



IT255 - VEB SISTEMI 1

Uvod u Angular veb aplikacije

Lekcija 05

PRIRUČNIK ZA STUDENTE

IT255 - VEB SISTEMI 1

Lekcija 05

UVOD U ANGULAR VEB APLIKACIJE

- ✓ Uvod u Angular veb aplikacije
- ✓ Poglavlje 1: Prva Angular aplikacija - priprema
- ✓ Poglavlje 2: Instrukcije za Windows korisnike
- ✓ Poglavlje 3: Pisanje koda Angular aplikacije
- ✓ Poglavlje 4: Rad sa nizovima
- ✓ Poglavlje 5: Komponenta potomak
- ✓ Poglavlje 6: Komponente Angular aplikacije
- ✓ Poglavlje 7: Pokazni primer - proširenje aplikacije
- ✓ Poglavlje 8: Objavljivanje aplikacije
- ✓ Poglavlje 9: Vežbe 5
- ✓ Poglavlje 10: Domaći zadatak 5
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

AngularJS je JavaScript MVC framework napravljen od strane Google-a.

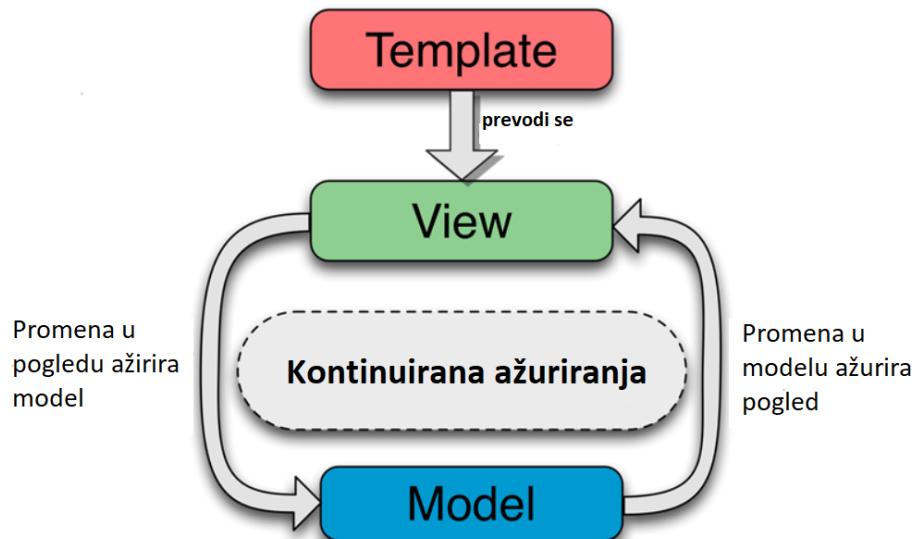
AngularJS je JavaScript MVC okvir (framework) napravljen od strane Google-a koji omogućava veb programerima kreiranje dobro strukturiranih, lакih za testiranje i održavanje front-end aplikacija.

AngularJS je MVC framework koji definiše brojne koncepte za pravilno organizovanje veb aplikacije koja se kreira.

Aplikacija je definisana uz pomoć modula koji mogu biti zavisni jedni od drugih. AngularJS proširuje HTML pružajući direktive oznakama pomoću kojih je moguće praviti moćne i dinamičke stranice. Takođe moguće je napraviti sopstvene direktive pomoću kojih će se vršiti DOM manipulacije. Takođe implementira dvosmerno vezivanje podataka, vezujući HTML (poglede) sa JavaScript objektima (modelom) na veoma lak način.

Jednostavno rečeno to znači da će se bilo kakva promena na modelu smesta reflektovati na view strani bez potrebe za bilo kakvom DOM manipulacijom ili rukovanjem događaja.

Angular je vrlo fleksibilan što se tiče serverske komunikacije. Kao i većina JavaScript okvira dopušta vam rad sa bilo kojom server-side tehnologijom dokle god može da opslužuje aplikaciju preko RESTful API-ja.



Slika-1 Dvosmerno povezivanje podataka

✓ Poglavlje 1

Prva Angular aplikacija - priprema

KREIRANJE JEDNOSTAVNOG REDDIT KLONA

Lekcija je opredeljena da najznačajnije koncepte pokazuje na konkretnim primerima.

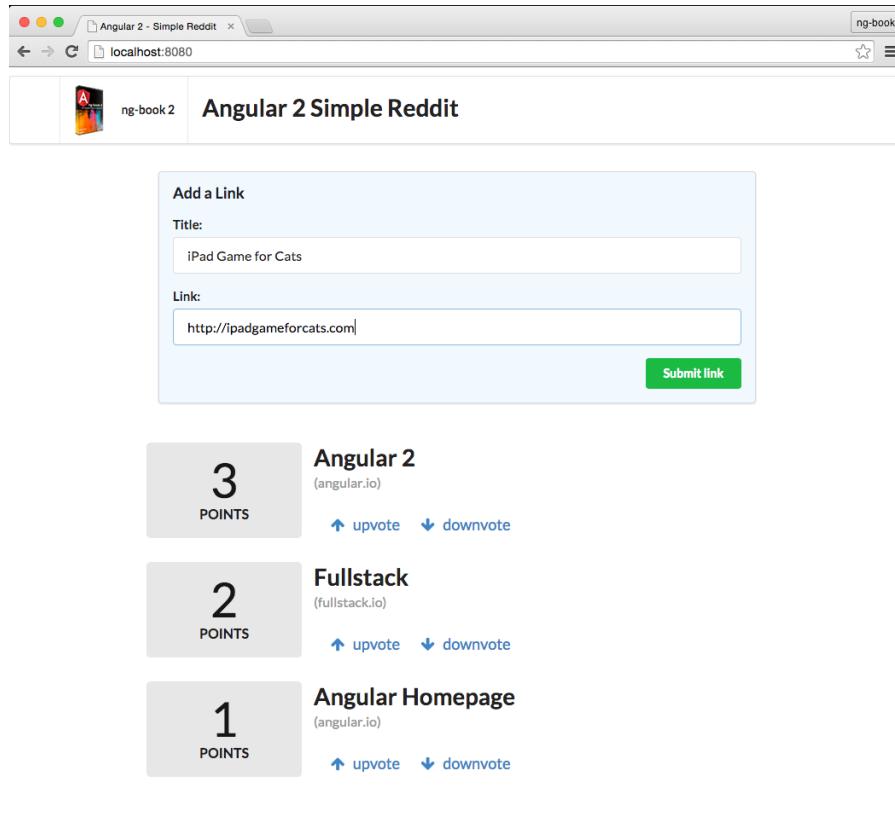
U ovoj lekciji cilj će biti kreiranje aplikacije koja će omogućiti korisnicima postave članak (zajedno sa nazivom i linkom) i da, nakon toga, mogu da glasaju (pozitivno ili negativno) za ostale postove. Aplikacija će biti pojednostavljena verzija dobro poznatih veb aplikacija poput *Reddit* ili *Product Hunt*.

Lekcija je opredeljena da najznačajnije koncepte pokazuje na konkretnim primerima. Otuda, u ovoj jednostavnoj aplikaciji biće neophodno pokriti najznačajnije Angular koncepte, poput:

- izgradnje vlastitih komponenata;
- kreiranje formi i prihvatanje unosa sa istih;
- kreiranje pogleda iz liste objekata;
- presretanje interakcije korisnika i GUI komponenata i rukovanje njima;
- angažovanje aplikacije na veb serveru.

Ideja je da kada student završi izučavanje ove lekcije moći će da iskoristi prazan folder, kreira osnovnu Angular aplikaciju, a zatim je angažuje na serveru i pokrene. Upravo ovakva aplikacija će predstavljati osnov za razvoj iskustva iz kojeg će student moći da kreira veći broj različitih front-end aplikacija koristeći okvir *Angular*.

Sledećom slikom je prezentovan mogući izgled aplikacije čije su osnove izložene u prethodnom izlaganju.

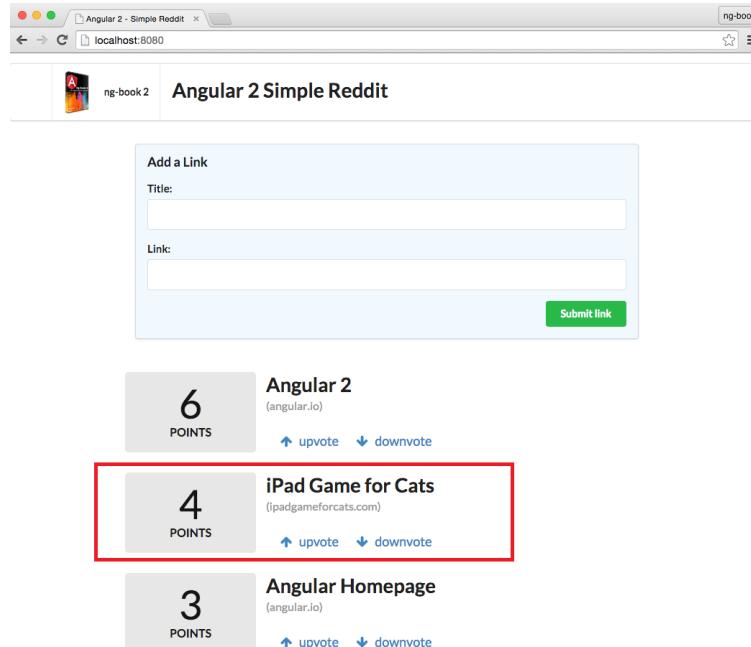


Slika 1.1 Predloženi izgled Angular aplikacije

FORMA I TYPESCRIPT

TypeScript predstavlja nadskup za JavaScript ES6 sa dodatkom tipova podataka.

U aplikaciji se pojavljuje koncept bez kojeg nije moguće zamisliti nijednu ozbiljnu veb aplikaciju - forma. Preko forme će korisnik moći da doda u aplikaciju novi link za kojeg će ostali korisnici moći da glasaju (pozitivno ili negativno). Pored svakog linka će stajati ocena koja predstavlja utisak korisnika o korisnosti sadržaja koji se krije iza linka. Dodati link je prikazan sledećom slikom.



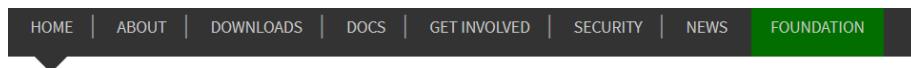
Slika 1.2 Dodavanje podataka preko forme

Kroz ovaj projekat, a i u daljem izlaganju u ovom predmetu, biće korišćen [TypeScript](#). [TypeScript](#) predstavlja nadskup za JavaScript ES6 sa dodatkom tipova podataka. O ovoj notaciji će biti detaljno diskutovano u lekcijama koje slede. Studenti će bez problema moći da prate lekciju ukoliko su savladali sintaksu baznog JavaScript jezika.

NODE.JS I NPM

Neophodno je instaliranje podrške izvršavanju JavaScript instrukcija u formi Node.js.

Da bi bilo moguće raditi bilo šta sa Angular okvirom, za početak, neophodno je instaliranje podrške izvršavanju JavaScript instrukcija u formi serverskog okruženja [Node.js](#). Postoji nekoliko različitih načina da se instalira Node.js i svi su detaljno objašnjeni na sajtu <https://nodejs.org/en/>.



Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

Download for Windows (x64)

[10.16.1 LTS](#)

Recommended For Most Users

[12.7.0 Current](#)

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#) [Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [Long Term Support \(LTS\) schedule](#).

Sign up for [Node.js Everywhere](#), the official Node.js Monthly Newsletter.

Slika 1.3 Node.js sajt za preuzimanje instalacije

Uvek bi bilo dobro preuzeti i instalirati poslednju stabilnu verziju za Node.js i preporuka je da to svakako bude verzija 8.9.0 ili neka novija. Sa slike je moguće videti da je poslednja stabilna verzija 10.16.1.

Dalje, za rad je takođe potreban i [npm](#) (*Node Package Manager*) koji bi trebalo da bude instaliran kao deo Node.js paketa. Da bi bila ovavljena provera uspešnog instaliranja [npm](#) - a kao dela razvojnog okruženja, neophodno je otvoriti *command prompt* i u terminalu otkucati sledeću naredbu:

```
$ npm -v
```

Ukoliko nije prikazan broj verzije ili je primljena greška, neophodno je proveriti da li je instaliran Node.js koji uključuje npm. Verzija bi trebalo da bude 5.6.0 ili novija.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

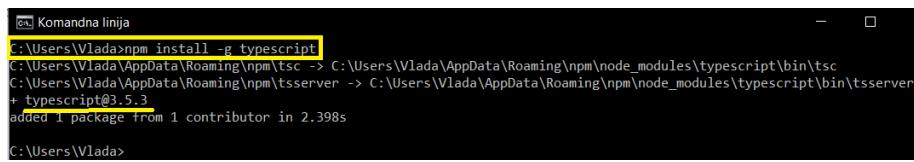
TYPESCRIPT I PREGLEDAČ - PODEŠAVANJE

Za dalji rad je neophodno instalirati TypeScript podršku na radni računar i izabrati pregledač.

Za dalji rad je neophodno instalirati TypeScript podršku na radni računar. Ukoliko je sve prethodno navedeno ispoštovano (instaliran Node.js), u command promt-u, na komandnoj liniji, neophodno je uneti sledeću naredbu:

```
npm install -g typescript
```

Ukoliko je dobijen izlaz kao na sledećoj slici, sve je dobro obavljeno.



```
C:\ Komandna linija
c:\Users\Vlada>npm install -g typescript
C:\Users\Vlada\AppData\Roaming\npm\tsc -> C:\Users\Vlada\AppData\Roaming\npm\node_modules\typescript\bin\tsc
C:\Users\Vlada\AppData\Roaming\npm\tsserver -> C:\Users\Vlada\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
+ typescript@3.5.3
added 1 package from 1 contributor in 2.398s
C:\Users\Vlada>
```

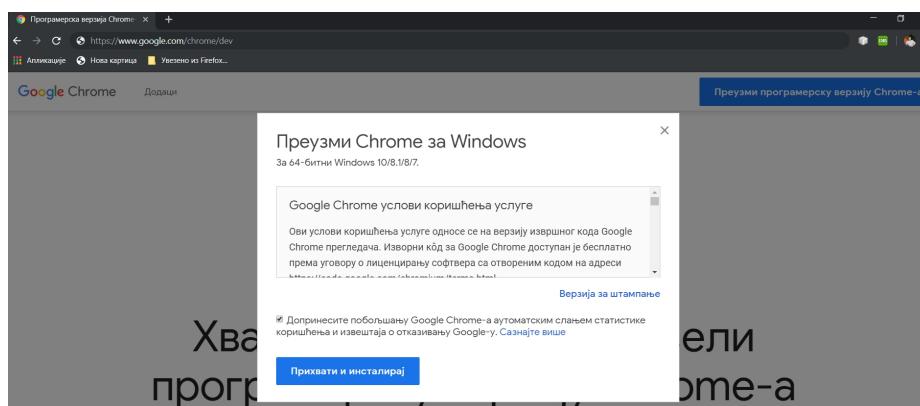
Slika 1.4 Instalacija TypeScript podrške

Takođe, ovde je neophodno voditi računa o verziji, morala bi da bude 2.1 ili novija (ovde je konkretno 3.5.3).

Za pokretanje i testiranje kreiranih veb aplikacija pomoću Angular okvira, Google preporučuje vlastiti Google Chrome veb pregledač. Takođe, preporuka je instaliranje programerske verzije ovog pregledača, a to je omogućeno ako u standardnom pregledaču navedete link:

<https://www.google.com/chrome/dev>

Sve što je potrebno u nastavku jeste da pratite uputstva koja vam nudi čarobnjak za instalaciju prikazan na sledećoj slici.



Slika 1.5 Chrome developer toolkit instalacija

▼ Poglavlje 2

Instrukcije za Windows korisnike

ANGULAR CLI

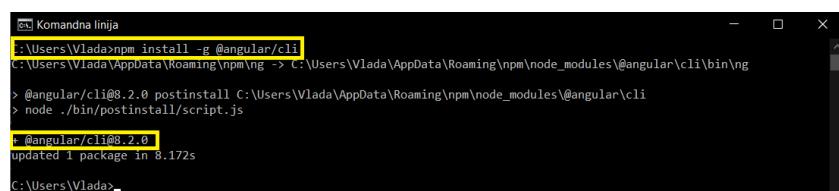
Angular CLI dozvoljavaju korisnicima kreiranje i upravljanje projektima sa komandne linije.

Angular obezbeđuje alate koji dozvoljavaju korisnicima kreiranje i upravljanje projektima sa komandne linije. Ovi alati su univerzalni za različite platforme i nazivaju se [Angular CLI](#). Zapravo kreiranje novih projekata, dodavanje kontrolera i ostalih komponenata Angular aplikacije primenom Angular CLI alata predstavlja jako dobru ideju. Ovi alati omogućavaju, takođe, kreiranje i održavanje različitih šabloni koji se koriste u aplikaciji.

Za instaliranje alata [Angular CLI](#) dovoljno je u komandnoj liniji Command Prompt - a navesti naredbu:

```
npm install -g @angular/cli
```

Ako je rezultat izvršavanja navedene naredbe identičan kao na sledećoj slici, sav posao instalacije Angular CLI je uspešno obavljen.



```
Administrator: Komanda linija
C:\Users\Vlada>npm install -g @angular/cli
C:\Users\Vlada\AppData\Roaming\npm\ng -> C:\Users\Vlada\AppData\Roaming\npm\node_modules\@angular\cli\bin\ng
> @angular/cli@8.2.0 postinstall C:\Users\Vlada\AppData\Roaming\npm\node_modules\@angular\cli
> node ./bin/postinstall/script.js
+ @angular/cli@8.2.0
updated 1 package in 8.172s
C:\Users\Vlada>
```

Slika 2.1 Angular CLI instalacija za Windows operativni sistem

Nakon uspešne instalacije, programeri imaju mogućnost korišćenja naredbe [ng](#) na komandnoj liniji. pozivom ove naredbe dobija se detaljan izlaz sa podacima u vezi sa Angular CLI. Na primer, pozivom:

```
ng --version
```

dobija se izlaz kao na sledećoj slici.

```
C:\Users\Vlada>ng --version

Angular CLI: 8.2.0
Node: 10.16.1
OS: win32 x64
Angular:
...
Package          Version
-----
@angular-devkit/architect    0.802.0
@angular-devkit/core         8.2.0
@angular-devkit/schematics   8.2.0
@schematics/angular          8.2.0
@schematics/update           0.802.0
rxjs                         6.4.0

C:\Users\Vlada>
```

Slika 2.2 Instalirana Angular CLI verzija

Takođe, pozivom: `ng --help` moguće je dobiti i dodatne informacije.

