za klijente. U kontekstu Spring Boot MVC aplikacija, fasada se često koristi za integraciju više servisa ili dodavanje dodatne apstrakcije te mapiranje JPA modela u Dto-e (Data Transfer Object) prikladne za frontend prikaz i mapiranje podataka s frontenda (Form) u model.

Kontroler

Kontroler prima HTTP zahtjeve, poziva fasadni sloj da obradi podatke i vraća odgovore klijentima.

Koristi se Maven za upravljanje i generiranje projekta, Liquibase za kontroliranje verzija baze podataka.

Frontend

Aplikacija koristi React js library i bootstrap za css.

Koriste se sve uobičajene dobre prakse pri razvoju u React frameworku, npr. dizajn komponenti koje se mogu ponovno koristiti te podjela koda u prikladne direktorije Aplikacija je podijeljena na prikladne url-ove kako bi korisničko iskustvo bilo što intuitivnije.

Obrazloženje odabira arhitekture

Odabir Spring Boot frameworka i Java za backend temeljio se na njihovoj stabilnosti, skalabilnosti i bogatom ekosustavu. Spring Boot omogućava brz razvoj aplikacija s minimalnom konfiguracijom, što ubrzava razvojni proces i olakšava održavanje.

Zbog ograničenja besplatne verzije Rendera, privremeno je odabrana H2 baza podataka s PostgreSQL dijalektom kako bi se simuliralo produkcijsko okruženje. U produkciji, PostgreSQL bi bio korišten zbog svoje pouzdanosti, robusnosti i podrške za napredne funkcionalnosti.

Organizacija sustava na visokoj razini

Sustav je organiziran u klijent-poslužitelj arhitekturu. Klijent (*frontend*) prikazuje podatke korisniku te šalje zahtjeve poslužitelju (*backendu*). Backend prima zahtjeve, obrađuje ih te šalje odgovore frontendu koji ih onda prikazuje korisniku.

Frontend je React aplikacija koja se pokreće u web pregledniku. React aplikacija generira UI (User Interface) koji preglednik prikazuje korisniku te šalje zahtjeve *backendu*.

Backend je Spring Boot aplikacija na serveru koja prima zahtjeve, obrađuje ih te šalje odgovore klijentu koji je postavio zahtjev.

Backend koristi PostgreSQL bazu podataka u kojoj su spremljeni svi podaci potrebni za ispravan rad aplikacije.

Tipičan tijek podataka između klijenta i poslužitelja:

1. Korisnička interakcija (na klijentu):

Na primjer, korisnik pritisne gumb na stranici.

2. Slanje zahtjeva (klijent prema poslužitelju):

React aplikacija šalje *HTTPS* zahtjev prema poslužitelju. Zahtjev može biti za dobavljanje, mijenjanje, brisanje ili slanje podataka.

3. Obrada zahtjeva (na poslužitelju):

Poslužitelj prima zahtjev, procesira ga i vraća odgovor.

4. Obrada odgovora (na klijentu):

Klijent prima odgovor od poslužitelja. Ako je to *JSON* odgovor, React aplikacija koristi te podatke za ažuriranje korisničkog sučelja (view).

5. Prikaz podataka (na klijentu):

Na temelju ažuriranog viewa, preglednik prikazuje novi izgled sučelja.

Organizacija aplikacije

Sustav je organiziran u klijent-poslužitelj arhitekturu. Klijent (*frontend*) se sastoji od *web preglednika* koji je zadužen za slanje *HTTPS* zahtjeva (GET/POST/PUT/DELETE) za traženim resursima te prikaz stranice, React aplikacije koja, koristeći React komponente, generira view koji preglednik prikazuje (*rendera*) korisniku. *Frontend* šalje *HTTPS* zahtjeve prema poslužitelju (*backendu*), a za slanje zahtjeva koristi se *Axios* biblioteka. Zahtjevi se šalju putem *REST API-ja* kako bi se dohvatali potrebni podaci (npr. popis ormara ili artikala), poslali novi podaci (npr. kreiranje ormara ili artikala), ažurirali postojeći podaci (npr. promjena imena ili strukture ormara) te brisali podaci (npr. uklanjanje artikala iz ormara).

Poslužitelj (*backend*) organiziran je *Spring MVC* principom. Modeli predstavljaju podatke koji se spremaju u bazu podataka (npr. informacije o korisniku, ormaru, artiklu...), dok kontroler zaprima korisničke zahtjeve (*HTTPS* zahtjevi s *frontenda*) te poziva fasadu koja obrađuje podatke. Nakon što se podaci obrade oni se šalju *web pregledniku* (u *JSON* formatu) u kojem React aplikacija generira novi view koji se onda prikazuje korisniku (NF-2.2).

Fasada je sloj koji pruža jednostavniji API za složene operacije koje uključuju interakciju s više servisa. Fasada pomaže smanjenju ovisnosti između kontrolera i implementacijskih detalja servisa.

Sloj servisa koristi se za implementaciju poslovne logike aplikacije. Servisi su zaduženi za komunikaciju s repozitorijem (koji pristupa bazi podataka) i za vršenje složenih operacija ili transformacija podataka.

Repozitorij je sloj koji pristupa bazi podataka i obavlja *CRUD* (Create, Read, Update, Delete) operacije. U ovoj je aplikaciji za ostvarenje repozitorija korišten *Spring Data JPA*. Repozitorij komunicira s bazom podataka putem entiteta (klasa koje odgovaraju tablicama u bazi podataka), a koji su zapravo modeli u ovoj aplikaciji.

Koristi se *PostgreSQL* relacijska baza podataka u koju se spremaju svi podaci ključni za ispravan rad aplikacije.

Aplikacija je povezana s vanjskim servisom za dobavljanje vremenske prognoze koja je potrebna za lakšu izradu prijedloga odjevnih kombinacija (F-011).

Prednosti

Brza razvojna produktivnost: Odvajanje frontenda i backenda te korištenje MVC principa omogućuje različitim timovima paralelan rad na različitim uslugama aplikacije, koristeći optimalne tehnologije za svaki dio aplikacije. Korištenje tehnologija kao što su React, Axios te Spring Boot pruža pristup unaprijed konfiguriranim rješenjima koja olakšavaju razvoj aplikacije.

Modularnost i održavanje: Svaki sloj ima jasno definiranu odgovornost što omogućuje bolju organizaciju koda i lakše održavanje te jednostavnije praćenje promjena.

Testabilnost i otpornost: Svaki se sloj može neovisno testirati što olakšava pronađazak i ispravljanje pogrešaka. Također, neovisnost slojeva znači da pogreška u jednom od njih ne utječe na cijeli sustav.

Skalabilnost: Razdvajanje aplikacije na slojeve omogućuje da se svaki sloj može skalirati neovisno o drugim dijelovima aplikacije.

Baza podataka

Koristi se PostgreSQL baza podataka. Baza je podijeljena u dvije sheme. Shema *public* sadrži tablice koje će se mijenjati na zahtjev korisnika, a shema *lut* tablice koje sadrže statičke podatke koji su neophodni za rad aplikacije. U tablici *public.users* pohranjeni su podaci o imenu, prezimenu, e-mail adresi i lozinki korisnika. Tablica *public.closets* sadrži podatke o nazivu i vlasniku ormara, a tablica *public.closet_closet_components* povezuje ormara sa njegovim sastavnim dijelovima. Tablica *public.articles* sadrži informacije o nazivu, slici i opisu artikala, njegovu lokaciju u ormaru te kojim kategorijama artikl pripada. SQL kod za generiranje baze: https://github.com/borisspanic/portane/blob/devdoc/Dokumentacija/baza%20podataka/Portane_db.sql

Opis tablica

public.articles

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator artikla (primarni ključ, sekundarni ključ za sifr.stil)
label	varchar	Naziv artikla
picture	varchar	Put do slike
public	bool	true ako je artikl podijeljen s drugim korisnicima
closet_component_id	integer	Identifikator dijela ormara u kojem se artikl nalazi (sekundarni ključ za <i>public.closet_closet_component</i>)
category_id	integer	Identifikator opće kategorije kojoj artikl pripada (sekundarni ključ za <i>lut.categories</i>)
condition_id	integer	Identifikator stanja u kojem se artikl nalazi (sekundarni ključ za <i>lut.conditions</i>)

Atribut	Tip podatka	Opis varijable
primary_color_id	integer	Identifikator primarne boje artikla (sekundarni ključ za lut.colors)
secondary_color_id	integer	Identifikator sekundarne boje artikla (sekundarni ključ za lut.colors)
active	bool	true ako je artikl aktivan
created_at	timestamp	Vremenska oznaka unosa artikla u bazu
last_updated	timestamp	Vremenska oznaka zadnje izmjene artikla

public.closets

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator ormara (primarni ključ)
title	varchar	Naziv ormara
user_id	integer	Identifikator korisnika ormara (sekundarni ključ za public.users)
active	bool	true ako je ormar aktivan
created_at	timestamp	Vremenska oznaka unosa ormara u bazu
last_updated	timestamp	Vremenska oznaka zadnje izmjene ormara

public.users

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator korisnika (primarni ključ)
firstname	varchar	Ime korisnika
lastname	varchar	Prezime korisnika
email	varchar	Email adresa korisnika
password	varchar	Zaporka
active	bool	true ako je korisnik aktivan
created_at	timestamp	Vremenska oznaka unosa korisnika u bazu
last_updated	timestamp	Vremenska oznaka zadnje izmjene podataka korisnika

public.closet_closet_components

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator para (ormar, dio ormara) (primarni ključ)

Atribut	Tip podatka	Opis varijable
closet_id	integer	Identifikator ormara (sekundarni ključ za public.closets)
closet_component_id	integer	Identifikator dijela ormara (sekundarni ključ za public.closet_components)
quantity	integer	Broj dijelova ormara u danom ormaru

public.vendor_articles

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator artikla oglavlivača (primarni ključ)
vendor_id	integer	Identifikator oglavlivača koji je vlasnik artikla (sekundarni ključ za public.vendors)
article_id	integer	Identifikator artikla (sekundarni ključ za public.articles)
cost	float	Cijena artikla

public.users

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator oglavlivača (primarni ključ)
name	varchar	Ime oglavlivača
logo	varchar	Putanja prema slici logotipa oglavlivača
active	bool	true ako je oglavlivač aktivan
created_at	timestamp	Vremenska oznaka unosa oglavlivača u bazu
last_updated	timestamp	Vremenska oznaka zadnje izmjene podataka oglavlivača

lut.closet_components

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator dijela ormara (primarni ključ)
label	varchar	Naziv dijela ormara

lut.styles

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator stila (primarni ključ)
name	varchar	Naziv stila

lut.footwear_types

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator otvorenosti obuče (primarni ključ)
name	varchar	Naziv otvorenosti

lut.seasons

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator godišnjeg doba (primarni ključ)
name	varchar	Naziv godišnjeg doba

lut.categories

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator kategorije (primarni ključ)
name	varchar	Naziv kategorije

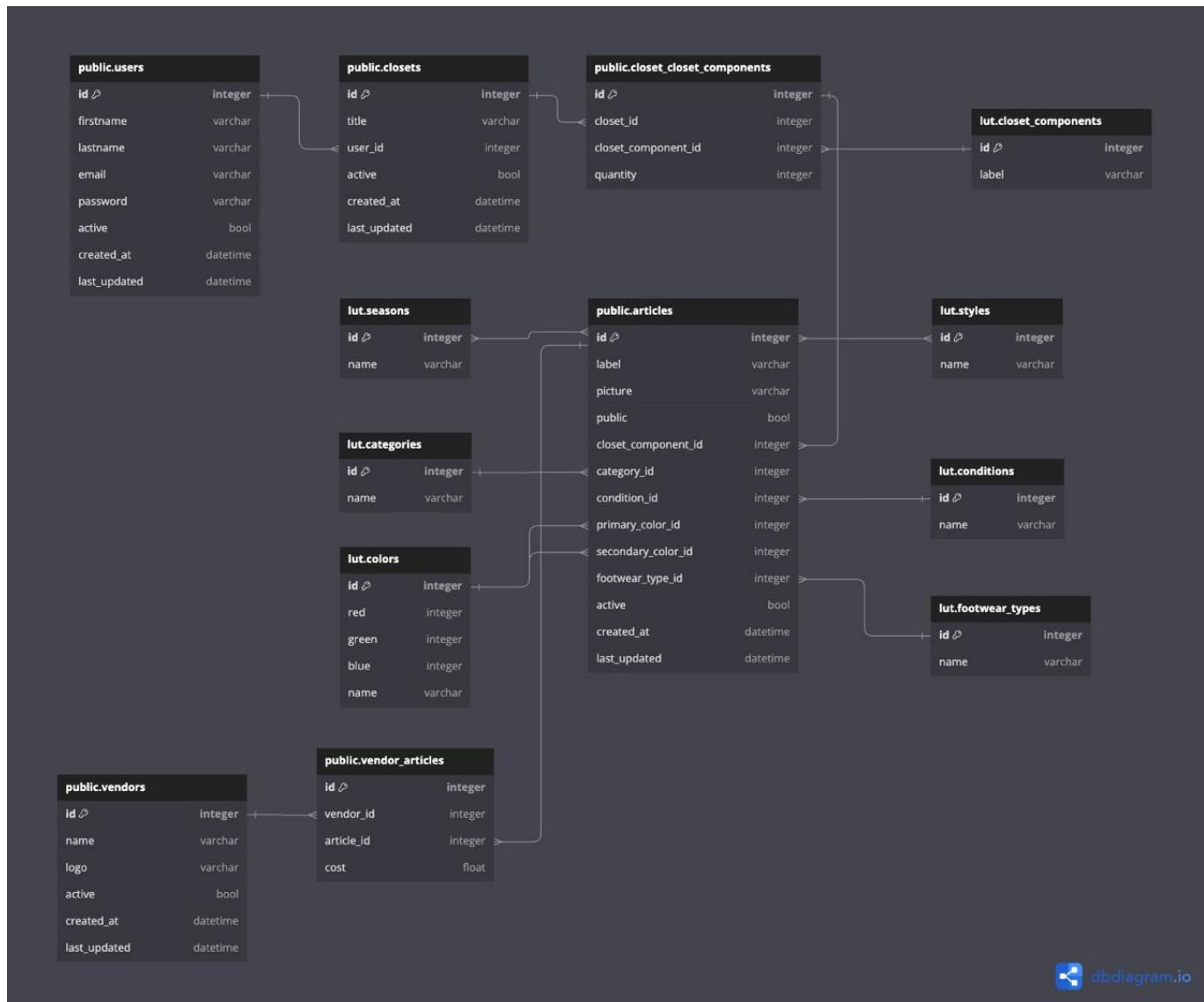
lut.conditions

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator stanja (primarni ključ)
name	varchar	Naziv stanja

