

9. Rutiranje: napraviti sledeću aplikaciju:



Statički sadržaj

Ovde će se nalaziti navbar

Dinamički sadržaj koji se menja rutiranjem

Na početnoj strani nalazimo se u “main” stanju. Klikom na linkove prelazimo između stanja

- “main”:
 - link localhost:8080/#/main
 - Sadržaj: “ovde će se nalaziti navbar”
- “wines”:
 - link localhost:8080/#/wines
 - Sadržaj “Ovde će biti prikazana tabela sa vinima”
- “addWine”:
 - link localhost:8080/#/add
 - Sadržaj: “Ovde će biti prikazana forma za dodavanje novog vina”

Koraci:

- Napraviti stranicu index.html. Dodati angular na stranicu i proveriti da li je aktivan (ispisati vrednost 555/33). Za detalje ovog koraka videti zadatak 1.
- Napraviti modul “winesModule” i povezati ga sa aplikacijom (videti zadatak 2).
- U index.html uvući biblioteku angular-ui-router.js (pazite da se ova biblioteka nalazi na navedenom mestu). Proveriti da nema greške u konzoli.
- U app.js u listu zavisnosti od modula dodati 'ui.router' modul. Proveriti da nema greške u konzoli.
- Konfigurisati aplikaciju u app.js – dodati app.config(...)
 - U app.config se prosleđuje callback funkcija koja služi za podešavanje rutiranja. U ovu funkciju injektovati provajdere \$stateProvider i \$urlRouterProvider
 - Iskoristiti \$urlRouterProvider da se definiše link na koji će se korisnik prebaciti kada ukuca neku adresu koja nije definisana rutiranjem. Neka ta adresa bude “/main”.
 - Iskoristiti \$stateProvider za definisanje mogućih stanja aplikacije. Primer za stanje main:

```
.state("main", {  
  url: "/main",  
  template: "<p>Ovde će se nalaziti navbar</p>"  
})
```

Slično definisati stanja “wines” i “addWine”.
- U index.html dodati div tag sa ui-view direktivom. Ovde će se smenjivati prikazi stanja.
- U index.html dodati linkove na koje se može kliknuti kako bi se prešlo u određeno stanje. Primer za main stanje: <a ui-sref="main">Main. Ovde je iskorišćena ui-sref direktiva kojoj je kao vrednost dodeljeno ime stanja. Pošto stanje main ima definisan url (/main), kada korisnik klikne na ovaj link, biće izgenerisana odgovarajuća vrednost za href atribut. Slično definisati linkove za “wines” i “addWine”.

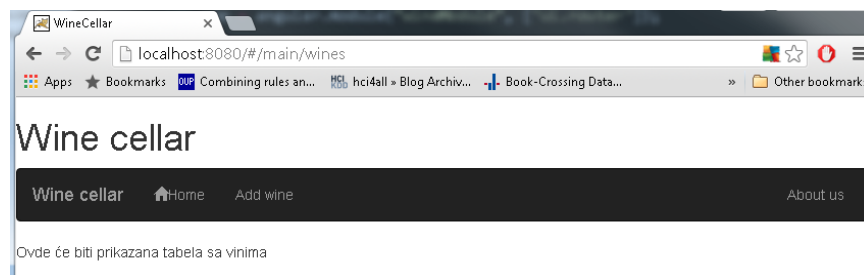
10. Namestiti da „main“ stanje bude roditeljsko stanje stanjima „wines“ i „addWine“



Koraci:

- U app.js stanja “wines” i “addWine” su preimenovana u “main.wines” i “main.addWine”, respektivno. Ovim smo rekli da je main roditeljsko stanje.
- U app.js smo kod stanja main umesto da direktnog navođenja templata, definisali putanju do main.html (za stanje koristiti property templateUrl umesto dosadašnjeg property-ja template). U main.html prebaciti paragraf “Ovde će se nalaziti navbar”.
- U index.html jedini statički sadržaj je ostao heading “Wine cellar”. Link ka main stanju je uklonjen. Linkovi ka stanjima “wines” i “addWine” su premešteni u novu stranicu main.html.
- Stranica main.html:
 - Prepraviti linkove ka stanjima “wines” i “addWine”. Kako bi se prebacivali između child stanja roditeljskog stanja (main) koristiti tačka notaciju, dake “.wines” i “.addWine”, respektivno.
 - Potrebno je dodati div sa direktivom ui-view. U ovom div-u će se smanjivati child stanja.

