

# Dokumentacija projekta

## Online butik

### 1. Uvod

### 2. Prikaz korišćenih tehnologija

2.1. MongoDB

2.2. Angular

2.3. Spring

2.4. Maven

### 3. Detalji implementacije

#### 3.1. Backend

3.1.1. Struktura aplikacije

3.1.2. Spring aplikacija

3.1.3. Šema baze podataka

3.1.4. Bezbednost

3.1.5. Zavisnosti

#### 3.2. Frontend

3.2.1. Struktura aplikacije

3.2.2. Korisnički interfejs

3.2.3. Bezbednost

### 4. Zaključak

# 1. Uvod

„Online butik” predstavlja prototip veb aplikacije za online kupovinu. Projekat se sastoji iz dve aplikacije:

- Klijentska aplikacija kreirana korišćenjem Angular okvira.
- Serverska aplikacija kreirana korišćenjem Spring okvira.

Projekat je implementiran u vidu REST veb servisa i ima za cilj da omogućí:

- kreiranje naloga,
- kreiranje naloga preko Google Account-a,
- autentifikaciju korisnika,
- upravljanje nalogom,
- pregled proizvoda,
- filtriranje proizvoda po određenim parametrima,
- detaljan pregled proizvoda,
- dodavanje i brisanje proizvoda iz korisničke korpe,
- kupovinu proizvoda iz korisničke korpe,
- pregled i upravljanje omiljenim proizvodima,
- dodavanje komentara i ocenjivanje proizvoda,
- kao i pregled i upravljanje istorijom porudžbina.

Uputstvo za uspešno pokretanje projekta na mašini za razvoj:

1. Instalacija korišćenih tehnologija.
2. Kreiranje baze podataka, šeme i ubacivanje proizvoda.
3. Pokretanje serverske aplikacije na portu 8080
4. Pokretanje klijentske aplikacije.
5. Pokretanje veb pretraživača i poseta adrese localhost na podrazumevanom portu 4200 ili eksplicitno navedenom portu.

## 2. Prikaz korišćenih tehnologija

### 2.1. MongoDB

MongoDB je vodeća NoSQL baza podataka. U skladu sa rastućim interesovanjem za alternativne sisteme za upravljanje bazama podataka, koji se razlikuju od tradicionalnih relacionih baza podataka, pojavio se koncept takozvanih NoSQL baza podataka, koje ne koriste SQL za povezivanje, nerelacione su, distribuirane, otvorenog koda i horizontalno skalabilne. Napisana je u C++ jeziku i otvorenog je koda, izdata pod kombinacijom GNU Affero General Public License i Apache License. MongoDB čuva podatke kao JSON dokumente sa dinamičkim šemama. JSON (JavaScript Object Notation) je otvoreni standard zasnovan na tekstu, osmišljen za razmenu podataka koji su pogodni za čitanje ljudima.

MongoDB čini integraciju podataka u mnogim aplikacijama jednostavnijom i bržom. MongoDB je stvoren i podržan od strane kompanije 10gen.

## 2.2. Angular

Angular je okvir otvorenog koda koji održava Google i zajednica pojedinačnih programera i korporacija za rešavanje mnogih izazova prilikom kreiranja jednostraničnih aplikacija. Okvir je napisan u TypeScript-u.

Angular je jedan od najkorišćenijih okvira za razvoj klijentskih veb aplikacija i često se kombinuje sa Spring okvirom za razvoj.

## 2.3. Spring

Spring je okvir za razvoj veb aplikacija koji predstavlja kontejner za injekciju zavisnosti za Java platformu. Glavne karakteristike okvira se mogu koristiti u bilo kojoj Java aplikaciji, ali postoje i dodaci koji se namenski koriste za razvoj Java veb aplikacija. Jedne od najvećih prednosti ovog okvira jeste činjenica da je otvorenog koda i da ga razvija zajednica koja ima veliki broj programera.

Glavni moduli Spring okvira za razvoj:

- Spring Core Container
- AOP
- Authentication and Authorization
- Convention over configuration
- Testing
- Transaction management
- Inversion of control container

## 2.4. Maven

Maven je alat koji se koristi za build i rukovođenje bilo kojim Java projektom. Glavne funkcionalnosti:

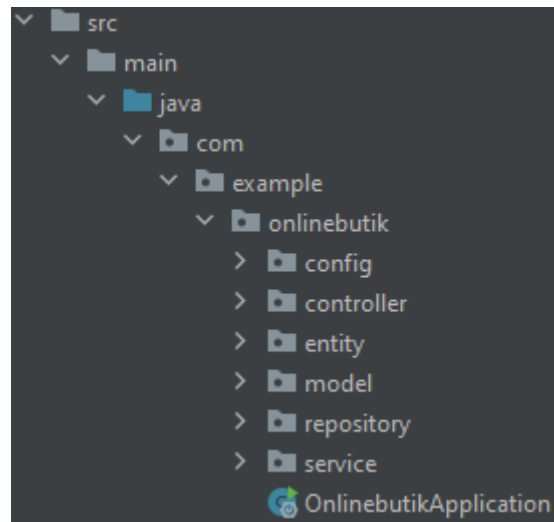
- Olakšava proces build-a aplikacije.
- Pruža kvalitetne informacije o projektu.
- Pruža smernice za razvoj uz primenu najboljih praksi.
- Dozvoljava lako dodavanje novih biblioteka.

## 3. Detalji implementacije

### 3.1. Backend

#### 3.1.1. Struktura aplikacije

Na slici je prikazana struktura Java paketa. Aplikacija je funkcionalno podeljena u module – configuration, entity, model, controller, repository i service.