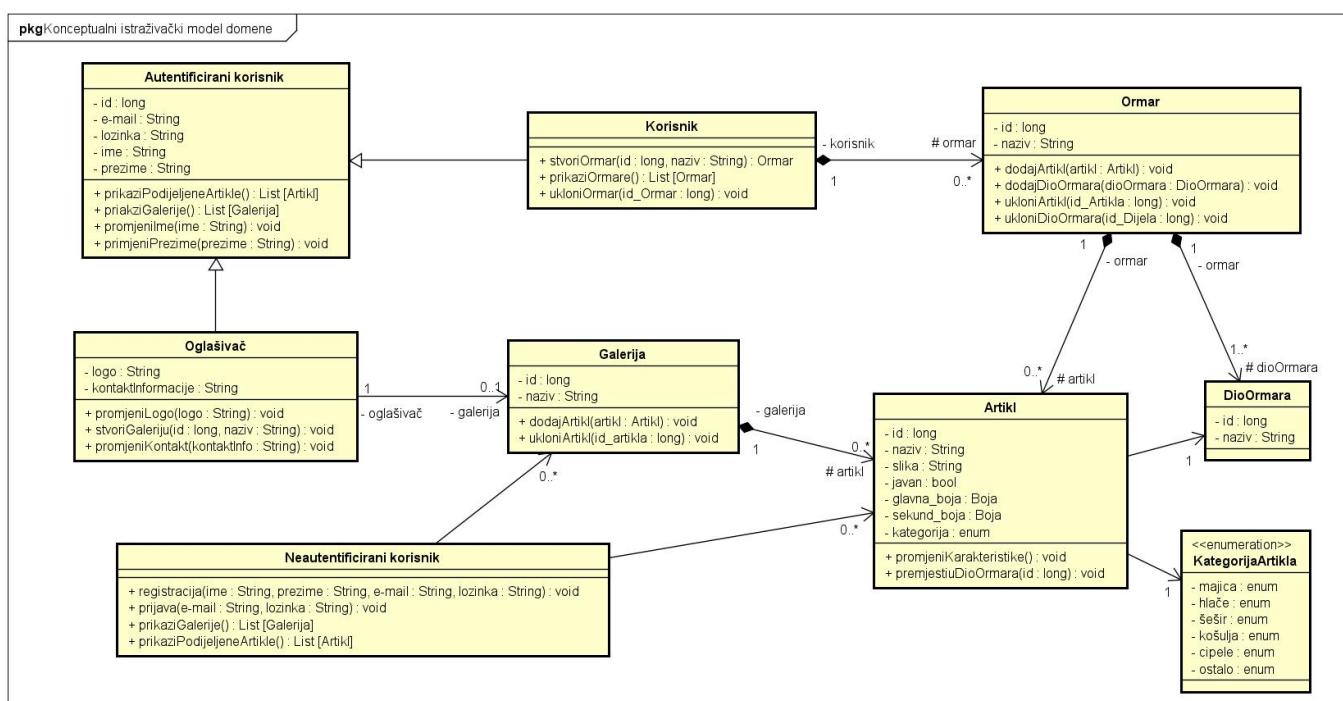
lut.colors

Atribut	Tip podatka	Opis varijable
id	integer	Jedinstveni identifikator boje (primarni ključ)
red	integer	Vrijednost crvene komponente (0, 255)
green	integer	Vrijednost zelene komponente (0, 255)
blue	integer	Vrijednost plave komponente (0, 255)
name	varchar	Naziv boje

Dijagram baze podataka



Dijagram razreda



Konceptualni istraživački model domene (najvažnije funkcionalnosti)

Neautentificirani korisnik može se registrirati ili prijaviti (ako nije registriran), može vidjeti articlje koji su podijeljeni (imaju atribut javan = true) te galerije (odnosno articlje unutar njih).

Autentificirani korisnik specijalizira se u običnog registriranog korisnika te oglavlivača. Običan registrirani korisnik ima nula ili vise vlastitih ormara (koji su vezani za njega), može dodavati nove ormare ili ih brisati te može vidjeti sve svoje ormare.

Svaki omar ima više articlja (koji su vezani za njega). Registrirani korisnik može preko odabranog ormara u njega dodavati ili iz njega uklanjati articlje ili pristupiti jedinstvenom articlu i njemu promjeniti karakteristike. Oma se također sastoji od više dijelova ormara (koji su za njega vezani) te vlasnik ormara može uklanjati ili dodavati dijelove u odabrani omar. Svaki je articlu smješten u određen dio ormara te se može premjestiti u drugi dio ormara.

Oglavlivač sadrži logo i kontakt informacije (koje može mijenjati) te može definirati svoju galeriju te u nju dodavati articlje (koji su vezani za galeriju). Oglavlivač može u galeriju dodavati ili iz nje uklanjati articlje

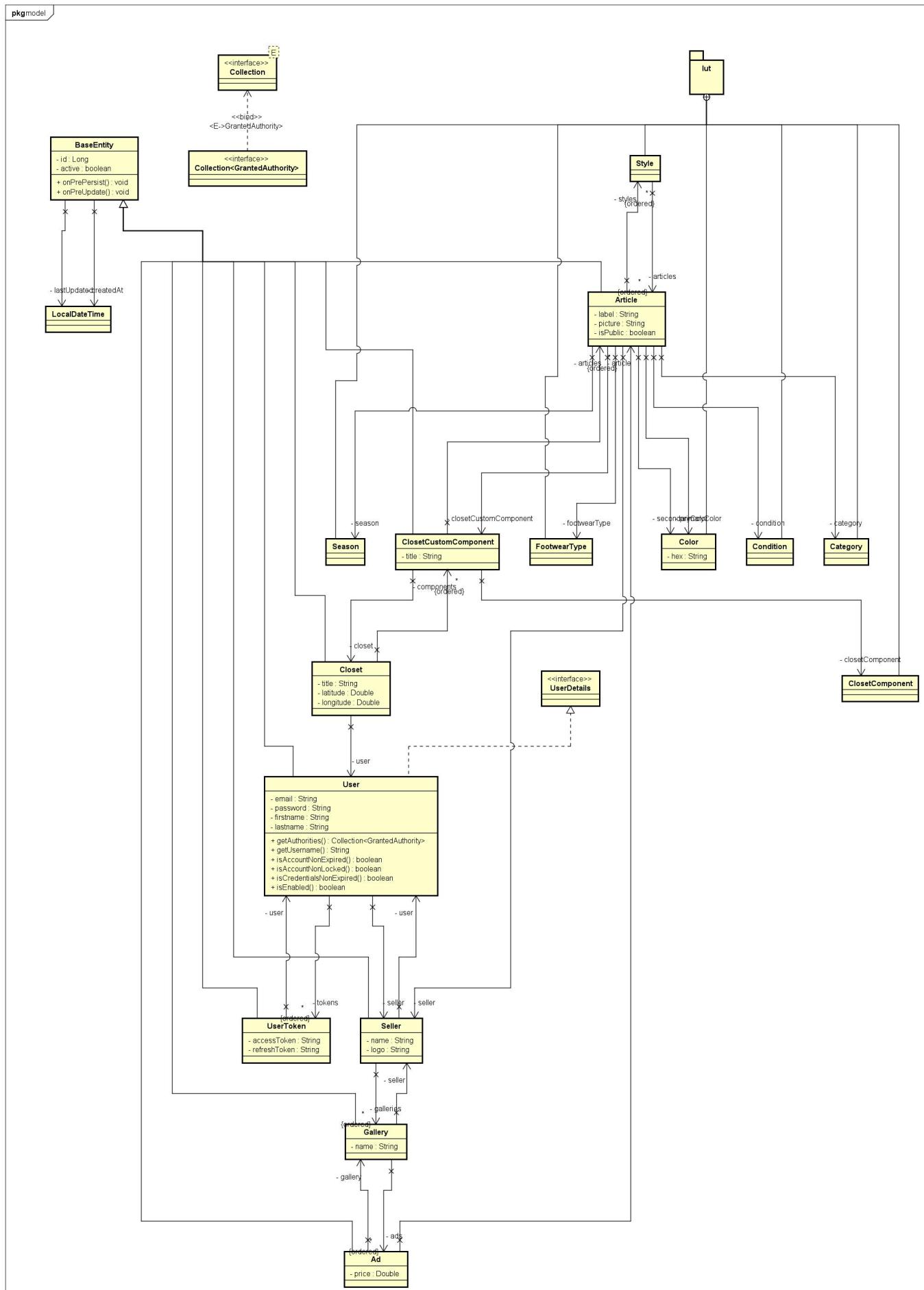
Automatski generirani dijagrami razreda

Automatski generirani dijagrami razreda nalaze se ovdje

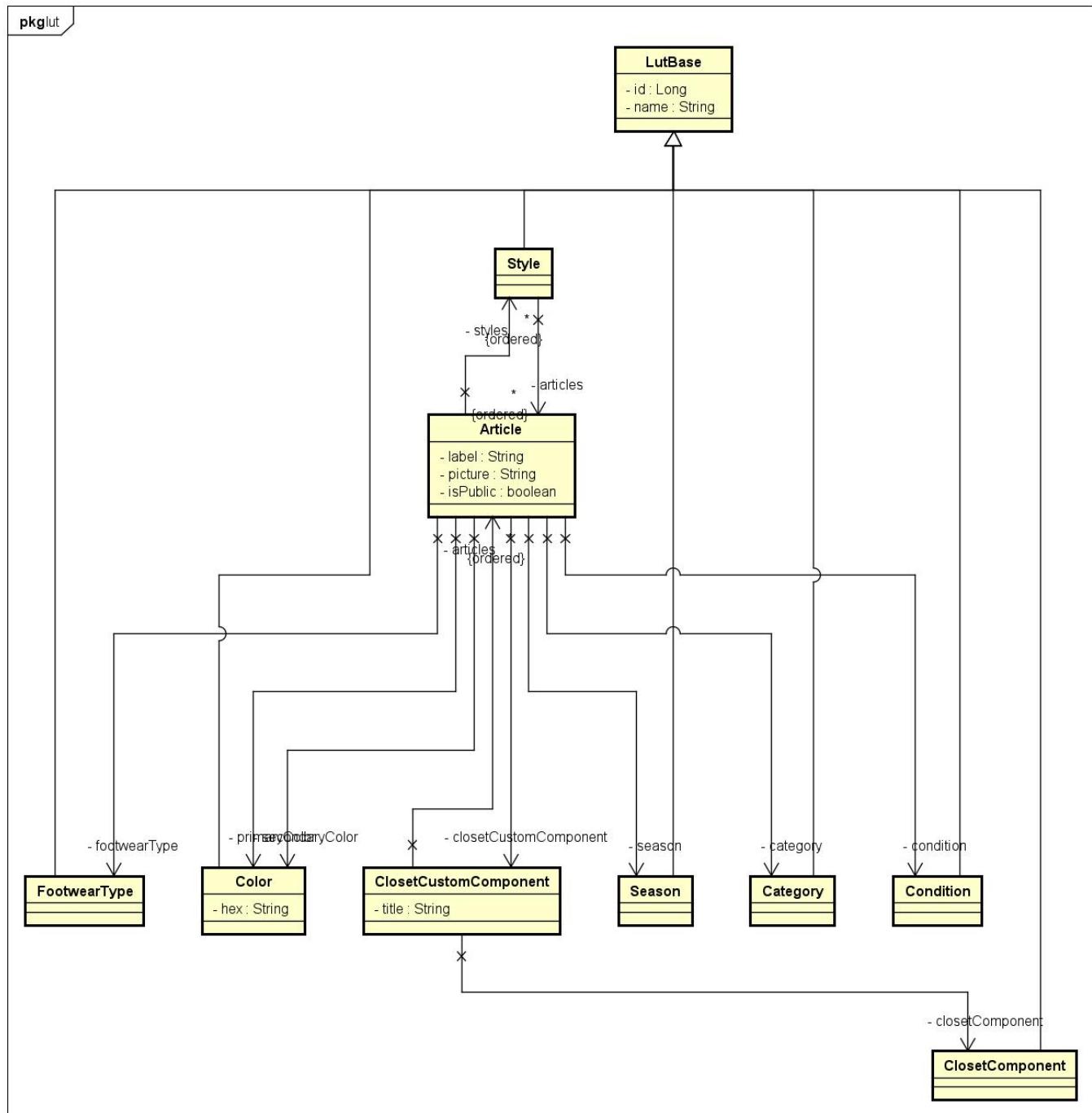
(<https://github.com/davidvodopija/portane/tree/devdoc/Dokumentacija/dijagrami%20razreda/generated>).

Neki od njih su vrlo veliki i relativno nepregledni pa nisu prikazani u wiki. Oni koji su generirani relativno pregledno i oni koji su ručno namješteni prikazani su ispod.

