# Teorijske osnove

## Frontend programiranje

Odnosi se na proces razvoja korisničkog interfejsa i funkcionalnosti koje korisnici direktno vide i interaguju sa njima prilikom korišćenja *web* aplikacija, mobilnih aplikacija ili drugih vrsta softvera. Ovo uključuje sve što se prikazuje u veb *browser*-u ili na ekranu mobilnog uređaja, uključujući tekst, slike, forme za unos podataka, dugmad, menije i druge elemente.

*Frontend* programeri koriste jezike kao što su HTML (*HyperText* *Markup* *Language*) za strukturu veb stranice, CSS (*Cascading* *Style* *Sheets*) za stilizovanje i izgled stranice, i JavaScript za interaktivnost i dinamičnost. Ovi jezici rade zajedno kako bi kreirali funkcionalne i estetski prijatne korisničke interfejse.

Takođe, *frontend* programiranje uključuje rad sa različitim alatima i okvirima (npr. React, Angular, Vue.js) koji olakšavaju razvoj kompleksnih korisničkih interfejsa i omogućavaju bolje upravljanje stanjem aplikacije.

Kroz *frontend* programiranje, programeri se fokusiraju na obezbeđivanje pozitivnog korisničkog iskustva tako što pravilno organizuju informacije, olakšavaju navigaciju i osiguravaju da aplikacija bude odzivna i brza.

## HTML (HyperText Markup Language)

Standardni jezik za označavanje i strukturiranje sadržaja veb stranica. To je osnovni jezik koji se koristi za izgradnju i oblikovanje elemenata koji se prikazuju na veb stranicama. Kroz HTML, možemo definisati različite delove veb stranice, kao što su naslovi, paragrafi, slike, veze, forme i druge komponente.

HTML koristi "oznake" ili "tagove" da označi različite delove teksta i elemenata na stranici. Oznake se obično sastoje od otvarajućeg i zatvarajućeg dela, gde otvarajuća oznaka označava početak elementa, dok zatvarajuća označava kraj. Na primer, <h1> označava početak naslova prvog nivoa (*heading* *level* 1), a </h1> označava kraj tog naslova. Pravilno strukturiran HTML je ključan za pravilno prikazivanje veb stranica na različitim uređajima i veb *browser*-ima.

## CSS (Cascading Style Sheets)

Predstavlja jezik za stilizovanje veb stranica i omogućava definisanje izgleda i prezentacije elemenata ( boje, fontovi, raspored, veličine i druge vizuelne karakteristike ) koje smo strukturirali koristeći HTML.

Omogućava odvajanje dizajna veb stranice od njenog sadržaja, čime se olakšava održavanje i ažuriranje veb stranica. Umesto da stilizujemo svaki element pojedinačno u HTML-u, koristimo CSS da definišemo stilove koji se primenjuju na različite elemente. Na primer, možemo definisati stil za sve naslove (<h1>, <h2>, <h3>, itd.) koji će važiti na svim stranicama na sajtu.

Princip "kaskadnog" (*Cascading*) u CSS-u se odnosi na to da stilovi mogu biti nasleđeni od roditeljskih elemenata, ali ih takođe možemo izmeniti ili nadjačati specifičnim stilovima za određene elemente.

CSS takođe omogućava prilagodljivost i odzivnost veb stranica na različite uređaje i veličine ekrana, što je ključno za kreiranje korisničkih interfejsa koji su funkcionalni i lepo izgledaju na svim platformama.

Ukratko, CSS je jezik koji omogućava oblikovanje i stilizovanje veb stranica kako bi se postigao željeni izgled i korisničko iskustvo.

## JavaScript

Programski jezik koji se često koristi za razvoj dinamičkih i interaktivnih veb stranica. To je jezik koji se izvodi na klijentskoj strani (u veb *browser*-ima korisnika), što znači da se kod izvršava na računarima korisnika, a ne na serverima.

Omogućava programerima da kontrolišu ponašanje elemenata na veb stranici, reaguju na korisničke akcije (kao što su klikovi i unos teksta) i dinamički ažuriraju prikaz stranice bez potrebe za ponovnim učitavanjem celokupne stranice. Takođe se koristi za validaciju podataka unetih u forme, manipulaciju dokumentima i animaciju elemenata.

Pored toga, JavaScript je često korišćen i izvan veb *browser*-a, kao što je u razvoju mobilnih aplikacija koristeći okvire kao što su React Native ili u razvoju serverske aplikacije koristeći Node.js.

U suštini, JavaScript je snažan alat koji omogućava interaktivnost, dinamiku i kompleksne funkcionalnosti na veb stranicama i u drugim vrstama softvera.

## TypeScript

Jezik za programiranje koji proširuje funkcionalnosti JavaScript-a dodavanjem statičkog tipovanja. To znači da programeri mogu definisati tipove podataka (kao što su brojevi, stringovi, objekti, itd.) i koristiti ih u kodu, čime se pomaže u detekciji i sprečavanju potencijalnih grešaka prilikom razvoja softvera.

