Kreiranje komponente

Kreiranje klase

```
ng g class osoba
```

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

Kreiranje komponente

- Angular CLI komandom **ng g c imeKomponente** automatski pravi TypeScript fajl, HTML šablon, CSS stilove i dodaje odgovarajući selector
- Kreira se folder sa imenom komonente i smesta u **app** folder aplikacije
- Ime klase komponente uvek počinje velikim slovom, bez obzira kako je naziv unet u komandi
- U verzijama pre Angular 20 klasa komponente je automatski dobijala nastavak Component, dok se u Angularu 20 ovaj nastavak više ne dodaje samostalno

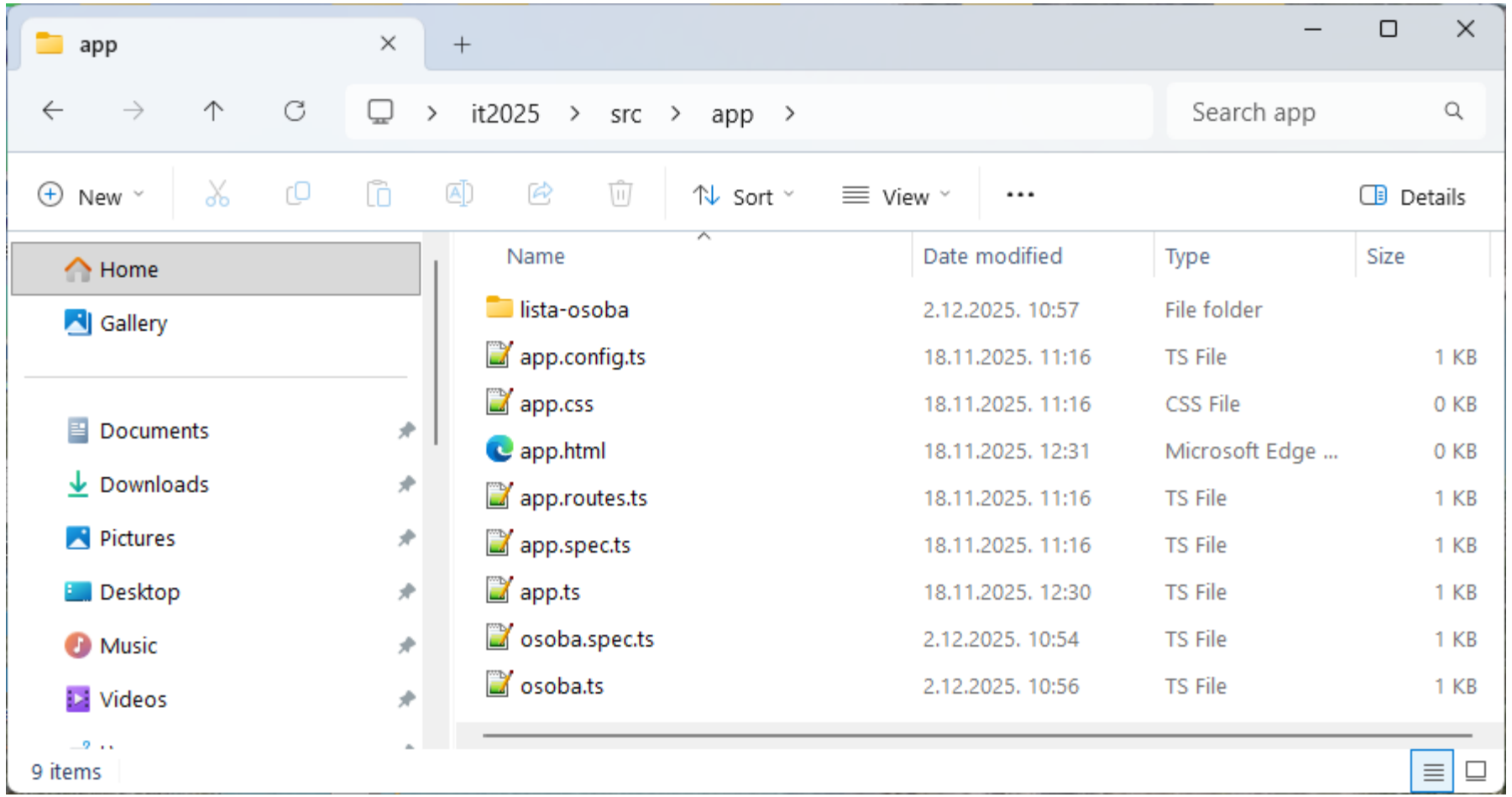
Kreiranje komponente

```
ng g c listaOsoba
```

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


@Component({
  selector: 'app-lista-osoba',
  imports: [],
  templateUrl: './lista-osoba.html',
  styleUrl: './lista-osoba.css',
})
export class ListaOsoba {

}
```

```
<p>lista-osoba works!</p>
```

Folder aplikacije

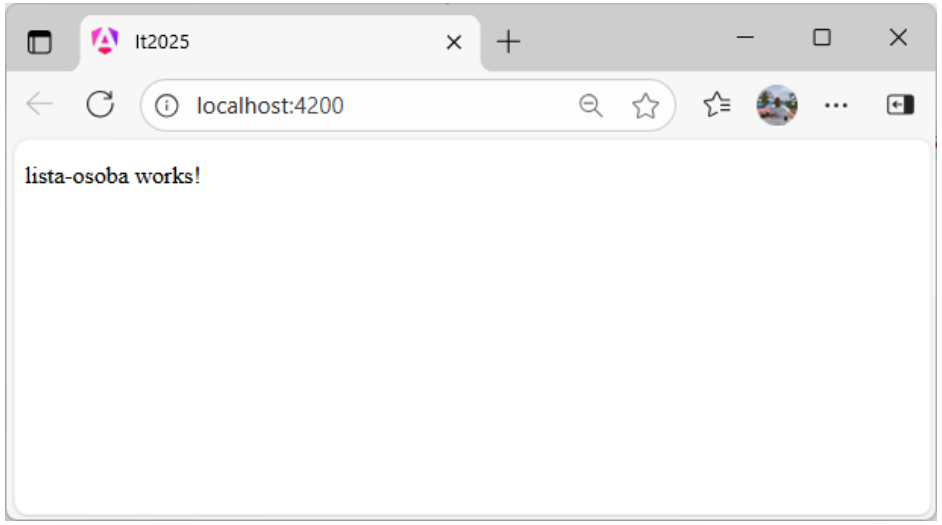


Glavna komponenta

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


@Component({
  selector: 'app-root',
  imports: [],
  templateUrl: './app.html',
  styleUrl: './app.css'
})
export class App {
  title = 'it2025';
}
```

