EXERCERA

Advanced Web Technologies Project

Hazem Al Saied
Hazemalsaied@gmail.com

Anh Duc Vu
duc1909@gmail.com

Liang Yiqing
Celestelyq@gmail.com

Supervised by:
Prof. Geoffray Bonnin

Université de Lorraine

Advanced Web Technologies, Natural Languages Processing

11 January 2016

Contents

1 Introduction

The objective of our project is to create a community of students for finding practical exercises and their solutions. the system should provide multiple functionalities attached to finding exercises in a given scientific domain, solving exercises, adding exercises, deleting exercises and rating the difficulty of a given exercise.

What is interesting in this project is that there is no famous website for archiving the official exercises and their solutions. In web we can find a lot of question answering website such as Quora or Stack Exchange.. but what we were thinking about is kind of interactive archive of official archive of exercises classified according to the category and the key words.

Our project can be considered as a reference for students in the period of exams where they need to solve a lot of exercises for owning the required flexibility in exams.
What could be difficult is to achieve an exhaustive system analysis. We think that our idea is not implemented yet, so the creativity in implementing it is the master key in success. Another difficulty is that we have to build the system in unfamiliar technologies such as Angular in very short and stressed period.

2 Eercera Funtionalities

The main functionalities of Exercera is listed here according to the permission of the user:

Guest Functionalities

- Exercises browsing.
- Solutions browsing.
- Reviews browsing.
- Auto complete filtering.

Student Functionalities:

- Adding Exercises.
- Proposing Solutions.
- Adding Reviews.
- Rating Exercises.
- Reporting Exercises.

- Deleting Personal Exercises.

Note.1: User Management System contains tow other permission, Profs and admins, bu there is no available interface for doing there functionalities.
Note.2: Each user role takes the permissions of the lower one.
Note.3: We could propose more integrated functionalities to obtain robust and transparent system . . . but that would cost a lot of time!

3 Exercera Interfaces

In this section we will show multiple screen shots of the user interface of Exercera, for owning a concrete understanding of the functionalities of the web application.

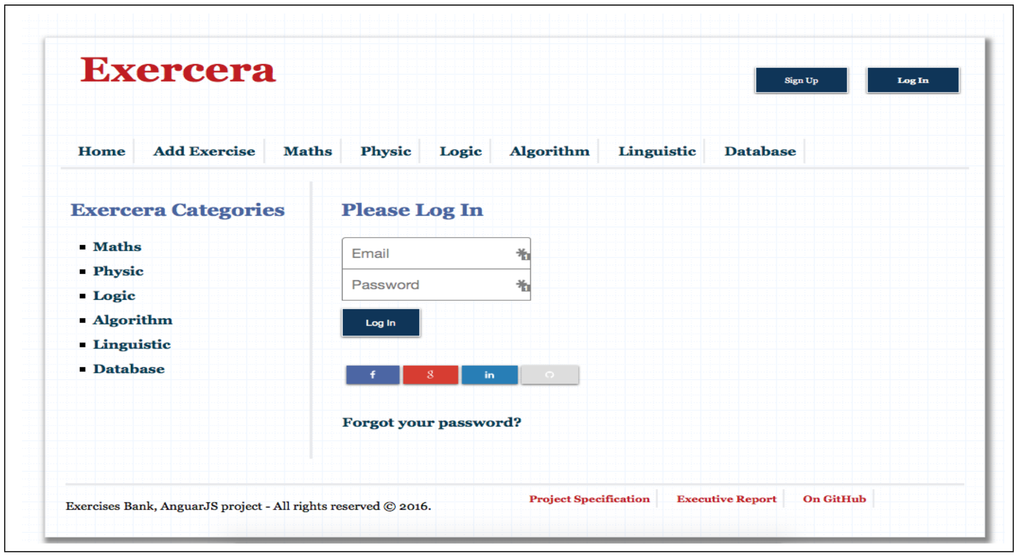
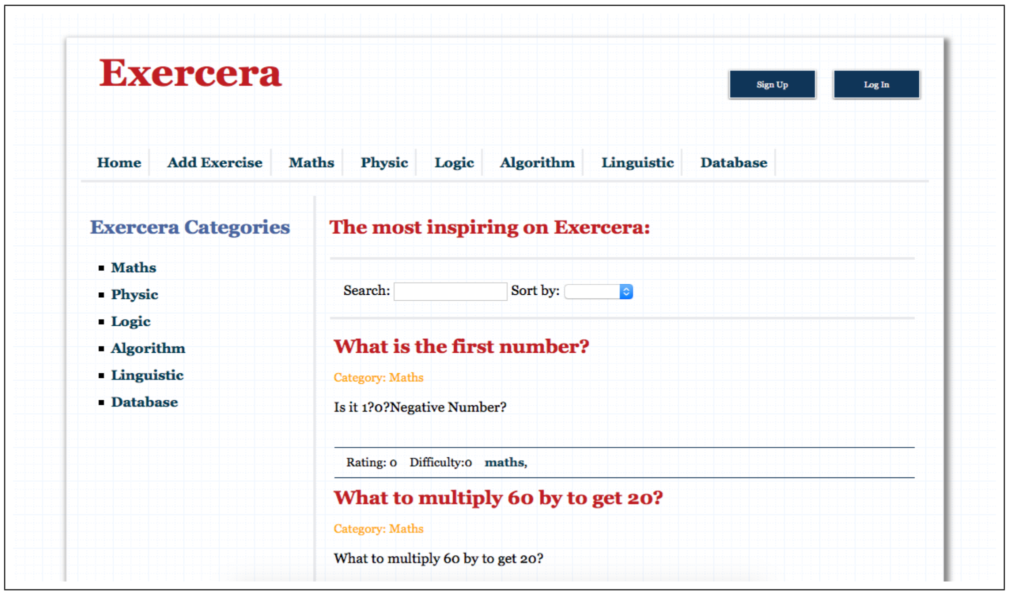