Osnovni cilj TypeScript-a je da obezbedi bolju strukturu i čitljivost koda, kao i veću sigurnost u pogledu tipova podataka. Programeri mogu definisati tipove za promenljive, funkcije, klase i druge konstrukte, a zatim TypeScript proverava da li se tipovi koriste na ispravan način u toku kompilacije.

Evo nekoliko ključnih karakteristika TypeScript-a:

* Statičko tipovanje: Omogućava programerima da definišu tipove promenljivih i drugih elemenata u kodu pre izvršavanja, čime se smanjuje broj grešaka vezanih za tipove tokom razvoja.
* Objektno-orijentisano programiranje: TypeScript podržava koncepte objektno-orijentisanog programiranja kao što su klase, nasleđivanje, interfejsi i apstraktne klase.
* Modularnost: Omogućava organizaciju koda u module, čime se olakšava ponovna upotreba i održavanje koda.
* Kompatibilnost sa JavaScript-om: TypeScript je nadskup JavaScript-a, što znači da je moguće koristiti većinu postojećeg JavaScript koda u TypeScript aplikacijama.
* Razvoj u timu: Poboljšava saradnju i razvoj u timu kroz bolje razumevanje koda zbog definisanih tipova i jasnijeg interfejsa.

TypeScript je postao popularan u razvoju veb i serverskih aplikacija, posebno u velikim projektima gde je važna stabilnost i sigurnost tipova podataka.

## Angular

Popularan open-source okvir (*framework*) za razvoj veb aplikacija. Razvila ga je kompanija Google i predstavlja moćan alat koji olakšava kreiranje dinamičkih, interaktivnih i skalabilnih veb aplikacija. To je klijentski (*frontend*) okvir koji se koristi za izgradnju modernih veb aplikacija koje mogu da rade u veb *browser*-ima.

Glavne karakteristike Angular-a uključuju:

* Komponentna arhitektura: Angular se bazira na komponentama, koje su nezavisne, ponovno upotrebljive i samostalne delove korisničkog interfejsa. Ovo olakšava organizaciju i upravljanje kodom.
* Dvosmerna veza podataka: Angular omogućava automatsko ažuriranje korisničkog interfejsa kada se podaci u modelu promene, i obrnuto. To olakšava praćenje i upravljanje podacima.
* Injekcija zavisnosti: Angular koristi koncept injekcije zavisnosti kako bi se omogućila bolja kontrola i organizacija komponenata i servisa.
* Modularnost: Kod se organizuje u module, što doprinosi boljoj organizaciji i skalabilnosti aplikacije.
* Rute i navigacija: Angular pruža mehanizam za definisanje ruta i navigaciju unutar aplikacije, omogućavajući korisnicima da prelaze između različitih delova aplikacije.
* Testiranje: Angular ima ugrađene alate za jednostavno testiranje koda, što doprinosi kvalitetu aplikacije.

Angular omogućava razvoj složenih aplikacija kao što su veb aplikacije, progresivne veb aplikacije (PWA) i veb stranice visokih performansi. Kroz svoje napredne funkcionalnosti, postao je popularan izbor za razvoj modernih veb aplikacija.

## Mikrofrontend

Arhitektonski pristup u razvoju veb aplikacija gde se aplikacija dekomponuje na manje, autonomne delove (mikrofrontendi) koji mogu da se razvijaju, testiraju, implementiraju i održavaju nezavisno. Ovi mikrofrontendi predstavljaju manje delove korisničkog interfejsa koji rade zajedno kako bi činili celinu aplikacije.

Svaki mikrofrontend je samostalan modul ili aplikacija koji je fokusiran na specifičnu funkcionalnost ili deo korisničkog interfejsa. Oni mogu biti razvijani od strane različitih timova ili pojedinaca, a zatim integrisani u glavnu aplikaciju. Ovaj pristup omogućava agilnost u razvoju, ubrzava implementaciju novih funkcionalnosti i olakšava održavanje kompleksnih veb aplikacija.

Prednosti mikrofrontend arhitekture uključuju:

* Svaki mikrofrontend može da bude razvijan od strane nezavisnog tima, omogućavajući paralelni razvoj i ubrzanje vremena do implementacije.
* Svaki mikrofrontend može da se implementira, testira i održava nezavisno, što olakšava upravljanje kodom i brže isporučivanje promena.
* Omogućava skalabilnost na nivou komponenata, gde se resursi mogu optimalno koristiti prema zahtevima svake komponente.
* Mikrofrontendi su modularni i mogu se koristiti u različitim delovima aplikacije ili u različitim aplikacijama, što doprinosi ponovnoj upotrebi koda.
* Moguće je optimizovati učitavanje stranica, jer se samo potrebni mikrofrontendi učitavaju za određenu stranicu.

Važno je naglasiti da, iako mikrofrontendi donose brojne prednosti, njihova implementacija zahteva pažljivo planiranje, upravljanje stanjem aplikacije i pravilno upravljanje komunikacijom između mikrofrontenda i glavne aplikacije.