Glavna komponenta - pogled

```
<div class="container mt-3">
  <div class="row">
    <div class="col-md-6">
      <app-lista-osoba></app-lista-osoba>
    </div>
  </div>
</div>
```

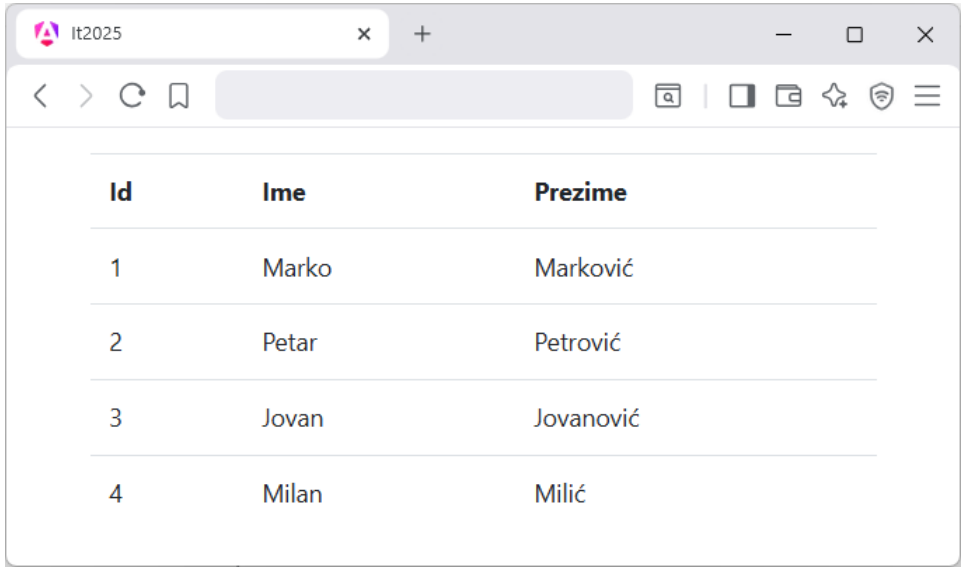


Polje osobe klase ListaOsoba

```
export class ListaOsoba {
osobe: Osoba[] = [
  new Osoba(1, 'Marko', 'Marković'),
  new Osoba(2, 'Petar', 'Petrović'),
  new Osoba(3, 'Jovan', 'Jovanović'),
  new Osoba(4, 'Milan', 'Milić')
];
}
```

Šablon komponente

```
<table class="table">
  <tr>
    <th>Id</th>
    <th>Ime</th>
    <th>Prezime</th>
  </tr>


  @for (osoba of osobe; track osoba.osobaId) {
    <tr>
      <td>{{ osoba.osobaId }}</td>
      <td>{{ osoba.ime }}</td>
      <td>{{ osoba.prezime }}</td>
    </tr>
  }
</table>
```



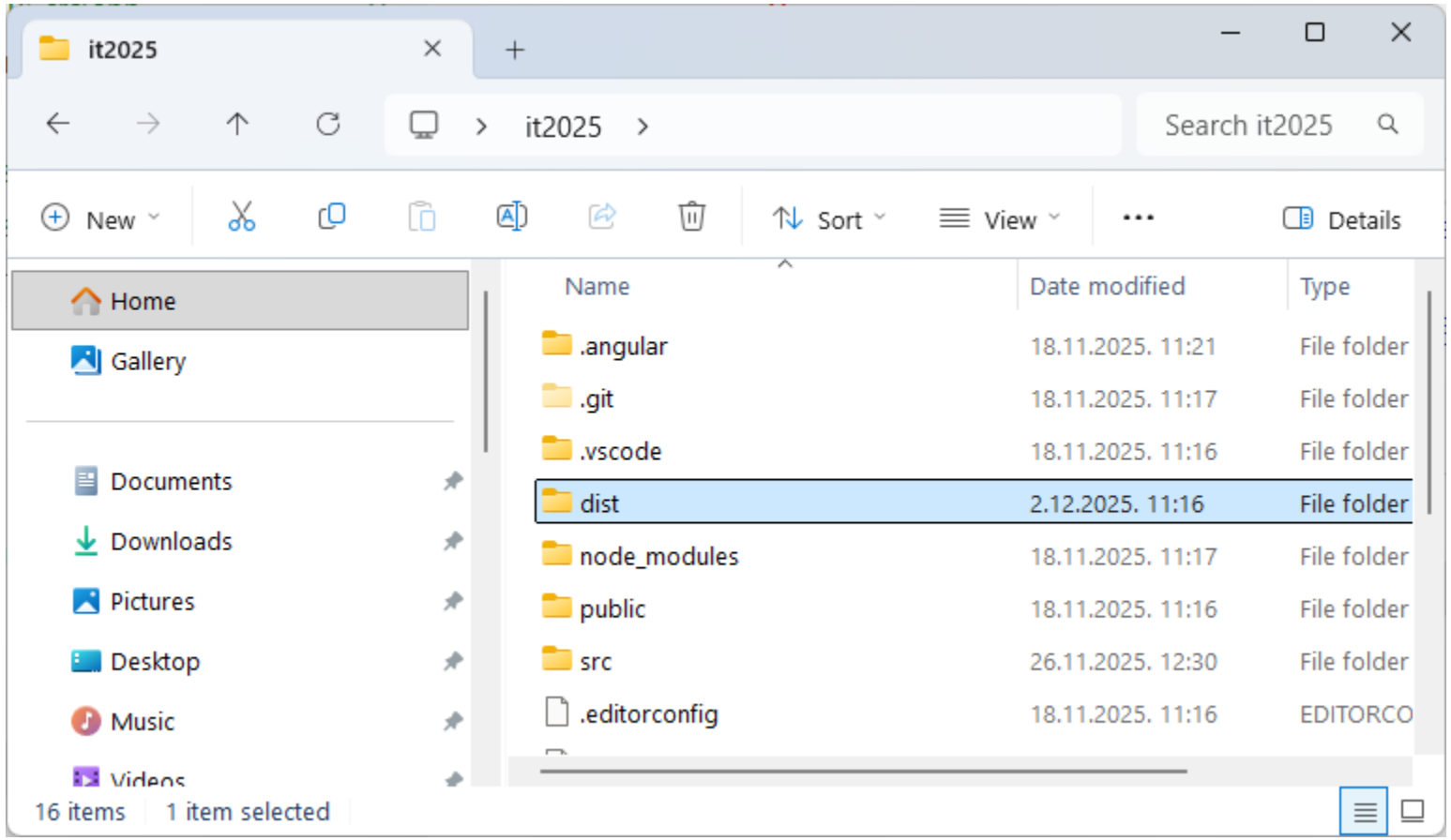
Razmeštanje angular aplikacije

- Unutar fajla index.html (koji je smešten u src folderu) u head delu promeniti base direktivu pre razmeštanja na web server:

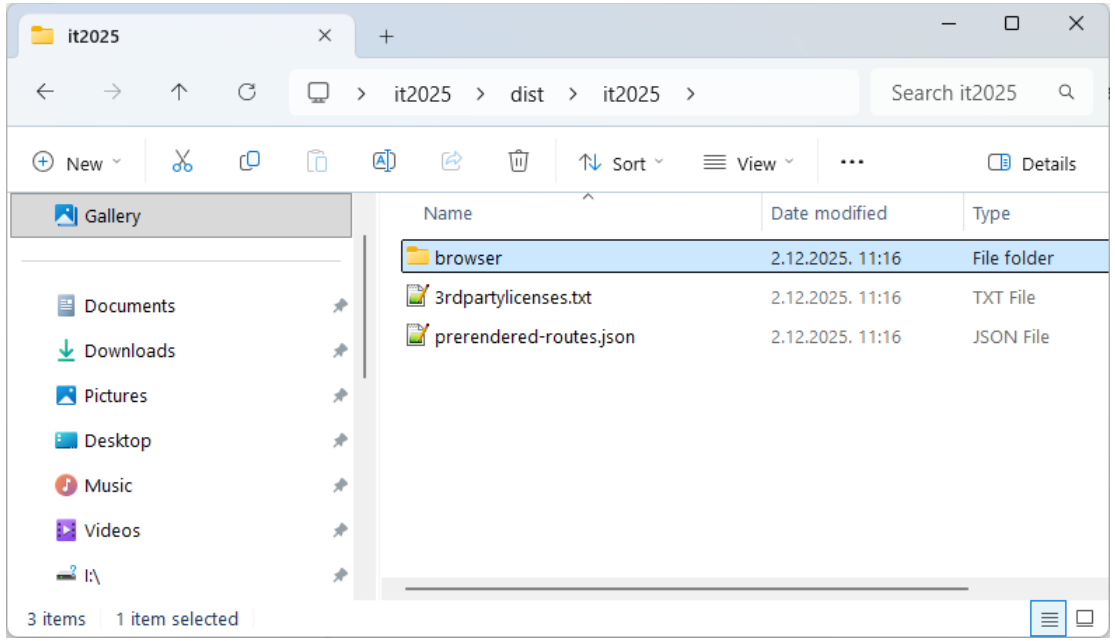
```
<base href="./">
```

- U terminalu kucamo **ng build**
- Kreiran je **dist** folder
- Prekopiramo sadržaj ovog foldera na web server

