Pla de Recuperació, Transformació i Resiliència
FPCV Formació Professional Comunitat Valenciana
Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur
GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura, Universitats i Ocupació
IES MARIA ENRIQUEZ

Unitat 5. Frameworks. Laravel

2n DAW - IES María Enríquez

7. Autenticació amb sessions

Autenticació per sessions

En l'arxiu `config/auth.php` es disposa d'algunes opcions de configuració generals d'autenticació recolzada en dos elements: els **guards** i els **providers**.

- Els **guards** són mecanismes que defineixen **com s'autenticaran els usuaris** per a cada petició. El mecanisme més habitual és mitjançant **sessions**, on es guarda la informació de l'usuari autenticat en la sessió. També es poden fer servir **tokens**.
- Els **providers** indiquen com s'obtindran els **usuaris de la base de dades** per a comprovar l'autenticació. Les opcions habilitades per defecte són mitjançant **Eloquent** (i el model d'usuaris que tinguem definit), o mitjançant **query builder**, consultant directament la taula corresponent d'usuaris.

Autenticació per sessions

A l'arxiu de configuració indiquem:

- Taula on estan els usuaris per defecte users i el model associat

```
'providers' => [
    'users' => [
        'driver' => 'eloquent',
        'model' => App\Models\Usuario::class,
    ],

    // 'users' => [
    //     'driver' => 'database',
    //     'table' => 'usuarios',
    // ],
],
```

Autenticació per sessions

A més, haurem d'assegurar-nos que la classe **Usuario** ha d'heretar de la classe **Authenticable**

```
use Illuminate\Foundation\Auth\User as Authenticatable;


class Usuario extends Authenticatable
```

Autenticació per sessions

Model o taula d’usuaris

Si triem el provider basat en **Eloquent**, haurem de tindre un **model d'usuaris** al qual accedir (Usuario).

Si optem per utilitzar el **query builder** en lloc de Eloquent, haurem de tindre una taula en la base de dades on estiguen les dades dels usuaris.

En qualsevol cas, com veurem a continuació, serà convenient que els passwords dels usuaris estiguen encriptats mitjançant **bcrypt**. Actualitzem el seeder UsuariosSeeder:

Autenticació per sessions

```
php artisan make:seeder UsuariosSeeder
```

En el Seeder

```
public function run()
{
    $usuario = new Usuario();
    $usuario->login = 'admin';
    $usuario->password = bcrypt('admin');
    $usuario->save();
}
```

Autenticació per sessions

Finalment, carreguem aquest nou seeder en la base de dades, o bé amb una migració completa nova (`php artisan migrate:fresh --seed`), o bé executant només el seeder amb això:

```
php artisan db:seed --class=UsuariosSeeder
```

Autenticació per sessions

Afegir autenticació a un projecte

Seguirem aquests passos:

1. Definirem un **formulari de login.**
2. Definirem un nou **controlador** que s'encarregue de gestionar el **login**: tant de mostrar el formulari quan l'usuari no estiga autenticat com de validar les seues credencials quan les envie.
3. Afegirem les **rutes** pertinents en l'arxiu `routes/web.php` tant per al formulari de login com per a l'autenticació posterior.
4. **Protegirem les rutes** que siguen d'accés restringit
5. Opcionalment, podem afegir també una opció de **logout**.

1. **Definirem un formulari de login** en la vista `resources/views/auth/login.blade.php`. També deixarem una zona per a mostrar un possible missatge d'error si l'autenticació no ha sigut exitosa:

```
<h1>Login</h1>
@if (!empty($error))
  <div class="text-danger">
      {{ $error }}
  </div>
@endif
<form action="{{ route('login') }}" method="POST">
        @csrf
        <div class="form-group">
            <label for="login">Login:</label>
            <input type="text" class="form-control"
             name="login" id="login" />
        </div>
        <div class="form-group">
            <label for="password">Password:</label>
            <input type="password" class="form-control"
             name="password" id="password" />
        </div>
        <input type="submit" name="enviar" value="Enviar"
         class="btn btn-dark btn-block">
    </form>
```

Autenticació per sessions

2. Controlador de login

Crearem un **nou controlador** que s'encarregue de gestionar tota l'autenticació:

```
php artisan make:controller LoginController
```

Dins, definim una funció que s'encarregarà de **mostrar el formulari** anterior:

```
public function loginForm()
{
    return view('auth.login');
}
```

I afegirem una segona funció que s'encarregue de **validar les credencials** enviades per l'usuari. Per a això, farem ús del facade d'autenticació, existent en `Illuminate\Support\Facades\Auth`.

Autenticació per sessions

```
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class LoginController extends Controller
{
    ...
    public function login(Request $request)
    {
        $credenciales = $request->only('login', 'password');

        if (Auth::attempt($credenciales))
        {
            // Autenticación exitosa
            return redirect()->intended(route('libros.index'));
        } else {
            $error = 'Usuario incorrecto';
            return view('auth.login', compact('error'));
        }
    }
}
```

Autenticació per sesions

Attempt

El mètode **attempt** accepta una sèrie de parells **clau-valor** com a primer paràmetre. En aquest cas, li passem login i password rebuts. En el cas dels passwords, Laravel **automàticament els encripta** en format bcrypt.