## Mikrofrontend vs. Standardni program

Razlika između mikrofrontend i standardnog (monolitnog) pristupa programiranju leži u tome kako se veb aplikacije razvijaju, organizuju i održavaju. Evo nekoliko ključnih tačaka koje ilustruju razlike između ovih dva pristupa:

1. Arhitektura:

* Mikrofrontend arhitektura se zasniva na dekompoziciji aplikacije na manje, autonomne mikrofrontende. Svaki mikrofrontend je nezavisan modul ili aplikacija koji može da se razvija, testira i implementira nezavisno. Ovo omogućava paralelni razvoj i bržu implementaciju novih funkcionalnosti.
* Monolitna arhitektura podrazumeva razvoj celokupne aplikacije kao jedne celine, gde svi delovi i funkcionalnosti dele istu bazu koda. Ovaj pristup može postati kompleksan i težak za održavanje kako aplikacija raste.

1. Nezavisnost:

* Mikrofrontendi su nezavisni i mogu koristiti različite tehnologije, okvire ili biblioteke. To omogućava različitim timovima da rade na različitim mikrofrontendima sa svojim preferiranim tehnologijama.
* U monolitnoj arhitekturi, svi delovi aplikacije dele isti tehnološki stek, što može ograničiti izbor tehnologija i otežati rad različitim timovima.

1. Skalabilnost:

* Mikrofrontendi omogućavaju skalabilnost na nivou komponenata. Možete skalirati samo određene delove aplikacije koji zahtevaju više resursa ili brže odzive.
* U monolitnoj arhitekturi, skaliranje znači skaliranje cele aplikacije, čak iako neki delovi ne zahtevaju dodatne resurse.

1. Učitavanje i performanse:

* Mikrofrontendi omogućavaju dinamičko učitavanje samo onih delova aplikacije koji su potrebni za određenu stranicu ili funkcionalnost, što može poboljšati performanse i brzinu učitavanja.
* U monolitnoj arhitekturi, cela aplikacija se učitava pri svakom zahtevu, bez obzira na to koji deo korisniku trenutno treba.

1. Održavanje:

* Održavanje mikrofrontend aplikacije je olakšano jer se svaki mikrofrontend može ažurirati nezavisno. Ovo smanjuje rizik od narušavanja funkcionalnosti drugih delova aplikacije.
* U monolitnoj arhitekturi, promene u jednom delu aplikacije mogu imati nepredviđene posledice na ostale delove, što otežava održavanje.

Izbor između mikrofrontend i standardnog programa zavisi od specifičnih zahteva projekta, kompleksnosti aplikacije i razvojnog tima. Svaki pristup ima svoje prednosti i nedostatke, i važno je pažljivo razmotriti koji će odgovarati konkretnom projektu.

## Single-spa

JavaScript okvir (*framework*) otvorenog koda koji omogućava razvoj mikrofrontend arhitekture. Ovaj okvir omogućava integraciju različitih mikrofrontenda u jednu koherentnu veb aplikaciju, omogućavajući im da rade zajedno kao deo istog korisničkog interfejsa.

Koncept Single Spa-a se fokusira na to da svaki mikrofrontend može biti razvijen nezavisno, koristeći različite tehnologije i okvire, i da se integriše u glavnu aplikaciju bez međusobnih konflikata. Single Spa omogućava dinamičko učitavanje mikrofrontenda samo kada su potrebni, čime se smanjuje početno vreme učitavanja aplikacije i poboljšava performanse.

Ključne funkcionalnosti Single Spa-a uključuju:

* Omogućava integrisanje mikrofrontenda u glavnu aplikaciju, bez obzira na to koji je tehnološki stek koristio svaki od njih.
* Mogućnost dinamičkog učitavanja mikrofrontenda prema potrebi, što doprinosi efikasnosti i brzini učitavanja.
* Pruža mehanizam za upravljanje rutama i navigacijom unutar aplikacije kroz različite mikrofrontend delove.
* Omogućava deljenje globalnog stanja između mikrofrontenda kako bi se sačuvala koherentnost aplikacije.
* Pruža metode i događaje za upravljanje životnim ciklusom komponenata, kao što su montiranje, demontiranje i ažuriranje.

Single Spa je popularan alat u svetu razvoja mikrofrontend arhitekture jer omogućava organizaciju i integraciju više mikrofrontenda u jednu aplikaciju sa minimalnim naporima i smanjenjem kompleksnosti.

## Role-based access control (RBAC)

Ovaj model se koristi za upravljanje pravima pristupa korisnicima na osnovu njihovih uloga u organizaciji. Svaka uloga ima *set* dozvoljenih akcija i aktivnosti koje korisnici koji imaju tu ulogu mogu da obavljaju u sistemu.

