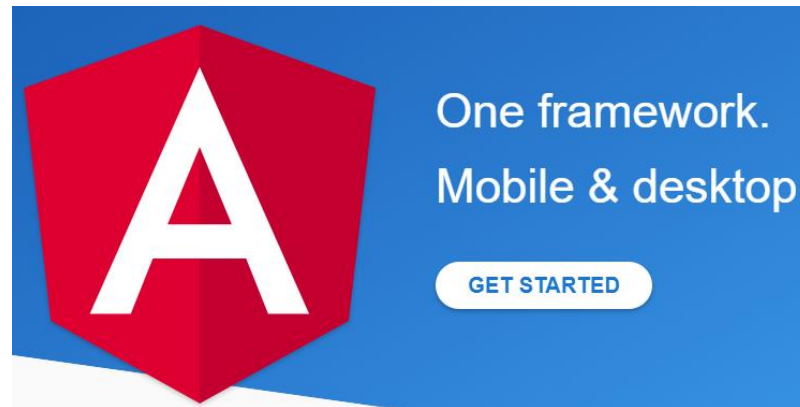


Angular

- Šta je Angular?
- Instalacija radnog okruženja i osnovne komande
- Struktura aplikacije i komponente
- Input komponente i prve direktive: Interpolacija i NgFor
- Motivacija za uvođenje Angulara

Šta je Angular?

- Radni okvir (eng. *framework*) za pravljenje klijentskih aplikacija



- Nastao je u *Google*, ali je sada *open-source* projekat
- Verzije:
 - Verzija 1: AngularJS <https://angularjs.org/>
 - Verzija 2 i više: Angular <https://angular.io/> → fokus ovog kursa

Instalacija radnog okruženja

- *Angular CLI* alat – instalacija i osnovne komande
- Instalacija *TypeScript* paketa u *Sublime*

- Izgradnju Angular aplikacija će nam olakšati *Angular CLI*
- *Angular CLI* je alat pomoću koga možemo
 - Kreirati projekat
 - Dodavati nove fajlove
 - Testirati našu aplikaciju u veb-čitaču (ima ugrađen server)
 - Izvršavati mnogo razvojnih zadataka – testiranje, deployment,...

Zašto Angular CLI?

- *Angular CLI*

- Omogućava efikasan rad (brzo kreiranje aplikacije i njenih komponenti)
- Forsira stilske preporuke Angularovog razvojnog tima

- Stilske preporuke

- Sintaksne konvencije, organizacija aplikacije (struktura direktorijuma),...
- Poštovanje ovih preporuka omogućava da naša aplikacija bude pregledna i nama i drugim programerima sa kojima sarađujemo
- Ovo je od izuzetne važnosti kada gradimo kompleksne aplikacije
- *Angular Style Guide* <https://angular.io/guide/styleguide>

Instalacija Angular CLI

- Preduslov je da imate instaliran *Node.js* <http://nodejs.org>
- Otvorite komandnu liniju i unesite komandu:
`npm install -g @angular/cli`
- Nakon instalacije restartujte komandnu liniju. Sada možete koristiti *Angular CLI* alat iz komandne linije
 - Sve komande počinju sa `ng`
 - `ng help` izlistava sve opcije koje nudi *Angular CLI*

Kreiranje nove aplikacije

- U komandnoj liniji:
 - Pozicionirajte se u direktorijum gde želite da se nalazi vaša aplikacija
 - Unesite komandu `ng new moja-aplikacija`
 - Ovo kreira novu Angular aplikaciju u direktorijumu *moja-aplikacija*

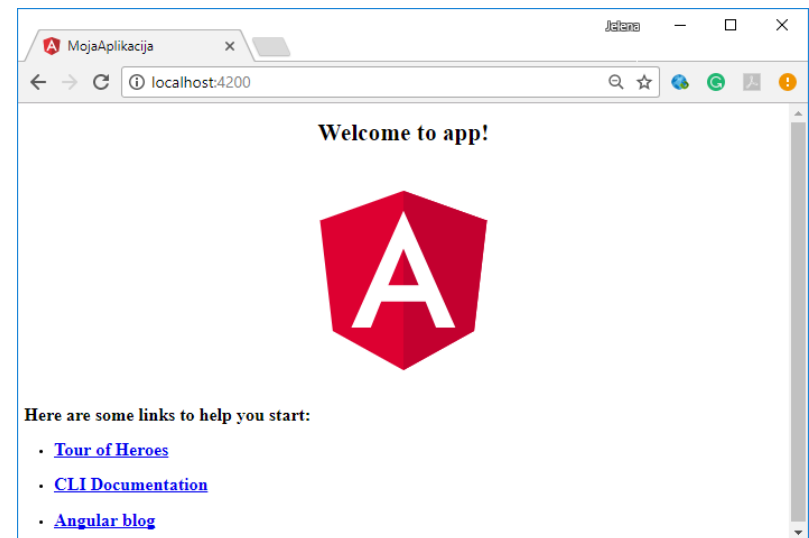
Pokretanje aplikacije

- Pozicionirajte se u direktorijum koji sadrži vašu aplikaciju (npr. unutar direktorijuma *moja-aplikacija*)
- Unesite komandu `ng serve`

```
PS C:\Users\Jelena\Documents\Nastava\Front-end-kurs\Anuglar4\moji_primeri> cd .\moja-aplikacija\  
PS C:\Users\Jelena\Documents\Nastava\Front-end-kurs\Anuglar4\moji_primeri\moja-aplikacija> ng serve  
** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **  
Date: 2017-10-12T09:15:18.046Z  
Hash: 63c5458b222315ee7f72  
Time: 11170ms  
chunk {inline} inline.bundle.js, inline.bundle.js.map (inline) 5.83 kB [entry] [rendered]  
chunk {main} main.bundle.js, main.bundle.js.map (main) 8.69 kB {vendor} [initial] [rendered]  
chunk {polyfills} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 217 kB {inline} [initial] [rendered]  
chunk {styles} styles.bundle.js, styles.bundle.js.map (styles) 11.3 kB {inline} [initial] [rendered]  
chunk {vendor} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.29 MB [initial] [rendered]  
  
webpack: Compiled successfully.
```

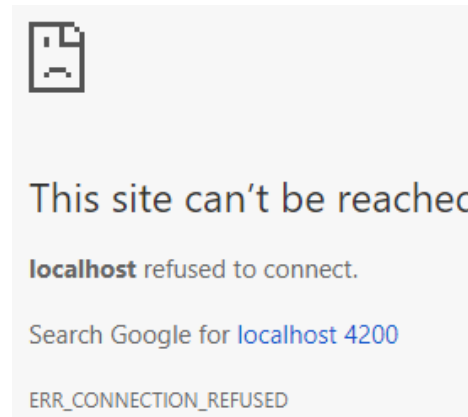
URL na kom je aplikacija dostupna

- Sada možete videti kako vaša aplikacija izgleda u veb-čitaču na URL-u <http://localhost:4200/>
- Nemojte gasiti komandnu liniju!
- Prilikom vaših promena aplikacija će se automatski kompajlirati



Pokretanje aplikacije

- Ako vidite ovaj ekran kada pokušate da testirate vašu aplikaciju u veb-čitaču:



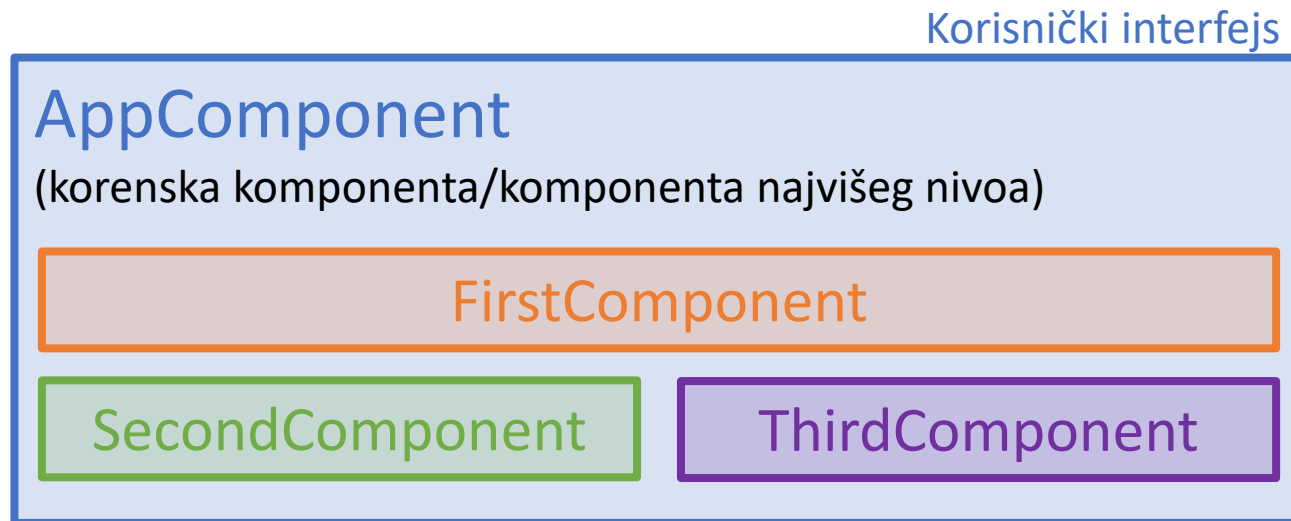
- To znači da ste ili:
 - Zatvorili komandnu liniju (ili na drugi način prekinuli *Angular CLI* server). Ako je ovo slučaj, ponovo (u direktorijumu aplikacije) izvršite komandu `ng serve`
 - Pogrešili URL (pogledajte šta kaže komanda `ng serve`)
- Ako imate potrebu da unesete novu komandu u komandnu liniju, a ne želite da gasite server – prosto otvorite još jednu komandnu liniju

Struktura aplikacije

- Komponente
- Moduli

Struktura aplikacije – Komponente

- Angular aplikacije su modularne i baziraju se na komponentima
 - Svaka komponenta je odgovorna za jedan deo korisničkog interfejsa – svakoj dodeljujemo jedan „deo ekrana“ za koji je zadužena
 - Komponenta je odgovorna za prikaz i funkcionalnost dela interfejsa za koji je zadužena



- *Napomena: u primerima koji slede nije prikazivana AppComponent, ali se podrazumeva da su sve prikazane komponente ugnježdene u nju*

YouTube

Header

Channel

YouTube

Search

Home

Trending

Subscriptions

LIBRARY

History

Watch later

Purchases

Liked videos

Show more

SUBSCRIPTIONS

The Health Nerd

CrashCourse

TED-Ed

Justine Leconte ...