PRIMER JEDNOSTAVNOG ANGULAR PROJEKTA

Moguće je pristupiti kreiranju prve Angular aplikacije.

Budući da su sva podešavanja za rad sa Angular okvirom uspešno obavljena, moguće je pristupiti kreiranju prve Angular aplikacije. U tu svrhu je moguće koristiti novu `ng` naredbu koju je neophodno navesti u terminalu razvojnog okruženja (preporuka je Visual Studio Code). Konkretno, naredba glasi:

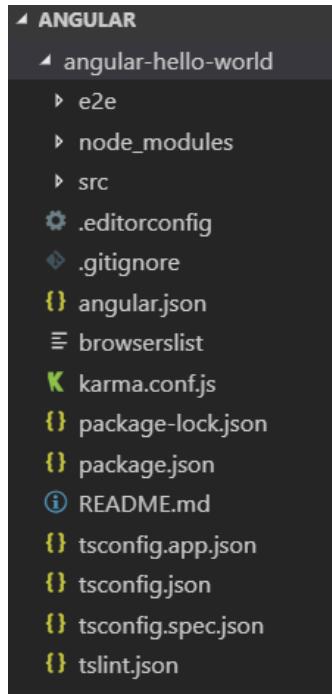
```
ng new angular-hello-world
```

Izvršavanjem naredbe kreira se izlaz prikazan sledećom slikom.

```
PS C:\Users\vlada\Documents\Angular> ng new angular-hello-world
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? stylus [ http://stylus-lang.com ]
CREATE angular-hello-world/angular.json (3785 bytes)
CREATE angular-hello-world/package.json (1293 bytes)
CREATE angular-hello-world/README.md (1034 bytes)
CREATE angular-hello-world/tsconfig.json (543 bytes)
CREATE angular-hello-world/tslint.json (1988 bytes)
CREATE angular-hello-world/.editorconfig (246 bytes)
CREATE angular-hello-world/.gitignore (631 bytes)
CREATE angular-hello-world/browserslist (429 bytes)
CREATE angular-hello-world/karma.conf.js (1031 bytes)
CREATE angular-hello-world/tsconfig.app.json (270 bytes)
CREATE angular-hello-world/tsconfig.spec.json (270 bytes)
CREATE angular-hello-world/src/favicon.ico (5430 bytes)
CREATE angular-hello-world/src/index.html (304 bytes)
CREATE angular-hello-world/src/main.ts (372 bytes)
CREATE angular-hello-world/src/polyfills.ts (2838 bytes)
CREATE angular-hello-world/src/styles.styl (80 bytes)
CREATE angular-hello-world/src/test.ts (642 bytes)
CREATE angular-hello-world/src/assets/.gitkeep (0 bytes)
CREATE angular-hello-world/src/environments/environment.prod.ts (51 bytes)
CREATE angular-hello-world/src/environments/environment.ts (662 bytes)
CREATE angular-hello-world/src/app/app-routing.module.ts (246 bytes)
CREATE angular-hello-world/src/app/app.module.ts (393 bytes)
CREATE angular-hello-world/src/app/app.component.html (1152 bytes)
CREATE angular-hello-world/src/app/app.component.spec.ts (1134 bytes)
CREATE angular-hello-world/src/app/app.component.ts (224 bytes)
CREATE angular-hello-world/src/app/app.component.styl (0 bytes)
CREATE angular-hello-world/e2e/protractor.conf.js (810 bytes)
CREATE angular-hello-world/e2e/tsconfig.json (214 bytes)
CREATE angular-hello-world/e2e/src/app.e2e-spec.ts (648 bytes)
CREATE angular-hello-world/e2e/src/app.po.ts (251 bytes)
```

Slika 2.3 Kreiranje Angular projekta

Navedeni proces će teći neko vreme dok se sve zavisnosti ne učitaju i neophodni fajlovi kreiraju. U ovom trenutku ne bi trebalo da brine studenta što trenutno ne razume ulogu pojedinih fajlova u aplikaciji - svi će biti detaljno objašnjeni u nastavku. Navedena naredba je kreirala projekat sa sledećom strukturu.

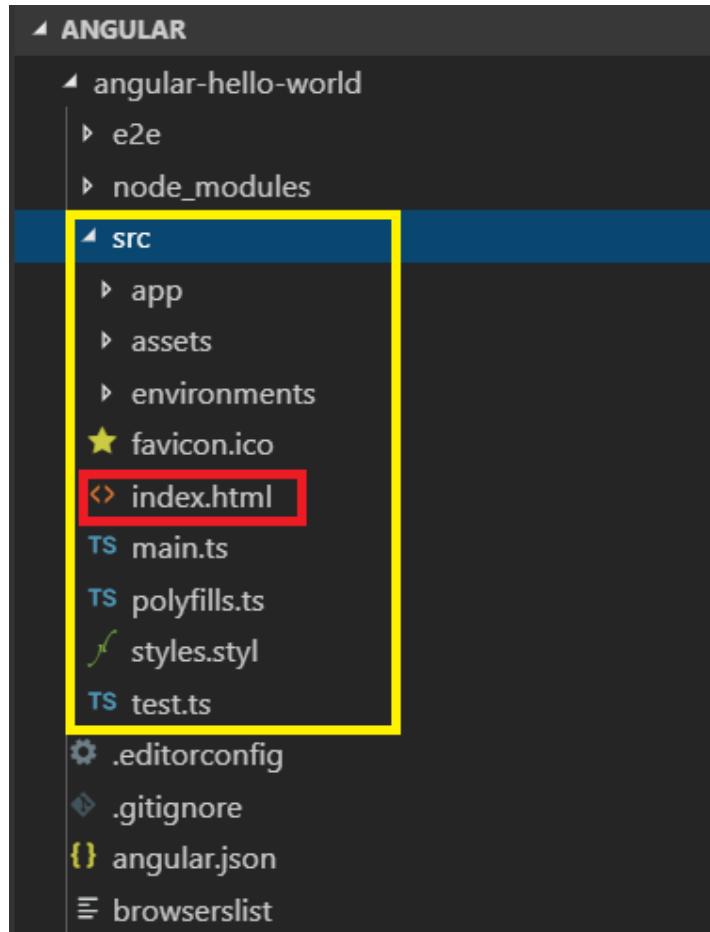


Slika 2.4 Struktura kreiranog projekta

GENERISANI KOD KREIRANOG ANGULAR PROJEKTA

Sve programske datoteke moguće je naći pod čvorom src u stablu projekta.

Kao što je istaknuto u prethodnom izlaganju, prilikom kreiranja aktuelnog projekta generisane su brojne datoteke. Među njima se nalaze i datoteke koje u sebi čuvaju programski kod: **JavaScript**, **HTML** i slično. **Sve programske datoteke moguće je naći pod čvorom src u stablu projekta.** Navedeno je prikazano sledećom slikom.



Slika 2.5 Lokacija programskih datoteka u strukturi projekta

ANALIZA GENERISANE DATOTEKE POGLEDA

Analiza kreirane datoteke index.html.

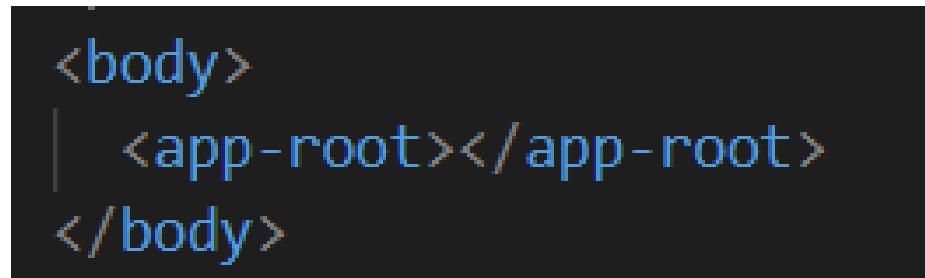
Na prethodnoj slici, crvenom bojom, posebno je istaknuta jedna datoteka. Radi se o datoteci **index.html** koja predstavlja pogled (veb stranicu) u kreiranoj aplikaciji i data je sledećim listingom.

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

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
```

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

Prvi deo koda je poprilično jasan za sve koji su imalo upoznati sa HTML - om. Definisan je set karaktera, naslov i osnovni link za učitavanje pogleda. Posebnu pažnju bi trebalo obratiti na deo koda koji je prikazan sledećom slikom.



Slika 2.6 Segment datoteke index.html.

Prikazani tag `<app-root>` određuje mesto gde će aplikacija biti kreirana na način koji je prezentovan krajnjem korisniku. Ovim tagom je određena komponenta koja je definisana Angular aplikacijom. U Angularu je moguće definisati vlastite HTML tagove i njihove funkcionalnosti. Navedeni tag `<app-root>` će predstavljati "ulaznu tačku" aplikacije na stranici index.html.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 3

Pisanje koda Angular aplikacije

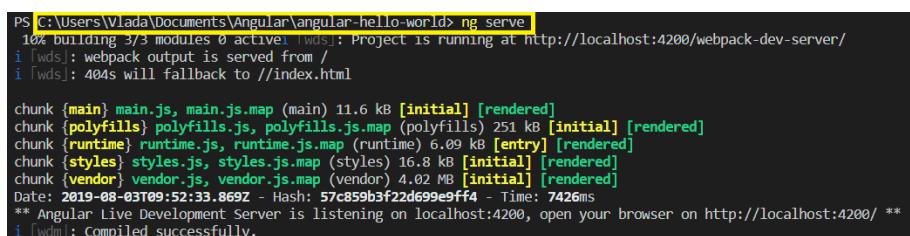
POKRETANJE KREIRANE APLIKACIJE

Neophodno je izvršiti demonstraciju prevođenja i pokretanja Angular aplikacije.

Pre bilo kakve intervencije nad kreiranim datotekama i dodavanja vlastitog programskog koda, neophodno je izvršiti demonstraciju prevođenja i pokretanja Angular aplikacije. U tu svrhu može da posluži i kreirana aplikacija sa osnovnim generisanim kodom. Za kreirani projekat, u terminalu razvojnog okruženja, neophodno je navesti naredbu:

```
cd angular-hello-world  
ng -serve
```

Naredbom **ng - serve** aplikacija se prevodi i angažuje na serveru, a rezultati prevođenja su prikazani sledećom slikom.



```
PS C:\Users\Vlada\Documents\Angular\angular-hello-world> ng serve  
1% building 3/3 modules 0 active [wds]: Project is running at http://localhost:4200/webpack-dev-server/  
i [wds]: webpack output is served from /  
i [wds]: 404s will fallback to //index.html  
  
chunk {main} main.js, main.js.map (main) 11.6 kB [initial] [rendered]  
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 251 kB [initial] [rendered]  
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.09 kB [entry] [rendered]  
chunk {styles} styles.js, styles.js.map (styles) 16.8 kB [initial] [rendered]  
chunk {vendor} vendor.js, vendor.js.map (vendor) 4.02 MB [initial] [rendered]  
Date: 2019-08-03T09:52:33.869Z - Hash: 57c859b3f22d699e9ff4 - Time: 7426ms  
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **  
i [wds]: Compiled successfully.
```

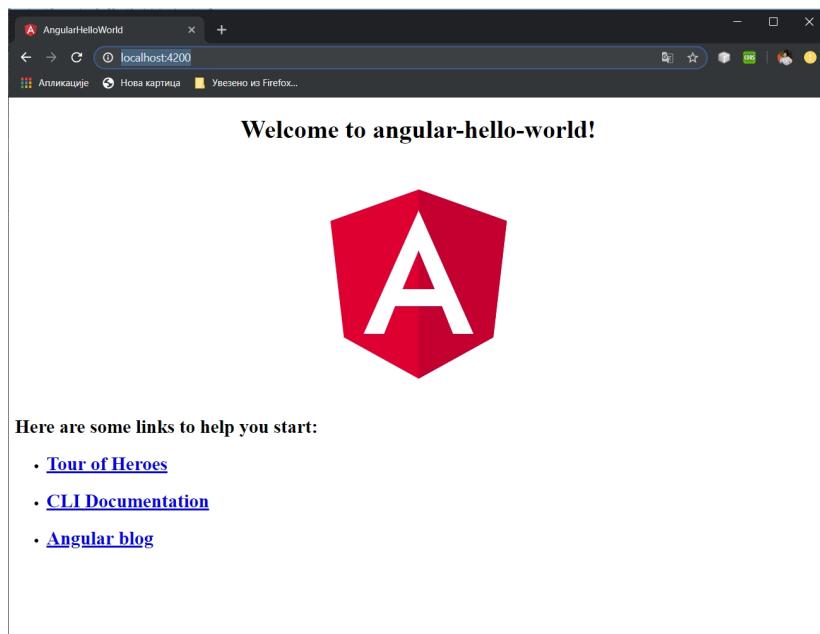
Slika 3.1 Prevođenje i angažovanje Angular aplikacije

Po podrazumevanim vrednostima, a to je moguće sagledati i sa slike, **Angular aplikacije se izvršavaju na portu 4200 na lokalnom računaru (localhost)**. Ukoliko je ovaj port zauzet i aplikaciju nije moguće pokrenuti na njemu, moguće je postupiti drugačiji. Prilikom prevođenja aplikacije navodi se i alternativni port preko kojeg je moguće pokrenuti aplikaciju. To je moguće obaviti sledećom naredbom.

```
ng serve --port 9001
```

Na ovaj način je omogućeno funkcionisanje aplikacije na portu 9001 lokalnog računara.

Sada je moguće otvoriti veb pregledač (**Google Chrome** je predloženi) i unosom linka <http://localhost:4200/> pokreće se kreirana aplikacija, tačnije učitava stranica <index.html>. Navedeno je prikazano sledećom slikom.



Slika 3.2 Početna strana kreirane Angular aplikacije

KREIRANJE KOMPONENTE

Jedna od velikih ideja razvoja Angular okvira je rad sa komponentama.

Jedna od velikih ideja razvoja Angular okvira je rad sa komponentama. U Angular aplikacijama programeri pišu HTML kod koji aplikaciju čini interaktivnom. Međutim, veb pregledač razume ograničen broj HTML tagova. Ugrađeni tagovi, poput `<select>`, `<form>` ili `<video>` poseduju funkcionalnosti koje je definisao proizvođač veb pregledača.

Postavlja se novo pitanje. Da li je moguće naučiti pregledač da izvršava nov tag? Na primer, ukoliko programer želi da primeni tag `<weather>` za prikazivanje podataka o vremenskim prilikama, ili `<login>` za prikazivanje forme za prijavljivanje korisnika, da li će pregledač moći da nauči da koristi nove funkcionalnosti?

Da bi sve bilo jasnije neophodno je pristupiti kreiranju prve Angular komponente. Komponenta će biti dostupna i u HTML dokumentu putem taga:

```
<app-hello-world></app-hello-world>
```

Za kreiranje nove komponente putem Angular CLI koristi se naredba generate, na primer u sledećem obliku:

```
ng generate component hello-world
```

Rezultat izvršavanja ove naredbe moguće je prezentovati sledećom slikom.

```
PS C:\Users\Vlada\Documents\Angular\angular-hello-world> ng generate component hello-world
CREATE src/app/hello-world/hello-world.component.html (26 bytes)
CREATE src/app/hello-world/hello-world.component.spec.ts (657 bytes)
CREATE src/app/hello-world/hello-world.component.ts (289 bytes)
CREATE src/app/hello-world/hello-world.component.styl (0 bytes)
UPDATE src/app/app.module.ts (493 bytes)
PS C:\Users\Vlada\Documents\Angular\angular-hello-world>
```

Slika 3.3 Kreiranje nove Angular komponente

DELOVI KOMPONENTE

Svaka komponenta je sastavljena iz dva dela.

Svaka komponenta je sastavljena iz dva dela:

- dekorator *Component*;
- *klasa definicije komponente*.

Za početak moguće je pogledati inicijalni kod komponente. Komponenta se nalazi u projektu na lokaciji *src/app/hello-world/hello-world.component.ts* i sledećim listingom je priložen njen inicijalni *TypeScript* kod:

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

@Component({
  selector: 'app-hello-world',
  templateUrl: './hello-world.component.html',
  styleUrls: ['./hello-world.component.styl']
})
export class HelloWorldComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

Studentima kod na prvi pogled može da bude zastrašujući. Bez brige, sve će biti analizirano detaljno i korak po korak.

Moguće je primetiti da je ekstenzija ove datoteke .ts umesto .js. Problem je u tome što veb pregledač nezna kako da interpretira TypeScript fajlove. Ovaj problem je jednostavno rešen tako što komandom ng - serve vrši se automatska konverzija .ts fajlova u .js fajlove.

DODAVANJE ZAVISNOSTI

Iskaz Import ukazuje na module koji se koriste prilikom pisanja koda.

Iskaz *Import* ukazuje na module koji se koriste prilikom pisanja koda. Ako se pogleda listing na prethodnoj sekciji moguće je primetiti da se dodaju dve zavisnosti: *Component* i *OnInit*. Uključuje se zavisnost *Component* iz modula `@angular/core`. Ovaj modul ukazuje programu na lokaciju na kojoj može da pronade traženu zavisnost. U konkretnom slučaju, ukazano je kompjajleru da `@angular/core` definiše i eksportuje dva *JavaScript/TypeScript* objekta *Component* i *OnInit*.

Na sličan način, iz istog modula se dodaje komponenta *OnInit*. Ova komponenta će biti posebna tema u nekom od narednih izlaganja i njen zadatak je da pokrene kod prilikom inicijalizovanja komponente.

Neka je dalji predmet analize struktura *Import* naredbi koja može biti prikazana u opštem obliku:

```
import { zavisnost } from modul
```

U delu `{ zavisnost }` obavlja se osobina *destrukturiranja*. Destrukturiranje je osobina obezbeđena putem JavaScript ES6 i TypeScript i o njoj će detaljno biti govora u narednoj lekciji.

COMPONENT DEKORATORI

Nakon dodavanja zavisnosti neophodno je obaviti deklarisanje komponente.

Nakon dodavanja zavisnosti neophodno je obaviti deklarisanje komponente `hello-world.component.ts`. Neka je dat deo koda ove datoteke:

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

Šta se ovde dešava? Ovaj blok se naziva dekoratorom. Dekorator je moguće razumeti kao metapodatke dodate u kod. Kada se klasa, na primer `HelloWorld`, obeleži anotacijom `@Component` može se reći da je kasa dekorisana kao komponenta.

Sada je moguće vratiti se na postavljeni problem korišćenja komponente u HTML kodu putem vlastitog taga. U ovom slučaju primena taga `<app-hello-world>` omogućena je definisanjem selektora unutar dekoratora. Navedeno je prikazano sledećom slikom.

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

Slika 3.4 Definisanje selektora

Sintaksa Angular selektora je slična CSS selektorima iako postoje neke specifičnosti u sintaksi Angular selektora o kojima će detaljno biti govora kasnije. Za sada, dovoljno je shvatiti da se selektorom definiše novi tag kojeg je moguće koristiti u HTML kodu.

Osobine selektor ovde pokazuju koji će DOM element koristiti komponenta. U konkretnom slučaju, tagovi `<app-hello-world></app-hello-world>` se pojavljuju unutar šablona i biće prevedeni preko klase `HelloWorldComponent` i posedovaće bilo koju pridruženu funkcionalnost.

DODAVANJE ŠABLONA PREKO TEMPLATEURL I TEMPLATE

U aktuelnoj komponenti specificiran je šablon pod ključem templateUrl.

U aktuelnoj komponenti specificiran je šablon pod ključem `templateUrl` koji odgovara linku `./hello-world.component.html`. To znači da će šablon biti učitan iz fajla `hello-world.component.html` u istom direktorijumu kao i komponenta. Sledećim listingom priložen je kod koji se nalazi u navedenom fajlu:

```
<p>hello-world works!</p>
```

Na ovaj način definisan je `<p>` tag sa nekim osnovnim tekstom. Kada Angular učita komponentu, takođe čita i iz ovog fajla i koristi ga kao šablon za komponentu.

Šablon može biti specificiran i na drugi način. U ovom slučaju kao ključ u dekorateru se koristi reč `template`, umesto prethodno korišćenje `templateUrl`. To može biti učinjeno na sledeći način:

```
@Component({
  selector: 'app-hello-world',
  template: `
    <p>
      hello-world works inline!
    </p>
  `
})
```

Iz listinga je moguće primetiti da je sting šablon smešten između znakova apostrofa. Ovo je nova Angular funkcionalnost koja omogućava pisanje šablona u formi stringova sa više linija. Na ovaj način se šabloni lakše dodaju u kod.

DODAVANJE CSS PREKO STYLEURLS

Angular koristi koncept poznat pod nazivom enkapsulacija stilova.

Ako se još jednom baci pogled na kod dekoratera `@Component` moguće je primetiti još jedan par (ključ, vrednost) prikazan sledećim kodom:

```
styleUrls: ['./hello-world.component.css']
```

Ovaj deo koda ukazuje da se za komponentu koriste CSS stilovi definisani u datoteci `hello-world.component.css`. Angular koristi koncept poznat pod nazivom *enkapsulacija stilova* što praktično znači da stilovi specificirani za određenu komponentu mogu biti primenjeni samo na tu komponentu. O ovome će biti detaljno govora u posebnoj lekciji. Za sada, neće biti korišćeni nikakvi lokalni stilovi za komponente i ovaj fajl neće biti menjan (ili ključ može biti obrisan u potpunosti).

UČITAVANJE KOMPONENTE

Kada je kompletiran kod komponente, moguće je obaviti njeno učitavanje u konkretnoj stranici.

Sada, kada je kompletiran kod komponente, moguće je obaviti njeno učitavanje u konkretnoj HTML stranici.

Ukoliko se ponovo pokrene aplikacija u pregledaču, neće biti moguće primetiti bilo kakvu promenu. Ovo je iz razloga što je komponenta kreirana ali nije upotrebljena. Sa ciljem realizacije navedenog, neophodno je dodati tag komponente u šablon koji je već kreiran. Za ovo je neophodno otvoriti datoteku `src/app/app.component.html`. Iz razloga što je komponenta `HelloWorldComponent` podešena preko selektora `app-helloworld` može biti korišćena u šablonu. Neka bude dodat tag `<app-hello-world>` u fajl `app.component.html`. Pre toga biće obrisan celokupan inicijalni kod ove datoteke. Novi listing datoteke `app.component.html` izgleda ovako:

```
<h1>
  {{title}}
<app-hello-world></app-hello-world>
</h1>
```

Sada je moguće osvežiti stranicu `index.html` i biće prikazan sadržaj kao na sledećoj slici.



angular-hello-world

hello-world works!

Slika 3.5 Osvežena stranica sa novim sadržajem

DODAVANJE PODATAKA U KOMPONENTU

Aplikacija dobija mogućnost prikazivanja liste korisnika i prikazuju se njihova imena.

Aplikacija, koja predstavlja pokazni primer za trenutno izlaganje, koristi komponente koje kreiraju statičke šablone, a to i nije preterano zanimljivo.

U nastavku, razmišljanje može da ide u drugom pravcu. Aplikacija dobija mogućnost prikazivanja liste korisnika i prikazuju se njihova imena. Pre kreiranja navedene liste neophodno je obaviti kreiranje pojedinačnih korisnika. U tu svrhu biće kreirana komponenta čiji će zadatak biti prikazivanje imena korisnika.

Da bi navedeno bilo obavljeno, u terminalu je neophodno navesti naredbu:

```
ng generate component user-item
```

Rezultat izvršavanja naredbe je prikazan sledećom slikom.

```
PS C:\Users\Vlada\Documents\Angular\angular-hello-world> ng generate component user-item
CREATE src/app/user-item/user-item.component.html (24 bytes)
CREATE src/app/user-item/user-item.component.spec.ts (643 bytes)
CREATE src/app/user-item/user-item.component.ts (281 bytes)
CREATE src/app/user-item/user-item.component.styl (0 bytes)
UPDATE src/app/app.module.ts (585 bytes)
PS C:\Users\Vlada\Documents\Angular\angular-hello-world>
```

Slika 3.6 Kreiranje komponente user-item

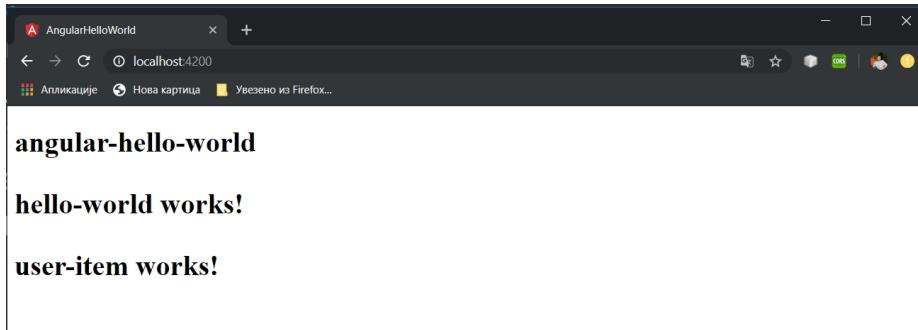
Da bi kreirana komponenta postala vidljiva, neophodno je obaviti njeno dodavanje u šablon. Ovo se obavlja dodavanjem taga `<app-user-item>` u šablon `app.component.html`. Izmenjen sadržaj datoteke prikazan je sledećim listingom:

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

```
<app-hello-world></app-hello-world>

<app-user-item></app-user-item>
</h1>
```

Osvežavanjem početne stranice proverava se da li je sve do sada obavljeno kako treba. Ako jeste, dobiće se izlaz kao na sledećoj slici.



Slika 3.7 Nova komponenta u šablonu

KODIRANJE KLASE KOMPONENTE

Moguće je pristupiti kodiranju klase komponente.

Pošto je sve spremno za da dalji rad, moguće je pristupiti kodiranju klase komponente. Cilj je da komponenta *UserItemComponent* prikazuje ime konkretnog korisnika.

Za klasu će biti uvedena nova osobina pod nazivom *name*. Dodavanjem ove osobine biće moguće višestruko koristiti komponentu za različite korisnike (uz zadržavanje istog HTML koda, logike i stilova). Sa ciljem realizovanja navedenog, klasa *UserItemComponent* je proširena kodom koji dodaje navedenu promenljivu i ona sada izgleda ovako:

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

@Component({
  selector: 'app-user-item',
  templateUrl: './user-item.component.html',
  styleUrls: ['./user-item.component.style']
})
export class UserItemComponent implements OnInit {

  name: string; // dodata nova osobina

  constructor() {
    this.name = 'Vlada';
  }

  ngOnInit() {
  }
}
```

Moguće je primetiti da su u klasi načinjene dve izmene:

1. dodata je nova osobina : Ovo je način kako *TypeScript* uvodi nove osobine u klase. Na ovaj način je obezbeđeno da će svaki kreirani objekat klase *UserItemComponent* imati osobinu *name*. Podešavanjem ove osobine na tip *string* ukazano je kompjajleru da podatak pridružen ovoj promenljivoj mora biti tipa *string* i da će se u suprotnom javiti greška;
2. dodat je nov kod u konstruktor klase: *TypeScript* klase, takođe, poseduju konstruktore - funkcije koje se pozivaju kada se kreiraju novi objekti klase. U konstruktoru je moguće, na primer, veoma jednostavno dodeliti konkretnu vrednost promenljivoj:

```
constructor() {  
    this.name = 'Vlada'; //dodata vrednosti promenljivoj  
}
```

KREIRANJE ŠABLONA ZA PRIKAZIVANJE KORISNIKA

Osobinu je moguće prikazati u šablonu primenom para vitičastih zagrada.

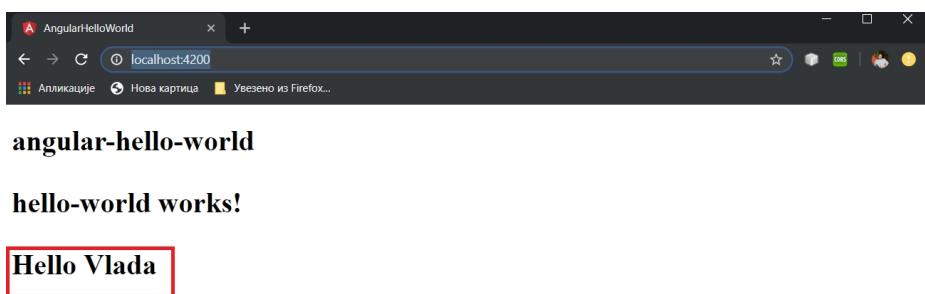
Komponenta poseduje osobinu. Osobinu je moguće prikazati u šablonu primenom para vitičastih zagrada `{} {}`. U konkretnom slučaju to može biti urađeno izmenom sadržaja datoteke *user-item-component.html* na sledeći način:

```
<p>  
    Hello {{name}}  
</p>
```

Još jednom neophodno je istaći da je u šablonu dodat kod koji predstavlja sintaksu koja do sada nije bila razmatrana - `{} name {}`. Par vitičastih zagrada predstavlja segment koji se naziva *šablonskim tagom* (template tag), po nekad se, u literaturi, šaljivo naziva i *brkovima* (mustaches).

Šta god da se nalazi između navedenih zagrada predstavlja *izraz*. Budući da je šablon povezan sa komponentom, vrednost izraza *name* odgovara vrednosti *this.name* iz konstruktora klase komponente.

Sada je moguće proveriti urađeni posao. Osvežava se stranica i, ako je sve urađeno kako treba, dobija se sledeći izlaz:



Slika 3.8 Prikazivanje korisnika

✓ Poglavlje 4

Rad sa nizovima

PETLJA NGFOR

Ideja je da se proširi primer tako da može da manipuliše kolekcijom objekata.

Pokazni primer funkcioniše tako što je moguće uputiti pozdravnu poruku samo jednom korisniku. U nastavku, ideja je da se proširi ova funkcionalnost tako da može da manipuliše kolekcijom objekata, odnosno da svaki od korisnika dobije pozdravnu poruku.

Angular omogućava iteraciju preko liste objekata primenom naredbe `*ngFor`. Takođe, ideja je da se HTML kod ostavi nepromjenjenim u najvećoj mogućoj meri i kada se radi sa kolekcijom objekata (biće dodata samo petlja umesto pojedinačnih objekata).

Ukoliko ste radili sa starijom verzijom AngularJS 1.x moguće je da ste koristili naredbu (direktivu) ng-repeat. Ova naredba radi identično kao ngFor.

KOMPONENTA ZA KOLEKCIJE

Neophodno je kreirati novu komponentu i šablon za prikazivanje liste korisnika

Da bi kolekcija korisnika mogla da bude prikazana, neophodno je kreirati novu komponentu i šablon za prikazivanje liste korisnika. U terminalu se navodi sledeća instrukcija:

```
ng generate component user-list
```

Takođe, u datoteci `app.component.html` neophodno je zameniti tag za prikazivanje pojedinačnih korisnika, tagom koji prikazuje informacije koje se odnose na kolekciju korisnika. To je prikazano sledećim listingom:

```
<h1>
  {{title}}
<app-hello-world></app-hello-world>

<app-user-list></app-user-list>
</h1>
```

Za razliku od klase `UserItemComponent`, klasa `UserListComponent` bi trebalo da ima mogućnost rada sa nizom objekata tipa `string`. Iz navedenog razloga, biće kreirana njena osobina `names` koja će predstavljati niz stringova, odnosno imena korisnika.

Dalje, u Angularu nizovi se predstavljaju uglastim zagradama - `[]`. Ovo može biti iskorišćeno da se u konstruktoru klase kreira konkretan niz koji odgovara osobini `names` klase komponente `UserListComponent`.

Sledećim listingom je demonstriran kod klase `UserListComponent` sa navedenim izmenama:

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

@Component({
  selector: 'app-user-list',
  templateUrl: './user-list.component.html',
  styleUrls: ['./user-list.component.css']
})
export class UserListComponent implements OnInit {

  names: string[];

  constructor() {
    this.names = ['Vlada', 'Tina', 'Laza', 'Sofija', 'Isidora'];
  }

  ngOnInit() {
  }

}
```

ŠABLON ZA KOLEKCIJE OBJEKATA

Neophodno je obaviti ažuriranje šablonu za prikazivanje liste imena

Da bi celokupan posao oko prikazivanja podataka, sa nizom korisnika, bio uspešno okončan, neophodno je obaviti ažuriranje šablonu za prikazivanje liste imena. U ovom kontekstu biće korišćena petlja `*ngFor` čiji će zadatak biti da:

- prođe preko liste objekata;
- generiše novi tag za svakog od njih.

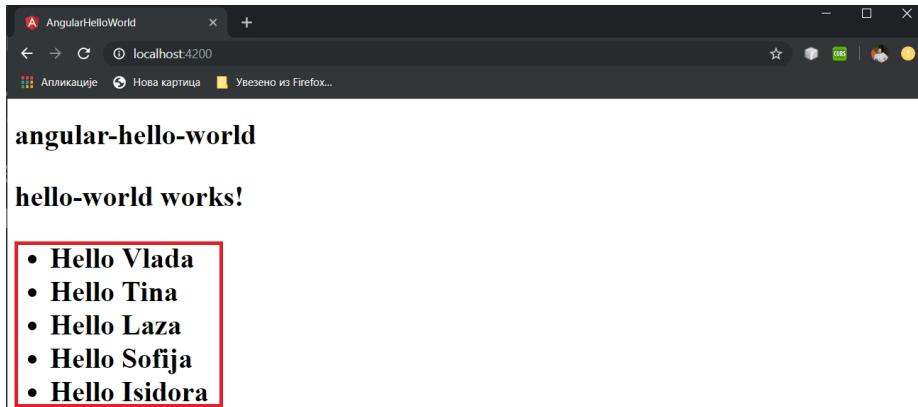
Ažurirani šablon `user-list.component.html` je prikazan sledećim listingom:

```
<ul>
  <li *ngFor="let name of names">Hello {{ name }}</li>
</ul>
```

I priloženog listinga može se primetiti da se u svakoj iteraciji dodaje nov član (tag ``) u neuređenu listu (tag ``). Kako funkcioniše petlja `*ngFor`? U principu, radi se o klasičnoj

`for-each` petlji. Deo koda ove petlje "`let name of names`" ukazuje da petlja kreira lokalnu promenljivu `name` u kojoj se čuva vrednost člana niza `names` iz tekuće iteracije petlje.

Sada je moguće učitati početnu stranicu aplikacije ponovo i, ako je sve urađeno na pravi način, trebalo bi da bude dobijen sledeći izlaz:



Slika 4.1 Pozdravne poruke za niz imena

▼ Poglavlje 5

Komponenta potomak

NOVA ULOGA KOMPONENTE USERITEMCOMPONENT

Namera je da se iskoristi komponenta UserItemComponent kao komponenta potomak

Za novo razmatranje biće ponovo aktivirana komponenta *UserItemComponent*. Umesto dodavanja pojedinačnih imena primenom komponente *UserListComponent*, namera je da se iskoristi komponenta *UserItemComponent* kao komponenta potomak. To znači da umesto direktnog kreiranja pozdravne poruke za članove niza korisnika biće omogućeno komponenti *UserItemComponent* da specificira šablon (i funkcionalnosti) za svaku stavku u listi. Da bi ovo bilo moguće neophodne su tri stvari:

- Podešavanje komponente *UserListComponent* da koristi komponentu *UserItemComponent* (u šablonu);
- Podešavanje komponente *UserItemComponent* da prihvata vrednosti za promenljivu *name* sa ulaza;
- Podešavanje šablona *UserListComponent* da prosleđuje vrednost *name* ka *UserItemComponent*.

U nastavku, neophodno je izvesti i objasniti navedene korake.

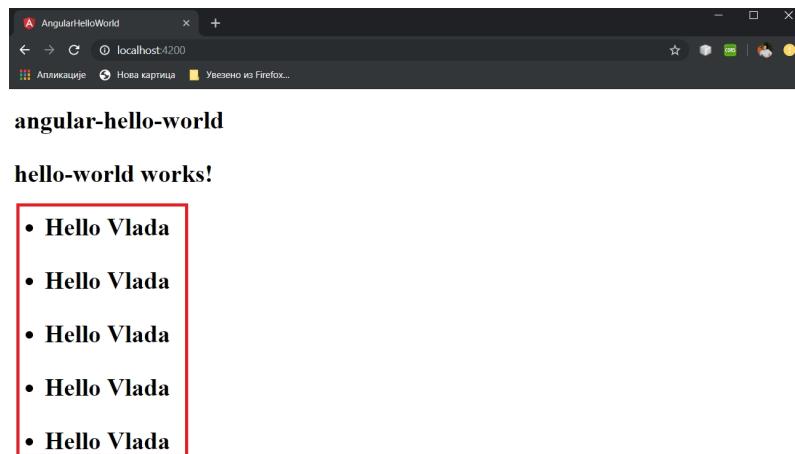
DODAVANJE POTOMAK KOMPONENTE USERITEMCOMPONENT

Dodaje se tag potomka u šablon roditeljske komponente.

Komponenta *UserItemComponent* specificira selektor *app-user-item* kojim je određen njen tag. Sledeći zadatak jeste dodavanje navedenog taga u šablon *user-list.component.html* roditeljske komponente. To može biti urađeno na sledeći način:

