Metode servisa za unos, modifikaciju i brisanje podataka

Kreiranje klase

```
export class Osoba {
     constructor(public osobaId: number, public ime: string, public prezime: string, public starost: number )
     {
     }
}
```

Kreiranje servisa za komunikaciju sa Api -jem

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpErrorResponse} from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { Osoba } from './osoba';
import { catchError } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class OsobaService {

  constructor(private http: HttpClient) { }

  private apiUrl = '/api/osobe';

    private errorHandler(error: HttpErrorResponse) {
    return throwError(() => error);
    ...
  }
}
```

Komponenta

```
export class ListaOsobaComponent implements OnInit {

  osobe: Osoba[] = [];
  odabranaOsoba: Osoba = new Osoba(0, '', '', 0);
  idOsobe = 0;
  imeOsobe = "";
  prezimeOsobe="";
  starostOsobe = 0;

  resetujPolja(): void {
    this.imeOsobe = '';
    this.prezimeOsobe = '';
    this.starostOsobe = 0;
  }

  constructor(private oServis: OsobaService) {
  }


  ....
  }
```

Slanje podataka na sever

- **Angular komponenta:**
  - poziva metodu servisa
  - kao argument prosleđuje objekat Osoba
- **Angular servis**:
  - prima objekat Osoba kao parametar metode
  - serijalizuje objekat u JSON
  - šalje JSON ka serveru asinhronim **HTTP POST** zahtevom
  - odmah vraća Observable<Osoba>
    - servis ne vraća odmah podatke, već vraća objekat koji će podatke isporučiti kasnije
    - U tom trenutku server još nije odgovorio, nema podataka, postoji samo Observable koji će emitovati podatke kada odgovor servera stigne

Slanje podataka na sever

- **Server (Web API)**
  - prima JSON koji predstavlja objekat Osoba
  - automatski mapira JSON u objekat tipa Osoba
  - snima podatke (npr. u bazu)
  - vraća Osoba kao JSON u HTTP odgovoru
- **Angular servis (nakon odgovora servera)**
  - prima JSON odgovor
  - mapira ga u objekat tipa Osoba
  - emituje objekat kroz Observable
- **Angular komponenta**:
  - preko subscribe() prima objekat Osoba koji je emitovao Angular servis
  - koristi podatke (prikaz)

POST metoda servisa za komunikaciju sa Web Api aplikacijom

Ova metoda šalje objekat tipa Osoba na server putem POST zahteva i vraća rezultat kao Observable, uz centralizovanu obradu grešaka.
Metoda vraća Observable, a stvarni podaci se dobijaju tek kada server pošalje odgovor i kada se izvrši subscribe()

```
ubaciOsobu(os: Osoba): Observable<Osoba> {
  return this.http.post<Osoba>(this.apiUrl, os).pipe(
    catchError(this.errorHandler)
  );
}
```

Komponenta – unos podataka

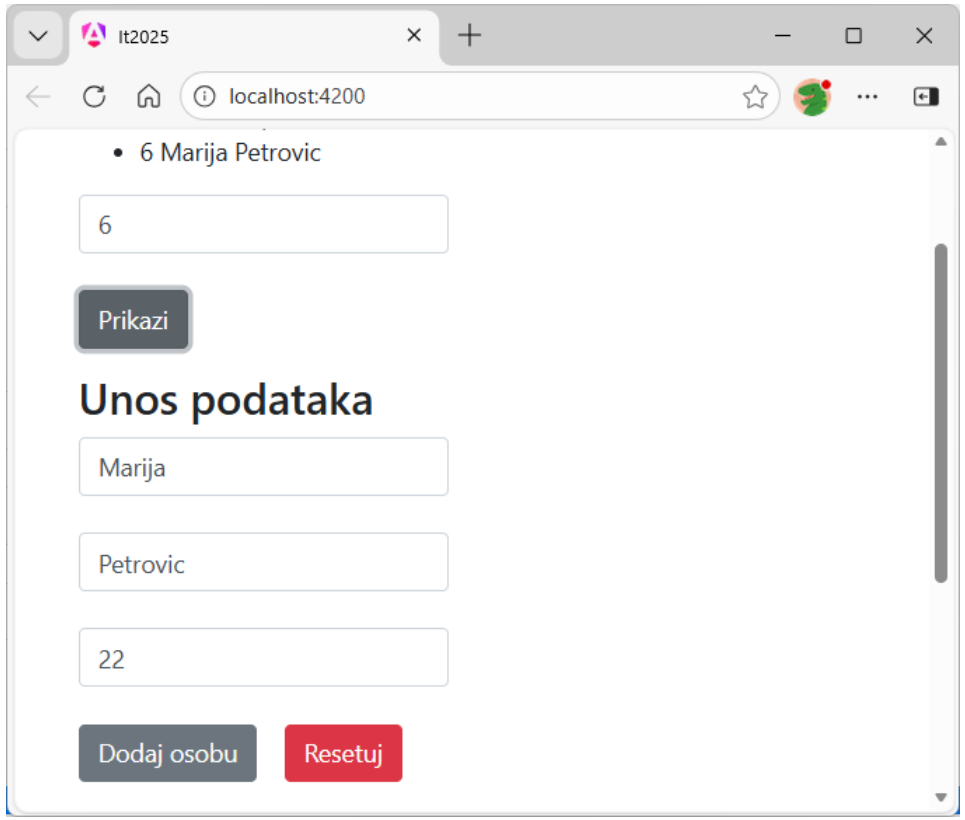
```
ubaciOsobu(ime: string, prezime: string, starost: number): void {

    this.idOsobe = this.osobe.length + 1;
    const os1 = new Osoba(this.idOsobe, ime, prezime, starost);
    this.oServis.ubaciOsobu(os1).subscribe({
      next: podaci => {
        console.log(podaci);
        this.prikaziOsobe();
      },
      error: gr => console.log(gr)
    });
  }
```

Komponenta kreira objekat Osoba, šalje ga servisu i kroz subscribe() prima odgovor servera.

Komponenta- interfejs za unos

```
<h3>Unos podataka</h3>
<div class="row">
    <div class="col-6">

        <div class="form-group">
            <input type="text" [(ngModel)]="imeOsobe" placeholder="Unesi ime osobe"
            class="form-control"><br>
            <input type="text" [(ngModel)]="prezimeOsobe" placeholder="Unesi prezime osobe“
            class="form-control"><br>
            <input type="number" [(ngModel)]="starostOsobe" placeholder="Unesi starost osobe“
            class="form-control"><br>

            <button class="btn btn-secondary" (click)="ubaciOsobu(imeOsobe, prezimeOsobe, starostOsobe)">
            Dodaj osobu</button>
              
            <button class="btn btn-danger" (click)="resetujPolja()">Resetuj</button>
          </div>
    </div>
</div>
```

Interfejs za unos podataka



DELETE metoda – brisanje podataka

- Komponenta poziva metodu servisa i prosleđuje ID
- Servis šalje asinhroni HTTP DELETE zahtev serveru
- Server briše podatke i vraća potvrdu
- Servis emituje ID obrisanog objekta kroz Observable
- Komponenta preko subscribe() ažurira prikaz
- **DELETE** zahtev ne vraća podatke, već informaciju da je brisanje uspešno

DELETE metoda servisa za komunikaciju sa Web Api aplikacijom

```
obrisiOsobu(id: number): Observable<number> {
  const url = `${this.apiUrl}/${id}`;
  return this.http.delete<void>(url).pipe(
    map(() => id),
    catchError(this.errorHandler)
  );
}
```

- DELETE odgovor servera često nema telo (nema podataka)
- map(() => id) omogućava da servis vrati ID obrisanog objekta, iako DELETE odgovor servera ne sadrži podatke

Komponenta – brisanje podataka

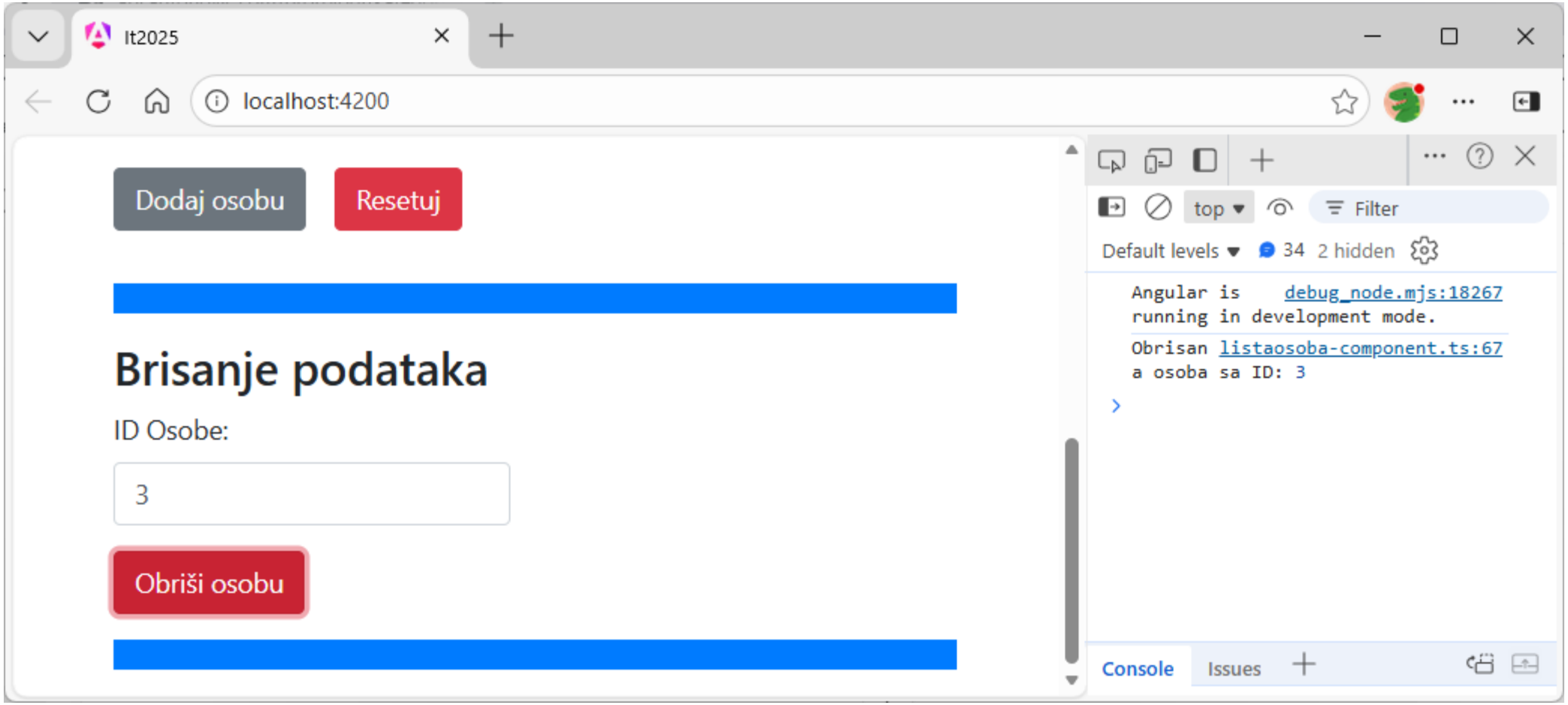
```
obrisiOsobu(): void {
  this.oServis.obrisiOsobu(this.idOsobe).subscribe({
    next: (id) => {
      console.log('Obrisana osoba sa ID:', id);
      this.prikaziOsobe();
    },
    error: (greska) => {
      console.log('Greška:', greska);
    }
  });
}
```

- Poziva se metoda servisa za brisanje sa prosleđenim ID-jem
- subscribe() aktivira HTTP DELETE zahtev
- next se izvršava nakon uspešnog brisanja na serveru
- error se izvršava ako dođe do greške tokom brisanja

Komponenta – interfejs za brisanje

