

Katedra za elektronsko poslovanje

**Domaći zadatak iz predmeta Internet tehnologije**

PHP, OOP, MySQL

Student:

Vladimir Jecić 2017/79

Beograd, 2023.

Contents

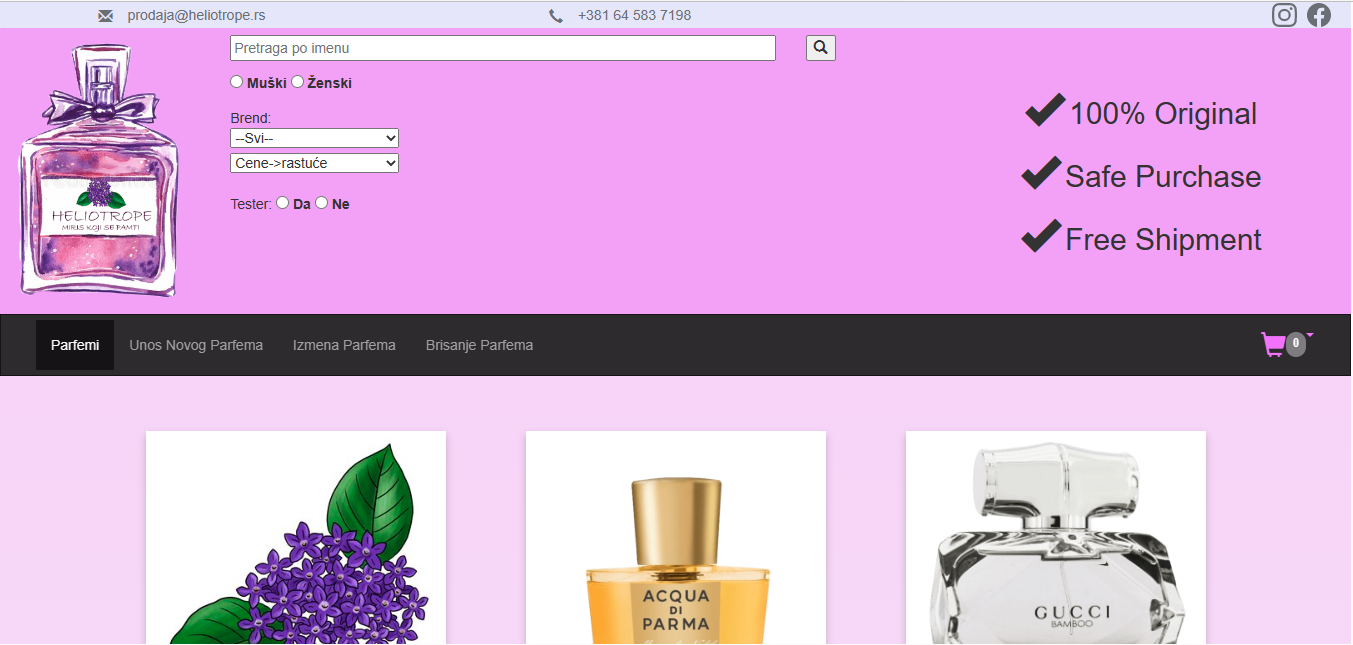
[1. Uvod 2](#_Toc136081370)

[2. Funkcionalnosti dostupne korisniku i korisničko iskustvo 2](#_Toc136081371)

# Uvod

Implementiran sajt služi kao primer web prodavnica parfema imaginarnog preduzeća “Heliotrope”.

Tema ovog zadatka bila je prvenstveno implementiranje našeg prvog rest API servisa koji je implementiran u objektno orjentisanom PHP-u kao, dok je dinamika na frontu realizovana kroz korišćenjem JavaScripta i jQuery-ja.



Slika 1 – Home stranica sajta Heliotrope

# Funkcionalnosti dostupne korisniku

Početna stranica predvidjene sledeće funkcionalnosti dostupne korisniku:

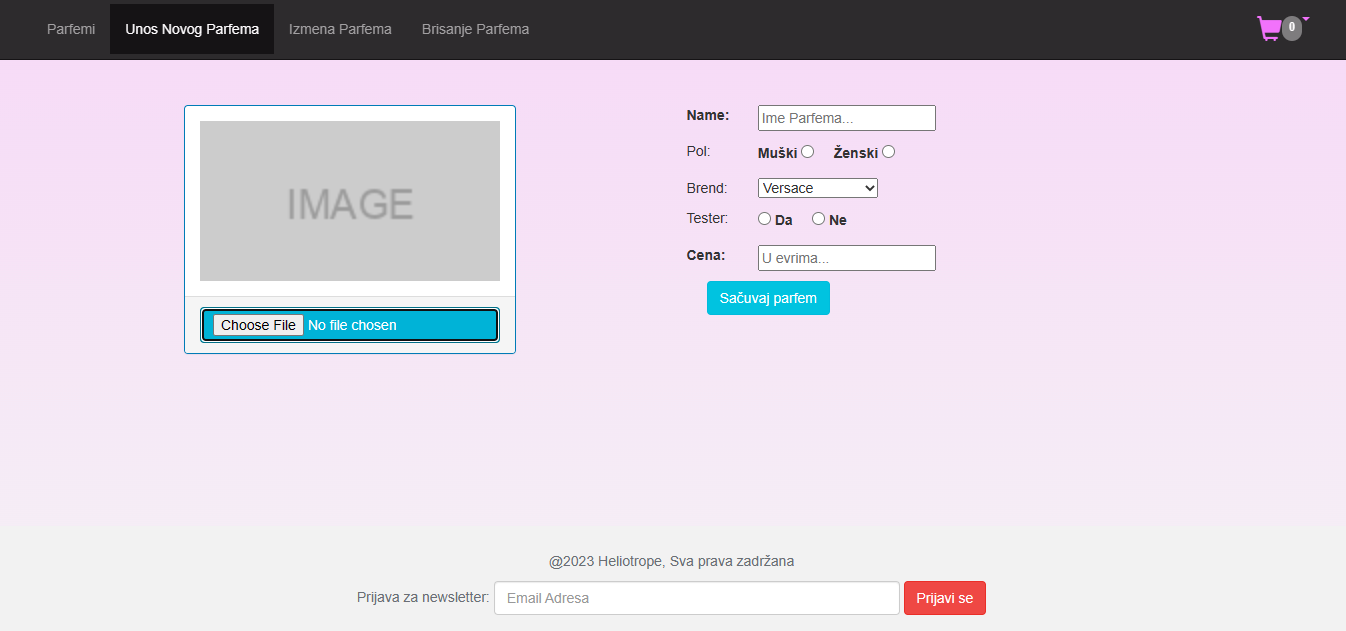
1. Pregled svih postojećih parfem i sužavanje izbora parfema korišćenjem pretrame u zaglavlju sajta. Pritom korisnik može pretraživati parfeme po delu imena(ćirilica i latinica), polu za koji je namenjen M/Ž,brendu i tome da li je parfem tester ili nije tester. Takodje korisnik može sortirati proizvode po imenu ili ceni rastuće/opadajuće. Pretraga se pokreće klikom na dugme koje liči na malu lupu.

