

LAB 10 – Programação com o framework Laravel [parte 2]

O objectivo deste laboratório é repetir a funcionalidade do site desenhado nos labs 5, 6, e 7 mas agora construído com o framework **Laravel** e a *template engine* **Blade**.

Assume-se aqui que já realizou com sucesso o LAB9.

1. Construa o template `register_template.blade.php` adaptando o template `register_template.tpl` realizado no LAB5

home login

Sign up

Name

Email

Password

Password confirmation

Go Clear

© 2016 Desenvolvimento de Aplicações Web Designed by [Aluno](#)

O template deve ser colocado na pasta
`public_html/LAB9_10/resources/views`

O código PHP do controlador responsável pela página de registo deverá encontrar-se na função `register()` e `register_action()` dentro do ficheiro que define a classe do controlador
(`public_html/LAB9_10/app/Http/Controllers/Blog.php`)

```
public function register()
{
}

public function register_action()
{
}
```

```
}
```

Adicione o código necessário para:

- O controlador da acção valida os dados introduzidos. Em caso de insucesso o formulario é retornado com os campos correctos já preenchidos
- Em caso de sucesso o controlador utiliza a função `register_user($username,$email,$password)` para actualizar a base de dados
- O controlador carrega o template “message_template” em caso de sucesso no registo (sugere-se a mensagem “Success: New user registered”)

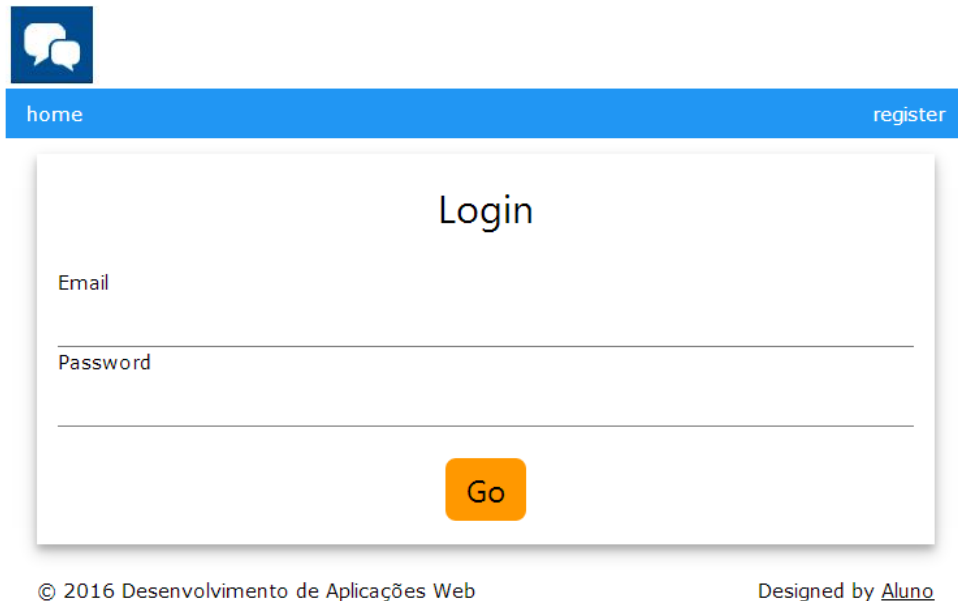
2. O controlador trabalha em colaboração com a classe responsavel pelo acesso à base de dados (public_html/LAB9_10/app/ Blog_model.php).

Construa a função `register_user($username,$email,$password)` responsável pela query à base de dados

```
public static function register_user($username,$email,$password)
{

}
}
```

3. Utilizando o template `login_template.blade.php` pretende-se construir a página de login no site. Adapte o template `login_template.tpl` realizado no LAB6.

A mockup of a web application's login page. At the top left is a blue square icon with two white speech bubbles. Below it is a blue horizontal bar with the word 'home' on the left and 'register' on the right. The main content area is a white box with a light gray border. Inside, the word 'Login' is centered at the top. Below it are two input fields: the first is labeled 'Email' and the second is labeled 'Password'. At the bottom center of the white box is an orange button with the word 'Go' in white. At the bottom of the entire page, there is a footer with two lines of text: '© 2016 Desenvolvimento de Aplicações Web' on the left and 'Designed by Aluno' on the right.

O código PHP do controlador responsavel pela página de login deverá encontrar-se na função login()

```
public function login()
{
}
```

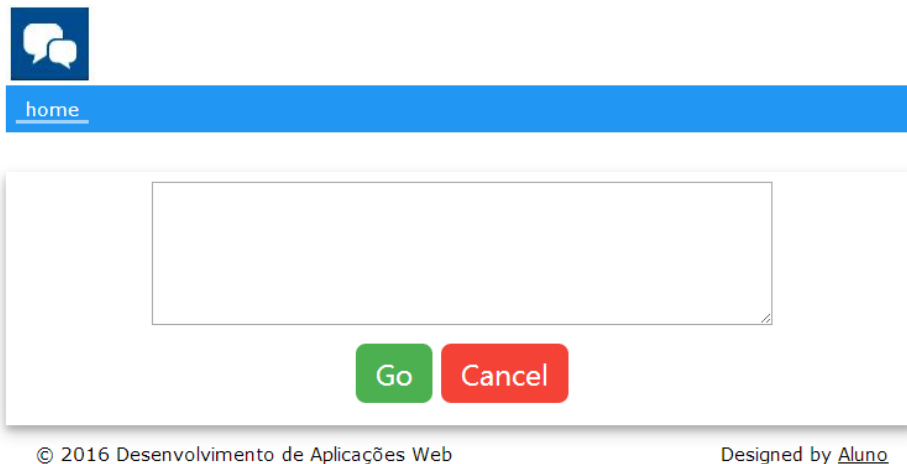
4. O código PHP do controlador responsavel pela acção de login deverá encontrar-se na função login_action()

```
public function login_action()
{
}
```

Adicione o código necessário para:

- Em caso de sucesso guardar no array de sessão o id, nome e email do utilizador validado
- Em caso de sucesso carregar o template 'message_template' (sugere-se "Welcome back!")
- Em caso de insucesso carregar novamente o template 'login_template' com a mensagem de erro "Login failed"

5. Utilizando o template `blog_template.blade.php` pretende-se construir a página de criar/alterar blog no site. Adapte o template `blog_template.tpl` realizado no LAB7.



O código PHP do controlador responsável pela página de registo deverá encontrar-se na função `post($blog_id)` e o controlador da acção na função `post_action($blog_id = FALSE)`

```
public function post($blog_id = FALSE)
{
}

public function post_action($blog_id = FALSE)
{
}
```

Adicione o código necessário para:

- O controlador utiliza a função `new_blog($user_id, $blog)` para inserir na base de dados o novo post
- O controlador utiliza as funções `get_blog($user_id, $blog_id)` e `update_blog($user_id, $blog_id, $content)` para actualizar a base de dados com o texto actualizado do post
- O controlador redirecciona para a pagina de rosto

NOTA: o seu código deve ser robusto contra tentativas de inserir posts em nome de outro utilizador, ou de alterar posts que não pertencem ao utilizador!

6. Construa as funções `new_blog()`, `get_blog()`, `update_blog()`, responsáveis pelas queries correspondentes à base de dados em “`app/Blog_model.php`”

7. Construa a função que destrói a sessão quando o utilizador faz logout. O controlador carrega o template 'message_template' (sugere-se “See you back soon!”), e que ao fim de 5 segundos redirecciona para a página de rosto)

```
function logout()
{
    session(['id' => '']);
    Cookie::queue('siteAuth', '');
    $message='See you back soon!';
    return view('message_template', compact('message'));
}
```

8. Actualize a função login_action() para, em caso de sucesso no login e a “checkbox” “remember_me” tiver sido seleccionada, implementar a funcionalidade enviando um cookie para o browser com os dados

```
$cookie_name = 'siteAuth';
$cookie_time = (60 * 24 * 30); // 30 days
$remember_digest = substr(md5(time()),0,32);
```

e actualizando o campo “remember_digest” na tabela “users” através da função set_remember_digest() em “app/Blog_model.php”

```
public static function set_remember_digest($email,$remember_digest)
{
}
```

Actualize a função index() para, se receber um cookie “siteAuth” verificar se o seu valor existe na tabela “users” e em caso de sucesso validar imediatamente o utilizador.

A função que verifica se o conteúdo do cookie existe na tabela “users” é

```
public static function check_remember_digest($remember_digest)
{
}
```

em “app/Blog_model.php”

9. Actualize o ficheiro “routes/web.php”:

```
Route::get('/blog/register', 'Blog@register' );
Route::post('/blog/register_action', 'Blog@register_action' );

Route::get('/blog/login', 'Blog@login' );
```

```
Route::post('/blog/login_action', 'Blog@login_action' );
Route::get('/blog/logout', 'Blog@logout' );

Route::get('/blog/post', 'Blog@post' );
Route::get('/blog/post/{blog_id}', 'Blog@post' );
Route::post('/blog/post_action', 'Blog@post_action' );
Route::post('/blog/post_action/{blog_id}', 'Blog@post_action' );
```

10. Teste o funcionamento do site no URL *

http://all.deei.fct.ualg.pt/~a12345/LAB9_10/index.php/blog

Considere o lab concluído quando obtiver a mesma funcionalidade que foi requerida nos LAB5, LAB6, e LAB7.

REFERÊNCIAS:

- http://all.deei.fct.ualg.pt/~a999990/LV_exame2/blog
- <https://laravel.com/docs/5.4>
- http://intranet.deei.fct.ualg.pt/DAW/slides/LV_overview.pdf
- <http://all.deei.fct.ualg.pt/laravel/>

* NOTA: o servidor http://intranet.deei.fct.ualg.pt/~a12345/LAB9_10/index.php/blog NÃO corre código Laravel

ANEXO 1. Estrutura da base de dados

A estrutura da base de dados pode ser consultada em <http://all.deei.fct.ualg.pt/phpMyAdmin>

```
CREATE TABLE `users` (  
  `id` int(11) NOT NULL auto_increment,  
  `name` varchar(255) default NULL,  
  `email` varchar(255) default NULL,  
  `created_at` datetime NOT NULL,  
  `updated_at` datetime NOT NULL,  
  `password_digest` varchar(255) default NULL,  
  `remember_digest` varchar(255) default NULL,  
  `admin` tinyint(1) default NULL,  
  `activation_digest` varchar(255) default NULL,  
  `activated` tinyint(1) default NULL,  
  `activated_at` datetime default NULL,  
  `reset_digest` varchar(255) default NULL,  
  `reset_sent_at` datetime default NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `index_users_on_email` (`email`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
CREATE TABLE `microposts` (  
  `id` int(11) NOT NULL auto_increment,  
  `content` text,  
  `user_id` int(11) default NULL,  
  `created_at` datetime NOT NULL,  
  `updated_at` datetime NOT NULL,  
  PRIMARY KEY (`id`),  
  CONSTRAINT FOREIGN KEY (`user_id`) REFERENCES `users`  
  (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```