```
<ul>
  <li *ngFor = "let name of names">
    <app-user-item></app-user-item>
  </li>
</ul>
```

Moguće je primetiti da je linija *Hello {{name}}* zamenjena tagom *<app-user-item>*. Ako se u veb pregledaču osveži stranica biće dobijen sledeći izlaz:



Slika 5.1 Izlaz nakon dodavanja potomka u šablon roditelja

Petljа radi svoj posao, ali umesto niza različitih imena uvek se pojavljuje isto ime. Problem je što trenutno ne postoji mehanizam u aplikaciji za prosleđivanje podataka komponenti potomka. Srećom, [Angular](#) poseduje rešenje za navedeni problem primenom dekoratora (anotacije) `@Input`.

PODEŠAVANJE KOMPONENTE DA PRIHVATA ULAZ

Angular poseduje rešenje za ovaj problem primenom dekoratora (anotacije) Input

Kao što je istaknuto, Angular poseduje rešenje za ovaj problem primenom dekoratora (anotacije) `@Input`. Za početak neophodno je prisetiti se da u konstruktoru klase `UserItemComponent` postoji podešena osobina `name` na način: `this.name = 'Vlada'`.

Za ostvarivanje zacrtanih ciljeva, prvo bi trebalo redefinisati kod ove klase na sledeći način:

```
import { Component,
  OnInit,
  Input // ovo je dodato
} from '@angular/core';

@Component({
  selector: 'app-user-item',
  templateUrl: './user-item.component.html',
  styleUrls: ['./user-item.component.style']
})
export class UserItemComponent implements OnInit {

  @Input() name: string; // dodata anotacija

  constructor() {
    //obrisano podešavanje osobine
  }
}
```

```
ngOnInit() {  
}  
}
```

Iz priloženog listinga je moguće primetiti promenu na osobini `name` koja je obeležena dekoratorom `@Input`. O ulazima i izlazima u Angular aplikacijama biće više govora u narednoj lekciji. Za sada, dovoljno je da se zna da ova sintaksa omogućava prosleđivanje vrednosti iz roditeljskog šablona.

Da mi ovaj dekorator mogao da bude korišćen, neophodno je obaviti i njegovo dodavanje u `import` delu koda klase komponente.

Konačno, odustaje se od zadavanja podrazumevane vrednosti za ime i ovaj deo koda se briše iz konstruktora.

Sada se postavlja novo pitanje. Kako se koristi promenljiva koja je označena kao ulazna vrednost?

PROSLEĐIVANJE ULAZNE VREDNOSTI

Za prosleđivanje vrednosti komponenti koristi se sintaksa uglaste zagrade u okviru šablona

Za prosleđivanje vrednosti komponenti koristi se sintaksa uglaste zagrade [] u okviru šablona. Ažurirani šablon `user-list.component.html` ima sledeću formu:

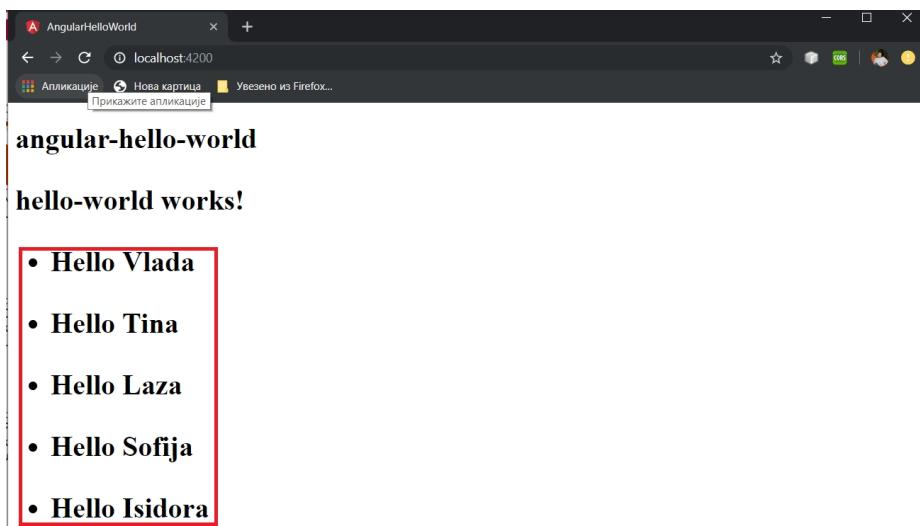
```
<ul>  
  <li *ngFor="let name of names">  
    <app-user-item [name]="name"></app-user-item>  
  </li>  
</ul>
```

Iz priloženog listinga moguće je sagledati da je problem prosleđivanja ulazne vrednosti rešen na veoma jednostavan način dodavanjem novog atributa unutra taga `<app-user-item tag>` - `[name] = "name"`.

Kroz jednostavan primer urađene su veoma bitne stvari koje studenti u ovom trenutnu moraju da razumeju:

- iteracija preko niza (imena);
- kreiranje nove komponente (`UserItemComponent`) za svaki član niza imena;
- prosleđivanje vrednosti (imena) u ulaznu osobinu (`name`) komponente potomka (`UserItemComponent`).

Nakon urađenih korekcija, program ponovo funkcioniše kako se od njega očekuje, a to je prikazano sledećom slikom.



Slika 5.2 Primer prosleđivanja vrednosti potomak komponenti

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 6

Komponente Angular aplikacije

ULAZNA TAČKA APLIKACIJE

Svaka aplikacija poseduje ulaznu tačku.

Svaka aplikacija poseduje ulaznu tačku. Aplikacija koja je kreirana i analizirana, u ovoj lekciji, prevedena je primenom alata [Angular CLI](#) koji se bazira na alatu poznatom pod nazivom [Webpack](#). Aplikacija se prevodi i pokreće pozivom komande:

```
ng serve
```

Naredba `ng` će pretražiti datoteku `angular.json` sa ciljem pronalaženja ulazne tačke aplikacije. Sledećim koracima je prikazano kako naredba `ng` pronađi kreirane komponente aplikacije:

- `angular.json` specificira glavni fajl aplikacije - u ovom slučaju to je `main.ts`;
- `main.ts` je ulazna tačka za aplikaciju i pokretanje aplikacije;
- pokretački proces pokreće `Angular modul` - o modulima će detaljno biti govora veoma brzo;
- `appModule.ts` specificira koja komponenta predstavlja komponentu najvišeg nivoa. U konkretnom slučaju to je `appComponent.ts`;
- navedena komponenta poseduje tag `<app-user-list>` i kreira listu svih korisnika.

SISTEM ANGULAR MODULA

Neophodno je staviti veći fokus na sistem Angular modula.

Kao što je navedeno u prethodnom izlaganju, neophodno je staviti veći fokus na [sistemu Angular modula](#) - [NgModule](#).

Angular neguje moćan sistem modula. Kada se podiže Angular aplikacija, ne pokreće se komponenta direktno. Umesto toga kreira se `NgModule` koji ukazuje na komponentu koju bi trebalo učitati.

Sledećim listingom je priložen delimičan kod datoteke `app.module.ts`.

```
@NgModule({
  declarations: [
    AppComponent,
    HelloWorldComponent,
    UserItemComponent,
    UserListComponent
```

```
],
imports: [
  BrowserModule,
  AppRoutingModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }
```

Prvo što je moguće primetiti iz listinga jeste dekorator `@NgModule`. Kao i svi dekoratori on ima zadatak da doda meta podatke u klasu. U konkretnom slučaju podeljuje četiri ključa:

- `declarations`;
- `imports`;
- `providers`;
- `bootstrap`.

DEKLARACIJE

Ključem declarations su specificirane komponente.

Ključem `declarations` su označene komponente koje definiše ovaj modul. Ovo predstavlja veoma važnu ideju Angular okvira. Neophodno je deklarisati komponente pre nogo što budu upotrebljene u šablonima.

`NgModule` može biti razmatran i kao paket, a deklaracije ukazuju na komponente koje modul poseduje. Takođe, moguće je primetiti da kada se kreira komponenta, primenom naredbe:

```
ng generate
```

ovaj `ng` alat će automatski dodati komponentu u listu deklarisanih komponenata. Ideja je upravo da kada se kreira nova komponenta, `ng` alat podrazumeva da ona mora da pripada aktuelnom `NgModule`.

IMPORTS KLJUČ

Ključ imports opisuje koje zavisnosti modul poseduje.

Ključ `imports` opisuje koje zavisnosti modul poseduje. Budući da je kreirana aplikacija koja za izvršavanje zahteva veb pregledač, neophodno je dodati modul `BrowserModule`. Ukoliko modul zavisi od još nekih modula i njih je potrebno dodati na ovu listu.

VAŽNO!!!

Neophodno je napraviti razliku između `import` i `imports`. `Import` se koristi u klasama, a `imports` u modulima. Najjednostavnije objašnjenje bi moglo biti da se pod ključem `imports` dekoratora `@NgModule` čuvaju elementi koji se koriste u `HTML` šablonima ili prilikom `umetanja zavisnosti`

(*dependency injection*). O konceptu umetanja zavisnosti biće detaljno govora u ovom predmetu.

KLJUČEVI PROVIDERS I BOOTSTRAP

Provajderi se koriste prilikom umetanja zavisnosti. Ključ bootstrap omogućava učitavanje AppComponent komponente.

Provajderi se koriste prilikom umetanja zavisnosti. Klasa obeležena dekoratorom `@NgModule` omogućava umetanje konkretnih servisa tokom izvršavanja aplikacije tako što ih deklariše pod ključem *decorators*.

Ključ `bootstrap` kazuje Angularu da kada se ovaj modul koristi za pokretanje aplikacije, neophodno je obaviti učitavanje `AppComponent` komponente kao komponente najvišeg nivoa.

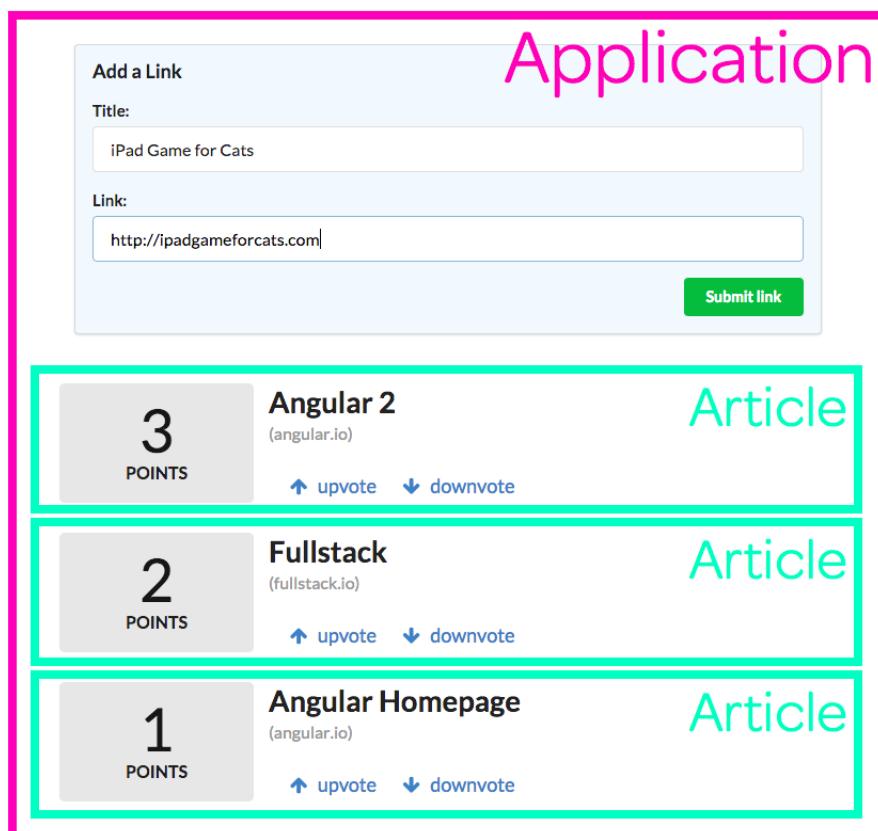
✓ Poglavlje 7

Pokazni primer - proširenje aplikacije

BUDUĆI IZGLED APLIKACIJE

Aplikacija bi trebalo da bude pojednostavljena kopija aplikacije Reddit.

Kao što je istaknuto u prethodnom izlaganju, konačni cilj je kreiranje aplikacije koja bi trebalo da bude pojednostavljena kopija aplikacije [Reddit](#). Sledećom slikom je priložena ideja samog izgleda aplikacije koja će u nastavku biti kreirana po principu korak - po - korak proširenjem inicijalnog koda.



Slika 7.1.1 Predloženi izgled aplikacije

KOMPONENTE APLIKACIJE

Neophodno je dodati dve nove komponente u aplikaciju.

Aplikaciju će činiti dve komponente:

1. celokupna aplikacija koja će sadržati formu za dodavanje novih proizvoda (na slici je istaknuto ljubičastom bojom);
2. pojedinačni proizvodi (istaknuti su zelenom mint bojom).

U velikim, realnim aplikacijama, forma za dodavanje novih proizvoda bi verovatno predstavljala novu komponentu. Međutim, postojanje nove komponente bi u ovom trenutku učinilo prosleđivanje podataka kompleksnijim. Za sada, da bi bolje i lakše shvatili ovu problematiku, akcenat će biti na jednostavnosti.

Sada je moguće početi sa izradom konkretne aplikacije. U terminalu razvojnog okruženja, primenom [Angular CLI](#), kucamo naredbu za kreiranje novog projekta pod nazivom [angular-reddit](#):

```
ng new angular-reddit
```

Ubrzo započinje kreiranje novog projekta što rezultuje sledećim izlazom u terminalu.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Vlada\Documents\Angular> ng new angular-reddit
? would you like to add Angular routing? Yes
? which stylesheet format would you like to use? CSS
CREATE angular-reddit/angular.json (3657 bytes)
CREATE angular-reddit/package.json (1288 bytes)
CREATE angular-reddit/README.md (1038 bytes)
CREATE angular-reddit/tsconfig.json (543 bytes)
CREATE angular-reddit/tslint.json (1988 bytes)
CREATE angular-reddit/.editorconfig (246 bytes)
CREATE angular-reddit/.gitignore (631 bytes)
CREATE angular-reddit/browserslist (429 bytes)
CREATE angular-reddit/karma.conf.js (1026 bytes)
CREATE angular-reddit/tsconfig.app.json (270 bytes)
CREATE angular-reddit/tsconfig.spec.json (270 bytes)
CREATE angular-reddit/src/favicon.ico (5430 bytes)
CREATE angular-reddit/src/index.html (300 bytes)
CREATE angular-reddit/src/main.ts (372 bytes)
CREATE angular-reddit/src/polyfills.ts (2838 bytes)
CREATE angular-reddit/src/styles.css (80 bytes)
CREATE angular-reddit/src/test.ts (642 bytes)
CREATE angular-reddit/src/assets/.gitkeep (0 bytes)
CREATE angular-reddit/src/environments/environment.prod.ts (51 bytes)
CREATE angular-reddit/src/environments/environment.ts (662 bytes)
CREATE angular-reddit/src/app/app-routing.module.ts (246 bytes)
CREATE angular-reddit/src/app/app.module.ts (393 bytes)
CREATE angular-reddit/src/app/app.component.html (1152 bytes)
CREATE angular-reddit/src/app/app.component.spec.ts (1119 bytes)
CREATE angular-reddit/src/app/app.component.ts (218 bytes)
CREATE angular-reddit/src/app/app.component.css (0 bytes)
CREATE angular-reddit/e2e/protractor.conf.js (810 bytes)
CREATE angular-reddit/e2e/tsconfig.json (214 bytes)
CREATE angular-reddit/e2e/src/app.e2e-spec.ts (643 bytes)
CREATE angular-reddit/e2e/src/app.po.ts (251 bytes)
```

Slika 7.1.2 Kreiranje novog Angular projekta

CSS

Pre bavljenja glavnom komponentom aplikacije cilj je da se izgled aplikacije malo ulepša.

Pre bavljenja glavnom komponentom aplikacije cilj je da se izgled aplikacije malo ulepša. Ukoliko pravite aplikaciju iz početka, možete se poslužiti urađenim primerom kojeg možete preuzeti odmah nakon ovog objekta učenja. U vaš projekat prekopirajte sledeće datoteke i foldere u folder sa vašom aplikacijom:

- *src/index.html*
- *src/styles.css*

- *src/app/vendor*
- *src/assets/images*.

Za ulepšavanje projekta korišćen je *Semantic-UI* (više na linku <https://semantic-ui.com/>).

GLAVNA KOMPONENTA APLIKACIJE

*Glavna komponenta aplikacije je određena datotekom *src/app/app.component.ts**

Sada je moguće pristupiti kreiranju nove komponente čiji će zadatak biti da:

1. čuva listu postojećih proizvoda;
2. sadrži formu za dodavanje novih proizvoda.

