AngularJS

AngularJS je open-source JavaScript framework za pravljenje klijentskih aplikacija. Uz pomoć AngularJS ćemo izdraditi aplikaciju koju ćemo nazvati "GitHub Viewer".

- Kreirati folder za našu aplikaciju GitHubViewer
- Koristeći bilo koji tekstualni editor (npr. Sublime) u /GitHubViewer/ kreirati index.html:

Dva uslova za dodavanje AngularJS na web stranicu:
1. Dodati <script> tag koji sadrži putanju do angular.js
2. Dodati ng-app Angular *direktivu* kao atribut

```
<html ng-app>
    <head>
        <script type="text/javascript" src="angular.js"></script>
    </head>
    <body>
        <h1>GitHub viewer</h1>
        {{ 555 / 33 }}
    </body>
</html>
```

{{ … }} predstavlja binding izraz → Angular pokušava da evaluira izraz koji se nalazi unutar zagrada

- angular.js je jedini script koji treba dodati u aplikaciju da bi osnovna funkcionalnost Angulara bila dostupna – ne postoji ni jedna druga zavisnost.
- ng-app je AngularJS direktiva koja se pojavljuje samo jednom na stranici. Funkcionalnost Angulara je dostupna samo u okviru taga u kome je dodata ova direktiva. Obično se dodaje u sam html tag.

JavaScript Šabloni

1. **Funkcije kao moduli - "Revealing module pattern"**

Moduli – objekti sa podacima i metodama. Npr. Želimo worker objekat sa dve metode - job1 i job2. Napisaćemo funkciju koja nam kao rezultat vraća worker objekat:

script.js

```
var createWorker = function(){
  var workCount = 0; //private polje
  var task1 = function () { //metoda
    workCount += 1;
    console.log("Performing job 1 ... " + workCount);
  }
  var task2 = function () { //metoda
    workCount += 1;
    console.log("Performing job 2 ... " + workCount);
  }
  return{ //vrednosti vidljive van funkcije
    job1: task1,
    job2: task2
  }
}
var worker = createWorker();
worker.job2();
```

2. **Poziv neimenovane funkcije – uklanjanje globalnih varijabli**

Immediately invoked function expression (IIFE) – funkcija koja odmah poziva samu sebe. Nema globalnih varijabli, sve varijable su u local scope u okviru IFFE funkcije

U script.js ceo kod obuhvatiti sa:

```
(function(){
    // kod
})();
```

Ukucavamo ime korisnika koga želimo da prikažemo. Podaci o korisniku će biti skinuti sa GitHub servisa. Ukoliko ne izvršimo pretragu za 10 sekundi, aplikacija će automatski prikazati detalje „angular“ korisnika (njegovo ime, avatar i listu repozitorijuma koju je moguće sortirati na razne načine)

1. Stranica koja prikazuje podatke o korisniku (user.html)

Kontroleri kontrolišu sadržaj stranice – odgovorni su za to šta ćemo prikazati na stranici, za snimanje podataka koje korisnik unosi na stranicu,... Napravićemo *UserController* koji će biti odgovoran za prikaz detalja o korisniku.

user.html

```
<html ng-app="githubViewer">
    <head>
        <script type="text/javascript" src="angular.js"></script>
        <script src="app.js"></script>
        <script src="UserController.js"></script>
    </head>
    <body>
        <h1>GitHub viewer</h1>
        <div ng-controller="UserController">
            <h2>{{username}}</h2>
        </div>
    </body>
</html>
```

Angularu se putem ng-app naglasi u kom modulu traži kontrolere

Pomoću ng-controller direktive specificiramo ime kontrolera koji kontroliše dati deo stranice

Sadržaj modela $scope.username (videti UserController.js)

Kontrolere obično smeštamo u module. Ovo nam omogućava da izbegnemo global namespace.

Napravićemo novi script fajl **app.js** u kome ćemo definisati modul githubViewer:

app.js

```
(function(){
    var app = angular.module("githubViewer", []);
})();
```

angular je jedina globalna varijabla

[] je lista zavisnosti – modula koje naš modul uvozi

Napravićemo novi kontroler **UserController.js**. Kontroler je u suštini funkcija koja je dodeljena varijabli (u ovom slučaju UserController varijabli). Angular će pozivati ovu funkciju kada mu bude potrebna za kontrolisanje sadržaja web stranice.

