Aplicatie e-commerce : E-COMM
Made by : Vlasceanu Silviu : 344

Aplicatia simuleaza la nivel minimal flow-ul unei aplicatii e-commerce.

Baza de date este mongodb v3.0.

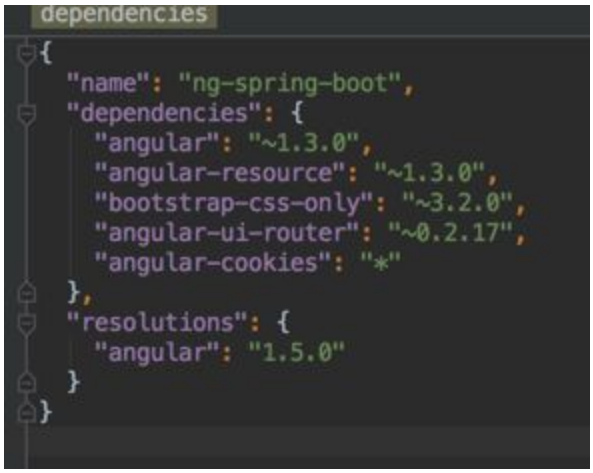
Este modelata pe arhitectura client-server.
Pentru server am folosit framework-ul Spring Boot.
Managerul de dependinte este Maven v3.0

pom.xml:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.ecomm</groupId>
    <artifactId>ecommerce</artifactId>
    <version>1.0-SNAPSHOT</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.3.2.RELEASE</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.hsqldb</groupId>
            <artifactId>hsqldb</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-mongodb</artifactId>
        </dependency>
    </dependencies>
</project>
```

Pentru client am folosit framework-ul Angular
Managerul de dependinte este Bower

bower.json:



Pentru conexiunea la baza de date am folosit o clasa de configurare:

Artifactul `spring-boot-starter-data-mongodb` contine toate dependintele necesare conectarii: driver Mongo, utilitare spring-mongo.

Serverul e construit cu arhitectura Spring Web MVC. Requesturile de la client sunt interceptate de metode din controllere, business-ul este definit in servicii, iar layerul de acces la baza de date e definit de repositoriuri.

Controller:

```java
@RestController
@RequestMapping("/items")
public class ItemController {

    @Autowired
    private ItemService itemService;

    @RequestMapping(value = "/find-all", method = RequestMethod.POST)
    public ResponseDTO findAllItemsFiltered(@RequestBody ItemFilterDTO itemFilter) {
        ResponseDTO responseDTO = new ResponseDTO();
        responseDTO.setData(itemService.findAllFiltered(itemFilter));
        return  responseDTO;
    }

    @RequestMapping(value = "/save", method = RequestMethod.PUT)
    public ResponseDTO updateItem(@RequestBody Item updatedItem) {
        ResponseDTO responseDTO = new ResponseDTO();
        responseDTO.setData(itemService.saveItem(updatedItem));
        return responseDTO;
    }

    @RequestMapping(value = "/count-cart-items", method = RequestMethod.GET)
    public ResponseDTO getTotalCartItemNumber() {
        ResponseDTO responseDTO = new ResponseDTO();
        responseDTO.setData(itemService.countCartItems());
        return responseDTO;
    }

}
```

Serviciu:

```java
/**
 * Created by Silviu on 2/10/16.
 */
public interface ItemService {

    Set<Item> findAllFiltered(ItemFilterDTO filterDTO);

    Item saveItem(Item item);

    Long countCartItems();

}
```

Repository:

```java
package com.ecomm.repository;

import ...

public interface ItemRepository extends MongoRepository<Item, String> {

    List<Item> findAllByInCartTrue();

    Set<Item> findAllByNameLike(String itemName);

}
```

Pentru securizarea accesului la resurse, am folosit Spring Security:

```java
@Configuration
@Order(SecurityProperties.ACCESS_OVERRIDE_ORDER)
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .csrf().disable() //elimin cauze de 403
                .httpBasic()
                .and()
                .authorizeRequests()
                .antMatchers("/bower_components/**", "/app/**", "/").permitAll() //url uri pt non-auth
                .anyRequest().authenticated()
                .and()
                .logout();

