

Creant múltiples rols d'usuari amb Laravel pel projecte del M07UF3

A- 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 rols diferents:

- Usuari tipus **basic**:
 - Només pot visualitzar la llista de treballadors i part de les dades dels 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 sessió 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 afegix_camp_role_a_users`

i comprovem que es crea un fitxer `xxxx_xx_xx_XXXXXX_afegix_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.

Ara obrim el fitxer i farem que tingui aquest codi:

```
<?php
```

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

class AfegixCampRoleAUsers 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.

B) 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 esborrades 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();
    }
}
```

C) 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 classe **ControladorTreballador** amb el següent codi:

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

3- 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
```

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

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

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</th>
                <th scope="col">VALOR</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>tid</td>
                <td>{{ $dades_treballador->tid }}</td>
            </tr>
            <tr>
                <td>Nom</td>
                <td>{{ $dades_treballador->nom }}</td>
            </tr>
            <tr>
                <td>Cognoms</td>
                <td>{{ $dades_treballador->cognoms }}</td>
            </tr>
            <tr>
                <td>Categoria</td>
                <td>{{ $dades_treballador->categoria }}</td>
            </tr>
            <tr>
                <td>Nom de la feina</td>
                <td>{{ $dades_treballador->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 **dashboard-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@index_basic')->name('trebs.index_basic');

Route::get('trebs/show_basic/{tid}', 'ControladorTreballador@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 correcte funcionament dels **Routes::get** afegits.

D) Utilitzant seeders per afegir usuaris amb les seves dades des de Laravel

1- Laravel:

- Inclou la possibilitat d'omplir amb dades les taules d'una base de dades abans de començar a utilitzar l'aplicació utilitzant "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 seeder per omplir, per exemple, la taula d'**usuaris**, executa des del directori **empresa**:

php artisan make:seeder UserSeeder

a on **UserSeeder** serà el nom de la classe que es crearà en el fitxer **database/seeders/UserSeeder.php**.

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

```
public function run(): void
{
    $llista_usuaris = [
        [
            'name' => 'leniad',
            'role' => 'admin',
            'email' => 'leinad@fjeclo.net',
            'password' => Hash::make('fjeClot23#')
        ],
        [
            'name' => 'aletse',
            'role' => 'basic',
            'email' => 'aletse@fjeclo.net',
            'password' => Hash::make('clotFje24@')
        ],
    ];
    DB::table('users')->insert($llista_usuaris);
}
```

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

```
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()**.

4- 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" **UserSeeder**. El codi complet de **DatabaseSeeder.php** pel nostre cas es pot veure a continuació (a la següent pàgina):

```

<?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([
            UserSeeder::class,
        ]);
    }
}

```

5- 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**.