**Información General del Proyecto**

* **Nombre del Proyecto: Sistema de Gestión de Proyectos Académicos (PIA)**
* **Framework utilizado: Laravel 11**
* **Lenguaje de programación: PHP 8.x**
* **Base de Datos: PostgreSQL**
* **Autenticación: Laravel Breeze**
* **Seguridad y permisos: Spatie Laravel-Permission**
* **Diseño Frontend: Tailwind CSS**
* **Editor de desarrollo: Visual Studio Code**
* **Control de versiones: Git y GitHub**

**Entidades Principales del Sistema**

**El sistema gestiona la información relacionada con los proyectos académicos de instituciones educativas. Las entidades obligatorias implementadas son:**

1. **Instituciones**
2. **Facultades**
3. **Departamentos**
4. **Programas**
5. **Asignaturas**
6. **Docentes**
7. **Estudiantes**
8. **Evaluadores *(se registran incluso si ya son docentes)***
9. **Tipos de Proyecto**
10. **Proyectos**
11. **Evaluaciones**

**Entidades del Subsistema de Autenticación y Seguridad**

1. **Usuarios**
2. **Roles**
3. **Permisos**

**Funcionalidades del Sistema**

* **Registro e inicio de sesión de usuarios (Laravel Breeze)**
* **Recuperación de contraseña mediante correo electrónico**
* **Actualización de contraseña desde la cuenta de usuario**
* **Eliminación de cuenta personal**
* **CRUD completo de usuarios desde el panel de administrador**
* **Gestión completa de roles y permisos con Spatie**
* **Visualización y administración de proyectos académicos por docentes, evaluadores y estudiantes**
* **Evaluación de proyectos por parte de evaluadores**
* **Menú de navegación adaptado según el rol del usuario**

**Gestión de Usuarios y Seguridad**

**El sistema incluye un completo subsistema de autenticación y control de accesos:**

**Laravel Breeze (Autenticación)**

**Se utilizó el paquete Breeze, que implementa:**

* **Registro de nuevos usuarios**
* **Login y logout**
* **Restablecimiento de contraseña mediante email**
* **Actualización de contraseña**
* **Eliminación de cuenta personal**
* **Interfaz moderna con Tailwind CSS**

**Spatie Laravel-Permission (Roles y Permisos)**

**Este paquete permite:**

* **Crear, asignar y editar roles**
* **Asignar permisos personalizados a roles o usuarios individuales**
* **Controlar acceso a rutas y vistas específicas según permisos**
* **Permitir una navegación y uso adaptado al perfil de cada usuario**

**Rol Permisos asignados**

| **Admin** | **CRUD completo de todas las entidades, usuarios, roles y permisos** |
| --- | --- |
| **Docente** | **Crear y editar proyectos, asignar estudiantes** |
| **Evaluador** | **Ver y calificar proyectos asignados** |
| **Estudiante** | **Ver sus propios proyectos y calificaciones** |

**Menú de Opciones del Sistema**

**El sistema cuenta con un menú dinámico adaptado a los permisos del usuario. Algunas opciones generales son:**

* **Dashboard**
* **Gestión de Proyectos**
* **Asignaturas**
* **Estudiantes**
* **Docentes**
* **Evaluadores**
* **Tipos de Proyecto**
* **Evaluaciones**
* **Administración (Roles, Usuarios, Permisos)**
* **Perfil / Cambiar contraseña / Eliminar cuenta**
* **Cerrar sesión**

**Experiencia Técnica de Desarrollo**

**Durante el desarrollo del proyecto se presentaron diversos retos, principalmente en la integración entre el backend (controladores y base de datos en PostgreSQL) y el frontend. La implementación de Breeze fue sencilla gracias a su integración nativa en Laravel, mientras que el paquete Spatie exigió una comprensión más profunda sobre control de accesos, rutas protegidas y asignación de permisos. En general, se logró una arquitectura sólida y escalable.**

**En mi caso, trabajé principalmente en la parte visual del sistema, incluyendo la maquetación HTML, la estructura de las vistas con Blade, el diseño responsivo con Tailwind CSS y la organización general del contenido gráfico (como los logos e imágenes). (**[**JUAN DAVID ARBOLEDA ARANGO**](mailto:juan.arboleda417@pascualbravo.edu.co)**)**

**Importancia del Trabajo en Equipo**

**El trabajo en equipo fue esencial para distribuir las responsabilidades y avanzar de manera organizada. Cada miembro del equipo se encargó de un módulo o capa del sistema, lo cual permitió que el desarrollo fuera más eficiente. La comunicación constante y las reuniones de sincronización evitaron errores y ayudaron a integrar las distintas partes de forma efectiva.**

**Uso de Metodologías de Trabajo Colaborativo y GIT**

**El uso de Git y GitHub nos permitió mantener el control de versiones del código, gestionar cambios y realizar pruebas sin afectar la versión principal. Trabajamos con ramas por funcionalidad y utilizamos *pull requests* para revisar los cambios antes de integrarlos. Esta práctica es clave para cualquier equipo de desarrollo moderno, ya que mejora la organización, previene conflictos y deja un historial claro del progreso del proyecto.**