### 3.1.2. Spring aplikacija

Aplikacija sadrži 8 entita - Users, Orders, OrdersItem Items, Size, Cart, CartItem i Comment. Prikaz entita Items.

```
@Data
@Document(collection = "items")
@NoArgsConstructor
@AllArgsConstructor
public class Items {

    @Id
    private String id;
    @Field("name")
    @NotBlank
    private String name;
    @Field("image")
    @NotBlank
    private String image;
    @Field("category")
    @NotBlank
    private String category;
    @Field("type")
    @NotBlank
    private String type;
    @Field("amount")
    @NotBlank
    private List<Size> amount;
    @Field("stars")
    private Double stars;
    @Field("price")
    @NotBlank
    private Double price;
    @Field("description")
    @NotBlank
    private String description;
    @Field("country")
    @NotBlank
    private String country;
    @Field("slug")
    private String slug;
}
```

Aplikacija sadrži 5 repozitorijuma. Prikaz repozitorijuma IItemRepository.

```
public interface IItemRepository extends MongoRepository<Items, String> {
    Items findBySlug(String slug);
    @Query("{ 'category': ?0 , 'type': ?1 }")
    List<Items> findAllByCategoryAndType(String category, String type);
    @Query("{ 'category': ?0 , 'type': ?1, 'price': { '$gt': ?2, '$lt': ?3 } }")
    List<Items> getAllByPriceBetween(String category, String type, Double gtPrice, Double ltPrice);
}
```

Aplikacija sadrži veliki broj servisa. Na sledećoj prikazan je ItemService kao jedan od servisa koji služi za poslovnu logiku koja se tiče entiteta.

```
public interface IItemService {

    ItemsModel findBySlug(String slug);
    List<ItemsModel> findAllByCategoryAndType(String category, String type);
    List<ItemsModel> findAll();
    ItemsModel findById(String id);
    List<ItemsModel> findAllByCart(CartModel cart);
    ItemsModel updateItemAmount(String itemId, String size, Integer amount);
    List<ItemsModel> getSimilarItems(String id);
    List<ItemsModel> getFavouriteItems(Users user);
    ItemsModel updateItemStars(Items item);
}
```

```
@Service
public class ItemService implements IItemService {

    private static final Double RANGESIMILAR = 200.00;

    @Autowired
    private IItemRepository itemRepository;
    @Autowired
    private AutoMapperService autoMapperService;

    @Override
    public ItemsModel findById(String id) {
        if(itemRepository.findById(id).isPresent())
            return autoMapperService.map(itemRepository.findById(id).get(), ItemsModel.class);
        return new ItemsModel();
    }

    @Override
    public List<ItemsModel> findAllByCart(CartModel cart) {
        List<ItemsModel> list = new ArrayList<>();
        cart.getItems().forEach(cartItemModel -> {
            if(itemRepository.findById(cartItemModel.getItemId()).isPresent())
                list.add(autoMapperService.map(
                    itemRepository.findById(cartItemModel.getItemId()).get(), ItemsModel.class));
        });
        return list.stream().distinct().collect(Collectors.toList());
    }

    @Override
    public ItemsModel updateItemAmount(String itemId, String size, Integer amount) {
        Items i = autoMapperService.map(this.findById(itemId), Items.class);
        i.getAmount().forEach(s->{
            if (s.getName().equals(size)){
                s.setAmount(s.getAmount()-amount);
            }
        });
        return autoMapperService.map(itemRepository.save(i), ItemsModel.class);
    }
}
```

```

@Override
public List<ItemsModel> getSimilarItems(String id) {
    ItemsModel item = this.findById(id);
    Double gt = item.getPrice() - ItemService.RANGESIMILAR;
    Double lt = item.getPrice() + ItemService.RANGESIMILAR;
    List<Items> items = itemRepository.getAllByPriceBetween(item.getCategory(), item.getType(), gt, lt);
    List<ItemsModel> models = new ArrayList<>();
    items.forEach(i -> {
        models.add(autoMapperService.map(i, ItemsModel.class));
    });
    return models;
}

@Override
public List<ItemsModel> getFavouriteItems(Users user) {
    List<ItemsModel> items = new ArrayList<>();
    user.getFavouriteItems().forEach(itemId -> {
        ItemsModel i = this.findById(itemId);
        items.add(i);
    });
    return items;
}

@Override
public ItemsModel updateItemStars(Items item) {
    return autoMapperService.map(itemRepository.save(item), ItemsModel.class);
}

@Override
public ItemsModel findBySlug(String slug) {
    return autoMapperService.map(itemRepository.findBySlug(slug), ItemsModel.class);
}

```

```

@Override
public List<ItemsModel> findAllByCategoryAndType(String category, String type) {
    List<Items> items = itemRepository.findAllByCategoryAndType(category, type);
    List<ItemsModel> models = new ArrayList<>();
    items.forEach(item -> {
        models.add(autoMapperService.map(item, ItemsModel.class));
    });
    return models;
}

public List<ItemsModel> findAll() {
    List<Items> items = itemRepository.findAll();
    List<ItemsModel> models = new ArrayList<>();
    items.forEach(item -> {
        models.add(autoMapperService.map(item, ItemsModel.class));
    });
    return models;
}
}

```

Aplikacija sadrži 5 rest kontrolera. Na sledećoj slici je prikazan ItemController.

```
@RestController
@RequestMapping(value="/items")
public class ItemController {

    @Autowired
    private ItemService itemService;

    @GetMapping(value="/findallbycategoryandtype")
    @CrossOrigin("*")
    public List<ItemsModel> findAllByCategoryAndType(String category, String type){
        return itemService.findAllByCategoryAndType( category, type );
    }

    @GetMapping(value="/findbyslug/{slug}")
    @CrossOrigin("*")
    public ItemsModel findBySlug( @PathVariable String slug ){ return itemService.findBySlug(slug ); }

    @GetMapping(value="/findbyid/{id}")
    @CrossOrigin("*")
    public ItemsModel findById(@PathVariable String id) { return itemService.findById(id); }

    @PostMapping(value="/updateitemstars")
    @CrossOrigin("*")
    public ItemsModel updateItemStars(@RequestBody Items item) { return itemService.updateItemStars(item); }

    @PostMapping(value="/getfavouriteitems")
    @CrossOrigin("*")
    public List<ItemsModel> getFavouriteItems(@RequestBody Users user) { return itemService.getFavouriteItems(user); }

    @PostMapping(value="/updateitemamount")
    @CrossOrigin("*")
    public ItemsModel updateItemAmount(@RequestBody String requestData){
        JSONObject data = new JSONObject(requestData);
        String itemId = (String) data.get("itemId");
        String size = (String) data.get("size");
        Integer amount = (Integer) data.get("amount");
        return itemService.updateItemAmount( itemId, size, amount);
    }

    @GetMapping(value="/getsimilaritems/{id}")
    @CrossOrigin("*")
    public List<ItemsModel> getSimilarItems(@PathVariable String id) { return itemService.getSimilarItems( id ); }
}
```