Figure 1: Global shot of the web application with the loginf functionality, the users can register using their social accounts in a couple of seconds. They can also use the classical way. The Horizontal menu list quick links of functionalities and the categories of the application which are repeated in the vertical one!.



What is the first number?

Category: Maths

Is it 1?0?Negative Number?

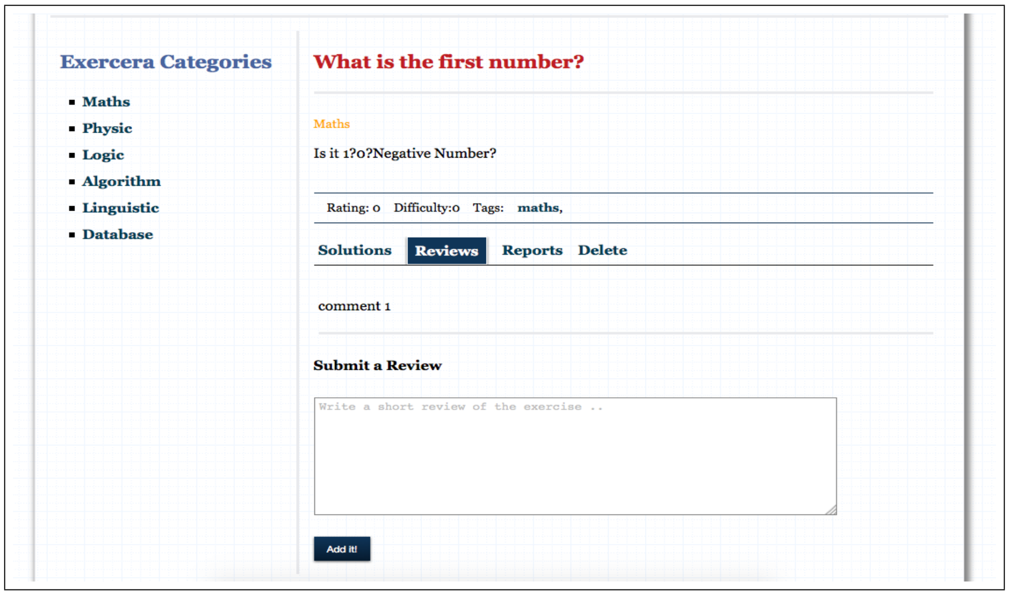
Rating: 0 Difficulty:0 maths,

What to multiply 60 by to get 20?

Category: Maths

What to multiply 60 by to get 20?

Figure 2: Exercises interface of the web application with a list of the newly ad exercises and the ability of filtering the exercises, filtering them and listing them according to the category and the tags.


