UNIVERSIDAD TECNOLÓGICA DEL VALLE DE TOLUCA

Dirección de Carrera de Tecnologías de la Información y Comunicación

Técnico Superior Universitario en Tecnologías de la Información

Área Desarrollo de Software Multiplataforma

Aplicaciones WEB 4.0

Nombres del Estudiante

Octavio Duarte Villavicencio-222310422

Francisco Jesus Perez Medina-222310604

Dayana Salvador Sanchez-222310493

Grupo

TI DSM – 54

Cuatrimestre

5º

Lugar

Santa María Atarasquillo, Lerma, México.

Fecha de Entrega

Enero 2025

Descripción Técnica

Repositorio Usado
Link del video del uso del login

1. AuthController.php

Método home

- **Verifica si el usuario está autenticado**:
  - Usa Session::has('user') para comprobar si hay un usuario en sesión.
  - Si no hay usuario, redirige a la vista de login con un mensaje de error.
- **Obtiene el usuario de la sesión** con Session::get('user').
- **Carga la vista auth.home** y le pasa el usuario con compact('user').
- **Método showLoginForm():** Devuelve la vista del formulario de inicio de sesión.

```php
public function home()
{
    if (!Session::has('user')) {
        return redirect()->route('login')->withErrors(['error' => 'Debe iniciar sesión.']);
    }

    $user = Session::get('user');
    return view('auth.home', compact('user'));
}
```

Ilustración 1 Metodo home utilizado

Método showLoginForm

- **Muestra el formulario de inicio de sesión** llamando a la vista auth.login.

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

Ilustración 2 Metodo showLogin

Método login():

- **Recoge las credenciales enviadas desde el formulario** con $request->only('nombre', 'email', 'password').

- **Hace una solicitud HTTP GET a la API externa** (http://localhost:3000/usuarios) usando Http::get().
- **Maneja errores si la API falla**: Si la respuesta falla, devuelve al formulario de login con un error.
- **Convierte la respuesta JSON en un array de usuarios** con $response->json().
- **Busca un usuario que coincida con las credenciales** usando collect($users)->first():
    - Compara nombre, email y password de forma sensible a espacios y mayúsculas/minúsculas.
- **Si encuentra el usuario**, lo almacena en la sesión (Session::put('user', $user)) y redirige a home.
- **Si no lo encuentra**, redirige al formulario con un mensaje de error.

```php
    public function login(Request $request)
{
    // Verifica que los datos se estén enviando correctamente
    $credentials = $request->only('nombre', 'email', 'password');

    $response = Http::get('http://localhost:3000/usuarios');
    if ($response->failed()) {
        return back()->withErrors(['email' => 'Error al conectarse con el servidor.']);
    }

    $users = $response->json();

    // Buscar el usuario en la API
    $user = collect($users)->first(function ($user) use ($credentials) {
        return strtolower(trim($user['nombre'])) === strtolower(trim($credentials['nombre'])) &&
               strtolower(trim($user['email'])) === strtolower(trim($credentials['email'])) &&
               trim($user['password']) === trim($credentials['password']);
    });

    if ($user) {
        Session::put('user', $user);
        return redirect()->route('home');
    }

    return back()->withErrors(['email' => 'Las credenciales proporcionadas no son correctas.']);
}
```

Ilustración 3 Método Login

Método logout(): Elimina al usuario de la sesión y redirige al login.

```php
public function logout(Request $request)
{
    Session::forget('user');
    return redirect()->route('login');
}
```

Ilustración 4 Metodo Logout

2. Vista auth.home

Ilustración 5 Vista Home con rol Admin

- La vista extiende el layout principal ‘layouts.app’ que contiene la estructura principal de la aplicación.

```php
@extends('layouts.app')
```

Ilustración 6 Extension del archivo App.layout

- Utiliza el sistema de grillas de Bootstrap para centrar el contenido y limitarlo a 8 columnas en pantallas medianas, además de aplicar un efecto de sombra al card para mejorar la apariencia visual.

```html
<div class="container mt-4">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card shadow-lg">
```

Ilustración 7 Sistema para centrar el contenido y limitarlo a 8 columnas

```
<div class="card-header bg-success text-white text-center">
    <h4 class="mb-0">Dashboard</h4>
</div>
```

Ilustración 8 Configuración del encabezado

- El encabezado del Dashboard tiene fondo verde, texto blanco y centrado, con un título sin margen inferior.
- Verifica si existe un mensaje ‘status’ en la sesión y si lo hay, lo muestra en una alerta de éxito.

```
@if (session('status'))
    <div class="alert alert-success" role="alert">
        {{ session('status') }}
    </div>
@endif
```