Osnovne komponente RBAC modela uključuju:

* Korisnici: Osobe, sistemi ili procesi koji zahtevaju pristup informacionom sistemu.
* Uloge: Skupovi prava koji određuju tipične zadatke ili funkcije u organizaciji. Uloge grupišu korisnike na osnovu sličnih potreba za pristupom.
* Dozvole: Prava koja se dodeljuju ulogama, određujući šta korisnici sa određenom ulogom mogu ili ne mogu da rade u sistemu.

Prednosti RBAC-a uključuju jednostavno upravljanje pravima pristupa, lakše održavanje sistema, bolju kontrolu nad sigurnošću i bolje razumevanje sistema u celini. Pored toga, RBAC može značajno doprineti sigurnosti sistema ograničavanjem pristupa osetljivim informacijama samo na osnovu uloga i ovlašćenja koje te uloge imaju.

U implementaciji RBAC-a, važno je pažljivo definisati uloge, njihove ovlašćenja i način dodeljivanja uloga korisnicima kako bi se postigla efikasna i sigurna kontrola pristupa u informacionim sistemima.

## HTTP (Hypertext Transfer Protocol)

Protokol koji se koristi za prenos informacija ( komunikaciju između veb servera i veb klijenata ) na *World Wide Webu* (*WWW*).

Funkcioniše na osnovu klijent-server arhitekture, gde klijent šalje HTTP zahteve (*request*) serveru, a server odgovara na te zahteve. Zahtevi uključuju različite metode, kao što su GET, POST, PUT, DELETE, koje definišu željenu akciju nad resursima na serveru.

Osnovne karakteristike HTTP-a uključuju:

* HTTP je *stateless*, što znači da svaki zahtev koji klijent šalje serveru sadrži sve potrebne informacije i ne zavisi od prethodnih zahteva. Ovo pojednostavljuje proces obrade zahteva na serveru.
* Klijenti šalju HTTP zahteve (*request*) serveru, a server odgovara na te zahteve HTTP odgovorima (*response*), koji sadrže informacije ili tražene resurse.
* HTTP podržava različite metode, uključujući GET (čitanje), POST (slanje podataka), PUT (ažuriranje podataka) i DELETE (brisanje podataka), koje se koriste za manipulaciju resursima.
* URI (*Uniform Resource Identifier*) se koristi za identifikaciju resursa (npr. veb stranica, slika, dokument) na internetu.
* HTTP zaglavlja (*headers*) omogućavaju prenos dodatnih informacija između klijenta i servera, kao što su informacije o tipu sadržaja, sesiji, kolačići i druge relevantne informacije.

HTTP je osnova za komunikaciju na internetu i omogućava pristup veb sadržaju, slanje podataka na servere, preuzimanje resursa i obavljanje različitih interakcija na *World Wide Webu*.

## REST (Representational State Transfer)

Predstavlja arhitektonski stil za dizajniranje distribuiranih sistema, a posebno se često koristi u razvoju web servisa. Ovaj koncept se oslanja na prenos podataka putem HTTP protokola i ima za cilj da obezbedi skalabilnost, pouzdanost, performanse i laku upotrebljivost sistema.

Ključne karakteristike REST arhitekture uključuju:

* Resursi (*Resources*): predstavljaju ključne koncepte u REST arhitekturi. Svaki resurs je identifikovan jedinstvenim URI (*Uniform Resource Identifier*). Na primer, u veb razvoju, resursi mogu biti entiteti kao što su korisnici, proizvodi ili narudžbine.
* HTTP Metode (*Methods*): GET, POST, PUT i DELETE koriste se za izvršavanje operacija nad resursima. Na primer, GET se koristi za čitanje podataka resursa, POST za kreiranje novog resursa, PUT za ažuriranje postojećeg resursa i DELETE za brisanje resursa.
* Reprezentacija (*Representation*): Resursi se mogu predstaviti u različitim formatima, kao što su JSON, XML ili HTML, u zavisnosti od potreba aplikacije i klijenta.
* Bez stanja (*Statelessness*): REST je *stateless* što znači da svaki zahtev od klijenta serveru mora sadržati sve potrebne informacije za obradu tog zahteva, bez oslanjanja na prethodne zahteve.
* Jednostavan interfejs (*Uniform Interface*): REST ima jednostavan i konzistentan interfejs između klijenta i servera, što olakšava razvoj, održavanje i skaliranje aplikacija.
* Hipermediji (*Hypermedia*): REST može da koristi hipermediji (npr. linkovi) da omogući navigaciju kroz aplikaciju i informacije o resursima.

REST se često koristi za izradu API-ja (*Application Programming Interface*) koji omogućava komunikaciju između različitih sistema i servisa na internetu. Ovo je popularan pristup u veb razvoju zbog svoje jednostavnosti, skalabilnosti i široke primenljivosti.

# Specifikacija zahteva

Frontend se sastoji od nekoliko mikrofrontend-a. Oni su : Authentication Mikrofrontend, Dashboard Mikrofrontend, Product Management Mikrofrontend, Category Management Mikrofrontend, Bill Management Mikrofrontend, Review Management Mikrofrontend, User Management Mikrofrontend.