Folder dist – mesto browser aplikacije

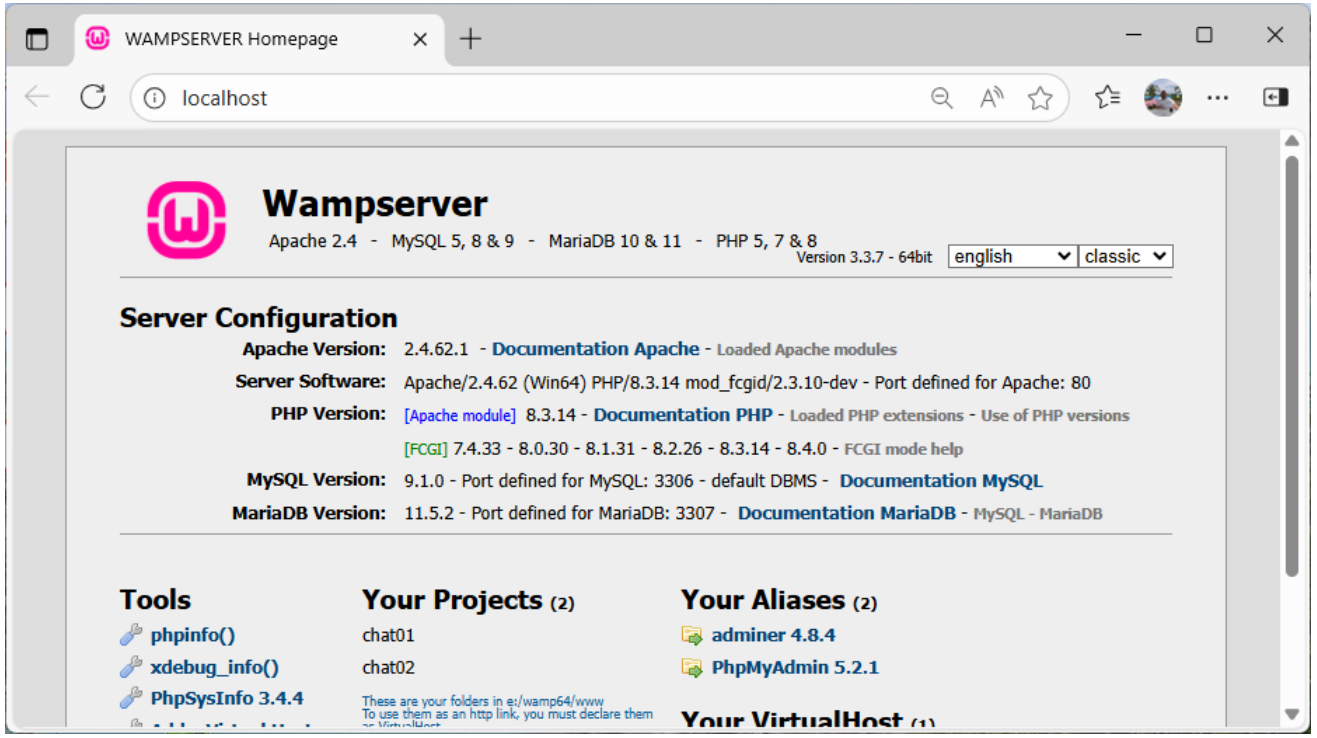


Folder browser

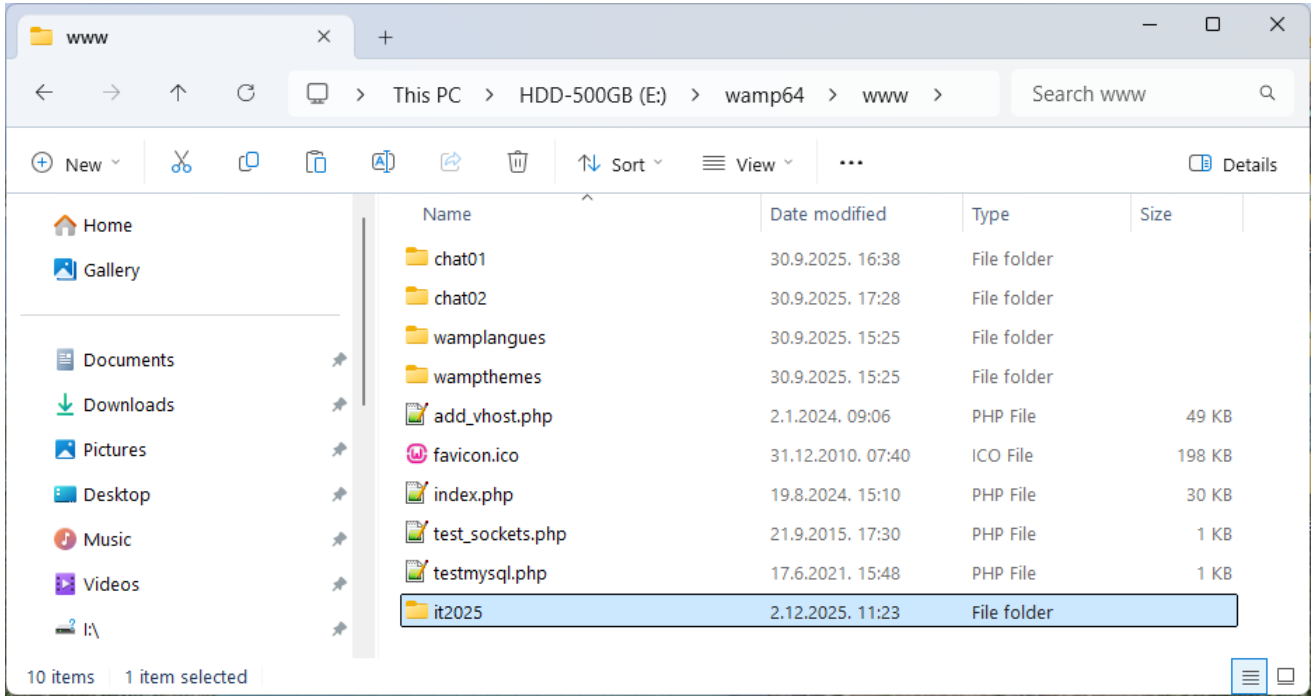


Folder browser se preimenje npr u it2025 i kopira na web server

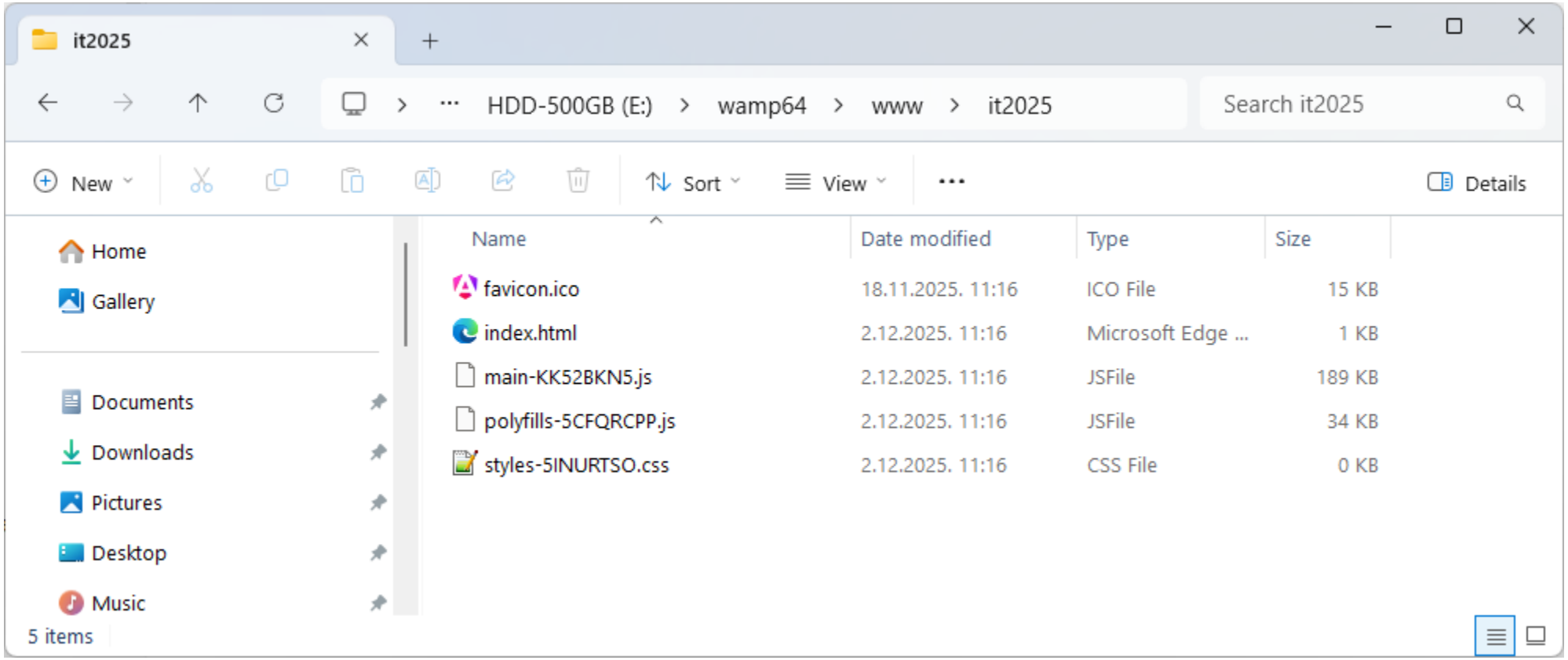
Wampserver


WAMPSERVER Homepage
localhost
Wampserver
Apache 2.4 - MySQL 5, 8 & 9 - MariaDB 10 & 11 - PHP 5, 7 & 8
Version 3.3.7 - 64bit
english
classic
Server Configuration
Apache Version: 2.4.62.1 - Documentation Apache - Loaded Apache modules
Server Software: Apache/2.4.62 (Win64) PHP/8.3.14 mod_fcgid/2.3.10-dev - Port defined for Apache: 80
PHP Version: [Apache module] 8.3.14 - Documentation PHP - Loaded PHP extensions - Use of PHP versions
[FCGI] 7.4.33 - 8.0.30 - 8.1.31 - 8.2.26 - 8.3.14 - 8.4.0 - FCGI mode help
MySQL Version: 9.1.0 - Port defined for MySQL: 3306 - default DBMS - Documentation MySQL
MariaDB Version: 11.5.2 - Port defined for MariaDB: 3307 - Documentation MariaDB - MySQL - MariaDB
Tools
phpinfo()
xdebug_info()
PhpSysInfo 3.4.4
Your Projects (2)
chat01
chat02
These are your folders in e:/wamp64/www
