Servisi

Servisi

- Komponenta je fokusirana na interaktivnu logiku sa korisnikom
- Uzimanje podataka sa servera ne treba da se radi u komponenti
- Servis je klasa koja komunicira sa serverom i obezbeđuje podatke za aplikaciju
- Servis je klasa sa uskom i jasno definisanom funkcionalnošću
- Komponenta koristi servis putem mehanizma koji se zove dependency injection (DI)
- Pomoću anotacije @Injectable() specificira se da je servis raspoloživ za dependency injection

Dependency Injection u Angularu

- Angular ima svoj DI framework
- Zavisnosti su servisi ili objekti koji su potrebni nekoj klasi da bi izvršavala svoju funkciju
- DI je kodni šablon gde klasa traži ubacivanje zavisnosti od ekstenog izvora umesto da sama instancira objekte klase od kojih zavisi
- Traženje zavisnosti od DI frameworka vrši se posredstvom konstruktora zavisne klase

Kreiranje klase

ng g class osoba

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

Fajl podaci.ts unutar app foldera

```
import { Osoba } from './osoba';

export const OSOBE: Osoba[] = [
    new Osoba(1, 'Marko', 'Markovic', 25),
    new Osoba(2, 'Petar', 'Petrovic', 34),
    new Osoba(3, 'Jovan', 'Jovanovic', 27),
    new Osoba(4, 'Milan', 'Mirkovic', 22)
  ];
```

Kreiranje servisa

ng g service osoba

```
import { Injectable } from '@angular/core';

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

  constructor() { }
}
```

```
import { Injectable } from '@angular/core';
import { OSOBE } from './podaci';
import { Osoba } from './osoba';

@Injectable({
  providedIn: 'root'
})
export class OsobaService {
  vratiOsobe():Osoba[]
  {
    return OSOBE;
  }

  constructor() { }
}
```

Singleton servis je servis za koga postoji jedna instance u aplikaciji.
providedIn: ‘root‘ – obezbeđuje singleton servis, znači da će servis biti dostupan na nivou celokupne aplikacije i biće jedinstvena instanca deljena između svih komponenti i servisa u toj aplikaciji

Ubacivanje objekta servisa u konstruktor komponente

```
export class Osobe1Component implements OnInit
{
  title = 'Servisi';
  osobe: Osoba[] =[];

  constructor(private oServis: OsobaService) {
}

  ngOnInit(): void {
    this.osobe = this.oServis.vratiOsobe();
  }

}
```

Šablon komponente

```
<div class="row">
    <div class="col-6">
      <ul>
        <li *ngFor="let osoba of osobe">
          {{ osoba.ime }}
        </li>
      </ul>
    </div>
  </div>
```

- Marko
- Petar
- Jovan
- Milan

Observables

- Observable je objekat koji predstavlja tok podataka i može emitovati niz vrednosti tokom vremena
- Observable može emitovati vrednosti asinhrono
- Vrednosti koje emituje Observable se šalju "observerima" koji prate Observable
- Kada se Observer pretplati na Observable, on postaje "slušalac" koji reaguje na svaku emitovanu vrednost, grešku ili obaveštenje o završetku

Observer

- Observer je objekat koji ima tri moguće funkcije: next, error, i complete
- Funkcija next se poziva se kada Observable emituje novu vrednost
  - Ova funkcija prima vrednost koja je upravo emitovana
- Funkcija error se poziva se kada Observable emituje grešku
  - Ova funkcija prima objekat koji predstavlja grešku
- Funkcija complete se poziva kada Observable završi emitovanje vrednosti

