VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Bakalářská práce - Hra Port Royal a moderní vývoj softwaru

Bachelor thesis - Game Port Royal and modern software development

2017 Jan Čech

Zadání bakalářské práce

Jiří Dvorský

Ukázka sazby diplomové nebo bakalářské práce

Diploma Thesis Typesetting Demo

+++

Podpis vedoucího katedry

+++

Podpis děkana fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. dubna 2016 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 20. dubna 2017 .....

Rád bych na tomto místě poděkoval Ing. Davidovi Ježkovi, Ph.D., za pomoc a vedení u této práce.

Abstrakt

reklama - kratsi uvod - par vet

Klíčová slova: typografie, LaTeX, diplomová práce

Abstract

This is English abstract. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Fusce tellus odio, dapibus id fermentum quis, suscipit id erat. Aenean placerat. Vivamus ac leo pretium faucibus. Duis risus. Fusce consectetuer risus a nunc. Duis ante orci, molestie vitae vehicula venenatis, tincidunt ac pede. Aliquam erat volutpat. Donec vitae arcu. Nullam lectus justo, vulputate eget mollis sed, tempor sed magna. Curabitur ligula sapien, pulvinar a vestibulum quis, facilisis vel sapien. Vestibulum fermentum tortor id mi. Etiam bibendum elit eget erat. Pellentesque pretium lectus id turpis. Nulla quis diam.

Key Words: typography, LaTeX, master thesis

Obsah

Seznam použitých zkratek a symbolů

SPA	–	Single Page Application
REST	–	REpresentational State Transfer
WWW	–	World Wide Web
HTML	–	Hyper Text Markup Language
W3C	–	World Wide Web Consortium
XML	–	Extensible Markup Language
JSON	–	JavaScript Object Notation
J2EE	–	Java 2 Platform, Enterprise Edition
URI	–	Uniform Resource Identifier
URL	–	Uniform Resource Locator
TDD	–	Test-driven development

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Jako vše v oblasti informačních technologii i oblasti vývoje webových stránek se objevuje spousta nových trendů. Tyto trendy se týkají jak nových technologii, jenž usnadňují vývoj. Tak také návrhu celé architektury aplikace, která vývoj zpřehledňuje a usnadňuje rozšířitelnost aplikace do budoucna. Také se do popředí dostává testování software, jenž byl kdysi okrajovou záležitostí. V dnešní době se však testování software, stalo standardní součástí vývoje aplikace. Cíle této práce bylo všechny tyto trendy zachytit. Práce by se dala shrnout do následujících bodu. Všechny body níže byly splněny.

1. Nastudovat moderní webové technologie Spring, AngulaJS a REST služby
2. Nastudovat principy testování a vybrat SW pro testování výše zmíněných technologii.
3. Navrhnout a implementovat hru Port Royal.
4. Zprovoznit testovací technologie a napsat v nich testy.

1.1 Stručný obsah jednotlivých kapitol

bude navazovat na horni cast

2 Technologie

2.1 RESTful webove služby

Webové služby jsou typ architektury pro komunikaci na principu server - klient přes World Wide Web (WWW) HyperText Transfer Protocol (HTTP). Tak jak jej popisuje World Wide Web Consortium (W3C). [6]

REpresentational **S**tate **T**ransfer neboli REST je webová služba, jenž by měla plnit následující podmínky díky nimž je rychlá a jednoduchá:

- Dotazy by měly být sebe popisující - Tímto se myslí, že dotaz obsahuje všechny informace k jeho zpracování. Tyto informace lze zjistit například z URI či hlavičky dotazu.
- Bezstavovost - díky tomu, že dotazy nemají stav jsou mezi sebou nezávislé. Pěkný test této podmínky je například restart serveru, po němž by se na interakci mezi serverem a klientem nemělo nic změnit.
- Uložitelnost do krátkodobé paměti - Jelikož jsou dotazy bezstavové a obsahují všechny informace k jejich zpracování, jsou výsledky dotazů téměř vždy stejné. Tudíž je zde velký prostor pro ukládaní dotazu do krátkodobé paměti. Tímto se sníží objem přenesených dat a urychlí komunikace mezi serverem a klientem, při opakovaném dotazu.
- Interface - rest služby mají předepsaný interface, jenž je popsán v následující podkapitole.