```
<div class="row">
    <div class="col-6">
      <div class="form-group">
        <label for="idOsobe">ID Osobe:</label>
        <input type="number" id="idOsobe" [(ngModel)]="idOsobe“
         class="form-control" placeholder="Unesi ID osobe">
      </div>

      <button class="btn btn-danger" (click)="obrisiOsobu()">Obriši osobu
      </button>
    </div>
  </div>
```

Interfejs za brisanje podataka



PUT metoda servisa za komunikaciju sa Web Api aplikacijom

```
promeniOsobu(os: Osoba): Observable<Osoba> {
  const url = `${this.apiUrl}/${os.id}`;


  return this.http.put<Osoba>(url, os).pipe(
    catchError(this.errorHandler)
  );
}
```

PUT metoda šalje izmenjeni objekat serveru i vraća ažurirani objekat kroz Observable, uz obradu grešaka

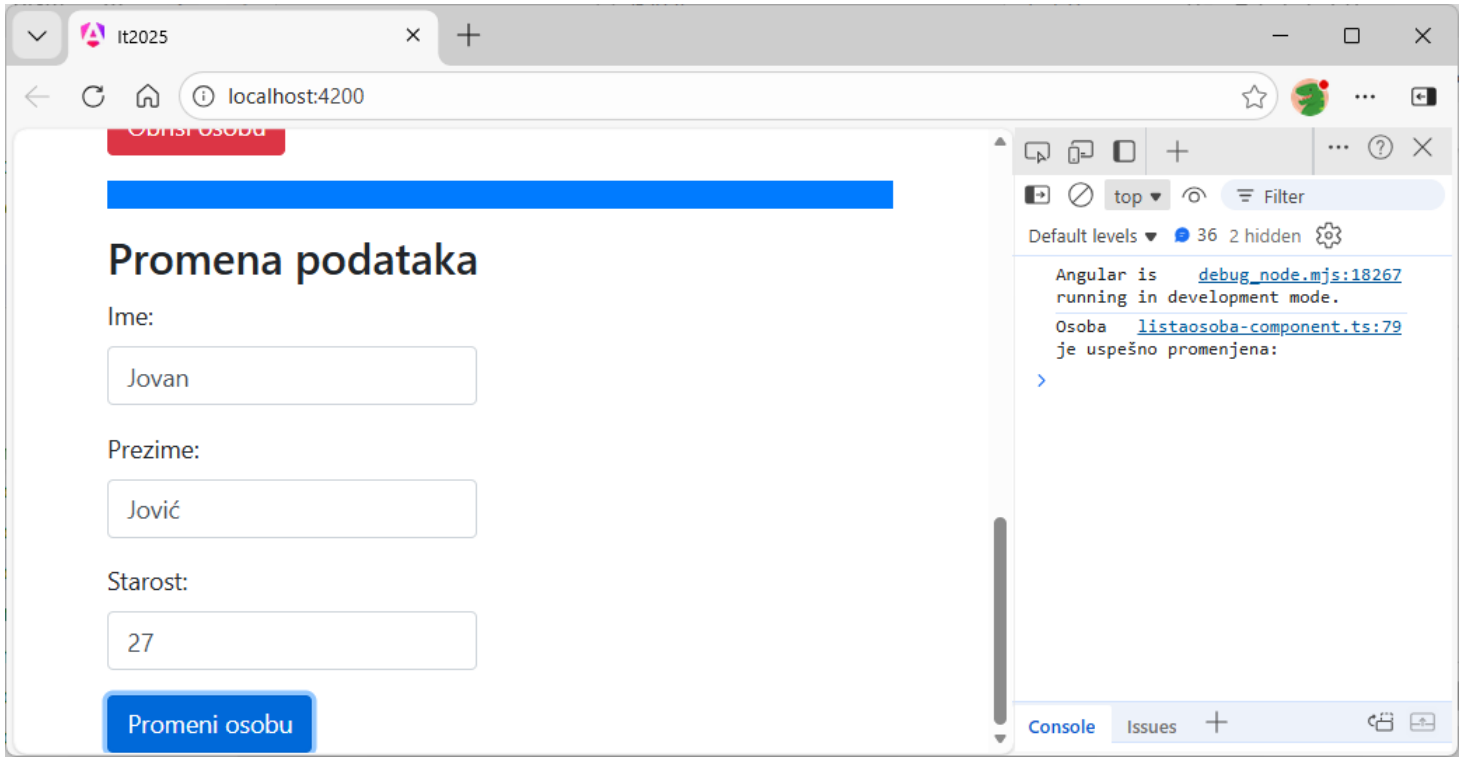
Komponenta -promena podataka

```
promeniOsobu(): void {
  this.oServis.promeniOsobu(this.odabranaOsoba).subscribe({
    next: (izmenjenaOsoba) => {
      console.log('Osoba je uspešno promenjena:');
      this.prikaziOsobe();
    },
    error: (greska) => {
      console.log('Greška prilikom promene osobe:', greska);
    }
  });
}
```

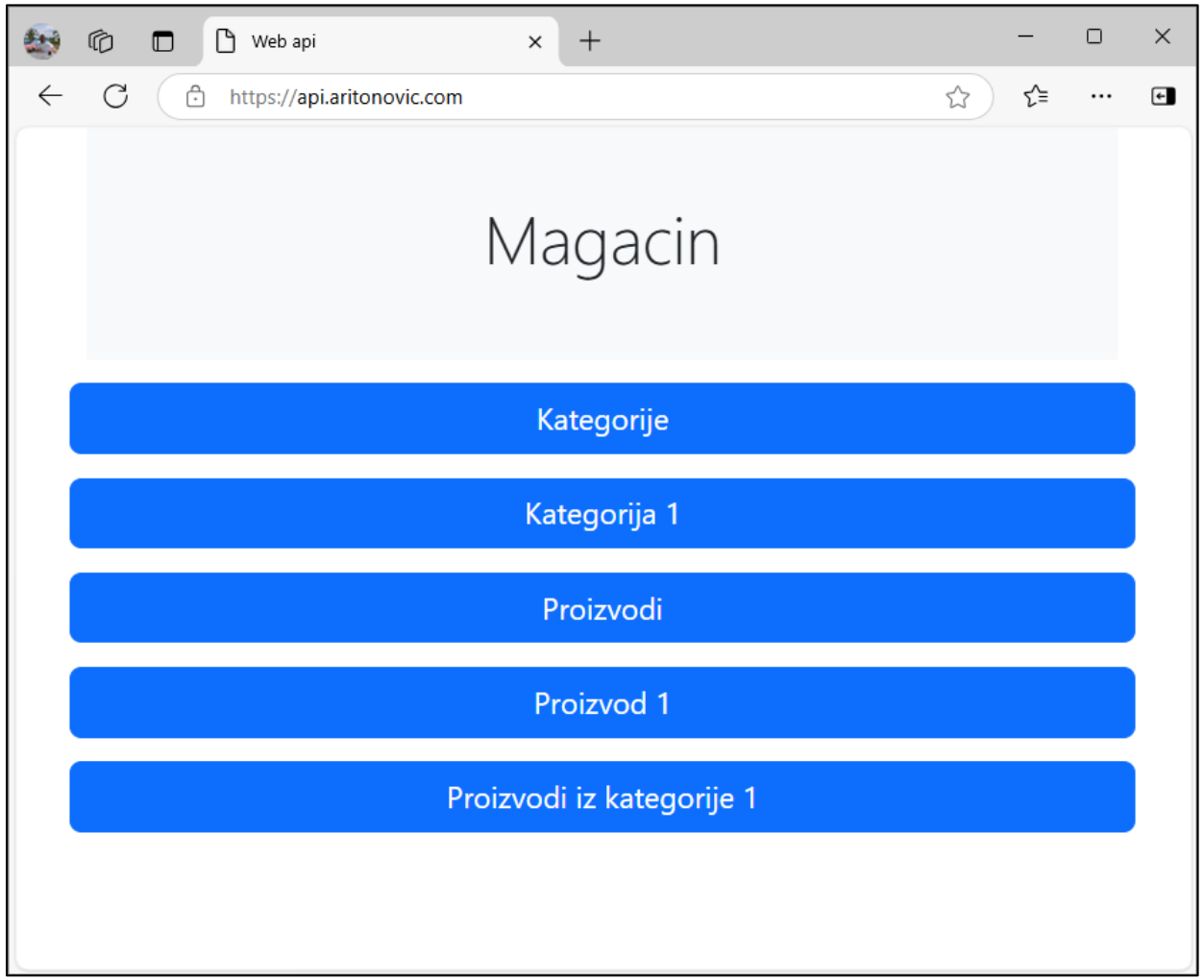
Komponenta – šablon za promenu

```
<h3>Promena podataka</h3>
  <div class="row">
    <div class="col-6">
      <div class="form-group">
        <label for="ime">Ime:</label>
        <input type="text" id="ime" [(ngModel)]="odabranaOsoba.ime" class="form-control">
      </div>
      <div class="form-group">
        <label for="prezime">Prezime:</label>
        <input type="text" id="prezime" [(ngModel)]="odabranaOsoba.prezime" class="form-control">
      </div>
      <div class="form-group">
        <label for="starost">Starost:</label>
        <input type="number" id="starost" [(ngModel)]="odabranaOsoba.starost" class="form-control">
      </div>