Observables

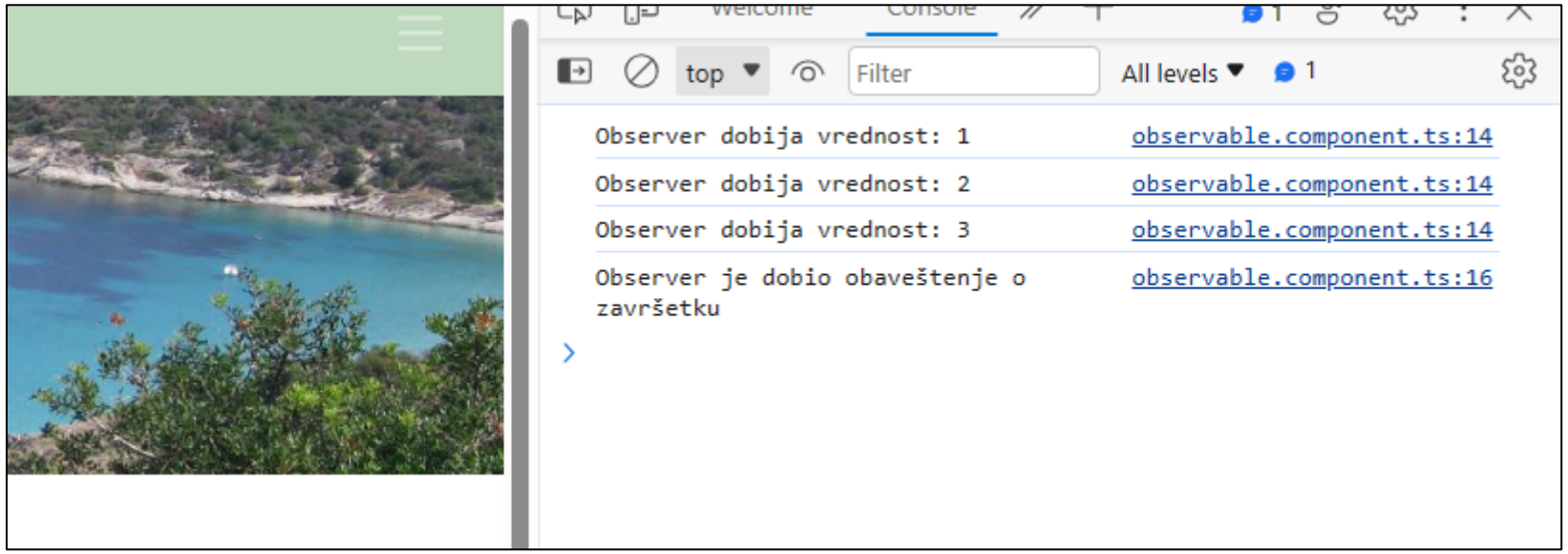
- Web server obično šalje podatke asinhrono i ti podaci se mogu modelovati kao Observable stream
- Observable je sekvenca stavki koje stižu asinhrono tokom vremena
- Observable može imati više pretplatnika i svi pretplatnici bivaju obavešteni kada se stanje observable menja
- Angular koristi biblioteku RxJs (Reactive Javascript) da bi implementirao observable
- HTTP modul koristi observables pri obradi AJAX zahteva i odgovora

Primer Observable i pretplata

```
export class ObservableComponent {
  // Jednostavni observable koji emituje tri vrednosti
  mojObservable = of(1, 2, 3);

  observer = {
    next: (x: number) => console.log('Observer dobija vrednost: ' + x),
    error: (err: string) => console.error('Observer je dobio grešku: ' + err),
    complete: () => console.log('Observer je dobio obaveštenje o završetku')
  };

  pretplata() {
    this.mojObservable.subscribe(this.observer);
  }

}
```

Šablon komponente

```
<button (click)="pretplata()">Observable</button>
```



Modifikacija servisa

```
@Injectable({
  providedIn: 'root'
})
export class Osoba1Service {

  vratiOsobe(): Observable<Osoba[]> {
    return of(OSOBE);
  }
}
```

Pretplata na observable – prosleđivanje metode

```
export class Osobe2Component implements OnInit
{

  osobe: Osoba[] =[];

  constructor(private oServis: Osoba1Service) {
}

  ngOnInit(): void {
    this.oServis.vratiOsobe()
      .subscribe(os => this.osobe = os);
  }

}
```

Komunikacija sa udaljenim serverom

Angular HttpClient API

- Klasa HttpClient omogućava asinhronu komunikaciju sa udaljenim serverom
- Nalazi se u paketu @angular/common/http
- **HttpClientModule** se importuje u koreni modul aplikacije
- Ovaj modul se ubacuje u imports niz glavnog modula aplikacije

```
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Klasa HttpClient

- Klasa HttpClient je tzv. injectable klasa i ubacuje se u klasu servisa posredstvom njegovog konstruktora
- Klasa HttpClient ima metode kojima se generišu http zahtevi ka serveru
- Podatke za aplikaciju najčešće obezbeđuje web api
- Metoda get() generiše GET zahtev (čitanje podataka)
- Metoda post() generiše POST zahtev (kreiranje novih podataka)
- Metoda put() generiše PUT zahtev (promena postojećih podataka)
- Metoda delete() generiše DELETE zahtev (brisanje postojećih podataka)

Angular In-Memory Web API

- Ovaj modul simulira REST (REpresentational State Transfer) API back-end

```
npm i angular-in-memory-web-api
```

Kreiranje baze podataka u memoriji

ng g class baza

```
import { InMemoryDbService } from 'angular-in-memory-web-api';
import { Osoba } from './osoba';

export class Baza implements InMemoryDbService {
    createDb() {
      const osobe: Osoba[] = [
                new Osoba(1, 'Marko', 'Markovic', 25),
                new Osoba(2, 'Petar', 'Petrovic', 34),
                new Osoba(3, 'Jovan', 'Jovanovic', 27),
                new Osoba(4, 'Milan', 'Mirkovic', 22)
              ];

      return {osobe};
    }
}
```

'api/osobe' adresa web api -ja

Konfigurisanje glavnog modula aplikacije

