APLICAÇÕES PARA A INTERNET

Engenharia Informática

PHP Laravel

Objectives:

(1) Understand the structure of a Laravel application;

(2) Be able to create routes and controllers in a Laravel application;

(3) Be able to create views using Blade templating;

(4) Understand Laravel's authentication facilities;

(5) Understand Laravel's authorization facilities;

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];

Setting up the environment:

- If the Vagrant Homestead environment is not installed and configured, follow the instructions of worksheet 1.
- Run “vagrant up”
- Inside the guest machine, go to the c:/ainet/pratica folder (or your usual exercises folder) and run composer to create a new Laravel project (this will create the laravel project inside ficha6 folder):

```
> cd c:/ainet/pratica
> composer create-project laravel/laravel --prefer-dist ficha6
```

- Before proceeding type "**http://ainet.pratica.6.test/**" on the browser and you should see the default Laravel 5 welcome page

In this worksheet you will redo the previous worksheet but this time using Laravel. Part I of the worksheet is mandatory and part II is optional. Part I explores core concepts of Laravel that you should master before starting the project while Part II explores advanced concepts that requires some knowledge on the framework's lifecycle.

In Part I you should ignore all code related to authentication and authorization. Just assume that all routes are public and all operations are permitted.

Part I - Core Concepts

1. Before starting, edit the `.env` file located at the project's root folder and change the database's name to **ainet_p6** (`DB_DATABASE= ainet_p6`). If the .env file does not exist, copy the .env.example to .env. After that you need to type:

   ```
   > cd ~/Homestead
   > vagrant ssh
   vagrant@homestead:~$ cd ainet/pratica/ficha6
   vagrant@homestead:~/ainet/pratica/ficha6$ php artisan key:generate
   ```

2. Add to the current user's table migration the code required to add the column 'type' that stores the user's type (0 – administrator, 1 – publisher, 2 – client).
   Detailed steps:

   ```
   vagrant@homestead:~/ainet/pratica/ficha6$ php artisan make:migration add_type_to_users_table --table=users
   ```

   Then, update the “add_type_to_users_table” migration file to add the “type” column that should have 3 options: 0, 1 and 2.
   (https://laravel.com/docs/5.7/migrations).

3. Run the pending migrations by invoking "**php artisan migrate**" from the guest machine:

   ```
   vagrant@homestead:~/ainet/pratica/ficha6$ php artisan migrate
   ```

4. Use adminer to confirm that the “users” table in “ainet_p6” has the previously created “type” column.

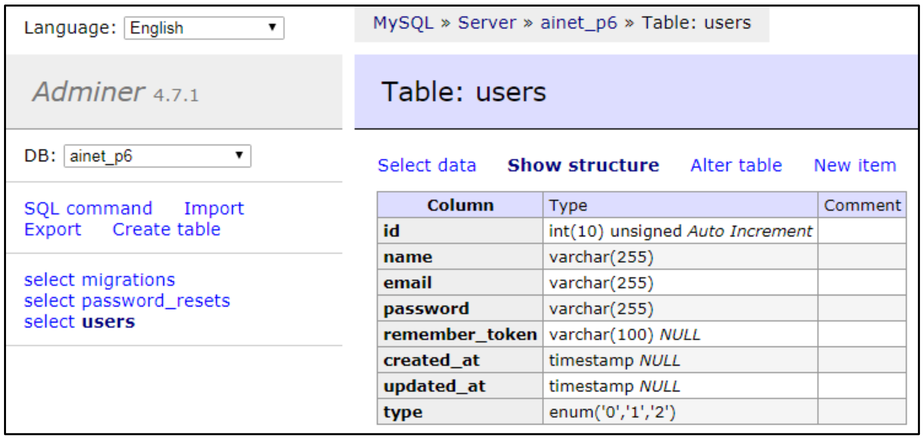

Table: users

Column	Type	Comment
id	int(10) unsigned *Auto Increment*	
name	varchar(255)	
email	varchar(255)	
password	varchar(255)	
remember_token	varchar(100) *NULL*	
created_at	timestamp *NULL*	
updated_at	timestamp *NULL*	
type	enum('0','1','2')	

5. Create an empty **UserController** controller using artisan. After running the command confirm that the app/Http/Controllers/UserController.php file was created. Open the file and analyze its contents.

   vagrant@homestead:~/ainet/pratica/ficha6$ **php artisan make:controller UserController**

6. Edit the routes/web.php file and add the following routes:
   - /users – should list all users (GET);
   - /users/create – should handle the creation of new users. Two routes are required: one for rendering the form (GET) and another for posting (POST);
   - /users/{id}/edit – should handle the edition of existing users. Two routes are required: one for rendering the form (GET) and another for posting (PUT);
   - /users/{id} – should handle the deletion of existing users (DELETE).

   All routes should point to methods on controller **UserController.**

   (https://laravel.com/docs/5.7/routing)

7. Implement the code required to render the route **/users**. Copy the views from the previous worksheet, change their name and update the content with Blade's directives. You should also create a layout file called 'master.blade.php' for the master layout. Note that "fullname" is now "name" and "registered_at" is "created_at".

   (https://laravel.com/docs/5.7/blade)

   For testing purposes, you can create some users from the command line using the **tinker** artisan command. Detailed steps for creating 10 users on the database:

   vagrant@homestead:~/ainet/ws6$ **php artisan tinker**

   >>> factory(App\User::class, 10)->create();

8. Next you will implement all the code required to implement the route **/users/create**. Replace the validation code from the previous worksheet with the Laravel's builtin `validate` method. Copy the views from the previous worksheet, change their name and update the content with Blade's directives.
   (https://laravel.com/docs/5.7/validation)

9. At this point you should be able to implement the remaining routes (edit and delete an existing user) without any additional assistance.

10. To finish Part I add to your **UserController** and views the code required to show a flash message informing the user if the operation (create, edit or delete) was executed successfully or not.
    (https://laravel.com/docs/5.7 /redirects#redirecting-with-flashed-session-data)

Part II – Advanced concepts

11. Use the **"make:auth"** artisan command to scaffold all the code required to enable Laravel's builtin authentication workflow.

12. Add the required middleware to your **/users** routes to allow only authenticated users to access them. Don't forget to add the logout button and welcome message.

13. In the **UserController** replace the builtin "validate" calls with a FormRequest class. You can create one for the Create user form by using the following artisan command:

```
vagrant@homestead:~/ainet/pratica/ficha6$ php artisan make:request StoreUserRequest
```

14. In the **UserController** replace all the {id} parameters with route model binding.

15. Add the required code to enforce the authorization rules so that:
    - Administrators can execute all operations;
    - Publishers can list and edit users;
    - Clients can list users and only edit their own record.

16. Add support to create/update/delete a user's profile image.