UserController.js

```
(function() {
  var UserController = function($scope) {
      $scope.username = "Angular";
  };
  var app = angular.module("githubViewer");
  app.controller("UserController", UserController);
}());
```

$scope je veza sa modelom – objekti koji se dodele $scope su model

U varijablu app preuzimamo referencu na kreirani modul githubViewer (nema []!) i u modulu registrujemo UserController

Obratiti pažnju da je u user.html prvo naveden app.js, a tek onda UserController.js jer se u app.js definiše modul githubViewer koji se posle koristi u UserController. Pogledati link http://localhost:8080/user.html

U okviru user.html stranice želimo da prikažemo podatke dobijene sa GitHub web servisa.

Na slici je prikazan poziv GitHub API-ja:

Vraća se JSON objekat koji Angularov http servis automatski konvertuje u JavaSkript objekat (sa properties koji se vise na slici: login, id, avatar_url, gravatar_id...)

2. GitHub servis

Napravićemo svoj servis (singleton objekat specijalizovan za određeni zadatak). Svrha *github* servisa koga pravimo je da komunicira sa GitHub putem HTTP zahteva. Ovo je u Angularu omogućeno $http servisom: objekat sa metodama koje se mogu pozivati u cilju slanja http zahteva - GET, POST, PUT, DELETE. Servis se koristiti na sledeći način:

```
$scope.user = $http.get("/users/1783").then(function(response){
        $scope.user = response.data;
});
```

$http.get("/users/1783") ne vraća direktno podatke nego obećava da će vratiti podatke u budućnosti - *promise objekat*)

nad vraćenim promise objektom se može pozvati *then* metoda. Kao parametar *then* metoda prima funkciju koja će se pozvati kada podaci stignu. Ovu funkciju smo definisali inline - prima parametar *response* → objekat koji nam http zahtev vrati kao odgovor. U *data* propertiju odgovora se nalaze traženi podaci

github.js

```
(function(){
 var github = function($http){

  var getUser = function(username){
   return $http.get("https://api.github.com/users/" + username).then(function(response){
     return response.data;
    });
  };

  var getRepos = function(user){
   return $http.get(user.repos_url).then(function(response){
     return response.data;
    });
  };

  return {
   getUser : getUser,
   getRepos : getRepos,
  };
 };
 var module = angular.module("githubViewer");
 module.factory("github", github);
}());
```

Naš github servis zavisi od $http servisa

Ove metode takođe vraćaju promise objekat (Promise da je izvršen i HTTP zahtev i da se izvršila funkcija prosleđena u *then* metodu (funkcija koja iz raw podataka vraća objekat))

"Revealing module" šablon. Dve funkcije:

- funkcija koja prima username i vraća objekat koji reprezentuje traženog korisnika
- funkcija koja za datog korisnika (user objekat) vraća njegove repozitorijume

Registracija servisa uz pomoć factory objekta (nije jedini način)

U kontroleru UserController koji će da iskoristićemo kreirani github servis:

UserController.js

```
(function() {

 var UserController = function($scope, github) {
  $scope.username = "Angular";

  var onUserComplete = function(data) {
   $scope.user = data;
   github.getRepos($scope.user).then(onRepos, onError);
  };

  var onRepos = function(data) {
   $scope.repos = data;
  };
  var onError = function(reason) {
   $scope.error = "Could not fetch the data, reason: " + reason.statusText;
  };

  github.getUser($scope.username).then(onUserComplete, onError);
 };

 var app = angular.module("githubViewer");
 app.controller("UserController", UserController);
}());
```

Poziva se samo ukoliko je get metoda bila uspešna - samo tada će se setovati user objekat. U suprotnom, poziva se onError

Sada u user.html možemo iskoristiti dobijene objekte **user** i **repos** za prikaz stranice:

user.html

```
<html  ng-app="githubViewer">
    <head> … <script src="github.js"></script>…</head>
    <body>
        <h1>GitHub viewer</h1>
        <div ng-controller="UserController">
            <h2>{{user.name}}</h2>
            {{error}}
            <img ng-src="{{user.avatar_url}}" title="{{user.name}}" height="40" width="40"/>
            <table>
                <thead>
                    <tr><th>Name</th><th>Stars</th><th>Language</th></tr>
                </thead>
                <tbody>
                    <tr ng-repeat="repo in repos">
                        <td>{{repo.name}}</a> </td>
                        <td>{{repo.stargazers_count}}</td>
                        <td>{{repo.language}}</td>
                    </tr>
                </tbody>
            </table>
        </div>
    </body>
</html>
```

ng-src a ne src!

