Laravel (continuação, parte 2: DB + forms)

Databases and migrations

Pré-requisitos: servidor de base de dados (xampp mysql server)

Criar uma nova base de dados no prog. Gestor de bases de dados (phpmyadmin)
OU na linha de comandos:
mysql –uroot
create database mysite;
use mysite;

.env file:
Editar o ficheiro **.env** na raiz do projecto com os dados da base de dados:

DB_DATABASE=mysite
DB_USERNAME=root
DB_PASSWORD=

O ficheiro ./config/database.php
lê as variáveis do ficheiro .env

Migrations

Sistema de controlo de versões da base de dados
Representa uma tabela ou mudança a fazer numa tabela.

Conceito do laravel: as tabelas da base de dados e alterações a estas tabelas são representadas como classes.
Isto permite definir rapidamente o schema de uma tabela sem aprender uma nova linguagem

Comando para efectuar as migrations **(ainda não executadas):**
php artisan migrate

NOTA: Caso retorne um erro relacionado com sql é preciso editar o ficheiro **app/providers/appServiceProvider**
E adicionar um import/use e a linha na função boot, como indicado abaixo)

```
use Illuminate\Support\Facades\Schema; //Import Schema

function boot()
{
    Schema::defaultStringLength(191); //Solved by increasing StringLength
}
```

As migrations ficam na pasta:
./database/migrations

Comando para reverter última migration:
php artisan migrate:rollback

Para efectuar mudanças nas últimas migrations executadas, usar os comandos
php artisan migrate:rollback & php artisan migrate

php artisan migrate:fresh (apaga todas as tabela e cria de novo, **nunca correr em produção! Apenas localmente**)

Cofinanciado por:

php artisan help make:migration
Comando para criar uma nova migration (para criar tabela):
php artisan make:migration create_projects_table
Ao usarmos create no nome da migration é um atalho para esta criar uma tabela

Os métodos base e mais comuns das migrations são: up() que é executado ao correr a migration (criação é o mais comum) e o método down() que é executado no rollback e normalmente é drop (eliminação). No down() pode ser definido para não apagar (comentando).

Documentação: https://laravel.com/docs/6.x/migrations

Eloquent, Namespacing, and MVC

Eloquent Models

Laravel active record implementation.
Um model é a forma singular da tabela, representa uma entrada da tabela.

Criar um model:

```
php artisan make:model Project
```

este comando vai criar o ficheiro
/app/project.php

Isto dá-nos poder de manipulação sobre a tabela, para demonstrar vamos usar a consola:

php artisan tinker (consola)

```
>>> 2+2
App\Project::all();
App\Project::first();
App\Project::latest()->first(); // order inversa e depois primeiro

$project = new App\Project;
$project->title = 'My first project';
$project->description = 'Lorem ipsum';
$project->save();
```

Cofinanciado por:
MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

```
App\Project::first();
App\Project::first()->title;
App\Project::first()->description;

App\Project::all();

$project = new App\Project;
$project->title = 'My second project';
$project->description = 'Lorem ipsum';
$project->save();
```

Collections: arrays on steroids

```
App\Project::all()[0];
App\Project::all()[1];

App\Project::all()[1]->title;
App\Project::all()->map->title; // returns a collection of only titles of all
```

Os models completam o MVC, já vimos as views, controllers e agora os models.

Namespaces (nos controllers)

Permitem o autoload em vez de imports e evita conflitos de nomes.

```
$projects = \App\Project::all(); # \ para começar na raiz
```

Mas é preferível importar com **use**:

```
use App\Project;
…
        $projects = Project::all();
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Directory Structure review

Ficheiros da raíz do projecto

.editorconfig: opções para o editor de código, pode ser apagado
.env: variáveis de configuração. Ignorado de controlo de versões: um ficheiro para desenvolvimento local e outro diferente para produção, pois as variáveis de configuração são diferentes
.gitignore: especifica ficheiros e pastas a ignorar do repositório git
composer.json: lista os packages do gestor de dependências
package.json: para desenvolvimento do front end e suas dependências
webpack.mix.js: para facilitar front end (SASS), veremos melhor mais à frente

Pastas

Vendor: onde as composer dependencies são instaladas
Storage: logs, cache, ficheiros compilados. Não é importante na maior parte dos casos
Routes: web ou outros casos como comandos, api, channels (broadcast de server para client)
Resources: views, js, sass
Public: **imagens; js, css** compilados a partir dos ficheiros na pasta **resources,** usando o **mix (webpack.mix.js)**
Database: migrations, factories (para criarmos **dados aleatórios** para testar ou preencher models da BD)
Config: configurações lidas do ficheiro .env (local e production)
App: models, controllers, middleware (camadas internas com código que é processado em certos casos)

- Providers (services providers): classes que dão instruções como ligar componentes ao laravel/app (componentes do laravel: laravel/src/Illuminate). Se criarmos um novo package, criamos o seu provider para definir como liga ao laravel

Cofinanciado por:
MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Form Handling and CSRF Protection

Processo de um form:

- Access created route (/projects/create) that loads controller that simply loads a view (displays a form)
- Submit form **post** request to
- Route store (/projects/store) that calls
- Store method in Controller that saves the data and redirects

csrf_field: Cross-site request forgery, é exemplo de middleware (http/kernel.php). se não tivesse o middleware não obrigava a ter o csrf mas é recomendado para segurança. Verifica se token do form é igual ao da sessão

Routing Conventions Worth Following