Exercera Categories

- Maths
- Physic
- Logic
- Algorithm
- Linguistic
- Database

What is the first number?

Maths

Is it 1?0?Negative Number?

Rating: 0 Difficulty:0 Tags: maths,

Solutions Reviews Reports Delete

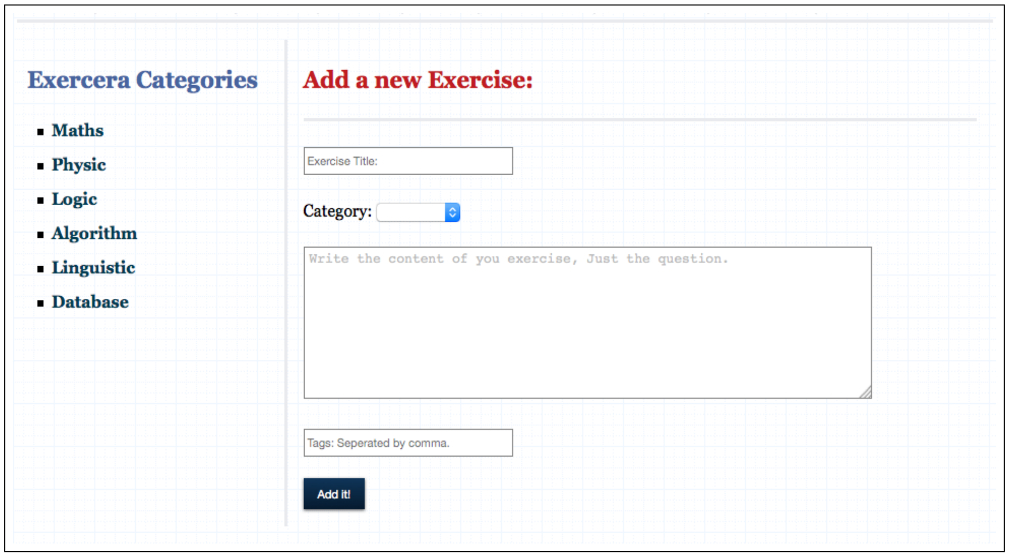
comment 1

Submit a Review

Write a short review of the exercise ..

Add it!

Figure 3: A Single Exercise interface of the web application tabs of solutions, reviews, reporting and deleting functionalities. It shows also other details as rating, difficulty, category and tags. The authenticated user only can add solutions , reviews ..


Exercera Categories

- Maths
- Physic
- Logic
- Algorithm
- Linguistic
- Database

Add a new Exercise:

Exercise Title:

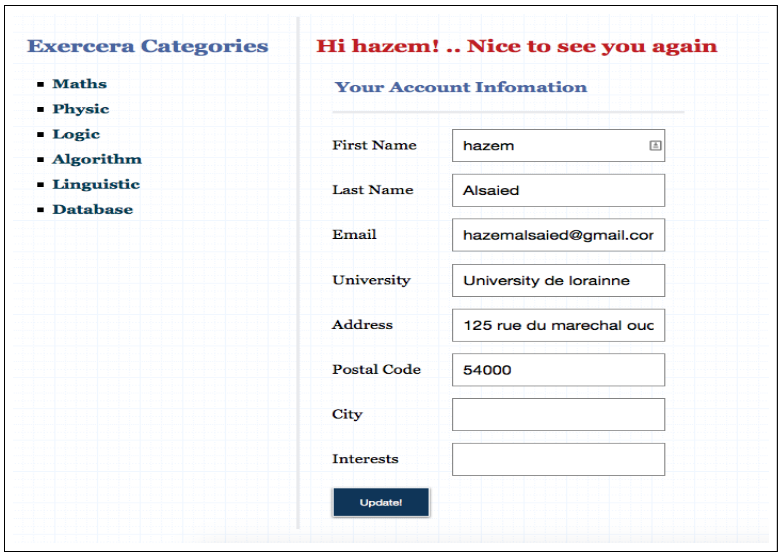
Category:

Write the content of you exercise, Just the question.

Tags: Seperated by comma.

Add it!

Figure 4: Adding new Exercise interface of the web application where the authenticated user can add the content of an exercise, specify its category and add its details as rating, difficulty, category and tags.



