Conceptes bàsics de Laravel

0- Namespace
1-Models i Migrations
2-Controllers
3-Views
4- Blades i Blade Templates
5-Routing
6- Request & Responses
7-Middleware

Treballant amb git, migrations i afegint el bastiment Laravel Breeze

PART 1 - Treballant amb Git

1- Comprova que dins del directoridel projecte **empresa**, durant el procés d'instal·lació del framework de Laravel s'ha creat un fitxer **.gitignore** i que el fitxer **.env** i el directori vendor estan inclosos dins de la llista dels fitxers que no es trobaran en seguiment localment i per tant, tampoc no es pujaran a **Github**.

2- Comprova també que s'ha dins del directori creat un directori **.git** amb un dipòsit local.

3- Fes un un **add** del projecte i un **commit** amb el missatge *"Commit 1 del projecte empresa"*. Si et demana configurar git, recorda que:

- Utilitza el mateix mail que el que vas utilitzar per crear el teu compte de Github
- Utilitza el mateix nom d'usuari que el del teu teu compte de Github

4- Crea un dipòsit **privat** de **Github** de nom **empresa**. Sincronitza el dipòsit local amb el dipòsit remot. Puja el codi del teu projecte a **Github**. Comprova que el codi ha estat pujat al teu dipòsit remot.

PART 2 - Treballant amb migrations

a) Creant taules

1- L'eina **artisan** s'inclou amb cada instal·lació del framework de **Laravel** i necessita l'interpret de **php** per ser interpretada i executada. Aquesta eina es troba sempre a l'arrel del projecte i permet fàcilment, entre altres opcions, crear **taules, models**, **controladors** i **vistes**. L'ordre **artisan** es trobarà dins del directori **empresa** que és on s'ha instal·lat el framework de **Laravel**.

2- Les **migrations** de **Laravel:**

- Permeten crear, esborrar, reanomenar o canviar l'estructura de les taules d'una base de dades sense haver d'executar ordres de MySQL entrant a la base de dades o crear un script amb ordres SQL per fer aquesta feina.
- Les Migrations es defineixen dins de fitxers que es troben al subdirectori **database/migrations**. Per defecte ja existeixen unes fitxers de migrations bàsics (failed_jobs, password_reset_tokens, personal_access_token i users) definits que es van crear en el moment d'afegir al nostre projecte el frameworl de **laravel**.
- Haurem d'utilitza l'eina **artisan** per executar les migrations que s'aniran definint dins del projecte. Per exemple, per crear taules dins de la base de dades empresa amb les migration per defecte, hem d'anar al directori empresa i executar l'ordre:

  ```
  php artisan migrate
  ```

  i veurem (s'ha de comprovar) que es crean les taules: **failed_jobs**, **password_reset_tokens**, **personal_access_tokens** i **users** dins de la base de dades **empresa**.
- En el moment de fer la primera migration, es crearà a la base de dades empresa, una taula amb el nom **migrations** a on deserà els **logs** de les tasques de creació/modificació/esborrament de taules dins de la base de dades empresa.

3- De totes les taules creades, és especialment important parar atenció a la taula **users** perquè és a on definirem els nostres usuaris de l'aplicació. Dins d'aquesta taula podem trobar, els camps per l'dentificador de l'usuari (que és primary key), el nom d'usuari, correu electrònic, contrasenya, etc..

4- Anem a crear una taula per desar les dades dels empleats de l'empresa. Ens caldrà per tant afegir un fitxer dins del directori de **migrations** per poder crear la taula que anomenarem **empleats**. En aquest cas, després d'analitzar les necessitats de l'empres i la nostra futura aplicació, hem arribat a la conclusió que ens calen els següent camps per la nova taula: **id** (identificador), **nom** de tipus **text**, **telefon** de tipus **text**, **email** de tipus **text** i els **timestamps** que ens informin dels accessos a la taula i els seus registres. Per tant, crearem un fitxer de **migrations** i per fer-ho, només ens cal executar dins del directori **empresa**:

```
php artisan make:migration create_empleats_table --create=empleats
```

i a continuació, comprovarem que s'ha creat el fitxer **2024_03_07_105517create_empleats_table.php** dins del directori **database/migrations**. Compte que **2024_03_07_105517** representa la data i hora de creació i per tant pot ser d'un valor diferent en funció del moment de la creació del fitxer.