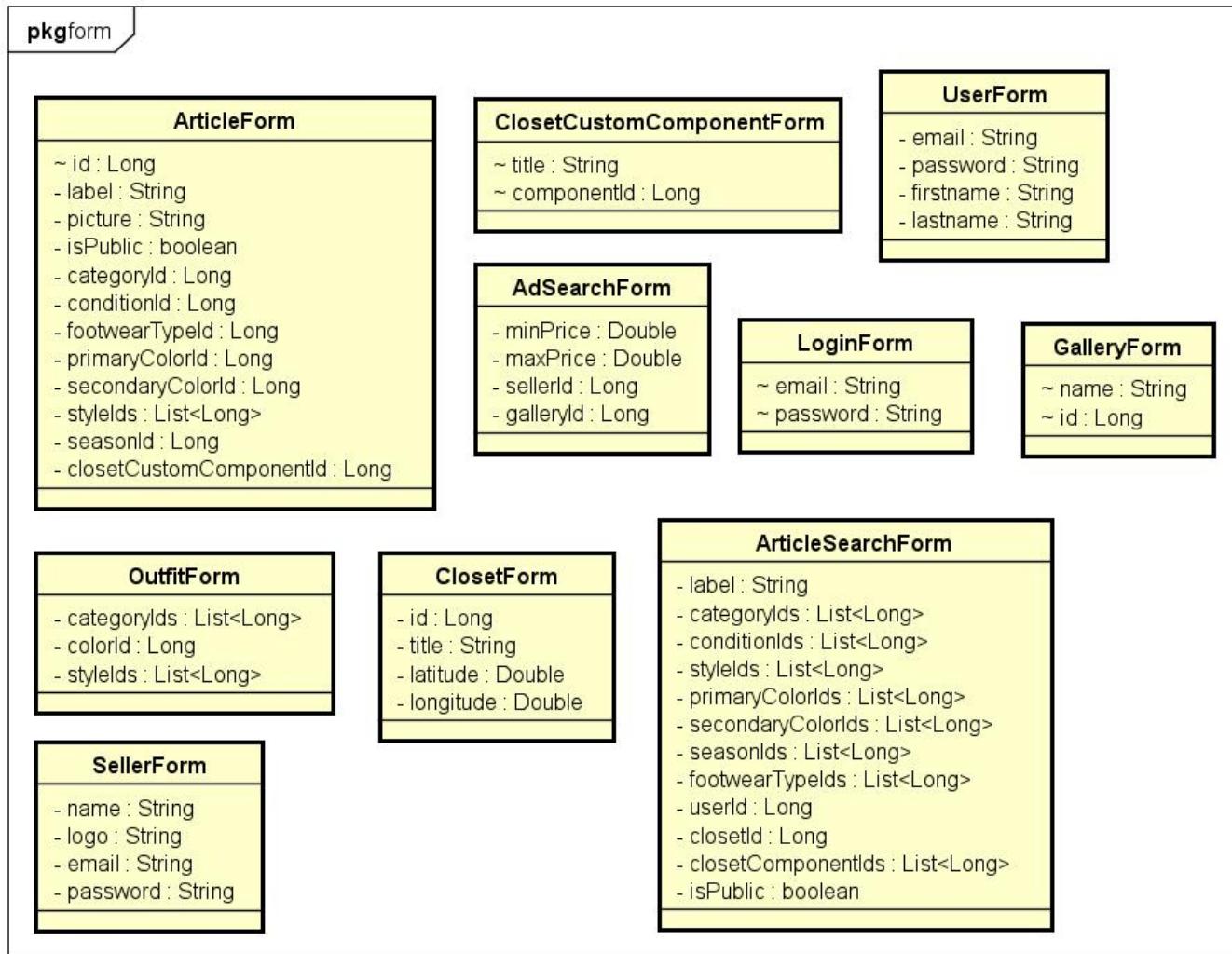
Dijagram razreda komponente model.



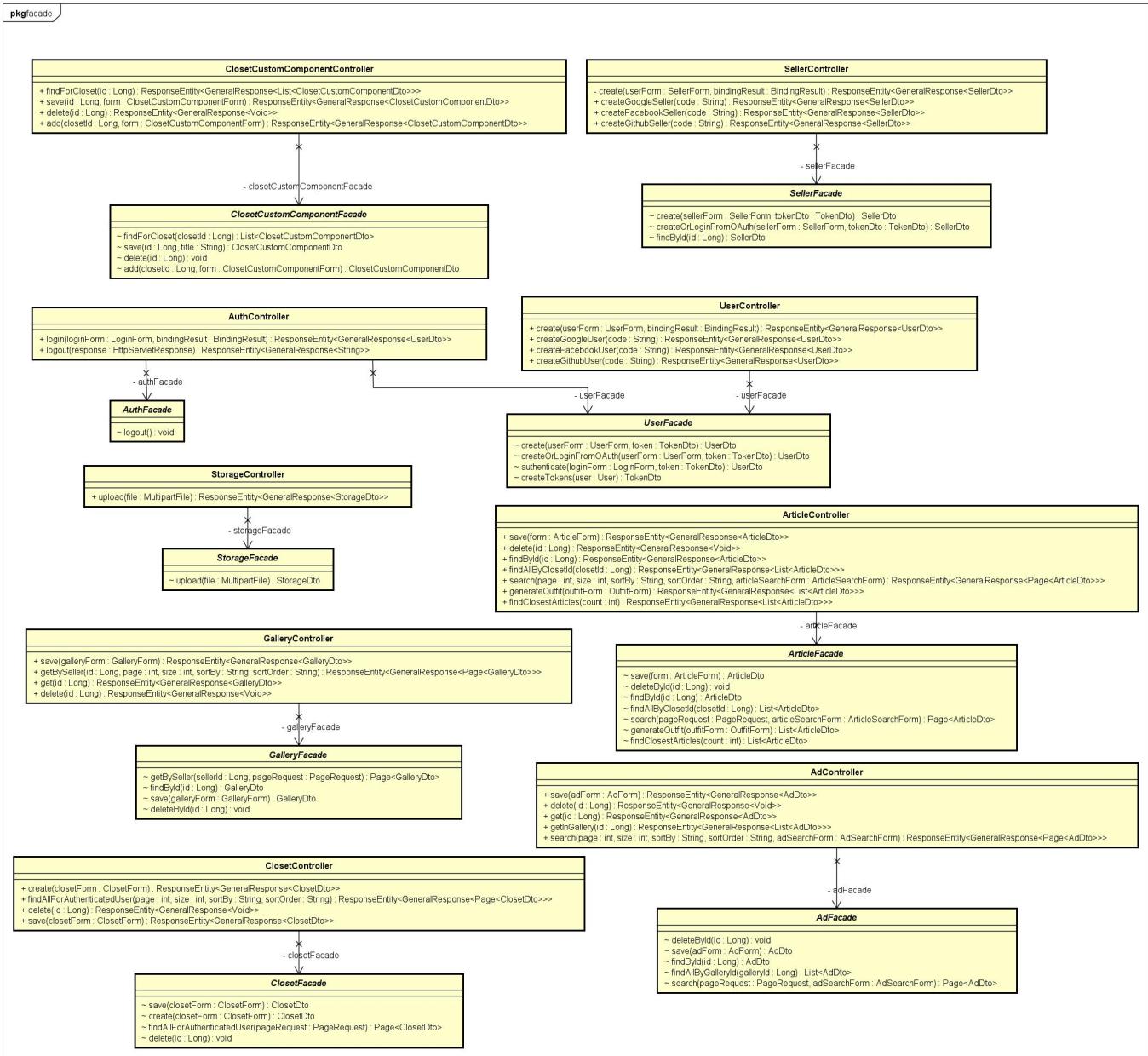
Dijagram razreda komponente lut.



Dijagram razreda komponente form.



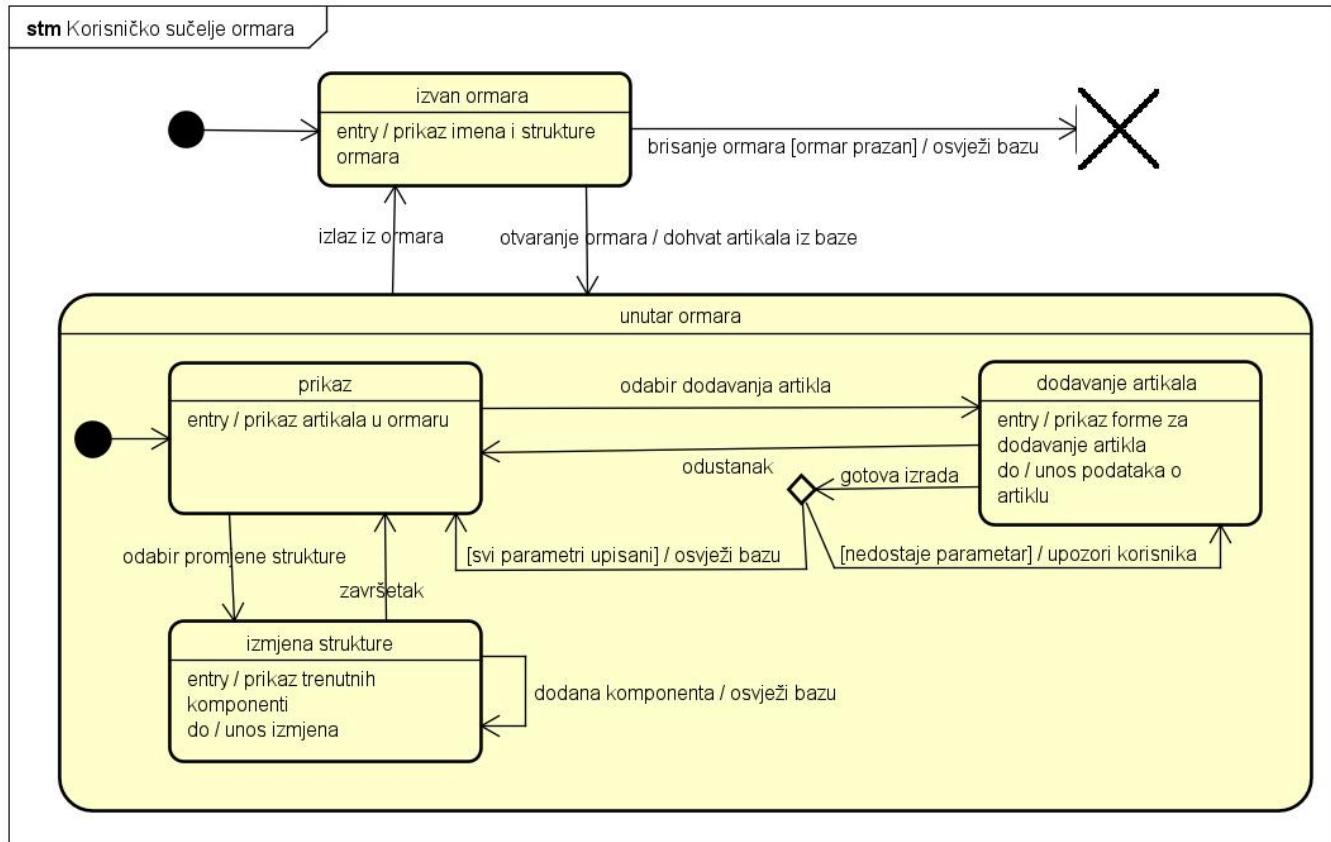
Dijagram razreda komponente facade.



Dinamičko ponašanje aplikacije

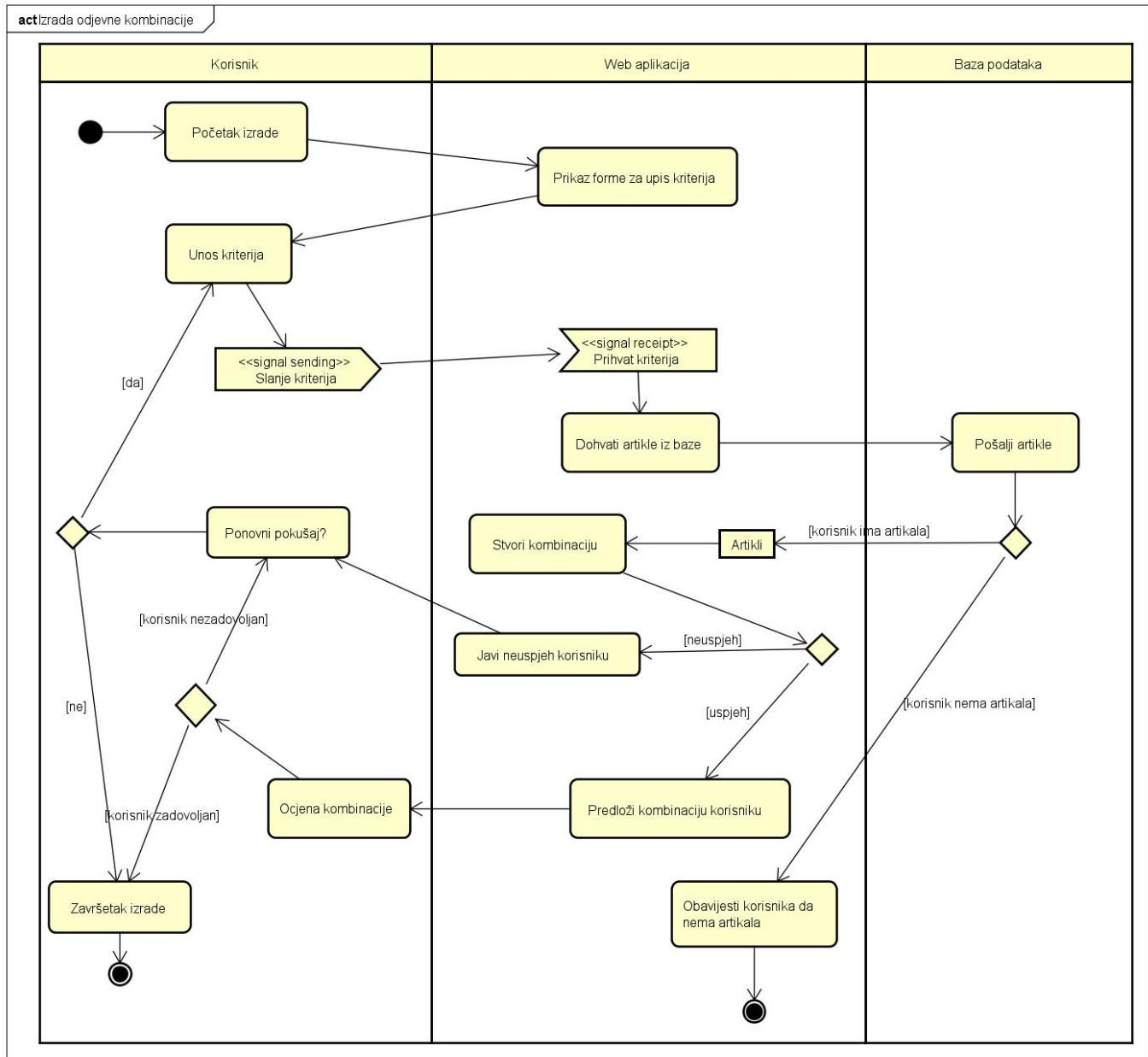
UML dijagrami stanja

Donji dijagram pokazuje stanja u kojima se može nalaziti korisničko sučelje ormara. Nakon kreacije ormara, u kojoj se definiralo njegovo ime i komponente, korisniku se prikazuje kartica sa tim podacima. Korisnik sada može obrisati ormara ili ga otvoriti. Nakon otvaranja korisniku se prikazuju artikli koji se nalaze u ormaru (ako ih ima) te opcije dodavanja novih artikala ili izmjena strukture ormara. U stanju izmjene strukture korisnik može dodati ili brisati komponente ormara, a nakon svake izmjene ostaje u istom stanju. U stanju dodavanja artikla korisnik ispunjava formu o podacima novog artikla (slika, naziv, boja...). Ako je forma dobro ispunjena, podaci o novom artiklu šalju se u bazu i prelazi se u stanje prikaza. Naravno, iz stanja dodavanja artikla može se izaći i jednostavnim odustajanjem od izrade.