      <button class="btn btn-primary" (click)="promeniOsobu()">Promeni osobu</button>
    </div>
```

Interfejs za promenu podataka



Primer web api -aplikacije

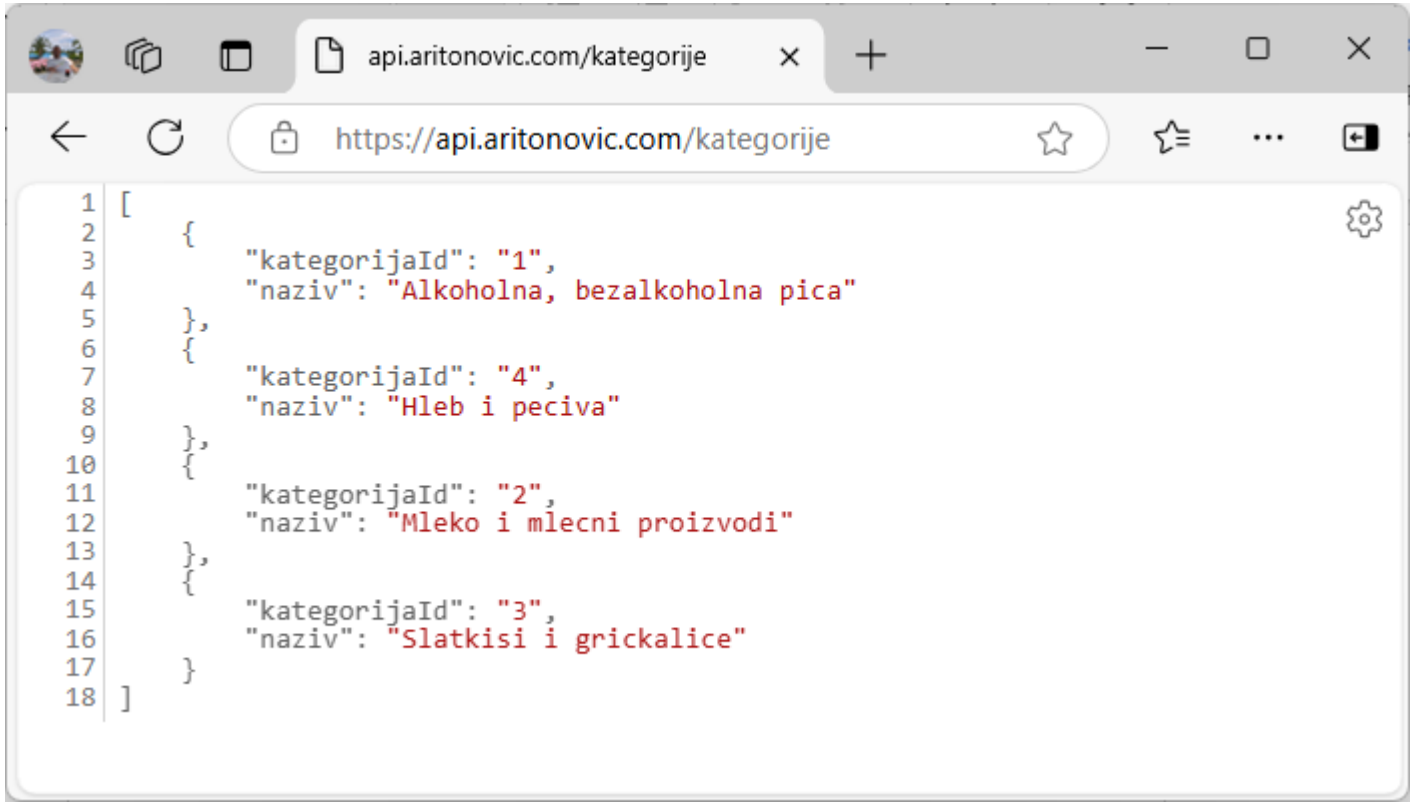


https://api.aritonovic.com/

web api dozvoljava CORS zahteve

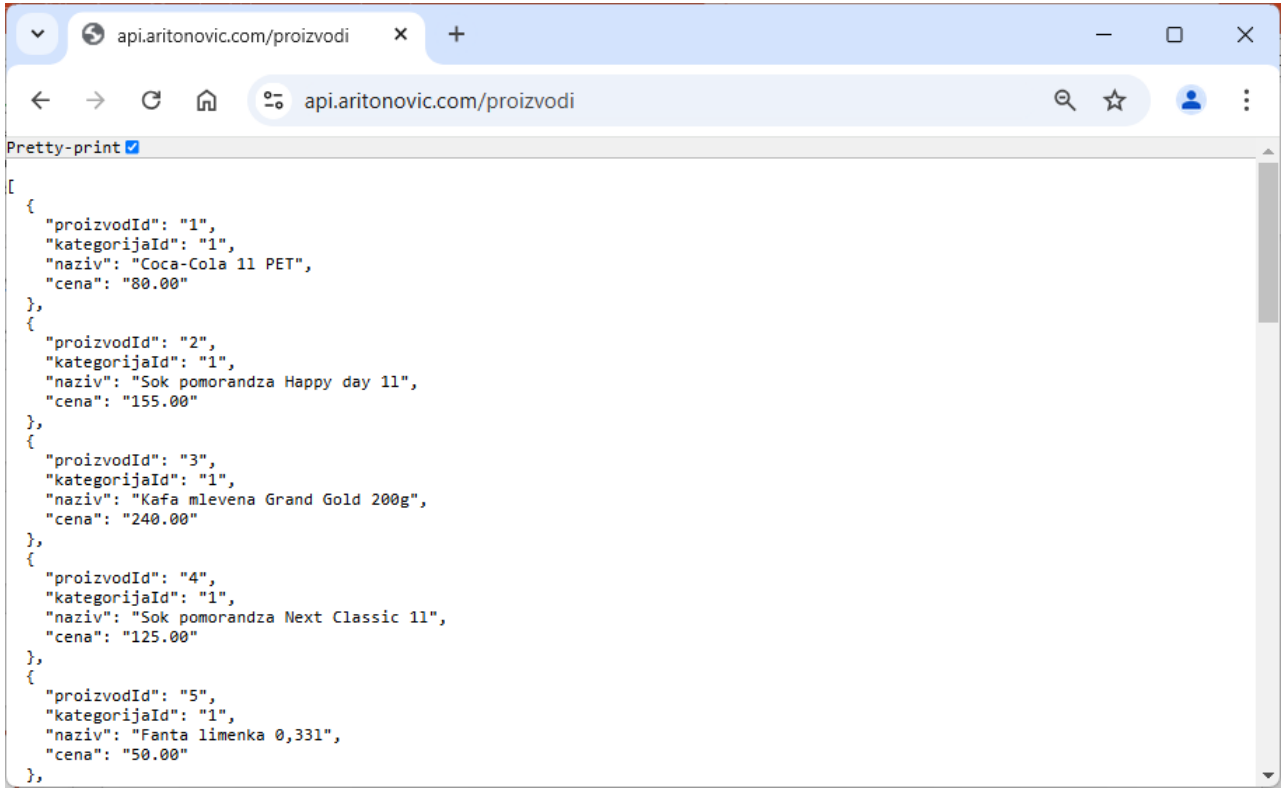
Prikaz kategorija

https://api.aritonovic.com/kategorije



Prikaz proizvoda

https://api.aritonovic.com/proizvodi




```
[
  {
    "proizvodId": "1",
    "kategorijaId": "1",
    "naziv": "Coca-Cola 1l PET",
    "cena": "80.00"
  },
  {
    "proizvodId": "2",
    "kategorijaId": "1",
    "naziv": "Sok pomorandza Happy day 1l",
    "cena": "155.00"
  },
  {
    "proizvodId": "3",
    "kategorijaId": "1",
    "naziv": "Kafa mlevena Grand Gold 200g",
    "cena": "240.00"
  },
  {
    "proizvodId": "4",
    "kategorijaId": "1",
    "naziv": "Sok pomorandza Next Classic 1l",
    "cena": "125.00"
  },
  {
    "proizvodId": "5",
    "kategorijaId": "1",
    "naziv": "Fanta limenka 0,33l",
    "cena": "50.00"
  },
```

Bootstrap aplikacije

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app/app.component';
import { provideHttpClient } from '@angular/common/http';

bootstrapApplication(AppComponent, {
  providers: [
    provideHttpClient()
  ]
});
```

Klasa Proizvod

```
export class Proizvod {
    constructor(public proizvodId: number, public kategorijaId: number,
      public naziv: string, public cena: number, public opis: string) {
    }
  }
```

Klasa MagacinService

```
@Injectable({
  providedIn: 'root',
})
export class MagacinService {
  apiUrl = 'https://api.aritonovic.com';

  constructor(private http: HttpClient) { }

  vratiProizvode(): Observable<Proizvod[]> {
    return this.http.get<Proizvod[]>(this.apiUrl + '/proizvodi')
      .pipe(catchError(this.errorHandler));
  }

private errorHandler(error: HttpErrorResponse): Observable<never> {
  return throwError(() => error);
}
}
```

Observable<never> govori TypeScript-u: „ovde se ne emituju podaci, samo greška“

GET metoda servisa – preuzimanje podataka

- Šalje HTTP GET zahtev ka Web API-ju
- Preuzima listu proizvoda sa rute /proizvodi
- Server vraća podatke u JSON formatu
- JSON predstavlja **niz objekata tipa Proizvod**
- Angular automatski **mapira JSON u Proizvod[]**
- Vraća rezultat kao Observable<Proizvod[]>
  - Metoda ne vraća podatke odmah
  - Vraća Observable, jer je HTTP GET zahtev asinhron
  - Kada server odgovori: Observable emituje niz proizvoda (Proizvod[])
  - Komponenta dobija podatke tek nakon **subscribe**()
- Greške tokom zahteva se obrađuju pomoću errorHandler

Klasa komponente Proizvodi

```
export class ProizvodiComponent implements OnInit {

  proizvodi: Proizvod[] = [];

  constructor(private mServis: MagacinService) {}

  prikaziProizvode(): void {
    this.mServis.vratiProizvode().subscribe({
      next: proizvodi => this.proizvodi = proizvodi,
      error: greska => console.log('Greska: ' + greska)
    });
  }

  ngOnInit(): void {
    this.prikaziProizvode();
  }

}
```

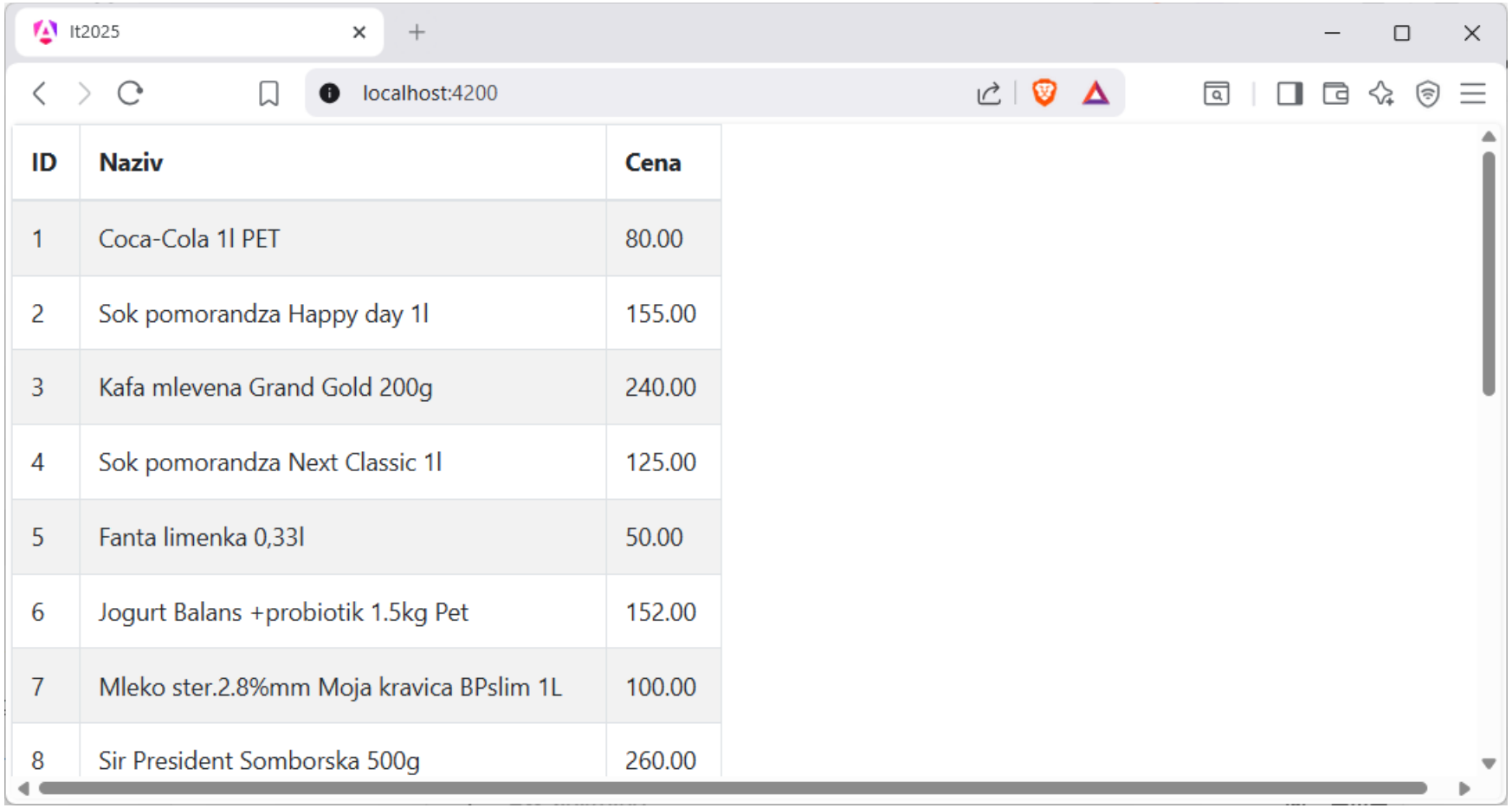
Metoda komponente

- Komponenta preko servisa asinhrono preuzima listu proizvoda i prikazuje ih nakon odgovora servera
- Metoda prikaziProizvode():
  - Poziva GET metodu servisa
  - Pretplaćuje se na Observable<Proizvod[]>
    - next:prima JSON podatke mapirane u Proizvod[]smešta ih u niz proizvodi
    - error:prima grešku sa serveraispisuje poruku u konzoli