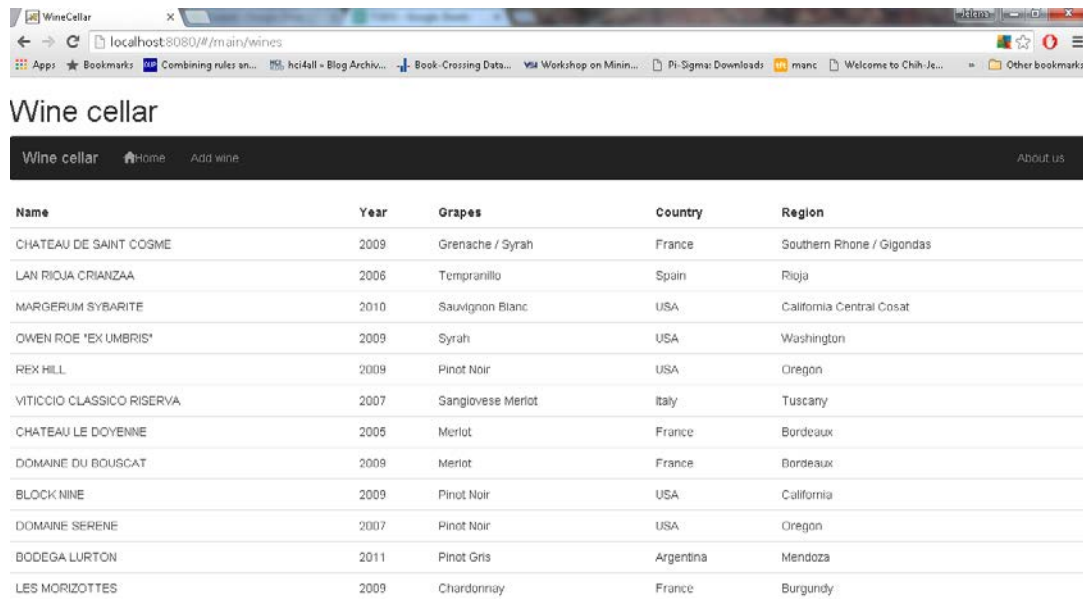
11. Link i tekst u main stanju zameniti sa navbarom. Namestiti da stanje main bude apstraktno. Prilikom učitavanja stranice inicijalno je aktivno stanje “wines”.



Koraci:

- app.js: Kako bi stanje main bilo apstraktno dodat mu je property abstract sa vrednošću true (abstract : true). Prepraviti da početna stranica (\$urlRouterProvider.otherwise) bude /main/wines umesto /main.
- U index.html uključiti biblioteke neophodne za bootstrap (potražiti detalje na 6. Slajdu sa Bootstrap predavanja). Nemojte zaboraviti da kopirate bootstrap i jquery biblioteke u folder Vaše aplikacije.
- U main.html dodati navbar. Primer imate u 14_BootstrapNavigationBars.html, poslednji primer – navbar koji se sažima na malim okvirima za prikaz . Podesiti linkove u navbar-u prema starim linkovima iz main.html i ukoniti te stare linkove.

12. Po ugledu na zadatak 8, u stanju “wines” prikazivati tabelu sa vinima. Podaci o vinima treba da budu dovučeni sa servera (za sada ne koristimo filtriranje, sortiranje i paginaciju). Za potrebe komunikacije sa serverom koristiti custom servis iz zadatka 8.