Za svaki mikrofrontend je napravljen poseban Use-Case dijagram. Actor-i će biti predstavljeni preko svojih uloga. Npr. ako piše da je Mušterija to znači da je korisnik ulogovan sa ulogom mušterije.

Paginacija je postupak razdvajanja ili raspoređivanja sadržaja na više stranica kako bi se olakšala navigacija i poboljšala preglednost. Ovo se često koristi u kontekstu veb stranica, aplikacija i baza podataka kako bi se omogućilo korisnicima da pregledaju velike količine informacija bez pretrpanosti na jednoj stranici. U ovoj aplikaciji prikaz svih elemenata kao i rezultat filtriranja elemenata će se prikazivati preko paginacije.

U narednom delu prvo će se prikazivati tabela slučaja korišćenja pa onda graf slučaja korišćenja.

## Authentication Mikrofrontend

|  |  |
| --- | --- |
| Use-Case | Deskripcija |
| Registruj se | Ne registrovana mušterija/prodavac žele da se registruju. Kliknu na dugme da se registruju i unesu informacije za email, lozinku, ime, prezime i izaberu ulogu ( Mušterija ili Prodavac ). Zatim kliknu na dugme da se registruju. Eprodavnica prima informacije i pravi novog korisnika. |
| Verifikacija lozinke | Uneta lozinka od strane ne registrovanog korisnika se proverava da li ispunjava pravila ( lozinka mora biti barem dužine 5 karaktera, mora imati broj i specijalan karakter ) . |
| Prikaži grešku za lozinku | Ova greška se pojavljuje kada uneta lozinka od strane ne registrovanog korisnika ne ispunjava pravila ( lozinka mora biti barem dužine 5 karaktera, mora imati broj i specijalan karakter ). |
| Prikaži grešku za registraciju | Ova greška se pojavljuje u slučaju da već postoji korisnik sa unetim mejlom. |
| Slanje email potvrde | U slučaju da je sve uredu, Eprodavnica šalje potvrdu na uneti mejl koja će važiti određeno vreme. |
| Prikaz obaveštenja da je verifikacija potrebna | U slučaju da je sve uredu prikazuje se obaveštenje da je potrebna mejl potvrda |
| Potvrdi registraciju | Na ovo ne registrovani korisnik dolazi preko linka koji je dobio na mejl za potvrdu. Koda potvrdi mejl, korisnik će moći da se uloguje. |
| Login | Ne ulogovani korisnik se uloguje. Pošalje informacije Eprodavnici a Eprodavnica mu vrati token zajedno sa fingerprint-om koji će trajati 30 min. Kada taj period istekne, moraće da se izloguje pa ponovo uloguje. |
| Verifikacija lozinke i mejla | Provera da li su polja za email i lozinku prazna i provera da li email polje sadrži majmunsko a. |
| Prikaz login greške | Ako su polja za email ili lozinku prazna ili ako u email polju fali majmunsko a prikazaće grešku. |
| Prikaz obaveštenja da je ulogovan | Posle uspešnog logovanja prikazaće obaveštenje da je uspešno ulogovan. |
| Slanje zahteva za zaboravljenu lozinku | U slučaju da je ne ulogovani korisnik zaboravio lozinku ili želi da promeni lozinku, on šalje svoj mejl za promenu lozinke. |
| Verifikacija mejla | Provera da li je dobro unet mejl. |
| Prikaz greške za loše unet mejl | Ako je polje za mejl prazno ili nema majmunsko a ova greška će se pojaviti. |
| Slanje linka za promenu lozinke na email | U slučaju da je sve uredu. Eprodavnica šalje link na mejl za promenu lozinke. |
| Promena lozinke | Korisnik dolazi na ovaj deo preko linka koji je dobio na mejlu. Ima dva polja koja mora uneti a to su lozinka i ponovo uneta lozinka. Ako je sve uredu onda Eprodavnica menja lozinku. |
| Verifikacija unete lozinke | Provera da li su polja za lozinku i ponovo unetu lozinku iste, da li sadrži broj i specijalan karakter. |
| Prikaz greške za unos | Ako su lozinke različite, fali im broj ili specijalan karakter pojavljuje se greška. |
| Potvrda admin akaunta | Kao i za mušteriju i prodavca, admin mora da potvrdi svoj mejl. Posle potvrde moći će da se uloguje na taj korisnički nalog. |



## Dashboard Mikrofrontend

|  |  |
| --- | --- |
| Use-Case | Deskripcija |
| Pregled svih produkata | Na glavnoj stranici svi korisnici imaju pregled svih produkata koji nisu povučeni . |
| Filtriranje produkata | Na glavnoj stranici svi korisnici mogu da filtriraju sve produkte koji nisu povučeni. Filtrira se po : nazivu, ceni, kategorijama, ocenama. |
| Resetovanje filtracije | Na glavnoj stranici svi korisnici mogu da resetuju filtraciju produkata. Kada se resetuje, svi produkti će se vratiti a polja za filtraciju će biti prazna. |