Šablon komponente proizvodi

```
<div class="row">
  <div class="col-6">
    <table class="table table-bordered table-striped">
      <thead>
        <tr>
          <th>ID</th>
          <th>Naziv</th>
          <th>Cena</th>
        </tr>
      </thead>


      <tbody>
        @for (proizvod of proizvodi; track proizvod.proizvodId) {
          <tr>
            <td>{{ proizvod.proizvodId }}</td>
            <td>{{ proizvod.naziv }}</td>
            <td>{{ proizvod.cena }}</td>
          </tr>
        }
      </tbody>
    </table>
  </div>
</div>
```

Prikaz podataka



ID	Naziv	Cena
1	Coca-Cola 1l PET	80.00
2	Sok pomorandza Happy day 1l	155.00
3	Kafa mlevena Grand Gold 200g	240.00
4	Sok pomorandza Next Classic 1l	125.00
5	Fanta limenka 0,33l	50.00
6	Jogurt Balans +probiotik 1.5kg Pet	152.00
7	Mleko ster.2.8%mm Moja kravica BPslim 1L	100.00
8	Sir President Somborska 500g	260.00

HTML forme

Element <form>

- Element <form> omogućava unos podataka od strane korisnika
- Atribut action elementa <form> određuje URL adresu (ili serversku metodu) na koju se podaci sa forme šalju
- Atribut method specificira način slanja podataka
  - GET metoda je podrazumevana opcija
  - POST metoda se koristi za slanje većih ili osetljivijih podataka
- Atribut enctype određuje način enkodovanja podataka sa forme pre slanja serveru

Kontrole forme - element <input>

- Element <input> je osnovni HTML element za unos podataka
- Ima više različitih oblika u zavisnosti od atributa **type**
  - type="text"
  - type="password"
  - type="hidden"
  - type="checkbox"
  - type="radio"
  - type="reset"
  - type="submit"
  - type="image"
  - type="button"
  - type="file"
  - type="email"

Atributi <input> elementa

- Atribut **id** koristi se za pristup elementu iz CSS-a ili JavaScripta
  - Atribut id mora biti jedinstven u okviru stranice (forme)
- Atribut **value** postavlja podrazumevanu vrednost za numeričke i tekstualne kontrole za unos podataka
- Atribut **name** se koristi od strane servera da se referenciraju polja forme kada se ona submituje, takođe se koristi za pristup elementu iz klijentskog koda
- Angular koristi atribut **name** za identifikaciju kontrole unutar forme
- Atribut **placeholder** opisuje očekivanu vrednost unutar tekstualnog polja
- Atribut **required** označava da je polje obavezno i da se forma ne može poslati ukoliko je polje prazno

Text polje

- Element <input type="text"> kreira polje za unos teksta
- Svojstvo **value** služi za čitanje i postavljanje unetog teksta iz JavaScript koda

```
<input type="text" name="ime" id="textIme">
```

Element <**label**>

- Element <**label**> služi za pridruživanje teksta ulaznoj kontroli na osnovu vrednosti atributa for, koji mora odgovarati id atributu ulaznog elementa
- Klik na tekst <label> elementa automatski postavlja fokus na povezanu ulaznu kontrolu
- Element <label> poboljšava pristupačnost forme, jer čitači ekrana povezuju opis sa odgovarajućim poljem

```
<label for="textIme">Ime:</label> <br>
<input type="text" name="ime" id="textIme">
```

Element **<fieldset>**

- Element <fieldset> je opciona HTML kontrola koja služi za grupisanje povezanih kontrola unutar forme
- Unutar <fieldset> elementa može se nalaziti element <legend>, koji opisuje grupu kontrola
- Ako se koristi, <legend> mora biti prvi element unutar <fieldset> elementa

Element <button>

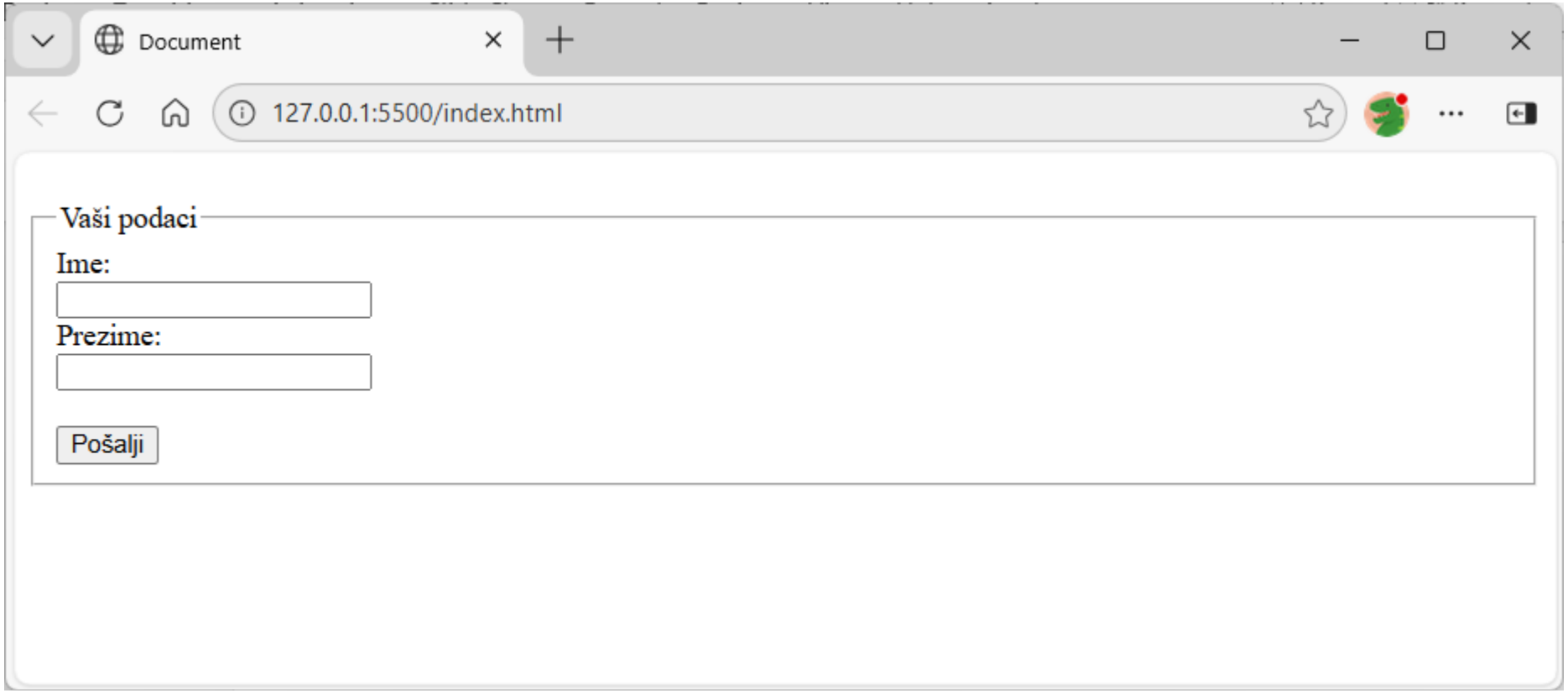
- Element <**button**> se koristi za definisanje dugmeta u HTML formi
- Podrazumevana vrednost atributa type za <button> element je **submit**
- Atribut type može imati sledeće vrednosti:
  - submit
  - reset
  - button
- U Angular aplikacijama preporučuje se eksplicitno navođenje atributa type, kako bi se izbeglo nenamerno slanje forme

```
<button type="submit">Posalji</button>
```

Primer HTML forme sa upotrebom elementa fieldset

```
<form action="/primer01.html" method="get">
<fieldset>
    <legend>Vaši podaci</legend>


    <label for="textIme">Ime:</label><br>
    <input type="text" name="ime" id="textIme" required><br>


    <label for="textPrezime">Prezime:</label><br>
    <input type="text" name="prezime" id="textPrezime" required><br><br>


    <button type="submit">Pošalji</button>
</fieldset>
</form>
```

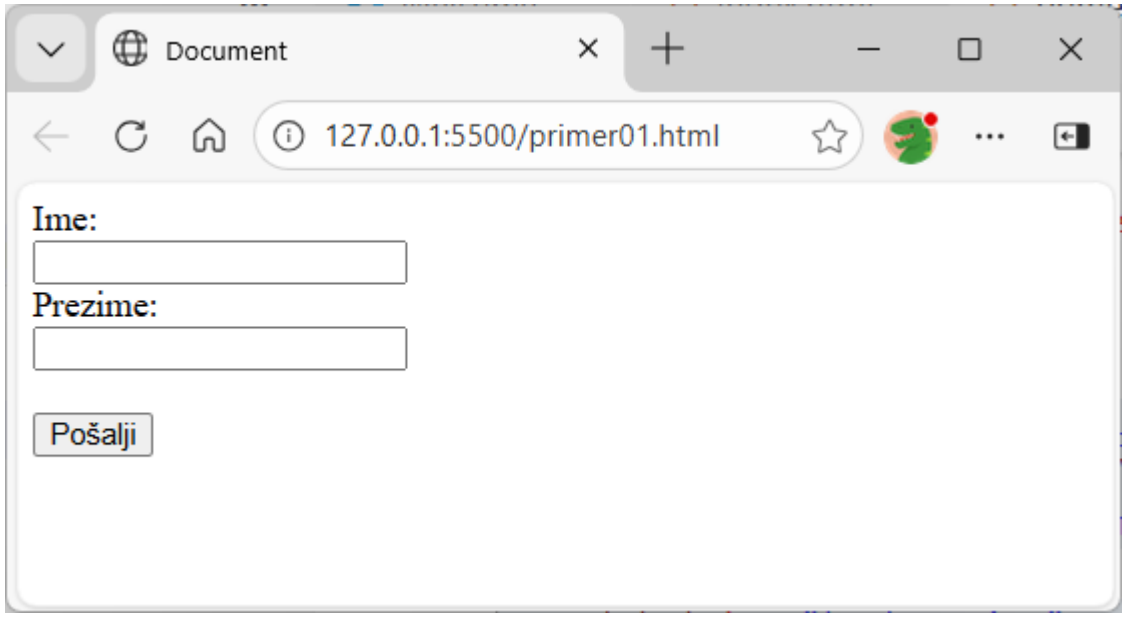
Primer HTML forme sa upotrebom elementa fieldset