Kurzgesagt – In ...

The School of Li...

Wisecrack

Justine Leconte officiel

VideoDetails

Videos

Fabric types & weights

Fabrics: woven or knitted? Which weight? How to

Justine Leconte officiel

21K views • 22 hours ago

ad CHARTPAK vs. COPIC

How to color with markers | COPIC vs. AD CHARTPAK |

Justine Leconte officiel

4K views • 4 days ago

Summer → Fall

Summer to fall: transitional wardrobe tips | CAPSULE

Justine Leconte officiel

69K views • 3 weeks ago

HOW TO clean shoes

How to clean shoes: leather, boots, sneakers, white shoes,

Justine Leconte officiel

59K views • 3 weeks ago

Essential Accessories

10 essential accessories | CAPSULE GUIDE | Justine

Justine Leconte officiel

96K views • 4 weeks ago

Recommended

Videos

WHY IS IT SO HARD TO CURE CANCER?

Why is it so hard to cure cancer? - Kyuson Yun

TED-Ed

244K views • 1 day ago

MUST. RESIST. PIZZA.

How to Stay Motivated to Lose Weight: 5 Science

The Health Nerd

306K views • 9 months ago

IS DAIRY BAD FOR YOU?

Is Dairy Bad for You? 6 Facts About Dairy for Your Health

The Health Nerd

158K views • 5 months ago

OPTIMISTIC NIHILISM

Optimistic Nihilism

Kurzgesagt – In a Nutshell

4.3M views • 2 months ago

The Surprising Cause of Stomach Ulcers

The surprising cause of stomach ulcers - Rusha Modi

TED-Ed

438K views • 1 week ago

BREAKING SUGAR ADDICTION

The STRANGE CASE of the CYCLOPS SLEEP

FAHRENHEIT

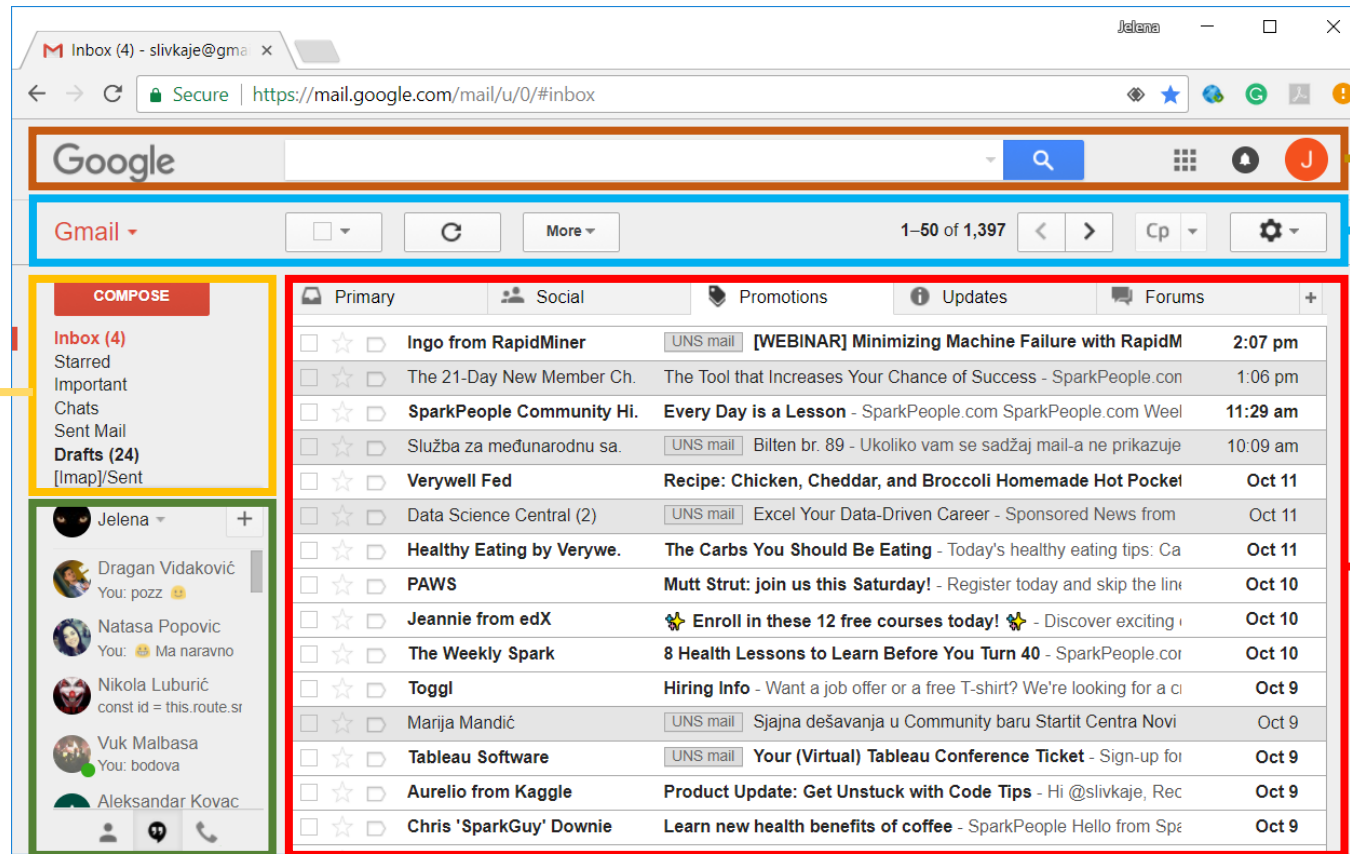
HOW ASPIRIN WAS DISCOVERED

WHY WE FEEL LONELY AND ODD

SideBar

Channel

Gmail



Header

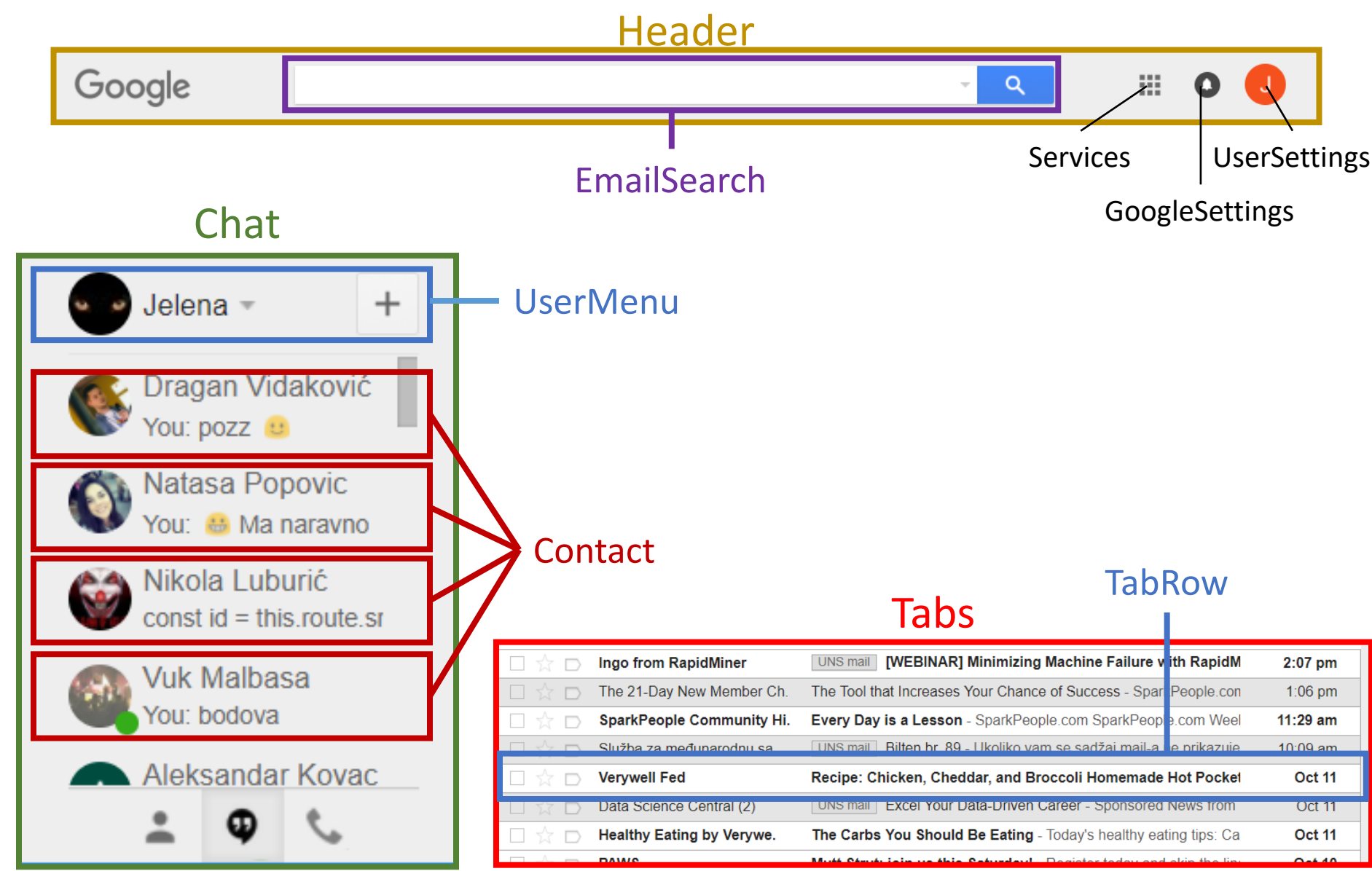
Menu

SideBar

Chat

Tabs

Gmail



Twitter

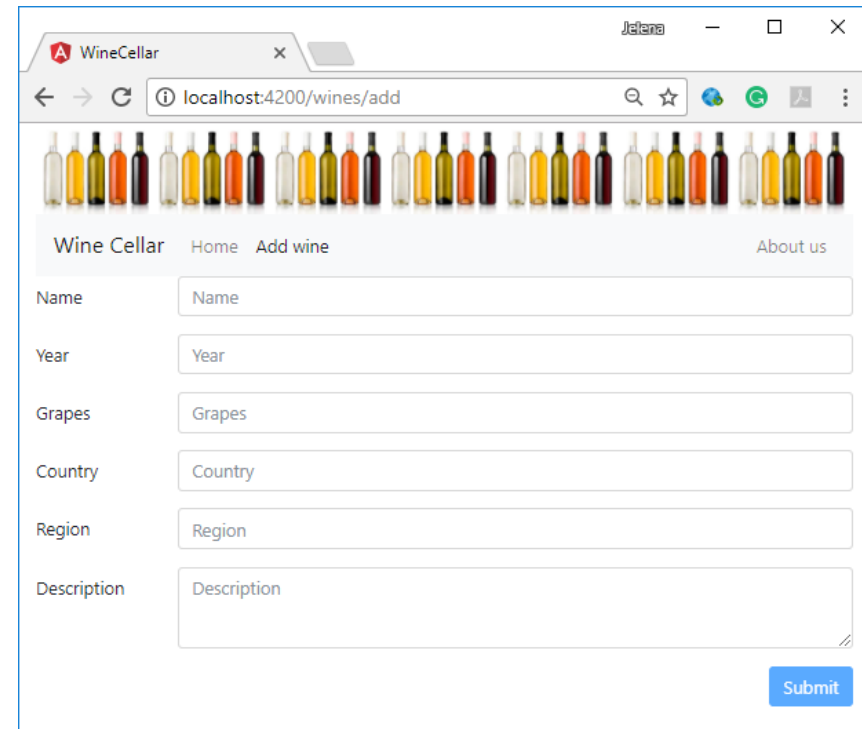
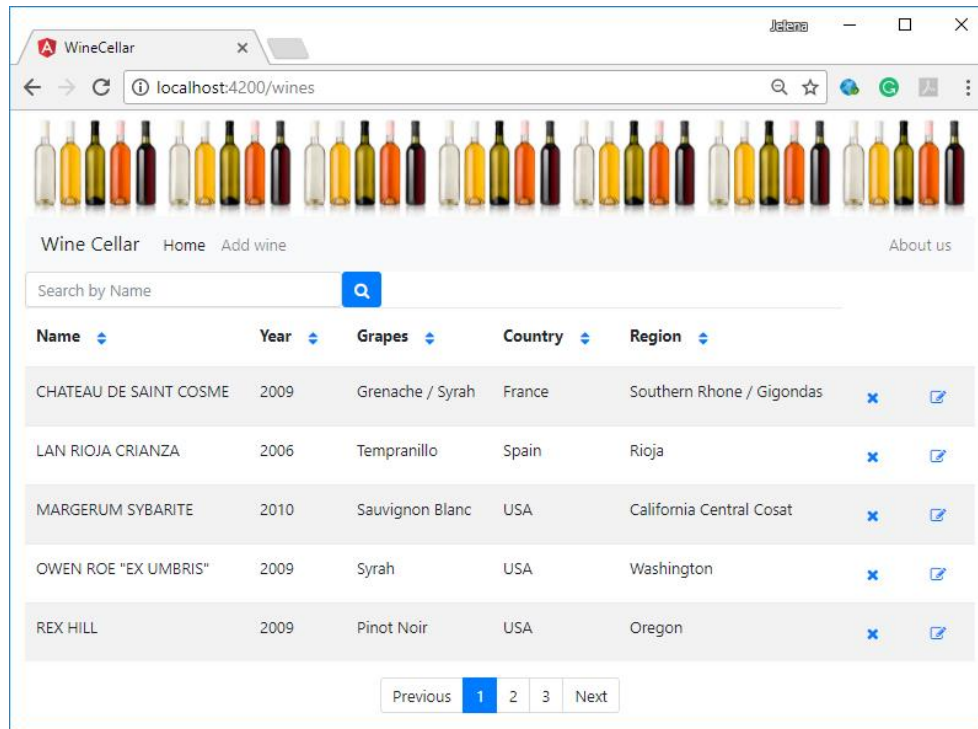
The image shows a screenshot of Donald J. Trump's Twitter profile page. The page is annotated with several colored boxes and labels:

- Header** (Red line): Points to the top navigation bar containing Home, Notifications, Messages, and a search bar.
- Navbar** (Orange line): Points to the profile header area, which includes the profile picture, a large American flag banner, and statistics: Tweets (36.1K), Following (45), Followers (40.4M), Likes (17), and Moments (5). A 'Follow' button is also present.
- UserDetails** (Purple line): Points to the bio section on the left, which includes the name 'Donald J. Trump', handle '@realDonaldTrump', title '45th President of the United States of America', location 'Washington, DC', and join date 'Joined March 2009'. Below this is a 'Tweet to Donald J. Trump' button.
- Photos** (Green line): Points to a grid of 2,310 photos and videos.
- Recommendations** (Red line): Points to the 'Who to follow' section, which lists 'President Trump', 'Hillary Clinton', and 'CNN' with 'Follow' buttons.