RESTy nemají upřesněný formát dotazu. Tudíž RESTový dotaz může být ve formátu XML, JSON, ale také například i ve formátu PDF. [7] [8]

2.1.1 Vzorové REST rozhraní

RESTové rozhraní je založeno na typech HTTP dotazů a jejich URI. Použít se standardní HTTP metody. POST pro vytvářená záznamu, GET pro získání záznamu, PUT pro úpravu záznamů a DELETE pro mazání. Dají se použít i další metody, tyto jsou ovšem nejpoužívanější. Následující tabulka ukazuje vzorové rozhraní, podobné tomu které jsem použil. [9]

Tabulka 1: Vzorové REST rozhraní

HTTP metoda	**URI**	**Operace**
GET	/administrace/uzivatele	Vrátí list všech uživatelů
GET	/administrace/uzivatele/1	Vrátí data uživatele s ID 1
POST	/administrace/uzivatele	Vloží nového uživatele
PUT	/administrace/uzivatele/1	Upraví údaje uživatele s ID 1
DELETE	/administrace/uzivatele/1	Smaže uživatele s ID 1

2.2 AngularJs

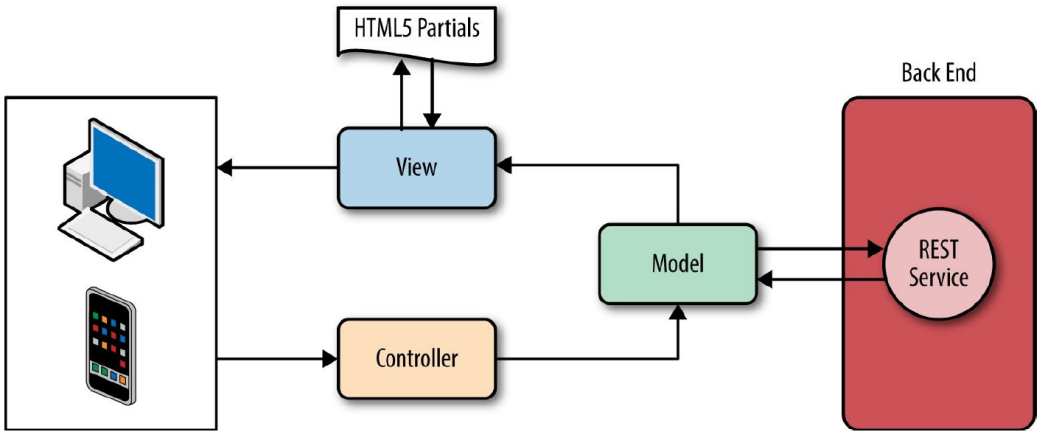
AngularJS je javascriptový framework založený na model-view-controller architektůře, určený pro vývoj single page aplikací. AngularJS rozšiřuje HTML direktivy, používá two way data binding, dependency injection a u něj vysoká znovupoužitelnost komponent. Celkově tak usnadňuje a urychluje vývoj webové částni aplikace. Pro přístup na backend AngularJS využívá RESTové webové služby. [2]

2.2.1 Jednostránková aplikace

Single page aplikace (SPA) je taková aplikace, která běží uvnitř prohlížeče na straně uživaetle a nevyžaduje znovunačtení stránky během používáni. Funguje to tak, že při vstupu na stránku si prohlížeč stáhne celou aplikaci. Následně se nestahují HTTP šablony, ale pouze data, která zpracovává aplikace. Práce s takovouto stránkou je pak mnohem rychlejší, jelikož se přenáší mnohen nižší objem informaci. [3]

2.2.2 Model–View–Controller architektura

Model–View–Controller architektura dělí aplikaci na 3 části. Model, pohled a kontrolér. Uživateli je zobrazen pohled, pokud provede nějakou akci spustí funkci v kontroléru. Ten aktualizuje model, buď to svými daty, nebo se doptá se serverové strany přes REST. Aktualizovaný model se následně zobrazí v pohledu. Krásně je to zobrazeno v následujícím obrázku.



Obrázek 1: MVC architektura v AngularJS

2.2.3 Two way data binding

Jedna z výhod frameworku AngularJS kterou programátoři bezpochyby ocení je Two way data binding. Jedná se o automatickou synchoronizaci mezi modelem a pohledem. Což umožňuje rychnou a pro prográmátora celkem nenáročnou synchronizaci mezi modelem v javascriptu a

pohledem v HTML šabloně. Což se děje v například pokud dorazí nová data z backendu nebo pokud uživatel provede nějakou akci. [5]

2.2.4 Dependency injection v AngularJS

Dependency injection je návrhový vzor, ve kterém jsou závislosti definovány jako součást konfigurace aplikace. Díky této vlastnosti nemusí programátor ručně vytvářet závislosti. AngularJS načte a inicializuje všechny závislosti při spuštění aplikace a následně se stará o celý jejich životní cyklus. Programátor si tak jen napíše, kterou závislost chce použít a pak už volá její metody.