Forma bez upotrebe fieldset elementa

```
  <form action="/primer02.html" method="get">
   <label for="textIme">Ime:</label><br>
   <input type="text" name="ime" id="textIme" required><br>


   <label for="textPrezime">Prezime:</label><br>
   <input type="text" name="prezime" id="textPrezime" required><br><br>


   <button type="submit">Pošalji</button>
</form>
```

Forma bez upotrebe fieldset elementa

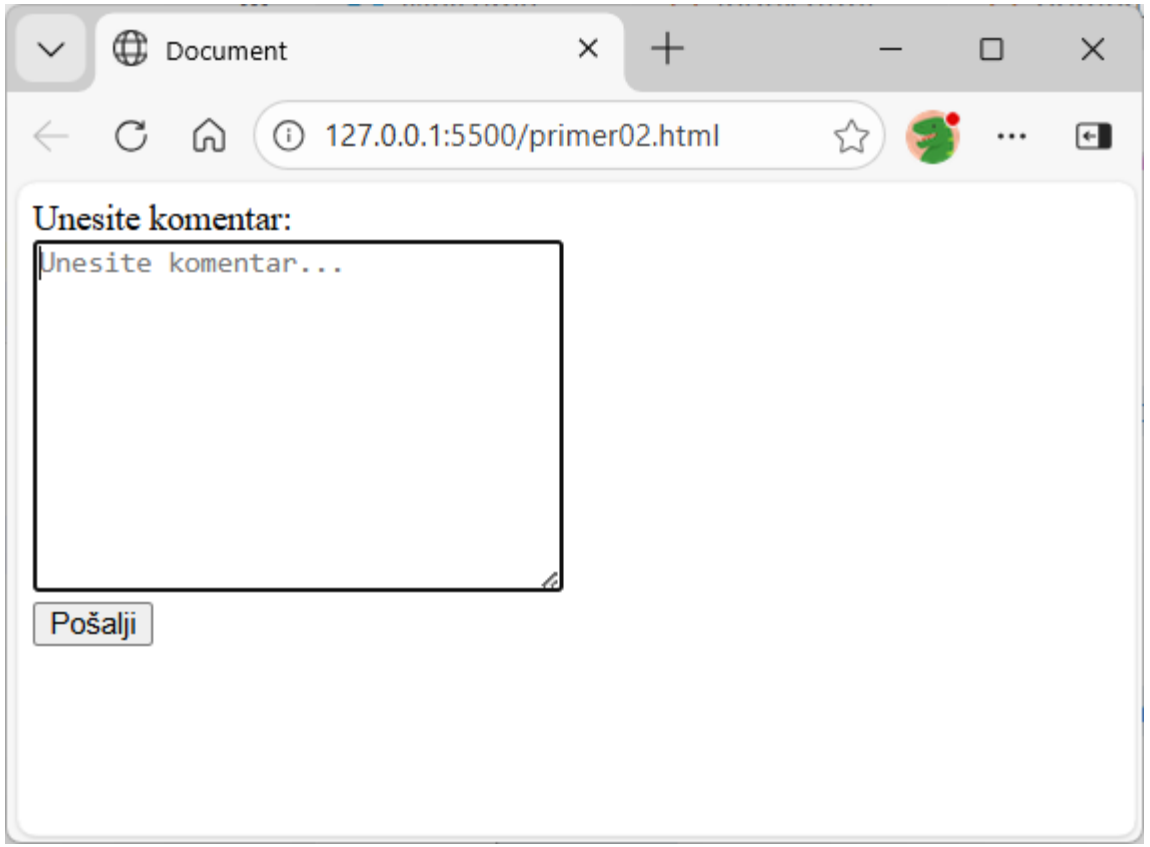


Element <textarea>

- Element **<textarea>** predstavlja multiline polje za unos teksta

```
<form action="">
    <label for="komentar">Unesite komentar:</label><br>
    <textarea
    name="komentar"
    id="komentar"
    cols="30"
    rows="10"
    placeholder="Unesite komentar..."
    required></textarea>


    <br>
    <button type="submit">Pošalji</button>
</form>
```

Prikaz tekst oblasti

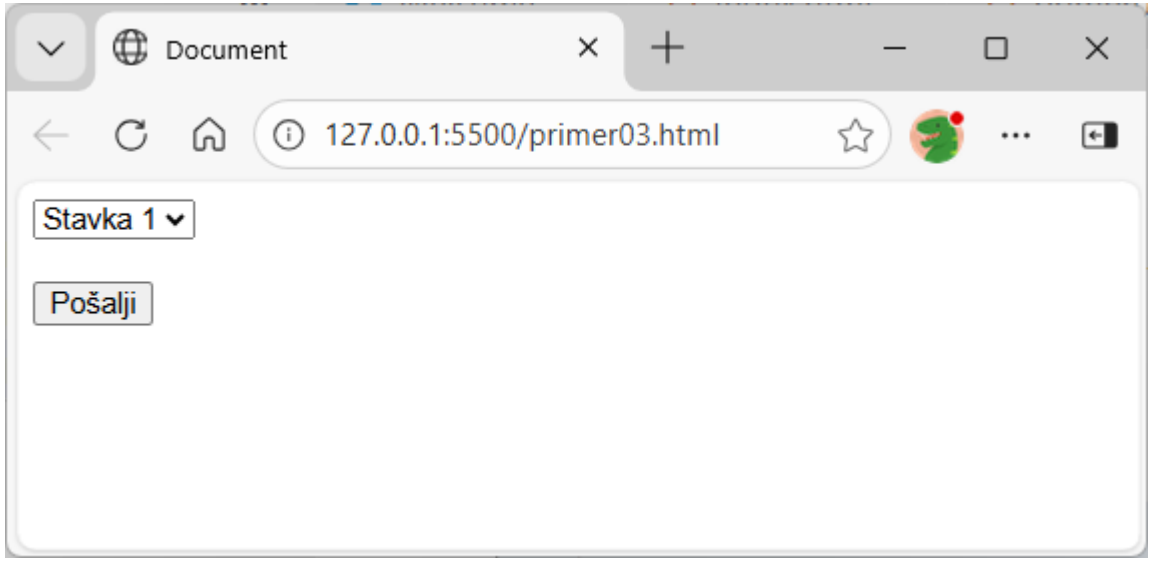


Element <select>

- Element <select> se koristi za definisanje drop-down liste ili listbox kontrole
- Element <option> definiše pojedinačne stavke liste
- Atribut value određuje vrednost koja se šalje serveru
- Atribut selected određuje podrazumevano izabranu stavku
- Ako se koristi atribut size sa vrednošću većom od 1, lista se prikazuje kao listbox

Element <select>

```
<form action="">
    <select name="stavka" id="stavka">
        <option value="1" selected>Stavka 1</option>
        <option value="2">Stavka 2</option>
        <option value="3">Stavka 3</option>
    </select>
    <br><br>
    <button type="submit">Pošalji</button>
</form>
```

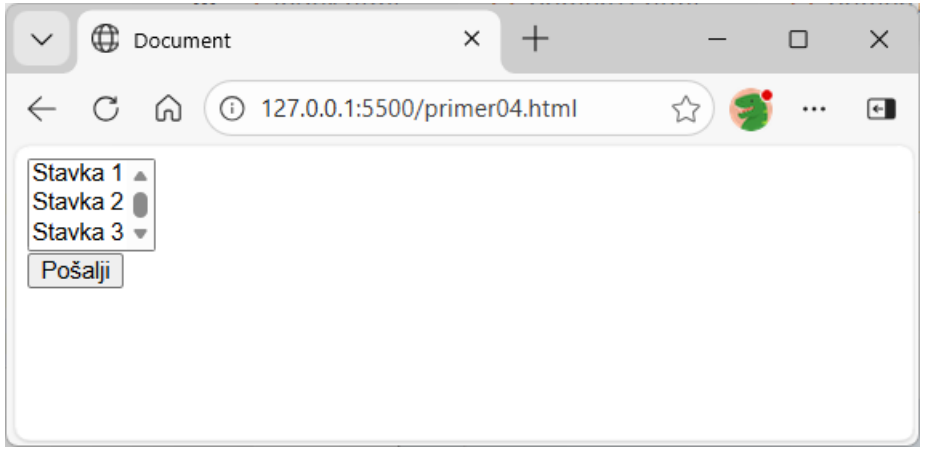


Element <select> sa atributom size

- Atribut size određuje broj vidljivih stavki u <select> elementu
- Ako je vrednost atributa size veća od 1, <select> se prikazuje kao listbox
- Korisnik može da izabere jednu stavku (više stavki samo uz atribut multiple)

Element <select> sa atributom size

```
<form action="">
    <select name="stavka" id="stavka" size="3">
        <option value="1">Stavka 1</option>
        <option value="2">Stavka 2</option>
        <option value="3">Stavka 3</option>
        <option value="4">Stavka 4</option>
    </select>
    <br>
    <button type="submit">Pošalji</button>
</form>
```



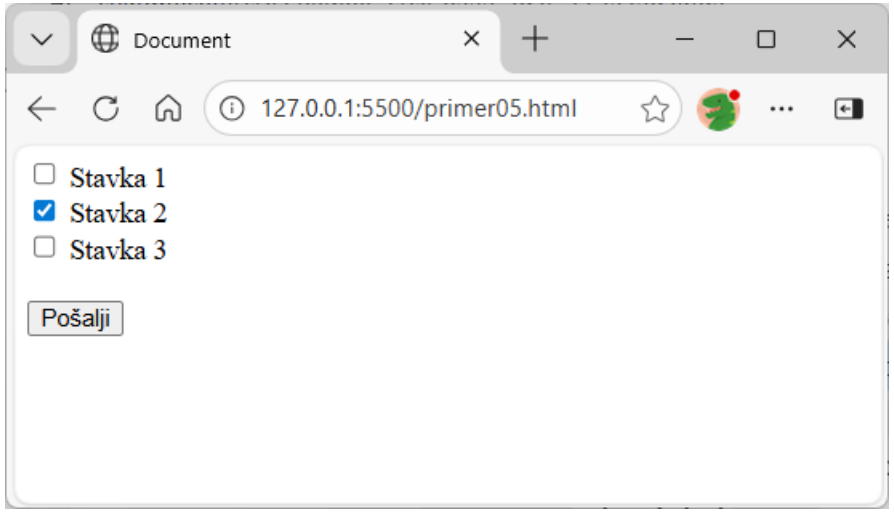
Kontrole checkbox i radio button

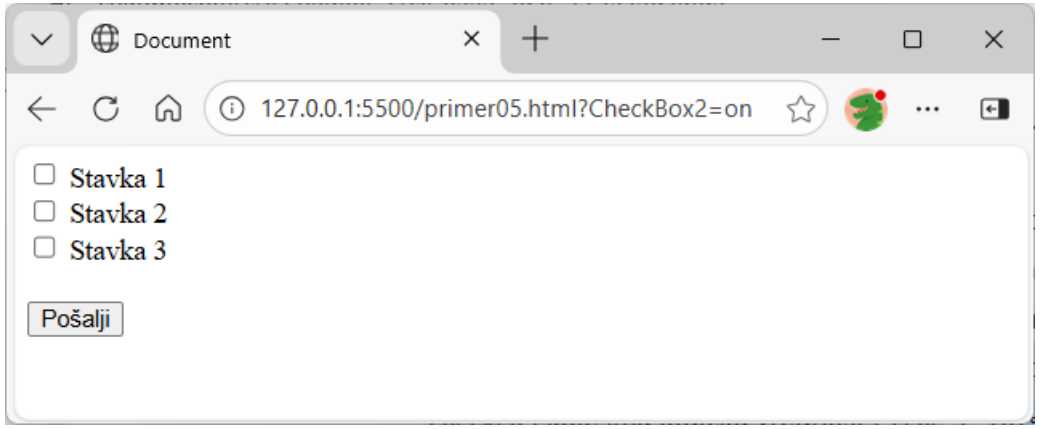
- Element <input type="checkbox"> kreira checkbox kontrolu i omogućava nezavisan izbor više opcija
- Element <input type="radio"> kreira radio button i omogućava izbor jedne opcije unutar iste grupe
- Atribut **checked** određuje da je kontrola podrazumevano selektovana
- U JavaScript klijentskom kodu, trenutno stanje kontrole čita se preko svojstva **checked**, koje ima logičku vrednost (true / false)
- Prilikom slanja HTML forme, za selektovane checkbox i radio kontrole šalje se par name=value; ako atribut value nije naveden, podrazumevana vrednost je **on**