```
import { Baza } from './baza';
import { InMemoryWebApiModule } from 'angular-in-memory-web-api';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    InMemoryWebApiModule.forRoot(Baza)
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

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 Osoba2Service {

  constructor(private http: HttpClient) { }

  private apiUrl = '/api/osobe';

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

GET metode servisa za komunikaciju sa web api aplikacijom

```
vratiOsobe(): Observable<Osoba[]> {
    return this.http.get<Osoba[]>(this.apiUrl)
      .pipe(catchError(this.errorHandler));
  }

  vratiOsobu(id: number): Observable<Osoba> {
    const url = `${this.apiUrl}/${id}`;
    return this.http.get<Osoba>(url).pipe(
      catchError(this.errorHandler)
    );
  }
```

pipe() funkcija kao argumente uzima funkcije koje treba da kombinuje
catchError () funkacija hvata greške

Obrada grešaka u http komunikaciji

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

throwError funkcija iz RxJS biblioteke kreira Observable koji će emitovati grešku
Anonimna funkcija vraća samu grešku kao vrednost koja će biti emitovana kroz Observable.

Komponenta – poziv get metode vratiOsobe()

```
export class Osobe3Component 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: Osoba2Service) {
  }
  ngOnInit(): void {
    this.prikaziOsobe();
  }
  prikaziOsobe(): void {
    this.oServis.vratiOsobe()
      .subscribe({
        next: os => this.osobe = os,
        error: grr => console.log('Greska: ' + grr)
      });
  }
```

Komponenta – metoda prikaziOsobu

```
prikaziOsobu(): void {
    this.oServis.vratiOsobu(this.idOsobe)
      .subscribe({
        next: os => this.odabranaOsoba = os,
        error: grr => {
          console.log('Greska: ' + grr);
          this.odabranaOsoba = new Osoba(0, '', '', 0);
        }
      });
  }
```

Šablon komponente za prikaz podataka-1

```
<div class="row">
    <div class="col-6">
        <ul>
            <li *ngFor="let osoba of osobe">
                {{osoba.id}} {{osoba.ime}} {{osoba.prezime}}
            </li>
        </ul>
    </div>

    <div class="col-6" *ngIf="odabranaOsoba.id !== 0">
        <p>
            {{odabranaOsoba.ime}} {{odabranaOsoba.prezime}}
        </p>
    </div>
</div>
```

Šablon komponente za prikaz podataka-1

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

        <div class="form-group" >
            <input type="text" [(ngModel)]="idOsobe" placeholder="Unesi ID osobe" class="form-
control"> <br>
            <button class="btn btn-secondary" (click)="prikaziOsobu()">Prikazi</button> <br>
        </div>
    </div>
</div>
```

Korisnički interfejs

POST metoda servisa za komunikaciju sa web api aplikacijom

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

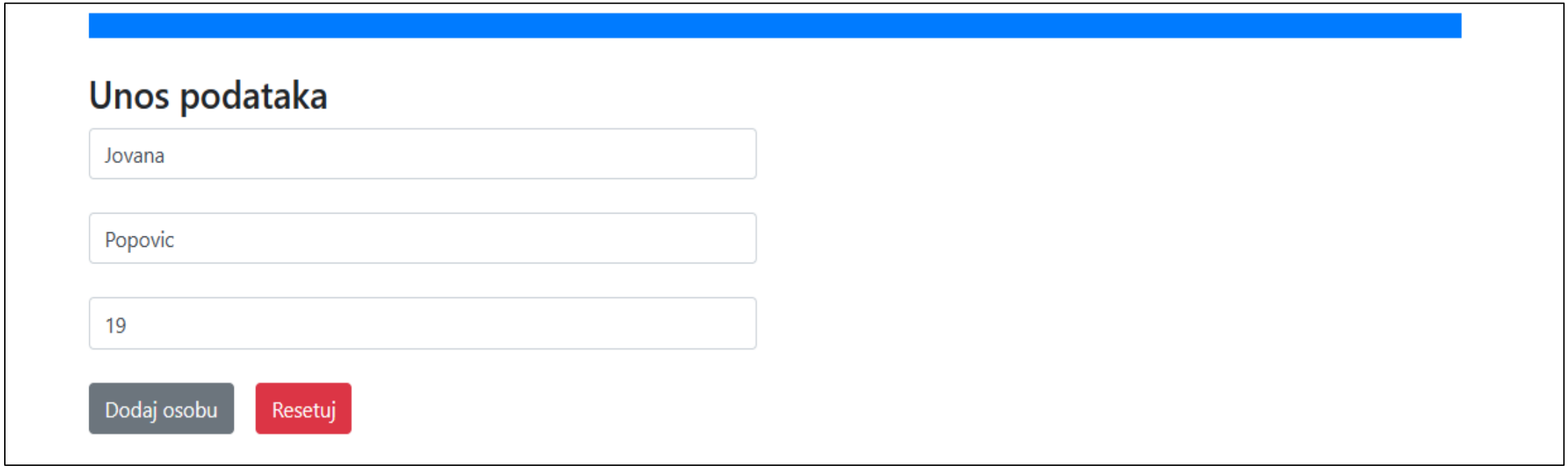
Komponenta – unos podataka