## Product Management Mikrofrontend

|  |  |
| --- | --- |
| Use-Case | Deskripcija |
| Pregled produkta detaljno | Svi korisnici pregledaju produkt detaljno. Na stranici se prikazuju : kategorije kojima produkt pripada, slika, ocena, cena, deskripcija i sve recenzije vezane za produkt. Recenzije se prikazuju preko paginacije. Jedna strana sadrži 4 recenzije. |
| Pregled svih recenzija vezano za produk | Svi korisnici vide recenzije. Recenzije se prikazuju preko paginacije. Jedna strana sadrži 4 recenzije. |
| Filtriranje recenzija vezano za produkt | Svi korisnici filtriraju recenzije preko ocena. Rezultat se prikazuje preko paginacije. |
| Resetovanje filtriranja recenzija vezano za produkt | Svi korisnici resetuju rezultat filtriranja recenzije što dovodi do ponovnog prikaza svih recenzija. |
| Dodaj u korpu | Mušterija dodaje u korpu produkt. Ako je korpa bila prazna Eprodavnica pravi novi račun i dodaje artikal sa produktom. Ako korpa nije prazna onda Eprodavnica samo dodaje artikal sa produktom. Zatim Eprodavnica prikazuje obaveštenje da je produkt dodat u korpu. |
| Unese se željena količina produkta | Pre nego što mušterija doda produkt u korpu, upisuje količinu željenog produkta. |
| Verifikacija količine | Proverava se da li je polje za količinu prazno i da li je manje od 1. |
| Greška za unetu količinu | Ako je polje prazno ili je količina manja od 1 prikazuje se greška. |
| Dodaj u wishlist | Mušterija dodaje produkt u wishlist. Wishlist je lista produkata za koje je mušterija zainteresovana. Preko nje prati kada je produkt na akciji. |
| Ukloni iz wishlist-a | Mušterija uklanja produkt iz wishlist-a |
| Napravi recenziju | Mušterija pravi recenziju. U njoj ostavlja ocenu i komentar za dati produkt. Muže da uradi ovu operaciju ako je kupila produkt, |
| Edituj recenziju | Mušterija menja sadržaj recenzije. |
| Pregled istorije kupovine | Mušterija pregleda sve produkte koje je kupovao u prošlosti. Produkti se prikazuju preko paginacije. Jedna stanica prikazuje 9 produkata. |
| Pregled wishlist-a | Mušterija pregleda sve produkte koje je stavio u wishlist. Wishlist je lista produkata za koje je mušterija zainteresovana. Produkti se prikazuju preko paginacije. Jedna stanica sadrži 9 produkata. |
| Filtriranje produkata | Mušterija filtrira podatke iz wishlist-a i istorije kupovine. Filtrira se po : nazivu, ceni, kategoriji (tipovima), ocenama. |
| Resetovanje filtriranja produkata | Mušterija resetuje rezultat filtriranja što dovodi do ponovnog prikaza svih produkata a polja za kriterijume filtriranja su prazna. |
| Edit produkta | Prodavac menja vrednosti za produkt. Menja se slika, naziv, deskripcija, cena i akcija. |
| Dodavanje produkta | Prodavac dodaje produkt u Eprodavnicu. Postavlja sliku (opciono), naziv, deskripciju, cenu. |
| Verifikacija unetih podataka za produkt | Kod pravljenja i menjanja vrednosti produkta se proveravaju sva polja za unos. |
| Greška sa unetim podacima za produkt | Kod pravljenja i menjanja vrednosti produkta se javlja greška ako su polja prazna, cena je manja od 1 i akcija manja od 0 ili veća od 100. |
| Postavljanje akcije | Prodavac postavlja akciju za produkt. Svaki put kada se prikazuje produkt Eprodavnica će izračunati cenu sa akcijom. |
| Verifikacija unetih podataka za akciju | Provera da li je polje za akciju prazno. |
| Greška sa unetim podacima za akciju | Ako je polje za unos akcije prazno, prikazaće grešku. |
| Povlačenje produkta | Admin i prodavac povlače produkt. Više se ne može kupovati. Povlačenje je posebno za admina i prodavca tj. da bi se produkt kupovao mora biti odobren i od admina i od prodavca. |
| Slanje na mejl svim mušterijama koji su wishlist-ovali produkt | Kada se produkt povuče ili se postavi akcija sve mušterije koje su stavile produkt na wishlist biće obaveštene. |
| Pregled produkata prodavca | Prodavac pregleda sve svoje produkte. Oni se priakzuju preko paginacije. Jedna stranica sadrži 9 produkata. |
| Filtriranje produkata prodavca | Prodavac filtrira svoje produkte preko naziva, cene, kategorija (tipova) i ocena. |
| Resetovanje filtriranja produkata prodavca | Prodavac resetuje rezultat filtriranja svojih produkata čime se ponovo prikazuju svi a polja za kriterijume filtriranja su prazna. |
| Pregled svih produkata | Admin pregleda sve produkte na Eprodavnici. Produkti se prikazuju preko paginacije. Jedna stranica prikazuje 9 produkata. |
| Filtriranje svih produkata | Admin filtrira prikaz svih produkata na Eprodavnici preko naziva, cene, kategorija (tipova) i ocena. |
| Resetovanje filtriranja svih produkata | Admin resetuje rezultat filtriranja svih produkata čime se ponovo prikazuju svi a polja za kriterijume filtriranja su prazna. |