Glavna komponenta aplikacije je određena datotekom *src/app/app.component.ts*. Otvara se datoteka i u njoj se trenutno nalazi dobro poznat inicijalni kod:

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

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

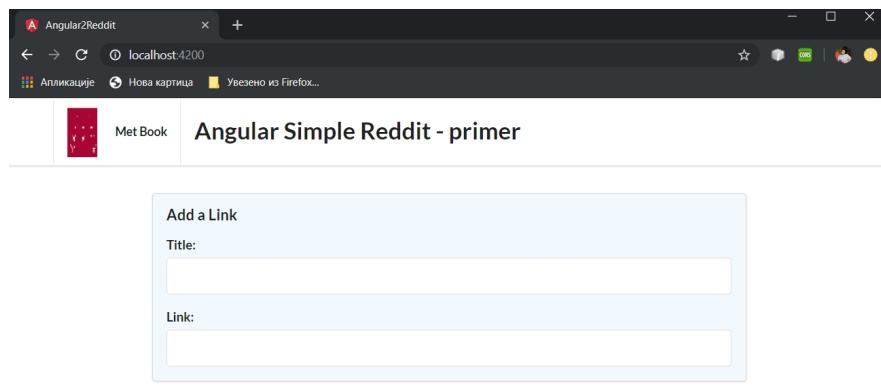
Moguće je primetiti da osobina se *title* automatski dodaje u klasu *AppComponent*. **Ovu liniju je moguće slobodno obrisati iz razloga što nigde u aplikaciji neće biti korišćen naslov komponente.**

Sada je moguće načiniti izmene i u šablonu glavne komponente kreiranjem forme za dodavanje novih linkova. Biće korišćeni stilovi iz paketa *semantic-ui* za lepši izgled stranice. Sledećim listingom data je redefinisana datoteka *app.component.html*.

```
<form class="ui large form segment">
  <h3 class="ui header">Add a Link</h3>

  <div class="field">
    <label for="title">Title:</label>
    <input name="title">
  </div>
  <div class="field">
    <label for="link">Link:</label>
    <input name="link">
  </div>
</form>
```

Kreiran je šablon koji sadrži dva taga za unos (*input*): jedan za unos naziva proizvoda, a drugi za odgovarajući link. Sada je moguće pokrenuti aplikaciju sa ciljem sagledavanja do sada postignutih rezultata.



Slika 7.1.3 Kreirana forma na stranici Angular aplikacije

DODAVANJE INTERAKCIJE

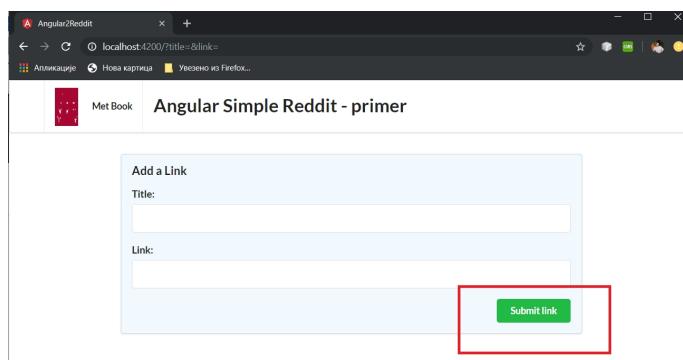
Neophodno je kreirati mehanizam interakcije u formi dugmeta za potvrđivanje unosa sa forme.

U dosadašnjem radu je kreirana **forma** koja poseduje tagove za unos podataka. Međutim, ne postoji mehanizam za potvrđivanje i slanje unetih podataka na dalju obradu. U tom svetlu, neophodno je kreirati mehanizam interakcije u formi dugmeta za potvrđivanje unosa sa forme.

Kada je unos na formi potvrđen, neophodno je pozvati funkciju koja kreira i dodaje link. Ovo je moguće obaviti dodavanjem interaktivnog događaja za GUI element (dugme). Na primer, dodavanje funkcije, za događaj dugmeta tipa *onClick*, moguće je obaviti na sledeći način u kodu šablona razmatranog u prethodnom izlaganju:

```
<button (click)="addArticle()" class="ui positive right floated button">  
    Submit link  
</button>
```

Stranica će dobiti izgled kao na sledećoj slici.



Slika 7.1.4 Dodato dugme za interakciju na formi

Klikom na dugme se još uvek ne dešava ništa. Neophodno je definisati funkciju `addArticle()`, u okviru klase `AppComponent` koja se poziva i izvršava klikom na kreirano dugme. To je moguće obaviti na sledeći način:

```
export class AppComponent {  
  addArticle(title: HTMLInputElement, link: HTMLInputElement): boolean {  
    console.log(`Adding article title: ${title.value} and link: ${link.value}`);  
    return false;  
}
```

Iz koda je moguće primetiti da funkcija `addArticle()` uzima dva argumenta: `title` i `link`. Neophodno je napraviti izmene u šablonu da bi dugme moglo da prosledi ove dve vrednosti u poziv metode `addArticle()`.

```
<form class="ui large form segment">  
  <h3 class="ui header">Add a Link</h3>  
  
  <div class="field">  
    <label for="title">Title:</label>  
    <input name="title" id="title" #newtitle> <!-- promenjeno -->  
  </div>  
  <div class="field">  
    <label for="link">Link:</label>  
    <input name="link" id="link" #newlink> <!-- promenjeno -->  
  </div>  
  
  <!-- dugme prosleđuje metodi parametre newtitle i newlink-->  
  <button (click)="addArticle(newtitle, newlink)" class="ui positive right floated button">  
    Submit link  
  </button>  
  
</form>
```

DODAVANJE INTERAKCIJE - ANALIZA

Primenom specijalnog znaka "hash" ukazuje se Angularu da pridruži tagove lokalnim promenljivim.

Iz prethodno priloženog listinga moguće je primetiti primenu specijalnog znaka `#` preko kojeg se ukazuje Angularu da pridruži tagove lokalnim promenljivim. Dodavanjem, parametara `#newtitle` i `#newlink` odgovarajućim `<input/>` elementima, moguće je njihovo prosleđivanje kao promenljivih u metodu `addArticle()` nakon klika na kreirano dugme.

Napravljene su sledeće korekcije:

1. Kreirano je dugme u HTML kodu koje navodi korisnika gde treba da klikne;

2. Kreirana je funkcija `addArticle()` kojom je definisano šta bi trebalo obaviti kada se klikne na dugme;
3. Za dugme je dodat atribut `(click)` koji ukazuje da se pozove funkcija `addArticle()` kada se klikne na dugme;
4. Dodati su atributi `#newtitle` i `#newlink` u odgovarajuće `<input>` tagove.

Neophodno je dati kratko objašnjenje za svaki od navedenih koraka.

POVEZIVANJE ULAZA SA VREDNOSTIMA

Povezivanje ulaza sa vrednostima ima za efekat dostupnost promenljive izrazima unutar pogleda.

Neophodno je obratiti pažnju na prvi `<input>` tag:

```
<input name="title" #newtitle>
```

navedeni kod ukazuje okviru Angular da poveže vrednost unetu u polju `<input>` sa promenljivom `newtitle`. Ova sintaksa se još naziva "rešavanjem" (`resolve`). Ona ima za efekat dostupnost promenljive izrazima unutar pogleda (šablonu, veb stranice). Varijablom `newtitle` je sada određen objekat koji reprezentuje ulazni DOM (Document Object Model) element (posebno, tip je `HTMLInputElement`). Pošto je `newtitle` objekat, vrednost `<input>` taga se dobija pozivom `newtitle.value`.

Potpuno identična priča važi za promenljivu `newlink` i neće biti ponavljana.

POVEZIVANJE AKCIJA SA DOGAĐAJIMA

Atributom `(click)` je definisana akcija koja se izvršava kada se klikne na dugme.

Ako je još jednom pogleda poslednji priloženi HTML listing moguće je primetiti da je tagu `<button>` dodat atribut `(click)` kojim je definisana akcija koja se izvršava kada se klikne na dugme. Kada se desi navedeni događaj, poziva se funkcija `addArticle()` sa dva argumenta: `newtitle`

i `newlink`. Odakle se javljaju funkcija i navedeni argumenti?

1. `addArticle()` je funkcija definisana u klasi komponente `AppComponent`;
2. `newtitle` se dobija "rešavanjem" izraza `(#newtitle)` koji pripada `<input>` tagu pod nazivom `title`;
3. `newlink` se dobija "rešavanjem" izraza `(#newlink)` koji pripada `<input>` tagu pod nazivom `link`;

Sve zajedno moguće je priložiti kroz sledeći listing.

```
<!-- dugme prosleđuje metodi parametre newtitle i newlink-->  
<button (click)="addArticle(newtitle, newlink)">
```

```
class="ui positive right floated button">
Submit link
</button>
```

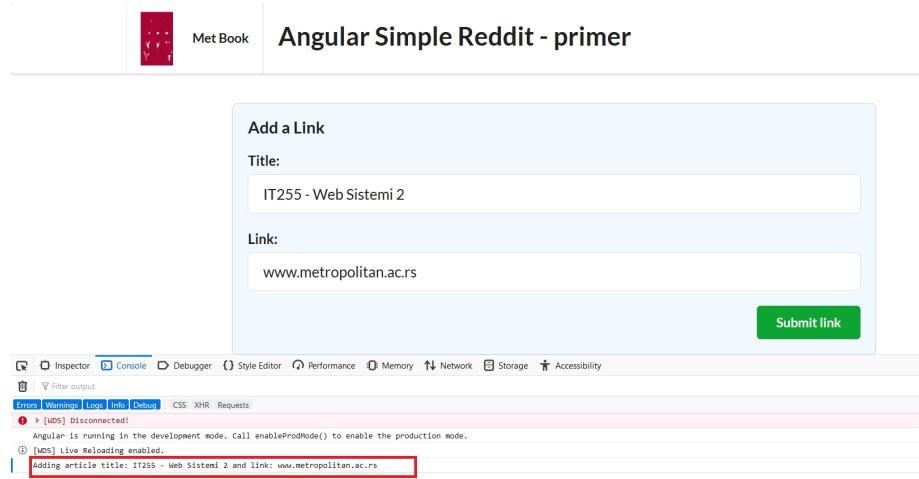
DEFINISANJE LOGIKE AKCIJE

Argumenti metode koja rukuje događajem su objekti tipa `HTMLInputElement` i ne predstavljaju direktno unete vrednosti.

Za klasu `AppComponent` definisana je funkcija pod nazivom `addArticle()`. Funkcija, kao što je već istaknuto, uzima dva argumenta: `title` i `link`. Opet, važno je razumeti da su oba argumenta objekti tipa `HTMLInputElement` i ne predstavljaju direktno unete vrednosti. Da bi se dobila uneta vrednost sa ulaza, na primer za objekat `title`, neophodno je obaviti poziv izraza `title.value`. Za sada će biti dovoljno prikazati izlaz, nastao pozivom ove metode, u konzoli. Ovo je omogućeno dodavanjem sledećeg koda u metodu `addArticle()` klase komponente `AppComponent`.

```
export class AppComponent {
  addArticle(title: HTMLInputElement, link: HTMLInputElement): boolean {
    console.log(`Adding article title: ${title.value} and link: ${link.value}`);
    return false;
  }
}
```

Sada je moguće obaviti testiranje urađenog posla. Učitava se ponovo stranica, popunjava se forma i u konzoli se prate rezultati nakon klika na dugme za potvrdu forme. Ako je sve u redu, rezultat može biti kao na sledećoj slici.



Slika 7.1.5 Prikaz rezultata obrade forme u konzoli veb pregledača

DODAVANJE KOMPONENTE ARTICLE

Javlja se potreba za kreiranjem nove komponente koja će rukovati dodavanjem članaka na stranicu.

Očigledno je da posao oko izrade željene aplikacije nije priveden kraju. Postoji forma za dodavanje proizvoda ali se onii ne prikazuju nigde. Iz razloga što se svaki proizvod, koji je unet putem forme, prikazuje u listi na odgovarajućoj stranici, javlja se potreba za kreiranjem nove komponente koja će rukovati ovim zadatkom.

Pa neka u sledećem koraku bude kreirana tražena komponenta koja će predstavljati pojedinačne postavljene proizvode (kao na slici 6).

ng generate component article



Slika 7.1.6 Izgled dodatog članka na stranici

Za kreiranu komponentu je potrebno obaviti posebnu brigu oko krajnjih definicija za sledeće delove:

1. Definisanje *ArticleComponent* pogleda u šablonu;
2. Definisanje *ArticleComponent* osobina obeležavanjem klase dekoratorom *@Component*
3. Definisanje *ArticleComponent* klase komponente koja čuva programsku logiku navedene komponente.

U nastavku sledi diskusija u vezi sa istaknutim zadacima.

KREIRANJE ŠABLONA ARTICLECOMPONENT

Šablon će biti kreiran preko datoteke article.component.html.

Šablon će biti kreiran preko datoteke article.component.html.

```
<div class="four wide column center aligned votes">
  <div class="ui statistic">
    <div class="value">
      {{ votes }}
    </div>
    <div class="label">
      Points
    </div>
  </div>
</div>
<div class="twelve wide column">
  <a class="ui large header" href="{{ link }}">
    {{ title }}
  </a>
  <ul class="ui big horizontal list voters">
```

```
<li class="item">
    <a href (click)="voteUp()">
        <i class="arrow up icon"></i>
        upvote
    </a>
</li>
<li class="item">
    <a href (click)="voteDown()">
        <i class="arrow down icon"></i>
        downvote
    </a>
</li>
</ul>
</div>
```

Šablonom su određene dve kolone:ž

1. broj glasova za proizvod, na levoj strani;
2. informacije o proizvodu, na desnoj strani.

Kolone su ulepšane CSS klasama: *four wide column* i *twelve wide column*, respektivno (rečeno je na početku da će biti korišćen *SemanticUI's CSS*).

Glasovi i naslovi se prikazuju u šablonu primenom izraza `{{ votes }}` i `{{ title }}`. Vrednosti se dobijaju preko osobina `vote` i `title` klase komponente `ArticleComponent` koja će ubrzo biti definisana.

Takođe, moguće je koristiti šablonske stringove u vrednostima atributa, kao u slučaju: `href="{{link}}"`. U ovom slučaju, vrednost za `href` biće dodata dinamički preko vrednosti linka iz klase komponente.

Za linkove `upvote` i `downvote` neophodno je definisati akcije. Biće upotrebljen `(click)` za povezivanje metoda `voteUp()` i `voteDown()` sa odgovarajućim kontrolama na stranici. Kada se klikne na link `upvote`, biće pozvana metoda `voteUp()` klase komponente `ArticleComponent`. Identičan slučaj je sa linkom `downvote` i odgovarajućom metodom `voteDown()`.

DEKATOR KLASE ARTICLECOMPONENT

Obeležavanje dekoratorom omogućava primenu novog taga za prikazivanje članaka.

U nastavku, neophodno je obaviti obeležavanje klase komponente odgovarajućim dekoratorom `@Component`. Posebno je bitna vrednost ključa `selector` kojom je određen naziv taga koji će omogućiti primenu komponente u HTML stranici.

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

Iz priloženog listinga je moguće primetiti da je definisan tag `<app-article> </app-article>` koji odgovara aktuelnoj komponenti.

ZAVRŠNA DEFINICIJA KLASE ARTICLECOMPONENT

Kreira se programska logika klase komponente ArticleComponent.

Konačno, na red dolazi i završna definicija klase komponente `ArticleComponent`. Neophodno je otvoriti datoteku `article.component.ts` i dodati kod priložen sledećim listingom:

```
export class ArticleComponent implements OnInit {  
  
    @HostBinding('attr.class') cssClass = 'row';  
    votes: number;  
    title: string;  
    link: string;  
  
    constructor() {  
        this.title = 'IT255 - Veb sistemi 1';  
        this.link = 'http://www.metropolitan.ac.rs';  
        this.votes = 10;  
    }  
  
    voteUp() {  
        this.votes += 1;  
    }  
  
    voteDown() {  
        this.votes -= 1;  
    }  
  
    ngOnInit() {  
    }  
}
```

Kreirane su četiri osobine klase `ArticleComponent`:

1. `cssClass` - predstavlja CSS klasu koja će biti primenjena na "hosta" ove komponente;
2. `votes` - broj koji prestavlja zbir glasova vraćen nakon svih poziva metoda `voteUp()` i `voteDown()`, respektivno;
3. `title` - string koji označava naziv proizvoda;
4. `link` - string koji označava link proizvoda.

Ovde je cilj da svaki novi proizvod bude dodat u poseban red na stranici. Za stilizaciju će biti upotrebljen, kao i do sada, `Semantic-UI` koji obezbeđuje `CSS` klasu za redove pod nazivom `row`.

U Angularu, komponenta `host` je određena elementom sa kojim je komponenta povezana. Moguće je podesiti osobinu host elementa primenom dekoratora `@HostBinding()`. U ovom slučaju, traži se od `Angulara` da vrednost klase host elemenata bude u sinhronizaciji sa svojstvom `cssClass`.

Da bi dekorator `@HostBinding()`, uopšte mogao da bude korišćen na ovakav način, neophodno je proširiti `import` sekciju klase na sledeći način:

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

KLASA ARTICLECOMPONENT - DODATNA RAZMATRANJA

Nastavlja se analiza kreirane klase komponente.

Nastavlja se analiza kreirane klase komponente. Sada je pažnja usmerena ka konstruktoru gde su podešene podrazumevane vrednosti za atribute na sledeći način:

```
constructor() {
  this.title = 'IT255 - Veb sistemi 1';
  this.link = 'http://www.metropolitan.ac.rs';
  this.votes = 10;
}
```

Konačno, definicija klase je završena metodama koje omogućavaju korisnicima glasanje za dati proizvod: `voteUp()` i `voteDown()`.

```
voteUp() {
  this.votes += 1;
}

voteDown() {
  this.votes -= 1;
}
```

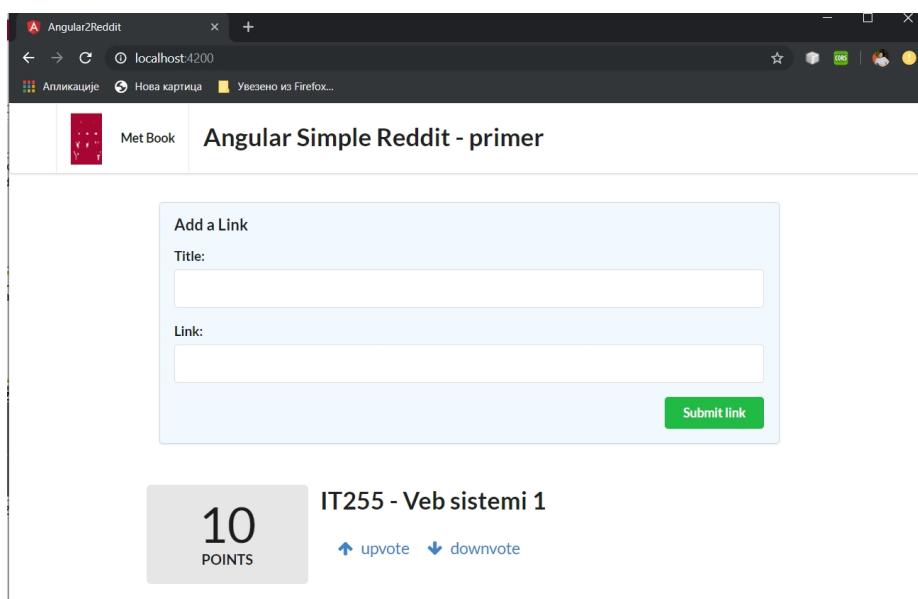
UPOTREBA KOMPONENTE APP-ARTICLE

Neophodno je dodati tag komponente na odgovarajuće mesto u HTML kodu.

