

Accés a BD MySQL realitzant operacions CRUD amb el framework de Laravel

1- Comprova que dins del directori **empleats** que en el moment de la instal·lació del framework de Laravel s'ha creat un fitxer **.gitignore** i que el fitxer **.env** està inclòs dins de la llista dels fitxers que no es trobaran en seguiment pel sistema i per tant tampoc es pujaran a **Github**.

2- Dins del directori **empresa** crea un dipòsit local **git**:

```
git init
```

3- Dins del directori **empresa** crea un fitxer **.gitignore** per evitar el seguiment del fitxer **empresa.sql**. A continuació fes un **add** del projecte i un **commit** amb el missatge "*Commit 1 del projecte*".

4- Crea un dipòsit **privat** de **Github** de nom **empresa**. Sincronitza el dipòsit local amb el dipòsit remot.

5- **artisan** és el nom d'una ordre que s'inclou amb cada instal·lació del framework de **Laravel** i que necessita l'interpret de **php** per ser interpretada i executada. Artisan permet fàcilment, entre altres opcions, crear **models**, **controladors** i **vistes**. L'ordre artisan es trobarà dins d'**empleats** que és on s'ha instal·lat el framework de **Laravel**.

Les **Migrations** de **Laravel** permeten crear, esborrar i canviar l'estructura 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. Amb les **Migrations** podem per exemple, crear, esborrar, reanomenar o canviar l'estructura de les taules d'una base de dades. Les Migrations es defineixen dins de fitxers que es troben al subdirectori **database/migrations**.

En el moment de crear un **Model** amb **Laravel** també es poden crear les **Migrations** per crear una taula dins de la base de dades en el moment d'executar l'aplicació.

Dins del directori **empleats**, crea el Model **Empleat** i les **Migrations** per crear que per defecte s'anomenarà **empleats** (tot en **mínuscula** i amb una **s** al final):

```
php artisan make:model Empleat -m
```

6- Comprova que s'ha creat dins de **database/migrations** un fitxer amb un nom similar a **2021_03_22_115750_create_empleats_table.php** però tenint en compte que **2021_03_22_115750** fa referència a la data de creació i hora de creació del fitxer.

7- Indica els camps que volem que tingui la taula **empleats** dins del fitxer de **Migrations**. Volem els següent camps: **id** (identificador), **nom** de tipus text, **telefon** de tipus text, **email** de tipus text i **timestamps**.

Per tant el contingut serà 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');
    }
}
```

8- Defineix els camps de la taula que podem ser omplerts a partir de les dades enviades dins d'una array. Això és útil per poder introduir dades enviades des d'un formulari omplert per un usuari. Això és un tema a resoldre dins del Model, de manera que hem d'accedir al model associat a la taula. Aquest Model ha estat creat en el punt 6 i el seu nom serà per tant **Empleat.php** i es trobarà a **app/Models**.

El fitxer **Empleat.php** tindrà aquest codi:

```
<?php

namespace App\Models;

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

class Empleat extends Model
{
    use HasFactory;
    protected $fillable = [
        'nom',
        'email',
        'telefon'
    ];
}
```

I això vol dir bàsicament que l'usuari omplirà per mitjà del formulari els camps **nom**, **email** i **telefon**.

9- Ara ja podem crear les taules del projecte. Executa dins del directori **empleats**:

```
php artisan migrate
```

10- Ara hem de crear el Controller. Executa dins del directori **empleats**:

```
php artisan make:controller ControladorEmpleat --resource
```

i comprova que es crea **app/Http/Controllers/ControladorEmpleat.php**.

NOTA IMPORTANT: No oblidis de fer disponible la ruta del controlador descomentant la **línia 29** de **app/Providers/RouteServiceProvider.php**. Per tant la línia:

```
protected $namespace = 'App\Http\Controllers';
```

del fitxer **app/Providers/RouteService/Provider.php** NO ha d'estar comentada.

11- Afegeix a **ControladorEmpleat.php** que es comunica amb el model **Empleat**. Afegeix el codi:

```
use App\Models\Empleat;
```

Escriu aquesta línia després de la línia `use Illuminate\Http\Request;`

12- Afegeix el següent codi al mètode **index** de la classe **ControladorEmpleat** definida dins del fitxer **ControladorEmpleat.php**:

```
public function index()
{
    $empleat = Empleat::all();
    return view('index', compact('empleat'));
}
```

El mètode **Empleat::all()** recupera tots els registres de la taula **empleats**. La classe **Empleat** forma part del Model **Empleat.php** associat a la taula **empleats**.

13- Afegeix el següent codi als mètodes **create** i **store** de la classe **ControladorEmpleat** definida dins del fitxer **ControladorEmpleat.php**:

```
public function create()
{
    return view('welcome');
}

public function store(Request $request)
{
    $nouEmpleat = $request->validate([
        'nom' => 'required|max:255',
        'email' => 'required|max:255',
        'telefon' => 'required|max:255',
    ]);
    $empleat = Empleat::create($nouEmpleat);

    return redirect('/empleats')->with('completed', 'Empleat creat!');
}
```

El mètode **Empleat::create()** crea un registre dins la taula **empleats**. La classe **Empleat** forma part del Model **Empleat.php** associat a la taula **empleats**.

14- Afegeix el següent codi als mètodes **edit** i **update** de la classe **ControladorEmpleat** definida dins del fitxer **ControladorEmpleat.php**:

```
public function edit($id)
{
    $empleat = Empleat::findOrFail($id);
    return view('actualitza', compact('empleat'));
}

public function update(Request $request, $id)
{
    $dades = $request->validate([
        'nom' => 'required|max:255',
        'email' => 'required|max:255',
        'telefon' => 'required|max:255',
    ]);

    Empleat::whereId($id)->update($dades);
    return redirect('/empleats')->with('completed', 'Empleat actualitzat');
}
```

15- Afegeix el següent codi al mètode **destroy()** de la classe **ControladorEmpleat** definida dins del fitxer **ControladorEmpleat.php**:

```
public function destroy($id)
{
    $empleat = Empleat::findOrFail($id);
    $empleat->delete();
    return redirect('/empleats')->with('completed', 'Empleat esborrat');
}
```

Per trobar informació sobre [findOrFail\(\)](#) i altres mètodes disponibles dins d'un Model gràcies a que Laravel inclou l'[ORM Eloquent](#) pots anar [aquí](#).

16- Ara crearem les rutes. Obre el fitxer **routes/web.php** i afegeix el següent codi al final del fitxer:

```
Route::resource('empleats', ControladorEmpleat::class);
```

De manera que l'adreça URL **http://localhost:8000/empleats** ens porti directament a l'aplicació, i s'executin els mètodes index, create, edit i destroy seguint els següents criteris:

| Mètode | URI | Nom | Acció |
|-----------|-------------------------|------------------|---|
| GET/HEAD | empleats | empleats.index | App/Http/Controllers/ControladorEmpleat@index |
| POST | empleats | empleats.store | App/Http/Controllers/ControladorEmpleat@store |
| GET/HEAD | empleats/create | empleats.create | App/Http/Controllers/ControladorEmpleat@create |
| GET/HEAD | empleats/{empleat} | empleats.show | App/Http/Controllers/ControladorEmpleat@show |
| PUT/PATCH | empleats/{empleat} | empleats.update | App/Http/Controllers/ControladorEmpleat@update |
| DELETE | empleats/{empleat} | empleats.destroy | App/Http/Controllers/ControladorEmpleat@destroy |
| GET/HEAD | empleats/{empleat}/edit | empleats.edit | App/Http/Controllers/ControladorEmpleat@edit |

Per més documentació sobre el tema de les rutes amb el mètode **resource()** de la classe **Route** podeu llegir el següent enllaç: <https://laravel.com/docs/8.x/controllers>

17- Ara crearem les **vistes**. Dins del directori **resources/views** crearem el fitxer un fitxer amb el disseny general (o layout) de les vistes. Aquí afegirem el framework **Bootstrap** que volem utilitzar. Crea un fitxer de nom **disseny.blade.php** amb el contingut que trobareu aquí:

<https://github.com/dacomo2021daw2/empresa/blob/main/empleats/resources/views/disseny.blade.php>

18- Després modificarem el fitxer **welcome.blade.php**. El contingut d'aquest fitxer el trobareu aquí:

<https://github.com/dacomo2021daw2/empresa/blob/main/empleats/resources/views/welcome.blade.php>

Fixat que:

- La vista presentada en aquest cas ja és el formulari per introduir un nou usuari i això quadra amb el mètode **create** de la classe **ControladorEmpleat** dins del controlador **ControladorEmpleat.php**.
- Utilitza el disseny general creat dins de **disseny.php**.
- El formulari utilitza el mètode POST i redirecciona al mètode **store** de de la classe **ControladorEmpleat** dins del controlador **ControladorEmpleat.php**. Per tant, amb les dades enviades es crea una nova entrada dins de la base de dades **empleats**.

19- Crea la vista **index.blade.php** (la qual es cridada dins del mètode **index** de de la classe **ControladorEmpleat** dins del controlador **ControladorEmpleat.php** tal i com es pot veure a l'apartat **11**). El contingut el trobareu aquí:

<https://github.com/dacomo2021daw2/empresa/blob/main/empleats/resources/views/index.blade.php>

És interessant comprovar com Laravel permet dur a terme una tècnica anomenada [Method Spoofing](#) per poder enviar dins d'un formulari HTML (que no tenen suport pels mètodes PUT, PATCH i DELETE) per mitjà d'un camp ocult que serà utilitzat per l'aplicació com si fos el mètode realment enviat. Això es pot veure a la línia de codi: `@method('DELETE')` ;

20- Crea la vista **actualitza.blade.php** (la qual es cridada dins del mètode **update** de de la classe **ControladorEmpleat** dins del controlador **ControladorEmpleat.php** tal i com es pot veure a l'apartat **13**). El contingut el trobareu aquí:

<https://github.com/dacomo2021daw2/empresa/blob/main/empleats/resources/views/actualitza.blade.php>

Comprova que utilitzem `@method('PATCH')` ; per enviar el mètode **PATCH** utilitzant [Method Spoofing](#).

21- Ara farem un **add** i un **commit** local del nostre projecte amb el missatge "*Commit 2 del projecte*" i després puja el projecte al **Github**.

22- Finalment, des de dins del directori **empleats**, executa:

```
php artisan serve --host=0.0.0.0 --port=8000
```

per posar en marxa un servidor web intern de Larvavel que escolta pel port **8000** i accepta connexions des de qualsevol interfície i adreça IP. Això ens permet testejar l'aplicació en fase de desenvolupament sense haver de pujar-la al servidor web de producció.

23- Afegeix uns quants empleats a la base de dades. Dins del directori **empresa** crea un script de nom **afegeix_empleats.sql** amb les ordres SQL per afegir **4** usuaris a la taula **empleats** de la base de dades **empleats**. El codi seria:

```
alter table empleats.empleats modify created_at timestamp not null;
alter table empleats.empleats modify updated_at timestamp not null;
insert into empleats.empleats (nom,email,telefon) values ("Joan Perez Pons", "jpp@gmail.com","666554433");
insert into empleats.empleats (nom,email,telefon) values ("Robert Capdevila Altamira", "capdevila@fje.edu","666778899");
insert into empleats.empleats (nom,email,telefon) values ("Sergi Lopez Miramon", "slopez@fjeclot.net","934443322");
insert into empleats.empleats (nom,email,telefon) values ("Laia Colomer Gonzalez", "laia.colomer@fje.edu","934445566");
```

Des del directori **empresa** i amb el terminal accedeix al gestor de **MariaDB** i afegeix l'usuari i la base de dades. Executa des del terminal del sistema:

```
sudo mysql
```

i des de dins del gestor **MariaDB**, executa:

```
source ./afegeix_empleats.sql
```

25- Amb el navegador de l'equip hosts accedeix a <http://localhost:8000> (o http://adreça_ip:8000) per començar a utilitzar la teva aplicació.