- Search** (Black line): Points to the 'Tweets & replies' tab in the feed menu.
- FeedMenu** (Black line): Points to the 'Tweets & replies' tab in the feed menu.
- Feed** (Yellow line): Points to a specific tweet in the feed.
- Feeds** (Green line): Points to the entire list of tweets in the feed.
- Recommendation** (Blue line): Points to a tweet by CNN in the 'Who to follow' section.

The feed itself contains several tweets from Donald J. Trump, including one about 'The Fake News' and another about 'FEMA'.

Zadatak 1

- Kako biste podelili sledeću aplikaciju na komponente?



Kreiranje komponente

- Komponente u našoj aplikaciji možemo kreirati pomoću *Angular CLI* alata
- U komandnoj liniji se pozicionirajte u direktorijum sa vašom aplikacijom (npr. direktorijum *moja-aplikacija*)
- Komponenta se generiše komandom:
`ng g component moja-komponenta`
 - Ovo generiše novu komponentu koja se zove *MojaKomponentaComponent*
 - Definicija komponente se nalazi unutar direktorijuma *moja-aplikacija/src/app/moja-komponenta*
- Prilikom generisanja komponente možemo specificirati direktorijum u okviru koga će komponenta biti kreirana:
`ng g component moj-direktorijum/moja-komponenta`
 - Ovo generiše novu komponentu koja se zove *MojaKomponentaComponent*
 - Definicija komponente se nalazi unutar direktorijuma *moja-aplikacija/src/app/moj-direktorijum/moja-komponenta*

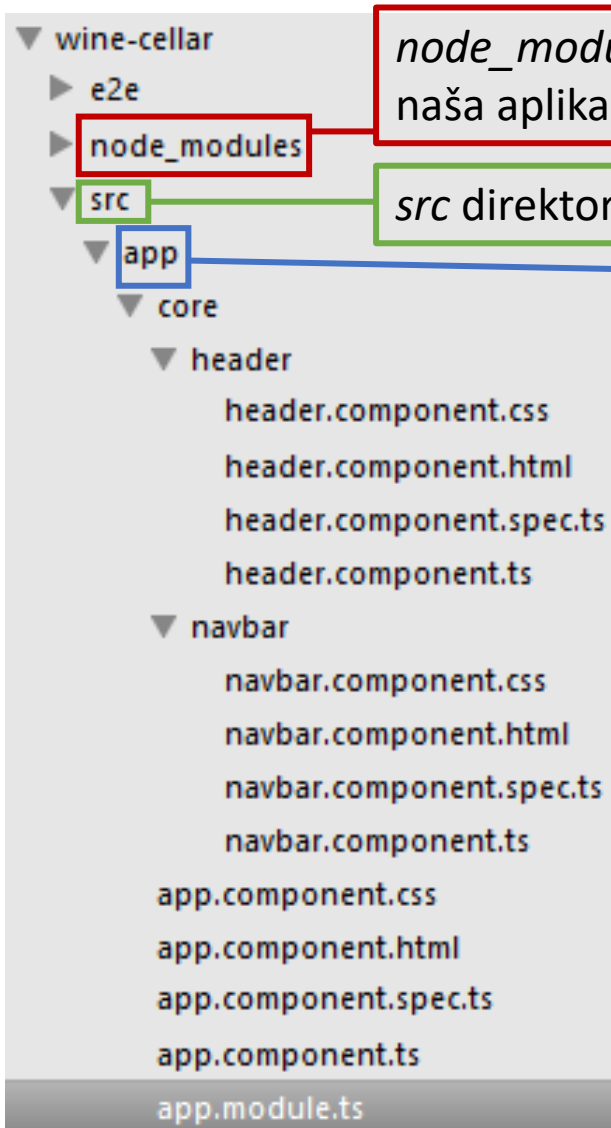
Generisanje komponenti – primer_1/wine-cellar

- Napravićemo komponente za aplikaciju iz zadatka 1:
 1. Generisaćemo novu aplikaciju koja se zove wine-cellar:
`ng new wine-cellar`
 2. Generisanje komponenti
 - Komponente koje se pojavljuju na svakom prikazu ćemo stavljati u direktorijum wine-cellar/app/`core`
 - Generisanje komponente `HeaderComponent`: `ng g component core/header`
 - Generisanje komponente `NavbarComponent`: `ng g component core/navbar`
 - Komponente vezane za manipulaciju vinima ćemo stavljati u direktorijum wine-cellar/app/`wine`
 - Generisanje `WineListComponent`: `ng g component wine/wine-list`
 - Generisanje `EditWineComponent`: `ng g component wine/edit-wine`

Zadatak 2

- *WineListComponent* se deli na tri komponente: *SearchFormComponent*, *TableComponent* i *PaginationComponent*.
- U započetoj aplikaciji wine-cellar kreirajte ove tri komponente tako da se nalaze unutar direktorijuma *wine-cellar/app/wine*

Struktura aplikacije



node_modules sadrži spoljne biblioteke na koje se naša aplikacija oslanja (uključujući Angular)

src direktorijum sadrži kod naše aplikacije

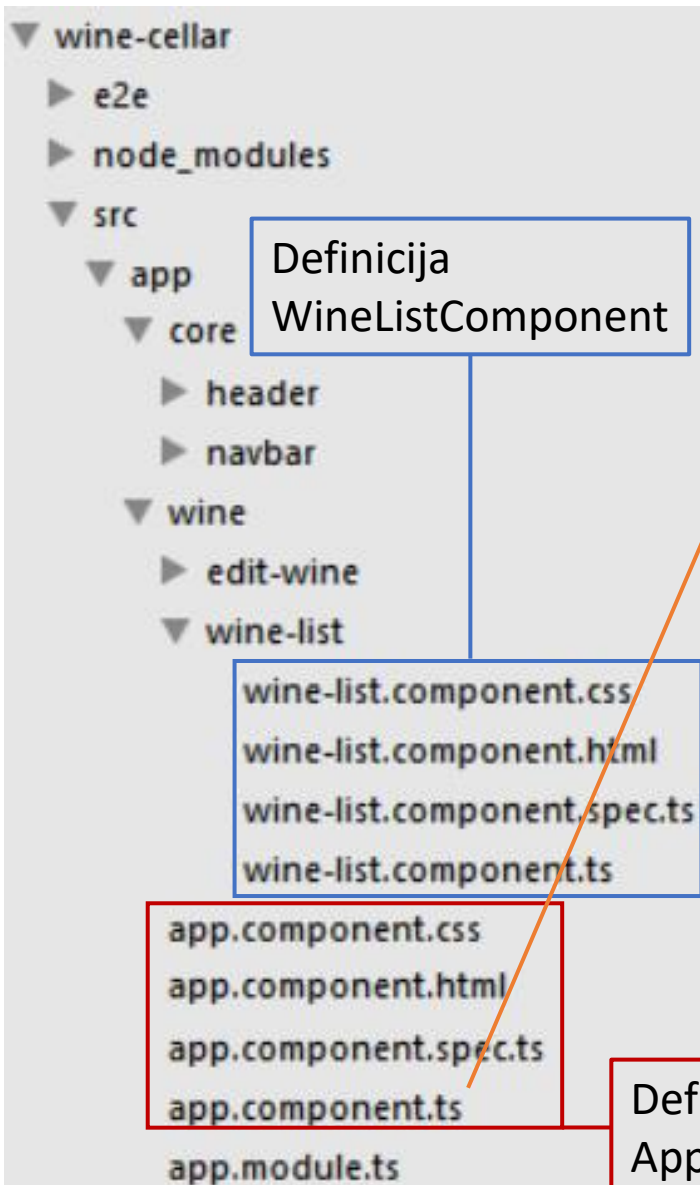
app sadrži komponente

Važna napomena:

Primeri okačeni na sajt kursa ne sadrže direktorijum *node_modules*. Da biste ih mogli pokrenuti u komandnoj liniji:

- Pozicionirajte se u direktorijum u kome je primer raspakovan (npr. ovde *wine-cellar*)
- Unesite komandu: `npm install`
- Za uspešno izvršavanje ove komande morate biti konektovani na internet. Izvršavanje će malo da potraje, a, nakon uspešnog izvršavanja, pojaviće se *node_modules*

Od čega se sastoji komponenta?



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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
}
```

Dekorator

Klasa

TypeScript klasa – ovde ćemo definisati funkcionalnost komponente

Dekorator – Specificira dodatne podatke o klasi

export – ovu klasu možemo importovati u neku drugu

Npr. u ovu klasu smo importovali *Component* klasu iz *@angular/core* (jer se u *Component* klasi nalazi definicija *@Component* dekoratora koji ovde koristimo)

Od čega se sastoji komponenta?

Dekorator komponente

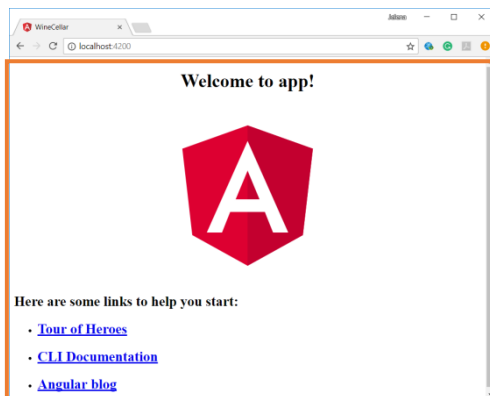
```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

styleUrls – u ovom fajlu možemo po potrebi definisati CSS stilove specifične za ovu komponentu

selector – pomoću `<app-root></app-root>` možemo iskoristiti ovu komponentu u prikazu (dodeliti joj deo ekrana)

templateUrl – definiše gde se nalazi templejt komponente (HTML) koji definiše kako komponenta izgleda

Selektor *AppRoot* komponente je u `index.html` iskorišćen da bi se ovoj komponenti dodelila nadležnost nad prikazom celog ekrana (ceo `<body>` element)



AppComponent

```
index.html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MojaAplikacija</title>
  <base href="/">

  <meta name="viewport" content=
  <link rel="icon" type="image/

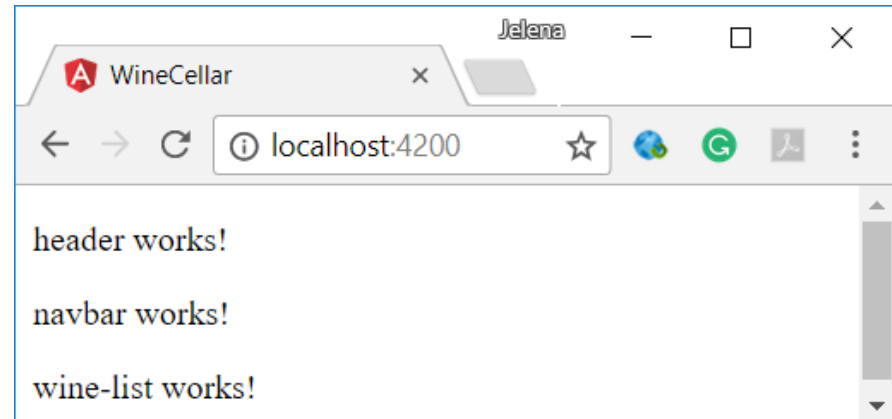
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Zadatak 3

Izmeniti započetu wine-cellar aplikaciju:

- a) Primetite da sve komponente koje ste kreirali imaju u selektoru prefiks “app-”. Generalno, dobra ideja je da ovaj prefiks bude specifičan za aplikaciju. Zbog toga, izmenite selektore svih komponenti da imaju prefiks “wc-”, tj. da selektori budu: *wc-wine-list*, *wc-edit-wine*, *wc-header*,... (ne morate menjati prefiks za *AppRootComponent*)
- b) Promeniti templejt (prikaz) *AppComponent* tako da se deo ekrana dodeljen ovoj komponenti deli na tri reda:
 - Prvim redom upravlja *HeaderComponent*
 - Drugim redom upravlja *NavbarComponent*
 - Trećim redom upravlja *WineListComponent*

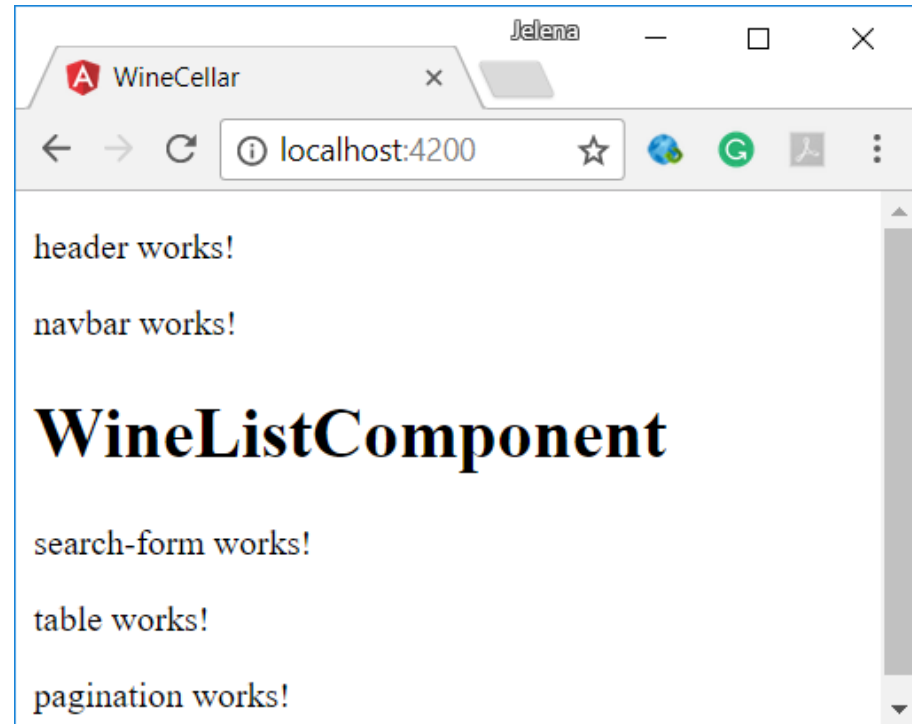
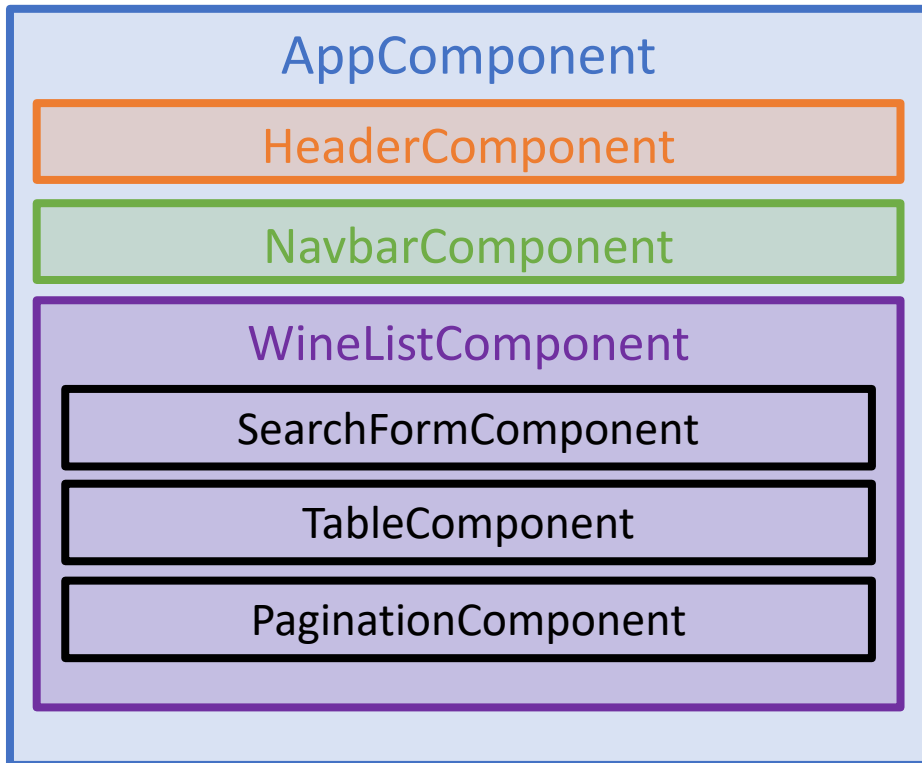
Korisnički interfejs



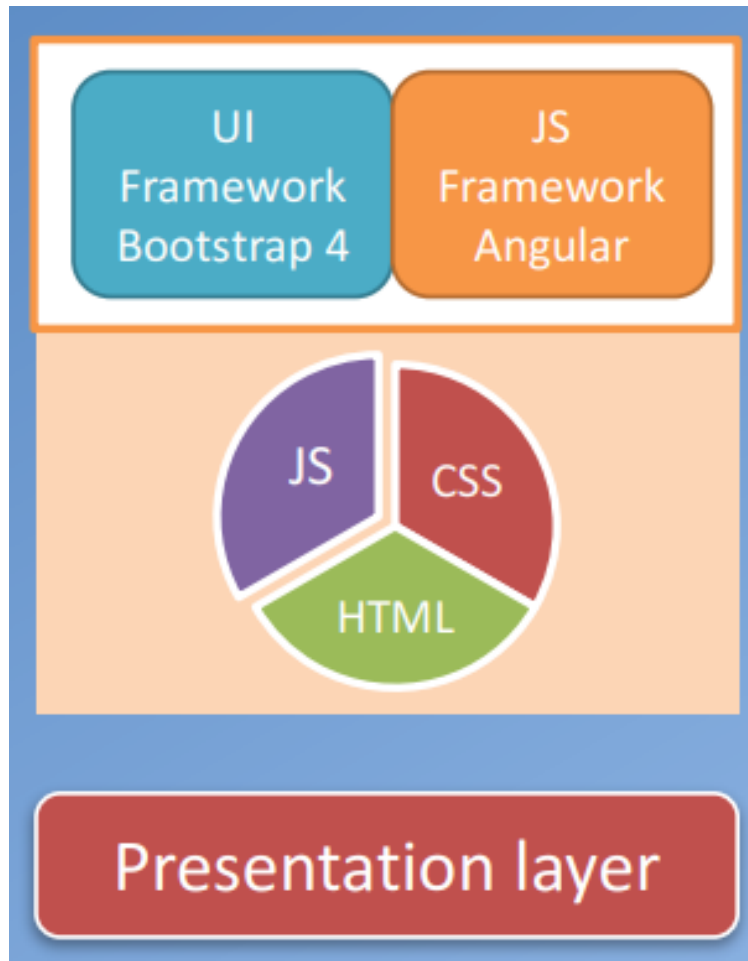
Zadatak 3

- c) U delu ekrana dodeljen komponenti *WineListComponent* dodati heading prvog nivoa sa tekstom “*WineListComponent*”, a ostatak ekrana podeliti na tri reda:
- Prvi red kontroliše *SearchFormComponent*
 - Drugi red kontroliše *TableComponent*
 - Treći red kontroliše *PaginationComponent*

Korisnički interfejs



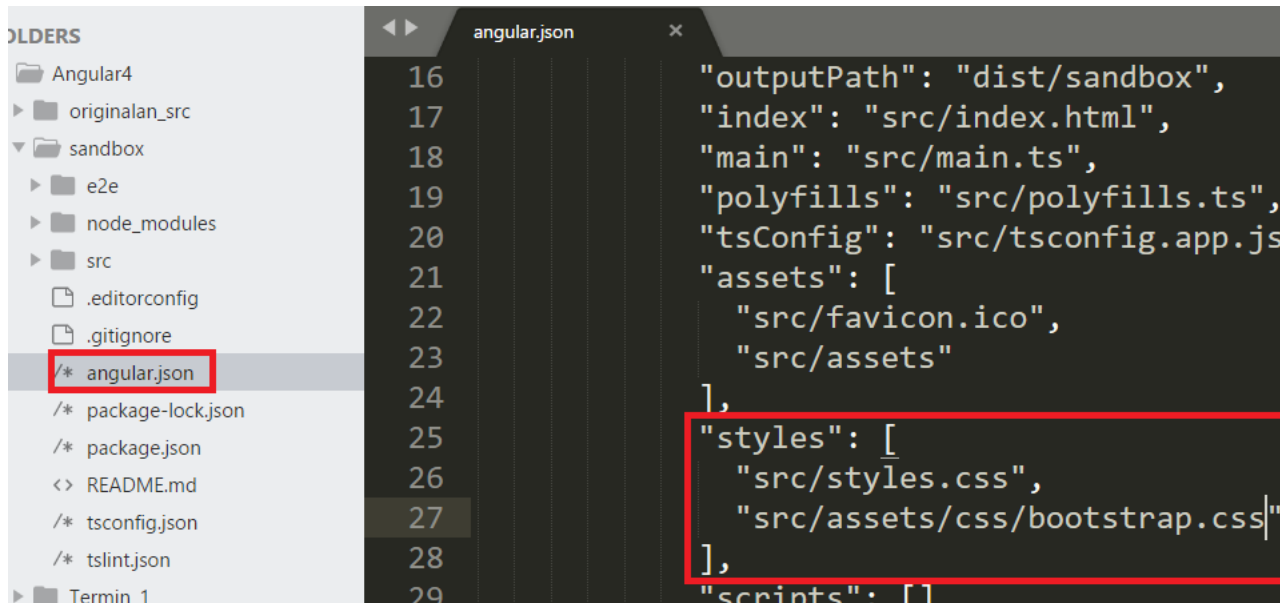
Bootstrap



- Do sada ste za definisanje izgleda koristili CSS
- Sada ćemo to znanje proširiti tako što ćemo uvesti *Bootstrap* – ovo je kolekcija predefinisanih CSS klasa
- Koristićemo (trenutno aktuelnu) verziju 4 <https://v4-alpha.getbootstrap.com/>
- Identičan prikaz na svim modernim veb-čitačima
- *Responsive* – sajt pravljen pomoću *Bootstrapa* automatski se prilagođava veličini uređaja na kome se prikazuje (mobilni, tablet, PC,...)
- *Open source*
- Dobro dokumentovan

Dodavanje Bootstrap biblioteke u aplikaciju

1. Skinite Bootstrap 4 sa <https://getbootstrap.com/docs/4.0/getting-started/download/>
2. U skinutoj arhivi, u direktorijumu `css` pronađite datoteke: `bootstrap.css` i `bootstrap.css.map`. Kopirajte ove datoteke u vašu aplikaciju u `/src/assets/css` direktorijum (`css` direktorijum treba da napravite i u njega smestite fajlove)
3. Izmenite `wine-cellar/.angular.json` tako da `styles` polje sadrži `src/assets/css/bootstrap.css` pored `styles.css` (obratite pažnju na navodnike):