Name	Year	Grapes	Country	Region
CHATEAU DE SAINT COSME	2009	Grenache / Syrah	France	Southern Rhone / Gigondas
LAN RIOJA CRIANZAA	2006	Tempranillo	Spain	Rioja
MARGERUM SYBARITE	2010	Sauvignon Blanc	USA	California Central Coast
OWEN ROE 'EX UMBRIS'	2009	Syrah	USA	Washington
REX HILL	2009	Pinot Noir	USA	Oregon
VITICCIO CLASSICO RISERVA	2007	Sangiovese Merlot	Italy	Tuscany
CHATEAU LE DOYENNE	2005	Merlot	France	Bordeaux
DOMAINE DU BOUSCAT	2009	Merlot	France	Bordeaux
BLOCK NINE	2009	Pinot Noir	USA	California
DOMAINE SERENE	2007	Pinot Noir	USA	Oregon
BODEGA LURTON	2011	Pinot Gris	Argentina	Mendoza
LES MORIZOTTES	2009	Chardonnay	France	Burgundy

Koraci:

- Iz zadatka 8 preuzeti WinesController i WinesDataService. Ove fajlove nije potrebno menjati.
- U index.html uključiti skripte WinesController.js i WinesDataService.js
- U app.js: Za stanje “main.wines” promeniti template tako da ukazuje na stranicu wines.html. Ovom stanju dodati i property controller – definisati da kontroler WinesController kontroliše prikaz datog templejta.
- U stranicu wines iskopirati tabelu napravljenu u 8. zadatku u index.html. Paziti da se WinesController kontroler ne specificira putem ng-controller directive jer je već zadat putem controller property-ja u definiciji stanja.

13. U stanju “addWines” prikazivati formu za unos vina.



Wine cellar

Name:

Year:

Grapes:

Country:

Region:

Description:

Koraci:

- U app.js: Za stanje “main.addWine” promeniti template tako da ukazuje na stranicu wineDetails.html.
- Kreirati stranicu wineDetails.html. Možete je u potpunosti preuzeti iz primera 03_resolve (predavanje 05_Angular_rutiranje).

14. Omogućiti dodavanje novog vina (kada korisnik klikne na dugme “Ok” slati novo vino na server). Onemogućiti započinjanje akcije dodavanja vina pre nego što su sva polja popunjena. Nakon uspešnog dodavanja novog vina, aplikacija se prebacuje na prikaz tabele sa vinima (novo vino treba da se nalazi u tabeli). Ako je dodavanje vina neuspešno, prikazati poruku o grešci i ostati na formi za unos vina. Koraci:
- U app.js: stanju addWine dodati property controller – definisati da kontroler WineDetailsController kontroliše prikaz ovog templejta.
 - Napraviti kontroler WineDetailsController i injektovati \$scope u njega. Uključiti skriptu WineDetailsController.js u index.html. Proveriti konzolu da ne prikazuje grešku.
 - Pogledati stranicu WineDetails.html. Svaki input ima ng-model direktivu, npr. input za ime ima ng-model=”wine.name” (ako to nije slučaj, dodajte!). To znači da će se u property-ima objekta \$scope.wine u kontroleru koji kontroliše ovaj deo prikaza (WineDetailsController definisan kroz stanje) naći ono što korisnik unese u inpute.
 - Svakom inputu u WineDetails.html dodati atribut required. Ovo sprečava da se okine submit događaj pre nego što su data polja popunjena (http://www.w3schools.com/tags/att_input_required.asp). Dodati dugme na formu i dodeliti mu tip „submit” kako bi klik na njega okidao submit događaj. Isprobajte šta će se desiti ako kliknete na dugme, a neko od polja nije popunjeno.
 - Klik na dugme izaziva submit događaj – na formu dodati ng-submit direktivu i staviti da se na submit poziva funkcija submit definisana na \$scope-u WineDetailsController-a. U funkciji možete za sada staviti samo jedan alert u kome se ispisuje ime vina koje se unosi (koristiti \$window za alert). Probati.
 - U WinesDataService dodati funkciju addWine koja prima vino koje treba da se doda na server. Nemojte zaboraviti da ovu funkciju dodate u return kako bi je mogle pozivati spoljne komponente. U ovoj funkciji treba da napravite POST zahtev na <http://localhost:3000/api/wines/> - videti slajd 17 “API servera – snimanje podataka” sa AJAX predavanja. Poslati novo vino kao parameter POST zahteva. Kao rezultat metode vratiti rezultat izvršenog POST zahteva – povratna vrednost je promise.
 - U funkciji \$scope.submit WineDetailsController-a pozvati upravo napravljnu metodu addWine i poslati joj novo vino kao parameter:
 - Injektovati WinesDataService u kontroler
 - Umesto alert-a pozvati metodu addWine i poslati joj \$scope.addWine
 - Povratna vrednost addWine je promise: pomoću then metode registrovati successCallback i errorCallback.
 - U onSuccess izvršiti redirekciju na stanje “main.wines” pomoću \$state.go funkcije (ne zaboravite da injektujete \$state servis).
 - U onError stavite alert da vino nije dodato.
 - Proverite da li je izvršena redirekcija i da li se novo vino nalazi u tabeli.
15. Omogućiti ažuriranje postojećeg vina: u tabeli, pored svakog vina u novoj koloni tabele dodati link “update”. Klikom na link “update” pored vina sa vrednošću _id=3, prebacujemo se na URL localhost:8080/#/main/update/3. Prikazuje se ista forma kao ona za dodavanje vina (wineDetails.html). Inputi na formi su popunjeni vrednostima selektovanog vina. Za ovo je pre redirekcije na stranicu sa detaljima neophodno uputiti get zahtev ka serveru – koristiti resolve. Kada korisnik izmeni vino i klikne na “Ok”, ažurirati dato vino na serveru.
- WinesDataService proširiti još jednom metodom get koja kao parametar prima id vina koje je potrebno dobiti sa servera. Kao rezultat se vraća povratna vrednost http GET zahteva (promise objekat). Nemojte zaboraviti da funkciju učinite “javnom”.

