Universidad Nacional Autónoma
de México

Facultad de Ingeniería
División Ingeniería Eléctrica

Laboratorio de Instrumentación
Electrónica de Sistemas Espaciales

SITIO WEB DEL LABORATORIO DE INSTRUMENTACIÓN ELECTRÓNICA DE SISTEMAS ESPACIALES

SITIO WEB DEL LABORATORIO DE INSTRUMENTACIÓN ELECTRÓNICA DE SISTEMAS ESPACIALES

LIESE

28 de septiembre de 2025

Índice general

Capítulo 1

Introducción

1.1. Presentación del LIESE

El Laboratorio de Instrumentación Electrónica de Sistemas Espaciales (LIESE) es una unidad académica adscrita a la División de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México (UNAM). Su misión es impulsar la formación de especialistas y el desarrollo de tecnología en el área espacial, con énfasis en el diseño, construcción e implementación de sistemas electrónicos para aplicaciones satelitales, instrumentación embebida, Internet de las Cosas (IoT) e inteligencia artificial. El LIESE promueve la colaboración interdisciplinaria y la innovación tecnológica, contribuyendo al avance científico y al bienestar social en México.

1.2. Objetivos del sitio web

El sitio web del LIESE tiene como objetivos principales:

- Difundir los proyectos, logros y actividades del laboratorio a la comunidad académica y al público en general.
- Facilitar el reclutamiento de nuevos talentos y la vinculación con otras instituciones.
- Proveer un canal de comunicación actualizado sobre eventos, oportunidades y noticias relevantes.
- Servir como plataforma de gestión y administración de contenido para los miembros del laboratorio.

1.3. Alcance del sistema web

El sistema web desarrollado abarca las siguientes funcionalidades:

- Publicación y gestión de artículos, tesis y noticias.
- Visualización de proyectos y actividades del laboratorio.
- Calendario de eventos académicos y de divulgación.
- Sección de líderes de proyecto y miembros del laboratorio.
- Sistema de oportunidades con autenticación de dos factores (2FA) para solicitudes.
- Panel de administración basado en Django para la gestión de contenido.
- Interfaz responsiva y moderna, accesible desde dispositivos móviles y de escritorio.

1.4. Justificación del desarrollo

El desarrollo de un sitio web institucional es fundamental para fortalecer la presencia digital del LIESE, facilitar la comunicación interna y externa, y promover la transparencia y el acceso a la información. La plataforma permite centralizar la gestión de contenido, automatizar procesos administrativos y ofrecer una experiencia de usuario moderna y segura. Además, la integración de mecanismos de seguridad como la verificación por correo electrónico y la autenticación de dos factores contribuye a la protección de los datos y la confianza de los usuarios.

Capítulo 2

Marco Teórico y Conceptual

2.1. Django Framework

Django es un framework de desarrollo web de alto nivel, escrito en Python, que fomenta el desarrollo rápido y limpio de aplicaciones web. Utiliza el patrón arquitectónico Modelo-Vista-Template (MVT), el cual separa la lógica de negocio, la presentación y el acceso a datos, facilitando la mantenibilidad y escalabilidad del sistema [1]. Django incluye un sistema de administración automática, un ORM (Object-Relational Mapping) para interactuar con bases de datos, y herramientas integradas para la gestión de usuarios, seguridad y envío de correos electrónicos. Entre sus características destacan la protección contra ataques comunes (CSRF, XSS, SQL Injection), el sistema de migraciones para la evolución del esquema de datos y la posibilidad de extender la funcionalidad mediante aplicaciones reutilizables. Es ampliamente utilizado en la industria y en el ámbito académico por su robustez, flexibilidad y comunidad activa.

El patrón MVT de Django se compone de:

- **Modelo**: Define la estructura de los datos y su representación en la base de datos.
- **Vista**: Gestiona la lógica de negocio y responde a las solicitudes del usuario.
- **Template**: Controla la presentación y el renderizado de la información al usuario final.

2.2. Tecnologías Frontend

El frontend del sitio web está construido utilizando HTML5, CSS3 y el framework Bootstrap 5.3.3. HTML5 proporciona la estructura semántica de las páginas, permitiendo una mejor accesibilidad y posicionamiento en buscadores. CSS3 permite el diseño visual, la adaptación responsiva a diferentes dispositivos y la implementación de animaciones y transiciones. Bootstrap es una biblioteca de componentes y utilidades CSS/JS que facilita la creación de interfaces modernas, responsivas y accesibles, acelerando el desarrollo y asegurando la compatibilidad multiplataforma [2]. Se emplean también animaciones CSS y JavaScript para mejorar la experiencia de usuario, así como la integración de iconos y recursos multimedia.