### 3.1.3. Šema baze podataka

Šema baze podataka sadrži 5 tabela:

- cart
- comments
- items
- orders
- users

### 3.1.4. Bezbednost

Aplikacija koristi Json Web Token za autentifikaciju i autorizaciju korisnika preko servisa Auth0. Implementacija nekih od komponenti koje su vezane za bezbednost aplikacije je prikazana na sledećim slikama.

```
@EnableWebSecurity
class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Value("http://localhost:8080/online_boutik")
    private String audience;

    @Value("${spring.security.oauth2.resourceserver.jwt.issuer-uri}")
    private String issuer;

    @Bean
    JwtDecoder jwtDecoder() {
        NimbusJwtDecoder jwtDecoder = (NimbusJwtDecoder)
            JwtDecoders.fromOidcIssuerLocation(issuer);

        OAuth2TokenValidator<Jwt> audienceValidator = new AudienceValidator(audience);
        OAuth2TokenValidator<Jwt> withIssuer = JwtValidators.createDefaultWithIssuer(issuer);
        OAuth2TokenValidator<Jwt> withAudience = new DelegatingOAuth2TokenValidator<>(withIssuer, audienceValidator);

        jwtDecoder.setJwtValidator(withAudience);

        return jwtDecoder;
    }

    @Override
    public void configure(WebSecurity webSecurity) {
        webSecurity.ignoring().antMatchers(
            ...antPatterns: "/users/findbyemail/",
            "/users/findall",
            "/cart/*",
            "/order/findallbyidUser/",
            "/order/otherboughtitems/",
            "/order/delete/",
            "/items/*",
            "/comment/findallbyiditem",
            "/comment/countstars"
        );
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .mvcMatchers(...patterns: "/users/findbyemail/", "/users/findall", "/cart/*", "/order/findallbyidUser/",
                "/order/otherboughtitems/", "/order/delete/", "/items/*", "/comment/findallbyiditem",
                "/comment/countstars").permitAll()
            .mvcMatchers(...patterns: "/cart/updateorderedcart", "/comment/insert", "/order/insert", "/order/update",
                "/order/isordereditembyemailandslug", "/users/insert", "/users/update").authenticated()
            .and().cors()
            .and().oauth2ResourceServer().jwt();
    }
}
```

```
class AudienceValidator implements OAuth2TokenValidator<Jwt> {
    private final String audience;

    AudienceValidator(String audience) { this.audience = audience; }

    public OAuth2TokenValidatorResult validate(Jwt jwt) {
        OAuth2Error error =
            new OAuth2Error("invalid_token", "The required audience is missing", uri: null);

        if (jwt.getAudience().contains(audience)) {
            return OAuth2TokenValidatorResult.success();
        }
        return OAuth2TokenValidatorResult.failure(error);
    }
}
```



### 3.1.5. Zavisnosti

Aplikacija ima zavisnosti koje se nalaze u pom.xml fajlu koji se nalazi u korenom direktorijumu aplikacije.

```
<dependencies>
  <dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
    <version>20210307</version>
  </dependency>
  <dependency>
    <groupId>org.modelmapper</groupId>
    <artifactId>modelmapper</artifactId>
    <version>2.4.2</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

