1- Conceptes bàsics de Laravel

Conceptes bàsics	Passos a seguir per un projecte bàsic
1- Namespace 2-Models i Migrations 3-Controllers 4-Views 5- Blades i Blade Templates 6-Routing 7- Request & Responses 8-Middleware 9- 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)

2- Gestió d'excepcions: Renderització d'una pàgina personalitzada per error d'accés a la base de dades

1- Atura el servidor **MySQL** i comprova que es mostra una pàgina genèrica d'error de Laravel no personalitzada.

2- Modifica **bootstrap/app.php** que entre altres funcions té la responsabilitat de gestionar les excepcions. Dins del seu codi, afegeix:

- A la secció dels **use** afegeix:

```
use Illuminate\Database\QueryException;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\View;
```

- El codi de la secció **->withExceptions** serà aquest:

```
->withExceptions(function (Exceptions $exceptions) {
        $exceptions->renderable(function(QueryException $exception) {
                if ($exception->getCode()=='2002'){
                        return new Response(View::make('errorbd'));
                };
        });
})->create();
```

3- Explicació bàsica del codi:

- Els **use** permeten afegir les classes necessàries per gestionar les excepcions.
- **->withExceptions** gestiona les excepcions (com per exemple, les excepcions que es produeixen quan hi ha un error d'accés a la base de dades) i permeten crear logs de les excepcions i renderitzar pàgines de resposta.
- El mètode **renderable()** mostra la vista **error.blade.php** si es produeix un error de connexió a la base de dades.
- La vista **errobd.blade.php** hauria de crear una pàgina mostrant un missatge d'error i d'aquesta manera personalitzar el comportament de gestió d'errors.

4- Una excepció es pot produir (entre altres motius) per:

- Tenir el servidor de base de dades aturat
- Tenir malament configurat el fitxer **.env**
- El port del servidor no és correcte
- La contrasenya o usuari no són correctes.
- L'adreça IP o nom del host no són correctes

5- Ara, afegeix a **resources/views** el fitxer de vistes **errobd.blade.php** al qual es quan es produeix una excepció. Dins de l'arxiu, escriu l 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>Error d'accés a la BD</title>
	</head>
	<body>
		<h1>Atenció!!!!</h1>
		<p>Error tipus: "SQLSTATE[HY000] [2002] Connection refused"</p>
		<p>
			Comprova que:
			<ol>
				<li>El servidor MySQL està en marxa</li>
				<li>L'adreça IP i/o nom del host són correctes</li>
				<li>El port del servidor és correcte</li>
				<li>El nom d'usuari i contrasenya són correctes</li>
				<li>El fitxer de configuració de Laravel és correcte</li>
			</ol>
		</p>
	</body>
</html>
```

6- Atura el servidor **MySQL** i comprova que es mostra **error.blade.php** quan s'intenta establir una connexió a l'aplicació.

2- Creant multiples roles d'usuari amb Laravel pel projecte del M07UF3

2.1- Modificació de la taula users de la base de dades empresa

1- Farem que la base de dades empresa treballi amb 2 tipus d'usuaris amb 2 roles difierents:

- Usuari tipus **basic**:
  - Només pot visualitzar la llista de treballardor i part de les dades del treballadors.
  - No té accés a l'opció de registre d'usuaris
- Usuari tipus **admin**:
  - Pot visualitzar la llista d'usuaris, totes les seves dades, crear-ne de nous, esborrar-ne i actualitzar dades dels usuaris.
  - Té accés a la sescció de registre d'usuaris

2- Primer de tot, haurem d'afegir el camp **role** a la taula **users** de la base de dades **empresa**. Anirem a **empresa** i executarem:

```
php artisan make:migration afegeix_camp_role_a_users
```

i comprovem que es crea un fitxer **xxxx_xx_xx_xxxxxx_afegeix_camp_role_a_users.php** dins de **database/migrations**. Compte que **xxxx_xx_xx_xxxxxx** representa l'any, mes, dia i hora de creació del fitxer.

3- Ara obrim el fitxem que hem creat i farem que tingui aquest codi:

```
<?php

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

class AfegeixCampRoleAUsers extends Migration
{
	/**
	 * Run the migrations.
	 *
	 * @return void
	 */

	public function up()
	{
		Schema::table('users', function (Blueprint $table) {
			$table->string('role')->after('name')->default('admin');
		});
	}