- b. U app.js dodati stanje "main.updateWine"
- Kao url postaviti /wines/:id – id je parameter url-a koji može da se menja i na osnovu koga znamo koje vino se ažurira
 - Template ovog stanja biće wineDetails.html – koristimo istu formu kao što je ona za dodavanje vina (samo ćemo je u ovom slučaju popuniti podacima o vinu).
 - Kontroler datog templejta će biti WineDetailsController – isti kontroler kao onaj za dodavanje vina. Ideja je da će se u ovaj kontroler injektovati objekat wine. Ako je ovaj objekat prazan – znamo da dodajemo novo vino. Ako nije (ima postavljen _id property) – znamo da se radi o ažuriranju postojećeg vina.
 - U kontroler WineDetailsController injektovati property wine koji je objašnjen u gornjoj tački. Obratiti pažnju da je wine objekat ono što će se vratiti kao povratna vrednost get metode sa servera – objekat koji pored headera, status text itd. Ima property data u kome se nalazi ono što je sever vratio (u ovom slučaju pojedinačno vino). Obratiti pažnju da je forma wineDetails.html napravljena tako da prikazuje ono što se nalazi u \$scope.wine objektu. Dakle, u kontroleru postaviti \$scope.wine na injektovanu vrednost wine.data.
 - U stanje main.**addWine** dodati resolve – resolve će da ima ključ wine (objekat injektovan u WineDetailsController), a kao njegovu povratnu vrednost staviti funkciju koja vraća prazan wine objekat.
 - U stanje main.updateWine dodati resolve
 - resolve će da ima ključ wine (objekat injektovan u WineDetailsController)
 - Za povratna vrednost resolve funkcije staviti rezultat get funkcije WinesDataService servisa (ovo će biti promise objekat koji se mora razrešiti pre redirekcije – time se obezbeđujemo da smo dobili vino pre nego što se prebacimo na stranicu)
 - Da bi smo koristili WinesDataService u okviru resolve funkcije moramo ga u nju injektovati
 - Funkcija get u okviru WinesDataService kao parametar prima id vina koje se menja. Ovaj id se može preuzeti iz url-a pomoću \$stateParams servisa. Nemojte zaboraviti da u resolve funkciju injektujete \$stateParams servis kako bi ste ga koristili.
- c. U index.html dodati još jednu kolonu u tabelu. Namestiti da se u koloni nalazi link "update" koji prebacuje aplikaciju na url /wines/:id gde se na mestu :id nalazi id konkretnog vina (property _id). Obratiti pažnju da je stanje updateWine child stanje od main stanja – url na koji link treba da ukazuje je **#/main/wines/konkretan_id_vina**.
- d. Proverite da li se klikom na definisane url-ove prebacujemo na formu za ažuriranje vina gde su sva polja popunjena. Ako je ovo slučaj, ostala je još sama update akcija.
- e. WinesDataService proširiti funkcijom update koja šalje PUT zahtev na http://localhost:3000/api/wines/konkretan_id_vina. Funkcija kao parametar prima wine objekat sa ažuriranim vrednostima. Konkretan id vina izvaditi iz ovog objekta (_id property). Ovim se ažurira vino na serveru (videti slajd videti slajd 17 "API servera – snimanje podataka" sa AJAX predavanja). Poslati vino prosleđeno u funkciju kao parameter PUT zahteva. Kao rezultat metode vratiti povratnu vrednost izvršenog PUT zahteva – promise objekat. Nemojte zaboraviti da update funkciju postavite kao "javnu".
- f. U WineDetailsController u okviru funkcije submit (poziva se na klik Ok dugmeta) izvršiti proveru da li je \$scope.wine._id prazan – ako jeste, radi se o dodavanju novog vina i zvati funkciju WinesDataService .addWine; ako nije, radi se o ažuriranju postojećeg vina i zvati funkciju