```

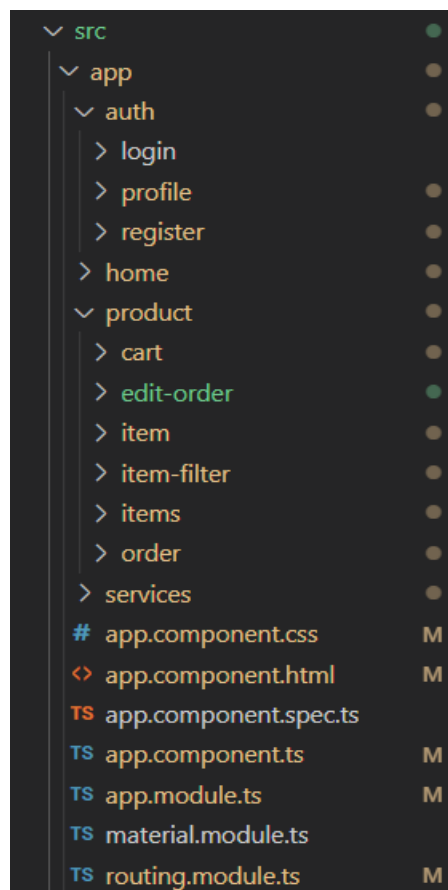
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>RELEASE</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-oauth2-client</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
</dependencies>

```

## 3.2. Frontend

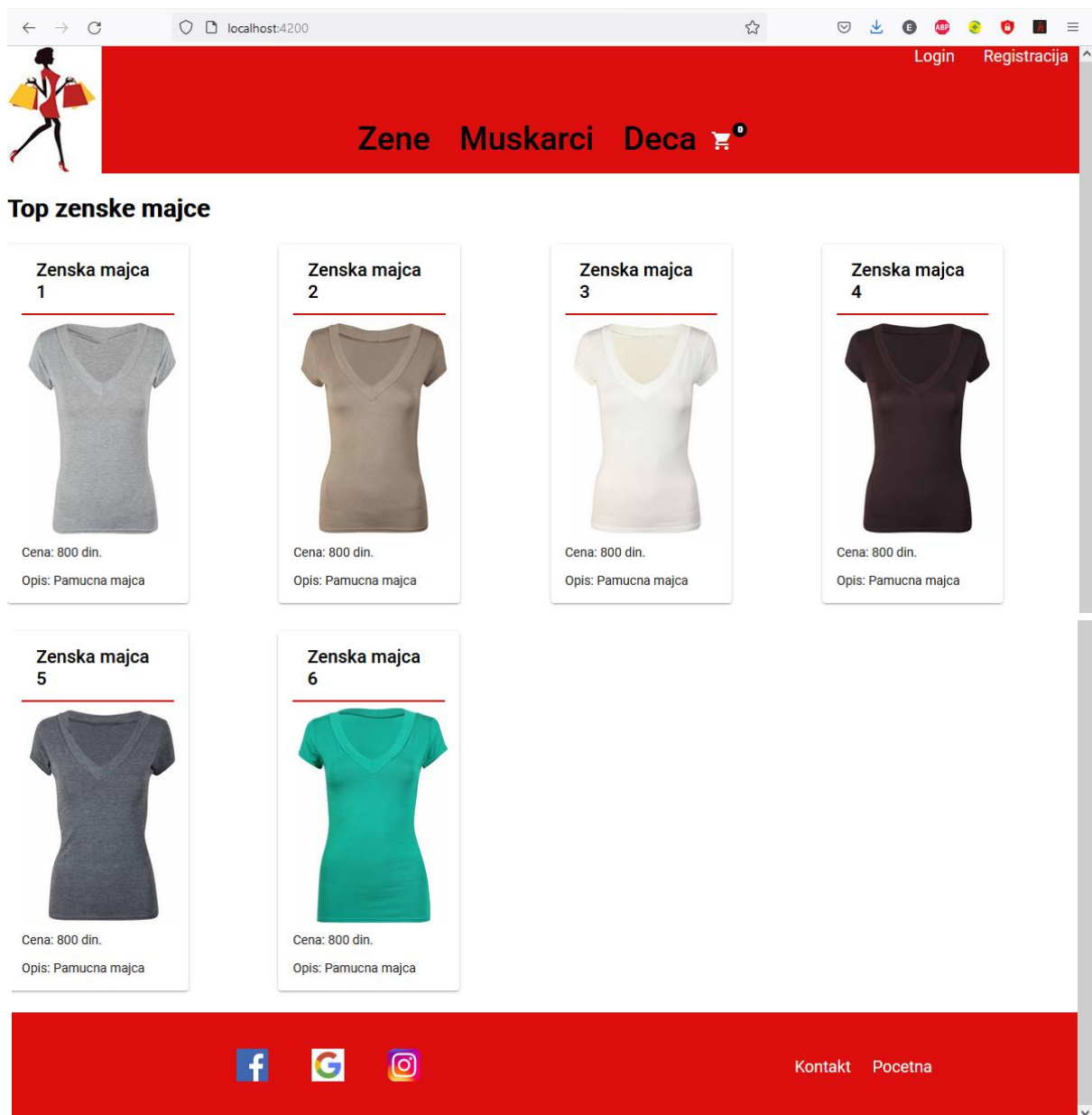
### 3.2.1. Struktura aplikacije

Aplikacija je logički podeljena na sledeći način - auth, home, product i services. Na slici je prikazana opšta struktura aplikacije.




### 3.2.2. Korisnički interfejs


Početna strana anonimnog korisnika



## Prijavljivanje – podrazumevana strana za prijavljivanje Auth0-a.

← → ↻ <https://dev-s2l3n3j7.us.auth0.com/u/login?state=hKFo2S8neTI0dGp3ZWwtZzZVEVXpmb1Zl> ☆






### Welcome

Log in to dev-s2l3n3j7 to continue to OnlineButik.

Email address  
ivana.ivanovic.18@singimail.rs


Password  
●●●●●●●● 


[Forgot password?](#)

[Continue](#)

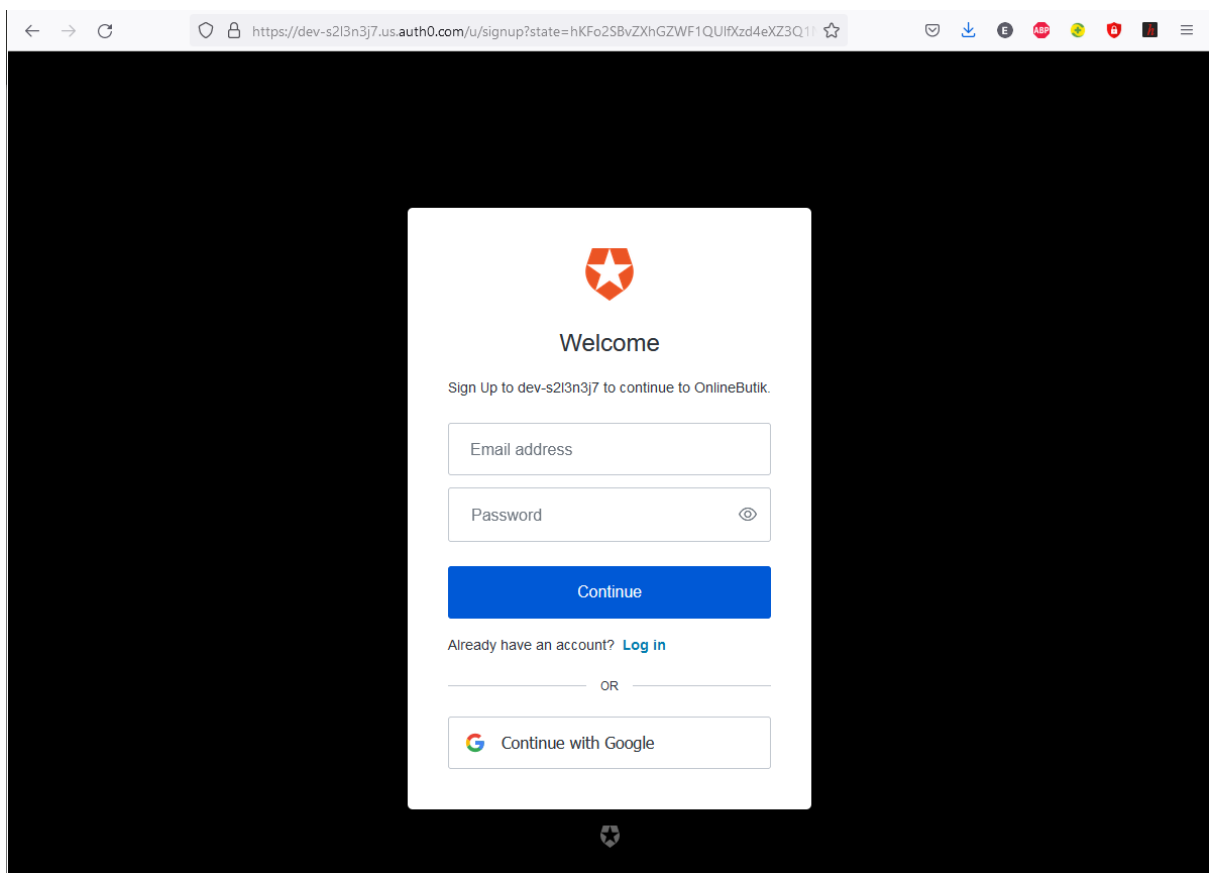
Don't have an account? [sign up](#)

OR

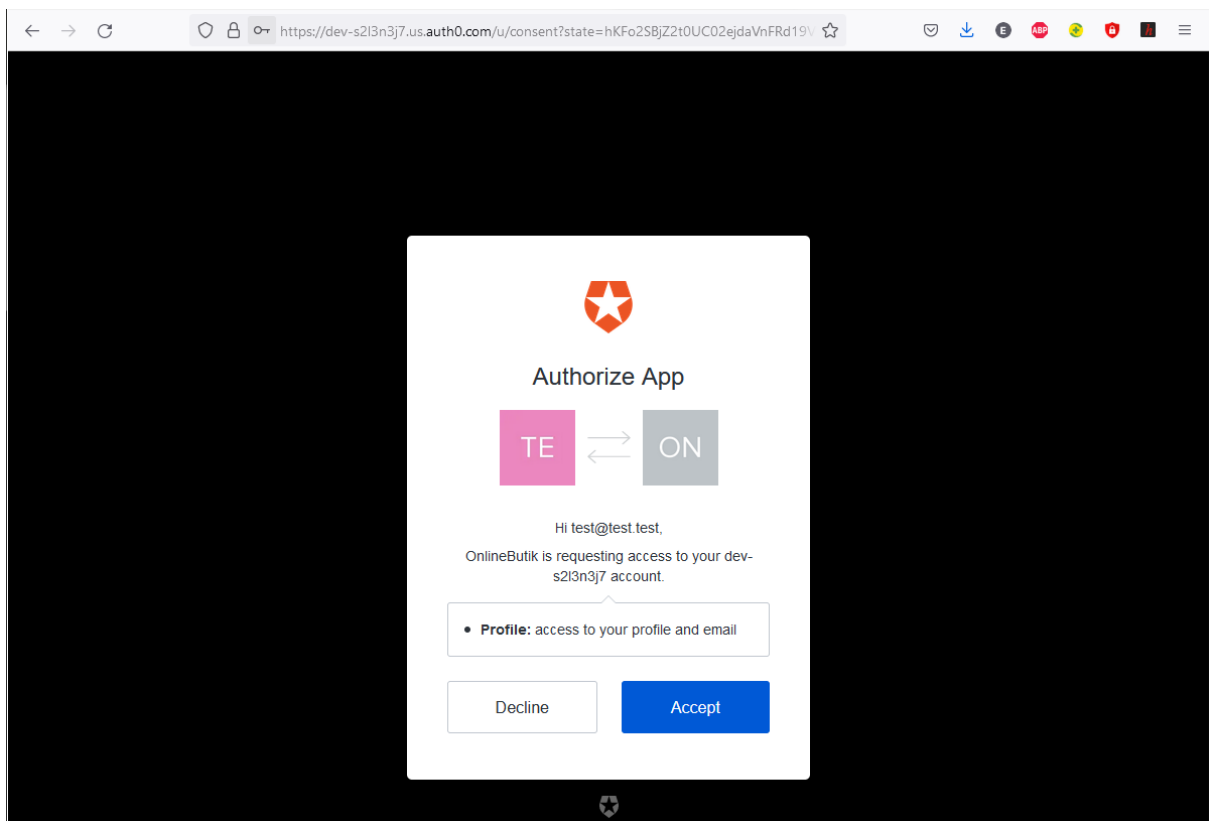
 [Continue with Google](#)




Registracija se sastoji iz dva dela. Prvi preko Auth0-a, drugi na aplikaciji.




A screenshot of a web browser showing the Auth0 'Welcome' page. The page has a dark background with a white central card. At the top of the card is the Auth0 logo (a red shield with a white star). Below the logo is the heading 'Welcome' and a subheading 'Sign Up to dev-s2l3n3j7 to continue to OnlineButik.' There are two input fields: 'Email address' and 'Password' (with a toggle eye icon). Below these is a blue 'Continue' button. Under the button, it says 'Already have an account? [Log in](#)'. Below that is a horizontal line with 'OR' in the center. At the bottom is a 'Continue with Google' button with the Google logo. The browser's address bar shows the URL: `https://dev-s2l3n3j7.us.auth0.com/u/signup?state=hKFo2SBvZXhGZWF1QUlFkzd4eXZ3Q11`.