    }

}
```

Am definit, prin Spring Security, un user cu rol de USER, in memorie:

```
security.user.name=user2
security.user.password=1234
```

Documentul de Mongo, definit ca POJO:

```java
@Document
public class Item {

    @Id
    private String id;

    private boolean inCart;
    private String name;
    private Double price;
    private List<NomCategory> categories;

    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public boolean isInCart() { return inCart; }

    public void setInCart(boolean inCart) { this.inCart = inCart; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public Double getPrice() { return price; }

    public void setPrice(Double price) { this.price = price; }

    public List<NomCategory> getCategories() { return categories; }

    public void setCategories(List<NomCategory> categories) { this.categories = categories; }
}
```

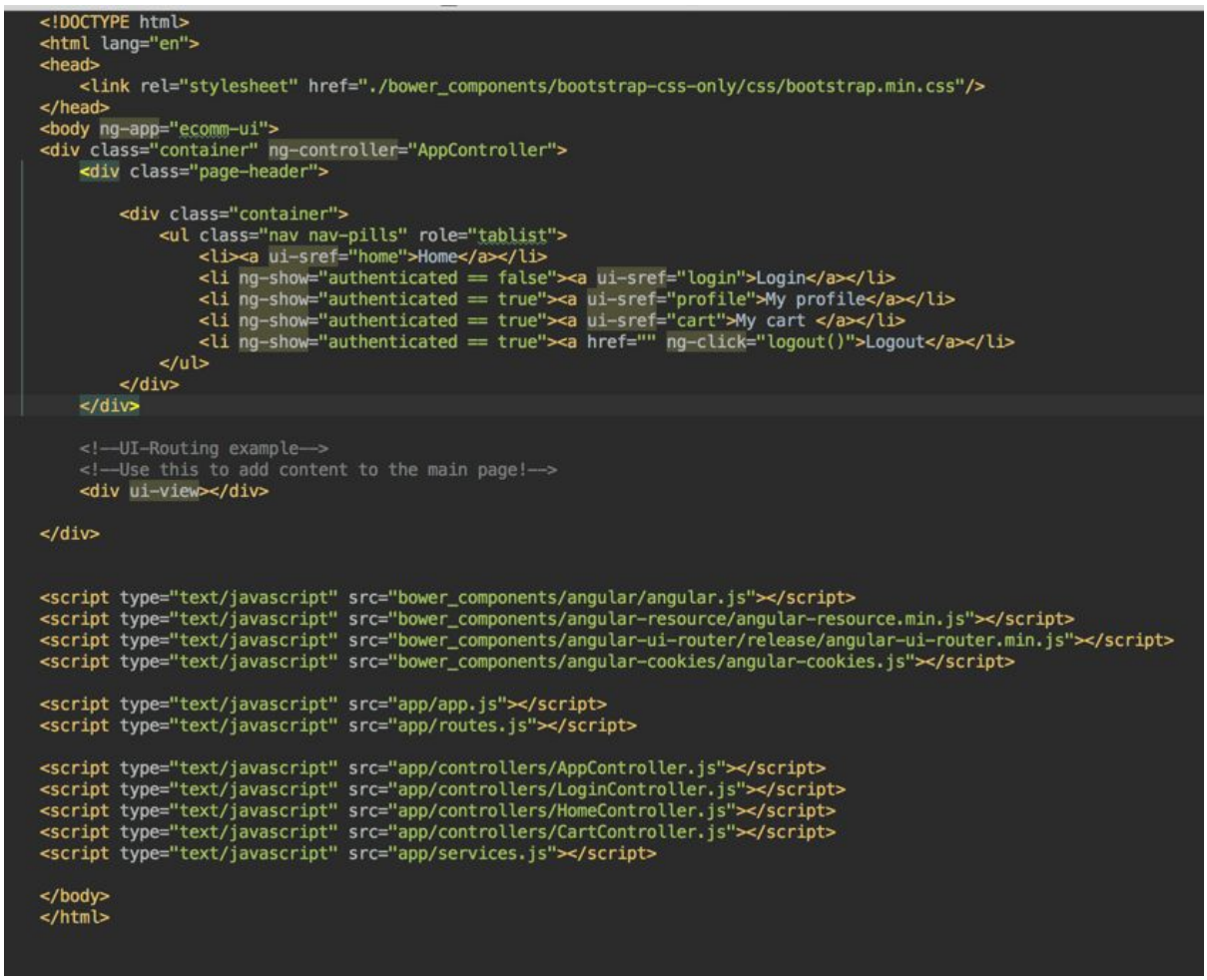
file:/Users/Silviu/E-COMM/src/main/resources/static/app/controllers/LoginController.js

Pentru pornirea serverului, am folosit metoda data default de spring-boot:

```java
//IMPORTANT: adnotarea asta cauta sub ierarhia de sub locatia clasei asteia !!!
 //sunt configurate implicit beanuri ce inlocuie web.xml
//e pornit implicit un tomcat pe 8080 !
@SpringBootApplication
public class Application {

    public static void main(String[] args) { SpringApplication.run(Application.class, args); }

    @Bean
    public EmbeddedServletContainerFactory servletContainer() {
        TomcatEmbeddedServletContainerFactory factory = new TomcatEmbeddedServletContainerFactory();
        factory.setPort(8081);
        factory.setSessionTimeout(30, TimeUnit.MINUTES);
        factory.setContextPath("");
        return factory;
    }

}
```

Am configurat un servlet de tomcat embedded sa porneasca pe portul 8081.

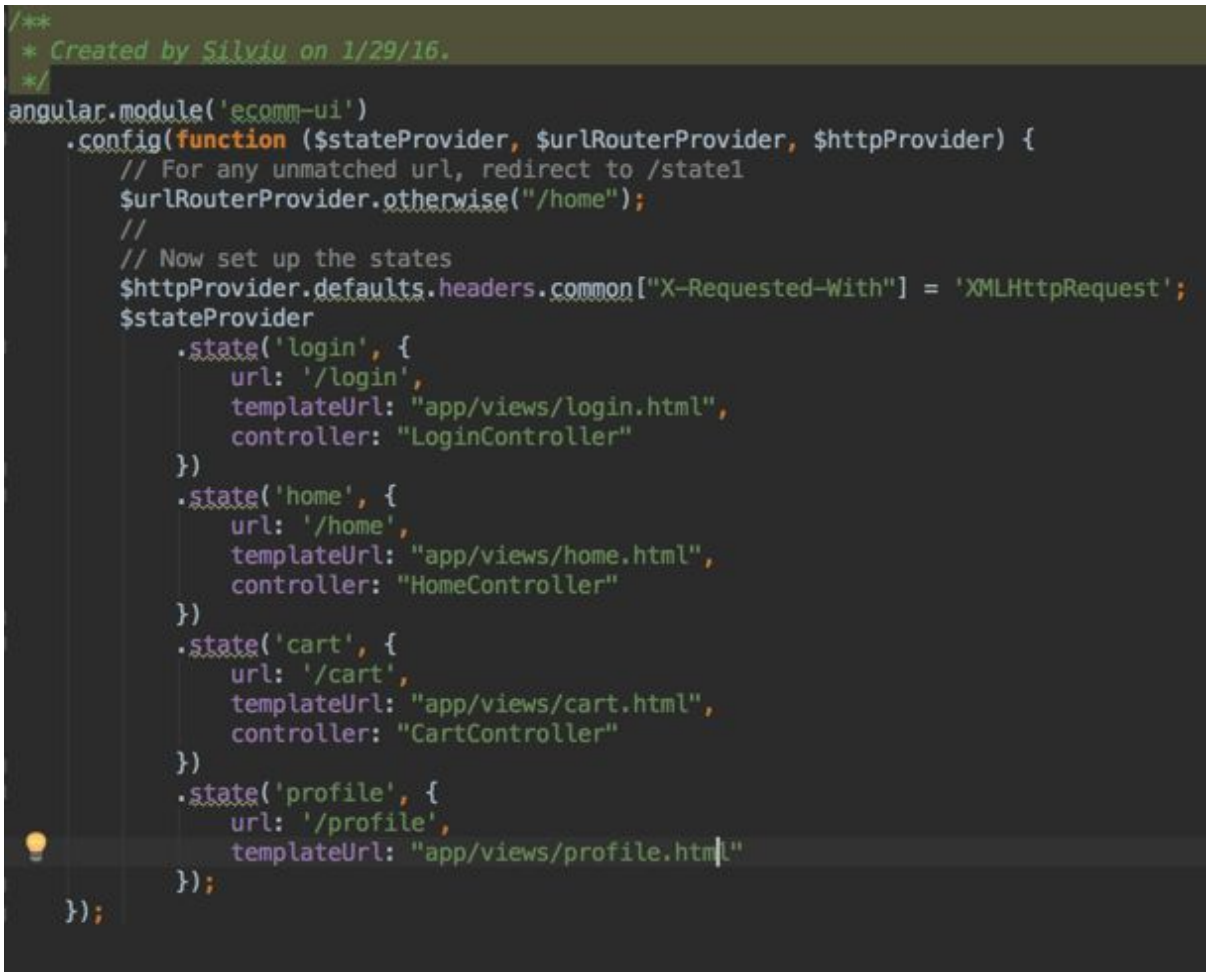
Pe partea de front, conform SPA, am folosit index.html, unde voi injecta view-uri:



```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="./bower_components/bootstrap-css-only/css/bootstrap.min.css"/>
</head>
<body ng-app="ecomm-ui">
<div class="container" ng-controller="AppController">
    <div class="page-header">

        <div class="container">
            <ul class="nav nav-pills" role="tablist">
                <li><a ui-sref="home">Home</a></li>
                <li ng-show="authenticated == false"><a ui-sref="login">Login</a></li>
                <li ng-show="authenticated == true"><a ui-sref="profile">My profile</a></li>
                <li ng-show="authenticated == true"><a ui-sref="cart">My cart </a></li>
                <li ng-show="authenticated == true"><a href="" ng-click="logout()">Logout</a></li>
            </ul>
        </div>
    </div>

    <!--UI-Routing example-->
    <!--Use this to add content to the main page!-->
    <div ui-view></div>

</div>


<script type="text/javascript" src="bower_components/angular/angular.js"></script>
<script type="text/javascript" src="bower_components/angular-resource/angular-resource.min.js"></script>
<script type="text/javascript" src="bower_components/angular-ui-router/release/angular-ui-router.min.js"></script>
<script type="text/javascript" src="bower_components/angular-cookies/angular-cookies.js"></script>

<script type="text/javascript" src="app/app.js"></script>
<script type="text/javascript" src="app/routes.js"></script>

<script type="text/javascript" src="app/controllers/AppController.js"></script>
<script type="text/javascript" src="app/controllers/LoginController.js"></script>
<script type="text/javascript" src="app/controllers/HomeController.js"></script>
<script type="text/javascript" src="app/controllers/CartController.js"></script>
<script type="text/javascript" src="app/services.js"></script>

</body>
</html>
```

Pentru routing in pagina, am folosit angular-ui, directiva ui-view fiind specifica.

Configurarea rutelor se face in routes.js:



```
/**
 * Created by Silviu on 1/29/16.
 */
angular.module('ecomm-ui')
    .config(function ($stateProvider, $urlRouterProvider, $httpProvider) {
        // For any unmatched url, redirect to /state1
        $urlRouterProvider.otherwise("/home");
        //
        // Now set up the states
        $httpProvider.defaults.headers.common["X-Requested-With"] = 'XMLHttpRequest';
        $stateProvider
            .state('login', {
                url: '/login',
                templateUrl: "app/views/login.html",
                controller: "LoginController"
            })
            .state('home', {
                url: '/home',
                templateUrl: "app/views/home.html",
                controller: "HomeController"
            })
            .state('cart', {
                url: '/cart',
                templateUrl: "app/views/cart.html",
                controller: "CartController"
            })
            .state('profile', {
                url: '/profile',
                templateUrl: "app/views/profile.html"
            });
    });
```

$httpProvider populeaza requesturile din front cu un header specific Spring Security; cu ajutorul sau, se va arunca 401 la client in cazul accesului neautorizat.

Modulul principal in client se numeste ecomm-ui si are 2 copii principali: ecomm-ui.controller si ecomm-ui.services; e configurat in app.js