Hi hazem! .. Nice to see you again

Your Account Infomation

First Name	hazem
Last Name	Alsaied
Email	hazemalsaied@gmail.cor
University	University de lorainne
Address	125 rue du marechal ouc
Postal Code	54000
City	
Interests	

Figure 5: My Account interface whose link appear after logging in and gic the user the abaility to add his personal details and list the exercises which the user added .

4 Exercera Modules

The project is developed using AngularJS which permit to us to benefit of the MVC structure to build our project. We will quickly list the layers and their technical requirements:

4.1 Model Layer

The Model layer represents the part of your application that implements the business logic of the controllers. It is responsible for retrieving data and converting it into suitable format for the application and its layers. it includes CRUD operation over the database, processing, associating and other tasks related to handling data.
Technical requirement: SQL, PHP, AngularJS.

4.2 View layer

The View renders a presentation of modeled data. Being separated from the Model objects, it is responsible for using the information it has available to produce meaningful interfaces for users.
Technical requirement: AngularJS, Php, Html, CSS.

4.3 Controller Layer

althought that we can't speak about coherent controller layer while developing using AngularJS but we still need some customizations. The Controller layer does this customization by handling user requests. It is responsible for rendering a response with the aid of both the Model and the View layer.
Technical requirement: AngularJS, Php, Html, CSS.

4.4 User Management Module

Using UserApp which is a cloud-based user management API for web apps with the purpose to relieve developers from having to program logic for user authentication, sign-up, feature/property/permission management..
Technical requirement: AngularJS, Php, Html, CSS.

5 UserApp: User Management System

As an application of the distributed systems concept and for not wasting time we decided to use a cloud service for managing user system and permissions. We chose UserApp which is a cloud-based user management API for web apps with the purpose to relieve developers from having to program logic for user authentication, sign-up, invoicing, feature/property/permission management, and more.

The basic idea of UserApp is to create an account to get a key for you integrating it with our application. Copying the necessary libraries to the web application, Inserting the key for attaching the libs with the application and configuring the plugin on the local side and on the service-dashboard side.

5.1 UserApp credits

The credits of UserApp are as following:
User name: Exercera
Email: hazemalsaied@gmail.com
Password: 9232@Ex

5.2 UserApp dashboard features

By entering the dashboard you have the ability of:

1. Browsing users, editing, deleting ..
2. Adding properties to the user schema. In other words, customized fields to my account pages.
3. Adding permissions or roles or features.

[0]For more details, review: UserApp AngularJS Module

4. Adding payment and invoicing infos.

5. Adding one of the payed plug in for more functionalities.

6. Configuring the parameters of session timeout, the social network authentication, forgetting password, using email activation or not and others..

5.3 Quick Tutorial

You can take a quick look of UserApp features and how are they integrated in the code by reviewing this tutorial written by a co-founder of UserApp. For example, for controlling the visibility of a div according to the permission we can write the following code:

```
<div ng-show="user.authorized">Welcome!</div>
```

We can also use the user object to access properties on the logged in user:

```
<span>{{ user.first_name }}</span>
```

And for creating routes and templates for login and signup, and use the directives to connect them to UserApp (examples: login.html and signup.html) we can modify the parameters of public/js/app,js as following:

```
$routeProvider.when('/login', {templateUrl:
'partials/login.html', public: true, login: true});
$routeProvider.when('/signup', {templateUrl:
 'partials/signup.html', public: true});

$routeProvider.otherwise({redirectTo: '/home'});
```

Where **Add public: true** make the routes you want to make it public (i.e. /login and /signup).
Add login: true on the login route. The otherwise() route should be set to the route that should be shown after login.

5.4 Technical Details

Here are details of using UserApp with AgularJS as front-end and php as Back-End:

1. Include the SDK in the file *index.html*.

2. Add the UserApp module to the app's dependencies in a javascript named *app.js*.

3. Inject and initiate the service in the run() block in *app.js* with an App Id.

4. Create routes & templates for login and signup, and use the directives to connect them to UserApp. For here, the routeProviders are also set in *app.js*.

5. Add a logout link in the app. When clicked, it will end the session and redirect to the login route.

6. Use *ng-show="user.authenticated"* to show elements that should only be visible when logged in, like "My Account" and "Log out" here. This is set in the file *header.html*.

7. Use the *user* object to access properties on the logged in user, including first name, last name, email ect.

8. Use the *ua-oauth-link* directive to sign up and log in users with social network accounts, such as Facebook, Google, Instagram and Github.