Napomena:Pretraga i sortiranje se mogu primeniti i pre operacija pod 4 i 5.

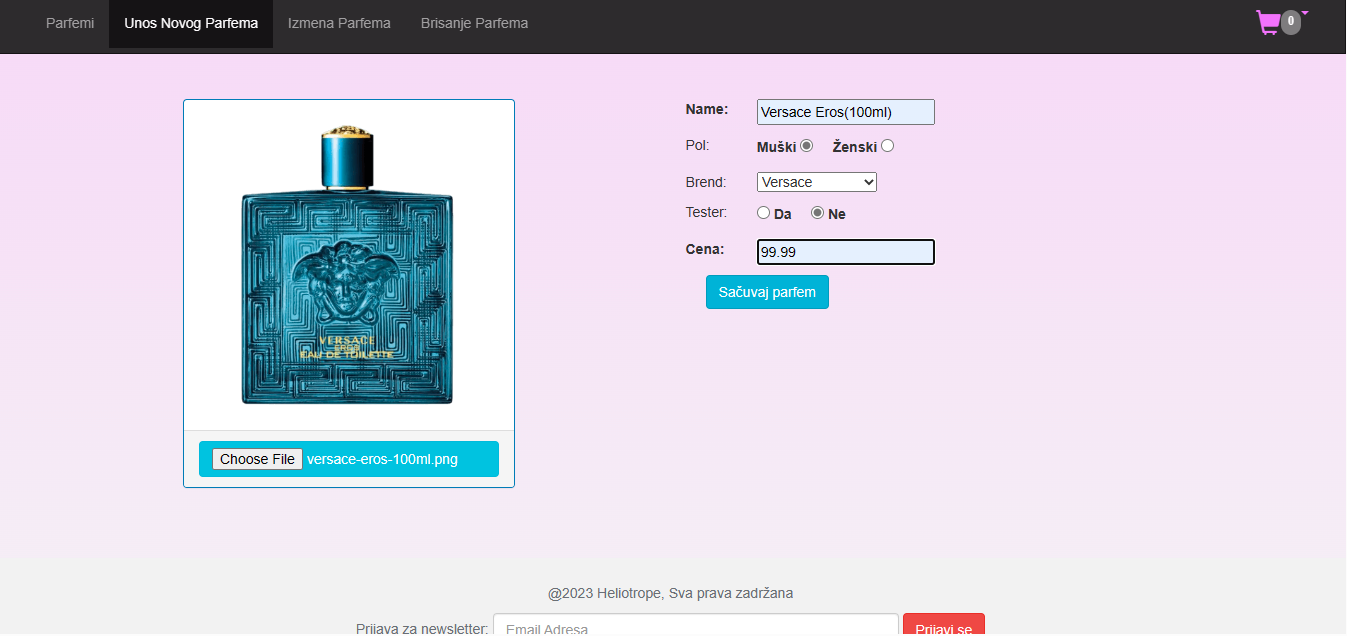
1. Unos novog parfema u bazu podataka(Slika 2 i 3). Korisnik prvo treba da klikne na Unos Novog Parfema u navigacionoj traci, a zatim da priloži sliku i druge neophodne podatke za novi parfem. Nakon što je potvrdio da je uneo korektne podatke on poziva sistem da zapamti novi parfem klikom na dugme sačuvaj parfem. Sistem validira formu i salje podatke ako su validni.

U slučaju da je parfem uspešno sačuvan:Korisniku će biti prikazana poruka o uspešnosti operacije čuvanja parfema zelenim slovima.(Slika 4)

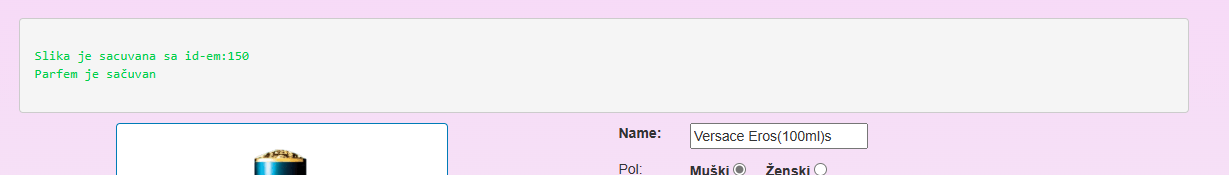
U slučaju da parfem nije uspešno sačuvan: Sistem će korisniku prikazati poruku crvenim slovima zašto parfem nije sačuvan.(Slika 5)



Slika 2 – Dodavanje novog parfema(prazna forma)



Slika 3 - Dodavanje novog parfema(popunjena forma)