```
(function(angular) {
    angular.module("ecomm-ui.controllers", []);
    angular.module("ecomm-ui.services", []);
    angular.module("ecomm-ui", ["ngResource", "ngCookies", "ecomm-ui.controllers", "ecomm-ui.services", "ui.router"])
}(angular));
```

Controller Angular:

```
/**
 * Created by Silviu on 2/7/16.
 */
angular.module("ecomm-ui")
    .controller("CartController", function ($scope, OrderService) {

        $scope.orders = [];

        $scope.loadOrders = function() {
            $scope.orders = [];
            OrderService.findAll(function (serverData) {
                console.log(serverData);
                $scope.orders = serverData.data;
            });
        };

        $scope.loadOrders();

        $scope.deleteOrder = function (order) {
            OrderService.delete({orderId: order.id}, function() {
                $scope.loadOrders();
            });
        };

        $scope.purchase = function (order) {
            if (confirm('Are you sure you want to buy these items? ' + item.name)) {
                $scope.deleteOrder(order);
            }
        }
    });
```

View Angular pt controllerul anterior

```
<div>
    <div class="form-group" ng-repeat="order in orders">
        <div class="row">
            Item name: {{order.item.name}} <br/>
            Item price {{order.item.price}} <br/>
            Ordered quantity: {{order.orderedQuantity}} <br/>
            Total Price: {{order.orderedQuantity * order.item.price}} <br/>

            <button class="btn-default" type="button" title="Add to cart" ng-click="deleteOrder(order)">
                Remove order
            </button>
            <button class="btn-default" type="button" title="Add to cart" ng-click="validateOrder(order)">
                Purchase these items
            </button>
        </div>
    </div>
    <div class="alert alert-info" role="alert" ng-show="orders.length == 0">
        No items in cart!
    </div>
</div>
```

Serviciu Angular:

```
/**
 * Created by Silviu on 1/29/16.
 */
angular.module("ecomm-ui.services")
    .factory("ItemService", function ($resource) {
        return $resource(null, {}, {
            findAll: {
                method: "POST",
                url: "/items/find-all"
            },
            update: {
                method: "POST",
                url: "/items/save"
            },
            countCartItems: {
                method: "GET",
                url: "/items/in-cart"
            }
        });
    })
```

Controllerul principal pt index.html este AppController:

```
angular.module("ecomm-ui.controllers")
    .controller("AppController", function ($scope, $rootScope, $http, $state) {

        /**
         * @author Silviu
         * @param credentials
         * @param callback
         *
         * Function to get user credentials from server session
         * It is called upon refreshing the browser page
         */
        $rootScope.authenticate = function (credentials, callback) {

            var headers = credentials ? {
                authorization: "Basic "
                + btoa(credentials.username + ":" + credentials.password)
            } : {};

            $http.get('login/user', {headers: headers}).success(function (data) {
                if (data.name) {
                    $rootScope.principal = data;
                    $rootScope.authenticated = true;
                } else {
                    $rootScope.authenticated = false;
                }
                callback && callback();
            }).error(function () {
                $rootScope.authenticated = false;
                callback && callback();
            });
        };

        $rootScope.authenticate();

        /**
         * Function for logging out
         * Logout url is managed by Spring Security
         */
        $scope.logout = function () {
            $state.go('login');
            $http.post('logout', {}).success(function () {
                $rootScope.authenticated = false;
                //$state.go('login');
            }).error(function () {
                $rootScope.authenticated = false;
            });
        };

    });
```

La refresh in client(browser), se va verifica daca userul exista pe sesiunea de server, caz in care se va popula o variabila $rootScope.authenticated, ce gestioneaza afisarea de informatii la ecran

Fisierul application.properties contine configurarea unor proprietati specifice spring-boot:

```
#spring.jpa.hibernate.ddl-auto=create-drop
spring.output.ansi.enabled=ALWAYS

#logging.level.root=WARN
logging.level.org.springframework.web=DEBUG
logging.level.org.hibernate=ERROR

security.user.name=user2
security.user.password=1234

#spring.data.mongodb.uri= mongodb://admin:TPPqfEw7PjLi@127.2.169.130:27017/local
spring.data.mongodb.repositories.enabled=true
```

Aplicatia poate fi impachetata intr-un jar executabil prin rularea comenzii mvn install; se va folosi un plugin default de build din spring-boot.