Sa ciljem korišćenja kreirane komponente i činjenjem vidljivih podataka kojima ona manipuliše, neophodno je dodati tag komponente na odgovarajuće mesto u HTML kodu. To mesto je u šablonu komponente `AppComponent` (u datoteci `app.component.html`) odmah iza završnog taga `</form>`. Sledećim, delimičnim, listingom ove datoteke, prikazana je opisana aktivnost.

```
***  
<button (click)="addArticle(newtitle, newlink)" class="ui positive right floated button">  
    Submit link  
</button>  
</form>  
  
<div class="ui grid posts">  
    <app-article>  
    </app-article>  
</div>
```

Sada je moguće testirati urađenu aplikaciju ponovnim učitavanjem stranice. Ukoliko se dobije izlaz kao na slici 7, sve je obavljeno na pravi način.



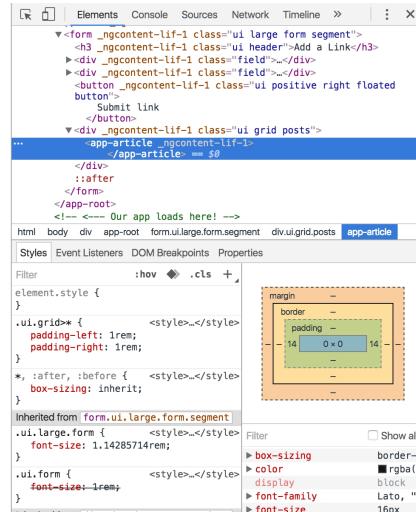
Slika 7.1.7 Izgled urađene aplikacije

DISKUSIJA O PREDNOSTI UPOTREBE ANGULAR CLI

Angular CLI automatski registruje kreiranu komponentu.

Ukoliko je komponenta *ArticleComponent* kreirana primenom Angular CLI, primenom instrukcije *ng generate*, po osnovnim podešavanjima Angular će biti upoznat sa postojanjem ove komponente jer će ona biti automatski dodata u datoteku *app.module.ts*. Aplikacija će biti uspešno prevedena i izvršena na način prikazan prethodnom slikom.

Međutim, ukoliko je izbegnut ovaj alat prilikom kreiranja komponente, prilikom osvežavanja veb pregledača moguće je primetiti da navedeni tag nije preveden, a to je moguće dodatno proveriti iz konzole veb pregledača (sledeća slika).



Slika 7.1.8 Greška prilikom interpretacije taga komponente

Problem je u tome da glavna komponenta *AppComponent* još uvek nije upoznata sa komponentom *ArticleComponent*.

Da bi navedeni problem bio rešen, neophodno je manuelno obaviti registrovanje komponente u datoteci *app.module.ts* na način prikazan sledećom slikom.

```
angular-reddit > src > app > ts app.module.ts > ...
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { ArticleComponent } from './article/article.component';
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11     ArticleComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
```

Slika 7.1.9 Manuelno registrovanje komponente u datoteci *app.module.ts*

PROBLEM OSVEŽAVANJA STRANICA

Problem: Klikom na linkove za glasanje vrši se ponovno učitavanje stranice, a ne ažuriranje liste.

Prilikom testiranja izvršavanja kreirane aplikacije, moguće je uočiti dodatni problem. Klikom na linkove za glasanje vrši se ponovno učitavanje stranice umesto prostog ažuriranja liste proizvoda. **JavaScript**, po osnovnim podešavanjima, širi klik događaj prema svim roditeljskim komponentama. Upravo zbog ovoga, veb pregledač pokušava da prati prazan link koji mu ukazuje na ponovno učitavanje sadržaja.

Da bi ovaj problem bio rešen neophodno je dodati malo koda u metodu koja rukuje događajem klika na link. Metoda mora da vrati vrednost **false** koja će osigurati da veb pregledač neće pokušati, nakon poziva metode, da osveži stranicu.

Jasno je da su metode koje rukuju događajima, u našem primeru, **voteUp()** i **voteDown()**. Sledi listing sa njihovim korigovanim kodom:

```
voteUp() {  
    this.votes += 1;  
    return false;  
}  
  
voteDown() {  
    this.votes -= 1;  
    return false;  
}
```

Sada je sve u redu sa linkovima i glasanje za proizvod ide glatko i nije praćeno ponovnim učitavanjima stranice.

✓ 7.1 Obrada više redova

PROBLEM DODAVANJA NOVIH PROIZVODA

Postoji samo jedan proizvod na stranici i ne postoji mehanizam za dodavanje novih

Prilikom testiranja izvršavanja aplikacije moguće je uočiti još jedan problem. Trenutno, **postoji samo jedan proizvod na stranici i ne postoji mehanizam za dodavanje novih**, ukoliko ne bude dodat još jedan tag. Međutim, ako ovo bude urađeno, svi proizvodi će imati identičan sadržaj, a to i nije ono što se očekuje od ove aplikacije.

KREIRANJE MODELSKE KLASE ARTICLE

Dobru praksu predstavlja pisanje koda kojim se razdvaja struktura podataka od koda komponente.

Dobru praksu predstavlja pisanje Angular kodakojim se razdvaja struktura podataka koja se koristi od koda komponente. Za realizovanje ovog zadatka, neophodno je kreirati strukturu

podataka koja predstavlja pojedinačne proizvode. Ovakva struktura podataka je poznata pod nazivom modelska klasa ili model u MVC (*Model - View - Controller*) šablonima.

U folderu `/article` aktuelnog projekta biće kreirana datoteka `article.model.ts` kojom će biti definisana modelska klasa `Article`. Sledеćim listingom je data njena definicija:

```
export class Article {  
    title: string;  
    link: string;  
    votes: number;  
  
    constructor(title: string, link: string, votes?: number) {  
        this.title = title;  
        this.link = link;  
        this.votes = votes || 0;  
    }  
}
```

Kao što je moguće primetiti iz koda, radi se o običnoj (*plain*) klasi, a ne o Angular komponenti. Svaki proizvod ima: naslov, link i broj glasova.

Kada se kreira nov objekat za proizvod, osobine `title` i `link` su obavezne. Parametar `vote` je opcionalan (obezbeđeno znakom ? na kraju naziva parametra) sa podrazumevanom vrednošću 0.

PRIDRUŽIVANJE MODELA KOMPONENTI

Neophodno je ažuriranje klase komponente da bi mogla da koristi model.

Neophodno je ažuriranje klase komponente `ArticleComponent` da bi mogla da koristi modelsku klasu `Article`. Umesto direktno čuvanja osobina u klasi komponente `ArticleComponent`, osobine će biti čuvane u instanci klase `Article`.

Prvi korak jeste kreiranje `import` instrukcije u klasi komponente `ArticleComponent` za klasu `Article`.

```
import { Article } from './article.model';
```

Nakon ovoga, moguće je upotrebiti objekat `Article` u `ArticleComponent` klasi komponente na sledeći način:

```
export class ArticleComponent implements OnInit {  
  
    @HostBinding('attr.class') cssClass = 'row';  
    article: Article;  
  
    constructor() {  
        this.article = new Article(  
            'IT255 - Veb sistemi 1',
```

```
'http://www.mwtropolitan.ac.rs',
10);
}

voteUp() {
    this.article.votes += 1;
    return false;
}

voteDown() {
    this.article.votes -= 1;
    return false;
}

ngOnInit() {
}

}
```

Iz koda je moguće primetiti šta je promenjeno: umesto čuvanja naziva, linka i broja glasova u odgovarajućim osobinama komponente, čuva se referenca na proizvod. Kada se vrši glasanje, glasovi se na ažuriraju za komponentu već za *article* objekat.

KOREKCIJE ŠABLONA

Neophodno obaviti izmene na HTML šablonima da bi mogli da prihvataju vrednosti promenljivih sa pravih lokacija.

Navedene promene u kodu uvode i dodatne promene koje je neophodno obaviti na *HTML* šablonima da bi mogli da prihvataju vrednosti promenljivih sa pravih lokacija. Da bi ovo bilo moguće, neophodno je obaviti promenu tagova šablonu da čitaju iz *article* objekata. To znači, da bi izraz `{{ votes }}` trebalo da bude zamenjen izrazom `{{ article.votes }}`. Potpuno isto važi i za osobine *title* i *link*.

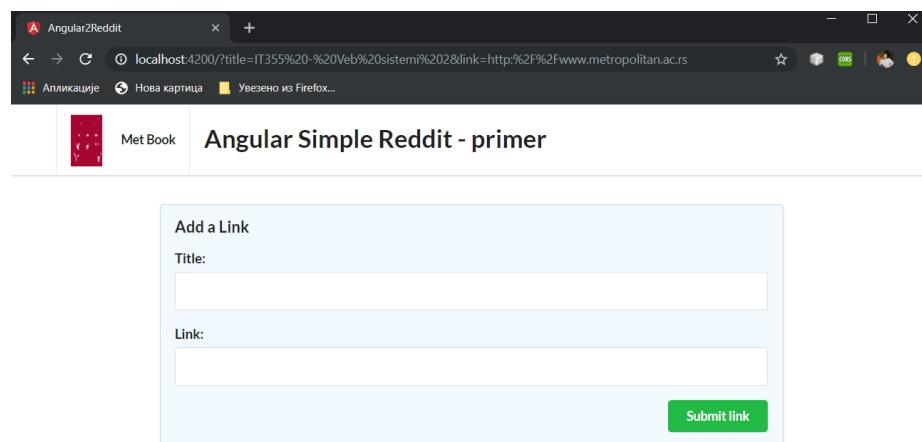
```
<div class="four wide column center aligned votes">
    <div class="ui statistic">
        <div class="value">
            {{ article.votes }}
        </div>
        <div class="label">
            Points
        </div>
    </div>
</div>
<div class="twelve wide column">
    <a class="ui large header" href="{{ article.link }}">
        {{ article.title }}
    </a>
    <ul class="ui big horizontal list voters">
        <li class="item">
```

```

<a href (click)="voteUp()">
    <i class="arrow up icon"></i>
    upvote
</a>
</li>
<li class="item">
    <a href (click)="voteDown()">
        <i class="arrow down icon"></i>
        downvote
    </a>
</li>
</ul>
</div>

```

Sada je aplikacija ponovo vraćena na nivo funkcionalnosti na kojem je bila pre korekcija. Sve radi osim dodavanja novih proizvoda.



Slika 7.2.1 Aplikacija sa delimičnim funkcionalnostima

DODATNI PROBLEM SA ENKAPSULACIJOM

Metode `voteUp()` i `voteDown()` narušavaju enkapsulaciju klase `Article`.

Trenutna situacija sa aplikacijom je nešto bolja od polazne ali i dalje ima nedostataka. Metode `voteUp()` i `voteDown()` narušavaju enkapsulaciju klase `Article` budući da menjaju direktno unutrašnje osobine njenih objekata. Ovde je narušen, konkretno, *Demeterov zakon* (https://en.wikipedia.org/wiki/Law_of_Demeter) koji govori da posmatrani objekat trebalo da ima što manje saznanja o strukturi drugih objekata.

Ovde je problem što klasa komponente `ArticleComponent` ima puno saznanja o unutrašnjoj implementaciji klase `Article`. Da bi ovaj problem bio prevaziđen, metode `voteUp()` i `voteDown()` će biti dodata u klasu `Article`. Takođe, biće dodata i funkcija `domain()` o kojoj će uskoro biti govora.

```
export class Article {
    title: string;
    link: string;
    votes: number;

    constructor(title: string, link: string, votes?: number) {
        this.title = title;
        this.link = link;
        this.votes = votes || 0;
    }

    voteUp(): void {
        this.votes += 1;
    }

    voteDown(): void {
        this.votes -= 1;
    }

    // domain() je funkcija koja izdvaja
    // domen iz URL, biće uskoro objašnjena
    domain(): string {
        try {
            // e.g. http://foo.com/path/to/bar
            const domainAndPath: string = this.link.split('/')[1];
            // e.g. foo.com/path/to/bar
            return domainAndPath.split('/')[0];
        } catch (err) {
            return null;
        }
    }
}
```

Sada je neophodno obaviti i izmene u klasi *ArticleComponent* da bi navedene komponente mogle neometano da budu pozivane.

```
export class ArticleComponent implements OnInit {

    @HostBinding('attr.class') cssClass = 'row';
    article: Article;

    constructor() {
        this.article = new Article(
            'IT255 - Veb sistemi 1',
            'http://www.mwtropolitan.ac.rs',
            10);
    }

    voteUp() {
        this.article.voteUp();
        return false;
    }
}
```

```
voteDown() {  
    this.article.voteDown();  
    return false;  
}  
  
ngOnInit() {  
}  
  
}
```

U suštini, sve je sada isto kao pre sa razlikom što je dobijen čistiji i jasniji kod.

ČUVANJE KOLEKCIJE PROIZVODA

Sada je moguće kreirati kod koji omogućava rad sa listom objekata.

Sada je moguće kreirati kod koji omogućava rad sa listom objekata klase [Articles](#). Prvi korak je korekcija klase [AppComponent](#) koja će sada imati mogućnost čuvanja kolekcije proizvoda.

```
import { Component } from '@angular/core';  
import { Article } from './article/article.model'; //dodata je ova linija  
  
@Component({  
    selector: 'app-root',  
    templateUrl: './app.component.html',  
    styleUrls: ['./app.component.css']  
})  
  
export class AppComponent {  
    articles: Article[]; //osobina komponente  
  
    constructor() {  
        this.articles = [  
            new Article('IT255 - Veb sistemi 1', 'http://www.metropolitan.ac.rs', 3),  
            new Article('Fullstack', 'http://fullstack.io', 2),  
            new Article('Angular Homepage', 'http://angular.io', 1),  
        ];  
    }  
}
```

Prva stvar koju je moguće primetiti jeste da klasa poseduje liniju koda:

```
articles: Article[];
```

Na ovaj način je rečeno da je osobina [articles](#), zapravo, niz objekata klase [Article](#). Ovo je moguće zapisati na još jedan način [Array<Article>](#). Radi se o primeni generičkih tipova - [Array](#) je kolekcija u kojoj je moguće čuvati samo [Article](#) objekte.

Da bi bilo moguće pristupati klasi [Article](#) neophodno je prvo izvršiti njen importovanje (linija koda 2). Kolekciju je moguće napuniti preko konstruktora komponente (linije koda 13 - 18).

PODEŠAVANE KLASE ARTICLECOMPONENT DEKORATOROM INPUT

Glavno pitanje je kako obaviti njihovo prosleđivanje liste proizvoda ka komponenti.

Sada postoji lista objekata `Articles` i glavno pitanje je kako obaviti njihovo prosleđivanje ka komponenti `ArticlesComponent`?

Ponovo na scenu stupa dekorator `@Input`. Na početku, klasa `ArticleComponent` ima sledeći oblik:

```
export class ArticleComponent implements OnInit {  
  
    @HostBinding('attr.class') cssClass = 'row';  
    article: Article;  
  
    constructor() {  
        this.article = new Article(  
            'IT255 - Veb sistemi 1',  
            'http://www.metropolitan.ac.rs',  
            10);  
    }  
}
```

Ovde je glavni problem što postoji "čvrsto" kodiran konkretni proizvod u konstruktoru. Poenta kreiranja komponenata nije samo enkapsulacija, već i višestruka upotrebljivost.

Ono što je zapravo cilj jeste pokušaj podešavanja proizvoda kojeg bi trebalo prikazati na stranici. Ako, na primer, postoje dva proizvoda `article1` i `article2`, trebalo bi da postoji mogućnost ponovne upotrebe komponente `app-article` prosleđivanjem proizvoda kao parametra komponenti, na sledeći način:

```
<app-article [article]="article1"></app-article>  
<app-article [article]="article2"></app-article>
```

Angular poseduje veoma jednostavno rešenje za ovaj problem - primenjuje dekorator `@Input` na osobinu komponente.

```
class ArticleComponent {  
    @Input() article: Article;  
    // ...
```

Sada, ukoliko bi postojao proizvod čuvan u varijabli `myArticle`, mogao bi iz pogleda da bude prosleđen komponenti ArticleComponent na sledeći način:

```
<app-article [article]="myArticle"></app-article>
```

Dalje, a ovo je veoma važno, vrednost `this.article` instance `ArticleComponent` biti podešena sa `myArticle`. O varijabli `myArticle` može se razmišljati kao o prosleđenom parametru (ulazu, inputu) komponenti.