Slika 4 - Uspešno čuvanje parfema

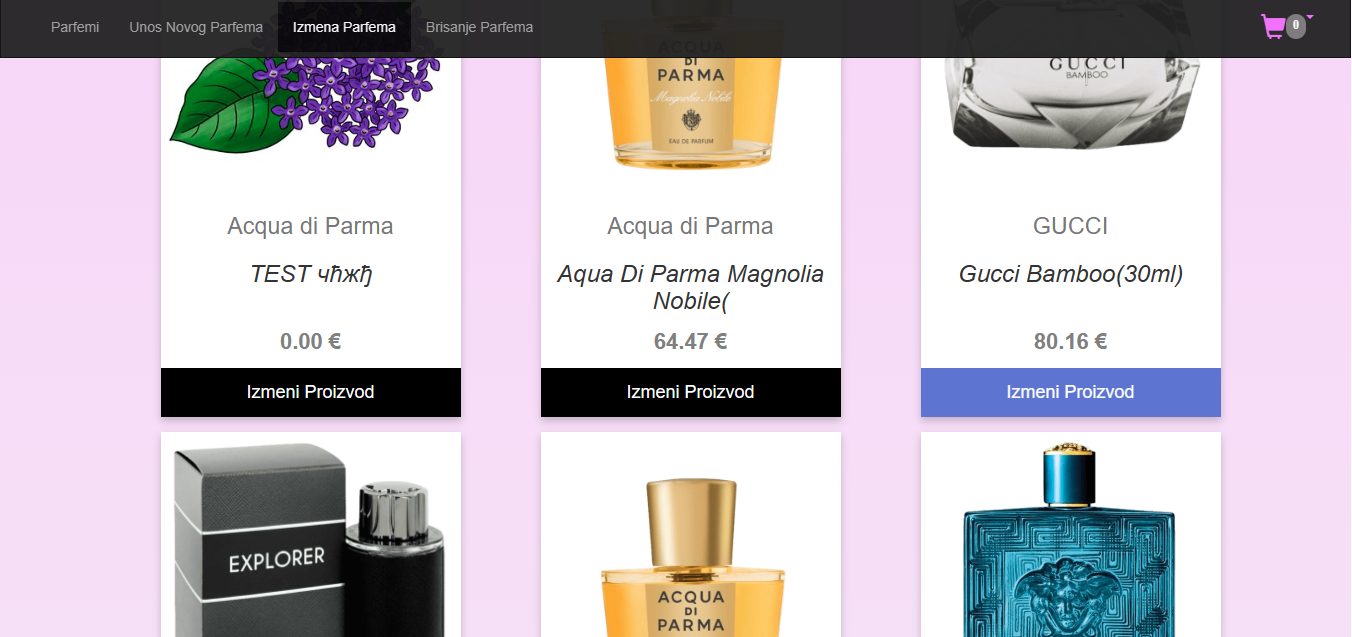


Slika 5 - Neuspešno čuvanje parfema

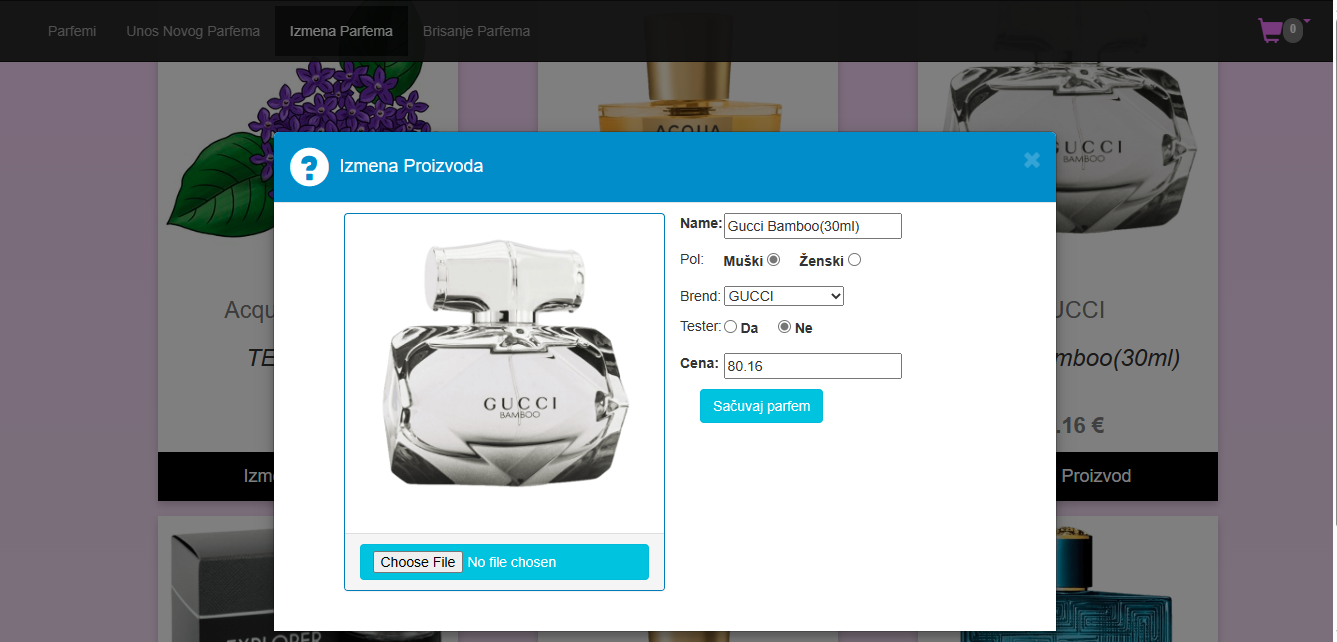
1. Izmena postojećeg parfema. Korisnik prvo treba da klikne na “Izmena Parfema” u navigacionoj traci, da izabere željeni parfem i klikne na “Izmeni Proizvod”(Slika 6). Nakon klika korisniku se otvara nov prozor u kojem može da izmeni neko od svojstava parfema:ime,sliku,da li je tester itd.(Slika 7). Nakon što je korisnik izmenio i proverio podatke on poziva sistem da zapamti novi parfem klikom na dugme sačuvaj parfem. Sistem validira formu i salje podatke ako su validni.

U slučaju da je parfem uspešno sačuvan:Korisniku će biti prikazana poruka o uspešnosti operacije čuvanja parfema zelenim slovima.(Slika 4)

U slučaju da parfem nije uspešno sačuvan: Sistem će korisniku prikazati poruku crvenim slovima zašto parfem nije sačuvan.(Slika 5)



Slika 6 - Izmena postojećeg parfema(svi proizvodi)

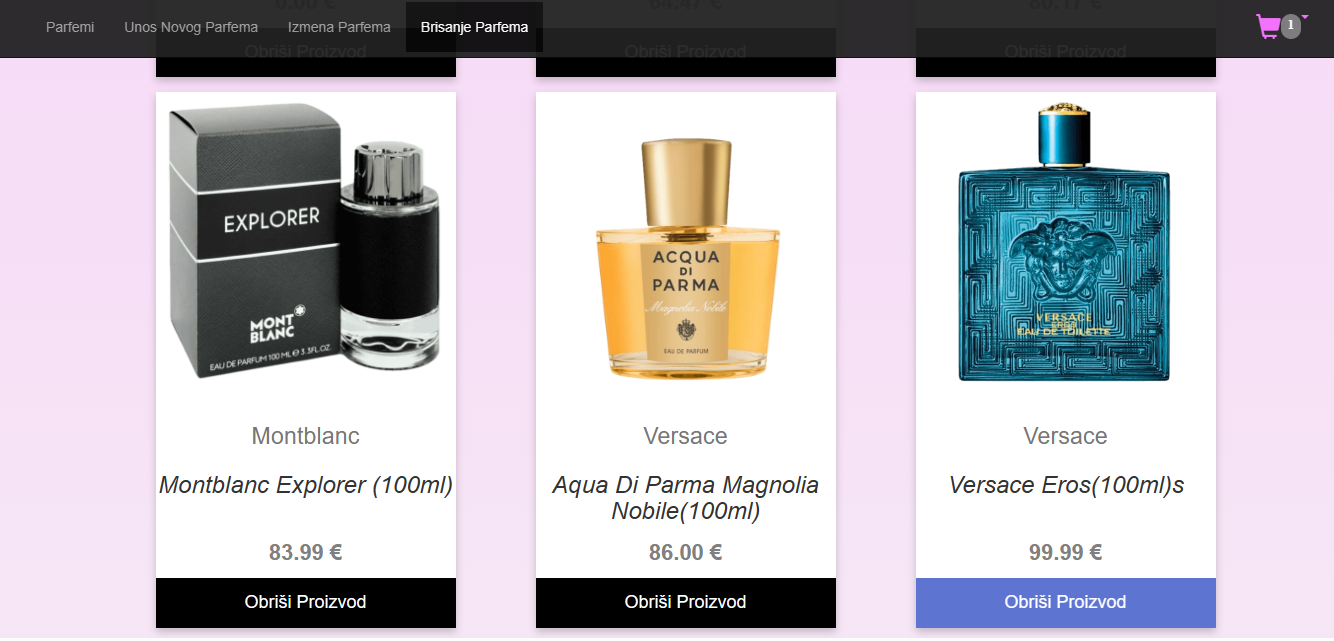


Slika 7 - Izmena postojećeg proizvoda(izmena podataka)