A screenshot of a web browser showing the Auth0 'Authorize App' screen. The page has a dark background with a white central card. At the top of the card is the Auth0 logo. Below the logo is the heading 'Authorize App'. In the center, there is a diagram showing a pink box labeled 'TE' and a grey box labeled 'ON' connected by two horizontal arrows pointing in opposite directions. Below this diagram, it says 'Hi test@test.test, OnlineButik is requesting access to your dev-s2l3n3j7 account.' There is a list of permissions: '• Profile: access to your profile and email'. At the bottom are two buttons: 'Decline' and 'Accept'. The browser's address bar shows the URL: `https://dev-s2l3n3j7.us.auth0.com/u/consent?state=hKFo2SBjZ2t0UC02ejdaVnFRd19\`.



ProfilOdjavi se

ZeneMuskarciDeca 

Ime \*

Ime je obavezno.

Prezime \*

Prezime je obavezno.

Email \*

test@test.test

Nick name \*

test

Url slike \*

<https://s.gravatar.com/avatar/dd46a756faad4727f>

Ulica i broj \*


Ulica i broj su obavezni

Grad \*

Grad je obavezan

Broj telefona \*




Broj telefona je obavezan

Datum rođenja \* 

☐ Prihvatam uslove korišćenja.


Posalji

Nazad



KontaktPocetna

## Profil korisnika



ProfilOdjavi se

ZeneMuskarciDeca

### Podaci o korisniku

Ime  
Ivana

Prezime  
Jevtic

Ulica i broj  
aaaaa

Grad  
a


Datum rođenja  
2021-09-30

Email  
ivana.ivanovic.18@singimail.rs


Broj telefona  
1111

Izmeni profil

### Omiljeni proizvodi



**Zenska majca 1**  
Zemlja porekla: Srbija  
Opis: Pamucna majca  
Obrisi






**Zenska majca 4**  
Zemlja porekla: Srbija  
Opis: Pamucna majca  
Obrisi

### Pregled narudzbina

Pretrazi

616ef6704b1a0419d91ec21b	Ukupno: 800din.	Status: canceled
616ef6dfec40590d16e41326	Ukupno: 5600din.	Status: going
61713a7af205ad67f50f9e8a	Ukupno: 1600din.	Status: going
61752e788a62b218a33075e9	Ukupno: 2400din.	Status: completed
6175714ab2582256b246b3cf	Ukupno: 3200din.	Status: going

Items per page: 51 - 5 of 5



KontaktPocetna

Omogućeno je menjanje stanja i brisanje porudžbina

## Pregled narudzbina


Pretraži


616ef6704b1a0419d91ec21b

616ef6dfec40590d16e41326

61713a7af205ad67f50f9e8a