```
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- 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 servisa za komunikaciju sa web api aplikacijom

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

Komponenta – brisanje podataka

```
obrisiOsobu()  {
      this.oServis.obrisiOsobu(this.idOsobe).subscribe({
       next: os => {
          console.log(os);
          this.prikaziOsobe();
        },
        error: grr => console.log('Greska: ' + grr)
      })
 }
```

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>
```

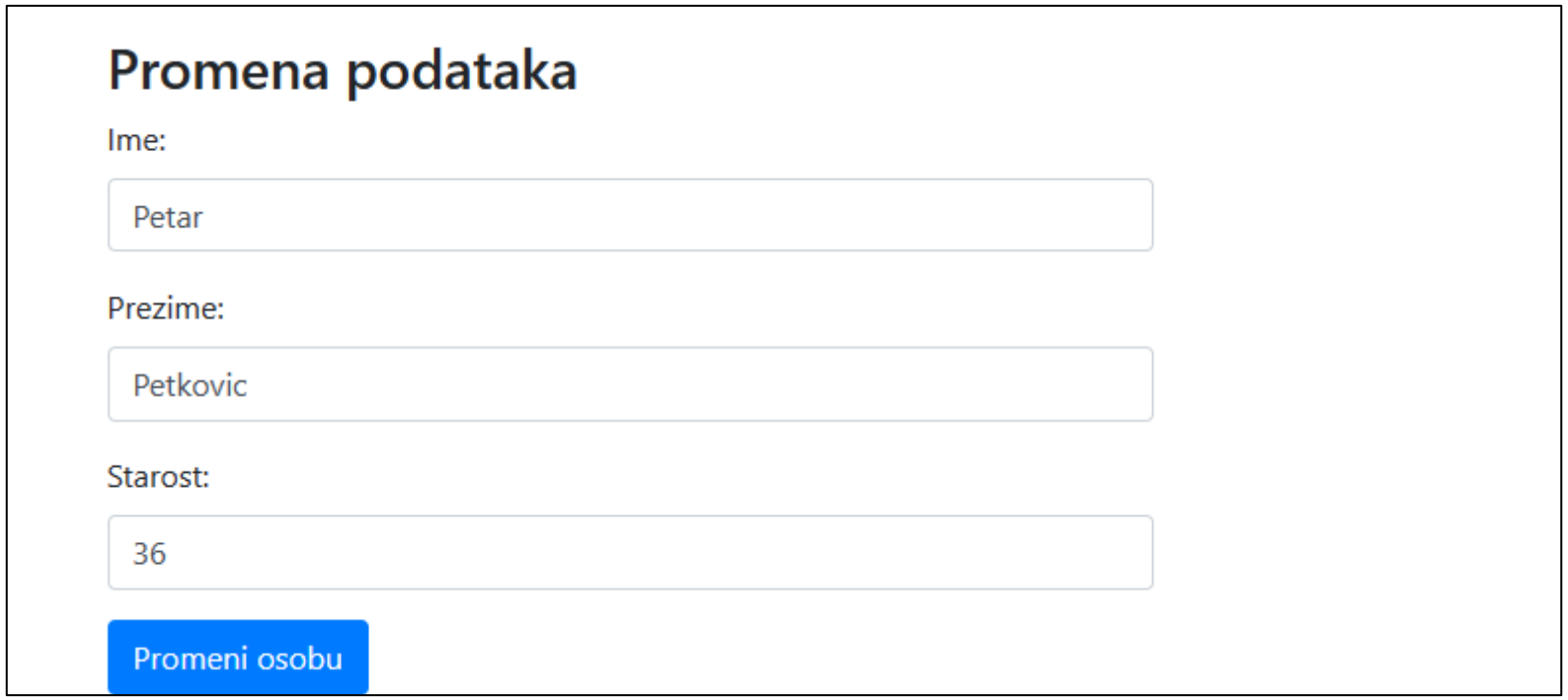
PUT metoda servisa za komunikaciju sa web api aplikacijom