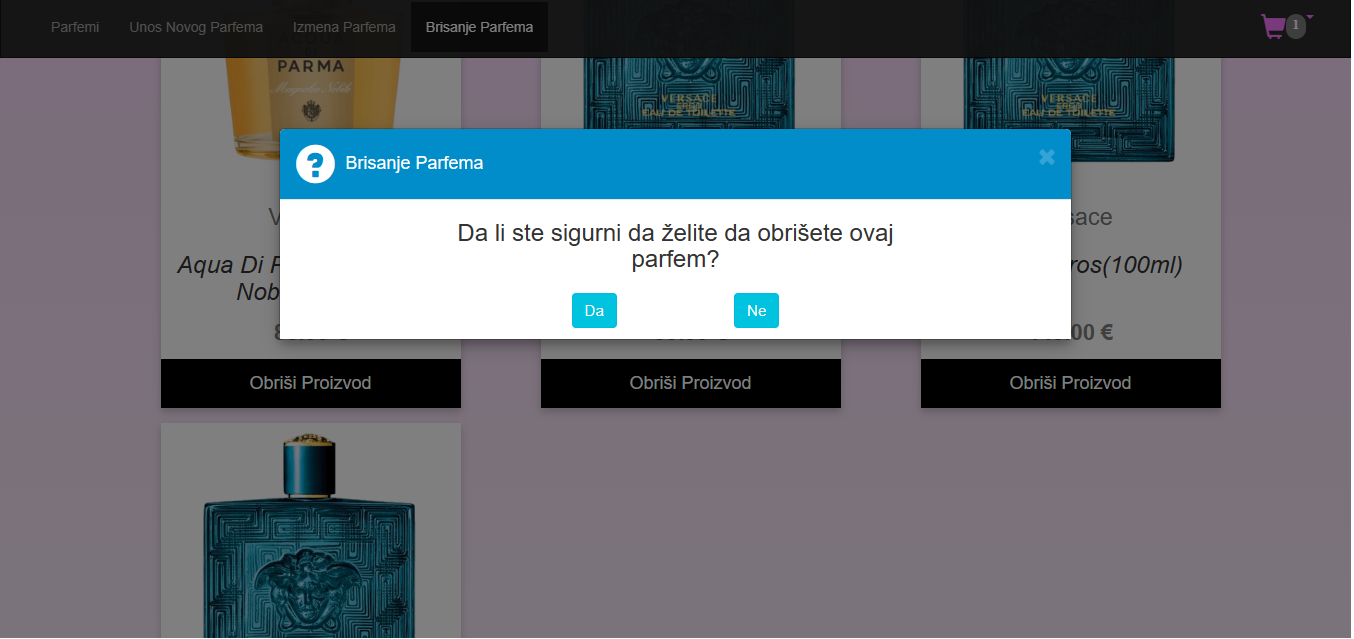
1. Brisanje postojećeg parfema. Korisnik prvo treba da klikne na “Brisanje Parfema” u navigacionoj traci, da izabere željeni parfem i klikne na “Obriši Proizvod”(Slika 8). Nakon klika korisniku se otvara nov prozor u kojem sistem pita korisnika da potvrdi brisanje parfema(Slika 9). Klikom na dugme ‘da’ korisnik poziva sistem da obriše postojeći parfem.

U slučaju da je parfem uspešno sačuvan:Korisniku će biti prikazana poruka o uspešnosti operacije brisanja parfema zelenim slovima.(Slika 10)

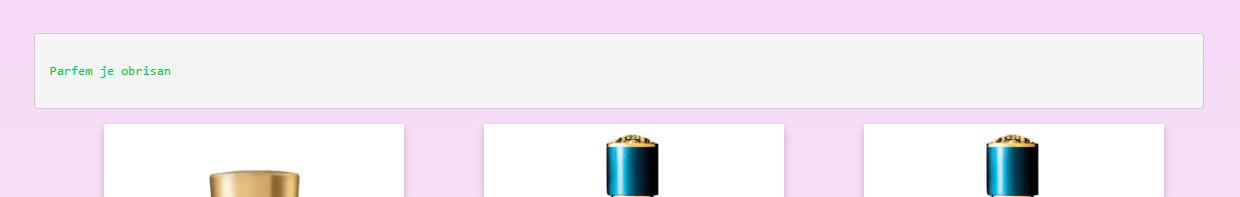
U slučaju da parfem nije uspešno sačuvan: Sistem će korisniku prikazati poruku crvenim slovima zašto parfem nije obrisan.