```
angular.json
16  "outputPath": "dist/sandbox",
17  "index": "src/index.html",
18  "main": "src/main.ts",
19  "polyfills": "src/polyfills.ts",
20  "tsConfig": "src/tsconfig.app.js",
21  "assets": [
22    "src/favicon.ico",
23    "src/assets"
24  ],
25  "styles": [
26    "src/styles.css",
27    "src/assets/css/bootstrap.css"
28  ],
29  "scripts": []
```

4. Restartujte Angular CLI

Bootstrap grid sistem

1 of 2	1 of 2	
1 of 3	1 of 3	1 of 3

```
<div class="container">
  <div class="row">
    <div class="col">
      1 of 2
    </div>
    <div class="col">
      1 of 2
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col">
      1 of 3
    </div>
    <div class="col">
      1 of 3
    </div>
  </div>
</div>
```

- Pomoću *grid* sistema možemo definisati raspored različitih komponenti na ekranu
- *Grid* sistem obezbeđuje da će se definisan interfejs ponašati isto na svim veličinama ekrana
- Ekran se deli na redove (klasa *row*)
- U svaki red se ugrađuju kolone (klasa *col*)
- Komponente korisničkog interfejsa smeštamo u kolone. Ista komponenta može da zauzima više kolona (videti sliku)

Primena *grid* sistema u wine-cellar aplikaciji

- Iskoristićemo *Bootstrap* grid sistem da definišemo *grid* sistem u templejtu komponente *AppComponent*:

```
<div class="container-fluid">  
  <div class="row">  
    <div class="col">  
      <wc-header></wc-header>  
    </div>  
  </div>  
  
  <div class="row">  
    <div class="col">  
      <wc-navbar></wc-navbar>  
    </div>  
  </div>  
  