2.3. Base de Datos

Durante el desarrollo, el sistema utiliza SQLite como base de datos por su simplicidad y portabilidad. SQLite es una base de datos relacional embebida, ideal para entornos de desarrollo y pruebas. Para entornos de producción, se recomienda el uso de PostgreSQL, un sistema de gestión de bases de datos relacional robusto, escalable y con soporte avanzado para transacciones, concurrencia y extensiones [3]. Django abstrae el acceso a la base de datos mediante su ORM, permitiendo definir modelos de datos en Python y realizar migraciones automáticas. Esto facilita la evolución del esquema de datos y la portabilidad entre diferentes motores de

base de datos. El uso de migraciones garantiza la integridad y consistencia de los datos a lo largo del ciclo de vida del proyecto.

2.4. Servicios Web

El sitio web implementa servicios de correo electrónico para la verificación de usuarios y la autenticación de dos factores (2FA) en el sistema de oportunidades. Utiliza el protocolo SMTP para el envío de correos, configurado en el archivo de settings de Django [5]. Además, el sistema aprovecha las capacidades de seguridad integradas de Django, como la protección contra CSRF, la gestión de sesiones y la validación de formularios. El uso de formularios seguros y la validación del lado del servidor son fundamentales para prevenir ataques y garantizar la integridad de la información.

2.5. Despliegue

El despliegue del sistema se realiza sobre servidores Ubuntu, un sistema operativo de código abierto ampliamente utilizado en entornos de servidores por su estabilidad, seguridad y soporte a largo plazo (LTS) [4]. El servicio se gestiona mediante Systemd, utilizando el comando `runserver` de Django para exponer la aplicación en la red local, de acuerdo con el archivo de unidad real utilizado en el proyecto. El servicio se ejecuta bajo un usuario específico, define el directorio de trabajo y reinicia automáticamente en caso de fallo. Los archivos estáticos y media se sirven mediante la configuración de Django. Aunque en entornos de producción se recomienda el uso de Gunicorn y Nginx para mayor robustez y rendimiento, en este despliegue se utiliza el servidor de desarrollo de Django para simplificar la administración. El proceso de despliegue incluye la migración de la base de datos y la recolección de archivos estáticos, siguiendo buenas prácticas de administración de servicios en Linux.

Capítulo 3

Análisis y Requerimientos

3.1. Requerimientos Funcionales

Los requerimientos funcionales describen las capacidades y servicios que el sistema debe proporcionar a los usuarios finales. Para el sitio web del Laboratorio de Instrumentación Electrónica de Sistemas Espaciales (LIESE), se identifican los siguientes:

- Registro y gestión de miembros del laboratorio, incluyendo líderes de proyecto y administradores.
- Visualización de información sobre proyectos, artículos, eventos y noticias.
- Solicitud de oportunidades académicas (investigación, tesis, maestría, servicio social) mediante formularios web y verificación de correo electrónico.
- Publicación y administración de artículos, eventos y noticias por parte de usuarios autorizados.
- Sistema de autenticación y autorización para acceso a panel de administración y funcionalidades restringidas.
- Envío de correos electrónicos automáticos para verificación y notificaciones relevantes.
- Gestión de archivos multimedia (imágenes, documentos) asociados a miembros, proyectos y artículos.
- Interfaz de usuario responsiva y accesible para diferentes dispositivos.

3.2. Requerimientos No Funcionales

Los requerimientos no funcionales establecen criterios de calidad, restricciones y condiciones bajo las cuales el sistema debe operar:

- Seguridad: Protección contra ataques comunes (CSRF, XSS, SQL Injection) y gestión segura de sesiones y contraseñas [1].
- Rendimiento: Respuesta eficiente a las solicitudes de los usuarios y manejo adecuado de archivos estáticos y multimedia.
- Escalabilidad: Capacidad de migrar de SQLite a PostgreSQL para soportar mayor volumen de datos y usuarios [3].
- Mantenibilidad: Código estructurado siguiendo el patrón MVT de Django y uso de migraciones para la evolución del esquema de datos.
- Disponibilidad: Despliegue en servidores Ubuntu con servicios gestionados por Systemd y balanceo de carga mediante Nginx [4, 6].