WinesDataService.update. Ne zaboraviti da je povratna vrednost ove funkcije promise – kao i kod addWine na povratnoj vrednosti update metode pomoću then funkcije definisati successCallback i errorCallback. Ove funkcije mogu biti iste one korišćene kod addWine metode (takođe želimo redirekciju nakon uspešnog ažuriranja vina).

16. Omogućiti brisanje postojećeg vina: u tabeli, pored svakog vina u novoj koloni tabele dodati link “delete”. Ako korisnik klikne na ovaj link, vino se uklanja na serveru. Aplikacija se zadržava na prikazu tabele sa vinima, samo se ta tabela ažurira tako da se uklonjeno vino više ne prikazuje. Koraci:

- a. WinesDataService proširiti funkcijom deleteWine koja šalje DELETE zahtev na http://localhost:3000/api/wines/konkretan_id_vina. Funkcija kao parametar prima id vina koje je potrebno ukloniti. Kao rezultat metode vratiti povratnu vrednost izvršenog DELETE zahteva – promise objekat. Nemojte zaboraviti da deleteWine funkciju postavite kao “javnu”.
- b. U index.html dodata je još jedna kolona u tabeli. Kolona sadrži dugme (klase btn-link kako bi izgledalo kao link). Pomoću ng-click propisati funkciju deleteWine() koja će se pozivati kada se klikne na ovo dugme (u app.js je definisano da kontroler WinesController kontroliše ovaj deo prikaza – funkciju deleteWine() definisati na \$scope-u ovog kontrolera). Funkciji deleteWine() kao parametar proslediti id vina na koje je kliknuto.
- c. U okviru funkcije deleteWine() WinesController-a pozvati deleteWine funkciju WinesDataService-a. Ne zaboraviti da je povratna vrednost ove metode promise objekat – pomoću then metode definisati successCallback i errorCallback.
 - i. U errorCallback-u pomoću \$window servisa prikazati alert o grešci
 - ii. U successCallback-u iz odgovora servera izvući id obrisano vino (server kao povratnu vrednost delete funkcije vraća obrisano vino. Ovo vino će se nalaziti u data property-ju response-a). Proći kroz niz vina koja su prikazana u tabeli (property \$scope.wines) i iz njega ukloniti obrisano vino.