Binding izrazima je moguće postaviti i vrednosti atributa

Vidljivo samo ako je došlo do greške prilikom get poziva ($scope.error) se setuje u onError() metodi UserController-a

Pogledati link http://localhost:8080/user.html. Obratiti pažnju da u okviru <img> taga umesto src atributa stoji ng-src atribut. Ukoliko se stavi src atribut, browser će odmah pokušati da iskoristi string "{{…}}" kao URL za skidanje slike - u konzoli se vidi greška 404. Sa ng-src će se tek nakon što je Angular evaluirao binding izraz setovati pravi src atribut za img tag (ng-src je *direktiva* za setovanje src atributa img taga kada je u pitanju binding izraz).

Dodaćemo search box u kome korisnik može da specificira kog korisnika želi da prikaže (kasnije ćemo ovaj prikaz izdvojiti na posebnu stranu).

user.html

```
…
<div ng-controller="UserController">
    {{username}}
    <form name="searchUser" ng-submit="search(username)">
     <input type="search" required="" placeholder="Username to find" ng-model="username" />
     <input type="submit" value="Search" />
    </form>
<h2>{{user.name}}</h2>
```

Pomoću ng-model direktive vrši se prenos podataka iz view-a u model. Vrednost direktive je property koji će se dodati na $scope

U ng-subnit direktivu stavljamo funkciju koja će se pozvati kada korisnik klikne na Search dugme

obezbeđuje da je unos obavezan

Primetiti kako se {{username}} menja sa svakim otkucanim karakterom. Inicijalizacija vrednosti {{username}} je u UserController.js ($scope.username = "Angular";). Međutim, i da nije postojalo username property na scope-u ne bi bilo problema (angular bi kreirao ovaj property za nas).

UserController.js - Poziv github.getUser staviti u okvir funkcije:

```
$scope.search = function(username) {
    github.getUser($scope.username).then(onUserComplete, onError);
};
```

3. Direktive

Problem: ružan prikaz (prazna tabela i image) dok se ne uradi search. Probajte da unesete neki drugi username (npr. robconery). Šta se dešava ako pokušamo da unesemo nepostojeći username?

Ovi problem se može rešiti upotrebom ng-show/ng-hide direktiva - u okviru user.html stranice u <table> i <img> tagove dodati atribut ng-show=”user”. Ova direktiva radi tako što pokuša da evaluira izraz i ukoliko je izraz „true“ (u ovom slučaju ako postoji user objekat) prikazaće tag na koji se odnosi. Isti efekat se može postići sa ng-hide=”!user”. Ako želimo da uklonimo prikaz prethodno prikazanog korisnika nakon što smo vršili pretragu za nepostojećim korisnikom, možemo u oviru onError() funkcije dodati $scope.user = null;

Sledeće ćemo izdvojiti deo stranice sa pretragom na posebnu stranicu (main.html) tako da nam u okviru user.html ostanu samo delovi vezani za prikaz samog korisnika (ime, slika i tabela korisnikovih repozitorijuma). Uz pomoć ng-include direktive moguće je prikazati html iz drugog izvora (zgodno za razbijanje kompleksnih stranica na razulmljivije delove ili ukoliko imamo deo koji se može reusovati). Napraviti stranicu main.html:

main.html

```
<html ng-app="githubViewer">
    <head>
        <script type="text/javascript" src="angular.js"></script>
        <script src="app.js"></script>
        <script src="github.js"></script>
        <script src="UserController.js"></script>
    </head>
    <body ng-controller="UserController">
        <h1>GitHub viewer</h1>
        {{username}} {{ error }}
        <form name="searchUser" ng-submit="search(username)">
            <input type="search" required="" placeholder="Username to find" ng-model="username" />
          <input type="submit" value="Search" />
        </form>
        <div ng-include="'user.html'" ng-show="user"></div>
    </body>
</html>
```

lokacija dodatnog markupa. Da Angular ne bi pokušao da evaluira izraz dat u ng-include direktivi stavljeni su jednostruki navodnici