Díky snadnosti vložení závislostí Dependency injection také usnadňuje testování, jelikož se snadno dají místo skutečných závislostí použít podvržené objekty neboli mocky.

Dependency injection je také použit ve frameworku Spring, v němž je jednou z klíčových součástí. Spring bude zmíněn v dalších kapitolách. [10]

2.2.5 Ukázka základního kontroléru

languagelanguage

```
1  var portRoyalApp = angular.module('portRoyalApp', [
2      'ngRoute',
3      'rzModule',
4      'portRoyalApp.gameCtrl'
5  ]);
6
7  var gameCtrl = angular.module('portRoyalApp.gameCtrl', ['ngRoute']);
8
9  gameCtrl
10     .controller('gameCtrl', function ($scope, $http) {
11         // http get a prirazeni do scope
12     })
13     .config(['$routeProvider', function ($routeProvider) {
14         $routeProvider.when('/game', {
15             templateUrl: 'components/templates/game.html',
16             controller: 'gameCtrl'
17         });
18     }]);
```

Výpis 1: Ukázka angulaří aplikace

Na řádku jedna se vytváří hlavní angulaří modul. Na řádcích 2 - 4 se do něj přidávají další moduly

i HTTP kod ?

2.3 Další technologie

2.3.1 Node.js

Většinu problémů které řeší programátor v javascriptu řešil už někdo před ním. Z tohoto důvodu by nebylo rozumné, aby programátor psal všechnu funkcionalitu sám. Je rozumnější podívat se zda už tato funkcionalita někde neexistuje. Proto vznikl Node.js. Node.js umožňuje programátorovi najít si funkcionalitu, a přidat ji do konfiguračního souboru. Po následném spuštění Node.js instalace se stáhnou závislosti, jenž programátor požaduje. Ty pak stačí pouze přidat pomocí dependency injection.

Node.js také slouží jako task runner což znamená, že dokáže spouštět různé testovací framoworky, či fungovat jako server.

2.3.2 Spring

Spring je populární open-source framework pro vývoj J2EE aplikací. Jeho první verze vyšla v říjnu 2003 od té doby nabyl na popularitě.

Spring je označován jako kontejner jelikož je modulární. V základu tudíž neumí téměř nic, až s přidáváním modulů získává další funkcionalitu. Ačkoli přidávání dalších modulů komplikuje přípravu prostředí je to výhoda, jelikož následně na serveru jede pouze to co je opravdu potřeba. Závislosti se přidávají přes Maven což je obdoba Node.js pro Javovské aplikace. V aplikaci Port Royal jsou použity například tyto moduly:

- Hibernate - toto je implementace Java Pesistance Api, která slouží k namapování javovských entit na relační databázi. Následně pak zajišťuje komunikaci mezi databází a Springem.
- Spring Security - Spring Security poskytuje autentizaci a autorizaci uživatele
- Tomcat plugin - Tomcat je open source webový server sloužící k nasazení aplikace. Implementuje většinu specifikace Java EE API. Tomcat je v této práci použit jako mavenovský plugin. Díky toho není potřeba instalovat Tomcat server. Stačí pouze vytvořit spustitelný war soubor a následně pustit tento plugin mavenovským příkazem
- AspectJ - Tento modul slouží k vytváření aspektu. Aspekty jsou metody, které se volají nad větší skupinou různých tříť u nichž je potřeba stejná funkcionalita. Například autentizaci nebo logování příchozího dotazu, které se nemusí psát pro všechny třídy zvlášť.
- spring websocket - Ve hře Port Royal, kterou hraje více hráčů najednou je potřeba informovat hráče o akcích spoluhráčku. Každý hráč by měl tuto informaci dostat právě jednou a to co nejdříve. Toto zařídí websockety, které mají dvě URL. Jednu na niž se přihlásí všichni hráči ve hře. A druhou na kterou hráči zasílají své akce. Pokud některý zašle svou akci na druhou URL budou o této akci informování všichni hráči kteří poslouchají na první.