Slika 8 - Brisanje parfema



Slika 9 - Brisanje Parfema- dijalog da/ne



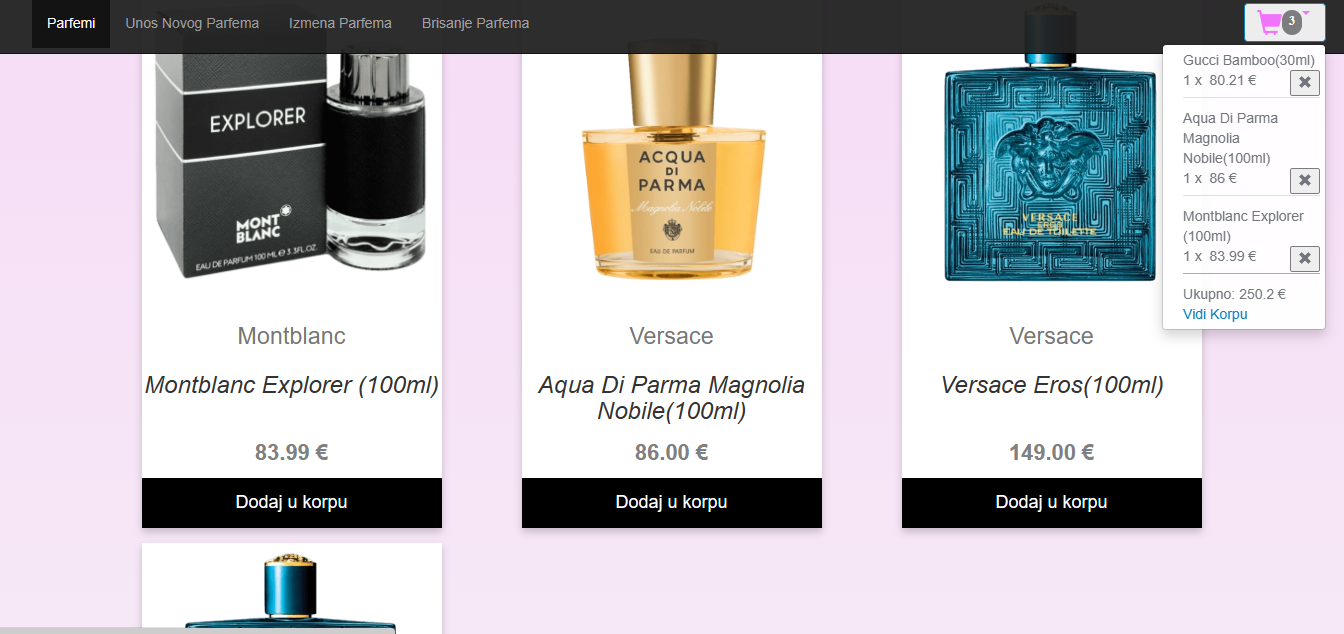
Slika 10 - Uspešno brisanje parfema

1. Dodavanje proizvoda u korpu na glavnoj stranici (Parfemi). Korisnik u svakom trenutku vidi broj postojećih proizvoda u korpi gledanjem na broj pored ikonice kolica,a klikom na samu ikonicu(Slika 11) dobija detaljnije informacije o broju svakog od proizvoda i ukupnom iznosu svih proizvoda zajedno. Ova ikonica je deo zaglavlja i zbog to je vidjiva gde god se korisnik trenutno nalazio (Parfemi,Unos Parfema,Izmena Parfema ili Brisanje Parfema). Klikom na dugme x korisnik smanjuje količinu ciljanog proizvoda za jedan ili ga potpuno uklanja iz korpe.

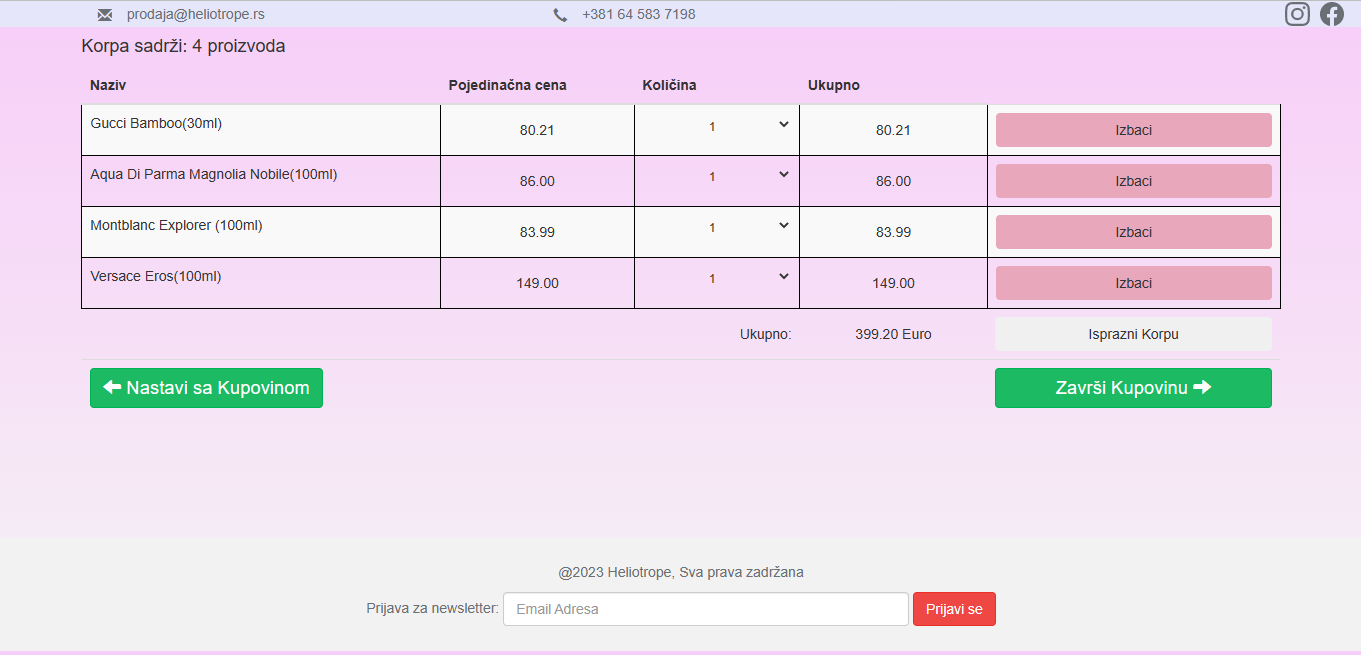
Klikom na dugme Vidi Korpu, učitava se nova strana sa tabelom u kojoj su učitani svi proizvodi,njihovi nazivi,izabrane količine,cene kao i ukupna cena(Slika 12). Izborom količine u drop-down boxu nekog od proizvoda količina se menja,ali takodje i ukupna cena tog proizvoda(cena pojedinačnog x količina), ukupan broj proizvoda u gornjem levom uglu i trenutna ukupna suma svih proizvoda i njihovih količina.

Korisniku je omogućena navigacija na prethodnu stranicu klikom na dugme “Nastavi kupovinu”. Pri čemu nakon klika na to dugme ili nakon osvežavanja bilo te ili glavne stranice stavke u korpi ostaju u poslednjem stanju do kojih je doveo klijent (sesija klijenta).

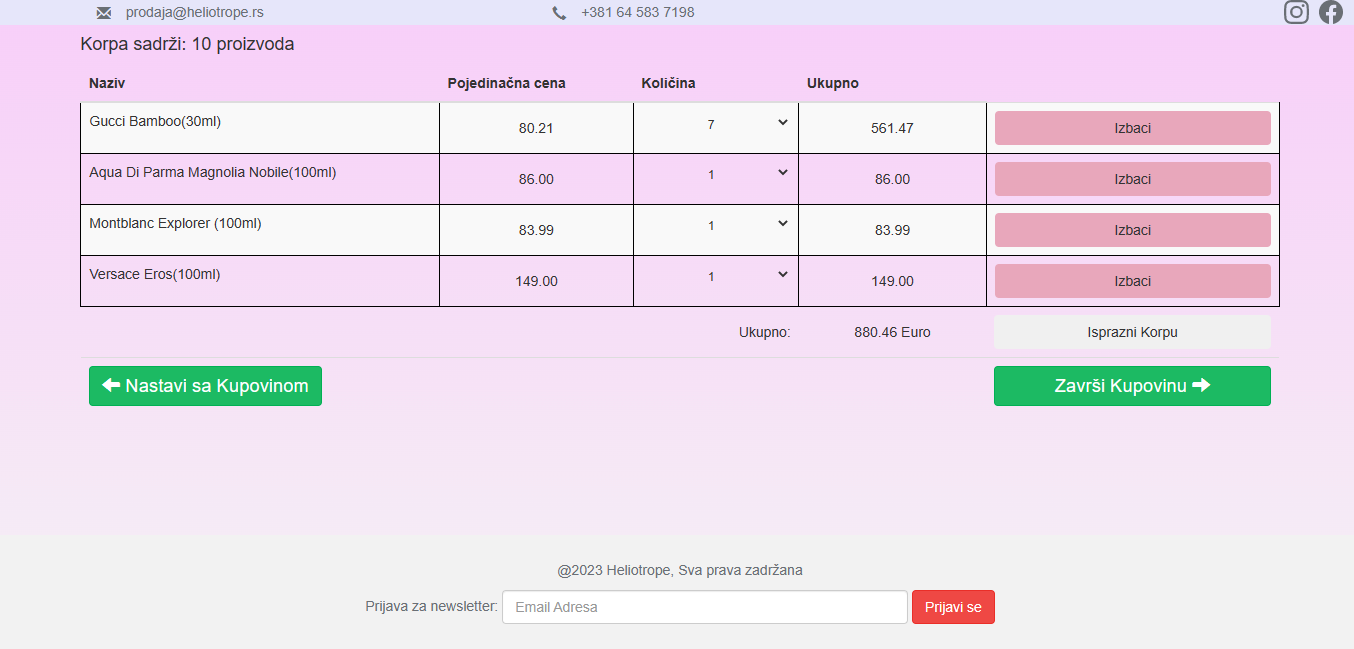
Korpu je takodje moguće i isprazniti klikom na isprazni korpu.(Slika 14)