HTML checkbox

```
<form action="">
    <input id="CheckBox1" type="checkbox" name="CheckBox1">
    <label for="CheckBox1">Stavka 1</label><br>


    <input id="CheckBox2" type="checkbox" name="CheckBox2">
    <label for="CheckBox2">Stavka 2</label><br>


    <input id="CheckBox3" type="checkbox" name="CheckBox3">
    <label for="CheckBox3">Stavka 3</label><br><br>


    <button type="submit">Pošalji</button>
</form>
```

Prikaz checkbox kontrola



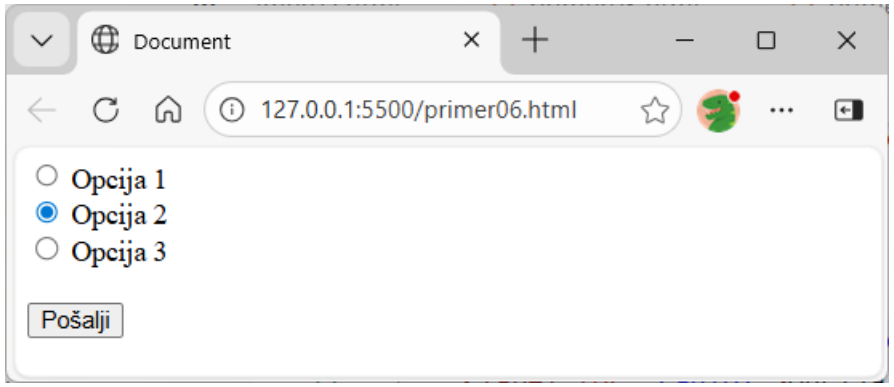


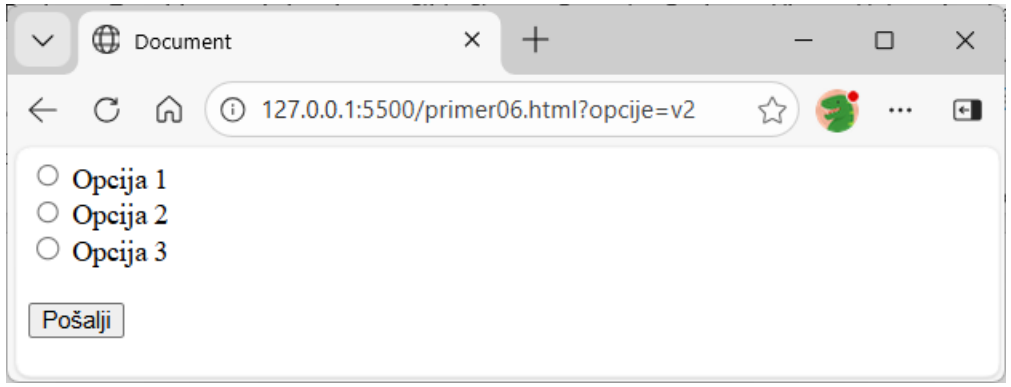
Grupa radiobutton kontrola, svojstvo name

- Element <input type="radio"> omogućava izbor jedne opcije.
- Radio buttoni pripadaju istoj grupi ako imaju istu vrednost atributa **name**
- Unutar jedne grupe može biti selektovana samo jedna radio kontrola
- Atribut value određuje vrednost koja se šalje serveru za izabranu opciju
- Prilikom slanja HTML forme, serveru se šalje jedan par name=value za selektovani radio button
- Ako radio button nije selektovan, **ne šalje se nikakva vrednost**

Prikaz radiobutton kontrola

```
<form action="" method="get">
    <input type="radio" name="opcije" id="radio1" value="v1">
    <label for="radio1">Opcija 1</label><br>

    <input type="radio" name="opcije" id="radio2" value="v2">
    <label for="radio2">Opcija 2</label><br>

    <input type="radio" name="opcije" id="radio3" value="v3">
    <label for="radio3">Opcija 3</label><br><br>

    <button type="submit">Pošalji</button>
</form>
```





Angular forme

Forma

- Forma se koristi za prikupljanje podataka od korisnika
- Angular koristi dva tipa formi:
  - Template-driven forme (TDF)
  - Reaktivne forme (model driven forms)
- Klasom **FormControl** predstavlja se jedno ulazno polje Angular forme
  - kod TDF ga Angular kreira automatski,
  - kod reaktivnih formi programer ga eksplicitno kreira
- Klasom **FormGroup** predstavlja se grupa (kolekcija) kontrola forme

FormControl klasa

- Polje forme možemo kreirati i iz koda:
  let ime = new FormControl();
- Svojstvo **value** vraća trenutni sadržaj ovog polja (ime.value)
- Svojstvo **errors** vraća objekat grešaka ili null
- Svojstvo **pristine** ima vrednost true ukoliko vrednost polja nije menjana
- Svojstvo **dirty** vraća true ukoliko je vrednost polja promenjena u odnosu na početnu
- Svojstvo **touched** ima vrednost true ukoliko je polje dobilo i izgubilo fokus
- Svojstvo **untouched** ima vrednost true ukoliko polje nikada nije bilo u fokusu
- Svojstvo **valid** vraća true ukoliko je polje prošlo validaciju

FormGroup klasa

```
let adresa= new FormGroup({
    ulica : new FormControl(""),
    grad : new FormControl(""),
    postanskiBroj : new FormControl("")
});
```

- Omogućava upravljanje grupom kontrola forme
- adresa.grad je način da se pristupi kontroli unutar grupe adresa
- Svojstvo **value** vraća JSON objekat sa vrednostima svih kontrola
- Svojstvo **errors** vraća objekat grešaka grupe ili null
- Svojstvo **valid** vraća true ukoliko su sve kontrole grupe prošle validaciju

Template Driven Forms (TDF)

- Importuje se modul **FormsModule** iz biblioteke @angular/forms
- Kada se uključi FormsModule Angular <form> elementu automatski dodaje direktivu **ngForm**
- Direktiva **ngForm** radi sledeće:
  - automatski kreira instancu **FormGroup** koja odgovara kompletnoj formi
  - kreira instancu klase **FormControl** za svaku kontrolu sa direktivom **ngModel**
- Instanci **ngForm** i svakoj instanci **FormControl** u šablonu možemo dodeliti lokalnu promenljivu
- Koristimo događaj **ngSubmit** da pošaljemo klasi komponente podatke sa forme

Komponenta

```
import { Component} from '@angular/core';
import { FormsModule } from '@angular/forms';


@Component({
  selector: 'app-root',
  imports: [FormsModule],
  templateUrl: './app.html',
  styleUrl: './app.css'
})
export class App {
  title = 'TDF forme';
}
```

Šablon komponente

```
<form>
  <div class="form-group">
    <label for="ime">Ime</label>
    <input type="text" name="ime" id="ime" class="form-control">
  </div>


  <div class="form-group">
    <label for="prezime">Prezime</label>
    <input type="text" name="prezime" id="prezime" class="form-control">
  </div>


  <div class="form-group">
    <button type="submit" class="btn btn-primary">Posalji</button>
  </div>
</form>
```

Direktive ngForm, ngModel, događaj ngSubmit

```
<form #form1="ngForm" (ngSubmit)="onSubmit(form1)">
  <div class="form-group">
    <label for="ime">Ime</label>
    <input type="text" name="ime" id="ime" class="form-control" ngModel />
  </div>

  <div class="form-group">
    <label for="prezime">Prezime</label>
    <input type="text" name="prezime" id="prezime" class="form-control" ngModel />
  </div>

  <div class="form-group">
    <button type="submit" class="btn btn-primary">Posalji</button>
  </div>
</form>
```

Direktive ngForm, ngModel, događaj ngSubmit

- Lokalna template promenljiva je identifikator definisan pomoću prefiksa # u HTML šablonu, koji predstavlja referencu na formu ili kontrolu
- Lokalna templejt promenljiva **#form1** daje referencu na instancu **NgForm** objekta
- Direktiva **ngModel** govori Angularu da za svako polje kreira instancu klase FormControl
- Događaj ngSubmit se aktivira prilikom slanja forme i prosleđuje NgForm instancu metodi komponente

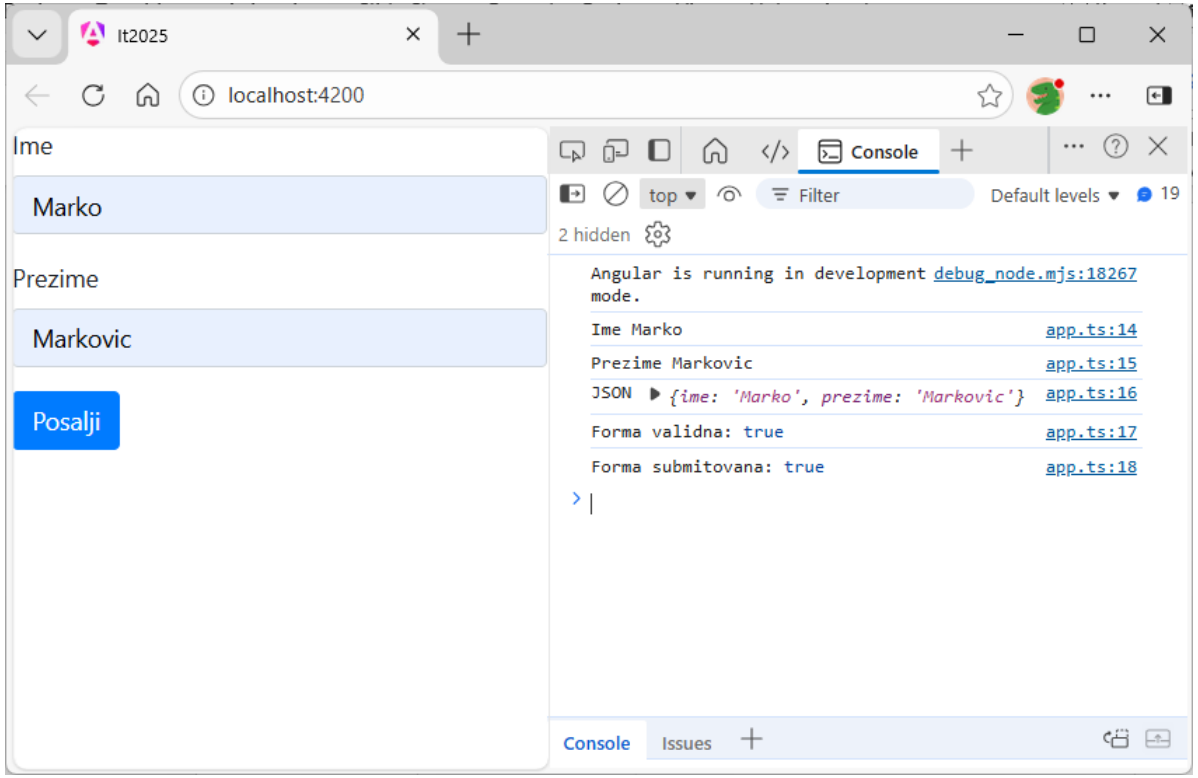
Pristup kontrolama i vrednostima u TDF

- Objekat **NgForm** predstavlja celu HTML formu
- Svojstvo controls je rečnik kontrola forme
- **Ključ** u rečniku je vrednost atributa **name** HTML kontrole
- **Vrednost** u rečniku je odgovarajući objekat tipa **FormControl**
- Pojedinačnoj kontroli pristupa se izrazom **forma.controls['ime’]**
- Vrednost kontrole dobija se preko **forma.controls['ime'].value**
- Izraz forma.value vraća objekat sa svim vrednostima kontrola (JSON struktura)
- Svojstvo **valid** ima vrednost true ukoliko su sve kontrole forme validne