To use them as an http link, you must declare them
Your Aliases (2)
adminer 4.8.4
PhpMyAdmin 5.2.1
Your VirtualHost (1)

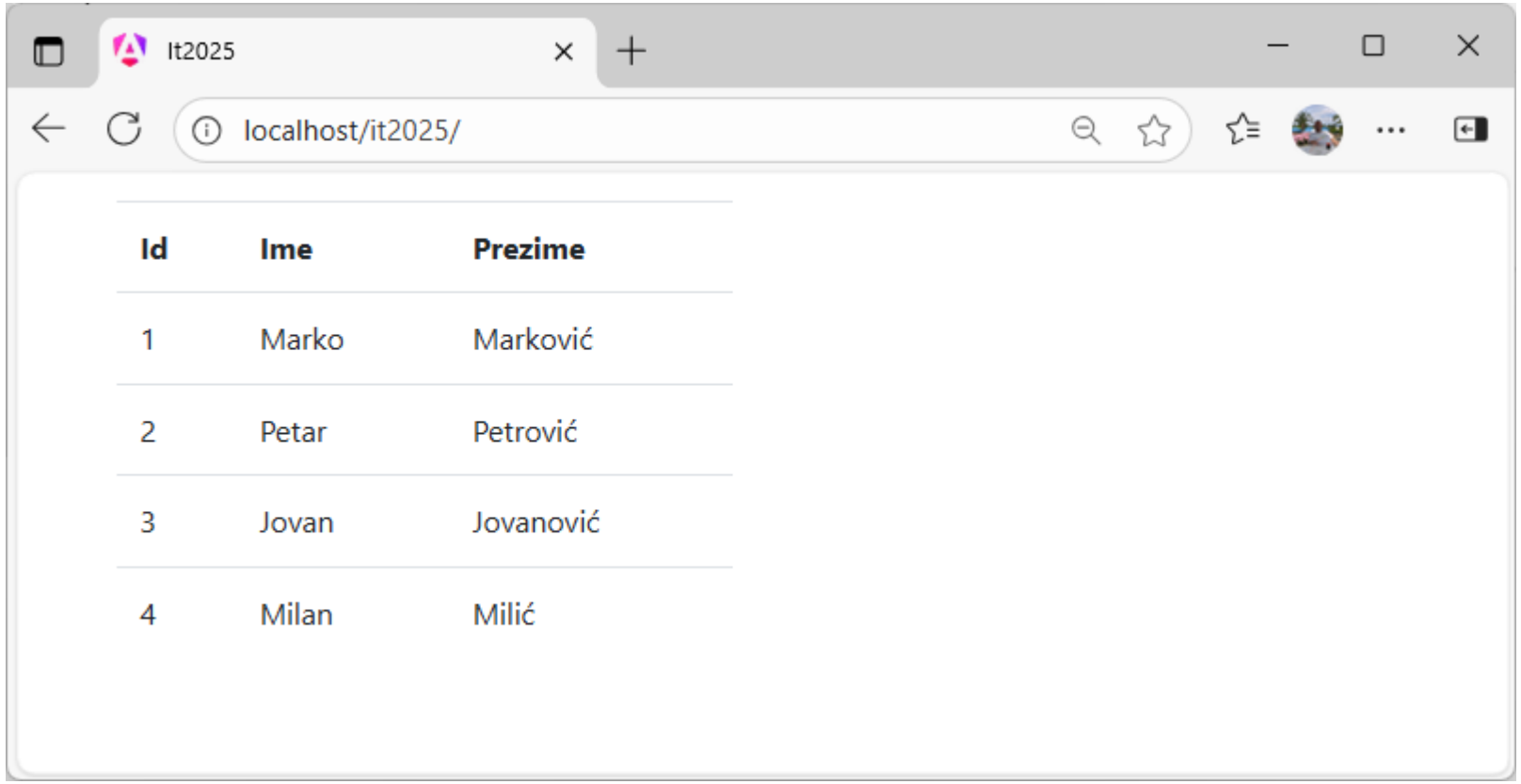
Kopiranje u folder wamp64/www



Folder Angular aplikacije(sajta)



Razmeštanje na lokalni Apache web server



Komunikacija komponenti

Komunikacija komponenti (parent -> child)

- Roditeljska komponenta komunicira sa child komponentom postavljajući njen property
- Child komponenta importuje **@Input** interfejs iz biblioteke **@angular/core**
- Property child komponente koji vrednost dobija od parent komponente označava se **@Input** dekoratorom

```
@Input() childValue: number;
```

Kreiranje child komponente

```
ng g c child
```

```
import { Component, Input } from '@angular/core';


@Component({
  selector: 'app-child',
  imports: [],
  templateUrl: './child.html',
  styleUrl: './child.css',
})
export class Child {
@Input() childValue: number = -1;
}
```

Šablon child komponente

```
<hr>
<h2>Child komponenta</h2>
Tekuća vrednost brojača je: {{ childValue }}
```