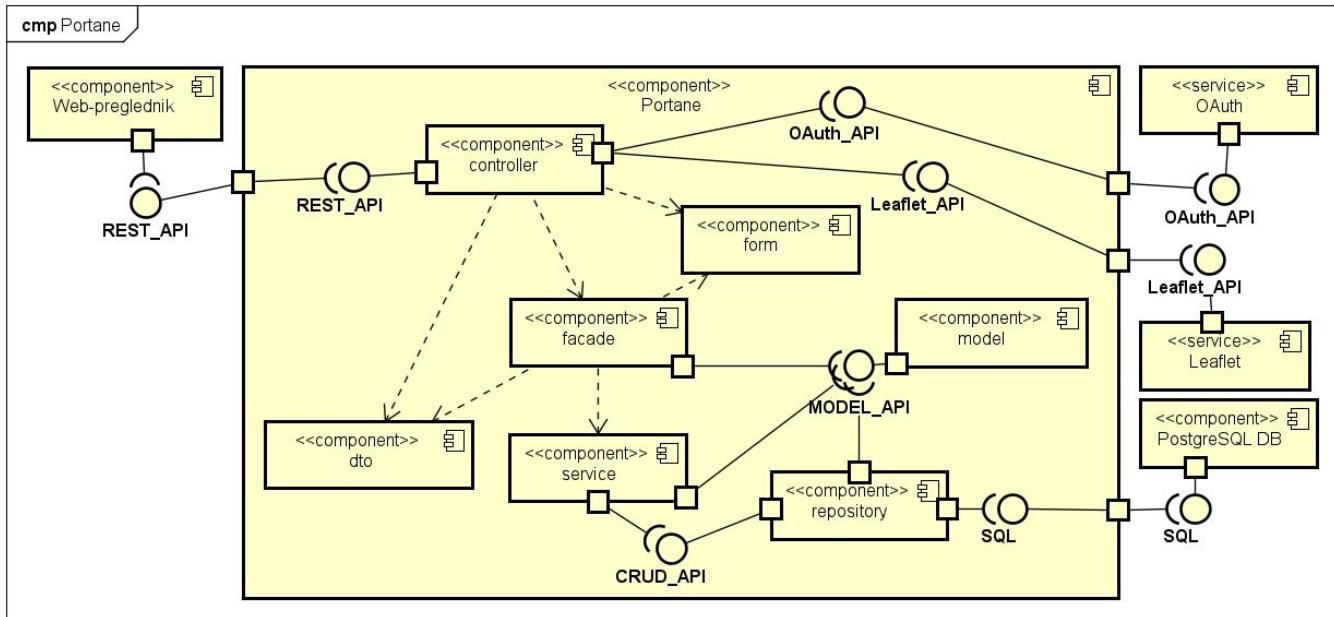
UML dijagrami aktivnosti

Ovaj dijagram aktivnosti prikazuje tijek izrade odjevne kombinacije. Na početku korisnik odabire opciju izrade odjevne kombinacije nakon čega mu se otvara forma za upis kriterija. Korisnik upisuje kriterije koje kombinacija mora zadovoljavati (npr. boja, stil, vremenska prognoza...) koji se šalju web aplikaciji. Zatim aplikacija dohvaća korisnikove artikle iz baze podataka i na temelju njih stvara kombinaciju. Ako korisnik nema artikala, aplikacija korisniku dojavljuje grešku i završava proces izrade. Ako aplikacija nije mogla ispuniti korisničke zahtjeve (npr. korisnik ne posjeduje artikle koji zadovoljavaju kriterije) onda se korisniku dojavljuje neuspjeh i nudi mu se opcija ponovnog unosa kriterija. Ako je aplikacija uspješno stvorila kombinaciju, ona se prikazuje korisniku koji odlučuje je li zadovoljavajuća. Ako je korisnik zadovoljan aktivnost se završava, a ako nije onda mu se omogućuje ponovni unos kriterija.



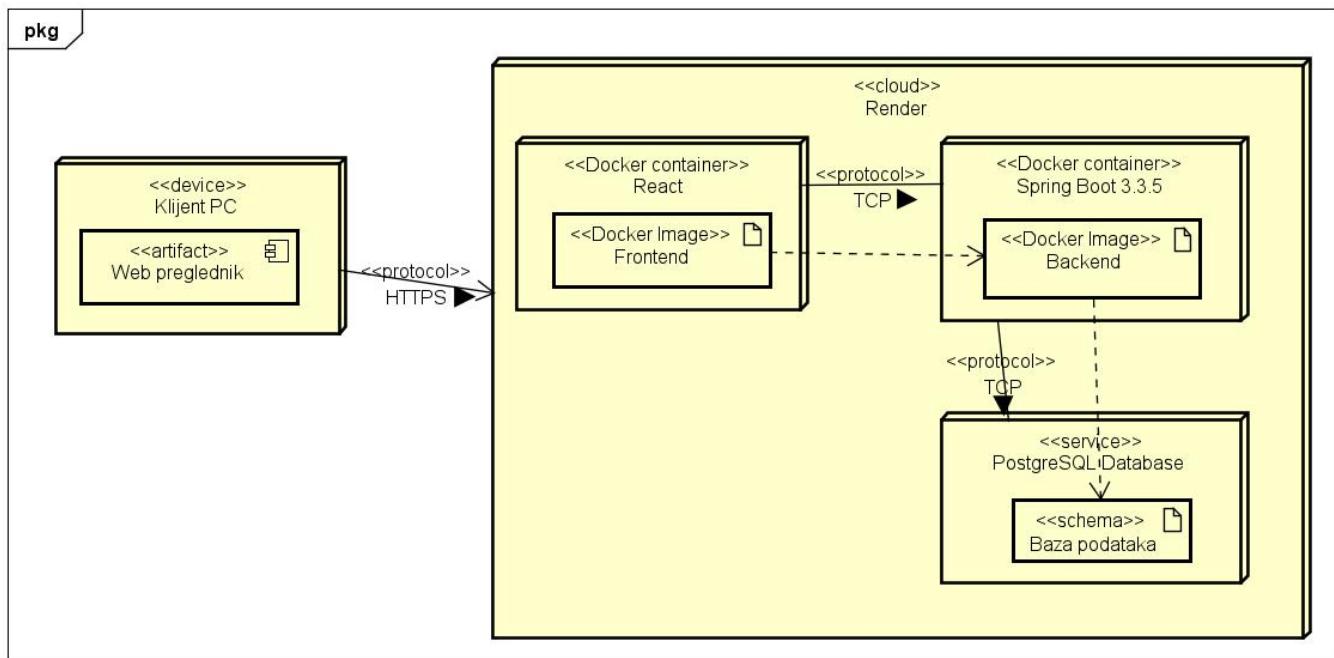
Dijagram komponenata

Dijagram komponenti prikazuje komponente web-aplikacije, sučelja i veze između komponenti. Glavne komponente su *kontroler* (eng. *controller*) koji implementira **REST API** te poziva fasadni sloj koji obrađuje podatke. U tome pomaže komponenta *form* koja omogućuje lako baratanje podatcima koje korisnici šalju na *backend*. *Fasada* predstavlja jedinstvenu ulaznu točku za obradu podataka na poslužitelju, a u tome joj pomaže komponenta *servis* (eng. *service*) koja obavlja poslovnu logiku aplikacije. *Servis* koristi **CRUD** (*Create, Read, Update, Delete*) operacije koje nudi komponenta *repozitorij* (eng. *repository*) koji pristupa **PostgreSQL** bazi podataka. Komponenta *model* predstavlja podatke koje aplikacija koristi i manipulira. Ovi se podaci mapiraju na entitete baze podataka. *Fasada* je također odgovorna za formiranje **DTO-a** (eng. *Data Transfer Object*) koji su prikladni za prikaz na *frontendu*. U implementaciji postoji još neke komponente koje ovdje nisu prikazane jer nisu ključne za rad aplikacije. To su *error* i *exception* komponente koje barataju iznimkama i pogreškama koje mogu nastati tijekom rada aplikacije. Aplikacija također koristi *Google Oauth* servis pa mora imati vezu sa istim. Pri izradi ormara prikazuje se interaktivna karta koja korisnicima služi za postavljanje lokacije. Pri tome se koristi *Leaflet* servis.