**Vreme:**  
**Adresa:** aaaaa  
**Grad:** a

**Zenska majca 1**  
Pamucna majca

**Zenska majca 6**  
Pamucna majca

61752e788a62b218a33075e9

6175714ab2582256b246b3cf

**Izmena narudzbine**

**Sifra:** 61713a7af205ad67f50f9e8a  
**Vreme:** 2021-10-21

**Ocena:**  
★ ★ ★ ★ ★

**Adresa:** aaaaa  
**Grad:** a  
**Status:** Završena




Izmeni

Nazad

Ukupno: 3200din.

Status: going


Items per page: 5 1 – 5 of 5

   Kontakt Pocetna


Na profilnoj stranici moguće je i menjati određene lične podatke

## Podaci o korisniku


Unesite ime \*

 Ivana


Unesite prezime \*

 Jevtic



Unesite ulicu i broj \*

 aaaaa


Unesite grad \*

 a


Unesite datum rođenja \*

 9/30/2021 

Email

 ivana.ivanovic.18@singimail.rs

Unesite broj telefona. \*

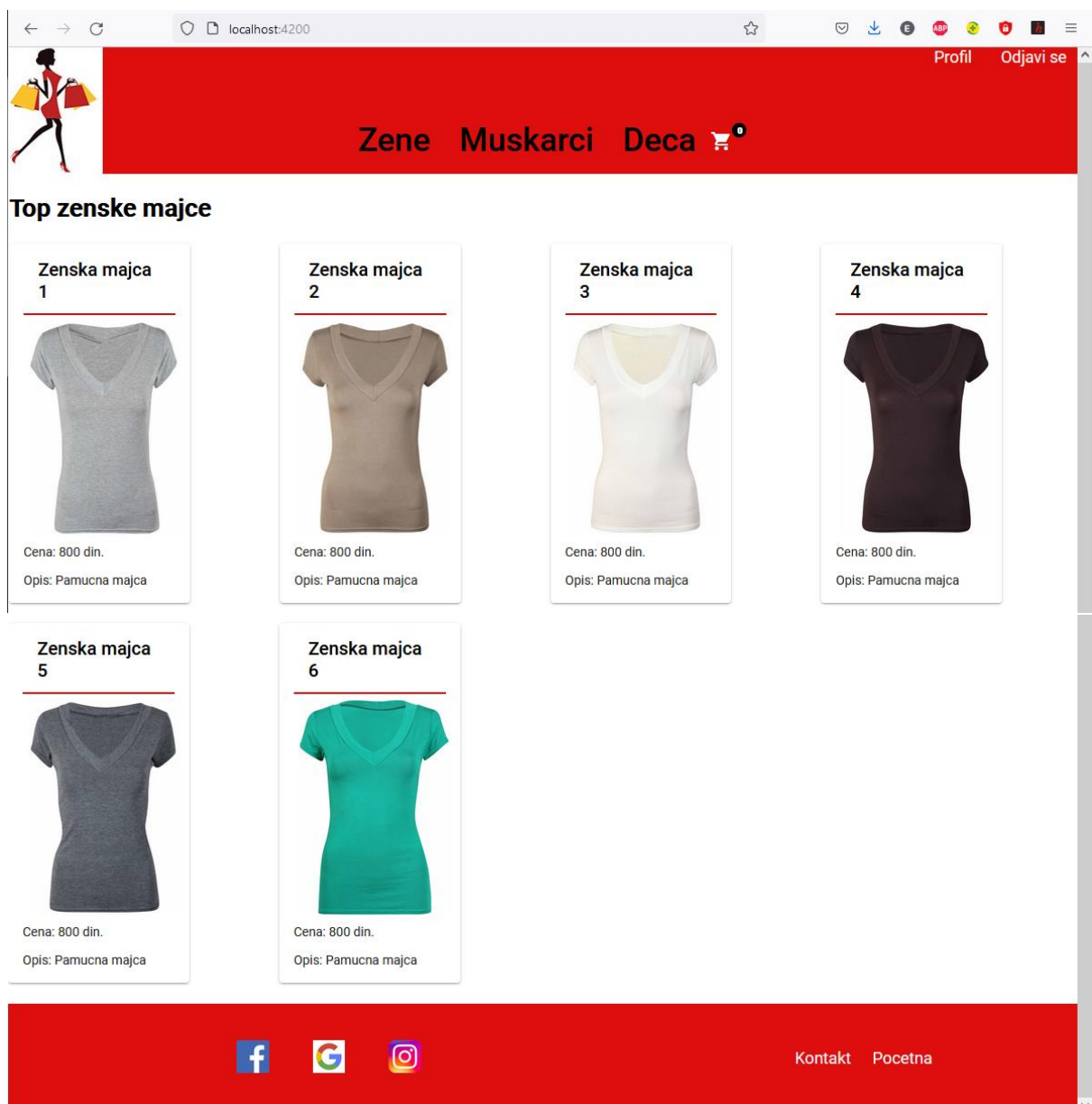
 1111

Sacuvaj


Izmeni profil



## Početna strana autentifikovanog korisnika




## Detalji o proizvodu



ProfilOdjavi se

ZeneMuskarciDeca



### Zenska majca 4

Cena: 800 din.

Opis: Pamucna majca

Zemlja porekla: Srbija


Ocena: 5.00

SMLXL

Dodaj u korpu

### Slicni proizvodi


#### Zenska majca 1



Cena: 800 din.

Opis: Pamucna majca


#### Zenska majca 2



Cena: 800 din.

Opis: Pamucna majca


#### Zenska majca 3



Cena: 800 din.

Opis: Pamucna majca

#### Zenska majca 5




Cena: 800 din.

Opis: Pamucna majca

### Drugi su kupili i ovo:

#### Zenska majca 5



Cena: 800 din.

Opis: Pamucna majca

### Komentari

ivana.ivanovic.18@singimail.rs

★5

Proba




2021-10-19T16:44:31.120+00:00

### Dodaj komentar:

Ocenite: ★ ★ ★ ☆ ☆

Komentar:

Posalji



KontaktPocetna

## Izgled korisničke korpe anonimnog korisnika

The screenshot shows a web browser at localhost:4200/cart. The header is red with a shopping bag icon on the left and 'Login' and 'Registracija' links on the right. Below the header, there are three category links: 'Zene', 'Muskarci', and 'Deca', followed by a shopping cart icon with a '0' badge. The main content area displays a table of items in the cart:

Slika	Naziv	Velicina	Kolicina	Cena	Obrisi