Parent komponenta

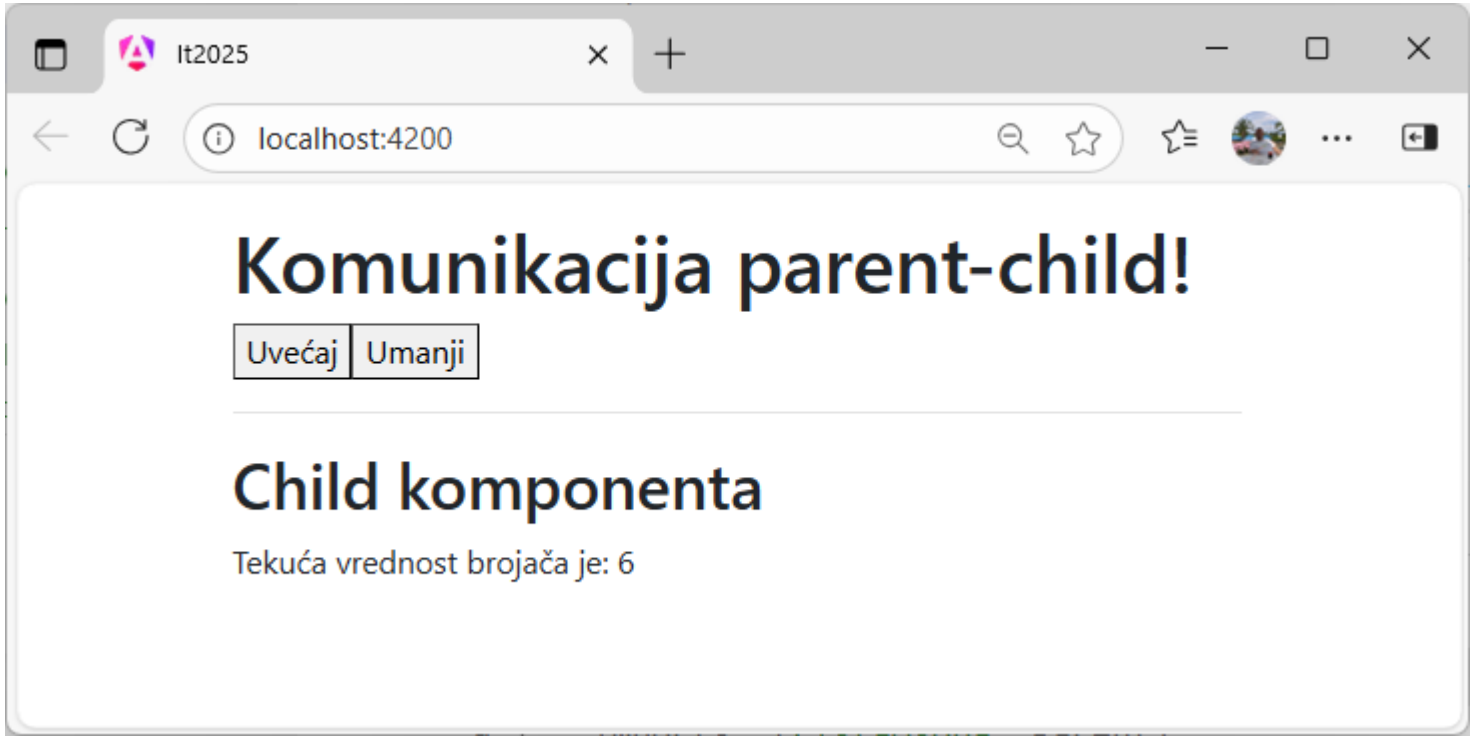
```
import { Component } from '@angular/core';
import { Child } from '../child/child';

@Component({
  selector: 'app-parent',
  imports: [Child],
  templateUrl: './parent.html',
  styleUrl: './parent.css',
})
export class Parent {
  naslov = 'Komunikacija parent-child';
  parentValue = 5;


  uvecaj(): void {
    this.parentValue++;
  }

  umanji(): void {
    this.parentValue--;
  }
}
```

```
ng g c parent
```

Šablon parent komponente

```
<h1>{{ naslov }}!</h1>


<button (click)="uvecaj()">Uvećaj</button>
<button (click)="umanji()">Umanji</button>


<app-child [childValue]="parentValue"></app-child>
```

Komunikacija komponenti



Child komponenta šalje podatake parent komponenti

- U child komponenti potrebno je importovati **Input**, **Output** interfejse i **EventEmitter** klasu iz modula @angular/core
- Da bi generisala događaj, child komponenta treba da definiše svojstvo tipa **EventEmitter<T>**, gde je T tip koji se šalje parent komponenti
- Svojstvo tipa EventEmitter opisuje se dekoratorom @**Output**
- Događaj se generiše pozivom metode **emit**(), kojoj se prosleđuje vrednost za parent komponentu
- Parent komponenta vrši pretplatu na događaj definisan u child komponenti i definiše handlersku funkciju za taj događaj
- Pretplata na događaj vrši se tehnikom **event bindinga**

Child komponenta

```
import { Component, EventEmitter, Output } from '@angular/core';
@Component({
  selector: 'app-child1',
  imports: [],
  templateUrl: './child1.html',
  styleUrl: './child1.css',
})
export class Child1 {
  childValue: number = -1;

  // childValueChange event
  @Output() childValueChange = new EventEmitter<number>();

  uvecaj() {
    this.childValue++;
    this.childValueChange.emit(this.childValue);
  }
  umanji() {
    this.childValue--;
    this.childValueChange.emit(this.childValue);
  }
}
```

Šablon child komponente

```
<hr>
<h2>Child komponenta</h2>
  <button (click)="uvecaj()">Uvecaj</button>
  <button (click)="umanji()">Umanji</button>
<br>
```