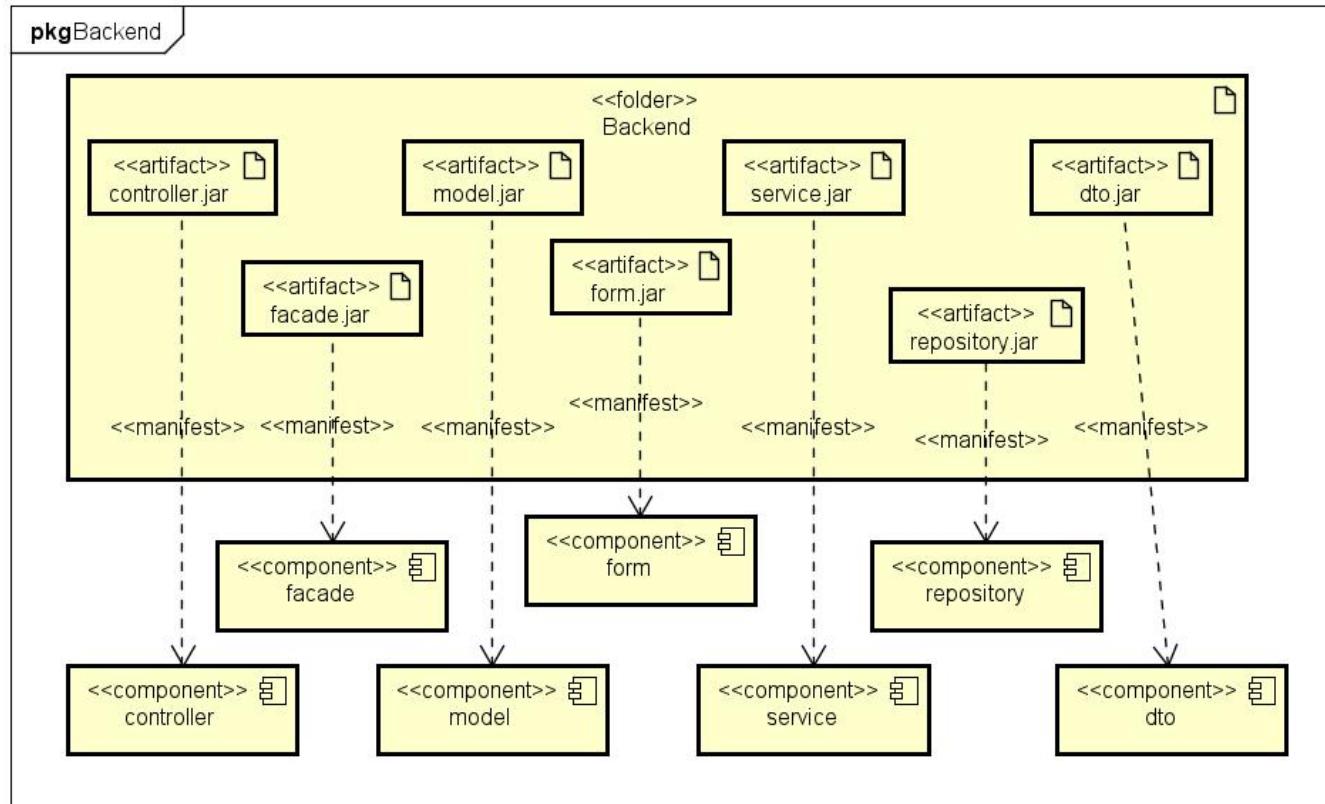


Dijagram razmještaja

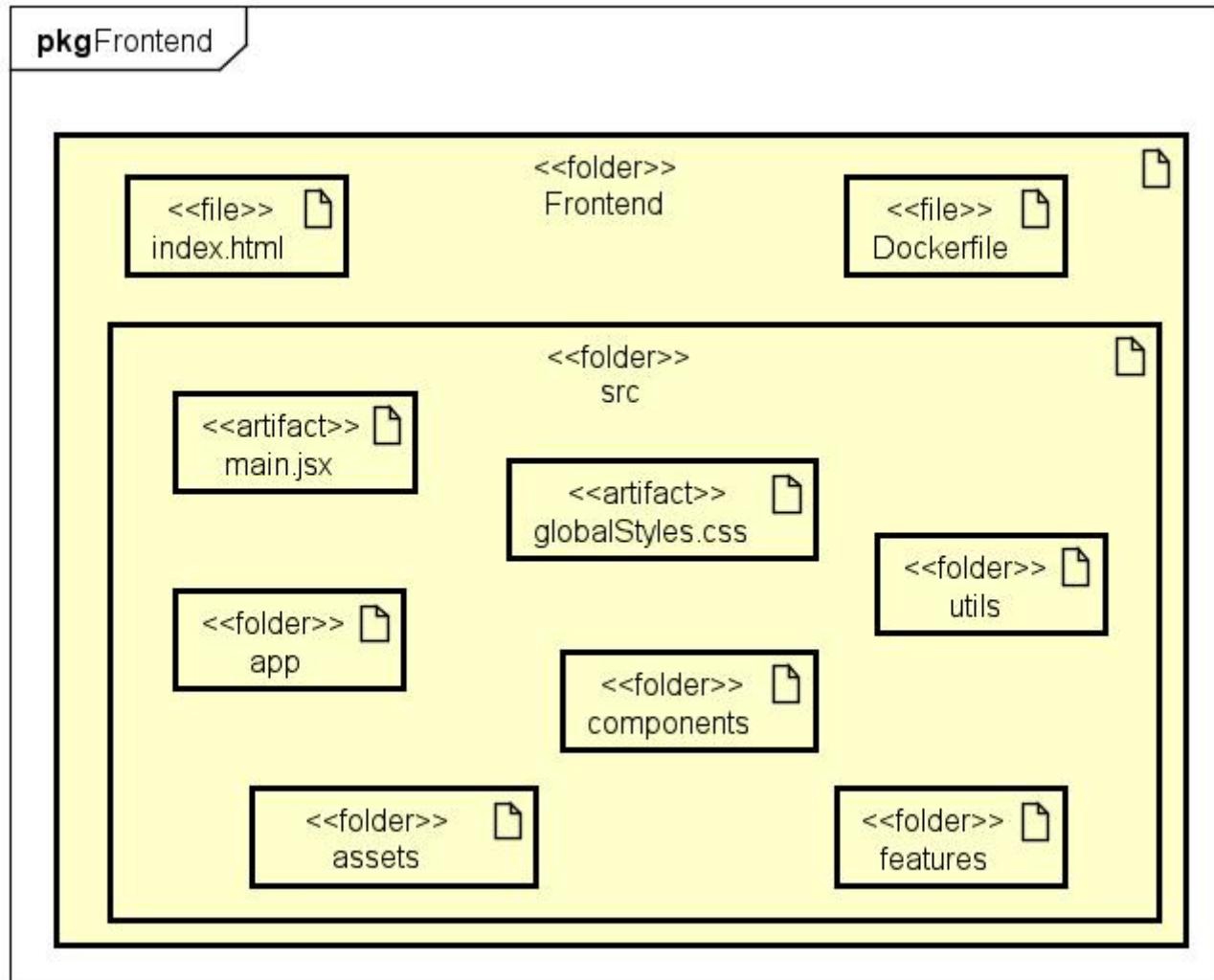
Dijagram razmještaja prikazuje razmještaj programskih artefakata na virtualnom okruženju. Na klijentskom PC-u pokrenut je *web preglednik* koji komunicira sa poslužiteljem koji je *deployan* na *Renderu*. Specifično, odvojeno su *deployani frontend* i *backend* u vlastitim *Docker Container-ima*. Na *Renderu* je smještena i *PostgreSQL* baza podataka koju koristi aplikacija.



Dijagram prikazuje detalje *backenda*. Cjelokupni kod smješten je u mapu *backend* u kojoj se nalaze artefakti koji manifestiraju komponente prikazane u dijagramu komponenti. Napomena: na ovome dijagramu nisu prikazane veze između komponenti, ali te se veze mogu vidjeti na gornje priloženom dijagramu komponenti.

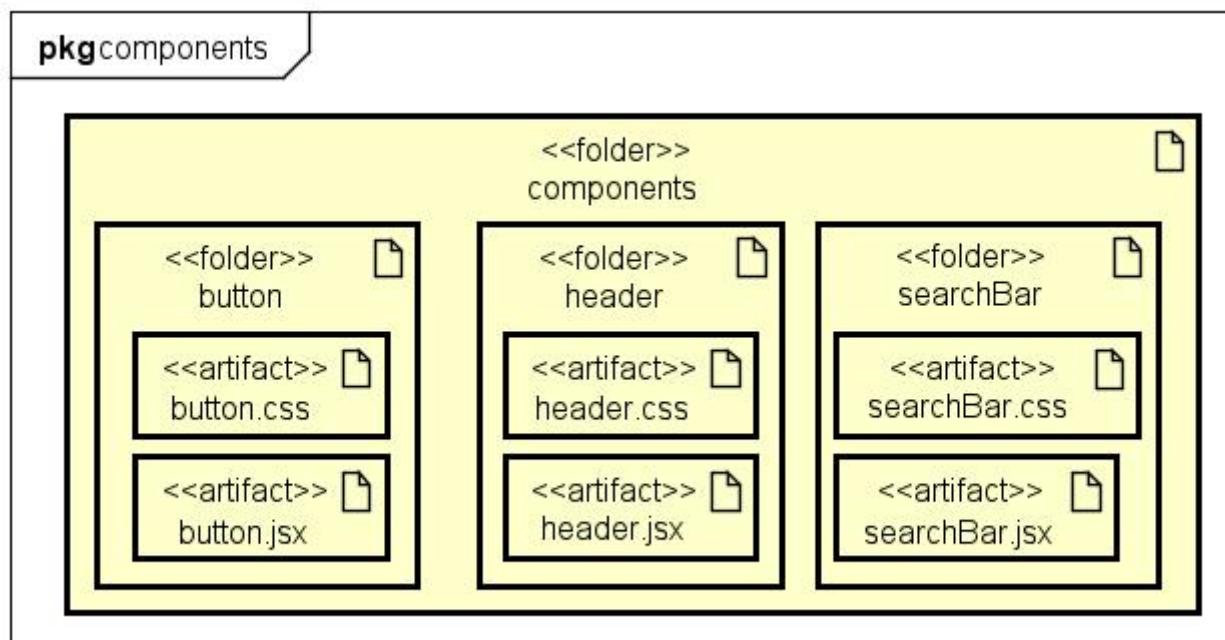


Na ovome je dijagramu prikazan razmještaj *frontenda*. U mapi *frontend* nalazi se početna *indeks.html* datoteka te *Dockerfile*, ali i još neke konfiguracijske datoteke koje su izostavljene zbog preglednosti dijagrama. U istoj mapi nalazi se podmapa *src* u kojoj je smješten kod za *frontend*. Kod je razmješten u zasebne mape, zavisno o tome koja mu je primjena. Tako se u mapi *components* nalaze često korištene komponente *web-aplikacije* kao što su *header*, *footer*, *button*, *searchBar*... U mapi *features* nalazi se kod za implementaciju funkcionalnosti aplikacije na *frontendu* kao što je kreiranje ormara, prikaz ormara, autentifikacija, traženje artikala... U mapi *assets* nalaze se slike i ikone koje se prikazuju na *web-stranici*.

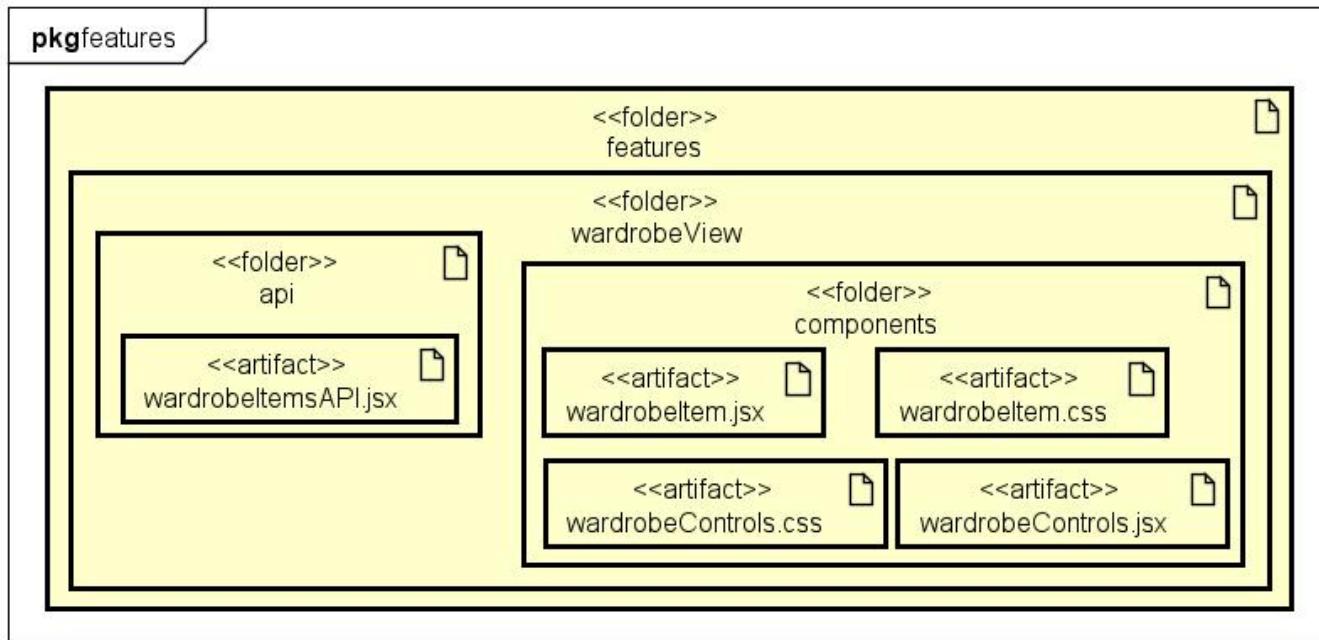


Na iduća dva dijagrama prikazana je detaljnija struktura nekih od navedenih mapa. To nisu sve mape, niti je u svakom dijagramu prikazan cijeli sadržaj mape. Razlog tomu je osiguravanje preglednosti dijagrama.

Dijagram prikazuje razmještaj unutar mape `components`. U toj se mapi nalazi druge mape, a svaka od njih sadrži jednu `.css` i jednu `.jsx` datoteku koje su ključne za implementaciju te komponente u aplikaciji.



Dijagram prikazuje razmještaj unutar mape *features*. Prikazan je samo jedna unutarnja mapa (*wardrobeView*), no ostale podmape prate istu strukturu. Zbog toga nisu prikazane na dijagramu, ali ih korisnik može sam vidjeti ukoliko se odluči direktno proučavati kod aplikacije. Svaka se podmapa sastoji od mape *api* u kojoj je implementiran poziv prema poslužitelju i mape *components* koja sadrži *.jsx* i *.css* datoteke ključne za prikaz i implementaciju funkcionalnosti.



Ispitivanje komponenti

Test 1: `testFindById_withExistingId`

Pronalaženje galerije prema ID-u kada galerija postoji u bazi podataka.

Ispitni slučaj:

- **Ulazni podaci:**
 - ID galerije: **1L**
- **Očekivani rezultati:**
 - Metoda `findById` vraća galeriju.
 - Galerija ima ID **1L** i naziv "Sample Gallery".
 - Metoda `galleryRepository.findById` pozvana je točno jednom.
- **Dobiveni rezultati:**
 - Test prolazi.

Postupak provođenja ispitivanja:

1. Priprema:

- Stvara se mock objekt za `galleryRepository` koji vraća optionalni objekt galerije s ID-jem **1L**.

2. Izvršavanje:

- Poziva se metoda `findById` s ID-jem `1L`.

3. Verifikacija:

- Provjerava se da su ID i naziv galerije ispravni.
 - Verificira se poziv metode `findById` točan broj puta.
-

Test 2: `testFindById_withNonExistingId`

Pronalaženje galerije prema ID-u kada galerija ne postoji u bazi podataka.

Ispitni slučaj:

- **Ulazni podaci:**

- ID galerije: `1L`

- **Očekivani rezultati:**

- Metoda `findById` baca iznimku s porukom: "Gallery with id 1 not found."
- Metoda `galleryRepository.findById` pozvana je točno jednom.

- **Dobiveni rezultati:**

- Test prolazi.

Postupak provođenja ispitivanja:

1. Priprema:

- Stvara se mock objekt za `galleryRepository` koji vraća prazan `Optional`.

2. Izvršavanje:

- Poziva se metoda `findById` s ID-jem `1L`.

3. Verifikacija:

- Provjerava se da metoda baca očekivanu iznimku.
 - Verificira se poziv metode `findById` točan broj puta.
-

Test 3: `testfindAllBySellerId_withExistingSellerId`

Pronalaženje svih galerija povezanih s određenim prodavačem kada galerije postoje.

Ispitni slučaj:

- **Ulazni podaci:**

- ID prodavača: `10L`
- Stranica: `0`
- Broj elemenata po stranici: `5`

- **Očekivani rezultati:**

- Metoda `findAllBySellerId` vraća stranicu s dvije galerije.

- Galerije imaju nazine "Gallery 1" i "Gallery 2".
- Metoda `galleryRepository.findAllBySellerId` pozvana je točno jednom.

- **Dobiveni rezultati:**

- Test prolazi.

Postupak provođenja ispitivanja:

1. **Priprema:**

- Stvara se mock objekt za `galleryRepository` koji vraća stranicu s dvije galerije.

2. **Izvršavanje:**

- Poziva se metoda `findAllBySellerId` s ID-jem `10L` i postavkama paginacije.

3. **Verifikacija:**

- Provjerava se da stranica sadrži točan broj galerija i njihove nazine.
 - Verificira se poziv metode `findAllBySellerId` točan broj puta.

Test 4: `testfindAllBySellerId_withNoGalleries`

Pronalaženje svih galerija povezanih s određenim prodavačem kada galerije ne postoje.

Ispitni slučaj:

- **Ulazni podaci:**

- ID prodavača: `10L`
 - Stranica: `0`
 - Broj elemenata po stranici: `5`

- **Očekivani rezultati:**

- Metoda `findAllBySellerId` vraća praznu stranicu.
 - Metoda `galleryRepository.findAllBySellerId` pozvana je točno jednom.

- **Dobiveni rezultati:**

- Test prolazi.

Postupak provođenja ispitivanja:

1. **Priprema:**

- Stvara se mock objekt za `galleryRepository` koji vraća praznu stranicu.

2. **Izvršavanje:**

- Poziva se metoda `findAllBySellerId` s ID-jem `10L` i postavkama paginacije.

3. **Verifikacija:**

- Provjerava se da je stranica prazna.
 - Verificira se poziv metode `findAllBySellerId` točan broj puta.

Test 5: `testCreateUserSuccessfully`

Dodavanje novog korisnika kada korisnik s istom email adresom ne postoji u sustavu.

Ispitni slučaj:

- **Ulazni podaci:**

- `UserForm` objekt s podacima:
 - Email: `test@example.com`
 - Lozinka: `aBcD$1234`
 - Ime: `John`
 - Prezime: `Doe`

- **Očekivani rezultati:**

- Metoda `create` uspješno dodaje korisnika.
- Vraćeni objekt `UserDto` ima isti email kao `UserForm`.
- Generirani token nije null.
- Metode u `userService`, `userFormMapper`, i `jwtService` se pozivaju očekivani broj puta.