17. U tabeli sa vinima dodati sortiranje. Koraci:

- a. Pored imena svake kolone dodati dugme za sortiranje. Dugme ima glyph ikonu glyphicon-sort. Svakom dugmetu dodeliti ng-click direktivu koja poziva funkciju istu sort (pošto je u app.js definisano da kontroler WinesController kontroliše template wines.html u stanju main.wines, sort funkciju definisati na \$scope-u WinesController-a). Ova funkcija kao parametar treba da primi kriterijum pretrage prema kome je sortirano. Dakle, za dugme koje se nalazi pored “Name” kolone staviti da se poziva sort(“name”). Paziti da ovi stringovi koji se prosleđuju (“name”, “year”,...) odgovaraju property-jima vina! Ovo će biti važno zbog parametara koje prima server – ako želimo sortiranje prema property-ju “name” moramo mu poslati sort=“name”. Napomena: nekada je potrebno očistiti istoriju da bi se uradio refresh. Za sada kao funkcionalnost sort funkcije postaviti alert da se ispisuje koji je kriterijum sortiranja prosleđen. Proveriti da li ovo radi.
- b. U WinesController-u je do sada poziv WinesDataService.getAll() bio van funkcije. Sada ćemo ga staviti da bude u okviru nove funkcije getWines kako bi smo mogli ovaj poziv izvršavati na više različitih mesta bez dupliranja koda. Odmah po definiciji funkcije getWines pozvati je kako bi se izvršio ovaj poziv prilikom inicijalnog otvaranja stranice. Refreshovati stranicu i proveriti da li se i dalje prikazuje tabela sa vinima.

- c. Inicijalno ćemo u WinesController-u postaviti da nema sortiranja – definisati property `$scope.sortCriteria = ''`; Isto tako će inicijalni smer sortiranja biti ascending –definisati property `$scope.sortDirection = 'asc'` (prema serveru bilo šta sem desc znači rastući smer).
- d. U `getWines` želimo da kao parametre serveru uz `get` zahtev prosledimo `sort=trenutna_vrednost_promenljive_sortCriteria` i `sortDirection=trenutna_vrednost_promenljive_sortDirection` (videti slajd 14 “preuzimanje podataka, dodatni parametri” sa AJAX predavanja). Na ovaj način – šta god se trenutno nalazi u promenljivama `$scope.sortCriteria` i `$scope.sortDirection` će da propisuje kako će server da sortira podatke koje vraća:

```
var getParams = {
    sort: $scope.sortCriteria,
    sortDirection: $scope.sortDirection
}
```

Namestiti da metoda `getAll` prima kao parametar objekat `getParams` i da ga prosleđuje kao parametar `$http.get` metode koja se izvršava.

- e. U funkciji `$scope.sort` WinesController-a adekvatno podesiti vrednosti `$scope.sortCriteria` i `$scope.sortDirection`. Pozvati funkciju `$scope.getWines` kako bi se adekvatan zahtev prosledio serveru i tabela refresh-ovala (automatska sinhronizacija sa `$scope.wines` objektom koji se ovim pozivom menja).

18. Na stanju “wines” pored tabele sa vinima dodati i formu za pretragu vina po imenu. Koraci:

- a. U `wines.html` dodati formu za pretragu. Forma se sastoji od inputa i dugmeta. Za input je pomoću `ng-model` directive vezan `$scope.searchQuery` iz WinesController-a (koji kontrolira taj prikaz). Za dugme je putem `ng-click` directive vezana funkcija `$scope.search` iz WinesController-a.
- b. Napraviti funkciju `$scope.search` u WinesController-u – sve što funkcija treba da radi jeste poziv `getWines` kako bi se refresh-ovala tabela.
- c. Funkciju `getWines` WinesController-a promeniti tako da se među parametrima koji se šalju sa `get` zahtevom nalazi i filter prema imenu:

```
var filterParams = {};
filterParams['name'] = $scope.searchQuery; // šta god je korisnik uneo u input šalje se kao kriterijum pretrage po imenu

var getParams = {
    filter : filterParams,
    sort: $scope.sortCriteria,
    sortDirection: $scope.sortDirection
};
```

19. Proširiti pretragu vina tako da je moguće da se vrši i prema ostalim kriterijumima iz tabele (year, grapes,...). Koraci:

- a. U formu za pretragu vina dodati i jedan select. Imena opcija u select-u odgovaraju property-jima vina. Na select je dodata `ng-model` direktiva koja povezuje select sa `$scope.selectedItem`. Na ovaj način se u `$scope.selectedItem` nalazi property vina (odnosno ime selektovane opcije) prema kome treba da se vrši pretraga.
- b. Postaviti inicijalnu vrednost `$scope.selectedItems="name"`;
- c. Ispraviti funkciju `getWines` tako da parametri filtera odgovaraju selektovanoj opciji:

```
filterParams[$scope.selectedItem] = $scope.searchQuery;
```

20. U tabeli sa vinima dodati paginaciju. U jednom trenutku se u tabeli prikazuje maksimalno 5 vina.

- a. Definirati inicijalne vrednosti: `$scope.maxWines = 5;` i `$scope.page=1;` - nalazimo se na 1. Stranici pretrage i prikazujemo 5 vina u tabeli.

- b. Ispraviti funkciju `getWines` tako da se šalju i kriterijumi paginacije:

```
var getParams = {  
    filter : filterParams,  
    sort: $scope.sortCriteria,  
    sortDirection: $scope.sortDirection,  
    page: $scope.page,  
    pageSize: $scope.maxWines
```

```
};
```

Proveriti – u tabeli bi se sada inicijalno trebalo prikazivati samo prvih 5 vina.

- c. U `WinesController` dodaćemo funkciju `adjustPagination` koja prima broj vina kao parametar i na osnovu njega gradi komponentu paginacije. Napravićemo niz koji predstavlja ono što treba da bude ispisano na komponenti paginacije, npr. ukoliko izračunamo da treba da bude 3 stranice, niz će imati vrednosti `"<","1,2,3,">"`. Niz ćemo nazvati `paginationArray` i zakačiti ga za `$scope`. Pored niza u `$scope.noPages` sačuvaćemo broj stranica paginacije za kasniju upotrebu (funkcija `$scope.changePage` u tački h.).

- d. Funkciju `adjustPagination` bi trebalo pozvati kad god nam server vrati podatke o vinima, dakle u `onSuccessCallback` funkciji kada su nam sigurno prosleđeni podaci. Kao vrednost parametra funkciji `adjustPagination` proslediti broj vina koji je server vratio – `response.data.count`.

- e. U `wines.html` dodati komponentu za paginaciju. Iskoristićemo `ng-repeat` direktivu da na komponenti prikažemo stranice paginacije koje smo odredili kao vrednosti u `$scope.paginationArray`:

```
ul class="pagination" ng-repeat="i in paginationArray">
```

```
    <li><span>{{ i }}</span></li>
```

```
</ul>
```

Proveriti da li se na stranici prikazuje adekvatna komponenta paginacije.

- f. Stranici koja je aktivna treba dodeliti bootstrap klasu `active`. Ovo možemo postići putem `ng-class` direktive:

```
<ul class="pagination" ng-repeat="i in paginationArray">
```

```
    <li ng-class="{ 'active': i == page }"><span>{{ i }}</span></li>
```

```
</ul>
```

Kada se formira komponenta paginacije (stranicu po stranicu) kod one stranice koja je jednaka vrednosti koja se nalazi u `$scope.page` će primeniti klasa `active` na `li` tagu.

Proveriti da li je stranica 1 aktivna. Ako treba, obrisati istoriju u browser-u radi refresh-a.

- g. Ostaje još da se klikom na stranicu menja stranica paginacije. Ovo možemo postići putem `ng-click` direktive:

```
<ul class="pagination" ng-repeat="i in paginationArray">
```

```
    <li ng-class="{ 'active': i == page }"><span ng-click="changePage(i)">{{ i }}</span></li>
```

```
</ul>
```

Kada se klikne na stranicu, pozvaće se funkcija `$scope.changePage` kojoj će se proslediti `i` u zavisnosti od toga na koju je stranicu kliknuto (iz niza `paginationArray`, dakle `"<"` ili `1` ili `2`, ...)

- h. Dodati funkciju `$scope.changePage` u `WinesController`. Funkcija treba da promeni vrednost trenutne stranice (`$scope.page`) prema prosleđenoj vrednosti i da pozove funkciju `getWines` radi refresh-a tabele.