

¿Qué es la autenticación en Laravel?

La autenticación es el proceso que permite a un sistema identificar a un usuario y restringir acceso a ciertas partes de la aplicación solo a los que tengan sesión iniciada. Laravel lo hace fácil gracias a:

- **Middleware auth**
- **Helpers**

Funciones/facades accesibles en todo Laravel que facilitan tareas.

- **Auth::user()** → devuelve el usuario logueado o null si no hay sesión.
- **Auth::attempt()** → se usa para logueo, recibe usuario y clave.
- **Auth::check()** → permite verificar si hay sesión activa. Uso típico → mostrar el nombre del usuario en el navbar o controlar accesos o el id.
- **Auth::logout()** → se usa para cerrar sesión.
- **Directivas Blade (@auth, @guest)**

Middleware auth

El middleware es un "filtro" que se ejecuta antes de entrar a una ruta o controlador.

El middleware auth verifica si hay un usuario logueado:

- ✓ Si hay usuario → lo deja pasar.
- ✗ Si no hay → lo redirige automáticamente a /login.

¿Dónde se configura el middleware auth?

En `app/Http/Kernel.php` vas a ver algo así:

```
protected $middlewareAliases= [  
    'auth' => \App\Http\Middleware\Authenticate::class,  
    ...  
];
```

Eso significa que cuando usás `middleware('auth')`, en realidad se está ejecutando la clase `App\Http\Middleware\Authenticate` y si vemos `user.php` extiende de `Authenticate`.

¿Qué se necesita para un login en Laravel?

1. Modelo de usuarios

- Normalmente es `User` (`app/User.php`), pero también se puede reescribir
- Ese modelo debe implementar `Authenticatable` para que Laravel pueda autenticar.

2. Migración o tabla de usuarios

- La tabla `usuarios` debe tener al menos:
 - `email` o `usuario` → campo para identificar.
 - `password` → contraseña encriptada (`bcrypt`).

3. Controlador de login

- Un método `login` que reciba usuario/contraseña, valide y use `Auth::attempt()` para iniciar sesión.

- Un método `logout` para cerrar sesión.

4. Rutas de login/logout

- Definidas en `routes/web.php`.

5. Vista Blade del formulario de login

- Un form con `usuario/email` y `password`.

6. Middleware de autenticación

- `auth` → protege rutas solo accesibles para usuarios logueados.

El 1 y 2 ya lo tenemos cuando instalamos laravel

3 - Controlador

class LoginController extends Controller

```
{  
    public function showLoginForm()  
    {  
        return view('auth.login');  
    }  
    public function login(Request $request)  
    {  
        // Validar campos  
        $credentials = $request->validate([  
            'email' => 'required|email',  
            'password' => 'required',  
        ]);  
  
        // Intentar login  
        if (Auth::attempt($credentials)) {  
            $request->session()->regenerate();  
            return redirect()->intended('/dashboard'); // ruta post-login  
        }  
  
        // Si falla  
        return back()->withErrors([  
            'email' => 'Las credenciales no coinciden con nuestros registros.',  
        ]);  
    }  
}
```

```
public function logout(Request $request)  
{  
    Auth::logout();  
    $request->session()->invalidate();  
    $request->session()->regenerateToken();  
  
    return redirect('/login');  
}
```

4 - Rutas

```
Route::get('/login', 'LoginController@showLoginForm')->name('login');
```

```
Route::post('/login', 'LoginController@login');
```

```
Route::post('/logout', 'LoginController@logout')->name('logout');
```

5 - Vista Blade (resources/views/auth/login.blade.php)

```
@extends('layouts.admin')
@section('contenido')
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-4">
            <h3 class="text-center">Iniciar Sesión</h3>
            <form method="POST" action="{{ route('login') }}">
                @csrf
                <div class="form-group">
                    <label for="email">Correo electrónico</label>
                    <input type="email" name="email" id="email" class="form-
control" required autofocus>
                    @error('email')
                        <small class="text-danger">{{ $message }}</small>
                    @enderror
                </div>
                <div class="form-group">
                    <label for="password">Contraseña</label>
                    <input type="password" name="password" id="password"
class="form-control" required>
                    @error('password')
                        <small class="text-danger">{{ $message }}</small>
                    @enderror
                </div>
                <button type="submit" class="btn btn-primary btn-
block">Ingresar</button>
            </form>
        </div>
    </div>
</div>
@endsection
```