Ilustración 9 Manejo de alertas

- Muestra el nombre del usuario y su rol almacenados en la sesión.

```
<h3 class="text-center">Bienvenido, {{ session('user')['nombre'] }}</h3>
<p class="text-center">Tu rol es: <strong>{{ session('user')['rol'] }}</strong></p>
```

Ilustración 10 Metodo para que se muestre el nombre del usuario

- Contiene una sección de funciones por rol:
  - Si el rol es administrador, se muestran funciones como crear nueva luminaria, ver listado, editar y ver detalles.

```
@if (session('user')['rol'] === 'admin')
    <h4 class="text-center">Funciones de Administrador</h4>
    <ul class="list-group">
        <li class="list-group-item text-center"><a href="{{ route('postes.create') }}" class="btn btn-outline-primary w-100">Crear Nueva Luminaria</a></li>
        <li class="list-group-item text-center"><a href="{{ route('postes.index') }}" class="btn btn-outline-primary w-100">Ver Listado de Luminarias</a></li>
        <li class="list-group-item text-center"><a href="#" class="btn btn-outline-primary w-100">Editar Luminaria (Ejemplo ID 1)</a></li>
        <li class="list-group-item text-center"><a href="{{ route('postes.show', 1) }}" class="btn btn-outline-primary w-100">Ver Detalles de Luminaria (Ejemplo ID 1)</a></li>
    </ul>
```

Ilustración 11 Funciones del rol administrador

- Si el rol es técnico, se muestran funciones como ver listado y ver detalles de luminarias.

```
@elseif (session('user')['rol'] === 'tecnico')
    <h4 class="text-center">Funciones de Técnico</h4>
    <ul class="list-group">
        <li class="list-group-item text-center"><a href="{{ route('postes.index') }}" class="btn btn-outline-secondary w-100">Ver Listado de Luminarias</a></li>
        <li class="list-group-item text-center"><a href="{{ route('postes.show', 1) }}" class="btn btn-outline-secondary w-100">Ver Detalles de Luminaria (Ejemplo ID 1)</a></li>
    </ul>
@endif
```

Ilustración 12 Funciones del rol técnico

3. Vista auth.login

- Estructura HTML con Bootstrap 5 para diseño responsivo.

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
```

Ilustración 13 Script de Bootstrap 5

- Un formulario POST que envía los datos a /login.

```
<div class="card-body">
    <form method="POST" action="/login">
        @csrf
        <div class="mb-3">
            <label for="nombre" class="form-label">Nombre</label>
            <input type="text" name="nombre" id="nombre" class="form-control" required>
        </div>
        <div class="mb-3">
            <label for="email" class="form-label">Email</label>
            <input type="email" name="email" id="email" class="form-control" required>
        </div>
        <div class="mb-3">
            <label for="password" class="form-label">Contraseña</label>
            <input type="password" name="password" id="password" class="form-control" required>
        </div>
        <div class="d-grid">
            <button type="submit" class="btn btn-primary">Ingresar</button>
        </div>
    </form>
```

Ilustración 14 Codigo del formulario

- Campos nombre, email y password.
- Token @csrf para seguridad.

```
@csrf
```

Ilustración 15 Token de seguridad

- Botón para iniciar sesión.

4. web.php (Rutas Web)

- Rutas definidas con Route::post, Route::get y Route::post para manejar login, logout y la vista home.

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\AuthController;
use App\Http\Controllers\PosteController;
use App\Http\Controllers\SensorController;
use App\Http\Controllers\AlertaController;
use App\Http\Controllers\UserController;


Route::post('/login', [AuthController::class, 'login']);
Route::get('/', [AuthController::class, 'showLoginForm'])->name('login');
Route::post('/logout', [AuthController::class, 'logout'])->name('logout');
Route::get('/home', [AuthController::class, 'home'])->name('home');
```

Ilustración 16 Rutas de AuthController

El código implementa un sistema básico de autenticación en Laravel, utilizando sesiones y una API externa para validar usuarios, con vistas personalizadas según el rol.

5. ¿Qué hizo cada uno de los integrantes?

INTEGRANTE	RESPONSABILIDAD
Octavio Duarte Villavicencio	Pruebas y documentación
Dayana Sánchez Salvador	Diseño y desarrollo del frontend con Bootstrap
Francisco Jesus Perez Medina	Desarrollo del backend en Laravel y configuración de roles