2.3.3 GIT

git

3 Typy testování

3.1 Smysl testování software

Testování dnes hraje při vývoji software důležitou roli. Organizace i vývojáři pochopili výhody testování hlavně u velkých aplikací, které se vyvíjejí a pak udržují mnoho let. U takovýchto aplikací mnohdy nelze s jistotou vědět kde a jak se změny v kódu projeví. Pokud je ovšem určitá funkcionalita pokryta testy může se říci, že je tato funkcionalita splněna. A nejen to pokud se bude kód upravovat ví se, že testy pohlídají aby byla původní funkcionalita zachována. Takto se předejde vytvoření chyb.

Ve výsledku tedy testy nejsou něco co by programátora zdržovalo. Právě naopak testy zabraňují výskytu chyb a ohlídají, že má aplikace požadovanou funkcionalitu. Takto testování šetří čas programátorů a peníze organizacím.

3.2 Manuální oproti automatickým testům

Testy se dají rozdělit na manuální a automatické. Manuální testování je když se tester vžije do role uživatele a ručně otestuje danou funkcionalitu oproti specifikaci. Tento typ testování je rychlý a jednoduchý tudíž tester ani nemusí mít větší technické znalosti. Tyto testy jsou vhodné pro menší aplikace, které se nebudou měnit.

Problém nastává u větších a složitějších aplikací. U nichž nelze snadno a rychle zkontrolovat všechnu funkcionalitu manuálně. Této funkcionality začne být hodně. Zde jsou již potřeba automatické testy. Tyto testy sice trvá déle napsat, ovšem při několikanásobném opakování testu, jenž je potřeba po úpravách v aplikaci. Jsou již automatické testy časově výhodnější.

Tato práce se zabývá automatickými testy.

3.3 Test-driven development

Klasický vývojový cyklus byl takový, že programátor dostal zadání to nastudoval. Následně napsal kód, trochu to lidově řečeno proklikal, pokud se mu vše zdálo funkční kód odevzdal. Pak bylo dále na testerovi, aby kód pořádně otestoval.

V dnešní době se však do popředí dostává Test-driven development neboli zkráceně už jen TDD. Tento přístup k vývoji předpokládá krátký vývojový cyklus, neboli psaní software po menších částech. Vývojář praktikující TDD po obdržení a nastudování zadání nezačne psát implementaci zadání, ale nejprve vytvoří testy na požadovanou funkcionalitu. Tyto testy samozřejmě bez implementace neprojdou. Vytvoření požadované funkcionality je až další krok. Až v momentě kdy všechny testy projdou by měla být požadovaná funkcionalita vytvořena.

3.4 Úrovně testů

Testy se také dají rozdělit na více úrovní podle toho jak velký objem kódu testují. Programátor většinou při vývoji pouští pouze své testy. Jelikož u větších systémů proběhnutí všech testů zabere několik minut. Všechny testy se tudíž většinou pouštějí až po dokončení práce pro potvrzení, že původní funkcionalita nebyla narušena.

U všech testů platí, že by se navzájem neměly ovlivňovat.

3.4.1 Jednotkové testy

Jednotkové testy jsou zaměřené na otestování funkcionality základních jednotek kódu. Většinou jedné funkce jedné třídy. Výhodou je, že tyto testy jsou rychlé a snadno znovu spustitelné. Po napsání by měly zajistit, že daná část kódu plní svou funkcionalitu. Jedna funkcionalita může mít více testů pro různé vstupní parametry.

Při unit testech v AngularuJS a Springu se využívá Dependency injection. Komponenty, které využívá testovaný kód se nahrazují podvrženými komponentami. Díky čemuž není potřeba žádných závislostí a můžeme si nasimulovat různé situace.

3.4.2 Integrační testy

Integrační testy propojují více komponent. Testují zda jsou správně propojené a plní očekávanou funkcionalitu. Nevýhodou těchto testů je, složitější odhalení chyby pokud test neprojde. Jelikož je třeba debugovat více komponent.

3.4.3 End-To-End testy

End-To-End testy simulují chování koncového uživatele v aplikaci. Tyto testy kontrolují například zda se na stránce objeví určitý prvek, či zda se po kliknutí na něj provede očekávaná akce.