LISTING KLASE ARTICLECOMPONENT

Izmenjeni kod klase ArticleComponent sadrži dekorator Inputs

Nakon detaljne analize, moguće je sada priložiti izmenjeni kod klase `ArticleComponent` u koji je ugrađena primena dekoratora `@Input` na polje tipa `Article`.

```
import { Component,
         OnInit,
         HostBinding,
         Input // dodato je ovo
     } from '@angular/core';
import { Article } from './article.model';

@Component({
  selector: 'app-article',
  templateUrl: './article.component.html',
  styleUrls: ['./article.component.css']
})
export class ArticleComponent implements OnInit {

  @HostBinding('attr.class') cssClass = 'row';
  @Input() article: Article;

  constructor() {
    // ovako se više ne zadaje proizvod
    //zadaje se preko Input - a
  }

  voteUp() {
    this.article.voteUp();
    return false;
  }

  voteDown() {
    this.article.voteDown();
    return false;
  }

  ngOnInit() {
  }
}
```

PRIKAZIVANJE LISTE PROIZVODA

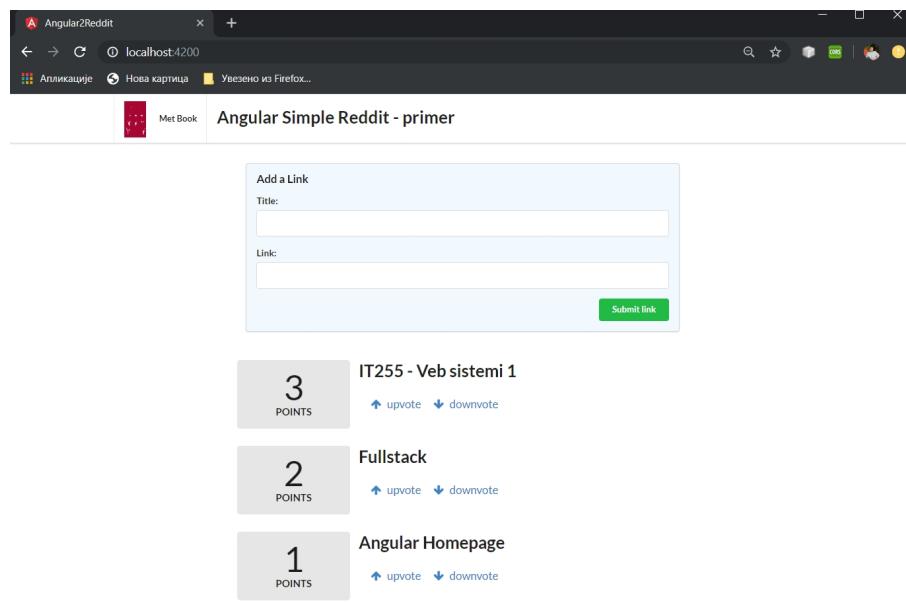
Podešava se AppComponent da bi mogla da omogući prikazivanje svih proizvoda.

U ranijem izlaganju podešena je komponenta *AppComponent* da čuva niz proizvoda. Dalje, podešava se *AppComponent* da bi mogla da omogući prikazivanje svih proizvoda. Da bi ovo bilo omogućeno, biće korišćena *ngFor* direktiva za iteraciju preko liste proizvoda i da prikaže tag *<app-article>* za svakog od njih. Upravo će navedeno biti dodato u šablon komponente *AppComponent* odmah ispod taga *<form>*:

```
...
Submit link
</button>
</form>

<!-- početak dodavanja ovde -->
<div class="ui grid posts">
  <app-article *ngFor="let article of articles" [article]="article">
    </app-article>
</div>
<!-- kraj dodavanja ovde -->
```

Ponovnim učitavanjem stranice, u veb pregledaču, dobijeni su inovirani rezultati našeg rada na projektu. To je prikazano sledećom slikom.



Slika 7.2.2 Inovirana aplikacija

✓ 7.2 Korišćenje forme

DODAVANJE NOVIH PROIZVODA

*Dodavanje novih proizvoda je rešeno posebnom metodom klase *AppComponent*.*

Sada je potreno posvetiti pažnju metodi `addArticle()` kojom će biti omogućeno dodavanje novih proizvoda u kada se klikne na dugme za potvrdu forme.

Metoda može da bude smeštena na kraj sadržaja datoteke `app.component.ts` ima sledeći listing:

```
addArticle(title: HTMLInputElement, link: HTMLInputElement): boolean {
    console.log(`Adding article title: ${title.value} and link: ${link.value}`);
    this.articles.push(new Article(title.value, link.value, 0));
    title.value = '';
    link.value = '';
    return false;
}
```

Izvršavanjem metode se postiže:

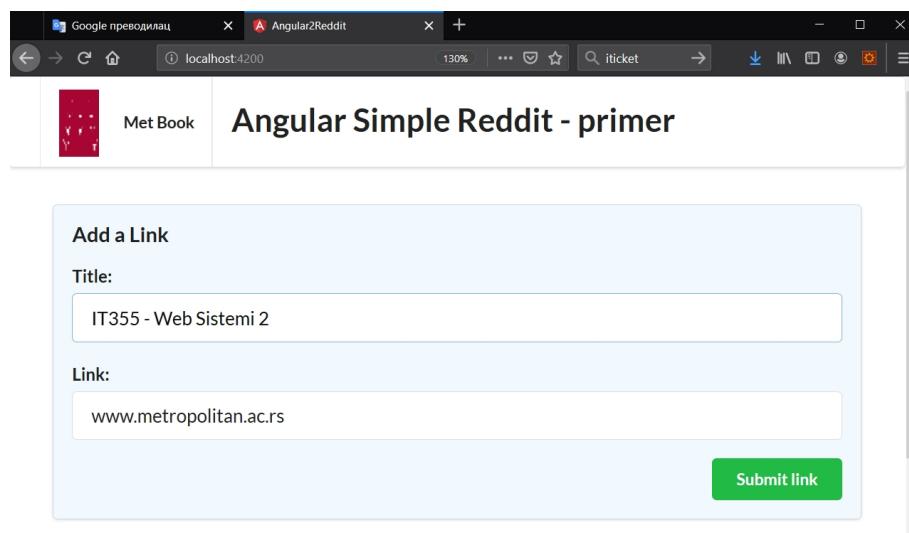
1. kreiranje novog `Article` objekta sa prosleđenim naslovom i linkom;
2. dodaje kreirani objekat u postojeći niz objekata;
3. resetuje polja za unos podataka za nov `Article` objekat.

Sada je aplikacija dostigla željeni nivo zrelosti i može biti demonstrirana.

DEMONSTRACIJA KOMPLETIRANE APLIKACIJE

Slede izgledi ekrana koji demonstriraju izvršavanje aplikacije.

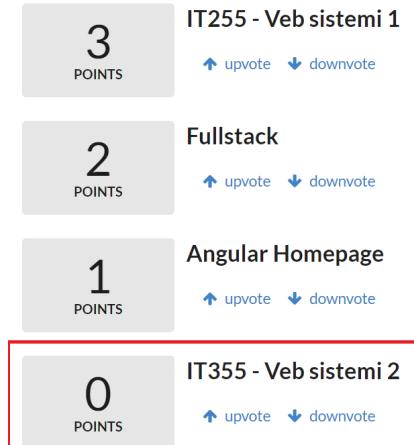
Ponovo je potrebno osvežiti veb pregledač. Učitava se stranica sa formom u kojoj je moguće uneti nov proizvod. To može biti dokumentovano sledećom slikom.



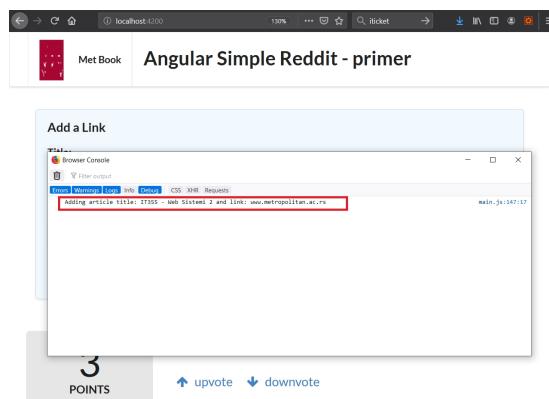
Slika 7.3.1 Dodavanje novog objekta putem forme

Klikom na dugme `Submit link` poziva se metoda `addArticle()` glavne komponente `AppComponent`. Metoda pokazuje rezultate na dva načina:

1. dodaje nov objekat u kolekciju i on biva vidljiv zajedno sa ostalim objektima (slika 2);
2. šalje rezultat u konzolu veb pregledača (slika 3).



Slika 7.3.2 Nov objekat u listi objekata



Slika 7.3.3 Rezultat slanja objekta sa forme u logu veb pregledača.

PRIKAZIVANJE DOMENA PROIZVODA

Pored naziva linka, prikazuje se i njegov domen.

Kao zanimljiv dodatak moguće je dodati napomenu pored linka u formi domena na koji će korisnik biti preusmeren ukoliko klikne na link. Kod ove metode je već viđen u klasi Article, a ovde zbog preglednosti može biti sagledan izolovan:

```
// domain() je funkcija koja izdvaja
// domen iz URL, biće uskoro objašnjena
domain(): string {
    try {
        // e.g. http://foo.com/path/to/bar
        const domainAndPath: string = this.link.split('/')[1];
        // e.g. foo.com/path/to/bar
        return domainAndPath.split('/')[0];
    }
}
```

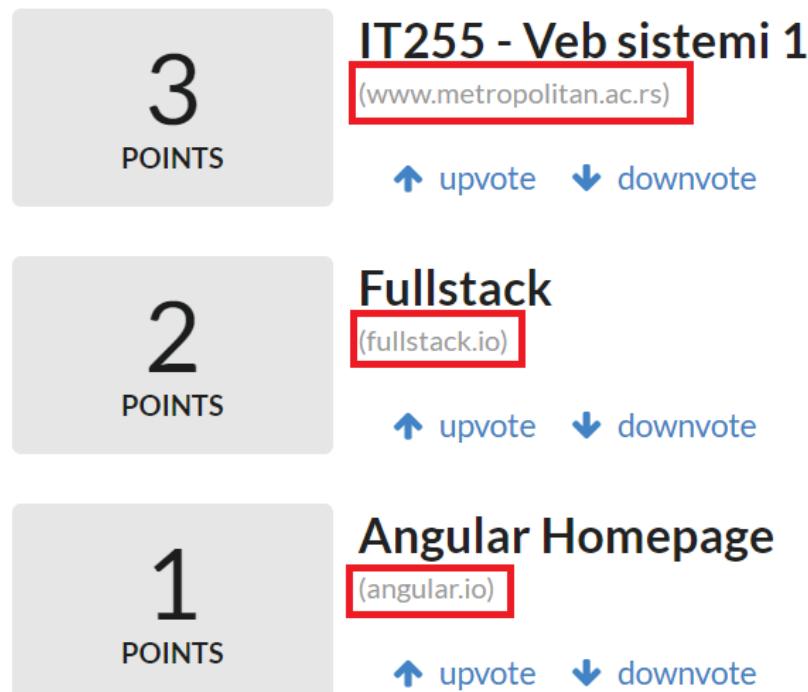
```
    } catch (err) {
        return null;
    }
}
```

Sada ide mala izmena u šablonu *ArticleComponent*.

```
<div class="twelve wide column">
  <a class="ui large header" href="{{ article.link }}"
  |   {{ article.title }}
  |>
</a>
<!-- domen proizvoda -->
<div class="meta">{{ article.domain() }}</div>
<ul class="ui big horizontal list voters">
  <li class="item">
    <a href (click)="voteUp()">
      <i class="arrow up icon"></i>
      upvote
    </a>
  </li>
</ul>
```

Slika 7.3.4 Dodavanje domena uz link u šablonu

Konačno, rezultat rada je moguće sagledati na sledećoj slici.



Slika 7.3.5 Domen i link na istom mestu

▼ Poglavlje 8

Objavljivanje aplikacije

PROCES OBJAVLJIVANJA APLIKACIJE

Objavljivanje aplikacije (deploying) predstavlja čin postavljanja kreiranog koda na server.

Objavljivanje aplikacije (deploying) predstavlja čin postavljanja kreiranog koda na server odakle mogu da joj pristupe razni korisnici. Šire rečeno, ideja je u izvođenju sledećih koraka:

- kompajliranje celokupnog TypeScript koda u JavaScript kojeg veb pregledač razume;
- pakovanje celokupnog JavaScript koda u jedan ili dva fajla;
- podizanje (upload) celokupnog JavaScript koda, HTML, CSS i slika na server.

Konačno, Angular aplikacija je HTML datoteka koja učitava JavaScript kod. Cilj je podizanje kreiranog koda na nekom računaru na Internetu.

KREIRANJE VERZIJE PROGRAMA ZA PRODUKCIJU

Angular CLI je upotrebljen za generisanje verzije kreirane aplikacije koja je spremna za produkciju

Angular CLI je alat koji može, između ostalog, da bude upotrebljen za generisanje verzije kreirane aplikacije koja je spremna za produkciju. Sama procedura je veoma jednostavna. Neophodno je ući u radni folder aplikacije, u ovom slučaju to je Angular\angular-reddit. Nakon toga se navodi naredba, ovde data u opštem obliku:

```
ng build --base-href /naziv-foldera/ --prod
```

U konkretnom slučaju to je urađeno kao na sledećoj slici.

```
C:\Users\Vlada\Documents\Angular\angular-reddit>ng build --base-href /prvaapp/ --prod
chunk {0} runtime-es2015.10a1f01eef199006286d.js (runtime) 1.41 kB [entry] [rendered]
chunk {1} main-es2015.a37f9e8f2880060c4e1b.js (main) 215 kB [initial] [rendered]
chunk {2} polyfills-es2015.e4a1185e6871d06f842f.js (polyfills) 36.4 kB [initial] [rendered]
chunk {3} styles.a77dd01d66ff048d780e.css (styles) 469 kB [initial] [rendered]
Date: 2019-08-05T19:54:38.705Z - Hash: 83e21855452f6cce70c9 - Time: 31676ms

chunk {0} runtime-es5.9ad22a88fcc70a015907.js (runtime) 1.41 kB [entry] [rendered]
chunk {1} main-es5.fe97e04018fd0ee317ad.js (main) 249 kB [initial] [rendered]
chunk {2} polyfills-es5.da479429bfd39fbf99d9.js (polyfills) 113 kB [initial] [rendered]
Date: 2019-08-05T19:55:05.067Z - Hash: f07b727bffc8fc0912fc - Time: 26248ms

C:\Users\Vlada\Documents\Angular\angular-reddit>
```

Slika 8.1 Kreiranje verzije programa za produkciju

Naredba će za izvršavanje trebati malo vremena ali će uskoro, kao rezultat njenog izvršavanja, biti kreiran folder *dist* u okviru radnog foldera i njegov sadržaj je prikazan sledećom slikom. Datoteke unutar ovog foldera predstavljaju potpun rezultat prevođenja kreirane aplikacije.

```
▲ dist
  ▲ angular-reddit
    ▶ assets
      3rdpartylicenses.txt
      ★ favicon.ico
      flags.9c74e172f87984c48ddf.png
      ▲ icons.97493d3f11c0a3bd5cbd.woff2
      ▲ icons.706450d7bba6374ca02f.ttf
      ▲ icons.2980083682e94d33a66e.svg
      ▲ icons.d9ee23d59d0e0e727b51.woff
      ▲ icons.f7c2b4b747b1a225eb8d.eot
      ◁ index.html
      JS main-es5.fe97e04018fd0ee317ad.js
      JS main-es2015.a37f9e8f2880060c4e1b.js
      JS polyfills-es5.da479429bfd39fbf99d9.js
      JS polyfills-es2015.e4a1185e6871d06f842f.js
      JS runtime-es5.9ad22a88fcc70a015907.js
      JS runtime-es2015.10a1f01eef199006286d.js
      # styles.a77dd01d66ff048d780e.css
```

Slika 8.2 Potpun rezultat prevođenja kreirane aplikacije

POSTAVLJANJE APLIKACIJE NA SERVER

Postoji puno načina da se hostuje HTML i JavaScript.

Postoji puno načina da se *hostuje* HTML i JavaScript. Za ovaj primer je možda najlakši za upotrebu server *now* dostupan na linku: <https://zeit.co/now>.

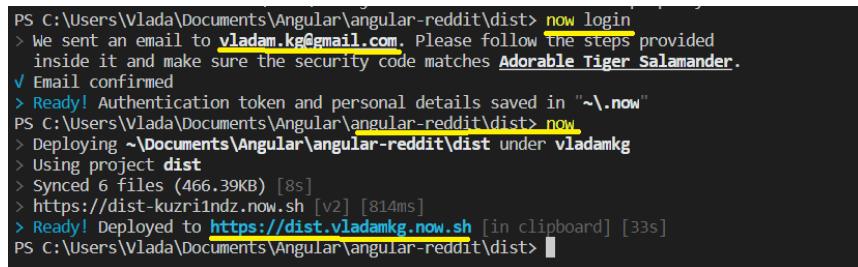
Instaliranje podrške za podizanje aplikacije na ovom serveru ide primenom *npm* naredbe:

```
npm install -g now
```

Zatim je, kada je ova podrška dostupna neophodno otići u folder *dist* i pokrenuti naredbu *now* kojom se aplikacija postavlja na server:

```
cd dist # change into the dist folder
now
```

Komanda će ubrzo pitati i za neke korisničke informacije, poput email adrese kojom je kreiran *now* nalog. Nakon potvrđivanja naloga započinje postavljanje aplikacije na serveru. Navedeno je prikazano sledećom slikom.



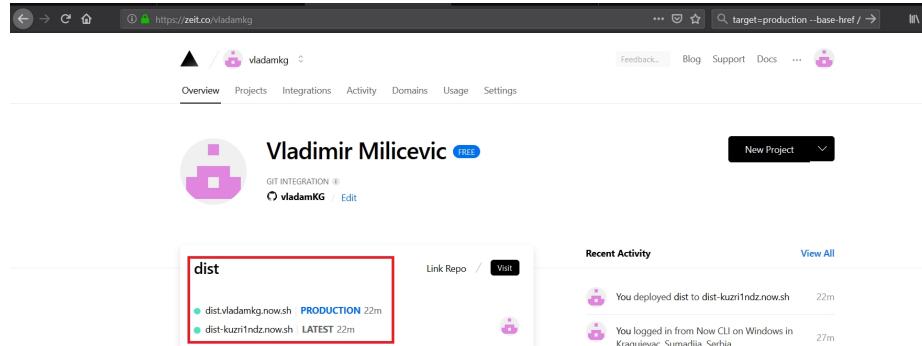
```

PS C:\Users\Vlada\Documents\Angular\angular-reddit\dist> now login
> We sent an email to vladam.kg@gmail.com. Please follow the steps provided
inside it and make sure the security code matches Adorable Tiger Salamander.
✓ Email confirmed
> Ready! Authentication token and personal details saved in "~\.now"
PS C:\Users\Vlada\Documents\Angular\angular-reddit\dist> now
> Deploying ~\Documents\Angular\angular-reddit\dist under vladamkg
> Using project dist
> Synced 6 files (466.39KB) [8s]
> https://dist-kuzri1ndz.now.sh [v2] [814ms]
> Ready! Deployed to https://dist.vladamkg.now.sh [in clipboard] [33s]
PS C:\Users\Vlada\Documents\Angular\angular-reddit\dist>

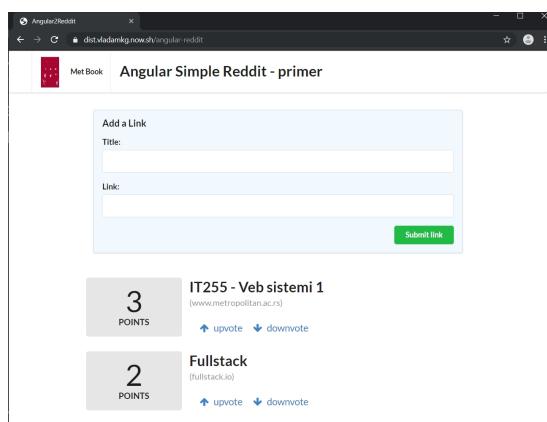
```

Slika 8.3 Postavljanje aplikacije na serveru

Sa slike je moguće videti i URL na kojem je dostupna aplikacija. Pozivom tog URL u veb pregledaču, pokreće se aplikacija sa servera.