Klasa komponente

```
export class App {
  title = 'TDF forme';
  onSubmit(forma: NgForm) {
    console.log('Ime', forma.controls['ime'].value);
    console.log('Prezime', forma.controls['prezime'].value);
    console.log('JSON', forma.value);
    console.log('Forma validna:', forma.valid);
    console.log('Forma submitovana:', forma.submitted);
  }
}
```

Konzola nakon slanja forme



Model klasa forme

```
export class Osoba {
    constructor(public osobaId: number, public ime: string, public prezime: string) {
    }
}
```

Klasa komponente

```
export class Forma1Component {
title = 'forme';


  model = new Osoba(1, 'Marko', 'Markovic');

  onSubmit() {

    console.log('Ime',this.model.ime);
    console.log('Prezime',this.model.prezime);
    console.log(this.model);
  }
}
```

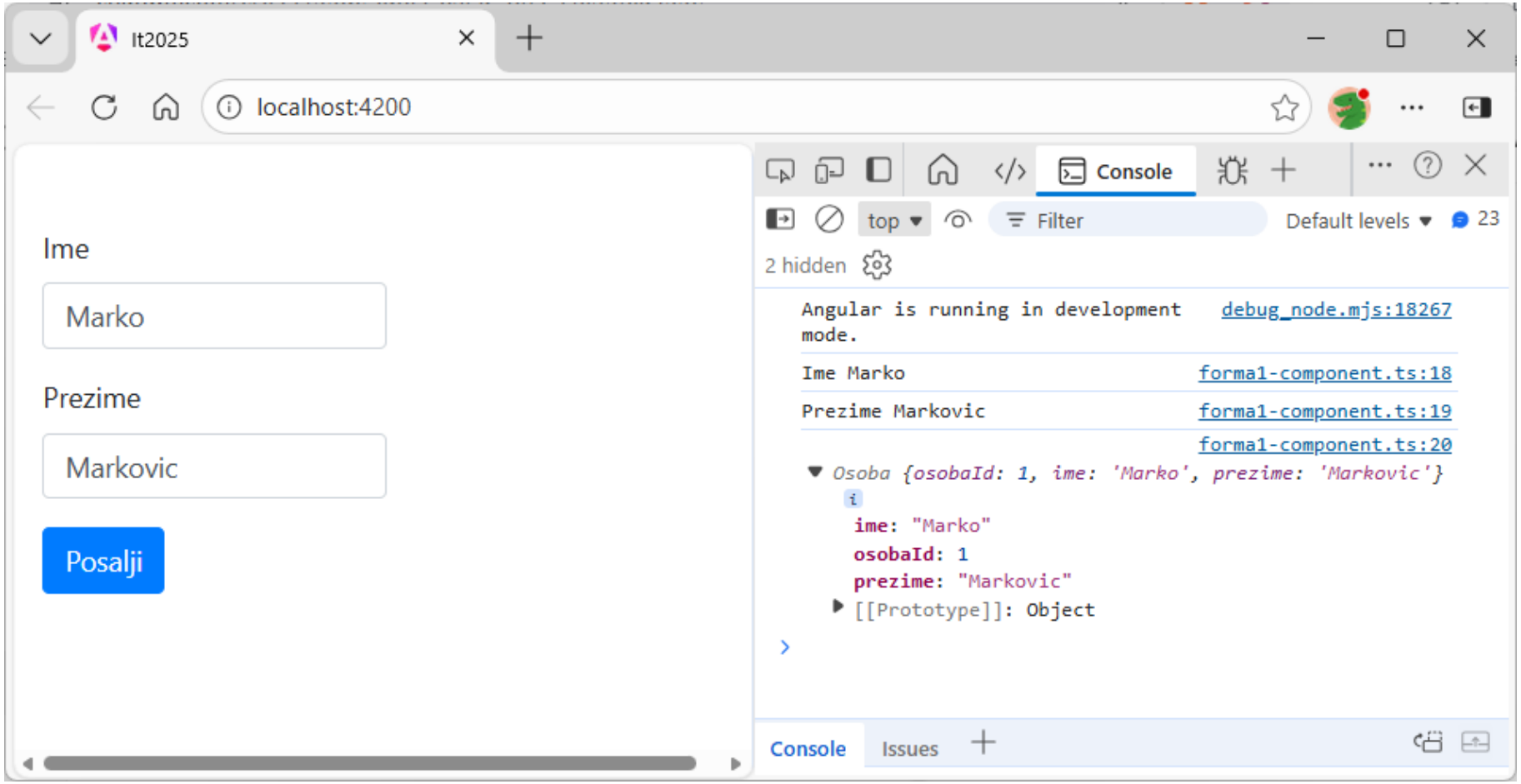
Komponenta sadrži model objekat u koji se automatski upisuju vrednosti forme.

Povezivanje sa modelom

```
<div class="row">
    <div class="col-6">

    <form (ngSubmit)="onSubmit()">

        <div class="form-group">
            <label for="ime">Ime</label>
            <input type="text" name="ime" class="form-control" [(ngModel)]="model.ime">
        </div>
        <div class="form-group">
            <label for="prezime">Prezime</label>
            <input type="text" name="prezime" id="prezime" class="form-control“
            [(ngModel)]="model.prezime">
        </div>
        <button class="btn btn-primary"  type="submit">Posalji</button>
    </form>
</div>
```

Direktiva [(ngModel)] povezuje polja forme sa modelom i omogućava dvosmernu razmenu podataka.

Korisnički interfejs



Načini rada sa Template-driven formama

TDF sa **NgForm**	TDF sa **modelom**
Podaci se čitaju iz NgForm objekta	Podaci se čitaju direktno iz modela
Pristup vrednostima preko form.value	Pristup vrednostima preko model objekta
NgForm se prosleđuje metodi onSubmit(form)	NgForm se ne koristi u metodi
Pogodno za proveru statusa forme	Pogodno za rad sa poslovnim modelom
Fokus na formu	Fokus na podatke

Validacija u TDF

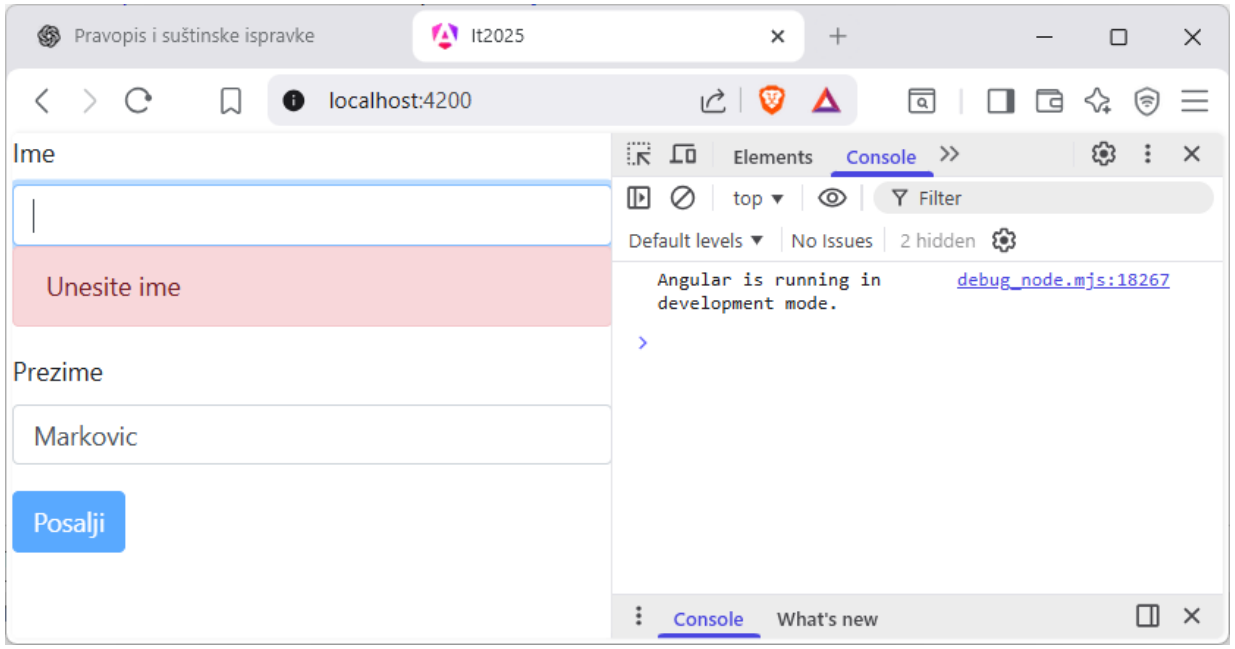
```
<div class="form-group">
<label>Ime</label>
<input type="text" name="ime" class="form-control"
        [(ngModel)]="model.ime" #ime="ngModel"
        required minlength="4">


@if (ime.invalid && (ime.dirty || ime.touched)) {
    <div class="alert alert-danger">
    @if (ime.errors?.['required']) { <div>Unesite ime</div> }
    @if (ime.errors?.['minlength']) { <div>Min. 4 karaktera</div> }
    </div>
}
</div>
```