## Category Management Mikrofrontend

|  |  |
| --- | --- |
| Use-Case | Deskripcija |
| Pregled svih kategorija | Admin ima pregled svih kategorija ( tipova ). Kategorije se prikazuju preko paginacije. Maksimalan broj prikazivanja kategorije na jednoj strani je 12. |
| Filtriranje kategorija | Admin može da filtrira kategorije po nazivu. Rezultat se prikazuje preko paginacije. |
| Resetovanje filtriranja kategorija | Admin posle filtriranja ima mogućnost da resetuje prikaz kategorija tj. ponovo će se prikazivati sve kategorije a polje za unos naziva će biti prazno. |
| Izmena kategorije | Admin menja naziv kategorije. |
| Napravi se nova kategorija | Admin pravi novu kategoriju. |
| Verifikacija naziva | Kada admin unese naziv kategorije, doći će do provere da li se taj naziv već koristi i da li je polje za unos naziva kategorije prazno. |
| Prikaz greške | Prikazaće se greška ako unet naziv kategorije već postoji ili ako je polje za unos naziva kategorije prazno. |



## Bill Management Mikrofrontend

|  |  |
| --- | --- |
| Use-Case | Deskripcija |
| Pregled korpe | Mušterija ima pregled svih artikala ( produkti zajedno sa količinom ) koje je stavila u korpu. Artikli se prikazuju preko paginacije. Jedna strana prikazuje 12 artikala. |
| Brisanje artikla iz korpe | Mušterije briše artikal iz korpe. Kada klikne na izbriši artikal će nestati iz tabele. |
| Plaćanje | Mušterija plaća ukupnu sumu korpe. |
| Slanje računa na mejl | Posle plaćanja, Eprodavnica šalje na mejl mušteriji račun u kojem će biti sadržaj korpe zajedno sa datumom kada je plaćeno. |
| Pregled računa mušterije | Mušterija ima pregled svih svojih računa. Računi se prikazuju preko paginacije. Jedna strana prikazuje 12 računa. |
| Pregled računa | Mušterija i admin imaju detaljan pregled računa. Artikli se prikazuju preko paginacije. Jedna strana prikazuje 12 artikala. |
| Pregled svih računa | Admin ima pregled svih računa koji su ikada napravljeni u Eprodavnici. Računi se prikazuju preko paginacije. Jedna strana prikazuje 12 računa. |



## Review Management Mikrofrontend

|  |  |
| --- | --- |
| Use-Case | Deskripcija |
| Pregled recenzija mušterije | Mušteriji se prikazuju sve recenzije koje pripadaju njoj. Prikazuju se preko paginacije. Jedna strana sadrži 6 recenzija. |
| Filtriranje recenzija mušterije | Mušterija filtrira njene recenzije preko ocena i datuma. Rezultat se prikazuje preko paginacije. Jedna strana sadrži 6 recenzija. |
| Resetovanje filtriranja recenzija mušterije | Mušterija resetuje rezultat filtriranja što dovodi do ponovnog prikazivanja svih recenzija koje pripadaju mušteriji. Polja za datume i ocene postaju prazna ako već nisu bila. |
| Promena recenzije | Admin i mušterija menjaju recenziju. |
| Prikaz obaveštenja o promeni recenzije | Posle promene recenzije , adminu i mušteriji se prikazuje obaveštenje o promeni recenzije. |
| Prikaz greške za komentar polje | Ako je polje za komentar prazno, postaće crveno . Dugme za promenu će biti zaključano. |
| Verifikacija komentar polja | Provera da li je polje komentar prazno. |
| Brisanje recenzije | Admin i mušterija brišu recenziju. |
| Prikaz obaveštenja o brisanju recenzije | Posle uspešnog brisanja recenzije prikazuje se obaveštenje da je recenzija obrisana. |
| Izračunavanje srednje vrednosti ocene produkta | Posle brisanja ili promene od strane mušterije ili admina Eprodavnica ponovo izračunava srednju vrednost produkta. |
| Pregled svih recenzija | Adminu se prikazuje sve recenzije preko paginacije. Jedna strana je 6 recenzija. |
| Resetovanje filtriranja svih recenzija | Admin resetuje rezultat filtriranja što dovodi do ponovnog prikaza svih recenzije. Polja za datume i ocene postaju prazna ako već nisu bila. |
| Filtriranje svih recenzija | Admin filtrira recenzije po datumima i ocenama. Rezultat se prikazuje preko paginacije. Jedna strana je 6 recenzija. |