5- Ara modificarem el contingut de fitxer **2024_03_07_105517_create_empleats_table.php** per adaptar-lo a les nostres necessitats. Fes que el codi sigui aquest:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateEmpleatsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('empleats', function (Blueprint $table) {
            $table->id();
            $table->string('nom');
            $table->string('email');
            $table->string('telefon');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('empleats');
    }
}
```

NOTA: Les línies en vermell són les nostres modificacions.

Quan executem aquest codi, es crearan la taula empleats (que s'esborrarà si ja existeix) amb els camps indicats. El camp **id** ja serà de tipus **primary key**, **bigint(20)**, **unsigned** i **auto_increment**.

6- A continuació, procedirem finalment a la creació de la taula **empleats**. Executarem des del directori empresa:

```
php artisan migrate
```

i la taula es crearà.

7- Si vols assegurar-te que la taula s'ha creat, entra dins de MySQL i executa:

```
use empresa;
show tables;
describe empleats;
```

i comprova que efectivament s'ha creat la taula, amb els camps demanats. Ara, ja sabem crear taules amb Laravel.

8- Consulta la taula **migrations** de la base de dades **empresa** executant:

```
select * from migrations;
```

i comprova que el resultat similar a:

```
MariaDB [empresa]> select * from migrations;
+----+-------------------------------------------------------+-------+
| id | migration                                             | batch |
+----+-------------------------------------------------------+-------+
|  1 | 2014_10_12_000000_create_users_table                  |     1 |
|  2 | 2014_10_12_100000_create_password_reset_tokens_table  |     1 |
|  3 | 2019_08_19_000000_create_failed_jobs_table            |     1 |
|  4 | 2019_12_14_000001_create_personal_access_tokens_table |     1 |
|  5 | 2024_03_07_105517_create_empleats_table               |     2 |
+----+-------------------------------------------------------+-------+
5 rows in set (0.001 sec)
```

Això és un **log** de les **migrations** realitzades fins ara i que a més a més, ens permet controlar la manera de fer **rollbacks**. Un **rollback** ens permet **revertir** la darrera, les N darreres o totes les **migrations** realitzades.

b) Fent rollbacks (revertint) d'una migration

1- Per comprovar que podem fer una tasca extremadament útil com és la de fer **rollbacks**, surt del client **mysql** i reverteix la darrera **migration** realitzada executant des de la carpeta **empresa**:

```
php artisan migrate:rollback
```

i torna a accedir al client MySQL. Comprova que la taula **empleats** ha desaparegut i que dins de la taula **migrations** ha desaparegut l'entrada indicat associada.

A continuació, si no vols que es torni a crear la taula **empleats** en la propera **migration**, entra a **empresa/database/migrations** i esborra el fitxer PHP **2024_03_07_create_empleats_table.php** (el valor inicial del nom del fitxer recorda que pot canviar en funció de l'hora de creació del fitxer).

Ara, ja poder revertir i i eliminar els efectes d'una **migration**.

c) Creació de taules i definint característiques dels camps

1- Anem a crear una taula més complexa i amb més camps que serna de diversos tipus. En comptes d'**empleats** la nova taula s'anomenarpa **treballadors**. Executa des de la carpeta **empresa**:

```
php artisan make:migration crea_taula_treballadors --create=treballadors
```

de manera que a **empresa/database/migrations** es crearà el fitxer: **2024_mm_dd_hhmmss_crea_taula_treballadors.php** (a on **mm** és el mes, **dd** és el dia i **hhmmss** és l'hora de creació del fixer).

2- A continuació, obre el fitxer PHP creat i fes que la estructura de camps de la taula treballadors sigui aquesta:

```
$table->id('tid');
$table->string('nom');
$table->string('cognoms');
$table->string('nif')->unique();
$table->date('data_naixement');
$table->char('sexe',1);
$table->string('adressa');
$table->integer('tlf_fixe');
$table->integer('tlf_mobil');
$table->string('email');
$table->binary('fotografia')->nullable();
$table->boolean('treball_distancia');
$table->enum('tipus_contracte',['temporal','indefinit','en formació','en pràctiques']);
$table->date('data_contractacio');
$table->tinyinteger('categoria');
$table->string('nom_feina',50);
$table->float('sou');
$table->timestamps();
```

NOTA:

- Tipus de columnes: https://laravel.com/docs/10.x/migrations#available-column-types
- Modificadors de columnes: https://laravel.com/docs/10.x/migrations#column-modifiers
- Tipus d'index: https://laravel.com/docs/10.x/migrations#available-index-types

Ara, des del directori empresa executa:

php artisan migrate

A continuació, entra dins de MySQL i executa:

use empresa;
show tables;
describe treballadors;

i comprova que s'ha creat la taula correctament i que la seva creació s'ha enregistrat dins de la taula de logs.

d) Modificació d'una taula creada

1- Si ara volem modificar un camp de la taula, per exemple, fent que el camp `'nom_feina'` tingui **30** caràcters en comptes de **50**, haurem de:

- Editar **2024_mm_dd_hhmmss_crea_taula_treballadors.php**
- Modificar el camp `'nom_feina' ==> $table→string('nom_feina',30);`

A continuació, executarem dins del directori **empresa**:

php artisan migrate:refresh

Després, amb l'ordre **describe treballadors;** dins de **MySQL** comprovarem que s'ha modificat el camp.

2- Així doncs, ara ja sabem:

- Fer migrations
- Revertir i eliminar migrations
- Treballar amb tipus de dades i els seus modificadors
- Modificar migrations

PART 3- Bastiment Laravel Breeze

1- Laravel té una sèrie de bastiments disponibles, fàcilment descarregables, que es poden afegir sense dificultats al projecte i que tenen com a objectiu afegir funcionalitats noves al projecte sense haver de fer aquesta feina des de zero.

2- Molts d'aquests bastiments fan servir l'entorn d'execució de javascript **Node.js** i per tant ens caldra **npm**, el gestor de paquets **Node.js,** per instal·lar-los. Si **npm** no ha estat prèviament intal·lat en el sistema, executa:

sudo apt-get update
sudo apt-get install npm

Pot durar un estona perquè descarrega uns **125MiB** de programari.

3- Un bastiment molt útil perquè té tot el codi necessari per proporcionar el servei de registre i autenticació d'usuaris per l'aplicació és **Laravel Breeze**. D'acord amb la documentació oficial: "*Laravel Breeze is a simple, minimal implementation of all of Laravel's authentication features, including login, registration, password reset, email verification, and password confirmation".*

*Aquest bastiment aprofitarà les taula users creades amb la primera **migration** realitzada per poder emmagatzemar usuaris i les seves contrasenyes. També afegirà al nostre projecte uns fitxers de vistes per poder fer feines típiques d'autenticació.*

Per instal·lar **Laravel Breeze** i les seves dependències dins del nostre projecte, només cal accedir al directori **empresa** i executar:

composer require laravel/breeze --dev

Tot i això, el bastiment d'autenticació no s'ha afegir al projecte. Només tenim el programari necessari per crear-lo.

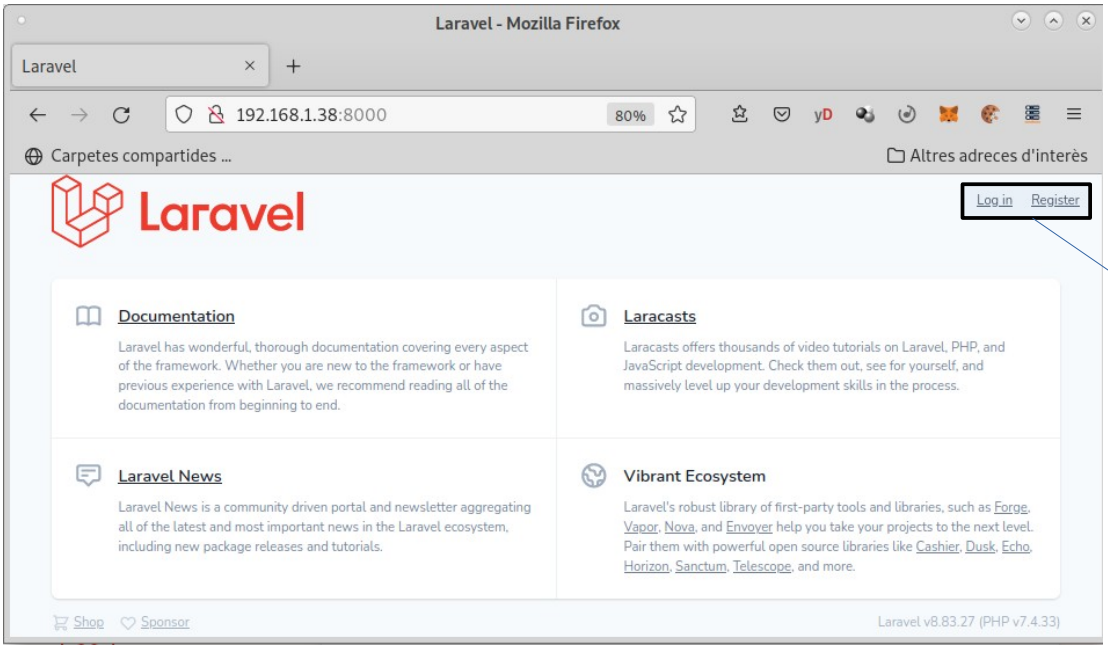
4- Ara hem d'afegir el bastiment d'autenticació al projecte. **Laravel Breeze** disposa de multiples bastments. El més bàsic és **Blade with Alpine** . Seleccionem aquest bastiment i deixem la resta d'opcions per defecte..Per iniciar **Laravel Breeze** amb **Blade** executa dins del directori **empresa**:

php artisan breeze:install

i es crearan les **vistes**, **controladors** i **rutes** necessàries per treballar amb autenticació utilitzant el bastiment seleccionat.

5- Comprova com està la nostra aplicació:

- Troba l'adreça IP de la teva màquina virtual gestionada amb vagrant. Executa: **ip a**
- Executa des del directori **empresa**: **php artisan serve --host=0.0.0.0 --port=8000**
- Des de la teva màquina física accedeix a l'adreça IP de la màquina virtual i utilitzant el port **8000** i veuràs alguna cosa similar a aquesta:



Noves opcions de Login i Registre ara disponibles.

6- Comprovacions:

- Accedeix a l'opció **Register** i comprova que pots enregistrar un nou usuari i que un cop registrat et porta al **Dashboard** de l'usuari.
- Comprova que pots fer un **logout** i tornes a la pàgina d'inici.
- Accedeix amb **Log in** i comprova que pots validar-te i accedeixes al **Dashboard** de l'usuari, i que també que pots tancar la sessió amb un **logout** i tornar a la pàgina inicial.
- Comprova que s'ha creat l'usuari a la taula **users** de la base de dades **empresa**. La contrasenya s'haurà desat en format **hash** i tindrem informació de la **data de creació**.

7- Finalment, personalitzarem la pàgina d'inici de l'aplicació. Per fer això, primer obrirem **routes/web.php** i canviarem l'**enrutament** de **/** perquè obri un fitxer de nom i**nici.blade.php** que es trobarà a **resources/views**. És molt fàcil, el codi de les línies 16 a 18 de **web.php** serà:

```
Route::get('/', function () {
	return view('inici');
});
```

Bàsicament, estem dient que enrutem **/** cap a **inici.blade.php**. No cal indicar l'extensió perquè la funció view() ja s'encarrega de buscar el fitxer dins de **resources/views**.

Després crearem el fitxer **inici.blade.php** dins de **resources/views** amb el següent codi:

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
	<head>
		<meta charset="utf-8">
		<meta name="viewport" content="width=device-width, initial-scale=1">
		<title>Empresa</title>
	</head>
	<body>
		<div>Pàgina inicial de l'aplicació web Empresa</div>
		@if (Route::has('login'))
			@auth
				<a href="{{ url('/dashboard') }}">Dashboard</a>
			@else
				<a href="{{ route('login') }}">Log in</a><br>
				@if (Route::has('register'))
					<a href="{{ route('register') }}">Register</a><br>
				@endif
			@endauth
		@endif
	</body>
</html>
```

Refresca el navegador i comprova que ara la pàgina inicial ha estat personalitzada, tot i que es pot millorar afegint CSS.

Aquest darrer pas, perment introduir 2 conceptes:

- Enrutament a partit de la pàgina **web.php** de la carpeta **routes**.
- Els fitxers **blade** que són plantilles per poder fer la part de les **vistes** del patró **MVC**, que es troben a la carpeta **resources/views** i que tenen el seu propi llenguatge conegut com Blade templating language amb:
  - Directives tipus @if, @switch, @for, @foreach, @while
  - Directives d'autenticació: @auth → Comprova si un usuari ha estat o no autenticat
  - {{ }} que es l'equivalent a executar un echo de PHP `{{ route('register') }}` fa un echo de la ruta al blade de registre.