Conceptes bàsics de Laravel

Conceptes bàsics	Passos a seguir per un projecte bàsic
1- Namespace i MVC 2- Migrations i Taules 3- Bastiments 4- Models i Controllers 5- Views i Controllers 6- Blades i Blade Templates 7- Routing 8- Request & Responses 9- Middleware 10- Seeders i Factories	1- Creació de base de dades, usuari i permisos amb un script sql 2- Creació d'un projecte amb l'ordre laravel 3- Configuració del projecte modificant l'arxiu .env 4- Creació de taules del projecte: • Creció dels fitxers de Migrations • Execució de les Migrations => Creació taules 5- [opcional] Afegiment de Middleware d'autenticació 6-Desenvolupament dels Models 7- Desenvolupament dels Controllers 8- Creació de Routes 9- Creació de Views (blades)

Treballant amb Models

1- En el **patró de programació MVC** (Model-Vista-Controlador), el **Model** és la part de l'aplicació responsable de les interaccions amb les taules de les bases de dades i per tant, és la part encarregada de fer accions típiques d'INSERT, SELECT, UPDATE, DELETE sobre les taules. El **Model** mai s'encarrega d'accions com interactuar amb l'usuari o acceptar mètodes HTTP GET, POST, PUT o DELETE.

2- Laravel:

- Inclou un ORM (Object-Relational Mapper que és una capa de software intermitja entre el codi PHP i les ordres SQL que actuen sobre la base de dades). Aquest ORM s'anomena Eloquent ORM.
- Eloquent ORM defineix una classe abstracta anomenada **Model** a partir de la qual, per herència, es poden crear noves classes que tinguin tots els atributs i mètodes necessàris per fer INSERT, SELECT, UPDATE i DELETE sobre les taules de la base de dades.
- Amb Eloquent ORM, totes i cadascuna de les taules de la base de dades de l'aplicació han d'estar associades al seu propi model, el qual ha de tenir una classe pròpia que hereta de **Model** i que tindrà tots els atributs i mètodes necessàris per fer INSERT, SELECT, UPDATE i DELETE sobre aquella taula en concret.
- Per tant, amb Eloquent ORM, per cada taula hem de crear un model propi que dins seu defineix una classe pròpia necessària per interactuar amb la taula.

3- Respecte del nom de taules i models Laravel utilitza les següents convencions per defecte que són importants a tenir en compte quan es desenvolupa l'aplicació:

- Les taules s'escriuen en plural i totes les seves lletres estan en minúscula. Un exemple seria la taula **treballadors** que vam crear a la tercera sessió (*pj6f4a14.3 – PART I – punt c*).
- La classes tenen el mateix nom que la taula però la primera lletra s'escriu amb majúscules i s'escriu en singular. Així doncs, el nom de la classe del model associat a la taula treballador hauria de ser **Treballador.**
- El fitxers PHP amb els models de les taules es desen a la carpeta **app/Models** de l'aplicació.
- El fitxers PHP amb els models tenen el mateix nom que la classe, de manera que per la taula **treballadors** és crearà una classe de nom **Treballador** que hereta de **Model**, i un fitxer **app/Model/Treballador.php**.

4- Laravel anomena també **Model** a un registre d'una taula d'una base de dades. Per tan hem de diferenciar quan parlem de les classes de tipus Model i d'un model, o sigui una entrada d'una taula.

5- Dins del projecte **empresa**, podem crear el fitxer del model amb la classe **Treballador** necessari per interactuar amb la taula **treballadors** i un altre de nom **Gestor** per interactuar amb una nova taula de nom **gestors** amb dades sobre el personal de direcció i gestió de l'empresa.

Seguint les convencions de Laravel, i tenint en compte que la taula **treballadors** ja existeix, per crear el model **Treballador** hem d'anar a la carpeta **empresa** i executar:

```
php artisan make:model Treballador
```

Ara bé, com que la taula direccions no existeix, podem crear el model **Gestor** i la taula **gestors** al mateix temps executant:

```
php artisan make:model Gestor --migration
```

6- Comprova que s'han creat dins de la carpeta del projecte **empresa** els fitxers de model **app/Models/Treballador.php** i **app/Models/Gestor.php** que dins tenen les classes **Treballador** i **Gestor** que hereten de la classe pare **Model**.

7- Comprova que s'ha creat el fitxer de migrations **aaaa_mm_dd_create_gestors_table.php** dins de **database/migrations**. Fes que l'estructura de la taula sigui aquesta:

```
$table->id('gid');
$table->string('nom');
$table->string('cognoms');
$table->integer('tlf_mobil');
$table->string('email');
$table->timestamps();
```

8- A continuació executa:

```
php artisan migrate
```

i comprova que s'ha creat la taula de gestor dins de la base de dades **Empresa**.

9- Laravel assumeix per defecte que la taula té una clau primària de nom **id**. En el cas de que aquest no sigui el cas, s'haurà de canviar el nom de la clau. Hi ha un exemple de com fer aquest canvi aquí. És interessant llegir Eloquent Model Conventions per coneixer altres assumpcions que fa Laravel per defecte.