U user.html ostaviti samo tagove vezane za prikaz korisnika i ukloniti ng-show direktive (ng-show je premešten u main.html):

user.html

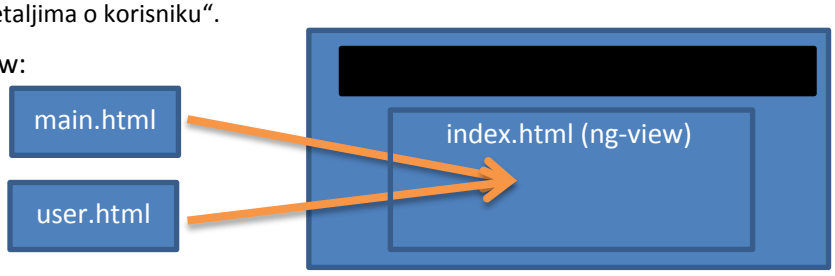
```
<div>
    <h2>{{user.name}}</h2>
    <h2>{{error}}</h2>
    <img ng-src="{{user.avatar_url}}" title="{{user.name}}" height="40" width="40"/>
    <table>
        <thead>
            <tr><th>Name</th><th>Stars</th><th>Language</th></tr>
        </thead>
        <tbody>
            <tr ng-repeat="repo in repos">
                <td>{{repo.name}}</a> </td>
                <td>{{repo.stargazers_count |number}}</td>
                 <td>{{repo.language}}</td>
            </tr>
        </tbody>
    </table>
</div>
```

Pogledati stranicu http://localhost:8080/main.html

4. Rutiranje

Rutiranje omogućava: navigaciju između različitih prikaza, razmenu parametara među kontrolerima koji upravljaju prikazima i back/forward u browseru. Za rutiranje se koriste pravila, npr. „ako je URL /user/{username} prikazati prikaz sa detaljima o korisniku“.

Layout view:



app.js

```
(function(){
 var app = angular.module("githubViewer", ["ngRoute"]);

 app.config(function($routeProvider){
    $routeProvider
        .when("/main", {
            templateUrl: "main.html",
            controller: "MainController"
        })
        .when("/user/:username", {
            templateUrl: "user.html",
            controller: "UserController"
        })
        .otherwise({redirectTo:"/main"})
 });
}());
```

Dodat je dependency od ngRoute. Za konfiguraciju pravila koristi se $routeProvider

kako URL izgleda (biće /#/main)

Šta URL radi: HTML koji Angular treba da prikaže za dati routh i kontroler koji se koristi za menadžment ovog template-a

: → kontroler ovo vidi kao parametar - podatak koji treba da ekstrahuje iz URL-a i prosledi drugim komponentama

Ukoliko putanja nije prepoznata, vrati se na main.html

Napraviti index.html:

index.html

```
<html ng-app="githubViewer">
    <head>
        <script type="text/javascript" src="angular.js"></script>
        <script type="text/javascript" src="angular-route.js"></script>
        <script src="app.js"></script>
        <script src="github.js"></script>
        <script src="UserController.js"></script>
        <script src="MainController.js"></script>
    </head>
    <body>
        <h1>Github Viewer</h1>
        <div ng-view></div> <!-- Here the routing engine loads our different templates -->
    </body>
```

Izmešteno iz main.html, u main.html ostaviti samo form element i obuhvatiti ga tagom. Ukloniti i ng-include za user.html

Iz main.html izbaciti sve sem <form> taga za pretragu korisnika (izbaciti i ng-include, user.html će se prikazivati u skladu sa pravilima rutiranja). Obuhvatiti sve u jedan tag.

Potrebno je da napravimo i MainController.js koji će da kontroliše main.html pregled. U njemu ćemo vršiti pretragu korisnika (i kasnije i odbrojavanje do pretrage):

MainController.js

Servis za editovanje lokacije u address baru browsera

```
(function() {
  var MainController = function($scope, $location) {
    $scope.search = function(username) {
      $location.path("/user/"+username);
    };
    $scope.username = "Angular";
  };
  var app = angular.module("githubViewer");
  app.controller("MainController", MainController);
}());
```

Izmešteno iz UserController

UserController.js

```
(function() {
  var UserController = function($scope, $routeParams, github) {
      ...
      $scope.username = $routeParams.username;
      github.getUser($scope.username).then(onUserComplete, onError);
  };
  ...
}());
```

U user.html dodati link za povratak na main.html: Back to search

Otići na link http://localhost:8080/

5. Odbrojavanje do pretrage

MainController.js

```
(function() {
 var MainController = function($scope, $location, $interval) {

  $scope.search = function(username) {
     if (countdownInterval) {
          $interval.cancel(countdownInterval);
          $scope.countdown = null;
     }
     $location.path("/user/"+username);
  };

  var decrementCountdown = function() {
   $scope.countdown -= 1;
   if ($scope.countdown < 1) {
    $scope.search($scope.username);
   }
  };

  var countdownInterval = null;
  var startCountdown = function() {
   countdownInterval = $interval(decrementCountdown, 1000, $scope.countdown);
  };

  $scope.username = "Angular";
  $scope.countdown = 10;
  startCountdown();
 };
 var app = angular.module("githubViewer");
 app.controller("MainController", MainController);
}());
```

U JavaScript postoje fukncije:
- setTimeout - pozovi funkciju nakon što istekne vreme (u Angularu servis $timeout)
- setInterval - ponavlja poziv funkcije svaki put nakon što istekne zadat period (u Angularu servis $interval)

$interval vraća se objekat: čuvamo ga i koristimo kada korisnik klikne na search() da prekinemo poziv interval. Postavljanje $scope.countdown na null će ukinuti prikaz broja sekundi pre autiomatske pretrage.

Koliko puta da se pozove funkcija

Funkcija koja se poziva svake sekunde

Inicijalizacija broja sekundi do pretrage

U main.html dodati {{countdown}} da se vidi koliko je sekundi ostalo do pretrage.

6. Sortiranje prikaza repozitorijuma

Za sortiranje ćemo korristiti orderBy filter. Opšti oblik filtera je: expression | filterName:parameter. Npr. ako želimo repozitorijume da izlistamo prema stargazers_count parametru u opadajućem redosledu (user.html):

```
<tr ng-repeat="repo in repos" | orderBy:'-stargazers_count'">
```

Obratiti pažnju na jednostruke navodnike - stavljaju se zbog toga da bi Angular stargazers_count razumeo kao string, a ne kao izraz koji treba da pokuša da evaluira. Da li će redosled biti u opadajućem ili rastućem redosledu kontrolišemo dodajući + ili -. U user.html dodaćemo drop-down listu iz koje korisnik može da odabere način sortiranja:

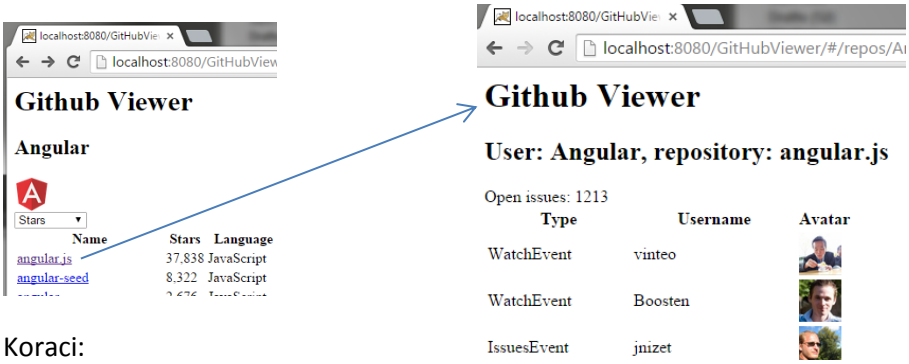
user.html

```
...
<select ng-model=”repoSortOrder”>
    <option value=“+name“>Name</option>
    <option value=“-stargazers_count“>Stars</option>
    <option value=“+language“>Language</option>
</select>
...
<tr ng-repeat="repo in repos | orderBy:repoSortOrder”>
```

Selektovana vrednost će biti dodata u model u parametar repoSortOrder uz pomoć ng-model direktive. U UserController postaviti inicijalnu vrednost:
$scope.repoSortOrder=“-stargazers_count“;

Zadatak 1

Na user.htlm stranici (prikaz detalja o korisniku) omogućiti selekciju jednog od izlistanih repozitorijuma - imena repozitorijuma u tabeli predstaviti kao linkove. Odabir jednog od datih linkova bi korisnika trebalo da prebaci na stranicu koja prikazuje detalje o repozitorijumu: broj otvorenih pitanja i događaje vezane za repozitorijum (videti link https://api.github.com/repos/angular/angular.js → detalji koji su nam potrebni su open_issues i events_url - link na kome se nalaze događaji vezani za repozitorijum).