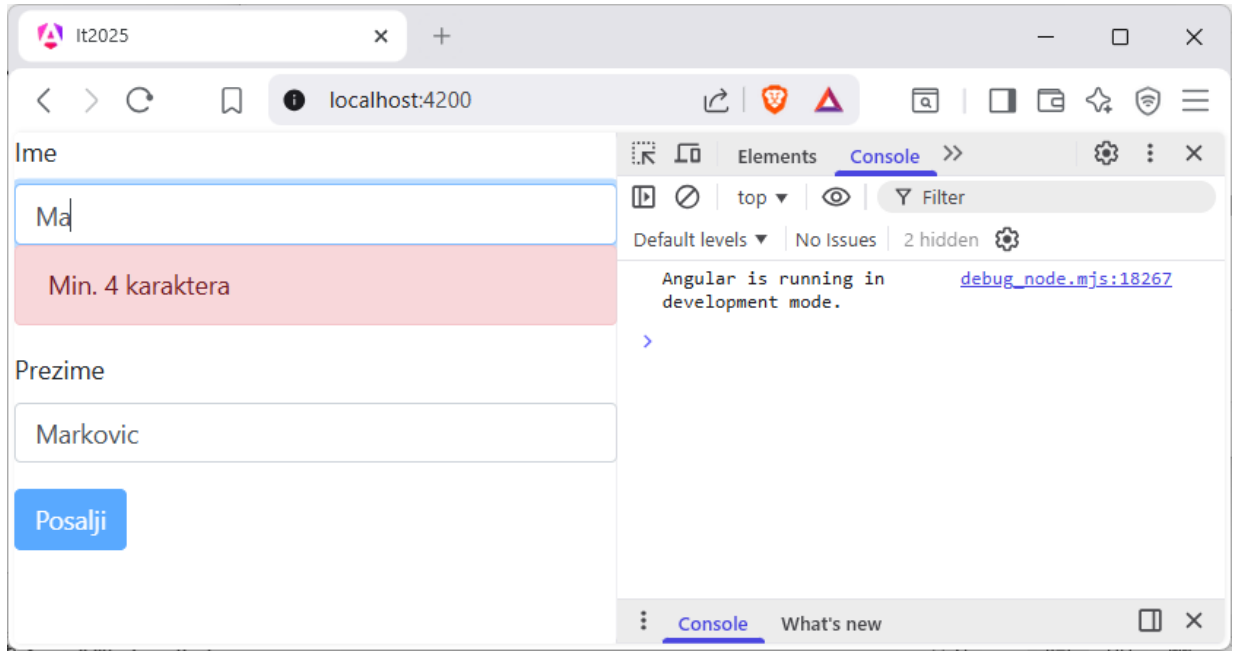
- Lokalna promenljiva **ime** je instance klase **NgModel** koja interno koristi **FormControl**
- dirty – ovaj atribut označava da li je korisnik promenio vrednost u input polju
- touched – ovaj atribut označava da li je korisnik pomerio fokus u input polje i zatim ga napustio (touched)

Validacija u TDF

```
<div class="form-group">
    <label for="ime">Ime</label>
    <input type="text" name="ime" class="form-control" [(ngModel)]="model.ime"
#ime="ngModel" required minlength="4">

    <div *ngIf="ime.invalid && (ime.dirty || ime.touched)" class="alert alert-
danger">
    <div *ngIf="ime.errors?.['required']">
        Unesite ime
    </div>
    <div *ngIf="ime.errors?.['minlength']">
        Ime mora imati najmanje 4 karaktera
    </div>
    </div>
</div>
```

Prikaz validacionih grešaka





Svojstvo valid forme

- Svojstvo **valid** Angular forme vraća true ako su sve kontrole validne
- Suprotno svojstvu valid je svojstvo **invalid**, koje vraća true ako bar jedna kontrola nije validna
- Svojstva valid i invalid imaju i forma i sve kontrole forme
- Forma se šalje korišćenjem događaja **ngSubmit**
- Potrebno je onemogućiti (disable) submit dugme dok forma ne postane validna

Zabrana slanja nevalidnih podataka

```
<button class="btn btn-primary"
        type="submit"
        [disabled]="form1.invalid">
  Posalji
</button>
```

Primer upotrebe TDF formi

```
export class Student {
  constructor(
    public ime: string,
    public prezime: string,
    public pol: string,
    public smer: string
  ) {}
}
```

Primer upotrebe TDF formi

```
export class StudentComponent {
  title = 'primer TDF';
  smerovi = ['Informatika', 'Marketing', 'Menadzment'];
  student: Student = new Student('', '', 'muski', 'Informatika');


  setDefault() {
    this.student.ime = 'Marko';
    this.student.prezime = 'Markovic';
    this.student.pol = 'muski';
    this.student.smer = 'Informatika';
  }


  resetuj() {
    this.student = new Student('', '', 'muski', 'Informatika');
  }
}
```

Metoda onSubmit()

```
  onSubmit() {
    console.log('Ime : ' + this.student.ime);
    console.log('Prezime : ' + this.student.prezime);
    console.log('Pol: ' + this.student.pol);
    console.log('Smer: ' + this.student.smer);
    this.resetuj();
  }
```

Podešavanje tekst polja

```
<form #studentForm="ngForm" (ngSubmit)="onSubmit()">
<div class="form-group mt-3">
 <label>Unesite ime:</label>
  <input name="ime" [(ngModel)]="student.ime"
         #ime="ngModel" required class="form-control">

  @if (ime.invalid && (ime.dirty || ime.touched)) {
    <div>Unesite ime</div>
  }
</div>
<div class="form-group mt-3">
 <label>Unesite prezime:</label>
  <input name="prezime" [(ngModel)]="student.prezime"
         #prezime="ngModel" required class="form-control">

  @if (prezime.invalid && (prezime.dirty || prezime.touched)) {
    <div>Unesite prezime</div>
  }
</div>
  .........
</form>
```

Podešavanje radio inputa

```
<div class="form-group mt-3">
  <label>Pol:</label>
<div>
  <input type="radio" name="pol" value="muski"
         [(ngModel)]="student.pol"
         #pol="ngModel" required >
  Muški
</div>

<div>
  <input type="radio" name="pol" value="zenski"
         [(ngModel)]="student.pol">
  Ženski
</div>

@if (pol.invalid && (pol.dirty || pol.touched)) {
  <div>Odaberite pol</div>
}
</div>
```

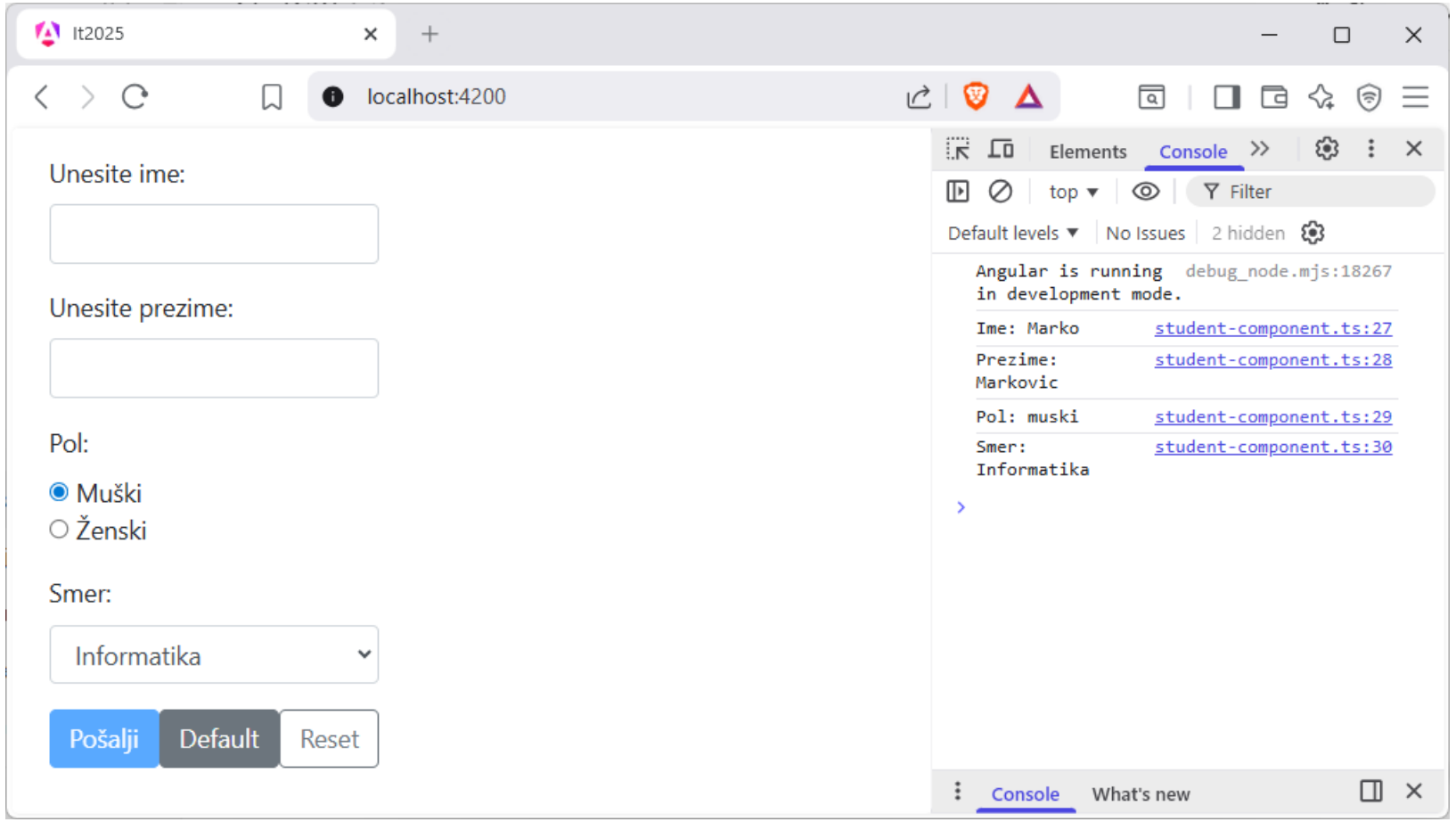
Select kontrola

```
<div class="form-group mt-3">
<label>Smer:</label>
<select name="smer" class="form-control"
        [(ngModel)]="student.smer"
        #smer="ngModel" required>
  <option value="" disabled selected>-- izaberite smer --</option>

  @for (s of smerovi; track s) {
    <option [value]="s">{{ s }}</option>
  }
</select>

@if (smer.invalid && (smer.dirty || smer.touched)) {
  <div>Odaberite smer</div>
}
</div>
```

Podešavanje button elemenata

```
<div class="form-group mt-3">
  <button class="btn btn-primary"
          [disabled]="studentForm.invalid">
    Pošalji
  </button>

  <button type="button"
          class="btn btn-secondary"
          (click)="setDefault()">
    Default
  </button>

  <button type="button"
          class="btn btn-outline-secondary"
          (click)="resetuj()">
    Reset
  </button>
</div>
```

Korisnički interfejs



Pitanje 1

Promena podataka na serveru vrši se korišćenjem sledeće metode HttpClient objekta:

a. get
b. put
c. update

Odgovor: b

Pitanje 2

Unos podataka na serveru vrši se korišćenjem sledeće metode HttpClient objekta:

a. get
b. post
c. update

Odgovor: b

Pitanje 3

Brisanje podataka na serveru vrši se korišćenjem sledeće metode HttpClient objekta:

a. get
b. delete
c. update

Odgovor: b

Pitanje 4

Ulazno polje angular forme predstavlja se klasom:

a. FormControl
b. Control
c. FormField

Odgovor: a

Pitanje 5

Ako se u komponentu importuje FormsModule tada se za svaki tag <form> dodaje direktiva:

a. NgModel
b. NgForm
c. NgField

Odgovor: b

Pitanje 6

Direktiva NgForm:

a. kreira Form instancu koja odgovara formi NgForm
b. kreira FormControl instancu koja odgovara formi
c. kreira FormGroup instancu koja odgovara formi

Odgovor: c

Pitanje 7

Svojstvo dirty instance klase FormControl vraća true ako je:

a. vrednost odgovarajućeg ulaznog polja promenjena
b. vrednost odgovarajućeg ulaznog polja nepromenjena
c. odgovarajuće ulazno polje prazno

Odgovor: a