Slika 11 - Korpa(minimalistički prikaz)



Slika 12 - Izgled stranice "Vidi korpu"



Slika 13 - Izgled stranice "Vidi korpu" nakon promene količine prvog proizvoda

# Objašnjenje implementacije

U ovom poglavlju ću prokomentarisati kod kojim sam ostvario ponašanje opisana u 2.gom poglavlju.

## Prikaz svih parfema, pretraga i sortiranje

**FrontEnd:**

Sam div sa id-em perfume-get u html-u je prazan, a njegov sadržaj se popunjavama dinamički asinhronim funkcijama koje su realizovane korišćenjem ajax-a. Na početku js fajla mojskript.js dodat je listener na dugme sa id-em search-submit. Kada dodje do klik eventa bice pozvana anonimna funkcija koja u poziva funkciju posaljiZahtev sa parametrom “get”.

document.getElementById("search-submit").addEventListener("click", function (e) {

  posaljiZahtev("get");

});

Ista funkcija sa paramterom “get” se poziva prilikom samog učitavanja javascripta zajedno sa html-om što se može videti na kraju fajla.

//////////<POZIVI\_FUNKCIJA>///////////////////

postaviAktivniNavbar();

posaljiZahtev("get");

prikaziBlok();

//////////</POZIVI\_FUNKCIJA>///////////////////

Funkcija posaljiZahtev() kao parametar prima string koji predstavlja tip zahteva koji se šalje ajax funkciom i može biti get,post,put ili delete. U nastavku je deo koji se odnosi na get.

//funkcija posaljiZahtev obrađuje pomoću AJAX - a zahteve koje šaljemo ka serveru

function posaljiZahtev(tipZahteva) {

  let form;

  let selectBrand;

  let id;

  let selectBrandName;

  try {

    document.getElementById(nizBlokova[0]).style.display = "none";

    document.getElementById(nizBlokova[5]).style.display = "none";

    switch (tipZahteva) {

      case "get":

        // SELECT \* FROM perfume JOIN brand ON (perfume.brand\_id=brand.id) JOIN image ON (perfume.image\_id=image.id)

        // WHERE LOWER(perfume.name)  LIKE '%versace%' AND perfume.gender = 'M' AND brand.name = 'Versace' AND perfume.tester = 'no'

        // ORDER BY perfume.name desc;

        let f = getFilters;

        $.ajax({

          type: "GET",

          url: "http://localhost:8080/iteh/domaci/ITEH\_Domaci1-PHP-MySQL-AJAX/api/parfemi/",

          contentType: "application/json; charset=utf-8",

          dataType: "json",

          data: getFilters(),

          success: function (data) {

            if ('message' in data) {

              if ('success' in data) {

                console.log(data['message']);

              } else {

                console.error(data['message']);

              }

            }

            if ('arrPerfume' in data) {

              arrPerfume = data['arrPerfume'];

              fillPerfumeGet();

              fillPerfumePut();

              fillPerfumeDelete();

              scrollProductIntoView();

            } else if (arrPerfume.length == 0) {

              var div\_empty\_response = document.createElement('div');

              div\_empty\_response.classList.add('alert');

              div\_empty\_response.classList.add('alert-warning');

              div\_empty\_response.innerHTML = "<h4><strong>!</strong> Nije nadjen nijedan parfem za unete podatke pretrage.</h4></div>";

              document.getElementById("perfume-get").appendChild(div\_empty\_response);

            } else {

              console.error("vracena je neinicijalizovana data['arrPerfume']");

            }

          }

        });

        break;

U slučaju case get poziva se f-ja getFilters kojom se iz polja u zaglavlju uzimaju vrednosti odredjenih filtera. Za sam ajax zahtev tip GET predstavlja tip zahteva,url je adresa servisa na koji se šalje zahtev. Content Type govori primaocu kakav mu sadržaj šaljemo. Osim application/json, može biti i multipart/form-data ako se salje forma sa fajlovima(npr slikama) ili plain tekst ako je običan tekst.

DataType-om govorimo koji tip podataka očekujemo kao povratnu vrednost.

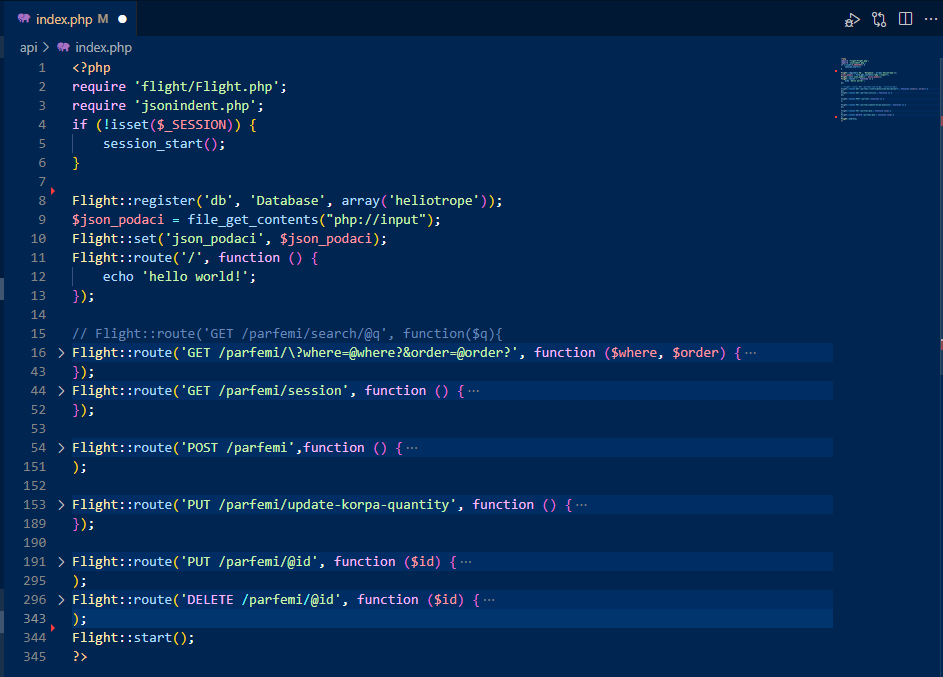
Polje data: sadrži podatke koje šaljemo serveru.

