Sistema de Gestión Educativa

Descripción

Este sistema de gestión educativa ha sido desarrollado utilizando **Laravel**, un framework de PHP que facilita la creación de aplicaciones web robustas y seguras.

El sistema permite la autenticación de usuarios, la gestión de proyectos educativos, el soporte para múltiples idiomas y la comunicación con otros sistemas mediante una API REST.

Características Implementadas

1. Landing Page y Autenticación

- Se ha diseñado una página de inicio (**Landing Page**) con un diseño atractivo y fácil de usar.
- Existen vistas diferenciadas para usuarios autenticados y no autenticados.
- El sistema de autenticación permite registrar usuarios, iniciar sesión y cerrar sesión de manera segura.
- Se utiliza **Laravel Breeze**, un paquete que facilita la creación de sistemas de autenticación en Laravel.
- El diseño es responsive, lo que significa que se adapta a distintos dispositivos (móviles, tabletas y ordenadores).

2. Gestón de Proyectos (CRUD)

CRUD significa **Crear, Leer, Actualizar y Eliminar**. En este caso, se implementa para la gestión de proyectos:

- Se pueden crear proyectos con los siguientes datos:
  - **Título** (texto)
  - **Horas previstas** (número entero)
  - **Fecha de comienzo** (fecha)
- Los formularios incluyen validaciones para evitar errores en la introducción de datos.
- Se han implementado mensajes de confirmación y alertas utilizando **SweetAlert2**.
- Solo los usuarios autenticados pueden acceder a la gestión de proyectos.

3. Relaciones entre Datos (1:N)

- Un **usuario** puede tener varios **proyectos**.
- Un **alumno** puede hablar varios **idiomas**.

4. Soporte Multilingüe

- Se ha implementado un selector de idioma en la barra de navegación.

- Los idiomas disponibles son:
    - **Español** (es)
    - **Inglés** (en)
    - **Francés** (fr)
    - **Alemán** (de)
- Se utiliza un middleware de localización para gestionar los cambios de idioma.
- Todas las traducciones han sido incluidas en archivos dedicados.

5. API REST

Una **API REST** permite que otras aplicaciones se conecten al sistema y realicen operaciones. En este caso:

- Se han implementado endpoints protegidos con **Laravel Sanctum**.
- Permite realizar operaciones CRUD sobre los proyectos.
- Las respuestas se devuelven en formato JSON.
- Se requiere autenticación mediante **tokens**.

6. Mensajes y Notificaciones

- Se muestran **mensajes flash** para notificar al usuario sobre el éxito o error de sus acciones.
- Se han integrado confirmaciones con **SweetAlert2** para mejorar la experiencia del usuario.
- Se han implementado mensajes de validación para alertar de errores en los formularios.

Estructura del Proyecto

El código del sistema se organiza en distintas partes para facilitar su gestión.

Modelos

Los modelos representan los datos en la base de datos:

- `User`: Representa a los usuarios registrados.
- `Project`: Representa los proyectos creados.
- `Alumno`: Representa a los alumnos.
- `Idioma`: Representa los idiomas.

Controladores

Los controladores gestionan las acciones de la aplicación:

- `ProjectController`: Maneja las operaciones CRUD de proyectos.
- `LanguageController`: Gestiona los idiomas.
- `Api/ProjectController`: Maneja las peticiones API para proyectos.

Vistas

Las vistas son archivos que muestran la interfaz de usuario:

- **Layouts** (Diseño base para todas las páginas):
  - `app.blade.php`: Contiene la estructura general de la aplicación.
  - `navigation.blade.php`: Contiene la barra de navegación.
- **Componentes** (Elementos reutilizables):
  - `nav-link.blade.php`: Enlaces de navegación.
  - `responsive-nav-link.blade.php`: Enlaces adaptados para móviles.
- **Páginas** (Secciones del sitio web):
  - `welcome.blade.php`: Página de inicio.
  - `dashboard.blade.php`: Panel de control.
  - `projects/*.blade.php`: Páginas de gestión de proyectos.

Middleware

Los **middlewares** son funciones intermedias que controlan el flujo de datos:

- `Localization`: Gestiona el cambio de idioma.
- `Authenticate`: Protege rutas para que solo usuarios autenticados puedan acceder.
- `VerifyCsrfToken`: Evita ataques CSRF (seguridad contra peticiones maliciosas).

Instalación

Para instalar el sistema en tu ordenador sigue estos pasos:

1. **Clonar el repositorio**

```
git clone <URL_DEL_REPOSITORIO>
cd nombre-del-proyecto
```

2. **Instalar dependencias**

```
composer install
npm install
```

3. **Configurar el entorno**

```
cp .env.example .env
php artisan key:generate
```

4. **Configurar la base de datos**

- Edita el archivo `.env` y configura la conexión a tu base de datos.

5. **Ejecutar migraciones**

```
php artisan migrate
```

6. **Compilar los archivos estáticos**

```
npm run dev
```

7. **Iniciar el servidor**

```
php artisan serve
```

Uso

1. Accede a la aplicación a través del navegador en `http://127.0.0.1:8000`
2. Regístrate como nuevo usuario o inicia sesión.
3. Gestiona los proyectos desde el **dashboard**.
4. Cambia el idioma usando el selector en la barra de navegación.
5. Accede a la API usando tokens de autenticación.

API Endpoints

La API REST permite interactuar con los proyectos mediante peticiones HTTP:

- `GET /api/projects` -> Lista todos los proyectos.
- `POST /api/projects` -> Crea un nuevo proyecto.
- `GET /api/projects/{id}` -> Muestra un proyecto específico.
- `PUT /api/projects/{id}` -> Actualiza un proyecto.
- `DELETE /api/projects/{id}` -> Elimina un proyecto.

Contribuir

Si deseas mejorar el sistema:

1. **Haz un fork** del repositorio.
2. **Crea una nueva rama** para tu función o corrección.
3. **Sube los cambios** y realiza un **pull request**.

Licencia