```
promeniOsobu(os: Osoba): Observable<Osoba> {
      const url = `${this.apiUrl}`;
      return this.http.put<Osoba>(url, os)
      .pipe(
        map(() => os),
        catchError(this.errorHandler)
      );
    }
```

Komponenta -promena podataka

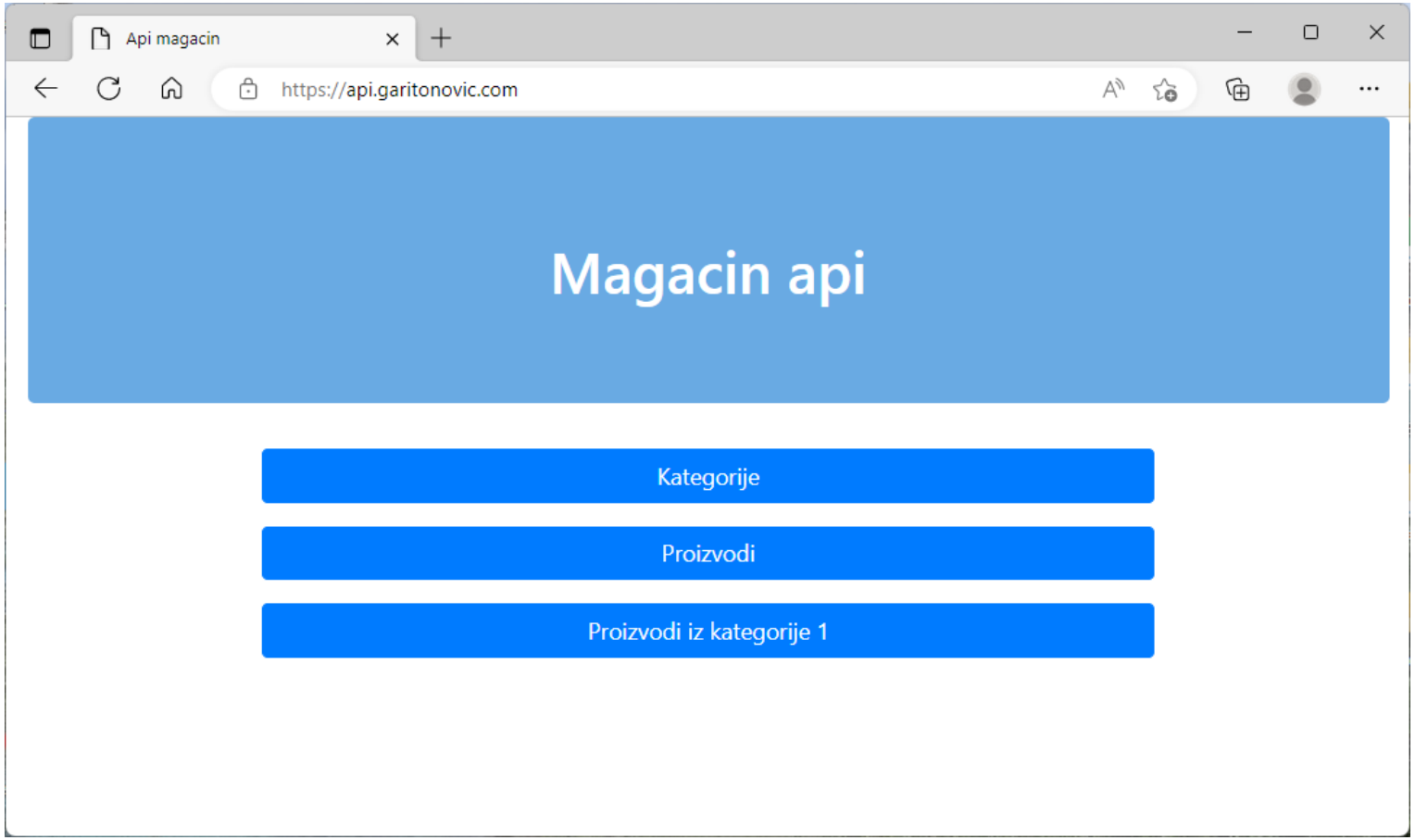
```
promeniOsobu() {
   this.oServis.promeniOsobu(this.odabranaOsoba).subscribe({
     next: novaOsoba => {
       console.log('Osoba je uspešno promenjena:', novaOsoba);