Success: predstavlja zaglavlje nakoj kojeg treba navesti šta se treba desiti ako klijent od servera primi potvrdan odgovor HTTP: 200 OK.

Takodje postoji zaglavje Error: koje je korišćeno za druge tipove zahteva da se greška loguje u konzulu i korisnika obavesti o neuspešnosti operacije(funkcija alert());

Primljeni podaci tj. parfemi se dinamički dodaju u funkciji fillPerfumeGet();

**BackEnd:**

****

Slika 14 - Izgled Flight ruta u fajlu api/index.php

Servis koji predstavlja server osluškuje na portu 8080 (korišćenjem XAMPP-a)

Nakon što zahtev pristigne,pokreće se php skripta index.php koja se interpretira red po red startujući php sesiju ako već nije startovana(na strani servera!) . Linija:

Flight::register('db', 'Database', array('heliotrope'));

Poziva konstruktor klase Database u fajlu Database.php, uspostavlja konekciju sa mySQL bazom ’heliotrope’ i binduje instancu te klase za promenljivu db. Linije:

$json\_podaci = file\_get\_contents("php://input");

Flight::set('json\_podaci', $json\_podaci);

Uzimaju podatke iz zateva ako su tipa JSON. Ako su podaci tipa form. Oni ce biti vidljivi,samo ne kao php input podaci vec kao vrednosti asocijativnog niza superglobalne promenljive $POST.

Nakon ove linije prolazi se kroz sve navedene route i traze sve koje se matchuju sa zahtevom koji se pristigao na server. Zato u glavnom f-je ruta u frameworku Flight vracaju false vrednost jer bi se u suprotnom posle te rute izvršile i sve naredne rute koje su se inicijalno poklopile sa zahtevom klijenta.

U slucaju get route, u samom url-u prosledjeni su delovi upita za where i order by koji ce kasnije biti korišćeni za gradjenje upita. Linija

  $where = rawurldecode(base64\_decode($where));

je korišćena jer se u upitu mogu naći i karakteri srpske ćirilice i latinice koji bi inače bili izgubljeni da su slani bez enkodovanja i dekodovanja.

Parametri se šalju funkciji objekta $db klase Database koja gradi odgovarajuć upit i izvršava ga, posle toga se objekti fetchuju iz resultseta koji je vratila baza i pravi odgovarajuć odgovor za klijenta, ako nastane greška u toku izvršavanja upita ona će biti uhvaće na u try-catch bloku .

U svakom slučaju poruka i podaci će biti pretvorena u asocijativan niz teksta tipa JSON i vraćena korisniku(komanda echo).

## Unos novog parfema

**FrontEnd:**

$("input[name=upload-image]").change(function () {

  imagePreview(this);

});

Postavljen je listener na input polje za upload slike i u slučaju da bi doslo do promene ovog polja pre uploada slike ovde će biti presretna ta akcija i okinuta f-ja imagePreview unutra anonimne f-je.

Unutar imagePreview f-je u zavisnosti da li se radi o unosu ili izmeni parfema pravi se odgovarajuci jquery selektor i postavlja onLoad f-ja na filereader kojom se kaže da se slika treba učitati u to polje i koje veličine treba da bude slika. Nakon toga se pokreće filereader.readAsDataURL i slika je učitana kao fajl i njen preview se prikazuje korisniku u formatu 300x300.

    fileReader.onload = function (event) {

      $(previewId).html(`<img src="${event.target.result}" width="300 height="auto"" alt='${alt}'/>`);

    };

    fileReader.readAsDataURL(fileInput.files[0]);

  }

Što se tiče slanja zahteva za čuvanje novog parfema. Presreće se klik na input submit forme, sprecava se defaulton podesavanje sa prevent default i poziva se f-je posaljiZahtev() sa parametrom ‘post’. Ulazi se u tu f-ju i prvo se validiraju input polja sa f-jom validateForm. Koja osim provere praznih polja takodje ima i jedan regex izraz kojim proverava da je uneta cena parfema u korektnom formatu.

  if (!/^((0|[1-9][0-9]\*)\.[0-9]\*|(0|[1-9][0-9]\*))$/.test($(form + " input[name^=perfume-price]").val())) {

    errMessage += "Cena mora biti broj\n";

  }

Nakon što se završe sve provere ako je poruka errMessage ne-prazna, baca se kao izuzetak koji se kasnije hvata u f-ji posalji zahtev i prikazuje u div-u error sa crvenim slovima

  } catch (err) {

    console.log(err);

    document.getElementById(nizBlokova[0]).innerHTML = err;

    document.getElementById(nizBlokova[0]).style.display = "block";

  }

Gde je nizBlokova niz kreiran na početku fajla

// kreiramo niz blokova, odnosno niz svih id vrednosti div sekcija

const nizBlokova = ["error", "perfume-get", "perfume-post", "perfume-put", "perfume-delete", "success"];

Nakon uspešne validacije kreira se nova instanca objekta FormData() i puni vrednostima koje je korisnik uneo. Zatim se poziva $.ajax f-ja na sličan način kao za get samo što je ovog puta type POST i contentType multipart/form-data. U slučaju da u rezultatu tipa array postoji ključ success, prikazuje se poruka servera zelenim slovima i time daje znak korisniku da je operacija prošla uspešno u suportnom će biti ispisana crvenim slovima i označiti problem,a ako dodje do greške prilikom obrade zahteva tj. ne predvidjene greške,ona će biti prikazana u delu error: kroz alert i pokrenuće debugger (Slika 15)



Slika 15 - ajax success i error funkcija kod POST zahteva

**BackEnd:**

Povezivanje sa bazom se vrši na isti način kao i kod GET zahteva. Samo što ovog puta sa obzirom da je potrebno neke podatke sačuvati u bazu potrebno je prethodno proveriti na strani serva da li svi ti podaci postoje(property\_exist) pre nego se izvade iz objekta $data.

U slučaju da neko od polja nije prisutno kao property objekta tu se dalji tok f-je prekida i korisniku se šalje odgovor da nije prosledio odgovarajuće podatke.

U suprotnom pozivaju se 2 insert upita jedan za drugim. Prvi upit čuva prosledjenu sliku kao zaseban entitet u bazi image, a zatim ako je operacija uspešna uzima ključ novog objekta koji je generisan kroz autoinkrement svojstvo u bazi podataka i dodaje ga kao spoljni ključ novom parfemu. Kao i u GET ruti i svim narednim rutama, svaka greška se hvata u try-catch bloku i odgovarajuća poruka se vraća korisniku.