  <div class="row">  
    <div class="col">  
      <wc-wine-list></wc-wine-list>  
    </div>  
  </div>  
</div>
```

Klasa *container-fluid* definiše da će se korisnički interfejs prostirati preko celog ekrana (kolikigod ekran bio – mobilni, ipad,...)

Ovom *div* elementu je dodeljena klasa *row* pa će sav njegov sadržaj predstavljati jedan red

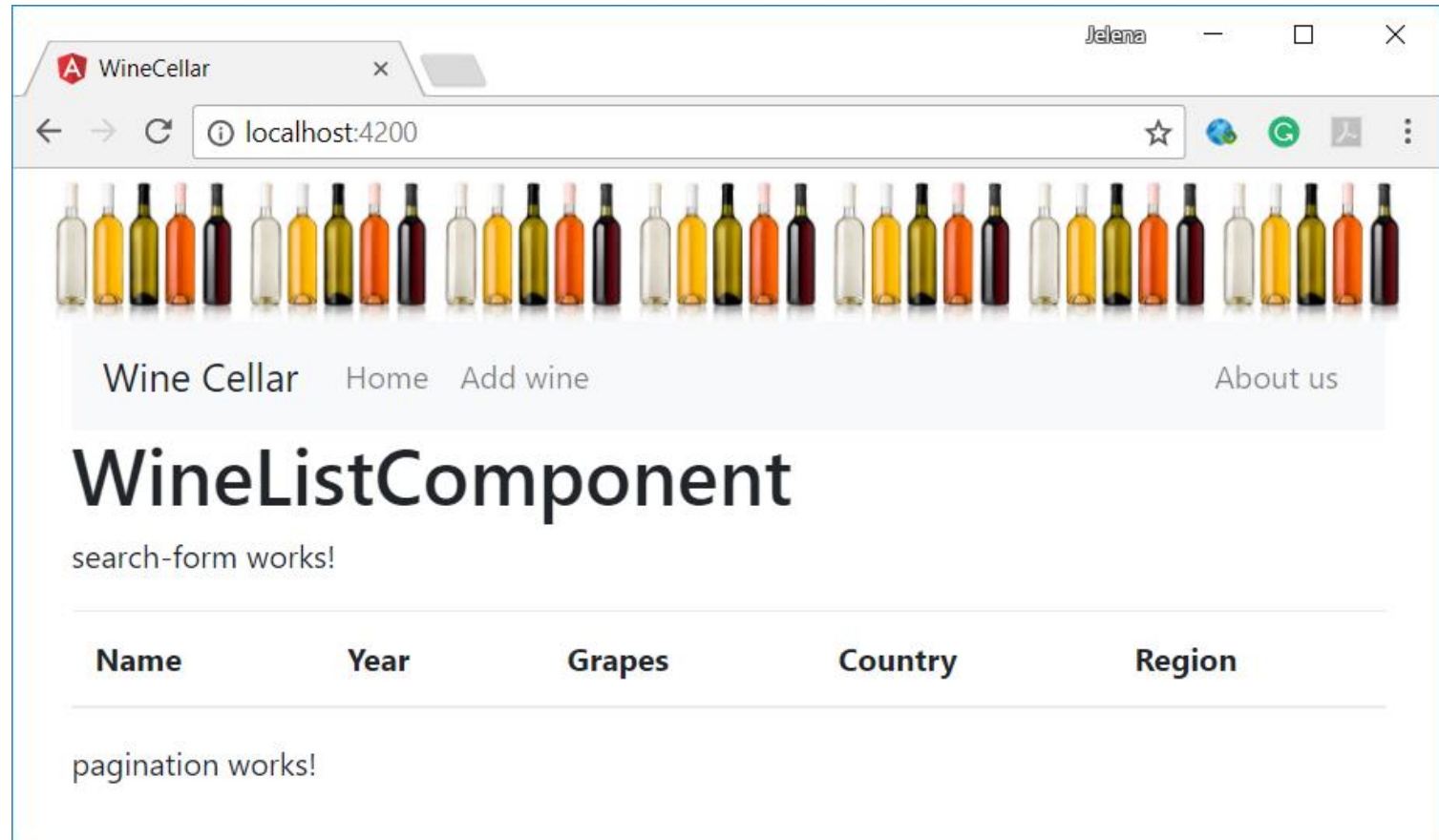
div elementu unutar reda smo dodelili klasu *col* – sadržaj ovog *div*-a (komponenta *NavbarComponent*) zauzima datu kolonu. Pošto je to jedina kolona u okviru reda, sadržaj *NavbarComponent* će se raširiti preko celog reda

Zadatak 4

- a) Dodati *Bootstrap* biblioteku u wine-cellar aplikaciju
- b) Dodati *Bootstrap* klase kojim ćemo definisati grid sistem u templejtu komponente *AppComponent*
 - U korenski *div* element dodati klasu *container-fluid*
 - Staviti da se svaka od podređenih komponenti nalazi u jednom redu i da zauzima svih 12 kolona tog reda
- c) Dodati *Bootstrap* klase kojim ćemo definisati grid sistem u templejtu komponente *WineListComponent*
 - Nemojte stavljati *container-fluid* (jer je templejt ove komponente već ugnježđen u *container-fluid* definisan u okviru *AppComponent*)
 - Staviti da se svaka od podređenih komponenti nalazi u jednom redu i da zauzima svih 12 kolona tog reda
- d) Zameniti templejte *HeaderComponent*, *NavbarComponent* i *TableComponent* sa odgovarajućim HTML kodom datim u fajlovima *header.html*, *navbar.html* i *table.html*
 - Datoteka *header.html* se oslanja na sliku *wine.jpg* (priloženu u zadatku)
 - smestiti ovu sliku u direktorijum *wine-cellar/src/assets/images/*

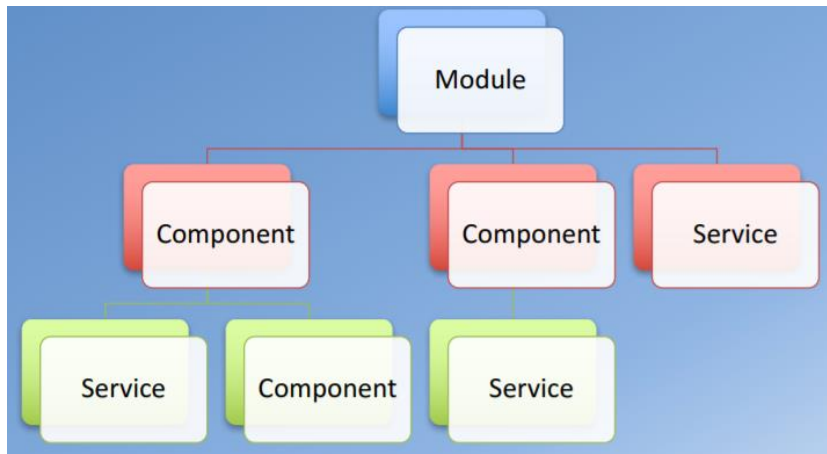
Zadatak 4

- Željeni izgled *WineListComponent*:

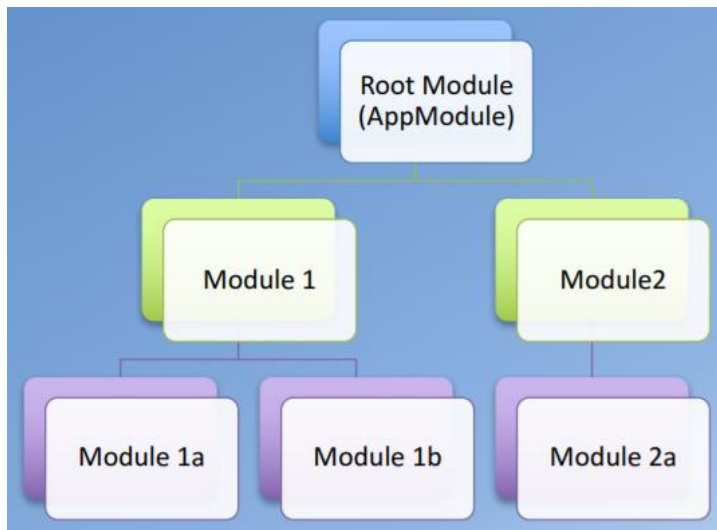


Struktura aplikacije - Modul

- Angular aplikacije su modularne



- Svi delovi neophodni za funkcionisanje naše aplikacije se stavljaju u **modul**
- U **modul** možemo staviti i druge **module**
- Recimo, mogli bismo modul sa našom aplikacijom staviti u modul neke druge aplikacije i tako u toj drugoj aplikaciji imati celokupnu funkcionalnost naše aplikacije



- Dakle, naša aplikacija se u svom izvršavanju može oslanjati i na druge module
- Moduli se organizuju u hijerarhijsku strukturu
- Na vrhu se nalazi *Root Module* (korenski modul, modul najvišeg nivoa)
- Tipično, korenski modul se zove *AppModule*

Struktura aplikacije – definicija *AppModule*

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5 import { HeaderComponent } from './core/header/header.component';
6 import { NavbarComponent } from './core/navbar/navbar.component';
7 import { WineListComponent } from './wine/wine-list/wine-list.component';
8 import { EditWineComponent } from './wine/edit-wine/edit-wine.component';
9 import { SearchFormComponent } from './wine/search-form/search-form.component';
10 import { PaginationComponent } from './wine/pagination/pagination.component';
11 import { TableComponent } from './wine/table/table.component';
12
13 @NgModule({
14   declarations: [
15     AppComponent,
16     HeaderComponent,
17     NavbarComponent,
18     WineListComponent,
19     EditWineComponent,
20     SearchFormComponent,
21     PaginationComponent,
22     TableComponent
23   ],
24   imports: [
25     BrowserModule
26   ],
27   providers: [],
28   bootstrap: [AppComponent]
29 })
30 export class AppModule { }
```

declarations – view klase koje pripadaju ovom modulu

imports – moduli koji su importovani u ovaj modul

Prilikom bootstrapovanja aplikacije, bootstrapovaće se *AppComponent* komponenta – ovo će biti korenska komponenta naše aplikacije

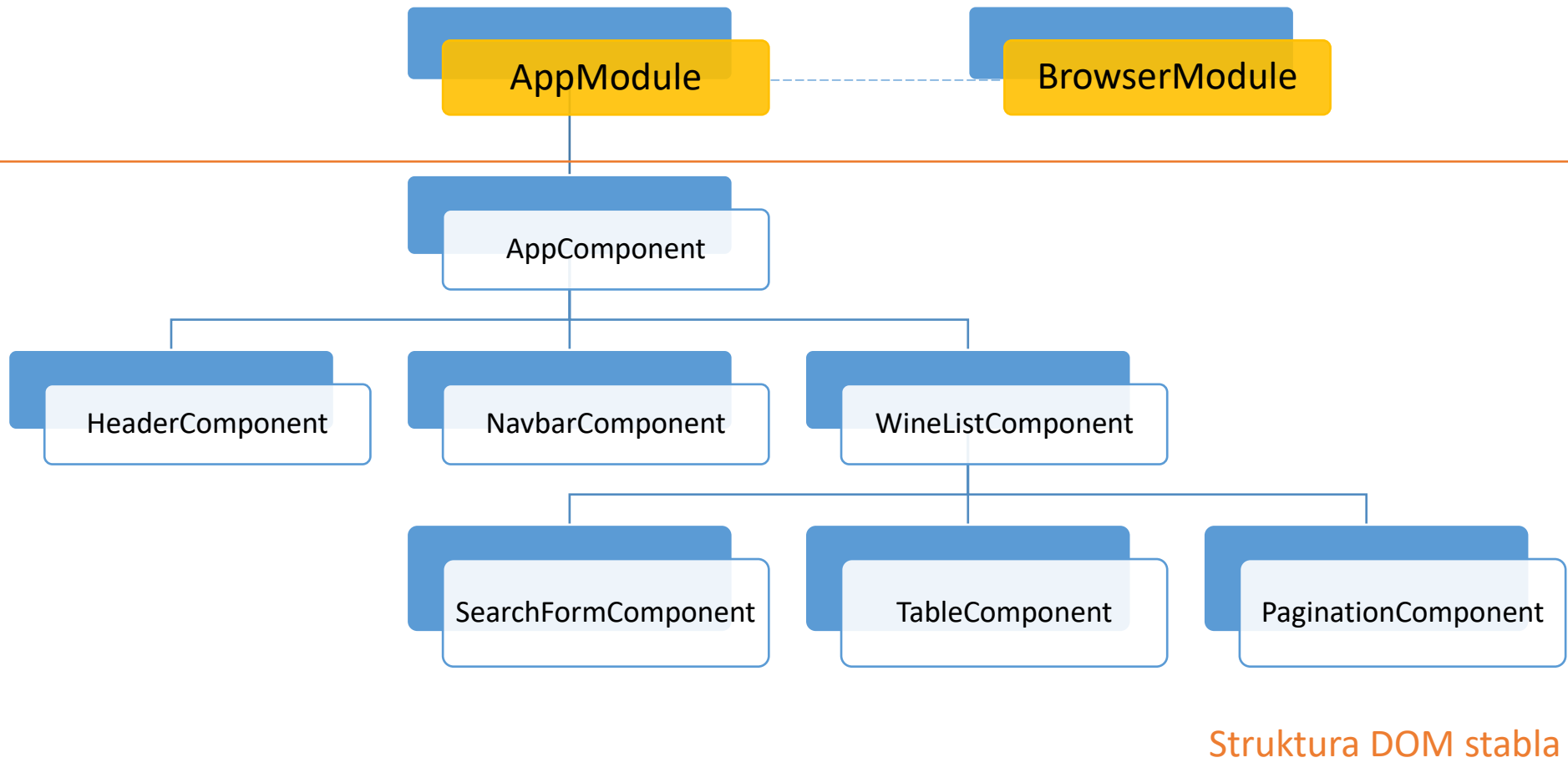
Modul je klasa

Dekorator klase

export – ovaj modul možemo importovati u druge module

Definicija *AppModule* se tipično nalazi u `app.module.ts` fajlu

Trenutna struktura *wine-cellar* aplikacije



Input komponente i direktive

- Interpolacija {{ }}
- Input
- NgFor

Interpolacija {{ izraz }} – primer_2/primer-interpolacije

- Interpolacija : deo HTML-a označen sa {{ izraz }} se zamenjuje vrednošću izraza navedenog unutar zagrada

```
<p>
  Racunamo vrednost izraza 555/33: {{555/33}}
</p>
```