       this.prikaziOsobe();
     },
     error: greska => console.log('Greška prilikom promene osobe:', greska)
   });
 }
```

Komponenta – šablon za promenu



Primer web api -aplikacije

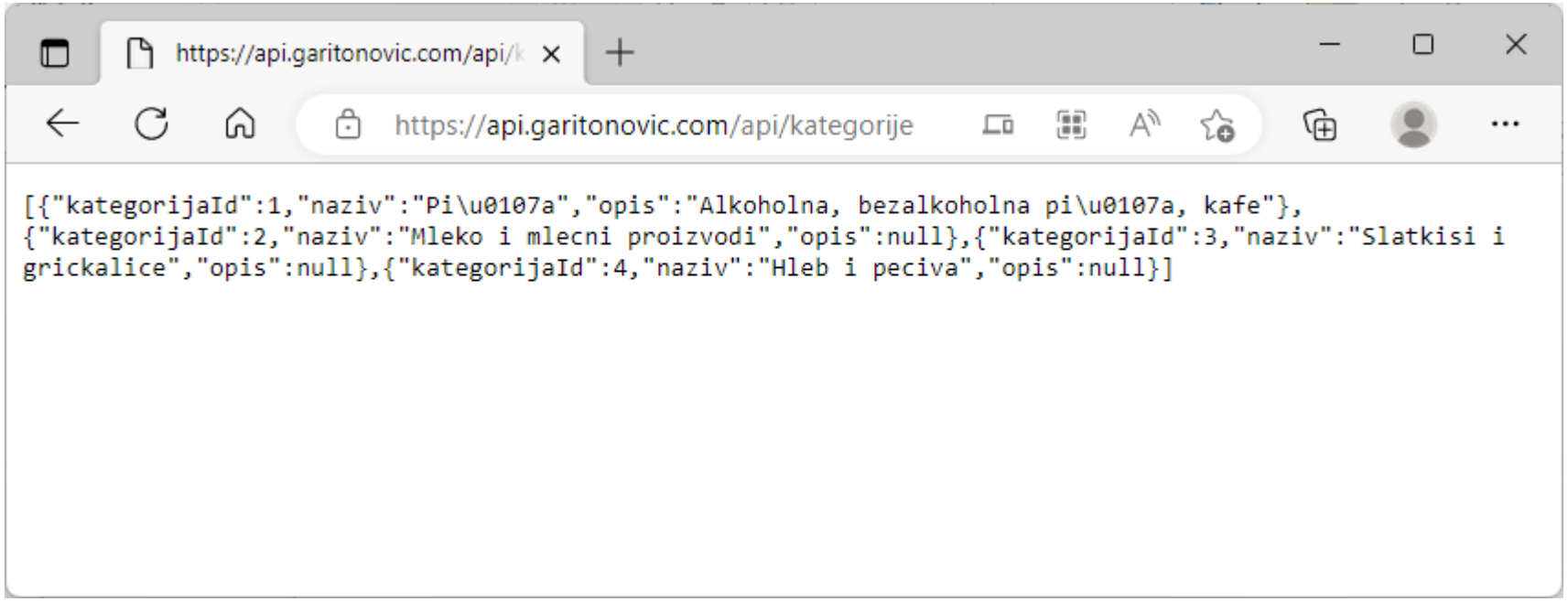
https://api.garitonovic.com/



web api dozvoljava CORS zahteve

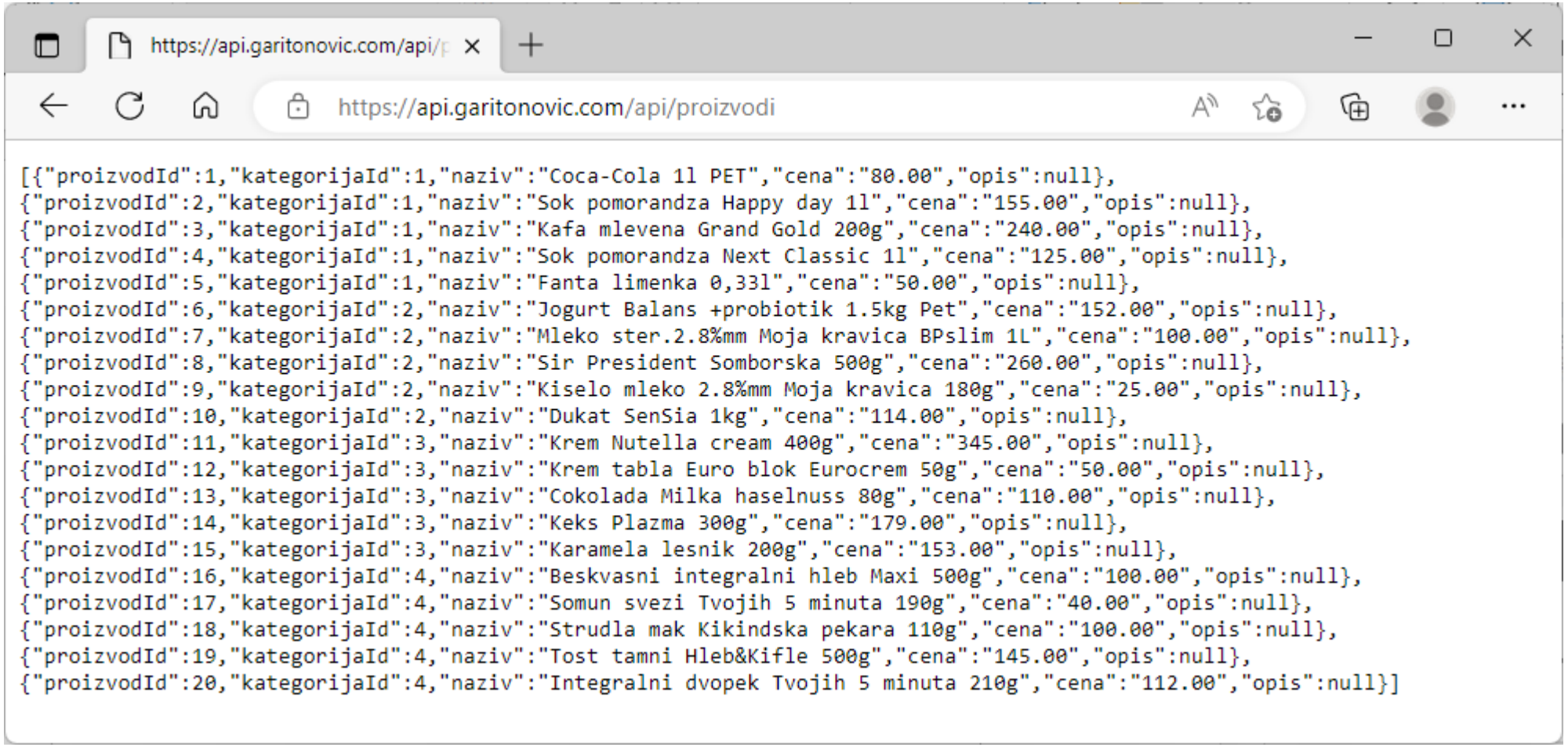
Prikaz kategorija

https://api.garitonovic.com/api/kategorije



Prikaz proizvoda

https://api.garitonovic.com/api/proizvodi