```
/*
   GET /projects (index) # show list
   GET /projects/create (create)
   POST /projects (store)
   GET /projects/1 (show)
   GET /projects/1/edit (edit)
   PATCH /projects/1 (update)
   #PUT /projects () # replacing a resource in it's entirety
   DELETE /projects/1 (destroy)
*/
```

Available Router Methods

The router allows you to register routes that respond to any HTTP verb:

```
Route::get($uri, $callback);

Route::post($uri, $callback);

Route::put($uri, $callback);

Route::patch($uri, $callback);

Route::delete($uri, $callback);
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

```
Route::options($uri, $callback);
```

Route Parameters

Required Parameters

```
Route::get('user/{id}', function ($id) {
    return 'User '.$id;
});
```

You may define as many route parameters as required by your route:

```
Route::get('posts/{post}/comments/{comment}', function ($postId, $commentId) {
    //
});
```

Documentação sobre routes: https://laravel.com/docs/6.x/routing

Mini Exercício: definir routes para todos os tipos (conventions/standards)

Comando para listar todas as routes registadas:
php artisan route:list

Atalhos

Para criar routes seguindo standards (RESTful standards):

```
Route::resource('projects', 'ProjectsController');
```

Criar controller com código boilerplate para resource

```
php artisan make:controller –r PostsController
```

Se não funcionar tentar com –r no final

Apagar manualmente o ficheiro do controller anterior (criado automaticamente)
Criar model e importar no controller

```
php artisan make:controller PostsController –r –m Post
```

Faking PATCH and DELETE Requests

Apoio para parte práctica

Link para usar Bulma css framework (copiar para o head da view layout.blade.php):

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bulma/0.7.2/css/bulma.min.css">
```

url para edit: http://127.0.0.1:8000/projects/1/edit

html do form edit

```
    <form method="">
  <div class="field">
    <label  class="label" for="title">Title</label>

    <div class="control">
      <input type="text" class="input" name="title" placeholder="Title">
    </div>

  </div>

  <div class="field">
    <label  class="label" for="description">Description</label>

    <div class="control">
      <textarea class="textarea" name="description"></textarea>
    </div>

  </div>

  <div class="field">

    <div class="control">
      <button type="submit" class="button is-link">Update project</button>
    </div>

  </div>

</form>
```

Cofinanciado por:

MADEIRA 14-20 · REGIÃO AUTÓNOMA DA MADEIRA · PORTUGAL 2020 · UNIÃO EUROPEIA Fundo Social Europeu

O browser não suporta o método **patch**, é preciso contornar, tem de ter **post** e adicionar um campo extra ao request:

```
<form method="POST" action="/projects/{{ $project->id }}">
    {{ method_field('PATCH') }}
```

Form DELETE Requests

Exercício: adicionar botão delete para apagar a entrada. Dica: tem de ser 2 forms (1 para cada ação)

Atalhos

```
<form method="POST" action="/projects/{{ $project->id }}">
    @method('DELETE')
    @csrf
```

Cleaner Controllers and Mass Assignment Concerns

```
public function showAlt(Project $project)
{
        return view('projects.show', compact('project'));  # ['project' => $project]
}
```

Mass Assignment

```
public function store()
{
        Project::create([
         'title' => request('title'),
         'description' => request('description')
    ]);
}
```

Cofinanciado por:

MADEIRA 14-20 · REGIÃO AUTÓNOMA DA MADEIRA · PORTUGAL 2020 · UNIÃO EUROPEIA Fundo Social Europeu

Por questões de segurança, ao usar o mass assignment é preciso definir, no respectivo model, os campos que podem ser preenchidos ou as excepções que não pode ser preenchidas.

```
protected $fillable = [
'title', 'description'
];
```

OU

```
protected $guarded = []; # colocar campos com excepcoes (nao podem ser preenchidos)
```

No update/edit:

```
$project->update(request(['title', 'description']));
```

Estes **atalhos**, sem definir separadamente as variáveis do request (vindas da view) e os campos da base de dados/migrations/model, **só funcionam quando os nomes são iguais.**

Fontes e mais recursos

https://laravel.com/
https://laracasts.com/series
https://laravel.com/docs
https://laracasts.com/skills/laravel

Cofinanciado por: