APLICAÇÕES PARA A INTERNET

Engenharia Informática

PHP MVC (Part 2)

Objectives:

(1) Use database storage to persist information on your web application;

(2) Follow the Model-View-Control architectural pattern;

Note the following:

- Every page MUST comply to the HTML5 standard and it must be properly validated as such;
- The web application should follow the Model-View-Control architectural pattern;
- The php code written MUST follow the "coding style guide" PSR [PSR-2, PSR-4];
- There is a video for each exercise showing the intended result;

Setting up the environment:

- If the Vagrant Homestead environment is not installed and configured, follow the instructions of worksheet 1.
- Run “vagrant up”
- Store your exercises on the “C:\<AInet>\pratica\ficha5” folder.
- Note: In alternative, create a site on Laragon (or similar tool) and, if possible, use the same URL to access the content (http://ainet.pratica.5.test/)

In this exercise, you will continue the previous worksheet by adding an authentication mechanism, by controlling the access to each page according with the type of user that is logged in, and by adding support to database storage. For some questions there is a video with the expected result. So, before starting each question please watch if a corresponding video is available.

1. Download from Moodle the "ws5.base.zip" file and extract it to the **ficha5** path. The folder structure is the complete working solution for the previous worksheet.

2. Because the zip file doesn't contain the vendor package, you must run composer install before proceeding. From command-line (suggestion: git bash), go to the **~/Homestead** folder and type:

   > vagrant ssh

   Inside the guest machine, go to the **ficha5** folder and run composer:

   > cd c:/ainet/pratica/ficha5

   > composer install

Storage

3. Download adminer and put it on the adminer folder;

4. Next, download from Moodle the file "ainet.ws5.sql". Go to "http://adminer.test" and login using the credentials "homestead" (username) and "secret" (password). Click on import and select the file "ainet.ws5.sql". The script will create a database called "ainet_p5". The database contains the table "users" filled with a single record that corresponds to an administrator profile with email ainet@ipleiria.pt and password 123123123;

5. Change the **User** model class by replacing the CRUD related methods with the corresponding database code. Use PDO to access the underlying database;

6. Change the **add** method on the **UserController** class by adding code that generates a password hash for the user from the user provided password. To do that you must call the **password_hash** function (note: use the PASSWORD_DEFAULT algorithm);

7. Add to your application some sort of visual tip to notify users if the add, save or delete operation were executed successfully. Tip: use sessions ($_SESSION);

8. Looking back to the code you have developed check if you could have followed a more generic approach. Some suggestions:
   (a) Use an abstract base **Model** class to capture code that could be reused across models;

(b) Use a db.php file that should define a database_config() function that returns an associative array with the database configuration (host, user, pass and dbname). This file should be loaded by composer (add it to the section "files" inside the file "composer.json");
(d) So far, all methods in class User are static. Try to figure out which one(s) could be converted to instance method(s).