	Zenska majca 2	M	2	800	<button>Obrisi</button>
	Zenska majca 3	M	2	800	<button>Obrisi</button>
	Zenska majca 6	M	2	800	<button>Obrisi</button>

Below the table, the total amount is shown: 'Za uplatu: 4800'. A message 'Ulogujte se da bi ste narucili' is displayed. At the bottom, there is a red footer with social media icons (Facebook, Google+, Instagram) and links for 'Kontakt' and 'Pocetna'.

## Izgled korisničke korpe ulogovanog korisnika

The screenshot shows the same web browser at localhost:4200/cart, but the user is logged in. The header is red with a shopping bag icon on the left and 'Profil' and 'Odjavi se' links on the right. The category links and cart icon remain the same. The table of items in the cart is identical to the anonymous user's view:


Slika	Naziv	Velicina	Kolicina	Cena	Obrisi
	Zenska majca 2	M	2	800	<button>Obrisi</button>
	Zenska majca 3	M	2	800	<button>Obrisi</button>
	Zenska majca 6	M	2	800	<button>Obrisi</button>

Below the table, the total amount is shown: 'Za uplatu: 4800'. Two buttons are visible: 'Naruci' and 'Isprazni korpu'. At the bottom, there is a red footer with social media icons (Facebook, Google+, Instagram) and links for 'Kontakt' and 'Pocetna'.

Klikom na dugme Naruči otvara se dijalog koji omogućava kreiranje narudžbina

localhost:4200/cart

Profil Odjavi se

Zene Muskarci Deca 

Slika Naziv Velicina Kolicina Cena Obrisi

**1** Unesite ime i prezime

Ime \*  
Ivana

Prezime \*  
Jevtic

Sledeci


**2** Unesite adresu za dostavu

**3** Potvrdite


f G Instagram Kontakt Pocetna

localhost:4200/cart

Profil Odjavi se

Zene Muskarci Deca 

Slika Naziv Velicina Kolicina Cena Obrisi

 **1** Unesite ime i prezime

**2** Unesite adresu za dostavu

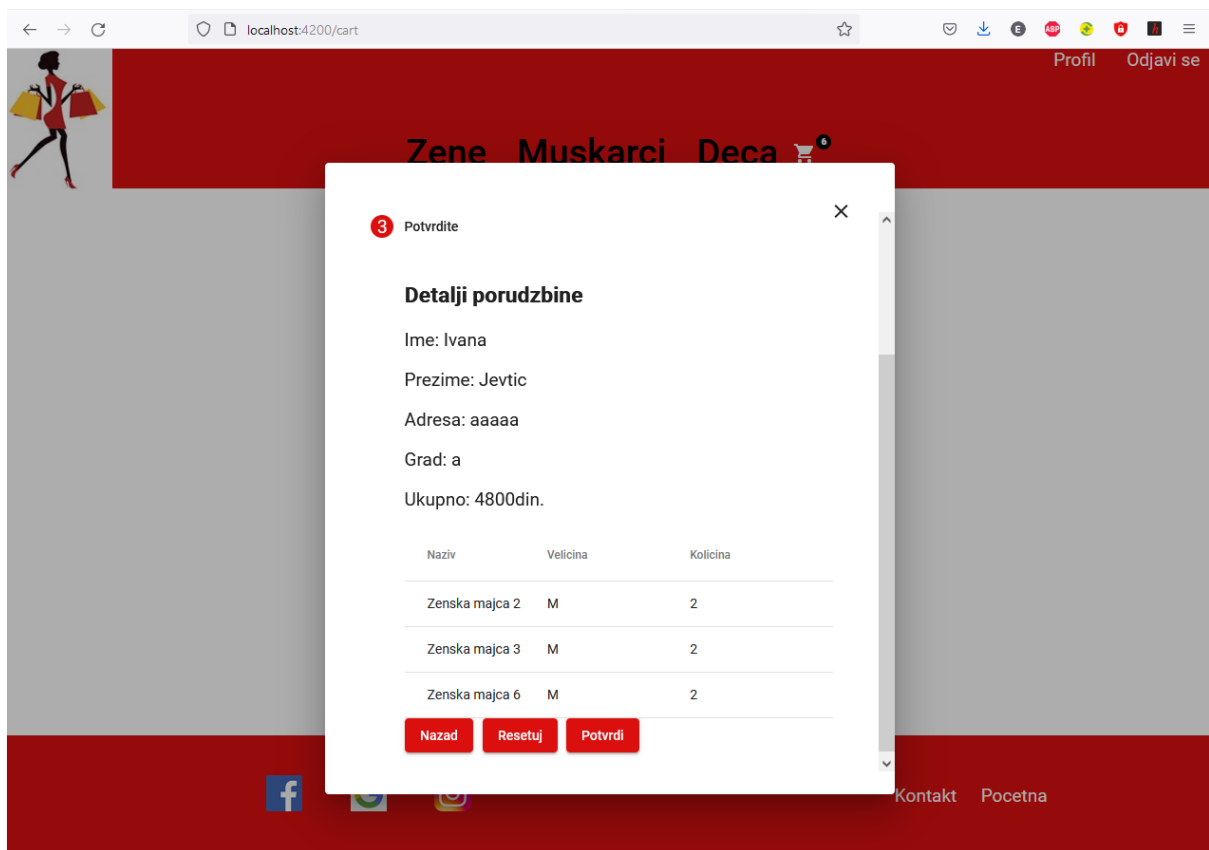
Ulica i broj \*  
aaaaa

Grad \*  
a

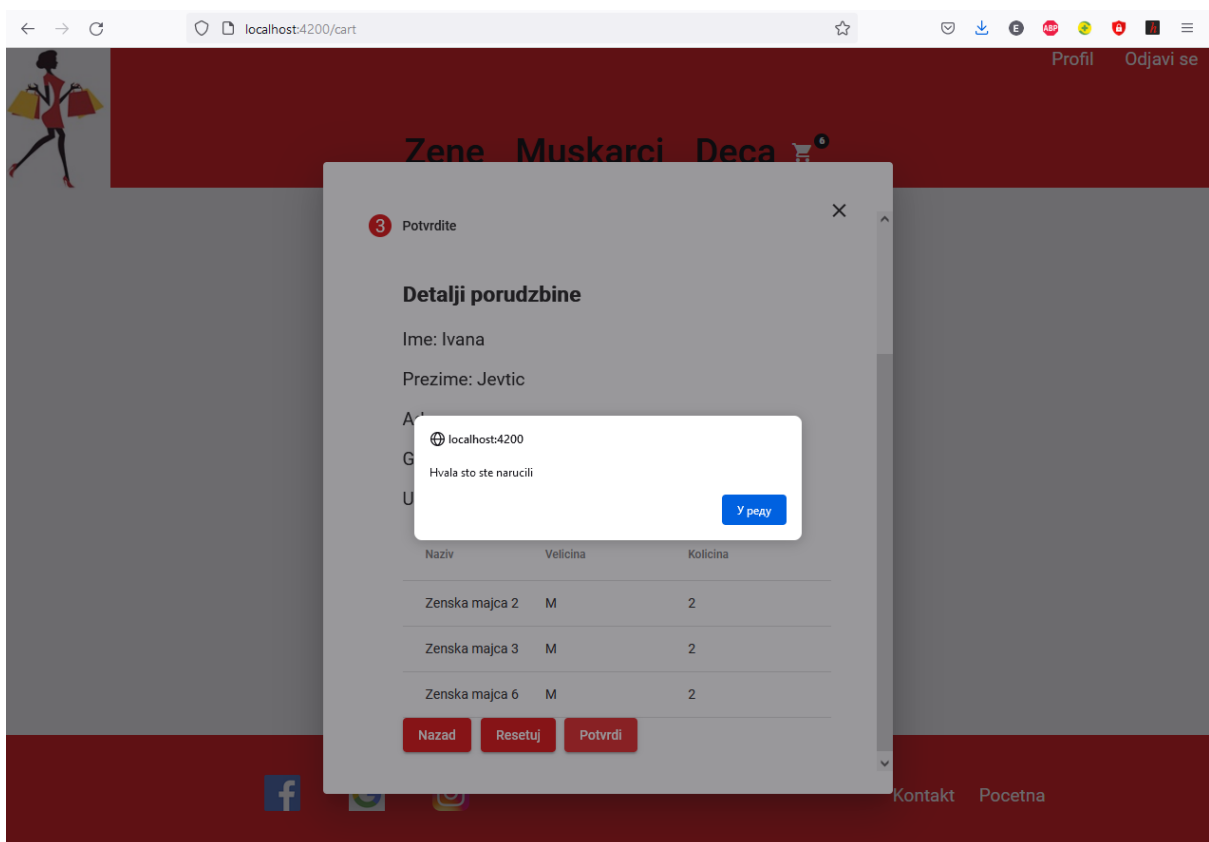
Nazad Sledeci

**3** Potvrdite

f G Instagram Kontakt Pocetna



Klikom na dugme Potvrdi porudžbina se šalje



### 3.2.3. Bezbednost

Na slikama su data podešavanja za Auth0 servis na klijentskoj strain.