9. Now it's possible to sign up and log in.

We use PHP as back-end, here are the following steps:

1. Download and load the PHP SDK together with the PHP widget

2. Include it into the app

3. Configure it with the App Id

4. To authenticate an API call, look for a cookie named *ua_session_token*. If this contains a value, we could validate it with *User::loginWithToken()*. If it is a valid token, a new session will be created and *User::authenticated()* will return true as long as this session is alive.

5. Run the PHP app and make log in to the front-end and then try to perform any API calls.

6 Database analysis

According to the requirements of project, we have following entities: exercise, solution, user, tag, comment, report, category. Each of them is represented by a table in database. *The structure of database is shown in Figure.*

Between exercise and tag, we have a n-n relationship. An exercise may have several tags and vice versa. To reduce the duplication of data, we use an intermediate table name exercise_tag to transform the n-n relationship into 2 1-n relationships:

exercise—exercise_tag and tag—exercise_tag. For the exercise entity, it contains some properties that are hardly changed such as: title, content and some properties changed frequently such as: difficulty, rating.

That's the reason why we divide exercise entity into 2 table: exercise to contain stable information and exercise_detail to contain fragile data. For user management, we use another system to handle. Therefore, the user table in the

ER diagram is use for representing the system structure, we do not have any operation on that table.

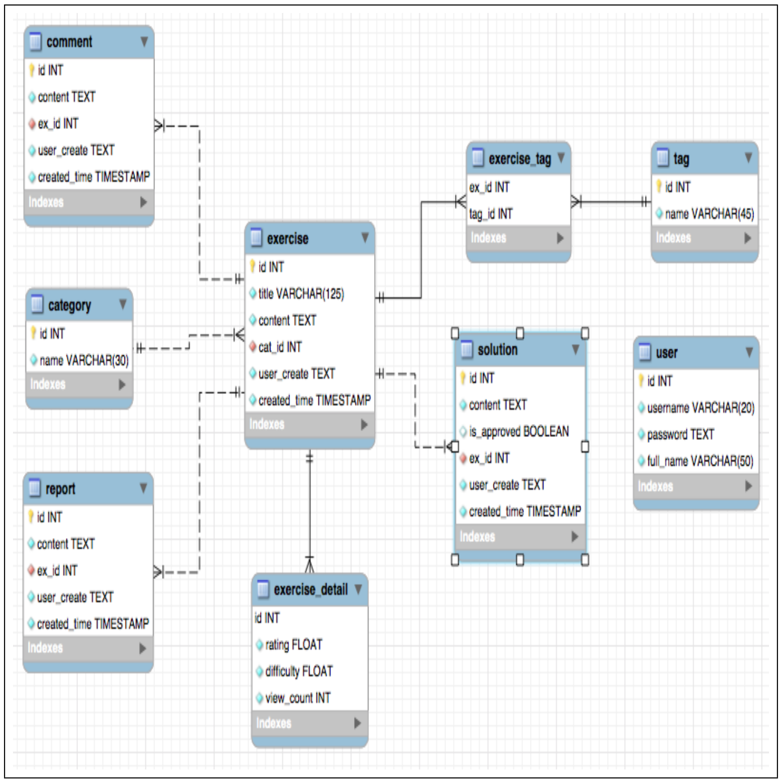


Figure 6: Database Diagram

7 Code Survey

In this section we will take a quick look of the code of the project.

7.1 Database-initiation Folder

The first folder of the project is database-initiation and it contains multiple files which are either sql script for creating the database schema and filling up the database with some demonstrative data or MySql Work Bench files for creating the database diagram.

7.2 Lib Folder

Since AngularJS is a front-end framework, all authentication with this module is only done on the client-side, with direct communication with UserApp.

So, to authorize the back-end API, we need to install the back-end of UserApp to get all needed services on the server-side .

This folder, then, contains the files of the back-end of UserApp. It's simething like a library and we didn't touch it!

7.3 Public Folder

This folder is the container of our code, the folder itself is created by UserApp and we had to move all our codes to it. The folder contains the following folders:

- **CSS and Img Folders:**
  These folders contains the scripts and the images necessary to create the visual identity of the website.

- **Includes Folder**
  It includes the files for configuring the encoding and the connection to the database.