## 3.3 Izmena Parfema

**FrontEnd:**

Klikom u navbaru na objekat klase perfume-put poziva se f-ja fillPerfumePut() koja je vrlo slična f-ji fillPerfumeGet() i fillPerfumeDelete() samo što dodaje dugme za izmenu proizvoda. Podaci se učitavaju dinamički u html i na svakom parfemu je definisana akcija koja bi trebalo da se desi klikom na submit, a to je onsubmit="return openModal(event,modalPut);”.

U openModal se na osnovu toga koji je modal.id poziva setUp modal za put modal ili delete modal. Koja samo uzima polje sa glavne forme I kopira ih u modal radi lekse preglednosti. ModalPut je identičan izgledu forme za unos novog parfema samo što sam elementi u htmlu imaju malo drugačija imena.

Implementacija ajaxa I validacija za PUT je skoro indentična POST case-u pa ga neću dalje analizirati. Sem što se nakon uspešne PUT metode,a ovo je slučaj i za DELETE poziva GET ajax f-ja unutar anonimne asinhrone f-je koja je uspešna kako bi se osvežio prikaz odmah nakon promene ili brisanja parfema.

**BackEnd:**

Slično kao i za POST metodu,prvo se pamti slika sa insert, samo što se nakon toga menja proizvod sa update() naredbom.

## 3.4 Brisanje parfema

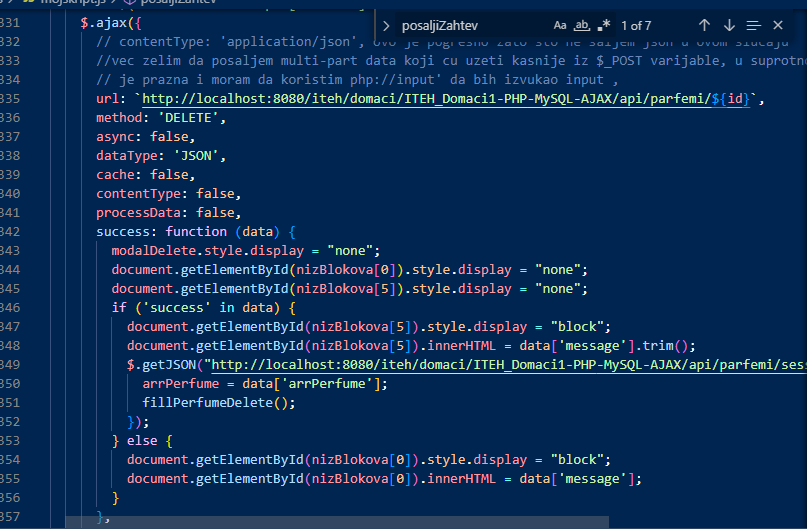
**FrontEnd:**

Klikom u navbaru na objekat klase perfume-delete poziva se f-ja fillPerfumeDelete() koja je vrlo slična f-ji fillPerfumeGet() i fillPerfumePut() samo što dodaje dugme za brisanje proizvoda. Podaci se učitavaju dinamički u html i na svakom parfemu je definisana akcija koja bi trebalo da se desi klikom na submit, a to je onsubmit="return openModal(event,modalDelete);”.

## U openModal se na osnovu toga koji je modal.id poziva setup modal za put modal ili delete modal.

setupDeleteModal jedino uzima id elementa koji je kliknut da bi bio obrisan i postavlja ga za id modala.

Taj id će kasnije biti korišćen u posaljizahtev() sa ‘delete’ parametrom da obriše taj specifičan proizvod gde se id salje u okviru samog URL-a. Kao i kod ajax PUT zahteva nakon brisanja će biti pozvan još jedan getJSON kako bi se dobili najnovniji elementi iz baze nakon brisanje I osvežili prikaz na stranici.



Slika 16 - ajax slanje DELETE zahteva i getJSON u success f-ji

**BackEnd:**

Zaboravio sam ranije da spomenem,ali u bazi podataka dodavanje reda.

 $this->dblink->autocommit(false);

je odgovornost commitovanja i rollbackovanja transakcija prebačena na REST API umesto baze podataka. Zbog toga je commit radjen tek nakon što je cela operacija ili više njih u slučaju ubacivanja slike npr kod POST i PUT naredbe. Dok je rollback; radjen u svakom drugom slučaju, pa i u catch bloku.

Nakon što je parfem obrisan iz baze,kako bi parfemi ostali u pravom redosledu kada korisnik sledeći put odluči da izvrši neku od operacija u okviru iste sesije,potrebno je obrisati parfem i iz super globalne SESSION varijable i reindeksirati niz, kako pogrešni brojevi kasnije ne bi dovodili do grešaka u operacijama sa id-evima.

            if ($db->delete(table: 'perfume', where: "id= $id")) {

                $response["success"] = "yes";

                $message .= "\nParfem je obrisan\n";

                //izmena u $\_SESSION varijabli

                $L = count($\_SESSION['arrPerfume']);

                for($i=0;$i<$L;$i++){

                    if($\_SESSION['arrPerfume'][$i]->id == $id){

                        unset($\_SESSION['arrPerfume'][$i]);

                        $\_SESSION['arrPerfume'] =array\_values($\_SESSION['arrPerfume']);//'reindex' array

                        break;

                    }

                }

                $db->commit();

            } else {

                $message .= "\nDošlo je do greške pri brisanju parfema\n";

                $message .= "\nUpit:" . $db->getLastQuery();

                $db->rollback();

            }

## 3.5 Korpa na glavnoj stranici i stranica na linku ?vidi\_korpu

Svi proizvodi u korpi se čuvaju u superglobalnoj promenljivoj $SESSION[‘cart’]. Dok se funkcionalnosti vezane za korpu nalaze u php skripti koja je uključena na početku php fajla pod imenom obrada.php.

Skripta osluckuje $POST zahteve sa odgovarajućim key-value parovima i kada je takav POST zahtev napravljen taj if se okida,npr

//POST Dodaj u Korpu(povecava kolicinu izabranog proizvoda za 1)

if (isset($\_POST['submit']) && $\_POST['submit'

] == "Dodaj u korpu") {

//POST Ukloni iz Korpe(smanjuje kolicinu izbaranog proizvoda za 1)

if(isset($\_POST['submit']) && $\_POST['submit'] == "Oduzmi iz Korpe"){

Obe ove funkcije rade sa SESSION[‘cart’] variajblom iteriraju kao kroz njen niz i dodaju/uklanjaju količinu ili čitav proizvod iz korpe ako je prethodno imao količinu 1 u ovom fajlu se takodje osluskuju i POST zahtevi za Izbacivanje proizvoda I svih njegovih instanci iz korpe, Isprazni korpu koja unsetuje ceo SESSION[‘cart’] i jedan koji osluskuje GET i vraća stranicu korpa.php

// GET ?vidi\_korpu(ucitava stranicu korpa.php)

if (isset($\_GET['vidi\_korpu'])) {

    include 'korpa.php';

    exit();

}