➡ Racunamo vrednost izraza 555/33: 16.818181818181817

- Možemo prikazati i vrednost promenljivih:

```
private title :string = 'Primer interpolacije';
private a :number = 5;
private b :number = 3;
```

```
<h1>{{title}}</h1>
```

 ➡ Primer interpolacije

```
<p>
  {{a}} + {{b}} = {{a+b}}
</p>
```

 ➡ 5 + 3 = 8

Polje *title* je tipa *string* pa možemo koristiti polje *length* koje postoji na promenljivama tipa *string*

```
<p>
  Ovaj naslov sadrži {{title.length}} karaktera
</p>
```

 ➡ Ovaj naslov sadrži 20 karaktera

Interpolacija `{{ izraz }}` – primer_2/primer-interpolacije

- Izraz koji nije definisan dobija vrednost `undefined` ili `null`. Ovakav izraz se neće prikazati:

```
<p>  
  Polje c ne postoji na komponenti AppComponent (ima vrednost undefined). Zato se ovo ne  
  prikazuje: {{c}}  
</p>
```

Polje c ne postoji na komponenti AppComponent (ima vrednost undefined). Zato se ovo ne prikazuje:

- Izraz se interpretira kao string (a ne promenljiva) ako stavimo navodnike:

```
<p>  
  Ovako se prikazuje vrednost promenljive this.a: {{a}}.  
  A ovako (sa navodnicima) string 'a':| {{'a'}}  
</p>
```

Ovako se prikazuje vrednost promenljive this.a: 5. A ovako (sa navodnicima) string 'a': a

Izrazi – primer_2/primer-interpolacije

- Možemo koristiti *if* izraze:

```
<p>  
  Ako je this.a>=0 prikazace se string positive, a u suprotnom negative  
  {{a >= 0? 'positive' : 'negative'}}  
</p>
```

Ako je this.a>=0 prikazace se string positive, a u suprotnom negative positive

- Možemo prikazati povratnu vrednost metode:

(metoda u *AppComponent*)

```
subtract(x :number, y :number) :number{  
  return x - y;  
}
```

```
<p>  
  {{a}} - {{b}} = {{subtract(a, b)}}  
</p>
```

→ 5 - 3 = 2

Prosledili smo this.a i this.b kao parametre

Izrazi – primer_2/primer-interpolacije

- Interpolacioni izrazi se mogu staviti gotovo bilo gde u HTML-u:
 - **Unutar HTML taga** `<div> {{ izraz }} </div>`
 - Kao što smo videli na dosadašnjim primerima
 - **Vrednost atributa** `<div att = {{ izraz }}></div>`

```
<p>  
<input type="checkbox" checked="{{checked}}">  
</p>
```



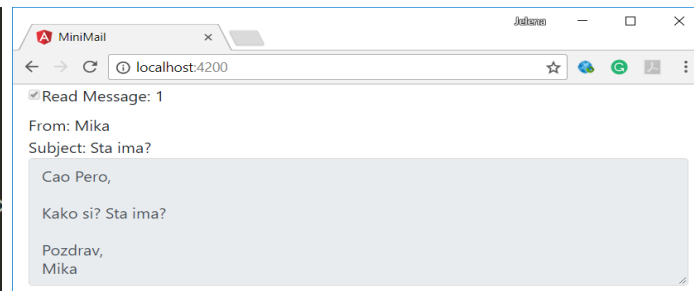
Polje `this.checked` sa vrednošću `true`

- Još primera upotrebe interpolacije naći ćete u [primer_3/mini-mail_faza1](#)

Input komponente

- Želimo da naše komponente budu *reusable* (da ih možemo koristiti na više mesta u aplikaciji)
- Razmotrimo komponentu *EmailComponent* iz [primer_3/mini-mail_faza1](#):

```
<div class="row">
  <div class="col">
    <label class="checkbox-inline"><input type="checkbox" checked="{{email.read}}"
    disabled>Read</label>
    Message: {{email.timeStamp}}
    <div>From: {{email.from}}</div>
    <div>Subject: {{email.subject}}</div>
    <textarea readonly |class="form-control" rows="6" id="comment" value="{{email.body}}">
    </textarea>
  </div>
</div>
```



- Ako podatke o prikazivanom emailu (vrednost polja *email*) „zabetoniramo“ u klasi *EmailComponent*, za prikaz novog maila morali bismo da pravimo novu komponentu. Ovo je očigledno loše rešenje!
- Bolje rešenje bi bilo da vrednost polja *email* bude prosleđena na ulaz komponente – bude „parametar“ koji možemo proslediti
- Tako bismo za prikaz različitih emailova jednostavno prosleđivali različite vrednosti na ulaz komponente
- Ovo je moguće uraditi pomoću *@Input* dekoratora

Input komponente – primer_3/mini-mail_faza2

- Ako ispred polja dodamo `@Input()`, time naglašavamo da će vrednost tog polja biti prosleđena komponenti na ulazu (prilikom njene inicijalizacije)
- Na slici je prikazano kako je polje *email* označeno kao polje čija vrednost će biti prosleđena komponenti na ulazu (c)
- Ovaj *Input* dekorator je definisan u modulu '@angular/core' pa ga moramo importovati (na slici b)

```
import { Component, OnInit, Input } from '@angular/core';
import { Email } from '../model/email'; b

@Component({
  selector: 'mm-email',
  templateUrl: './email.component.html',
  styleUrls: ['./email.component.css']
})
export class EmailComponent implements OnInit {
  c @Input() private email :Email;

  constructor() {}

  ngOnInit() {}
}
```


Odakle se prosleđuje vrednost?

primer_3/mini-mail_faza2

- Vrednost se prosleđuje iz roditeljske komponente na sledeći način:
`<selektor-child-komponente [ime_input_polja]="prosleđena_vrednost">`
- Koja komponenta je „roditeljska“ određuje struktura HTML-a

U templejt *ReceivedEmailsComponent* se koristi selektor *EmailComponent* (`<mm-email>`) pa je *ReceivedEmailsComponent* parent (roditeljska) komponenta, a *EmailComponent* child (dete) komponenta

```
<mm-email [email]="receivedEmails[0]"></mm-email>
<mm-email [email]="receivedEmails[1]"></mm-email>
<mm-email [email]="receivedEmails[2]"></mm-email>
```

email je polje iz *EmailComponent* označeno dekoratorom *@Input*. Naziv polja se mora poklapati sa ovde navedenim nazivom

receivedEmails je polje iz *ReceivedEmailsComponent* koja sadrži listu mailova. Ovde prikazujemo jedan email iz te liste

Direktiva NgFor

- Omogućava da prolazimo kroz kolekciju objekata (slično *for* petlji) i za svaki objekat iz kolekcije napravimo jedan element u DOM stablu
- Upotreba:

<element_koji_se_ponavlja *ngFor="let *brojač* of *kolekcija*">

brojač možemo nazvati kako želimo. Ima ulogu *i* u for(let i=1;...)

Mora biti definisana u komponenti u čijem se templateju ovo nalazi (mora postojati *this.kolekcija*)

Direktiva NgFor – primer_3/mini-mail_faza3

- Umesto

```
<mm-email [email]="receivedEmails[0]"></mm-email>
<mm-email [email]="receivedEmails[1]"></mm-email>
<mm-email [email]="receivedEmails[2]"></mm-email>
```

- Možemo staviti

```
<mm-email
  *ngFor="let receivedEmail of receivedEmails"
  [email]="receivedEmail">
</mm-email>
```

receivedEmail – brojač koji redom dobija vrednosti elemenata liste
this.receivedEmails definisane u *ReceivedEmailsComponent* (čiji je ovo templejt)