- **AJAX Folder**
  This folder can be considered as a part of the model layer. It contains more than 30 Php files responsible of doing all the CRUD operation against the database.
  the typical content of these files is: creating an instance of the connection with the database, preparing some query, executing the query inside exception handler, and returning the value of the query.

  One example: the *'category_get_all.php'* provide the service to get all the categories from database. It serves for displaying the side bar at front-end pages.
  To carry out the CRUD operations, we use the mysqli, an improvement of mysql api, with object oriented approach to do the tasks. The ajax services also handle the error when dealing with database.

  For some complex operations that have to work with multi table, we use the transaction management to assure the stable of the system.

  To avoid the duplication of database connection, we also apply the singleton pattern. This pattern is implemented in the *"includes/db_instance.php"* file. Whenever using a database connection, the system will check whether it is initiated, if yes, it will be reused, otherwise, it will be created.

  For data transfer between client and server, we use the json data format.

- **Test Folder**
  This folder can be considered as a part of testing the model layer. It contains more than 20 Php files responsible of testing all the CRUD operation

against the database.
It was used during the development of the AJAX Folder.

- **Partials Folder**
  This folder contains all the Html files of the interface except the index which exists at the root of Public folder and which is the start page of the application. The partials are very elegant Html files because of the Anjgular JS annotations.

  Constants folder contains the files of the header, the footer and the sidebar file which don't refresh their content most the time.

  Tabs folder contains html hooks for the page of showing a single exercise, *'Partials/exercise.html'* , which is the most complicated page of the application.
  They were created to keep the elegance of other pages and to keep the high readability of the code. the partial folders contains also predefined html pages for the UserApp functionalities as set-password.html

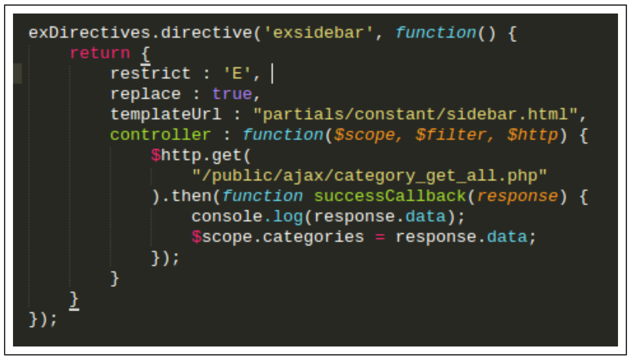
- **JS Folder**
  This folder contains the AngularJS codes. It contains only 4 files which conclude the most critical part of our work.

  - **App.js File**
    In this file we declare the main module of the whole application, myApp. We link this module with other modules, In technical words, we create its dependencies which are declared in the other files, such as controllers and directives, or predefined ones, such as routing module.
    In this file we also configure the routing of views for directing the links to the appropriate pages and add the parameters of UserApp for controlling the visibility of pages according to the given permissions.
    The last section of this file is for linking the UserApp id with our application.
  - **Directives.js FIle**
    In this file we declare multiple directives for linking the *'index.html'* with the files of constants folder or for linking the *'exercixe.html'* with the files of Constant folder.

    For the first directive nothing important exists in the controllers of the directives. but in the second linking, for directives, such as exreviews or exsolutions, the controller read user request, handle it and return the suitable response. It either add a solution, a review or report an exercise or other similar requests.

The technical concepts used in this directive is pretty simple, where all what we do is to parse data, invoke a service and show the result by writing the right expression.

– **Controllers.js FIle**
This file contains multiple controllers for creating the business logic of the partials pages. Each controller offer one or more functionalities invoked in the scope of the controller.
The accountCtrl for example is responsible for updating the personal information of users by doing a convocation of one function of User-App API.
The exsController is used to invoke the http service for retrieving the required data for displaying a list of the exercises . The ex controller contains the code for invoking the http service for retrieving the required data for displaying the given exercise and other functions for controlling the behavior of exercises tabs.



```
exDirectives.directive('exsidebar', function() {
    return {
        restrict : 'E',
        replace : true,
        templateUrl : "partials/constant/sidebar.html",
        controller : function($scope, $filter, $http) {
            $http.get(
                "/public/ajax/category_get_all.php"
            ).then(function successCallback(response) {
                console.log(response.data);
                $scope.categories = response.data;
            });
        }
    }
});
```

Figure 7: Controller example

To carry out the data get and request, we use the $http built-in module of AngularJS. The front-end home page is divided into several parts, each one in angularjs is called a directive. Each directive have its own controller. In the figure above, the controller of exsidebar is use to load the categories from database by using ajax service category_get_all.php. The response data is under the json format and is bind to $scopy.categories.

7.4 Root files

the files found at the root of the project are either lo files for recording the some information by Angular JS and UserApp or php files for loading and configuring the UserApp

8 Technical Guideline

For running the project the following notes should be considered:

- **For Creating the Database and testing data:**
  - open the database-initiation folder situated in the root of the project.
  - run the file exercera_table_creation-v2
  - run the file exercera_data_initiation-v2.sql
  - Change the hoster, username and the password of your Mysql inside the file public/includes/db_config.php
- **Internet Connection**

  we load the libraries of Angular JS using online lnks. From the other side, UserApp cloud service is needed for logging and other user management stuffs. So, Internet connection is required.
- **Deployment issue**

  Please, Consider that deploying the project in nested folder of the server, the htdocs for MAMP, could cause problems. these problems would be solved by deploying the folders and files of the project in the root folder of the server, htdocs in the case of MAMP. Another solution is to review the parameters of routing AngularJS found in Public/JS/app.js.

  Make sure also to add the word public after the localhost link. Your link to the application would be something close of : '*http://localhost:8888/public/*'
- **Platform issue**

  Please, make sure to use local server, MAMP as an example, with PHP 5.5 .
- **Connection Database**

  we have prepared a lot of test of database transactions, you can consult them by going to the Public/test folder or navigating a link similar to http://localhost:8888/public/test/ which is used in our case.

9 Conclusion

After long days of coding we can say that we were able to create the core of the target idea. We have now a cool site with working functionalities, developed using the most modern technologies such as AngularJS and using one application of the mainstreaming concept of distributed system which is UserApp.

What we are proud about is that we did not live a very stressed day, the day before the assignment. Another good point is that now we know the key concepts of AngularJS which is very cool technology.

One of the main difficulties we faced was the starting point. where we have a lot to do with tools we are not used to use. Another bad thing was that while learning new concepts of angular we were forced to rewrite some parts of the previous code to make it respect a very basic concept of angular which we learned late.

For example, after filling the controller module with the implementation of the business logic we discovered that directive is very good concept to organize the code and make it reusable.

But finally we have to admit that what we have now is very simplified version of the desired system specification. We don't have any interfaces for managing the content by admins and profs. Furthermore, we don't have enough convincing tools for the existing functionalities. For example, the text box for adding a new exercise is very poor, the user can't add image or special letters ..

we can say that what we did is not more than brainstorming about AngularJS technology and what should exercise bank contain.

Appendices

A Useful Tutorials

Here we will list the most inspiring tutorial for learning Angular JS which we used for understanding Angular JS and implementing the functionalities of our application.

- **Shaping up with Angular.js**

  Learn to use Angular.js by adding behavior to your HTML and speeding up your application's responsiveness.

  A very useful tutorial as a beginning for getting familiar with the basic concepts of Angular.js. especially that it's sponsored by Google and produced by CodeSchool

- **PhoneCat Tutorial App**

  It's the most important one. It walks you through the construction of an AngularJS web app. The app you get at last is a catalog that displays a list of Android devices, lets you filter the list to see only devices that interest you, and then view details for any device.
  When you finish the tutorial you will be able to:

  - Create a dynamic application that works in all modern browsers.
  - Use data binding to wire up your data model to your views.
  - Create and run unit tests, with Karma.
  - Create and run end to end tests, with Protractor.
  - Move application logic out of the template and into Controllers.
  - Get data from a server using Angular services.
  - Apply animations to your application, using ngAnimate.
  - Identify resources for learning more about AngularJS.

  The tutorial guides you through the entire process of building a simple application, including writing and running unit and end-to-end tests. Experiments at the end of each step provide suggestions for you to learn more about AngularJS and the application you are building.

- **Task manager application using AngularJS PHP MySQL**

  A very simple tutorial, but in the same time very effective for learning the basic concepts of building the Model Layer. It explains how to create a simple Task Manager application with CRUD functionalities using AngularJS for the client side, PHP for server side communication and MySQL for database.