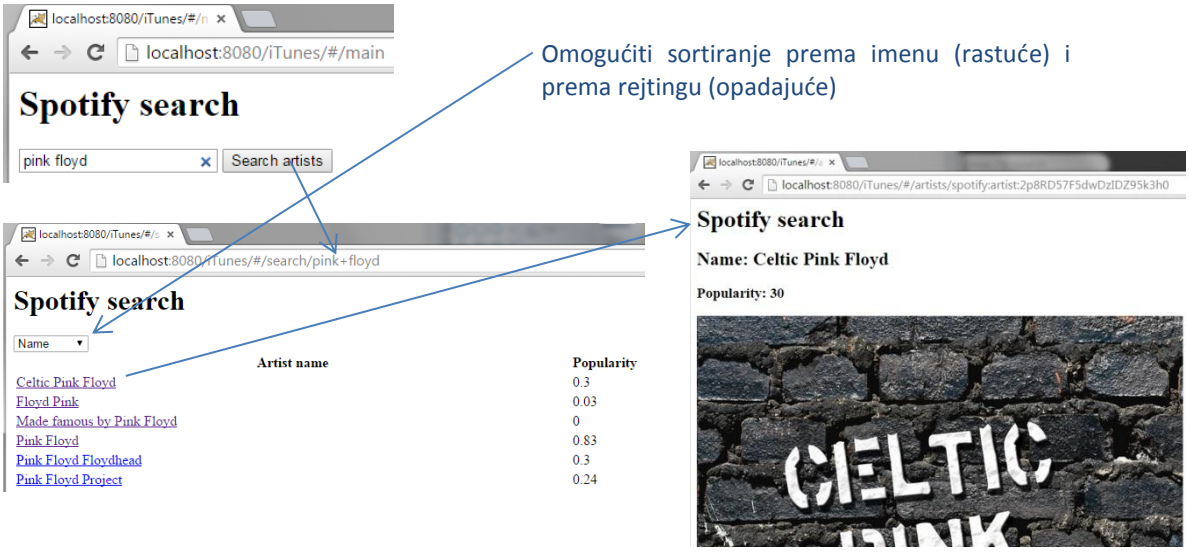
Koraci:

1. Napraviti repo.html (stranicu koja će prikazivati detalje o selektovanom repozitorijumu) i RepoController koji će je kontrolisati.
    a. U index.html dodati da se koristi skript RepoController.js
    b. U RepoController.html preuzeti ime korisnika i repozitorijuma iz linka
    c. U github service dodati funkciju getRepo koja prima ime korisnika i repozitorijuma i preuzima traženi repozitorijum sa linka (https://api.github.com/repos/<username>/<reponame>. Ne zaboravite da novu funkciju podesite da bude vidljiva (dodati je u return). Iskoristite ovu funkciju u RepoController za preuzimanje objekta koji predstavlja repozitorijum. Iskoristiti ovu funkciju u RepoController za preuzimanje objekta koji predstavlja repozitorijum.
    d. Podesiti repo.html da prikazuje ime korisnika, ime repozitorijuma i broj otvorenih pitanja.
2. Dodati novo routing pravilo koje će nas prebaciti na stranicu repo.html kada korisnik klikne na link.

3. U user.html u tabeli namestiti da su imena repozitorijuma linkovi. Ukoliko je potrebno da se u putanji linka koriste binding izrazi koje Angular treba da evaluira, može se koristiti ng-href atrribut umesto href atributa <a> taga.
4. Sličan postupak ponoviti za prikaz detalja o događajima (primer linka: https://api.github.com/repos/angular/angular.js/events → detalji koji su nam potrebni su type (tip događaja), actor.login (username korisnika) i actor.avatar_url (link ka avataru korisnika)).

Zadatak 2

Napraviti aplikaciju za pretragu i prikaz podataka o muzičarima sa spotify sajta. Izlged aplikacije:



Detaljne informacije za upotrebu spotify api-ja su dostupne na https://developer.spotify.com/technologies/metadata-api/search/

Primer pretrage muzičara: `http://ws.spotify.com/search/1/artist?q=pink+floyd` - razmaci u termu pretrage su zamenjeni + (može se uraditi pomoću .replace(" ", "+")).

Primer pretrage muzičara sa zadatim id (id se može izvući iz href atributa artist elementa - videti prethodni link): https://api.spotify.com/v1/artists/2p8RD57F5dwDzIDZ95k3h0 (u href atributu se nalaze linkovi u obliku spotify:artist:2p8RD57F5dwDzIDZ95k3h0, sam id se može izvaditi sa .split(':')[2]).