- **Dobiveni rezultati:**

- Test prolazi.

Postupak provođenja ispitivanja:

1. **Priprema:**

- Stvara se mock objekt za `userService` koji vraća prazan `Optional` prilikom poziva `getUserByEmail`.
- Mapper `userFormMapper` mapira `UserForm` u `User`.
- `userService.save` vraća spremljenog korisnika.
- `jwtService.generateTokens` vraća testni token.

2. **Izvršavanje:**

- Poziva se metoda `create` s pripremljenim `UserForm` i praznim `TokenDto`.

3. **Verifikacija:**

- Provjerava se da su vraćeni podaci očekivani (email i token).
- Provjerava se da su mock metode pozvane točan broj puta.

Test 6: `testCreateUserAlreadyExists`

Sprječavanje dodavanja korisnika ako korisnik s istom email adresom već postoji u sustavu.

Ispitni slučaj:

- **Ulazni podaci:**

- `UserForm` objekt s podacima:
 - Email: `existing@example.com`

- **Očekivani rezultati:**

- Metoda `create` baca iznimku s porukom "User already exists."
- Metoda `userService.getUserByEmail` pozvana je točno jednom.
- Mapperi `userFormMapper` i `userUserDtoMapper` nisu pozvani.

- **Dobiveni rezultati:**

- Test prolazi.

Postupak provođenja ispitivanja:

1. **Priprema:**

- Stvara se mock objekt za `userService` koji vraća `Optional` s postojećim korisnikom prilikom poziva `getUserByEmail`.

2. **Izvršavanje:**

- Poziva se metoda `create` s pripremljenim `UserForm` i praznim `TokenDto`.

3. **Verifikacija:**

- Provjerava se da metoda baca očekivanu iznimku s odgovarajućom porukom.
 - Provjerava se da mock metode koje nisu potrebne nisu pozvane.
-

Selenium testovi

Test: Registracija

Ulazi

- **Ime:** `Marko`
- **Prezime:** `Zovko`
- **E-mail:** `mz@gmail.com`
- **Lozinka:** `Mcz123`

Koraci ispitivanja

1. Otvoriti aplikaciju na osnovnoj URL adresi (/).
2. Postaviti veličinu prozora preglednika na `1280x680` piksela.
3. Kliknuti na poveznicu "**Registrijaj se**".
4. Kliknuti na polje za unos imena (ID: `name`).
5. Unijeti ime "**Marko**" u polje za unos.
6. Kliknuti na polje za unos prezimena (ID: `lastname`).
7. Unijeti prezime "**Zovko**" u polje za unos.
8. Kliknuti na polje za unos e-mail adrese (ID: `email`).
9. Unijeti e-mail adresu "**mz@gmail.com**" u polje za unos.
10. Kliknuti na polje za unos lozinke (ID: `password`).
11. Unijeti lozinku "**Mcz123**" u polje za unos.
12. Kliknuti na gumb za potvrdu registracije (CSS: `.btn-small`).

Očekivani rezultat

- Sustav uspješno prihvata unesene podatke i registrira korisnika.

- Korisnik se preusmjerava na početnu stranicu ili dobiva poruku o uspješnoj registraciji.

Dobiveni rezultat

The image contains two screenshots of a web browser. The top screenshot shows a registration form titled 'REGISTRIRAJ SE' with fields for Name, Surname, Email (mz@gmail.com), and Password (*****). The bottom screenshot shows a user profile page for 'Marko Zovko' with a placeholder profile picture, email (mz@gmail.com), and statistics: 0 wardrobes and 0 items. It includes buttons for 'DODAJ NOVI ORMAR' and 'IZGRADI ODJEVNU KOMBINACIJU'.

Test: Pogrešan login

Ulazi

- E-mail:** mz1@gmail.com (nepostojeća e-mail adresa u sustavu)
- Lozinka:** Mcz123

Koraci ispitivanja

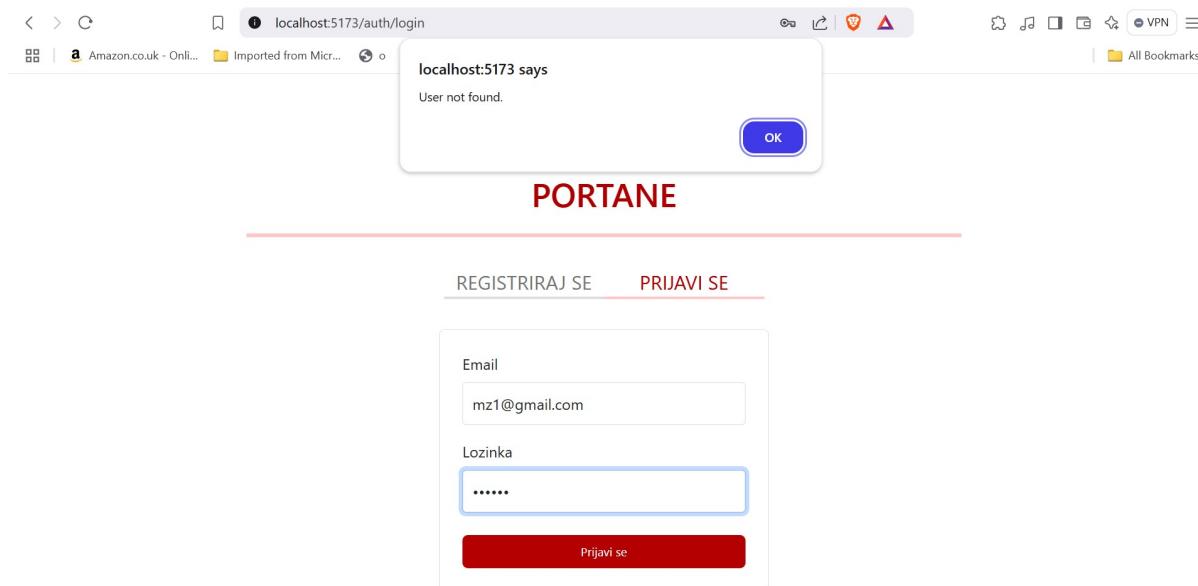
- Otvoriti aplikaciju na osnovnoj URL adresi (/).
- Postaviti veličinu prozora preglednika na 1280x680 piksela.
- Kliknuti na poveznicu "Prijavi se".
- Kliknuti na polje za unos e-mail adrese (ID: email).
- Unijeti e-mail adresu "mz1@gmail.com" u polje za unos.
- Kliknuti na polje za unos lozinke (ID: password).

7. Unijeti lozinku "**Mcz123**" u polje za unos.
8. Kliknuti na gumb za prijavu (CSS selektor: `.btn-login`).
9. Provjeriti prikaz poruke o pogrešci.

Očekivani rezultat

- Sustav prepoznaje da korisnik s unesenom e-mail adresom "**mz1@gmail.com**" i lozinkom "**Mcz123**" ne postoji.
- Na ekranu se prikazuje poruka "**User not found**".
- Korisnik ostaje na stranici za prijavu.

Dobiveni rezultat



Test: Pogrešan e-mail (bez @)

Ulazi

- **E-mail:** **Marko_Zovko** (neispravan e-mail bez @)
- **Lozinka:** **Mcz123**

Koraci ispitivanja

1. Otvoriti aplikaciju na osnovnoj URL adresi (/).
2. Postaviti veličinu prozora preglednika na **1280x680** piksela.
3. Kliknuti na poveznicu "**Registiraj se**".
4. Kliknuti na polje za unos e-mail adrese (ID: `email`).
5. Unijeti e-mail adresu "**Marko_Zovko**" u polje za unos.
6. Kliknuti na polje za unos lozinke (ID: `password`).
7. Unijeti lozinku "**Mcz123**" u polje za unos.
8. Kliknuti na gumb za potvrdu registracije (CSS selektor: `.btn-small`).
9. Provjeriti prikaz poruke o pogrešci.

Očekivani rezultat

- Sustav prepoznaće da uneseni e-mail "**Marko_Zovko**" nije ispravan jer nedostaje @.
- Na ekranu se prikazuje poruka "**Please include an @ in the email address**".
- Korisnik ostaje na stranici za registraciju.

Dobiveni rezultat

The screenshot shows a registration form on a website. The URL in the address bar is `localhost:5173/auth/register`. The form has fields for Name (Ime) and Surname (Prezime), both filled with "Marko". The Email field contains "Marko_Zovko". A tooltip message appears below the Email field: "Please include an '@' in the email address. 'Marko_Zovko' is missing an '@.'". There is also a password field with masked input and a checkbox for becoming a seller. A red button at the bottom right says "Registruj se".

Test: Nepostojeći URL

Ulazi

- **URL:** /**nepostojeći** (nepostojeći URL)

Koraci ispitivanja

1. Otvoriti aplikaciju na osnovnoj URL adresi (/).
2. U adresnu traku preglednika unijeti /**nepostojeći**.
3. Pritisnuti Enter za učitavanje stranice.
4. Provjeriti da li je korisnik preusmjeren na glavnu stranicu.

Očekivani rezultat

- Sustav prepoznaće da URL /**nepostojeći** ne postoji.
- Korisnik je automatski preusmjeren na glavnu stranicu (URL: /).

Dobiveni rezultat

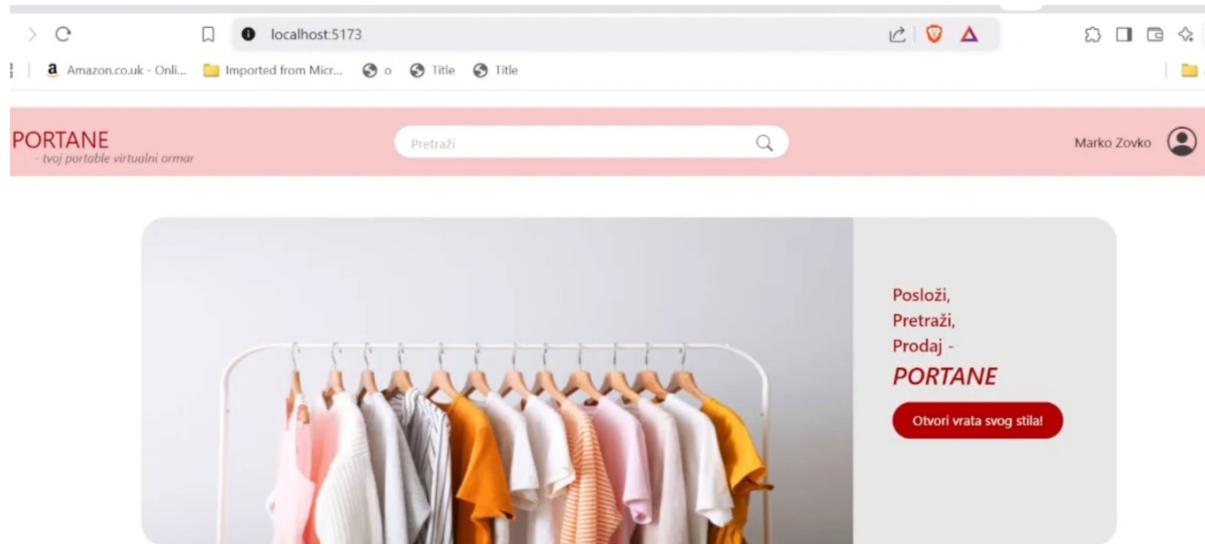
NOVI ORMAR - NOVI PROSTOR ZA STIL

NAZIV ORMARA
Unesite naziv ormara

POLICA
Unesite broj polica

LADICA
Unesite broj ladica

ŠIPKA ZA ODJEĆU
Unesite broj šipka za odjeću



KATEGORIJE

Test: Kreiranje ormara

Ulazi

- **E-mail:** mz@gmail.com
- **Lozinka:** Mcz123
- **Naziv ormara:** Rokov ormara 2
- **Broj polica:** 5
- **Broj ladica:** 4
- **Broj šipki za odjeću:** 10

Koraci ispitivanja

1. Otvoriti aplikaciju na osnovnoj URL adresi (/).
2. Postaviti veličinu prozora preglednika na 1280x680 piksela.
3. Kliknuti na poveznicu "**Prijavi se**".

4. Unijeti e-mail adresu "**mz@gmail.com**" u polje za unos e-maila (ID: `email`).
5. Unijeti lozinku "**Mcz123**" u polje za unos lozinke (ID: `password`).
6. Kliknuti na gumb za prijavu (CSS selektor: `.btn-login`).
7. Nakon uspješne prijave, kliknuti na poveznicu "**Kreiraj ormara**".
8. Unijeti naziv ormara "**Rokov ormara 2**" u polje za unos naziva ormara (ID: `wardrobe-name`).
9. Unijeti broj polica **5** u polje za unos broja polica (ID: `shelves`).
10. Unijeti broj ladica **4** u polje za unos broja ladica (ID: `drawers`).
11. Unijeti broj šipki za odjeću **10** u polje za unos broja šipki (ID: `hangers`).
12. Kliknuti na gumb za potvrdu kreiranja ormara (CSS selektor: `.btn-create-wardrobe`).
13. Provjeriti da li je ormara uspješno kreiran i da li se korisnik preusmjerava na stranicu s detaljima ormara.

Očekivani rezultat

- Korisnik se uspješno prijavljuje sa unesenim e-mailom "**mz@gmail.com**" i lozinkom "**Mcz123**".
- Nakon što klikne na "**Kreiraj ormara**", korisnik unosi naziv ormara "**Rokov ormara 2**", broj polica **5**, broj ladica **4** i broj šipki za odjeću **10**.
- Sustav uspješno kreira ormara s navedenim podacima i preusmjerava korisnika na stranicu s detaljima novog ormara.

Dobiveni rezultat

The screenshot shows the Portane login interface. At the top, there are two buttons: "REGISTRIRAJ SE" and "PRIJAVI SE". Below these are input fields for "Email" (mz@gmail.com) and "Lozinka" (represented by five asterisks). A red "Prijavi se" button is at the bottom of the form.