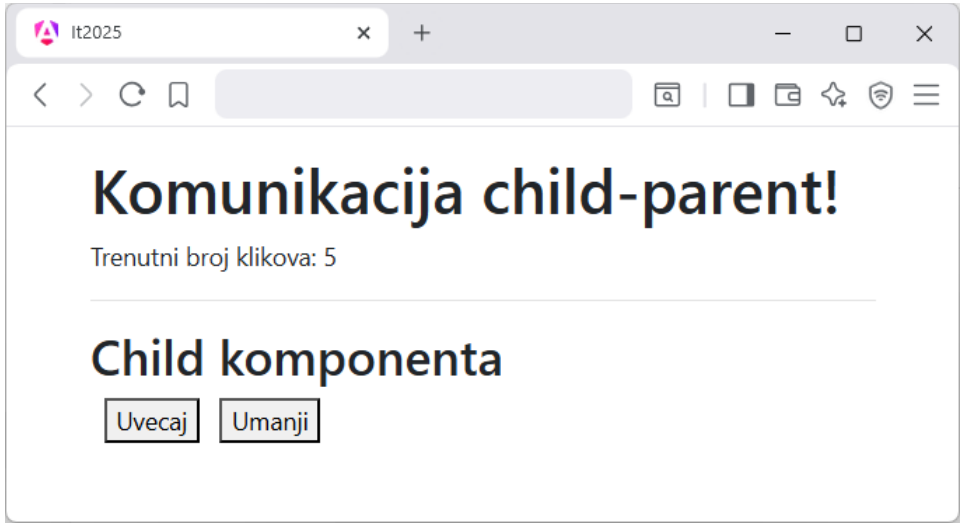
Parent komponenta

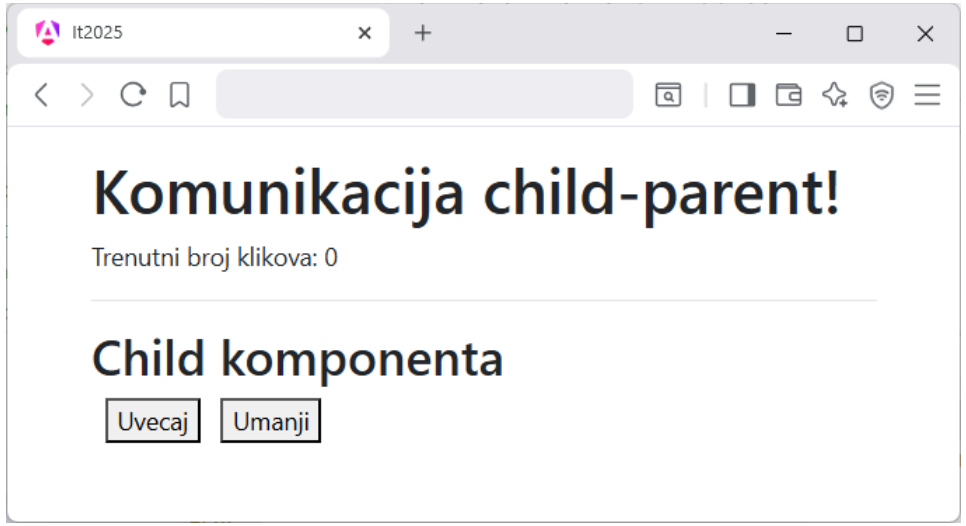
```
export class Parent1 {
  naslov = 'Komunikacija child-parent';
  parentValue = 5;


  childValueChangeHandler(event: number) {
    this.parentValue = event;
  }
}
```

```
<h1>{{ naslov }}!</h1>
<p>Trenutni broj klikova: {{ parentValue }}</p>

<app-child1 (childValueChange)="childValueChangeHandler($event)"></app-child1>
```

Parent komponenta pretplaćena na događaj child komponente

Rezultat prosleđivanja podataka





Primer komunikacije komponenti -primer

ng g class osoba

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

Klasa parent komponente

ng g c osobeComponent

```
export class OsobeComponent {
  title = 'Primer komunikacije komponenti';


  osobe: Osoba[] = [
    new Osoba(1, 'Marko', 'Markovic'),
    new Osoba(2, 'Petar', 'Petrovic'),
    new Osoba(3, 'Jovan', 'Jovanovic'),
    new Osoba(4, 'Milan', 'Mirkovic')
  ];

  selektovanaOsoba: Osoba = new Osoba(-1, '', '');

  selektuj(os: Osoba): void {
    this.selektovanaOsoba = os;
  }
}
```

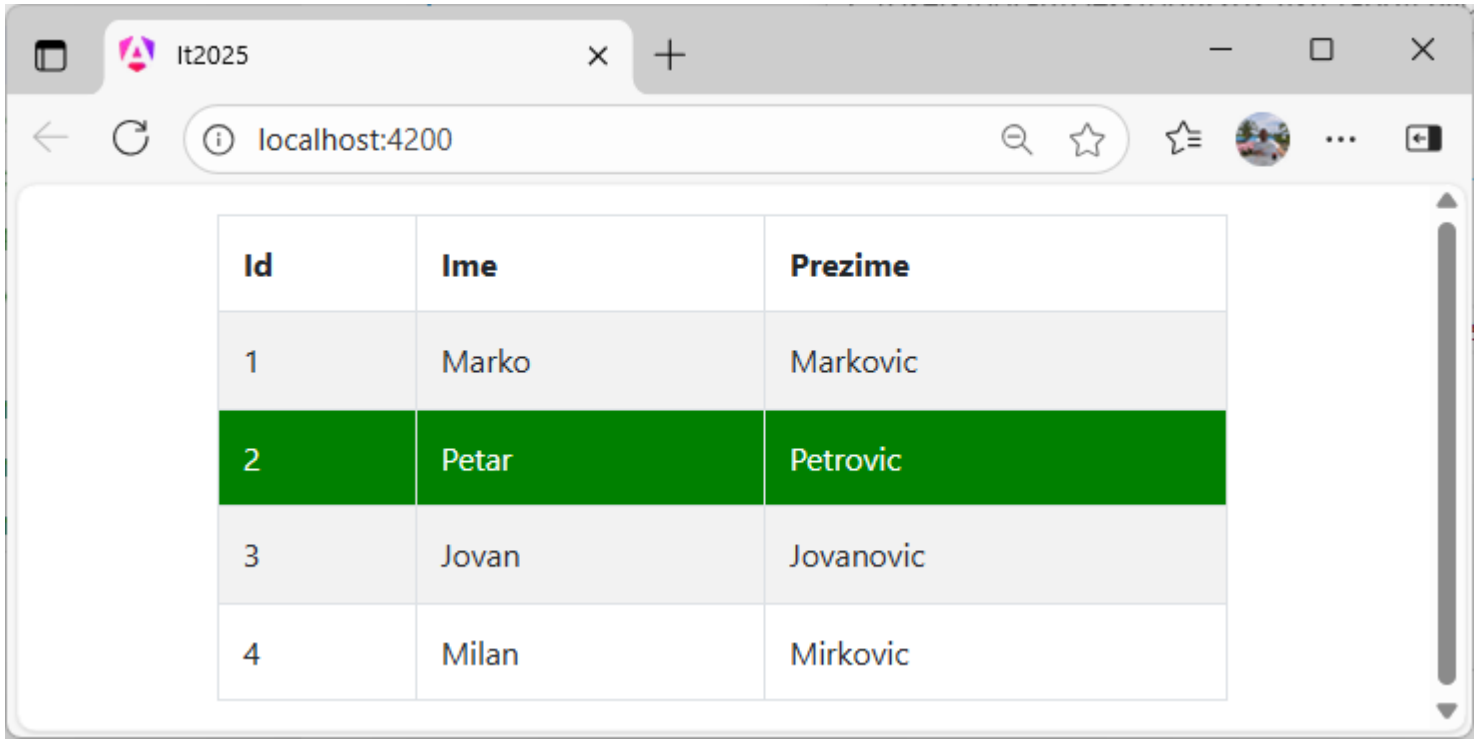
CSS fajl parent komponente

```
.selektujRed {
  background-color: green !important;
  color: white;
}
```