```
imports: [  
  BrowserModule,  
  BrowserAnimationsModule,  
  MaterialModule,  
  RouterModule,  
  HttpClientModule,  
  FlexLayoutModule,  
  FormsModule,  
  ReactiveFormsModule,  
  AuthModule.forRoot({  
    domain: "dev-s2l3n3j7.us.auth0.com",  
    clientId: "dbGVr0gDT5cvgwzBvWxaZgnA9qL3JSfF",  
    redirectUri: window.location.origin,  
    audience: "http://localhost:8080/online_butik",  
    httpInterceptor: {  
      allowedList: ['http://localhost:8080/cart/updateorderedcart',  
                   'http://localhost:8080/comment/insert',  
                   'http://localhost:8080/order/insert',  
                   'http://localhost:8080/order/update',  
                   'http://localhost:8080/order/isordereditembyemalandslug',  
                   'http://localhost:8080/users/insert',  
                   'http://localhost:8080/users/update']  
    },  
  })  
],
```

```
const routes: Routes = [  
  {path: '', component: HomeComponent},  
  {path: 'register', component: RegisterComponent},  
  {path: 'login', component: LoginComponent},  
  {path: 'profile', component: ProfileComponent, canActivate: [AuthGuard]},  
  {path: 'item/:slug', component: ItemComponent},  
  {path: 'items/:category/:type', component: ItemsComponent},  
  {path: 'cart', component: CartComponent}  
]
```

## 4. Zakljucak

U dokumentaciji je projekat prikazat sažeto. Za detaljniji opis projekta neophodno je analizirati izvorni kod klijentske i serverske aplikacije. Fokus dokumentacije je serverska aplikacija kao i izgled korisničkog interfejsa.

Veb servisi bazirani na REST arhitekturi jesu najpopularniji način kreiranja veb servisa. Prednost u odnosu na SOAP jeste veća fleksibilnost i jednostavnost. Iako veb servisi koji koriste GraphQL rapidno dobijaju na popularnosti, u bliskoj budućnosti neće premašiti popularnost RESTful veb servisa.