```
[{"proizvodId":1,"kategorijaId":1,"naziv":"Coca-Cola 1l PET","cena":"80.00","opis":null},
{"proizvodId":2,"kategorijaId":1,"naziv":"Sok pomorandza Happy day 1l","cena":"155.00","opis":null},
{"proizvodId":3,"kategorijaId":1,"naziv":"Kafa mlevena Grand Gold 200g","cena":"240.00","opis":null},
{"proizvodId":4,"kategorijaId":1,"naziv":"Sok pomorandza Next Classic 1l","cena":"125.00","opis":null},
{"proizvodId":5,"kategorijaId":1,"naziv":"Fanta limenka 0,33l","cena":"50.00","opis":null},
{"proizvodId":6,"kategorijaId":2,"naziv":"Jogurt Balans +probiotik 1.5kg Pet","cena":"152.00","opis":null},
{"proizvodId":7,"kategorijaId":2,"naziv":"Mleko ster.2.8%mm Moja kravica BPslim 1L","cena":"100.00","opis":null},
{"proizvodId":8,"kategorijaId":2,"naziv":"Sir President Somborska 500g","cena":"260.00","opis":null},
{"proizvodId":9,"kategorijaId":2,"naziv":"Kiselo mleko 2.8%mm Moja kravica 180g","cena":"25.00","opis":null},
{"proizvodId":10,"kategorijaId":2,"naziv":"Dukat SenSia 1kg","cena":"114.00","opis":null},
{"proizvodId":11,"kategorijaId":3,"naziv":"Krem Nutella cream 400g","cena":"345.00","opis":null},
{"proizvodId":12,"kategorijaId":3,"naziv":"Krem tabla Euro blok Eurocrem 50g","cena":"50.00","opis":null},
{"proizvodId":13,"kategorijaId":3,"naziv":"Cokolada Milka haselnuss 80g","cena":"110.00","opis":null},
{"proizvodId":14,"kategorijaId":3,"naziv":"Keks Plazma 300g","cena":"179.00","opis":null},
{"proizvodId":15,"kategorijaId":3,"naziv":"Karamela lesnik 200g","cena":"153.00","opis":null},
{"proizvodId":16,"kategorijaId":4,"naziv":"Beskvasni integralni hleb Maxi 500g","cena":"100.00","opis":null},
{"proizvodId":17,"kategorijaId":4,"naziv":"Somun svezi Tvojih 5 minuta 190g","cena":"40.00","opis":null},
{"proizvodId":18,"kategorijaId":4,"naziv":"Strudla mak Kikindska pekara 110g","cena":"100.00","opis":null},
{"proizvodId":19,"kategorijaId":4,"naziv":"Tost tamni Hleb&Kifle 500g","cena":"145.00","opis":null},
{"proizvodId":20,"kategorijaId":4,"naziv":"Integralni dvopek Tvojih 5 minuta 210g","cena":"112.00","opis":null}]
```

Klasa Proizvod

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

Klasa MagacinService

```
export class MagacinService {
  apiUrl = 'https://api.garitonovic.com/';

  constructor(private http: HttpClient) { }


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

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

}
```

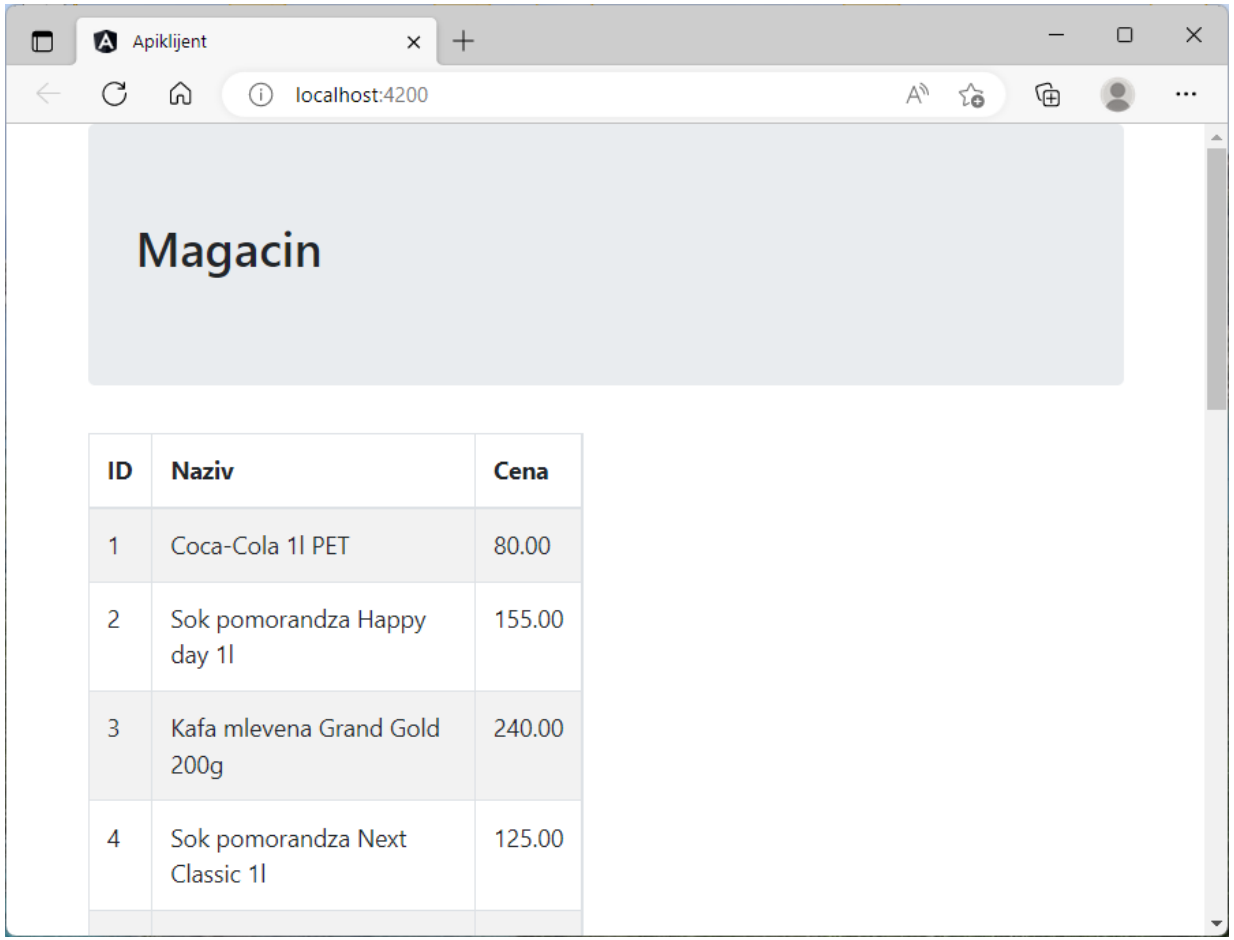
Klasa komponente Proizvodi

```
export class ProizvodiComponent implements OnInit {

  naziv = 'Magacin';
  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();
  }

}
```

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>
          <tr *ngFor="let proizvod of proizvodi">
            <td>{{proizvod.proizvodId}}</td>
            <td>{{proizvod.naziv}}</td>
            <td>{{proizvod.cena}}</td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
```

Prikaz podataka



Pitanje 1

Angular klasa koja komunicira sa serverom i obezbeđuje podatke za aplikaciju naziva se:

a. servis
b. pipe
c. komponenta

Odgovor: a

Pitanje 2

Angular servis opisuje se dekoratorom:

a. @Service
b. @HttpService
c. @Injectable

Odgovor: c

Pitanje 3

Ubacivanje objekta servisa u klasu komponente vrši se :

a. u metodi ngOnInit komponente
b. u konstruktoru komponente
c. u metodi ngDoCheck komponente

Odgovor: b

Pitanje 4

Klasa koja omogućava komunikaciju sa udaljenim api-jem naziva se:

a. HttpClient
b. HttpModule
c. HttpConnect

Odgovor: a

Pitanje 5

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

a. get
b. put
c. update

Odgovor: b

Pitanje 6

Da bi se koristila klasa HttpClient za komunikaciju sa udaljenim serverom u glavni modul aplikacije potrebno je importovati sledeći modul:

a. RemoteModule
b. ClientModule
c. HttpClientModule

Odgovor: c