3.5 Testovací technologie

3.5.1 Jasmine

Jasmine open source testovaní je framework pro Javascript. Má za cíl být nezávislý na ostatních frameworcích či vývojářských prostředích. Snaží se také o snadno čitelnou syntaxi. Využívá se pro unit testy nejen v prostředí angularJS. Před každým testem se nejprve injectuje testovaný modul, ten má spoustu závislostí. Pro tyto závislosti se vytvoří podvržené moduly. Takto se zajistí jednak, že se opravdu testuje pouze daný modul, ale také se takto dají nasimulovat různé vstupní data. Například určením hodnot, které budou vracet http odpovědi. Jasmine také umožňuje vytváření takzvaných spy objektů, které kontrolují zda byla metoda zavolána, případně s jakýma hodnotami.

3.5.2 Protractor

Protractor je end-to-end testovací framework pro AngularJS. Protractor spouští testy oproti skutečné aplikaci, která je již spuštěná na serveru. Spustí si prohlížeč, v němž simuluje chování skutečného uživatele. Protractor se na stránce naviguje pomocí http šablony, zde si nalezne elementy podle id, css stylu či angulářího modelu. S těmito elementy pak provádí akce nebo ověření jejich hodnoty oproti očekávané hodnotě. Protractor také umí počkat na načtené stránky tudíž programátor nemusí dávat pevnou dobu čekáni. Což by jednak prodlužovalo běh testu, ale také by při spoždění odpovědi serveru mohlo způsobit zahlášení selhání testu, jenž by ovšem bez mimořádného zpoždění prošel. [11]

3.5.3 Karma

Karma je spouštěč testu v Node.js. Využívá ho Jasmine i Protractor. Spouští prohlížeč, v němž běží testy.

3.5.4 Jersey

Jersey je testovací framework vytvořen pro ověření správnosti serverových komponent. Testuje RESTové služby, funguje to tak že si Jersey rozjede svůj vlastní aplikační server s testovanými REST funkcemi. V této bakalářské práce Jetty ovšem dá se použít i jiný server. Následně se volají tyto resty, odpovědi serveru se nakonec porovnají s očekávanými. Pokud vývojář dostane například zadáni aby vytvořil REST, jenž vrací nějakou hodnotu. Snadno vytvoří test, kde pouze zavolá REST podle zadání a porovná navrácenou hodnotu. U těchto testů je pouze zdlouhavější nastavení prostředí následné testy se píší rychle a snadno.

4 Obsah a funkcionalita aplikace

Hra obsahuje ...

4.1 zkrácená pravidla

Hra s hraje tak

4.2 Obsah a implementace stránky na vytváření hry

obrazek ?

Na stránce vytváření hry je použit posuvník k určení maximálního počtu hráčů. Při změně hodnoty na posuvníku se také změní hodnota angulářího modelu .. treba vysvetlit scope a http dirty check ?

4.3 Obsah a implementace stránky s hrou

obrazek ?

co je tam použito ... sockety

4.4 Obsah a implementace stránky s administraci

obrazek ?

Tato stránka je dostupná pouze uživatelům s rolí administrátora. Stránka obsahuje seznam uživatelů seřazen podle loginu. Je zde použit stránkování, kdy se na jedné stránce zobrazí pouze 10 uživatelů. Stránkování je udělané tak, že ...

5 Analýza použitých technologii

5.1 Spring základ aplikace

Hra je vyvinuta ve Springu.

struktura složek s popisem ?

