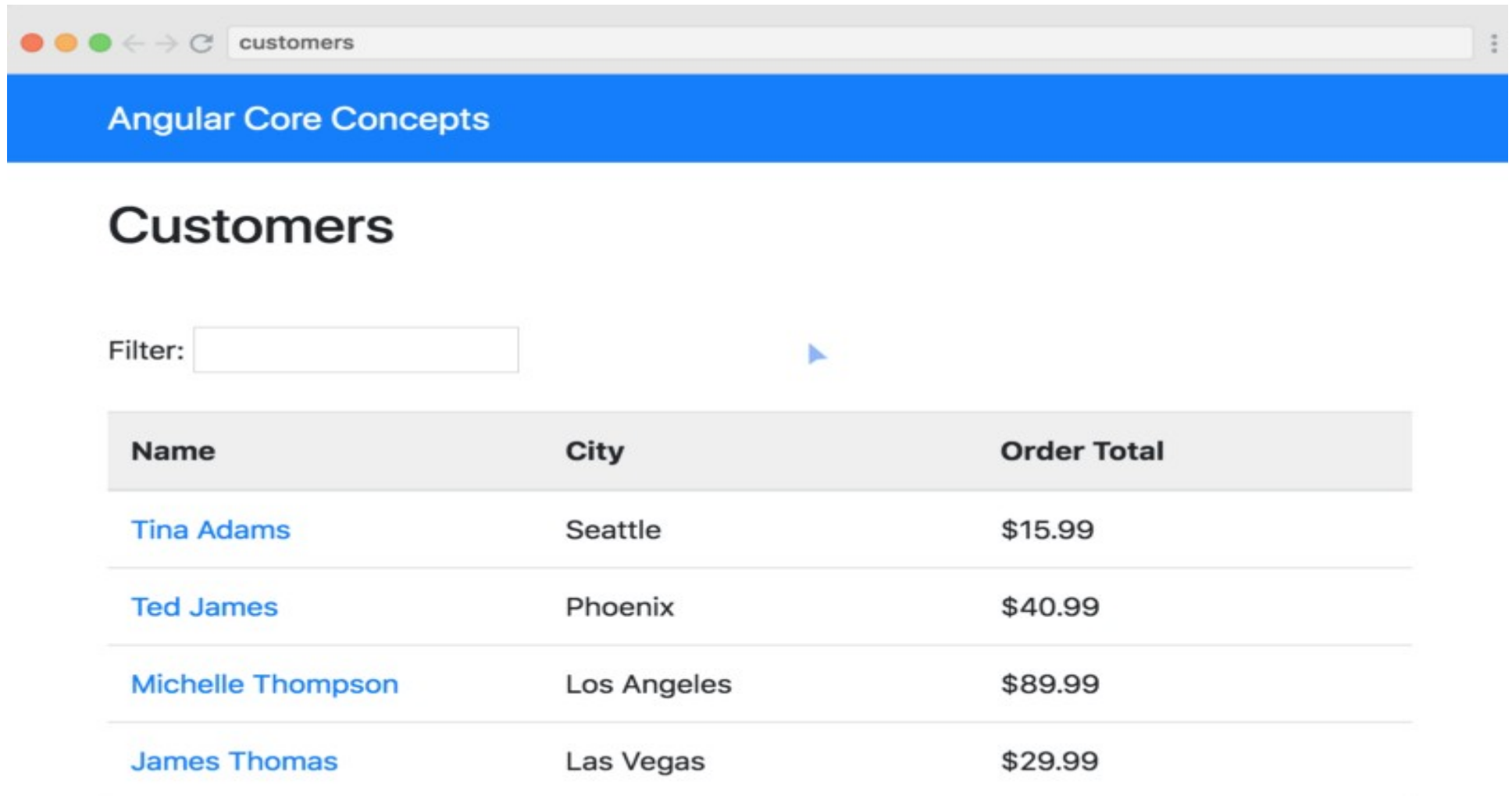


Angular

Co budujemy



customers

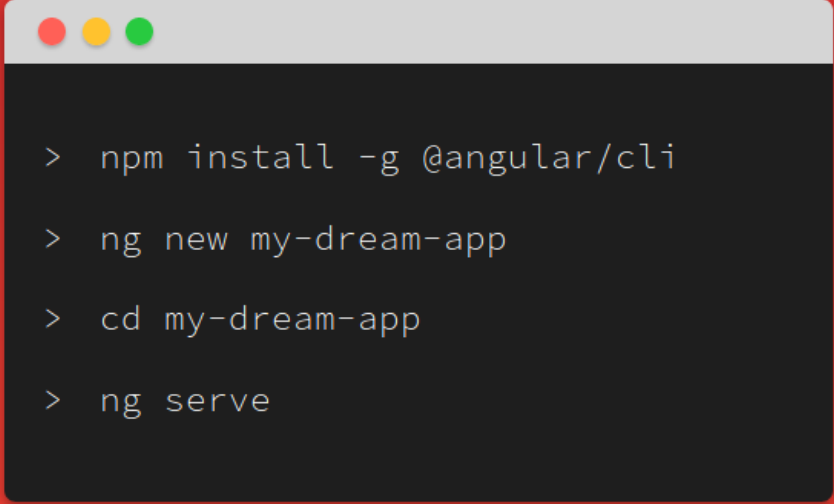
Angular Core Concepts

Customers

Filter:

Name	City	Order Total
Tina Adams	Seattle	\$15.99
Ted James	Phoenix	\$40.99
Michelle Thompson	Los Angeles	\$89.99
James Thomas	Las Vegas	\$29.99

IntelliJ



```
> npm install -g @angular/cli  
> ng new my-dream-app  
> cd my-dream-app  
> ng serve
```

Angular CLI

A command line interface for Angular

GET STARTED

Angular CLI

`ng --version` ▶

`ng --help`

`ng new my-app-name`

`ng generate`

`[component | directive | pipe | service | class | interface | enum | guard]`

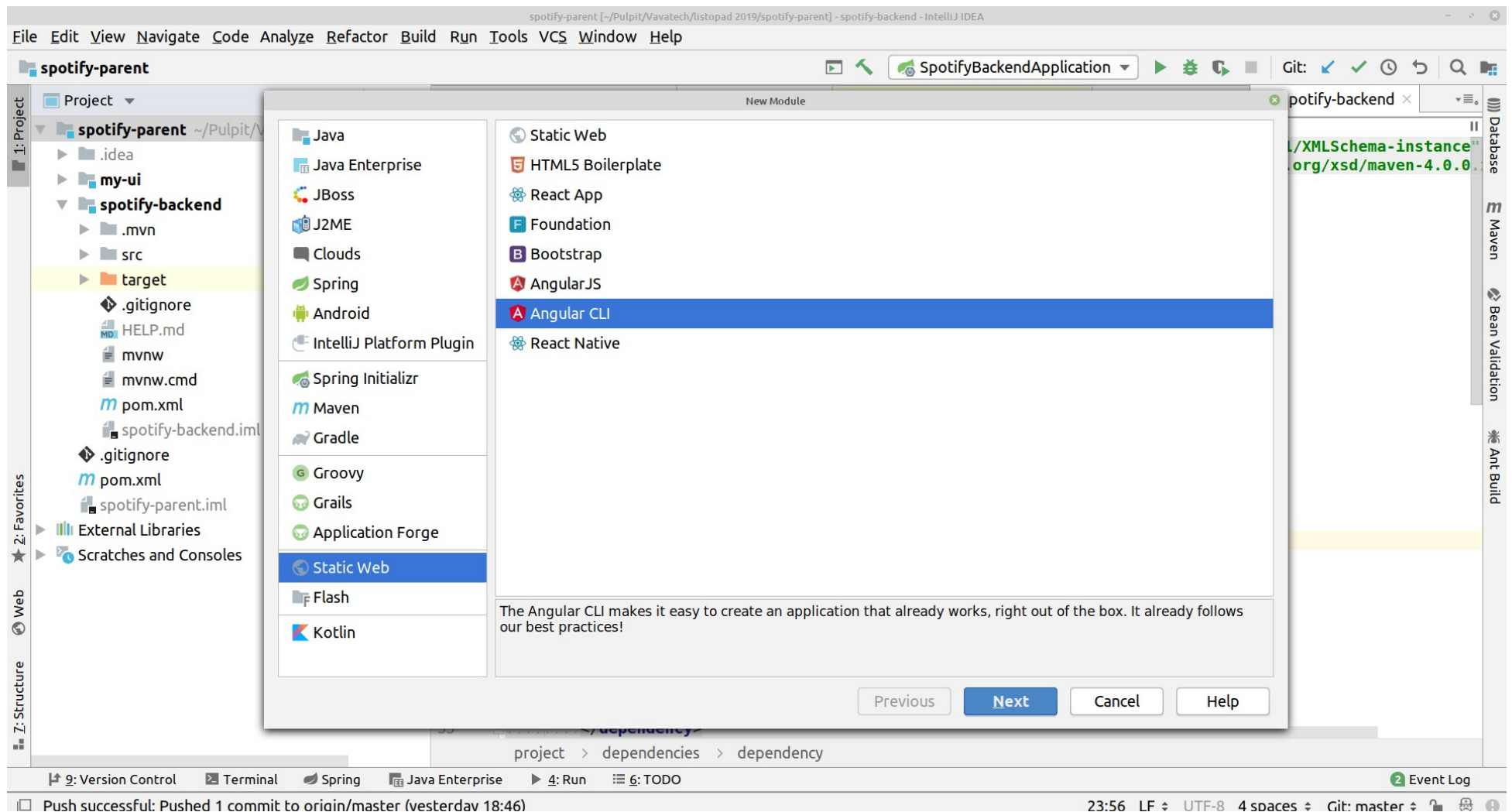
`ng build`

`ng serve`

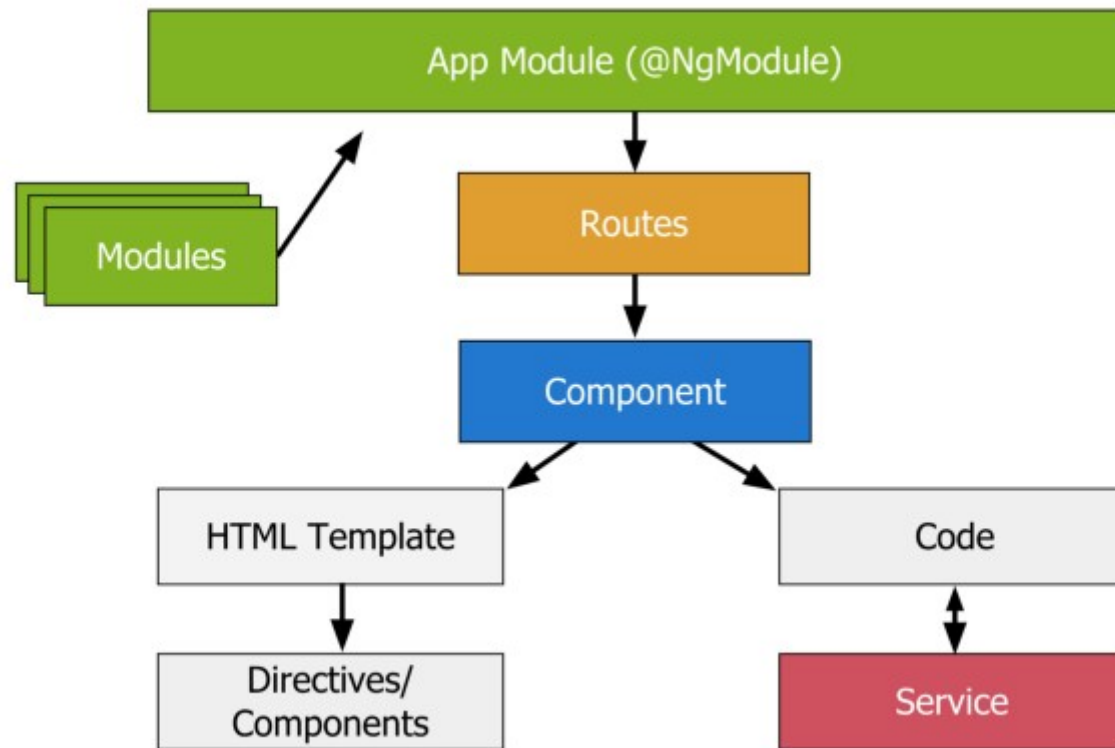
`ng lint`

`ng test`

IntelliJ



Komponenty i moduły



Komponenty i moduły

imports

```
import { Component } from '@angular/core';  
import { DataService } from '../services/data.service';
```

decorators

```
@Component({  
  selector: 'app-customers',  
  templateUrl: './customers.component.html'  
})
```

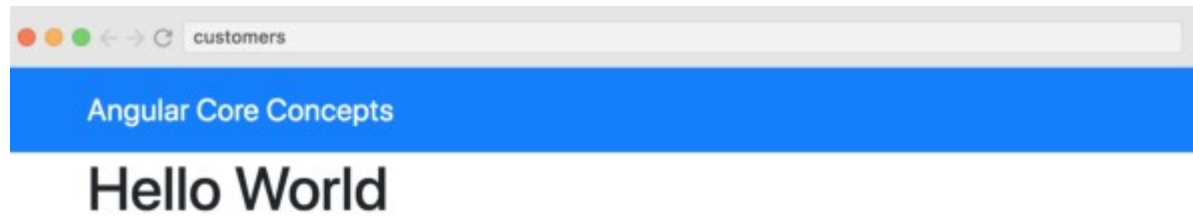
class

```
export class CustomersComponent {  
}
```

App Component

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    template: `
6      <h1>Hello World</h1>
7    `
8  })
9  export class AppComponent implements OnInit {
10
11    constructor() { }
12
13    ngOnInit() {
14
15    }
16
17  }
```


App Component

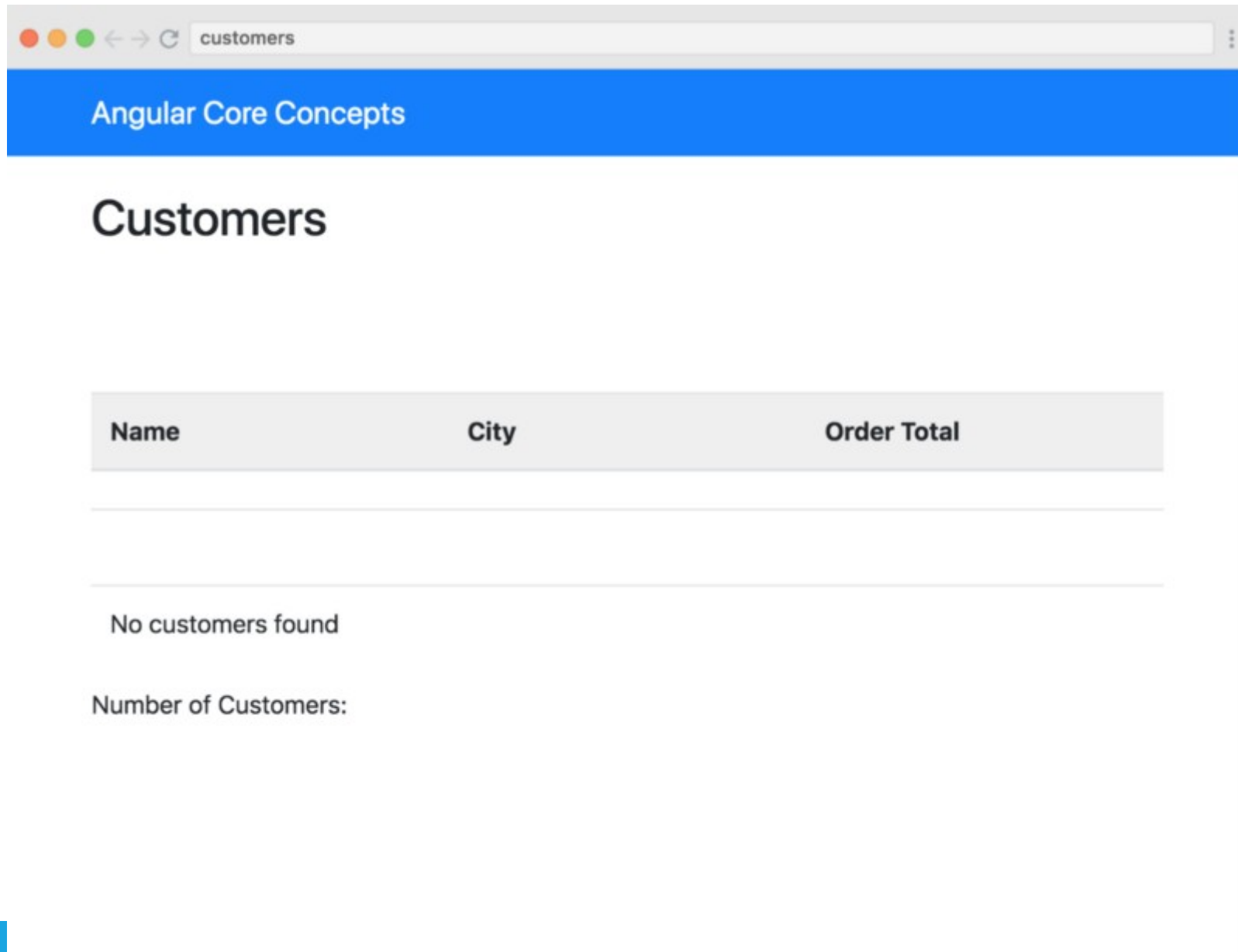


App Component - customers

src / app / customers / customers.component.ts

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-customers',
5    templateUrl: './customers.component.html'
6  })
7  export class CustomersComponent implements OnInit {
8    title: string;
9    people: any[];
10
11    constructor() {}
12
13    ngOnInit() {
14      this.title = 'Customers';
15      this.people = [
16        { id: 1, name: 'John Doe', city: 'Phoenix', orderTotal: 9.99, customerSince: new Date(2014, 7, 10) },
17        { id: 2, name: 'Jane Doe', city: 'Chandler', orderTotal: 19.99, customerSince: new Date(2017, 2, 22)},
18        { id: 3, name: 'Michelle Thomas', city: 'Seattle', orderTotal: 99.99, customerSince: new Date(2002, 10, 31)},
19        { id: 4, name: 'Jim Thomas', city: 'New York', orderTotal: 599.99, customerSince: new Date(2002, 10, 31)},
20      ];
21    }
22  }
```