## User Management Mikrofrontend

|  |  |
| --- | --- |
| Use-Case | Deskripcija |
| Promena ličnih informacija | Admin, prodavac ili mušterija menjaju svoje lične informacije. |
| Verifikacija unetih vrednosti | Provera mejl, lozinke, ime i prezime polja. |
| Prikaži grešku za pogrešne informacije | Greška će biti ako je jedno od polja prazno, ako mejlu fali majmunsko a, ako lozinki fali broj i specijalan karakter ili je lozinka manje dužine od 5 karaktera. |
| Prikaz obaveštenja za promenu ličnih informacija | Pri uspešnoj promeni ličnih informacija adminu, mušteriji ili prodavcu prikazaće se obaveštenje da je uspešno obavljena operacija. |
| Pregled svih mušterija | Adminu se prikazuju sve mušterije preko paginacije. Jedna strana sadrži 12 korisnika. |
| Pregled svih prodavaca | Adminu se prikazuju svi prodavci preko paginacije. Jedna strana sadrži 12 korisnika. |
| Povlačenje mušterije | Admin povlači mušteriju. Taj korisnik više nije u stanju da se uloguje. |
| Povlačenje prodavca | Admin povlači prodavca. Taj korisnik više nije u stanju da se uloguje. Svi njegovi proizvodi se ne mogu više kupovati. |
| Povlačenje admina | Superadmin povlači admina. Taj korisnik više nije u stanju da se uloguje. |
| Slanje obaveštenja na mejl za povlačenje akaunta | Kada se povuče korisnik, Eprodavnica šalje na njegov mejl obaveštenje da je povučen. |
| Prikaz obaveštenja da je akaunt povučen | Kada se povuče korisnik, prikazuje se obaveštenje da je uspešno povučen korisnik. |
| Pregled svih admina | Superadmin ima pregled svih admina preko paginacije. Jedna strana sadrži 12 korisnika. |
| Pravljenje admina | Superadmin pravi admin nalog. |
| Verifikacija admin vrednosti | Provera mejla, lozinke, imena i prezimena. |
| Slanje potvrde za pravljenje akaunta | Kada se admin nalog napravi, potrebno je da se potvrdi mejl. Eprodavnica šalje mejl potvrdu na uneti mejl. |
| Prikaz greške pravljenja admina | Greška se pojavljuje kada je uneti mejl već u bazi. |
| Prikaz greške za uneta polja | Greška se pojavljuje kada lozinki fali specijalan karakter , broj ili je manje dužine od 5 karaktera, kada su polja prazna, kada mejlu fali majmunsko a. |
| Prikaz obaveštenja da je admin napravljen | Pri uspešnom pravljenju admin naloga, Eprodavnica prikazuje obaveštenje da je uspešno odrađena operacija. |



# Dijagram klasa

## Asocijacija

A ima kolekciju B objekata dok B ima jedan A objekat.

A ima kolekciju B objekata dok B nema ništa od A.

## Realizacija

C implementira metode interfejsa D.

## Generalizacija

E interfejs je ekstenzija F interfejsa.



## Opis

Interfejsi UserDetails i GrantedAuthority su ekstenzije Interfejsa Serializable. Sva tri interfejsa dolaze od org.springframework.security i potrebni su za RUBAC sigurnosni sistem.

Klase Uloga i Privilegije implementiraju GrantedAuthority interfejs. Odnos je takav da uloga ima više privilegija i jedna privilegija može biti povezana sa više uloga. Privilegije se koriste u Controller klasama gde se proverava da li korisnik sa određenom ulogom može da pokrene metodu. Primer uloge bi bio ROLE\_MUSTERIJA dok bi primer privilegije bio LOGOUT.

Klasa Korisnik implementira interfejs UserDetails. Može imati više uloga a jedna uloga može biti povezana sa više korisnika. Uloge koje korisnika može imati su : Mušterija, Admin, Prodavac, Superadmin. Ima tri veze sa Produkt klasom. One su:

* Produkt ima jednog prodavca i prodavac može imati nula ili više produkata.
* Produkt su wishlist-ovali nula ili više mušterija dok mušterija može imati nula ili više wishlist-ovanih produkata
* Produkt su kupili nula ili više mušterija dok mušterija je kupila nula ili više produkata.

Klasa VerificationToken je povezana sa jednim korisnikom dok korisnik nema u sebi taj objekat.

Klasa Recenzija može imati samo jednog korisnika i samo jedan produkt dok produkt i korisnik mogu imati više recenzija.

Klasa Tip ( kategorija ) produkta je povezana sa više produkata dok produkt može imati više tipova ( kategorija ).

Klasa Artikal može da ima samo jedan produkt i jedan račun dok račun i produkt mogu imati više artikala.

Klasa Račun može da ima jednog korisnika i više artikala dok korisnik može imati više računa.

## Korisnički Interfejs