Slika 8.4 Podignuta aplikacija na now serveru



Slika 8.5 Pokrenuta aplikacija sa servera

▼ Poglavlje 9

Vežbe 5

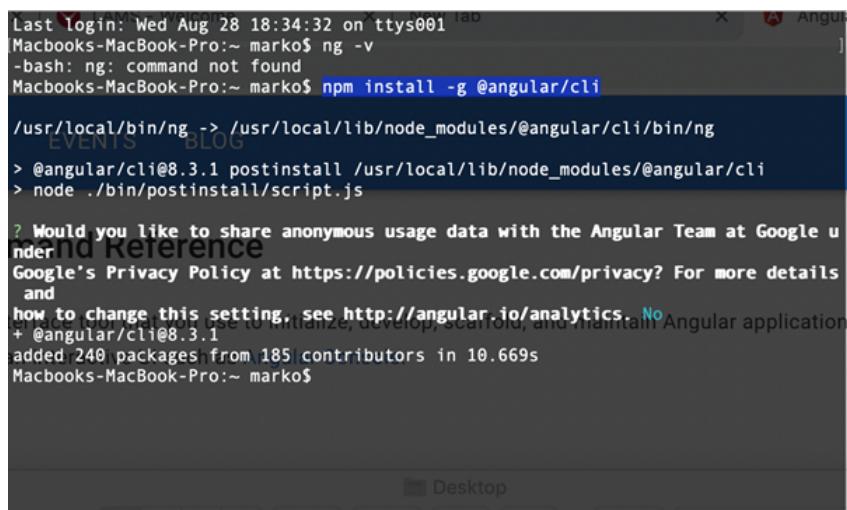
PRIPREMA ANGULAR CLI - POKAZNA VEŽBA

Cilj ove lekcije predstavlja kreiranje projekta kroz Angular CLI

Ukoliko to već nije učinjeno, prvi korak je preuzimanje i instaliranje [NodeJS](#) instalacionog paketa.

U nastavku je neophodno instalirati [Angular CLI](#) (*Angular Command Line Interface*). To možete učiniti navođenjem sledeće naredbe u terminalu vašeg razvojnog okruženja [Visual Studio Code](#):

```
npm install -g @angular/cli
```



```
Last login: Wed Aug 28 18:34:32 on ttys001
Macbooks-MacBook-Pro:~ marko$ ng -v
-bash: ng: command not found
Macbooks-MacBook-Pro:~ marko$ npm install -g @angular/cli
/usr/local/bin/ng -> /usr/local/lib/node_modules/@angular/cli/bin/ng
> @angular/cli@8.3.1 postinstall /usr/local/lib/node_modules/@angular/cli
> node ./bin/postinstall/script.js

? Would you like to share anonymous usage data with the Angular Team at Google under Google's Privacy Policy at https://policies.google.com/privacy? For more details and how to change this setting, see http://angular.io/analytics. No
+ @angular/cli@8.3.1
added 240 packages from 185 contributors in 10.669s
Macbooks-MacBook-Pro:~ marko$
```

Slika 9.1 Instaliranje Angular CLI

Po uspešnoj instalaciji, pokretanjem komande [`ng help`](#), možemo videti da je instalacija uspešna kao na slici 2.

```

added 240 packages from 103 contributors in 10.005s
[Macbooks-MacBook-Pro:~ marko$ ng -v
Available Commands:
  add Adds support for an external library to your project, you use to initialize, develop, si
  analytics Configures the gathering of Angular CLI usage metrics. See https://v8.angular.i
o/cli/usage-analytics-gathering.
  build (b) Compiles an Angular app into an output directory named dist/ at the given outpu
t path. Must be executed from within a workspace directory.
  >deploy (d) Invokes the deploy builder for a specified project or for the default project
in the workspace.
  config Retrieves or sets Angular configuration values in the angular.json file for the wo
rkspace.
  doc (d) Opens the official Angular documentation (angular.io) in a browser, and searches
for a given keyword.
  >e2e (e) Builds and serves an Angular app, then runs end-to-end tests using Protractor.
  generate (g) Generates and/or modifies files based on a schematic.
  help Lists available commands and their short descriptions.
  >lint (l) Runs linting tools on Angular app code in a given project folder.
  new (n) Creates a new workspace and an initial Angular app.
  run Runs an Architect target with an optional custom builder configuration defined in you
r project.
  >serve (s) Builds and serves your app, rebuilding on file changes.
  test (t) Runs unit tests in a project.
  update Updates your application and its dependencies. See https://update.angular.io/
  >version (v) Outputs Angular CLI version.
  xi18n Extracts i18n messages from source code.

For more detailed help run "ng [command name] --help"
Macbooks-MacBook-Pro:~ marko$
```

Slika 9.2 Provera instalacije Angular CLI

KREIRANJE PROJEKTA

Nakon uspešne instalacije, pristupamo kreiranju projekta.

Nakon uspešne instalacije, pristupamo kreiranju projekta sledećom komandom:

```
ng new it255
```

Prethodno navedena komanda će kreirati folder, sa svim propratnim kodom neophodnim za start određene aplikacije. Na slici 3 možemo videti da je neophodno izabrati Angular rutiranje, kao i SCSS stylesheet format, kako bismo dobili komajler za SCSS o kom je bilo reči u nekoj od prethodnih lekcija.

```

Macbooks-MacBook-Pro:~/Desktop marko$ cd Workspace/
Macbooks-MacBook-Pro:Workspace marko$ ng new it255
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? SCSS  [ https://sass-lang.com/documentation/on-syntax#scss ]
```

Slika 9.3 Angular rutiranje - izbor

Po uspešno izvršenoj komandi, unutar direktorijuma pokrećemo sledeću komandu: „npm start“ za pokretanje projekta.

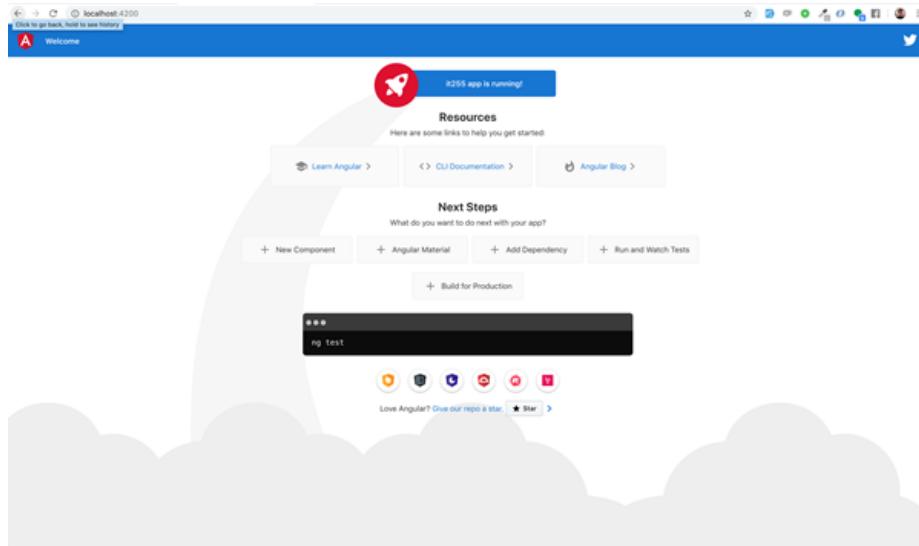
```

Macbooks-MacBook-Pro:it255 marko$ npm start
> it255@0.0.0 start /Users/marko/Workspace/it255
> ng serve
+ New Component + Angular Material + Build for Production
10% building 3/3 modules 0 active [wds]: Project is running at http://localhost:4200/webpack-dev-server/
[ wds]: webpack output is served from /
[ wds]: 404s will fallback to //index.html
chunk {main} main.js, main.js.map (main) 49.4 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 264 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 10 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 4.08 MB [initial] [rendered]
Date: 2019-08-28T17:07:24.458Z - Hash: 725a4b4889ed54d8d86b - Time: 7014ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
[ wdm]: Compiled successfully.
```

Slika 9.4 Prevodenje i pokretanje projekta

Vidimo da je naša aplikacija dostupna na sledećoj adresi:<http://localhost:4200/>.

Navođenjem navedene adrese u veb pregledač, dobija se ekran kao na sledećoj slici:



Slika 9.5 Inicijalni izgled kreiranog Angular projekta

KREIRANJE PRVOG VLASTITOG ANGULAR (TYPESCRIPT) KODA

Izmenom inicijalnog i dodavanjem novog koda prilagođavamo aplikaciju vlastitim zahtevima.

Izmenom inicijalnog i dodavanjem novog koda prilagođavamo aplikaciju vlastitim zahtevima.

Unutar `app.component.ts` dodaćemo sledeći sadržaj, tako da krajnji rezultat fajla izgleda kao sledeći listing:

```
public videos = [
  {
    title: "Video - 1",
    description: "Lorem ipsum, lorem ipsum",
    link: "https://www.youtube.com/embed/ltV0Xz4mx0o"
  },
  {
    title: "Video - 2",
    description: "Lorem ipsum, lorem ipsum",
    link: "https://www.youtube.com/embed/ltV0Xz4mx0o"
  },
  {
    title: "Video - 3",
    description: "Lorem ipsum, lorem ipsum",
    link: "https://www.youtube.com/embed/ltV0Xz4mx0o"
  }
];
```

```

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  public videos = [
    {
      title: "Video - 1",
      description: "Lorem ipsum, lorem ipsum",
      link: "https://www.youtube.com/embed/1v0tXeudu"
    },
    {
      title: "Video - 2",
      description: "Lorem ipsum, lorem ipsum",
      link: "https://www.youtube.com/embed/1v0tXeudu"
    },
    {
      title: "Video - 3",
      description: "Lorem ipsum, lorem ipsum",
      link: "https://www.youtube.com/embed/1v0tXeudu"
    }
  ];
}

```

Slika 9.6 Izgled datoteke app.component.ts

KREIRANJE KOMPONENTE

Potom ćemo koristeći komandu za generisanje komponenti, kreirati novu komponentu.

Potom ćemo koristeći komandu za generisanje komponenti, kreirati novu, koja će predstavljati pogled ([view](#)) za jedan video snimak.

```
ng g c components/video-box
```

Sadržaj komponente *video-box.component.ts* dat je ispod:

```

import { Component, OnInit, Input } from '@angular/core';
import { DomSanitizer } from '@angular/platform-browser';

@Component({
  selector: 'app-video-box',
  templateUrl: './video-box.component.html',
  styleUrls: ['./video-box.component.scss']
})
export class VideoBoxComponent implements OnInit {

  @Input() video: any;
  public link: any;

  constructor(private _sanitizer: DomSanitizer) { }

  ngOnInit() {
    this.embedUrl();
  }

  ngOnDestroy() {
    this._sanitizer.sanitize('img');
  }

  embedUrl() {
    this.link = this._sanitizer.bypassSecurityTrustResourceUrl(this.video);
  }
}

```

```
public embedUrl() {  
    this.link = this._sanitizer.bypassSecurityTrustResourceUrl(this.video.link);  
}  
}
```

HTML fajl iste komponente ([šablon](#)) prikazan je ispod:

```
<div class="media video-box mb-3">  
    <iframe height="100%" [src]="link"></iframe>  
    <div class="media-body pl-2">  
        <h5 class="mt-0">{{video.title}}</h5>  
        {{video.description}}  
    </div>  
</div>
```

Takođe, neophodno je da izmenimo i [app.component.html](#) tako da možemo da iteriramo i dodajemo (renderujemo) komponente u zavisnosti od sadržaja.

```
<div class="container mt-4 mb-4">  
    <app-video-box *ngFor="let video of videos; index as i;"  
    [video]="video"></app-video-box>  
</div>
```

FILTRIRANJE SADRŽAJA

U želji da filtriramo sadržaj, kreiraćemo filter pipe.

U želji da filtriramo sadržaj, kreiraćemo filter mehanizam ([filter pipe](#)) sa sledećom komandom:

```
ng generate pipe helpers/filter-pipe
```

Unutar kreiranog fajla, izmenićemo sadržaj po uzoru na dole navedeni:

```
import { Pipe, PipeTransform } from '@angular/core';  
@Pipe({  
    name: 'filter'  
)  
export class FilterPipe implements PipeTransform {  
  
    transform(items: any[], searchText: string): any[] {  
        if(!items) return [];  
        if(!searchText) return items;  
        searchText = searchText.toLowerCase();  
        return items.filter( it => {  
            return it.title.toLowerCase().includes(searchText);  
        });  
    }  
}
```

Takođe, neophodno je da ažuriramo fajl app **.component.html** file i uključimo naš mehanizam za filtriranje u njegov sadržaj:

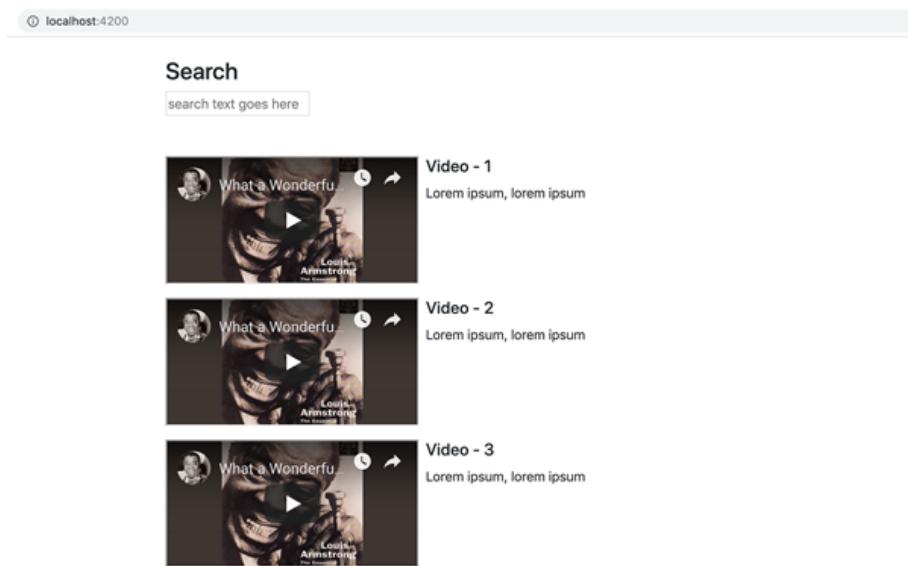
```
<div class="container mt-4 mb-4">
  <div class=" mb-5">
    <h3>Search</h3>
    <input [(ngModel)]="searchText" placeholder="search text goes here">
  </div>

  <app-video-box *ngFor="let video of videos | filter : searchText; index as i;" [video]="video"></app-video-box>
</div>
```

DEMO APLIKACIJE

Provera ispravnosti kreiranog koda.

Krajnji rezultat aplikacije prikazan je na slici ispod:



Slika 9.7 Izgled urađene aplikacije

Projekat je dostupan na sledećem linku: <https://github.com/markorajevic/it255-v05>

INDIVIDUALNA VEŽBA

Samostalna dopuna zadatka sa pokaznih vežbi.

Otvorite zadatak koji ste radili na pokaznim vežbama ili preuzmite delimično urađen sa linka: <https://github.com/markorajevic/it255-v05>.

Dodajte nove funkcionalnosti u kreirani *Angular* projekat po sledećim zahtevima:

1. Kreirati formu kroz koju će biti dodavane pesme unutar liste.
2. Kreirati dugme koje će izvršiti slučajni (*random*) raspored video snimaka unutar liste.

✓ Poglavlje 10

Domaći zadatak 5

DOMAĆI ZADATAK

Samostalna izrada domaćeg zadatka.

Uradite Angular projekat na osnovu sledećih zahteva:

1. Kreirati projekat pod imenom *MetHotels*, koji će na početnoj strani prikazivati listu smeštaja.
2. Neophodno je implementirati formu koja je prikazana u predavanju za dodavanje novih soba.
3. Takođe, neophodno je implementirati i *FilterPipe* koji će filtrirati smeštajne jedinice po ceni, tj. uneta vrednost treba da predstavlja maksimalnu granicu cene sobe. Dodati dugme koje će prilikom klika random izmeniti redosled soba u pokaznoj listi.

Domaći zadatak postaviti na lični *Github* nalog, gde naziv *commit*-a treba da bude „*IT255-DZ05*“. Link do zadatka poslati predmetnom asistentu na mail.

✓ Poglavlje 11

Zaključak

ZAKLJUČAK

U lekciju su postavljene osnove za dalji razvoj Angular aplikacija.

U lekciji je bilo detaljno govora o postavljanju ozbiljnih osnova za dalji razvoj složenijih Angular aplikacija. Naučeno je kako se:

1. deli aplikacija na komponente;
2. kreiraju pogledi;
3. definiše model;
4. prikazuje model;
5. dodaje i koristi interakcija.

Sve vreme lekcija je insistirala na kvalitetnim pokaznim primerima za ilustraciju i lakše savladavanje navedene problematike.

LITERATURA

Za pripremu lekcije korišćena je aktuelna pisana i elektronska literatura.

Pisana literatura:

1. Nate Murray, Felipe Coury, Ari Lerner, Carlos Taborda, ng-Book – The Complete Book on Angular 6, Fullstack.io, 2018
2. Cody Lindley, Frontend Handbook – 2017, Frontend masters, 2017
3. Sandeep Panda, AngularJS – From Novice to Ninja, SitePoint Pty. Ltd, 2014

Elektronska literatura:

4. <https://angular.io/>
5. <https://angular.io/tutorial>
6. <https://www.w3schools.com/angular/>
7. <https://www.tutorialspoint.com/angular4/>
8. <https://nodejs.org/en/>
9. <https://code.visualstudio.com/>