The screenshot shows the user profile page for "Marko Zovko". It features a large circular profile picture placeholder. Below it, the name "Marko Zovko" and email "mz@gmail.com" are displayed. Statistics show "UKUPNO ORMARA: 0" and "UKUPNO ARTIKALA: 0". Two buttons are present: a white "DODAJ NOVI ORMAR" button and a brown "IZGRADI ODJEVNU KOMBINACIJU" button. At the bottom, a section titled "MOJI ORMARI (0)" includes a search bar with the placeholder "Pretraži svoje ormare".

The screenshot shows a web browser window with the URL `localhost:5173/create-wardrobe`. The page title is "NOVI ORMAR - NOVI PROSTOR ZA STIL". It contains four input fields: "NAZIV ORMARA" with value "Rokov ormara 2", "POLICA" with value "5", "LADICA" with value "4", and "ŠIPKA ZA ODJEĆU" with value "10". A red button at the bottom right labeled "Kreiraj ormara" is visible.

The screenshot shows a web browser window with the URL `localhost:5173/create-wardrobe`. The page title is "NOVI ORMAR - NOVI PROSTOR ZA STIL". It contains one input field: "Unesite broj šipka za odjeću" with placeholder text "Unesite broj šipka za odjeću". A red button at the bottom right labeled "Kreiraj ormara" is visible.

The screenshot shows a web browser window with the URL `localhost:5173/create-wardrobe`. The page title is "MOJI ORMARI (1)". It displays a single item: "Rokov ormara 2" with a small icon of a double-door wardrobe. Below it, the details "Polica: 4", "Ladica: 10", and "Šipki za odjeću: 5" are listed. A search bar "Pretraži svoje ormare" is at the top right.

Korištene tehnologije i alati

1. Programski jezici:

- JavaScript (ES6): Odabran za razvoj frontenda zbog svoje fleksibilnosti, široke primjene u web razvoju i podrške za moderne biblioteke poput Reacta.
- JDK (21): Java Developer Toolkit.

2. Radni okviri i biblioteke:

- React (18.3.1): Popularna JavaScript biblioteka koja omogućuje modularan i komponentni pristup razvoju korisničkih sučelja. Odabrana je zbog lakoće učenja, velike zajednice i mogućnosti ponovne upotrebe komponenata.
- Bootstrap (5.3.3): CSS okvir za izradu responzivnih korisničkih sučelja. Njegov unaprijed definiran sustav rešetki i komponente značajno ubrzavaju razvoj dizajna, dok podrška za moderne preglednike omogućuje široku kompatibilnost.

- Axios (1.7.7): JavaScript biblioteka koja se koristila za izradu HTTP zahtjeva serveru što je olakšalo interakciju s API-ima poslužitelja.
- Spring Boot (3.3.5): Odabran za backend razvoj zbog svoje jednostavne integracije s bazama podataka i podrške za REST API-eve.
- Leaflet (1.9.4): JavaScript biblioteka za izradu interaktivnih mapa. Koristila se kako bi korisnik na jednostavan i intuitivan način mogao u aplikaciju staviti lokaciju ormara.

3. Baza podataka:

- PostgreSQL (15.4): Odabrana je zbog svoje pouzdanosti, naprednih SQL mogućnosti i podrške za transakcije.

4. Razvojni alati:

- IntelliJ IDEA (2024.3): Omogućuje snažne alate za razvoj, refaktoriranje i debuggiranje backend koda. IntelliJ je poznat po podršci za Spring Boot, što ubrzava rad s konfiguracijama.
- CLion (2024.3): Korišten za razvoj određenih logičkih dijelova aplikacije. Njegova podrška za više jezika i integracija s alatima za testiranje olakšavaju razvoj složenijih dijelova aplikacije.
- VS Code (1.96): Editor odabran za frontend razvoj zbog svoje brzine, prilagodljivosti i širokog izbora dodataka i ekstenzija koje ubrzavaju razvojni proces.
- GitHub: Koristi se za verzioniranje i suradnju u timu. Platforma omogućuje praćenje promjena, integraciju CI/CD procesa i jednostavnu kolaboraciju među članovima tima.
- Swagger UI: Alat za automatsko generiranje interaktivne API dokumentacije što je olakšalo komunikaciju između frontend i backend tima te uvjek ubrzalo razvoj aplikacije i omogućilo lakše otklanjanje grešaka.
- Maven (4.0.0): Alat za automatizaciju izrade Java projekta. Koristi se za kompilaciju, testiranje i izrade paketa.

5. Alati za ispitivanje:

- Selenium IDE (4.12): Jednostavan alat za snimanje i reprodukciju korisničkih interakcija, idealan za brzo kreiranje testnih slučajeva za korisničko sučelje.
- Selenium WebDriver (4.12): Napredniji alat za detaljno testiranje funkcionalnosti aplikacije, koji omogućuje automatizaciju interakcija s preglednicima.

6. Alati za razmještaj:

- Render: Cloud platforma za jednostavno hostanje aplikacije. Odabrana je zbog svoje jednostavnosti, integracije s GitHub-om i mogućnosti automatizacije razmještaja putem CI/CD procesa.
- MinIO (2024.12): Korišten za pohranu korisničkih datoteka zbog podrške za S3 protokol i jednostavne integracije s backend aplikacijom.

7. Mikroservisi:

- Weatherbit.io: API odabran za integraciju vremenskih podataka zbog svoje preciznosti i jednostavne implementacije. Omogućuje prikaz relevantnih informacija o vremenu za korisnike aplikacije.

Opis prisutpa aplikaciji na javnom poslužitelju

1. Instalacija

Preduvjeti:

1. JDK (21)
2. Spring Boot (3.3.5)
3. npm (11.0.0)
4. Maven (4.0.0)

Predlaže se svima da na stranici (<https://github.com/davidvodopija/portane/wiki/7.-Tehnologije-za-implementaciju-aplikacije>) pregledaju koje su se tehnologije koristile kako bi se osigurao ispravan rad aplikacije.

Kod web-aplikacije lokalno preuzimamo pomoću naredbe

```
git clone https://github.com/davidvodopija/portane.git
```

U folderu *front* moramo instalirati ovisnosti pomoću naredbe

```
npm install
```

2. Postavke

.env file treba se nalaziti u /back/portane i treba imati zadane sljedeće atribute:

Vjerodajnice i URL na bazu podataka pokrenutu u oblaku:

```
DB_URL=
```

```
DB_USERNAME=
```

```
DB_PASSWORD=
```

Ključ za potpisivanje JWT tokena:

```
SIGNING_KEY=
```

Trajanje Cookies-a:

```
REFRESH_TOKEN_EXPIRY=604800000
```

```
ACCESS_TOKEN_EXPIRY=86400000
```

Link i vjerodajnice za Minio (poslužitelj za fotografije):

```
MINIO_SECRET=
```

```
MINIO_ACCESS=
```

```
MINIO_LINK=
```

Najveća dopuštena veličina slike i zahtjeva

```
MAX_FILE_SIZE=25MB
```

```
MAX_REQUEST_SIZE=30MB
```

Ključ za pristup servisu vremenskoj prognozi

```
WEATHERBIT_API_KEY=
```

Vjerodajnice za OAuth:

```
GOOGLE_CLIENT_ID=
```

```
GOOGLE_CLIENT_SECRET=
```

```
BASE_URL=
```

3. Pokretanje aplikacije

Frontend development environment pokrećemo naredbom u folderu front:

```
npm run dev
```

U source folderu za backend. Uz ovu je naredbu potrebno priložiti .env:

```
.\mvnw clean package -DSkipTests
```

4. Primjer pokretanja na platformi Render

U fodleru (<https://github.com/davidvodopija/portane/tree/main/.github/workflows>) nalaze se .yaml datoteke za automatsku izgradnju aplikacije.

- Postavljanje na Render
 - Prijavite se na Render.
 - Kreirajte novi Web Service i povežite ga s vašim GitHub repozitorijem pomoću Deploy Hooka koji osigurava da će se aplikacija postaviti na Render nakon izgradnje.
 - Dodajte environmental varijable.
- Pokretanje aplikacije Render će automatski preuzeti repozitorij, instalirati ovisnosti i pokrenuti aplikaciju. Nakon deploya, aplikaciji možete pristupiti putem generiranog URL-a.

Opis prisutpa aplikaciji na javnom poslužitelju

Aplikaciji korisnici mogu pristupiti putem sljedeće poveznice: <https://portane-front.onrender.com/>. Korisnik može ostati anoniman i pregledavati podijeljene articke i oglase, ali ne može izrađivati vlastite ormare i u njih dodavati articke. Za to se mora registrirati. Korisnik se može registrirati i kao oglasivač (mora imati sliku) nakon čega može izrađivati galerije vlastitih articala.

Važno je napomenuti da servis za dobavljanje prognoze ima limit na 50 API poziva dnevno sto se može brzo potrošiti.

Izrada projekta Virtualni Ormar bila je zahtjevan, ali izuzetno edukativan proces koji je uključivao planiranje, razvoj i testiranje složenog sustava s različitim funkcionalnostima. Projekt je imao za cilj stvoriti aplikaciju koja olakšava organizaciju i upravljanje odjevnim predmetima, omogućujući korisnicima brži i jednostavniji pregled njihovih ormara, prijedloge odjevnih kombinacija te dijeljenje odjeće s drugim korisnicima.

Tijekom razvoja prepoznati su brojni tehnički izazovi, od osmišljavanja arhitekture do konkretne implementacije funkcionalnosti. Nadalje, dizajn prilagodljivog modela ormara, koji omogućuje korisnicima da definiraju strukturu svojih ormara prema vlastitim potrebama, bio je tehnički kompleksan, ali uspješno riješen. Još jedan značajan izazov bilo je upravljanje velikim brojem articala i optimizacija pretraživanja, kako bi aplikacija ostala brza i intuitivna čak i kod velikih količina podataka. Posebno zahtjevna bila je implementacija funkcionalnosti prijedloga odjevnih kombinacija, koja je zahtjevala razumijevanje kako algoritmi mogu uzeti u obzir različite parametre, poput vremenskih uvjeta, aktivnosti korisnika i stilskih preferencija.

Rad u timu bio je ključan za uspjeh ovog projekta. Suradnja između članova omogućila je podjelu zadataka prema stručnosti, čime je svatko dao svoj doprinos u određenim aspektima projekta, od frontend razvoja i dizajna korisničkog sučelja, do implementacije složenih backend funkcionalnosti i upravljanja bazom podataka. Redovite konzultacije, brainstorming sesije i agilni pristup upravljanju projektom pomogli su timu da ostane fokusiran na zajednički cilj. Ipak, koordinacija među članovima, posebno u trenucima kada su se zadaci međusobno isprepletali, predstavljala je izazov koji je zahtjevao jasniju komunikaciju i definiranje prioritetnih koraka. Ova iskustva dodatno su osnažila tim i razvila vještine rješavanja problema u grupnom okruženju. U komunikaciji između članova ponajprije smo se služili Discord aplikacijom kojom smo lagano uspostavili grupne pozive i „dijelili ekrane“ kako bi vise ljudi moglo raditi na rješenju istog problema.

Kroz ovaj projekt stečena su brojna znanja i vještine. Članovi tima su se upoznali s razvojem prilagodljivih korisničkih sučelja koristeći moderne tehnologije poput Reacta i Bootstrapa, kao i s naprednim tehnikama upravljanja bazama podataka pomoću PostgreSQL-a. Integracija vanjskih usluga putem API-ja proširila je razumijevanje tima o radu s distribuiranim sustavima, dok je korištenje alata poput Minio-a za pohranu podataka omogućilo bolje razumijevanje upravljanja datotekama u oblaku. Ipak, prepoznato je da bi dodatna stručnost u područjima poput algoritama preporuka, naprednog testiranja sustava i optimizacije performansi doprinijela kvalitetnijoj realizaciji projekta.

Ostvareni su svi glavni ciljevi aplikacije, ali ostalo je nekoliko manje važnih funkcionalnosti koje nisu implementirane. To su mogućnost brisanja vlastitog korisničkog profila ili izmjena postavki istog. Također, nije implementirana mogućnost personalizacije stranice (npr. dark mode) ili promjena jezika stranice.

Budući rad na aplikaciji obuhvaća niz mogućnosti za unapređenje. Planira se implementacija chata između korisnika kako bi mogli međusobno komunicirati i razmjenjivati savjete o odjevnim kombinacijama. Dodatno, proširenje postojećeg modela articala uključivanjem više kategorija i stilova odjeće omogućilo bi detaljniju

personalizaciju. Razvoj naprednih algoritama za prijedloge odjevnih kombinacija, temeljenih na povijesti korisnika i vremenskim uvjetima, značajno bi poboljšao korisničko iskustvo. Također, planira se integracija dodatnih API-ja za usluge poput modnih savjeta i detaljnih informacija o materijalima odjeće.

Dugoročne perspektive uključuju razvoj mobilne aplikacije, koja bi omogućila korištenje Virtualnog Ormara u pokretu, te poboljšanje vizualizacije ormara putem interaktivnih 3D prikaza. Osim toga, povezivanje s modnim trendovima i trgovinama otvara mogućnost komercijalizacije projekta, čime bi se dodatno proširila korisnička baza. Budući rad mogao bi uključiti i funkcionalnosti poput integracije umjetne inteligencije za prepoznavanje odjevnih predmeta putem slike te naprednih analitika za korisnike kako bi bolje razumjeli svoje modne preferencije.

Zaključno, projekt Virtualni Ormar ne samo da je ispunio postavljene ciljeve, već je i stvorio čvrste temelje za daljnji razvoj i unaprjeđenje. Iskustva stečena tijekom ovog procesa neprocjenjiva su za buduće projekte, a suradnja u timu dodatno je osnažila vještine komunikacije, rješavanja problema i upravljanja složenim sustavima. Perspektive za nastavak rada na ovom projektu izuzetno su obećavajuće, čime se otvara mogućnost da Virtualni Ormar postane jedan od vodećih digitalnih alata u upravljanju osobnom garderobom.