10- Els models tenen una sèrie d'atributs que es poden utilitzar per modificar el seu comportament per defecte. A la secció https://laravel.com/api/10.x/Illuminate/Database/Eloquent/Model.html pots trobar alguns d'aquest atributs i el seu significats. Alguns atributs interessants són:

- **$table** → Per canviar el nom de la taula associada al model
- **$primaryKey** → Per canviar el nom de la clau primària
- **$incrementing** → Per indicar si la clau primària és autoincremental(true) o no (false)
- **$keyType** → Tipus de la clau primària. Per exemple 'string' si és una cadena de caràcters.
- **$fillable** → És un array amb la llista dels camps de la taula que es poden omplir a partir d'un array amb dades normalment rebut des d'un del formulari. En Laravel aquest procés és anomenat Mass assignment. Els camps que no es trobin dins la llista no haurien d'estar disponibles en el formulari per l'usuari
- **$guarded** → Camps que NO poden ser omplert via Mass assignment. Tots aquells camps que NO es trobin aquí dins, per defecte es poden omplir via Mass assignment, és a dir, poden estar disponibles en el formulari per l'usuari.

10- Així doncs, d'una manera molt bàsica, si per exemple volem que només els camps **nom**, **cognoms** i **nif** de la taula **treballadors** es puguin omplir amb les dades enviades des d'un formulari, i si volem que la clau primària sigui **nif** (que és un string i no és autoincremental) hauríem de modificar la nostra classe **Treballador** des del fitxer **app/Models/Treballador.php** i escrivint:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Treballador extends Model
{
	use HasFactory;
	protected $primaryKey = 'nif';
	public $incrementing = false;
	protected $keyType = 'string';
	protected $fillable = [
		'nom',
		'cognoms',
		'nif'
	];
}
```

Ara bé, en el nostre cas:

- La **clau primària** de la taula **treballadors** és el camp **tid**, que és és **integer** i **autoincremental**.
- Els camps a omplir són: **nom**, **cognoms**, **nif**, **data_naixement**, **sexe**, **adressa**, **tlf_fixe**, **tlf_mobil**, **email**, **fotografia**, **treball_distancia**, **tipus_contracte**, **data_contractacio**, **categoria**, **nom_feina** i **sou**.

Per tant, un exemple de com podria ser la nostra classe **Treballador** dins del fitxer **app/Models/Treballador.php** seria la següent:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Treballador extends Model
{
	use HasFactory;
	protected $primaryKey = 'tid';
	protected $fillable = [
		'nom','cognoms','nif','data_naixement','sexe','adressa',
		'tlf_fixe','tlf_mobil','email','fotografia','treball_distancia',
		'tipus_contracte','data_contractacio','categoria','nom_feina','sou'
	];
}
```

11- Alguns mètodes interessants de la classe **Treballador** serien:

- **Treballador::all()** → Fa un **SELECT** de tots els registres de la taula **treballadors**.
- **Treballador::create($dades)** → Fa un **INSERT** i crea un nou registre en la taula **treballadors**.
- **Treballador::findOrFail($tid)** → Busca un registre a partir del valor de la clau primària **$tid**. Si el troba assigna el contingut del registre a una variable i si no el troba llença una excepció que pot ser capturada per mostrar un missatge de resposta del tipus **404**.
- **Treballador::where('nif', '=','10243876R')→first()** → Troba un registre. En aquest cas seria el registre amb el camp **nif** igual a **10243876R**. El resultat es pot assignar a una variable. Correspon a un **SELECT**.
- **Treballador::where('nif', '=','10243876R')→update($dades)** → Troba un registre i l'actualitza amb **$dades**. En aquest cas seria el registre amb el camp **nif** igual a **10243876R**. Correspon a fer un **UPDATE** a la base de dades.
- **Treballador::whereId($tid)→update($dades)** → Troba un registre amb la clau primària de valor **$tid** i l'actualitza amb **$dades**. Correspon també a fer un **UPDATE** a la base de dades. La variable **$dades** ha de ser un array associatiu amb els camps de la taula com a índex i els nous valors associats.
- També es pot fer un **UPDATE** així:
  $treballador=Treballador::where('nif','=','10243876R');
  $treballador→nif='98765432B';
  $treballador→save( );
- **Treballador::where('sou','=','2000')→update(['sou' => '2050']);** → Fa una actualització de multiples registres amb el nou valor de sou 2050 a partir dels registres que segueixen el criteri que el sou és 2000.
- **Treballador::findOrFail($tid)→delete( );** → Troba un registre amb la clau primària de valor **$tid** i l'esborra. Si no el troba llença una excepció. Correspon a fer un **DELETE** a la base de dades.
- **Treballador::where('nif', '=','10243876R')→delete( );**→ Troba un registre amb el nif indicat i l'esborra. Correspon també a fer un **DELETE** a la base de dades.

Enllaços interessants sobre Models

- https://laravel.com/docs/12.x/eloquent
- https://laravel.com/docs/12.x/eloquent#generating-model-classes
- https://laravel.com/docs/12.x/eloquent#table-names
- https://www.parthpatel.net/laravel-tutorial-for-beginner/
- https://www.educba.com/laravel-models/
- https://laravel-guide.readthedocs.io/en/latest/eloquent/#retrieving-models