Customers – customers-list.component



filter-textbox.component

Customers

Filter:

Name	City	Order Total

Shared module - customers

```
export interface ICustomer {  
  id: number;  
  name: string;  
  city: string;  
  orderTotal?: number;  
  customerSince: any;  
}
```

Data Binding

Data Binding Syntax

- Write value using interpolation: `{{ propName }}`
- Bind to DOM properties using **[property]** or **bind-property**
- Bind to DOM events using **(event)** or **on-event**

```
Interpolation: {{ customer.firstName }}
```

```
<button [disabled]="!isEnabled" [style.color]="textColor"  
        (click)="save()">Save</button>
```

```
<div [hidden]="!isVisible"  
    ▶ [class.active]="isActive">...</div>
```

Data bindings

```
<h1 [hidden]="!isVisible">{{ title }}</h1>
```

```
<button (click)="changeVisibility()">Show/Hide</button>
```

Data bindings - dyrektywy

```
12     <tr *ngFor="let cust of filteredCustomers">
13         <td>
14             <a>
15                 {{ cust.name }}
16             </a>
17         </td>
18         <td>{{ cust.city }}</td>
19         <td>{{ cust.orderTotal }}</td>
20     </tr>
21     <tr *ngIf="filteredCustomers && filteredCustomers.length">
22         <td colspan="2">&nbsp;  </td>
23         <td>
24             {{ customersOrderTotal }}
25         </td>
26     </tr>
27     <tr *ngIf="!filteredCustomers || !filteredCustomers.length">
28         <td colspan="4">No customers found</td>
29     </tr>
```


Data bindings - dyrektywy

Customers

Filter:

Name	City	Order Total
john Doe	Phoenix	9.99
Jane Doe	Chandler	19.99
Michelle Thomas	Seattle	99.99
Jim Thomas	New York	599.99

Sortowanie - customer-list.component

```
sort(prop: string) {  
  // A sorter service will handle the sorting  
}
```

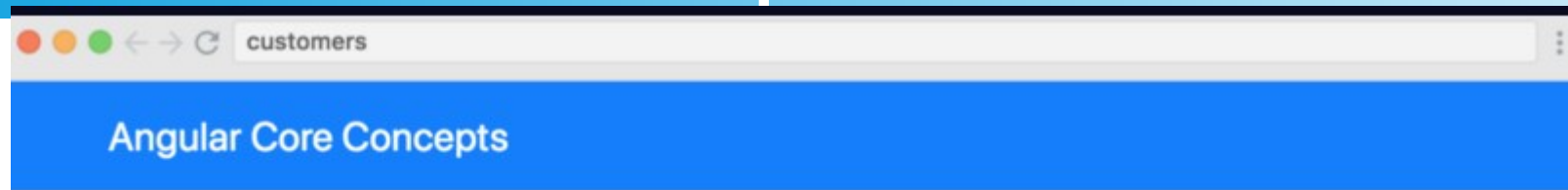
```
<tr>  
  <th (click)="sort('name')">Name</th>  
  <th (click)="sort('city')">City</th>  
  <th (click)="sort('orderTotal')">Order Total</th>  
</tr>
```