6 - Proteger rutas

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

En resumen

Para armar login en tu proyecto necesitás:

1. Modelo de user (**Usuarios**) extendiendo **Authenticatable**.
 2. Tabla **usuarios** con **email** y **password**.
 3. Controlador con **login()** y **logout()**.
 4. Rutas **/login** y **/logout**.
 5. Vista con formulario.
 6. Middleware **auth** para proteger secciones.
-

Paquetes oficiales:

Laravel Breeze → es simple (Blade o Vue/React). requiere correr **npm install** y **npm run dev** después de instalar Node.js para que se creen las vistas al estilo bootstrap, si no, puedes usarlo pero se verá crudo.

Laravel Jetstream → más avanzado (roles, teams, 2FA, etc.).

Laravel UI → ya trae plantillas bootstrap listas sin necesidad de compilar nada.

¿Qué hace Laravel UI?

Cuando instalás **Laravel UI** con **--auth**:

1. Te genera las **rutas** de login, register, logout.
2. Te crea las **vistas Blade** de autenticación (**resources/views/auth/**).
3. Te prepara el **User model** y migraciones.

El middleware **auth** no lo trae UI , ya viene con la instalación de Laravel.

Instalar Laravel ui

- 1 hacer backup de tus tablas
- 2 Borrar la tabla user
- 3 Cambiar la longitud por defecto de strings en Laravel para ello vamos a editar

`app/Providers/AppServiceProvider.php:`

```
use Illuminate\Support\Facades\Schema;

public function boot()
{
    Schema::defaultStringLength(191);
}
```

Esto hace que todos los campos **string** por defecto se creen con longitud **191** ($191 \times 4 = 764 < 1000$).

- 4 composer require laravel/ui
- 5 php artisan ui bootstrap --auth
- 6 php artisan migrate

- 7 editar resources/views/layouts/app.blade.php (en caso de no querer usar npm):

Buscá la parte donde está esto (probablemente en el `<head>`):
`@vite(['resources/sass/app.scss', 'resources/js/app.js'])`
y **borrala** o comentala.

- 8 Agregar Bootstrap por CDN

En el mismo `head`, poné:

`<!-- Bootstrap CSS desde CDN -->`

`<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css">`

`<!-- Opcional: FontAwesome para iconos -->`

`<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css">`

y antes del body

`<!-- Bootstrap JS con dependencias -->`

`<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>`

`<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js"></script>`

- 9 agregar en el nav var del layout/admin

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="{{ url('/') }}">MiApp</a>
  <ul class="navbar-nav ml-auto">
    @auth
      <li class="nav-item">
        <form method="POST" action="{{ route('logout') }}">
          @csrf
          <button type="submit" class="btn btn-link nav-link" style="display:inline; cursor:pointer;">
            <i class="fa fa-sign-out-alt"></i> Cerrar sesión
          </button>
        </form>
      </li>
    @endauth
    @guest
      <li class="nav-item">
        <a class="nav-link" href="{{ route('login') }}">
          <i class="fa fa-sign-in-alt"></i> Iniciar sesión
        </a>
      </li>
    @endguest
  </ul>
</nav>
```

Directivas a tener en cuenta

@guest

- Significa “**invitado**” (no autenticado).
- El bloque dentro de @guest ... @endguest solo se ejecuta si el usuario NO está logueado.

@auth

- Significa “**autenticado**”
- El bloque dentro de @auth ... @endauth solo se ejecuta si el usuario NO está logueado.

@guest → contenido visible **solo para no logueados**.

@auth → contenido visible **solo para logueados**.