Intended

El mètode **intended** tracta d'enviar a l'usuari a la **ruta a la qual intentava accedir** abans que se li sol·licitara autenticació. Li passem com a **paràmetre una ruta per defecte** en el cas que el destí previst no estiga disponible.

Autenticació per sessions

3. Rutes associades

Finalment, hem de definir les rutes tant per a mostrar el formulari (per get) com per a arreplegar les credencials i validar a l'usuari (per post).

```
Route::get('login', [LoginController::class, 'loginForm'])->name('login');
Route::post('login', [LoginController::class, 'login']);
```

Autenticació per sessions

4. Redirecció en cas d'error

Per defecte, si un usuari no autenticat vol accedir a una ruta protegida, se le redirigeix a la **ruta login**.

Si volem **personalitzar** aquesta acció hem de modificar el mètode `redirectTo` en el middleware d'autenticació `app/Http/Middleware/Authenticate.php`:

```
protected function redirectTo($request)
{
    ...
    return route('login');
}
```

Autenticació per sessions

5. Protegir les rutes d'accés restringit

Si tenim una ruta de recursos (`Route::resource`) en l'arxiu `routes/web.php`, llavors l'opció més còmoda és definir un constructor en el **controlador** associat , i especificar quines funcions volem protegir, bé amb only o amb except :

```
use Illuminate\Routing\Controller;
…
class LibroController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth',
            ['only' => ['create', 'store', 'edit', 'update', 'destroy']]);
    }
```

Autenticació per sessions

5. Protegir les rutes d'accés restringit

Si definim les **rutes soltes**, podem emprar el mètode `middleware` per a indicar en cadascuna si volem que s'aplique el middleware d'autenticació:

```
Route::get('prueba', [PruebaController::class, 'create'])->middleware('auth');
```

Autenticació per sessions

Detectar en les vistes a l'usuari autenticat

Pot ser necessari detectar en una vista si l'usuari s'ha autenticat o no, bé per a mostrar uns certs controls, o per a carregar informació pròpia de l'usuari.

Per exemple, d'aquesta manera podem **modificar el menú de navegació** (`resources/views/partials/nav.blade.php`) perquè mostre l'enllaç de crear nou llibre només si l'usuari s'ha autenticat:

```
@if(auth()->check())
    <li class="{{ setActivo('libros.create') }} nav-item">
        <a class="nav-link" href="{{ route('libros.create') }}">Nuevo libro</a>
    </li>
@endif
```

Autenticació per sessions

- mètode `auth()->guest()` si volem comprovar si l'usuari encara **NO** s'ha autenticat (enllaç a login)
- mètode `auth()->check()` per a comprovar si **SÍ** que està autenticat (enllaç a afegir llibre).
- mètode `auth()->user()` obté l'**objecte de l'usuari autenticat**, amb el que podem accedir als seus atributs:

```
Bienvenido/a {{ auth()->user()->login }}
```

Autenticació per sessions

6. Implementació del logout

Cridar mètode logout del facade Auth des del controlador de login

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class LoginController extends Controller
{
    ...

    public function logout()
    {
        Auth::logout();
        // ... Renderizar la vista deseada
    }
```

Autenticació per sessions

També farà falta definir la ruta associada en `routes/web.php`:

```
Route::get('logout', [LoginController::class, 'logout'])->name('logout');
```

Òbviament, també serà necessari afegir un enllaç per a fer **logout** en alguna part. Podem posar-ho en el menú de navegació

```
@if(auth()->check())
    <li class="{{ setActivo('libros.create') }} nav-item">
        <a class="nav-link" href="{{ route('libros.create') }}">Nuevo libro</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="{{ route('logout') }}">Logout</a>
    </li>
@endif
```

Exercicis

Exercici 1:

- Modifica l'arxiu `config/auth.php` perquè el provider acudisca al model correcte d'usuari.
- Modifica el `factory` d'usuaris perquè els passwords s'encripten amb `bcrypt`. Perquè siga fàcil de recordar, fes que cada usuari tinga com a password el seu mateix login encriptat. Executa després `php artisan migrate:fresh --seed` per a actualitzar tota la base de dades.
- Crea un formulari de login juntament amb un controlador associat, i les rutes pertinents per a mostrar el formulari o autenticar, com hem fet en l'exemple per a la biblioteca. Recorda cridar “login” a la ruta que mostra el formulari de login.

Exercicis

Exercici 1:

- En el controlador de posts, protegeix totes les opcions menys les de `index` i `show`.
- Afig una opció de Login en el menú de navegació superior, que només estiga visible si l'usuari no s'ha autenticat encara
- Fes que només es mostren els enllaços i botons de crear, editar o esborrar quan l'usuari estiga autenticat. En eixe mateix cas, fes que també es mostre una opció de logout en el menú superior, que hauràs d'implementar.
- Finalment, afig la funcionalitat que l'usuari autenticat només pot editar i esborrar els seus propis posts, però no els dels altres usuaris.

Autenticació per sessions

Documentació:

- Autenticación con Laravel
- Uso de middleware