Data Binding—Input Properties

```
@Input() get customers(): ICustomer[] {  
    return this._customers  
}  
  
set customers(value: ICustomer[]) {  
    if (value) {  
        this.filteredCustomers = this._customers = value;  
        this.calculateOrders();  
    }  
}
```

```
<app-customers-list [customers]="people"></app-customers-list>
```

Pipes



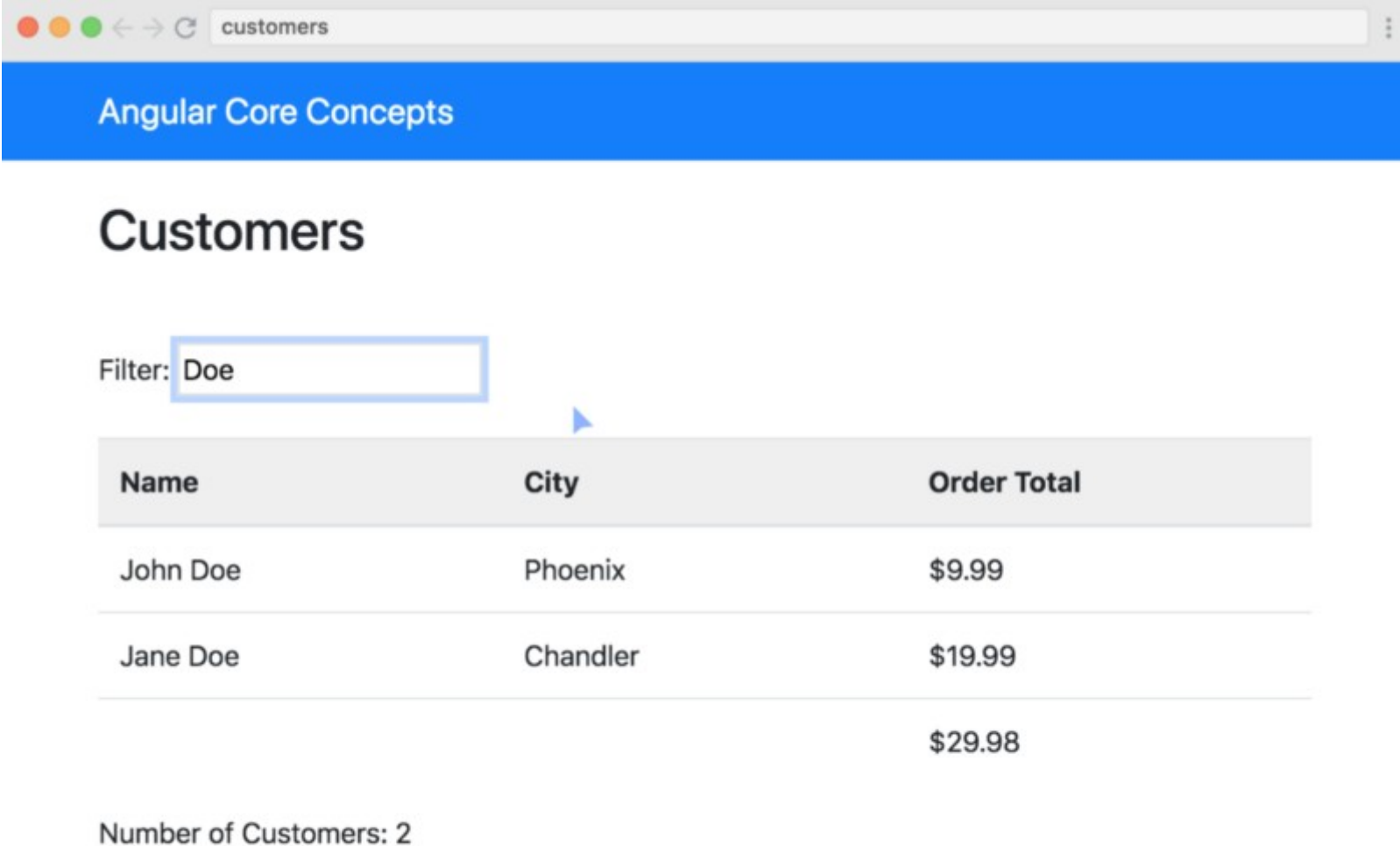
Customers

Filter:

Name	City	Order Total
john Doe	Phoenix	9.99
Jane Doe	Chandler	19.99
Michelle Thomas	Seattle	99.99
Jim Thomas	New York	599.99
		729.96

```
{{ cust.name | uppercase }} // renders JOHN
{{ cust.name | titlecase }} // renders John
```

Filter i EventEmitter



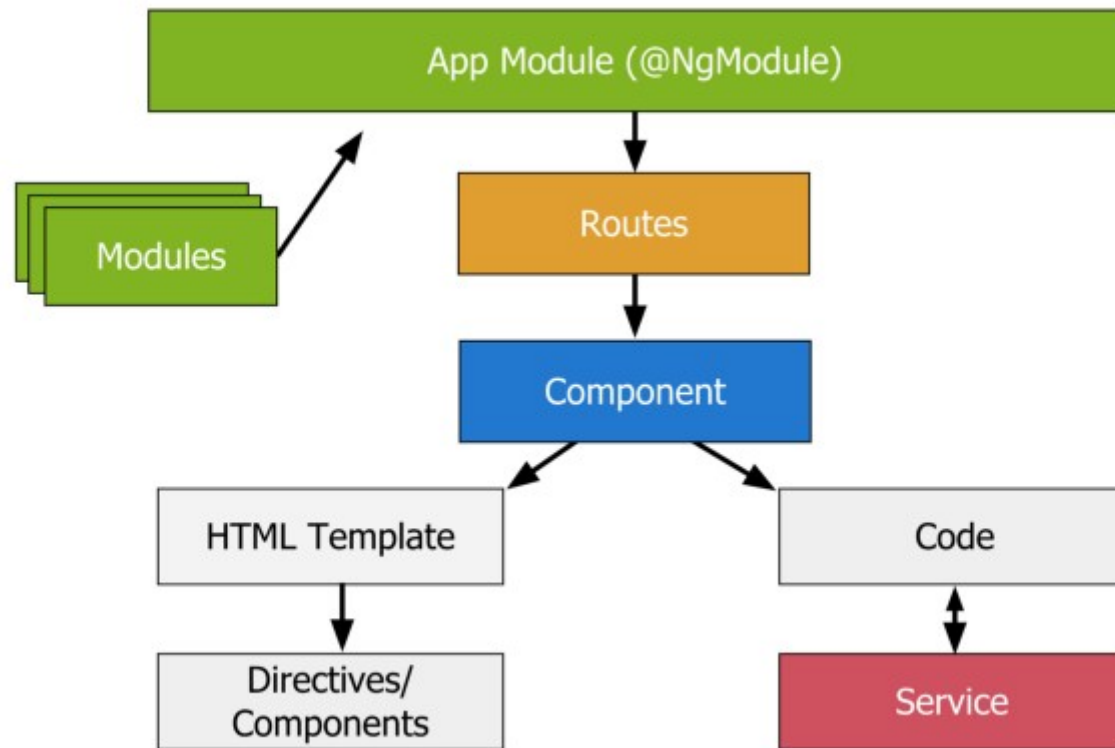
The screenshot shows a web browser window with the address bar displaying 'customers'. The page has a blue header with the text 'Angular Core Concepts'. Below the header, the title 'Customers' is displayed. A filter input field is labeled 'Filter: Doe'. Below the filter, a table lists customers with columns 'Name', 'City', and 'Order Total'. The table contains two rows: 'John Doe' from 'Phoenix' for '\$9.99' and 'Jane Doe' from 'Chandler' for '\$19.99'. A total row at the bottom shows '\$29.98'. Below the table, the text 'Number of Customers: 2' is displayed.

Name	City	Order Total
John Doe	Phoenix	\$9.99
Jane Doe	Chandler	\$19.99
		\$29.98

Number of Customers: 2

```
<filter-textbox (changed)="filter($event)"></filter-textbox>
```

Services



Services

Angular Services

- An Angular service is a class that can be reused throughout an app
- Acts as a "singleton"

data.service.ts

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

@Injectable()
export class DataService {
  constructor() { }

  myServiceFunction() {
    //return data
  }
}
```

Service cd..