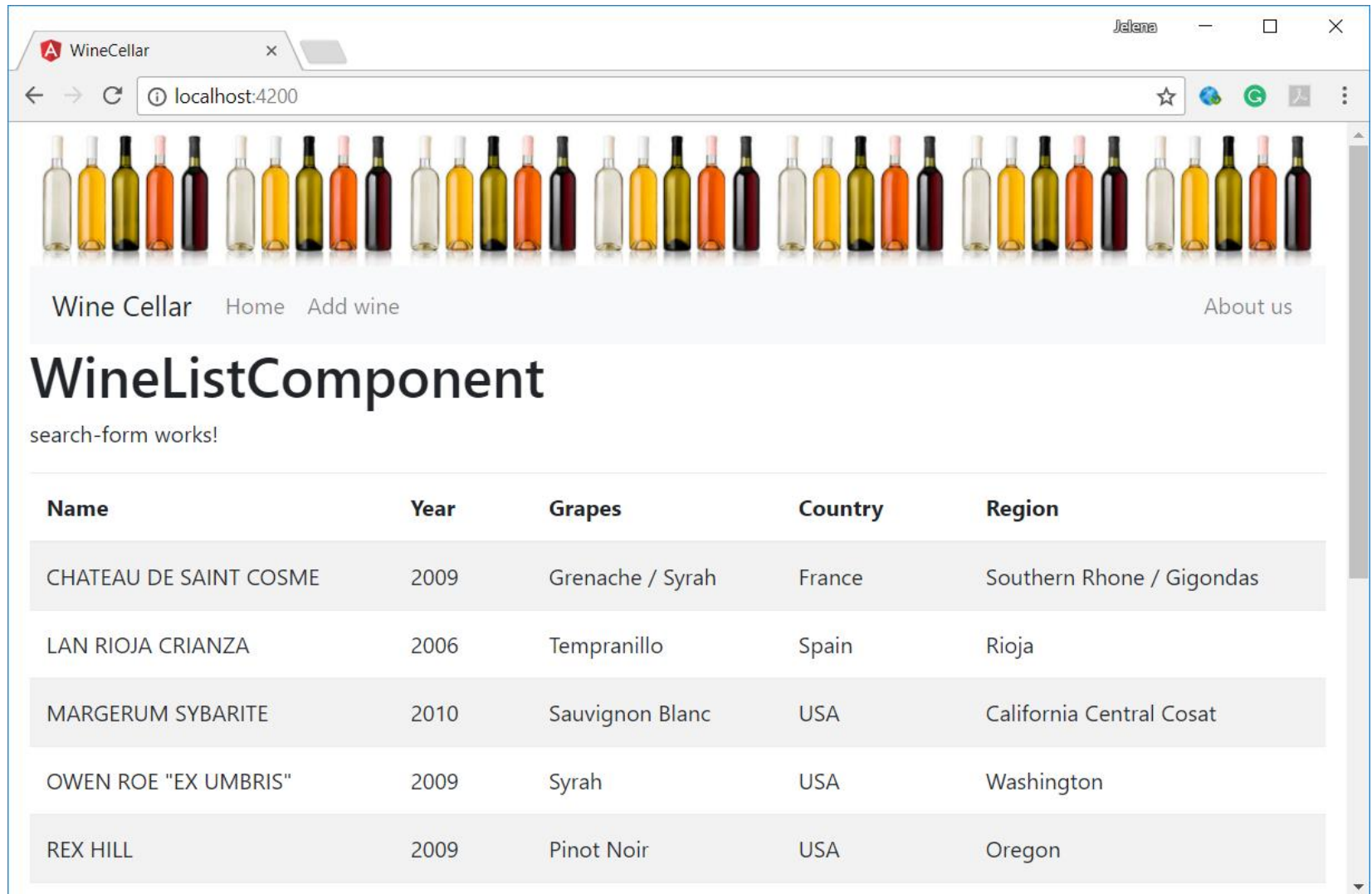
Umesto *receivedEmail* smo mogli pisati *pera* – jedina modifikacija bi bila da moramo promeniti naziv promenljive koja se prosleđuje *EmailComponent*:
[email]="pera"

Zadatak 5

Proširiti wine-cellar aplikaciju iz zadatka 4.

- a) Sa zadatkom vam je data datoteka *wines.ts* u kojoj je definisana konstanta *WINES* koje sadrži podatke o vinima. Po uzoru na ove podatke, u datoteci *src/app/wine/model/wine.model.ts* napraviti klasu *Wine* koja predstavlja jedno vino
- b) U komponenti *WineListComponent* napravite polje koje će da sadrži niz vina. Inicijalizujte ovaj niz vina u konstruktoru (kopirajte *WINES* konstantu u *WineListComponent* i iskoristite je u konstruktoru da popunite polje)
- c) Iz komponente *WineListComponent* proslediti niz vina komponenti *TableComponent*
- d) U *TableComponent*, primljeni niz vina prikazati u tabeli

Zadatak 5 – željeni izgled



Model podataka

- Zbog čega smo pravili klasu *Wine*? Zašto nismo direktno koristili *JavaScript* objekte iz konstante *WINES* već prepakivali u niz *Wine* objekata?
- Generalno je dobra ideja da se za napravi model za podatke sa kojima ćemo raditi jer se on brine da ti podaci budu konzistentni tako što im nameće određena ograničenja
 - Ne može se desiti da imamo vino sa nekim dodatnim poljima (koja tu ne pripadaju)
 - Možemo ograničiti da vino mora da sadrži određena polja
 - Možemo nametnuti ograničenja na sadržaj polja (npr. da godina ne može biti negativan broj)
 - Potrebno nam je zbog *TypeScripta* (da zna da su data polja definisana na objektu pa se ne buni kada pokušavamo da im pristupimo)

Motivacija za uvođenje Angulara

Zbog čega Angular (a ne čist JavaScript)?

- Pomoću **HTML** (sadržaj), **CSS** (izgled), **JS** (funktionalnost) tehnologija možemo izgraditi kompletan veb sajt

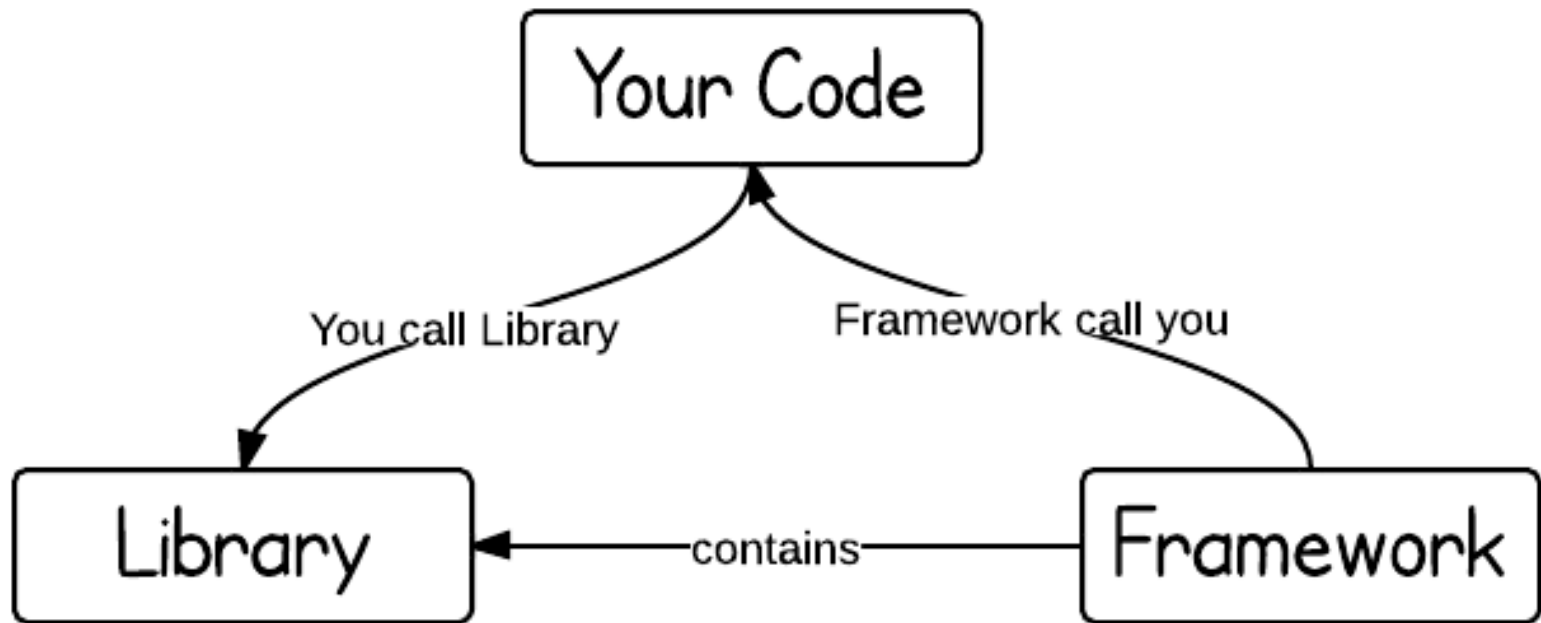


- Međutim, izgradnja *kompleksnog* veb sajta bi bila prilično teška:
 - JavaScript se koristi u svim delovima aplikacije
 - korisnički interfejs, klijent-server komunikacija, implementacija poslovne logike, validacija korisničkog unosa,...
 - Sa porastom kompleksnosti aplikacije, dramatično raste količina stvari o kojima mi (ručno) moramo da vodimo računa

Library vs. Framework

- Biblioteka (*library*)
 - Kolekcije funkcija koje implementiraju određeno ponašanje
 - Primer biblioteke je **jQuery**
 - Ovde mi pozivamo funkcije, mi odlučujemo šta se prosleđuje kao parametar, gde ih koristimo, itd.
- Radni okvir (*framework*)
 - Temelj nad kojim efikasno razvijamo aplikaciju
 - Primeri radnog okvira su **Angular**, **Ember**, **Vue**,...
 - Diktira način organizacije našeg koda i poziva ga spram svojih pravila
 - Daje „kostur“ aplikacije (bazičnu funkcionalnost) koju mi popunjavamo „mesom“ (konfiguriramo)

Library vs. Framework



Dodatni resursi

- <https://www.freecodecamp.org/>
- <https://www.coursera.org/learn/angular/>
- <https://www.codeschool.com/courses/discover-devtools>

Zadatak 6 – Aplikacija kućni-ljubimci

KućniLjubimci

localhost:4200

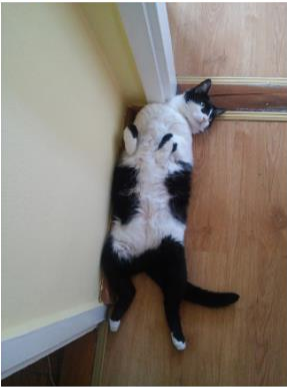
Glavna macke psi ptice glodari ribe reptili

Godine:

Datum objave:

Pol:

Rasa:

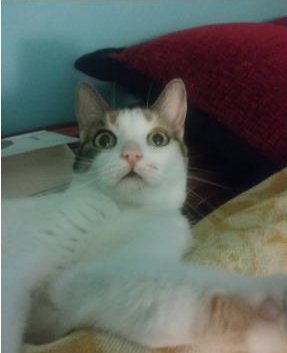


Veljko-Debeljko

Generalije
Godine: 8
Pol: f
Rasa: Evropska kratkodlaka macka
Datum objave: Jul 3, 2017

Hobiji

- Grickanje kutija
- Gledanje kroz prozor
- Suncanje stomaka
- Sedenje iza kreveta
- Gledanje snimaka sa pticama na youtube-u
- Boks, slobodni sparing sa Lazom-mazom



Laza-maza

Generalije
Godine: 9
Pol: f
Rasa: Evropska kratkodlaka macka
Datum objave: May 2, 2017

Hobiji

- Mazenje
- Vijanje loptice
- Setnja ispred monitora
- Prosnja ljudske hrane koju ne smem da jedem
- Grickanje mreznih kablova
- Boks, slobodni sparing sa Veljkom-debeljkom

- U datoteci *pets.ts* se nalazi konstanta *PETS* sa podacima o kućnim ljubimcima
- U datoteci *pettypes.ts* se nalazi konstanta *PET_TYPES* sa tipovima kućnih ljubimaca (macka, pas,...) pomoću koje treba da napravite navigacionu traku
- Slike kućnih ljubimaca su date u *img* direktorijumu
- Za prikaz datuma koristiti *pipe* koji se zove *date* <https://angular.io/api/common/DatePipe>