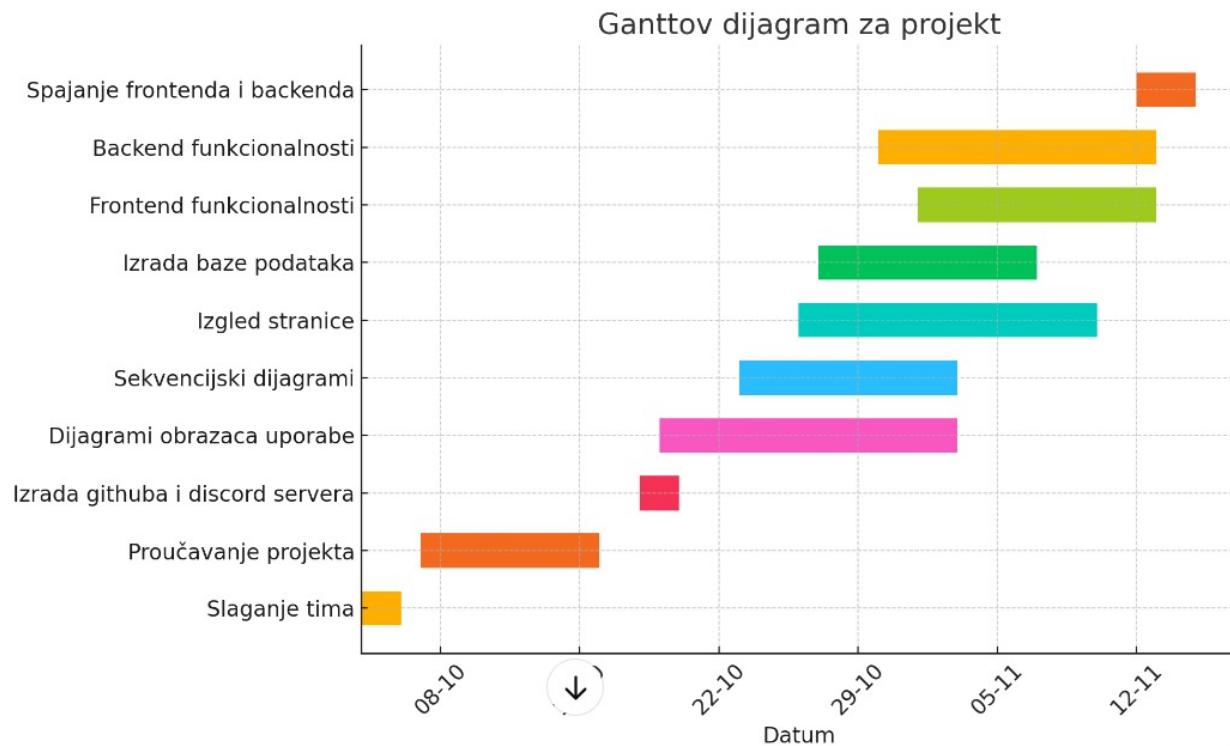
Kontinuirano osvježavanje Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.

1. Programsко inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/> books/SE
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Astah Community, <http://astah.net/editions/uml-new>
7. <https://spring.io/projects/spring-data-jpa>
8. <https://react.dev/>
9. <https://getbootstrap.com/>
10. <https://www.w3schools.com/>
11. <https://github.com/alan2207/bulletproof-react/tree/master>
12. <https://youtu.be/SqcY0GIETPk?si=rUiMQkRExZkszzql>, React Tutorial for Beginners : Programming with Mosh
13. weatherbit.io
14. https://www.w3schools.com/html/html5_geolocation.asp
15. <https://react-leaflet.js.org/>
16. <https://dev.to/hackmamba/how-to-implement-oauth2-in-web-apps-in-5-easy-steps-a-beginners-guide-2mgc>

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Opis projektnog zadatka	Marko Ćiril Zovko	01.11.2024
0.2	Analiza zahtjeva	Marko Cindrić	03.11.2024
0.3	Dijagram obrazaca uporabe	Marko Cindrić	05.11.2024
0.4	Sekvencijski dijagram	Marko Cindrić	05.11.2024
0.5	Dijagram baze podataka	Marko Cindrić, Boris Španić	10.11.2024

Rev.	Opis promjene/dodataka	Autori	Datum
0.6	Konceptualni dijagram razreda	Marko Cindrić	12.11.2024
0.7	Arhitektura i dizajn sustava	Boris Španić	14.11.2024
0.8	Gantogram	Marko Ćiril Zovko	18.11.2024
0.9	Dijagram aktivnosti	Marko Cindrić	21.12.2024
1.0	Dijagram stanja	Marko Cindrić	28.12.2024
1.1	Dijagram komponenta	Marko Cindrić	15.01.2025
1.2	Dijagram razmještaja	Marko Cindrić	19.01.2025
1.3	Prikaz aktivnosti grupe	Marko Ćiril Zovko	19.01.2025
1.4	Popis literature	Marko Ćiril Zovko	19.01.2025
1.5	Tehnologije za implementaciju aplikacije	Marko Ćiril Zovko	19.01.2025
1.6	Popis literature	Marko Ćiril Zovko	20.01.2025
1.7	Ispitivanje programskog rješenja	Boris Španić, Marko Ćiril Zovko	21.01.2025
1.8	Upute za puštanje u pogon	David Vodopija, Marko Cindrić	21.01.2025
1.9	Prikaz aktivnosti grupe	Marko Ćiril Zovko	22.01.2025
2.0	Dijagram razreda	Marko Cindrić	23.01.2025
2.1	Zaključak i budući rad	Marko Ćiril Zovko	23.01.2025

Plan rada



Dnevnik sastajanja

1. Sastanak, 1. Laboratorijska vježba

Datum: 7. listopada Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- sastanak s asistentom i demonstratorom
- početak projekta

2. Sastanak

Datum: 8. listopada Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- izrada discord servera
- odabir teme projekta

3. Sastanak, 2. Laboratorijska vježba

Datum: 14. listopada Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- sastanak s asistentom i demonstratorom
- funkcionalnosti projekta

- dijagrami obrazaca uporabe

4. Sastanak

Datum: 18. listopada Prisustvovali: Marko Cindrić, Marko Ćiril Zovko

Teme sastanka:

- početak dokumentacije
- dijagrami obrazaca uporabe

5. Sastanak

Datum: 20. listopada Prisustvovali: Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić

Teme sastanka:

- podjela rada frontenda i backenda
- izrada githuba

6. Sastanak, 3. Laboratorijska vježba

Datum: 21. listopada Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- sastanak s asistentom i demonstratorom
- funkcionalnosti projekta
- dijagrami obrazaca uporabe
- sekvenički dijagrami

7. Sastanak

Datum: 23. listopada Prisustvovali: Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić

Teme sastanka:

- rad na backendu u frontendu
- rasprava o izgledu stranice

8. Sastanak

Datum: 26. listopada Prisustvovali: Marko Cindrić, Marko Ćiril Zovko

Teme sastanka:

- opis plana projekta
- sekvenički dijagrami
- dijagrami obrazaca uporabe

9. Sastanak, 4. Laboratorijska vježba

Datum: 28. listopada Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović

Teme sastanka:

- sastanak s asistentom i demonstratorom
- funkcionalnosti projekta
- dijagrami obrazaca uporabe
- sekvencijski dijagrami

10. Sastanak

Datum: 29. listopada Prisustvovali: Marko Cindrić, Marko Ćiril Zovko

Teme sastanka:

- opis plana projekta
- sekvencijski dijagrami
- dijagrami obrazaca uporabe
- funkcionalni zahtjevi projekta

11. Sastanak

Datum: 30. listopada Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- funkcionalni zahtjevi projekta
- sekvencijski dijagrami
- rad na frontendu
- rad na backendu

12. Sastanak

Datum: 31. listopada Prisustvovali: Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić

Teme sastanka:

- rad na frontendu
- rad na backendu
- funkcionalnost registracije
- baza podataka

13. Sastanak, 5. Laboratorijska vježba

Datum: 4. studenog Prisustvovali: David Vodopija, Lara Topalović

Teme sastanka:

- sastanak s asistentom i demonstratorom
- baza podataka

14. Sastanak, 6. Laboratorijska vježba

Datum: 11. studenog Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- sastanak s asistentom i demonstratorom
- pregled funkcionalnosti
- dijagram razreda

15. Sastanak

Datum: 14. studenoga Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- opis arhitekture sustava
- spajanje frontenda i backenda
- priprema za predaju projekta

16. Sastanak

Datum: 13. prosinca

Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- nastavak projekta
- funkcionalnosti

17. Sastanak, 8. Laboratorijska vježba

Datum: 16. prosinca

Prisustvovali: Lara Topalović, David Vodopija

Teme sastanka:

- sastanak s asistentom i demonstratorom
- osposobljene funkcionalnosti

18. Sastanak

Datum: 20. prosinca

Prisustvovali: Marko Cindrić, Marko Ćiril Zovko

Teme sastanka:

- dijagram aktivnosti
- dijagram spajanja

19. Sastanak

Datum: 28. prosinca

Prisustvovali: Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić

Teme sastanka:

- rad na dizajnu stranice
- rad na backendu

20. Sastanak, 9. Laboratorijska vježba

Datum: 9. siječnja Prisustvovali: Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- sastanak s asistentom i demonstratorom
- dizajn stranice
- dijagrami spajanja

21. Sastanak

Datum: 10. siječnja

Prisustvovali: Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić

Teme sastanka:

- rad na backendu i frontendu
- geolokacija ormara

22. Sastanak

Datum: 11. siječnja

Prisustvovali: Marko Cindrić, Marko Ćiril Zovko

Teme sastanka:

- Selenium testovi
- dijagram spajanja

23. Sastanak

Datum: 12. siječnja

Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović

Teme sastanka:

- razvoj kategorija i stilova odjeće
- dijagram aktivnosti

24. Sastanak, 10. Laboratorijska vježba

Datum: 13. siječnja

Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- sastanak s asistentom i demonstratorom
- Selenium testovi
- kategorije i stilovi odjeće

25. Sastanak

Datum: 16. siječnja

Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- dijagram razmještaja
- geolokacija
- search engine

26. Sastanak, 11. Laboratorijska vježba

Datum: 20. siječnja

Prisustvovali: Marko Cindrić, Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko

Teme sastanka:

- funkcionalni zahtjevi projekta
- sekvensijski dijagrami
- rad na frontendu
- rad na backendu

27. Sastanak

Datum: 22. siječnja

Prisustvovali: Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko, Marko Cindrić

Teme sastanka:

- OAuth2 (Google, Facebook, Apple)
- search engine
- dijagram komponenti

28. Sastanak

Datum: 23. siječnja

Prisustvovali: Roko Domović, Boris Španić, David Vodopija, Lara Topalović, Borna Lešić, Marko Ćiril Zovko,

Marko Cindrić

Teme sastanka:

- OAuth2 (Google, Facebook, Apple)
- Završni rad na frontendu i backendu
- Zaključak
- dijagram komponenti

Teme sastanka:

- rad na frontendu
- rad na backendu
- funkcionalnost registracije
- baza podataka

Tablica aktivnosti

Dio projekta	Broj sati	Ime	Prezime
Frontend	76	Roko	Domović
Frontend	58	Borna	Lešić
Frontend	80	Lara	Topalović
Frontend	77	David	Vodopija
Backend	96	Boris	Španić
Devops	29	David	Vodopija
Dokumentacija	58	Marko	Zovko
Dokumentacija	80	Marko	Cindrić

Dijagram pregleda promjena

Commits over time

Weekly from Oct 20, 2024 to Jan 19, 2025



Marko-Cindric's Commits



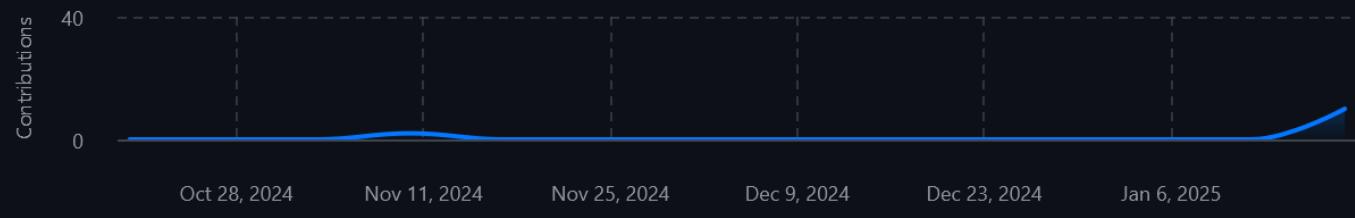
BorisBarca's Commits



davidvodopija's Commits



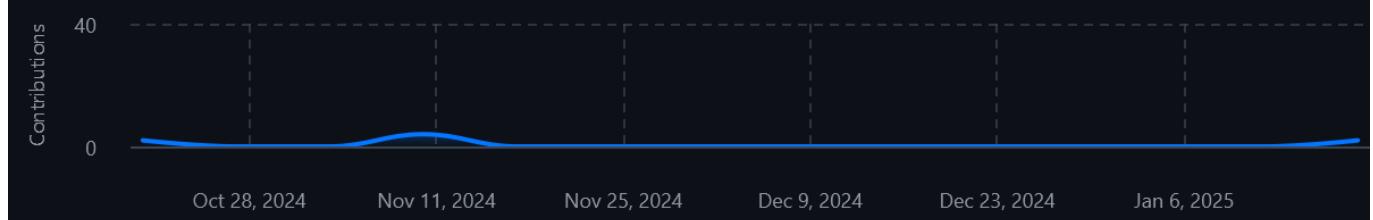
MarkoCirilZovko's Commits



RoKo30's Commits



RoKo30's Commits



Boban067's Commits



borisspanic's Commits



Kjučni izazovi i rješenja

Ključni izazov s kojim smo se susreli bio je savladavanje novih tehnologija kako bi se mogla izraditi kvalitetna web-aplikacija. Mnogi od nas su se prvi put susreli sa Reactom, Bootstrapom, cak nam je i GitHub bio nepoznanica. Ovaj je izazov riješen postepeno, kontinuiranim učenjem na vlastitim pogreškama. Tome je svakako pridonijelo bogatstvo edukativnog sadržaja na internetu.

Najvažnija lekcija koju smo naučili tijekom izrade ovog projekta je da nikad ne smiješ precijeniti vlastite sposobnosti. Bezbroj se puta dogodilo da nešto nije radilo, a mislili smo da će se to brzo i lako popraviti. Nerijetko se popravak takvih „sitnica“ odužio duboko u noćne sate. Izradu projekta svakako je potrebno započeti rano i ozbiljno kako bi se stigla napraviti kvalitetna aplikacija koja ispunjava funkcijeske zahtjeve.

Portane