```
9  @Injectable()
10  export class DataService {
11
12      baseUrl: string = 'assets/';
13
14      constructor() { }
15
16
17
18
19      private handleError(error: any) {
20          console.error('server error:', error);
21          if (error.error instanceof Error) {
22              const errMessage = error.error.message;
23              return Observable.throw(errMessage);
24              // Use the following instead if using lite-server
25              // return Observable.throw(err.text() || 'backend server error');
26          }
27          return Observable.throw(error || 'Node.js server error');
28      }
29
30  }
```


RxJavaScript

```
34  getOrders(id: number) : Observable<IOrder[]> {  
35      return this.http.get<IOrder[]>(this.baseUrl + 'orders.json')  
36          .pipe(  
37              map(orders => {  
38                  let custOrders = orders.filter((order: IOrder) => order.customerId === id);  
39                  return custOrders;  
40              }),  
41              catchError(this.handleError)  
42          );  
43  }  
44
```

Dodanie serwisu do komponentu

```
import { DataService } from '../core/data.service';
```

```
constructor(private dataService: DataService) {}
```

Zapisywanie się na odpowiedź

```
ngOnInit() {  
  this.title = 'Customers';  
  this.dataService.getCustomers()  
    .subscribe((customers: ICustomer[]) =>  
      this.people = customers);  
}
```

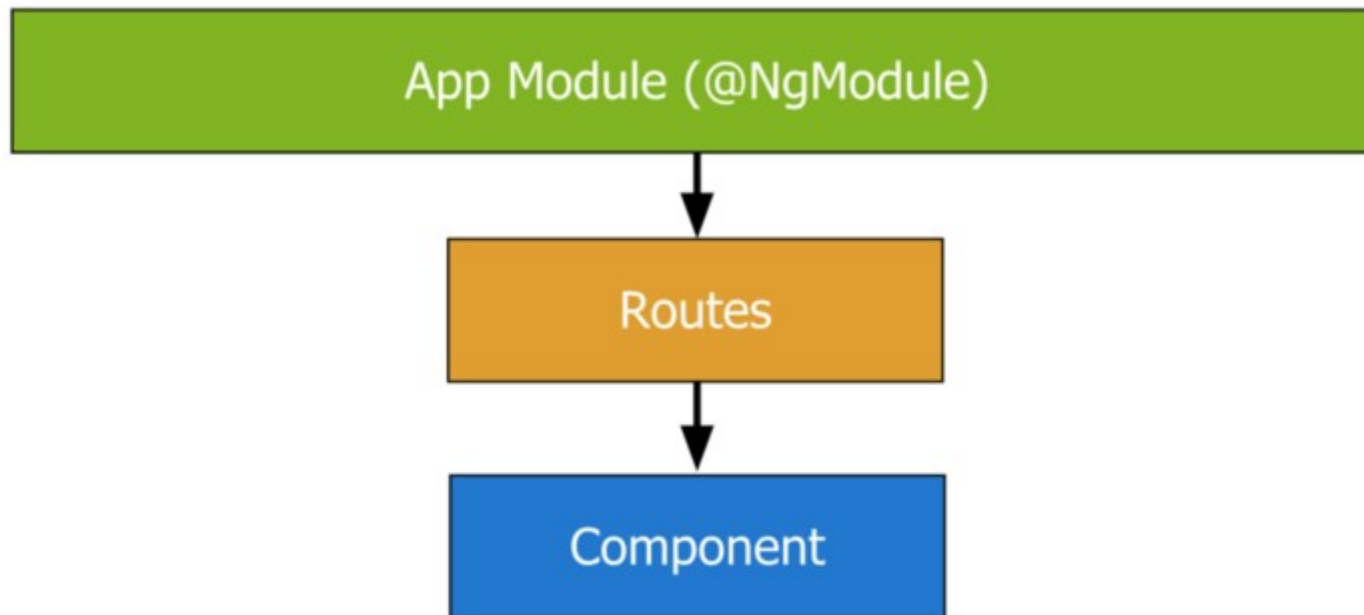
SorterService

```
import { SorterService } from '../core/sorter.service';
```

```
constructor(private sorterService: SorterService) {}
```

```
sort(prop: string) {  
  this.sorterService.sort(this.filteredCustomers, prop);  
}
```

Routing



Routing

```
const routes: Routes = [  
  { path: '', pathMatch: 'full', redirectTo: '/customers' },  
  { path: '**', redirectTo: '/customers' }  
];
```

path — where your user goes, so **path: ''** would be the root of your app. **path: '**'** is a wild card match. It is usually placed last and it's there to cover cases for any route that is not specified in **routes**

pathMatch — how exactly should the route match for a specific component to be displayed

redirectTo — when a path is matched, this is where we send the user. In our case, we send them to **/customers**.