	/**
	 * Reverse the migrations.
	 *
	 * @return void
	 */

	public function down()
	{
		Schema::table('users', function (Blueprint $table) {
			$table->dropColumn('role');
		});
	}
}
```

Això ens permet:

- La creació de la nova columna **role** de tipus **string** després de la columna **name**.
- Els usuaris existents no s'esborraran i per defecte tindran el role **admin**.
- Podem esborrar fàcilment la nova columna si fem un **rollback**.

A continuació executarem: **php artisan migrate** i comprovarem que s'ha creat la nova columna **role** dins de la taula **users** sense perdre els usuaris existents.

2.2- Utilitzant seeders per afegir usuaris amb les seves dades des de Laravel

1- Com ja hem vist, Laravel:

- Inclou la possibilitat d'omplir amb dades les taules d'una base de dades abans de començar a utilitzar l'aplicació utilizant "seeders".
- Treballant amb "seeders", cada taula de la base de dades estarà associada a un "seeder" que no és res més que una classe amb un mètode (de nom run) dins del qual inclourem el codi amb les dades que volem introduir.
- El codi amb la classe i les seves dependències estarà a un fitxer que es trobarà dins de **database/seeders** . per cada taula trobarem un fitxer dins d'aquest directori amb la classe definida dins del fitxer.
- Tots els "seeders" s'hauran d'enregistrar per ser cridats dins de la classe **DatabaseSeeder** del fitxer **database/seeders/DatabaseSeeder.php**.
- Es poden crear seeders i cridar-los per omplir les taules de les bases de dades utilitzant l'ordre de Laravel **php artisan**.
- Una opció típica d'utilització de "seeders" és afegir usuaris a l'aplicació de manera que ja puguem començar a utilitzar-los sense que calgui utilitzar la pròpia aplicació per crear-los.

2- Per crear un nou **Seeder** per omplir la taula d'**usuaris** executa des del directori **empresa**:

php artisan make:seeder UsuarisAmbRolesSeeder

3- Comprova que s'ha creat el fitxer **database/seeders/UsuarisAmbRolesSeeder.php** a on s'ha definit una nova classe de nom **UsuarisAmbRolesSeeder** amb els mètodes necessaris pel nostre objectiu.

4- Ara anem a desenvolupar el mètode **run()** de la classe **UsuarisAmbRolesSeeder**. Crea el següent codi:

```
public function run(): void
{
	$llista_usuaris = [
		[
			'name' => 'leniad',
			'role' => 'admin',
			'email' => 'leinad@fjeclot.net',
			'password' => Hash::make('fjeClot25#')
		],
		[
			'name' => 'aletse',
			'role' => 'basic',
			'email' => 'aletse@fjeclot.net',
			'password' => Hash::make('clotFje25@')
		],
	];
	DB::table('users')->insert($llista_usuaris);
}
```

i per poder utiliza les classes **Hash** i **DB** haurem d'escriure abans de la classe **UsuarisAmbRolesSeeder**:

```
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\DB;
```

En aquest cas és important remarcar que el mètode **insert()** de la classe **DB** permet **insertar un registre** dins de la taula indicada dins del mètode **table()**.

5- Quan s'executi l'ordre per utilitzar els "seeders", Laravel executa el mètode **run()** de la classe **DatabaseSeeder** que es troba a **database/seeders/DatabaseSeeder.php**. Dins d'aquest mètode cal indicar que s'ha de fer una crida al nou "seeder" **UsuarisAmbRolesSeeder**. El codi complet de **DatabaseSeeder.php** pel nostre cas es pot veure a continuació:

```
<?php

namespace Database\Seeders;

use App\Models\User;
// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{

	/**
	 * Seed the application's database.
	 *
	 * @return void
	*/
	public function run(): void
	{
		$this->call([
			UsuarisAmbRolesSeeder::class,
		]);
	}
}
```

6- Finalment, executarem l'ordre des del directori **empresa**:

php artisan db:seed

i comprovarem que s'han creat els nous registres dins de la taula **users** de la base de dades **empresa**.

2.3- Modificació de les vistes, rutes, controladors i models per usuaris tipus admin

1- Modifica la vista **inici.blade.php**. El nou codi complet del fitxer serà aquest:

```
<!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>
		<p>Pàgina inicial de l'aplicació web Empresa</p>
		@if (Route::has('login'))
			@auth
				<a href="{{ url('/dashboard') }}">Dashboard</a>
			@else
				<a href="{{ route('login') }}">Log in</a><br>
			@endauth
		@endif
	</body>
</html>
```

2- A **dashboard.blade.php** afegirem el següent codi després de la línia **16**:

```
<div class="p-6 bg-white border-b border-gray-200">
	<a href="{{ route('register') }}">Crea un nou usuari</a><br>
</div>
```

I afegirem l'opció de **registre** dins del **dashboard**. També modificarem la línia 4 amb el codi:

```
{{ __('Dashboard dels usuaris tipus admin') }}
```

3- En el projecte no es demanarà que els usuaris puguin enregistrar-se ells mateixos. Només un usuari administrador ho podrà fer. Això vol dir que farem alguns canvis en les rutes. Primer de tot:

- Comentarem dins del fitxer **routes/auth.php** dins de la secció **`Route::middleware('guest')`** les línies **15** a **18** => Un usuari sense validar ja no pot enregistrar-se.

- Dins del fitxer **routes/auth.php**, dins de la secció **`Route::middleware('auth')`** afegirem les línies comentades anteriorment després de la línia 38.

4- Dins de la vista **register.blade.php** que es troba a **resources/views/auth**, comentarem les línies **43 a 45** perquè ja no són necessàries. A continuació, afegirem després de la línia **41** el següent codi:

```
<!-- Role -->
<div>
	<x-input-label for="role" :value="__('Role')" />
	<x-text-input id="role" class="block mt-1 w-full" type="text"
	name="role" :value="old('role')" required />
	<x-input-error :messages="$errors->get('role')" class="mt-2" />
</div>
```

i finalment, després de l'etiqueta **</form>** afegeix:

```
<div class="p-6 bg-white border-b border-gray-200">
	<a href="{{ url('dashboard') }}">Torna al dashboard<a/>
</div>
```

5- Dins del controlador **app/Http/Controllers/Auth/RegisteredUserController.php** modificarem la funció **store()** de manera que tingui aquest codi:

```
public function store(Request $request)
{
	$request->validate([
		'name' => ['required', 'string', 'max:255'],
		'role' => ['required', 'string', 'max:5'],
		'email' => ['required', 'string', 'lowercase', 'email', 'max:255',
			'unique:'.User::class]
		'password' => ['required', 'confirmed', Rules\Password::defaults()],
	]);

	$user = User::create([
		'name' => $request->name,
		'role' => $request->role,
		'email' => $request->email,
		'password' => Hash::make($request->password),
	]);

	event(new Registered($user));

	Auth::login($user);

	return redirect('dashboard');
}
```

6- Ara, dins de**app/Models/User.php** haurem de modificar l'atribut **$fillable** de la següent manera:

```
protected $fillable = [
	'name',
	'role',
	'email',
	'password',
];
```

7- Finalment ens assegurarem d'entrar al **Dashboard** d'**administrador** només si l'usuari és de tipus **admin**. Accedeix a **app/Http/Controllers/Auth** i modifica el fitxer **AuthenticatedSessionController.php**. Fes que el mètode **store()** tingui aquest codi:

```
public function store(LoginRequest $request)
{
	$request->authenticate();

	$request->session()->regenerate();

	if (auth()->user()->role == 'admin') {
		return redirect('dashboard');
	}
	elseif(auth()->user()->role == 'basic'){
		return redirect('dashboard-basic');
	}
	else{
		return auth()->logout();
	}
}
```

2.4- Vistes, rutes, controladors i models per usuaris tipus basic

1- Dins **resources/views** crea **dashboard-basic.blade.php** amb el següent codi:

```
<x-app-layout>
	<x-slot name="header">
		<h2 class="font-semibold text-xl text-gray-800 leading-tight">
			{{ __('Dashboard dels usuaris tipus bàsic') }}
		</h2>
	</x-slot>

	<div class="py-12">
		<div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
			<div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
				<div class="p-6 bg-white border-b border-gray-200">
					<a href="{{ url('trebs/index_basic') }}">Treballadors:
					Visualització bàsica<a/>
				</div>
			</div>
		</div>
	</div>
</x-app-layout>
```

2- Dins **app/Http/Controllers/ControladorTreballador.php** afegeix un nou mètode de nom **index_basic()** dins de la clase **ControladorTreballador** amb el següent codi:

```
public function index_basic()
{
	$dades_treballadors = Treballador::all();
	return view('llista-basica', compact('dades_treballadors'));
}
```

3- Dins **app/Http/Controllers/ControladorTreballador.php** crea un altre mètode de nom **show_basic()** dins de la clase **ControladorTreballador** amb el següent codi:

```
public function show_basic($tid)
{
	$dades_treballadors = Treballador::findOrFail($tid);
	return view('mostra-basica', compact('dades_treballadors'));
}
```

4- Dins **resources/views** crea **llista-basica.blade.php** amb el següent codi:

```
@extends('disseny')
@section('content')
<h1>Llista d'empleats</h1>
<div class="mt-5">
	<table class="table table-striped table-bordered table-hover">
		<thead>
			<tr class="table-primary">
				<td>tid</td>
				<td>Nom</td>
				<td>Cognoms</td>
				<td>Accions sobre la taula</td>
			</tr>
		</thead>
		<tbody>
		@foreach($dades_treballadors as $treb)
			<tr>
				<td>{{$treb->tid}}</td>
				<td>{{$treb->nom}}</td>
				<td>{{$treb->cognoms}}</td>
				<td class="text-left">
					<a href="{{ route('trebs.show_basic', $treb->tid)}}"
				class="btn btn-info btn-sm">Mostra</a>
				</td>
			</tr>
		@endforeach
		</tbody>
	</table>
	<div class="p-6 bg-white border-b border-gray-200">
		<a href="{{ url('dashboard-basic') }}">Torna al dashboard<a/>
	</div>
<div>
@endsection
```

5- Dins **resources/views** crea **mostra-basica.blade.php** amb el següent codi per fer una visualització parcial de dades dels treballadors:

```
@extends('disseny')
@section('content')
<h1>Dades del treballador</h1>
<div class="mt-5">
	<table class="table table-striped table-bordered table-hover">
		<thead class="thead-dark">
			<tr class="table-primary">
				<th scope="col">CAMP</td>
				<th scope="col">VALOR</td>
			</tr>
		</thead>
		<tbody>
			<tr>
				<td>tid</td>
				<td>{{$dades_treballadors->tid}}</td>
			</tr>
			<tr>
				<td>Nom</td>
				<td>{{$dades_treballadors->nom}}</td>
			</tr>
			<tr>
				<td>Cognoms</td>
				<td>{{$dades_treballadors->cognoms}}</td>
			</tr>
			<tr>
				<td>Categoria</td>
				<td>{{$dades_treballadors->categoria}}</td>
			</tr>
			<tr>
				<td>Nom de la feina</td>
				<td>{{$dades_treballadors->nom_feina}}</td>
			</tr>
		</tbody>
	</table>
	<div class="p-6 bg-white border-b border-gray-200">
		<a href="{{ url('dashboard-basic') }}">Torna al dashboard<a/>
	</div>
<div>
```

```
@endsection
```

6- Finalment, afegeix a **routes/web.php** les rutes a **dasboard-basic** i als nous mètodes **index_basic** i **show_basic** de la classe **ControladorTreballador**. Afegeix les rutes:

```
Route::get('/dashboard-basic', function () {
        return view('dashboard-basic');
})-> name('dashboard-basic');

Route::get('trebs/index_basic', [ControladorTreballador::class, 'index_basic'])→name('trebs.index_basic');

Route::get('trebs/show_basic/{id}', [ControladorTreballador::class, 'show_basic'])→name('trebs.show_basic');
```

que haurien d'estar dins de la secció **Route::group(['middleware' => 'auth']**.

De manera que funcioni correctament, s'hauria de passar la línia **Route::resource('trebs', ControladorTreballador::class);** al final del **Route::group(['middleware' => 'auth']** o en cas contrari interferirà amb el correcta funcionament dels **Routes::get** afegits.

3- Pàgina d'inici amb accés a pàgina informàtiva i de login

1- Dins **resources/views** crea una pàgina de nom **info.blade.php** 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>
		<p>Pàgina informativa de l'aplicació web Empresa</p>
		<p>
		Aquesta aplicació té 2 tipus d'usuaris:
		<ol>
			<li>Administrador: crea/modifica/esborra/visualitza treballadors</li>
			<li>Basic: visualitzar treballadors</li>
		</ol>
		<a href="{{ url('/') }}">Inici</a><br>
		</p>
	</body>
</html>
```

2- Fes que el nou codi del fitxer **inici.blade.php** sigui aquest:

```
<!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>
		<p>Pàgina inicial de l'aplicació web Empresa</p>
		<a href="{{ url('/info') }}">Info</a><br>
		@if (Route::has('login'))
			@auth
				<a href="{{ url('/dashboard') }}">Dashboard</a>
			@else
				<a href="{{ route('login') }}">Log in</a><br>
			@endauth
		@endif
	</body>
</html>
```

6- Finalment, afegeix a **routes/web.php** la ruta a **/info** utilitzant el mètode GET:

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

Un bon lloc per afegir-la és després de la ruta a **/** pel mètode **GET** que ens redirecciona cap a **inici**.