5.2 pom Maven

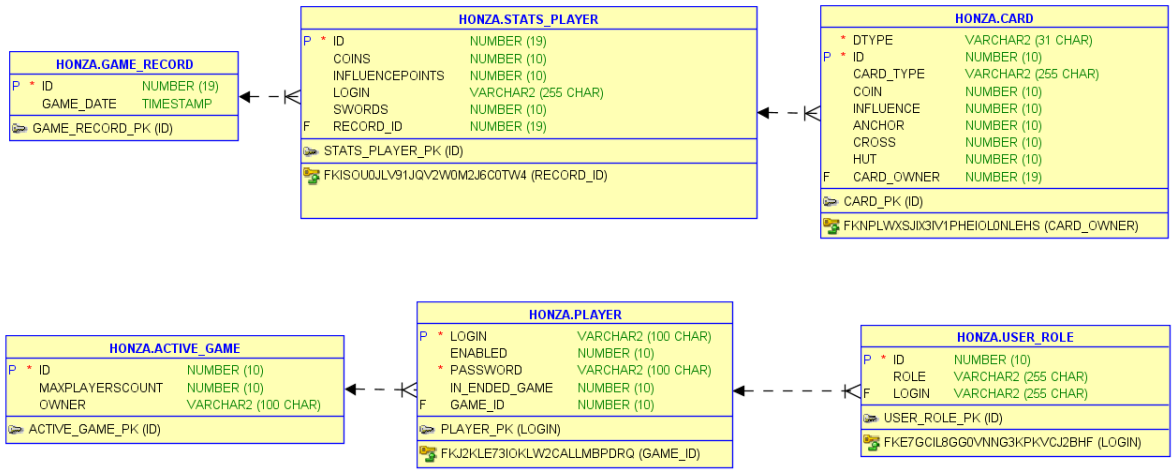
jak se pousti a instaluji zavislosti

5.3 Node.js

přidani slideru

5.4 Schema DB

model.png



Obrázek 2: Databázový model

5.5 Spring Security a Bcrypt

Bezpečnost aplikace je řešená pomocí Spring Security, jenž po nakonfigurování blokuje URI na jenž uživatel nemá přístup.

Hesla v databázi nejsou samozřejmě uloženy pouze jako text. Pro heslo se při registraci uživatele vygeneruje hash pomocí Bcryptu. Tento hash je pak v databázi uložen místo hesla samotného. Při přihlašování Spring Security kontroluje zda přihlašovací heslo odpovídá vygenerovanému hashi.

5.6 Návrh aplikace Angular / Spring

Jak to nakreslit, popr slovne ?

ulm tridni diagram

5.7 Aspecty

Par rádku kodu s popisem

5.8 Jak to komunikuje

P

5.9 Websockets

Websockety je nutno nastavit jak na straně serveru (Springu) tak na straně klienta (AngularuJS).

Moc kodu ?

6 Analýza a implementace testu

6.1 Jasmine

6.1.1 Konfigurace a pouštění

Jasmine testy se pouštění přes Node.js příkazem "npm test"ve složce se souborem package.json. V němž je uvedena cesta ke konfiguračnímu souboru karmy v tomto souboru jsou uvedeny cesty k testům a pluginy pro spuštění prohlížeče v němž se testy jeden po jednom spouštějí.

pridat par radku conf souboru ?

6.1.2 ukazkovy test

kod s komentari

6.2 Protractor

6.2.1 Konfigurace a pouštění

Protractor se pouští stejně jako Jasmine přes Node.js ve složce se souborem package.json. Ovšem příkazem "npm protractor run"

6.2.2 ukazkovy test

kod s komentari

6.3 Jersey

6.3.1 Konfigurace a pouštění

Jersey testy se spouštějí automaticky při kompilaci Jersey testy jsou naming

6.3.2 ukazkovy test

kod s komentari

7 Závěr

a

7.1 Největší komplikace při vývoji

co tech umi neumi

7.2 Největší komplikace při vývoji

a

7.3 Jak by se dala aplikace vylepšit

a

7.4 Co chybelo

a

Literatura

[1] Ferda Marvenec: Kdesi cosi.

[2] **angularjs** [cit. 2017-01-04]. *Dostupne z: https://docs.angularjs.org/guide/introduction*

[3] **nevim** [cit. 2017-01-04]. *Dostupne z: https://neoteric.eu/single-page-application-vs-multiple-page-application*

[4] **taka sablona** [cit. 2017-01-04]. *Dostupne z: https://www.zdrojak.cz/clanky/zaciname-s-angularjs/*

[5] **angularJS.org** [cit. 2017-02-04]. *Dostupne z: https://docs.angularjs.org/guide/databinding*

[6] **Oracle documentation** [cit. 2017-06-04]. *Dostupne z: https://docs.oracle.com/javaee/7/tutorial/webservices-intro001.htm#GIJVH*

[7] **Oracle documentation** [cit. 2017-06-04]. *Dostupne z: https://docs.oracle.com/javaee/7/tutorial/webservices-intro002.htm#GIQSX*

[8] **Oracle documentation** [cit. 2017-06-04]. *Dostupne z: http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html*

[9] **Tutorialspoint** [cit. 2017-06-04]. *Dostupne z: https://www.tutorialspoint.com/restful/restful_introducti*

[10] **Williamson, Ken.** [cit. 2017-07-04]. *Dostupne z: Learning Angularjs: A Guide to Angularjs Development. Sebastopol, CA: O, 2015.*

[11] **Jesse Palmer** [cit. 2017-08-04]. *Dostupne z: Testing Angular Appliaccti on Cover Angular 2. 2016*

A Instalace aplikace

Jak nainsatlovat maven a npm ?