Šablon parent komponente

```
<table class="table table-bordered table-striped">
  <thead>
    <tr><th>Id</th><th>Ime</th><th>Prezime</th></tr>
  </thead>


  <tbody>
    @for (osoba of osobe; track osoba.osobaId) {
      <tr (click)="selektuj(osoba)"
          [class.selektujRed]="osoba === selektovanaOsoba">
        <td>{{ osoba.osobaId }}</td>
        <td>{{ osoba.ime }}</td>
        <td>{{ osoba.prezime }}</td>
      </tr>
    }
  </tbody>
</table>
```

Izgled parent komponente



Child komponenta

```
ng g c osobaDetalji
```

```
export class OsobaDetalji {
  @Input() osoba: Osoba = new Osoba(-1, '', '');
}
```

```
<div class="col-md-4">
  @if (osoba.osobaId !== -1) {
    Id: {{ osoba.osobaId }} <br>
    Ime: {{ osoba.ime }} <br>
    Prezime: {{ osoba.prezime }}
  }
</div>
```

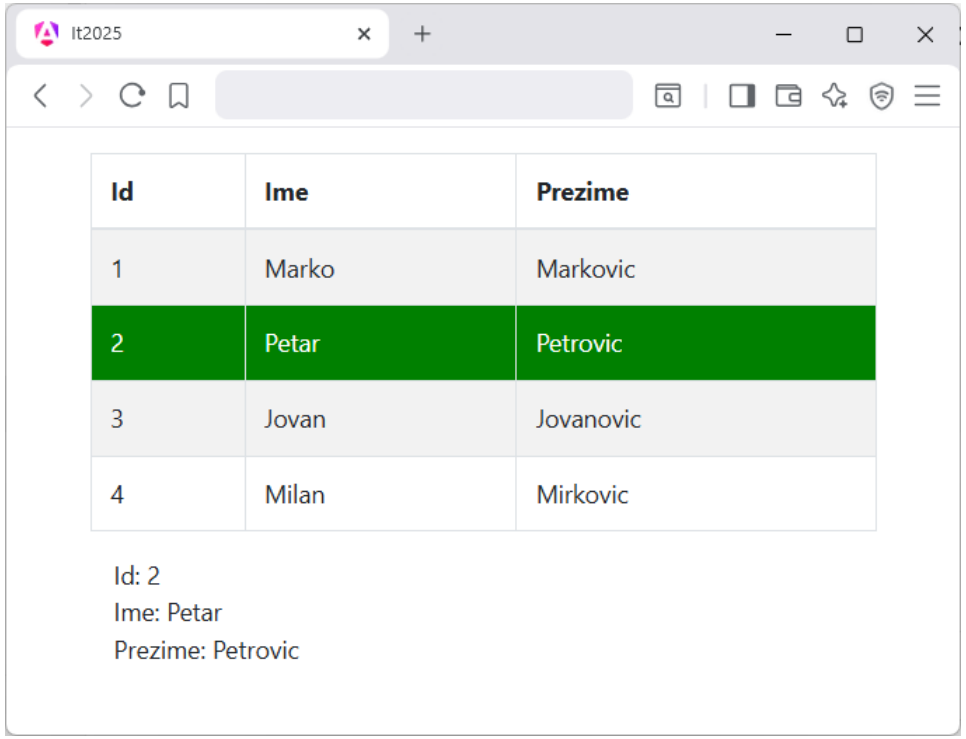
Poziv child komponente iz parent

```
<div class="row">
    <div class="col-md-7">
        <app-osoba-detalji [osoba]="selektovanaOsoba"></app-osoba-detalji>
    </div>
</div>
```

Go to component

```
<table class="table table-bordered table-striped">
  <thead>
    <tr><th>Id</th><th>Ime</th><th>Prezime</th></tr>
  </thead>

  <tbody>
    @for (osoba of osobe; track osoba.osobaId) {
      <tr (click)="selektuj(osoba)" ···
      </tr>

    }
  </tbody>
</table>
<div class="row">
    <div class="col-md-7">
        <app-osoba-detalji [osoba]="selektovanaOsoba"></app-osoba-detalji>
    </div>
</div>
```

Komunikacija komponenti



Pitanje 1

U Angular 20 aplikaciji je potrebno je kreirati komponentu Osobe korišćenjem Angular CLI alata. Koju komandu treba napisati u terminalu:

a. ng g c osobe
b. node g component osobe
c. ng make c Osobe

Odgovor: a

Pitanje 2

Child komponenta ima selektor **app-osoba-detalji** i ulazni property **osoba** koji je tipa klase **Osoba**. Na koji način parent komponenta poziva child komponentu i prosleđuje joj objekat tipa Osoba pod nazivom **selektovanaOsoba**?

a. <app-osoba-detalji [child]=selektovanaOsoba></app-osoba-detalji>
b. <app-osoba-detalji [osoba]="selektovanaOsoba"></app-osoba-detalji>
c. <chilld [emit]="selektovanaOsoba"></child>

Odgovor: b

Pitanje 3

Child komponenta treba da pošalje podatke parent komponenti generisanjem događaja. Da bi se to postiglo u child komponenti se definiše izlazno svojstvo koje je tipa

a. EventEmitter
b. Event
c. Observable

Odgovor: a

Pitanje 4

Property child komponente koji vrednost dobija od parent komponente označava sa dekoratorom:

a. @Input()
b. @Child()
c. @Parent()

Odgovor: a

Pitanje 5

Child komponenta treba da pošalje podatke parent komponenti generisanjem događaja. Da bi se to postiglo u child komponenti se definiše svojstvo tipa EventEmitter koga treba opisati dekoratorom

a. @Output()
b. @Input()
c. @Emmit()

Odgovor: a