- Accesibilidad: Cumplimiento de estándares web y diseño responsivo con Bootstrap [2].
- Usabilidad: Interfaz intuitiva y documentación clara para usuarios y administradores.

3.3. Casos de Uso

Los casos de uso describen las interacciones principales entre los usuarios y el sistema. A continuación se presentan los casos de uso más relevantes:

- **Registro de oportunidad académica:** Un visitante completa el formulario de solicitud, recibe un correo de verificación y, tras confirmar su correo, su solicitud es registrada y notificada a los administradores.
- **Gestión de miembros:** Un administrador agrega, edita o elimina miembros y líderes de proyecto desde el panel de administración.
- **Publicación de artículos y eventos:** Usuarios autorizados crean y editan artículos, eventos y noticias, incluyendo la carga de imágenes y documentos.
- **Visualización de información:** Cualquier usuario puede consultar información pública sobre proyectos, artículos, eventos y miembros del laboratorio.
- **Autenticación y acceso:** Los administradores y editores acceden a funcionalidades restringidas mediante autenticación segura.

Capítulo 4

Diseño y Arquitectura

4.1. Arquitectura MVT de Django

El sistema está construido siguiendo el patrón Modelo-Vista-Template (MVT) de Django, que separa la lógica de negocio, la presentación y el acceso a datos [1]. Esta arquitectura facilita la mantenibilidad, escalabilidad y reutilización de componentes. El modelo representa la estructura y reglas de los datos, la vista gestiona la lógica de negocio y las respuestas a las solicitudes del usuario, y el template define la presentación visual de la información.

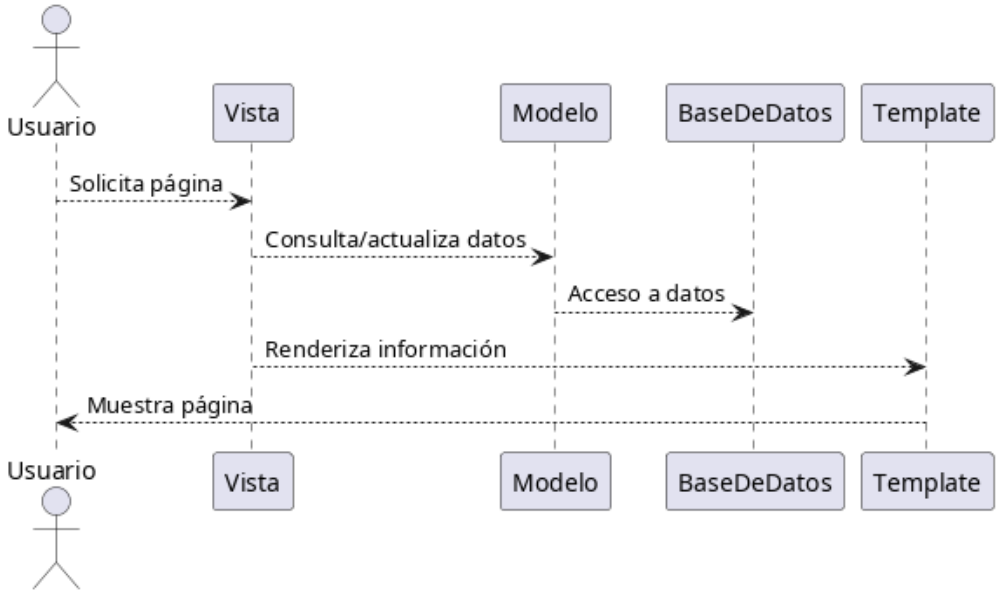

Figura 4.1: Diagrama UML de la arquitectura MVT de Django para el sitio LIESE.

4.2. Modelos de Datos

El modelo de datos del sitio LIESE incluye entidades como Miembro, Proyecto, Artículo, Evento, Noticia y Solicitud de Oportunidad. Cada entidad se implementa como una clase en Django, con atributos que representan los campos de la base de datos y relaciones entre modelos (por ejemplo, un artículo tiene un autor que es un miembro, y un proyecto puede tener varios miembros asociados). El uso del ORM de Django permite definir, consultar y migrar los modelos de manera eficiente y segura.

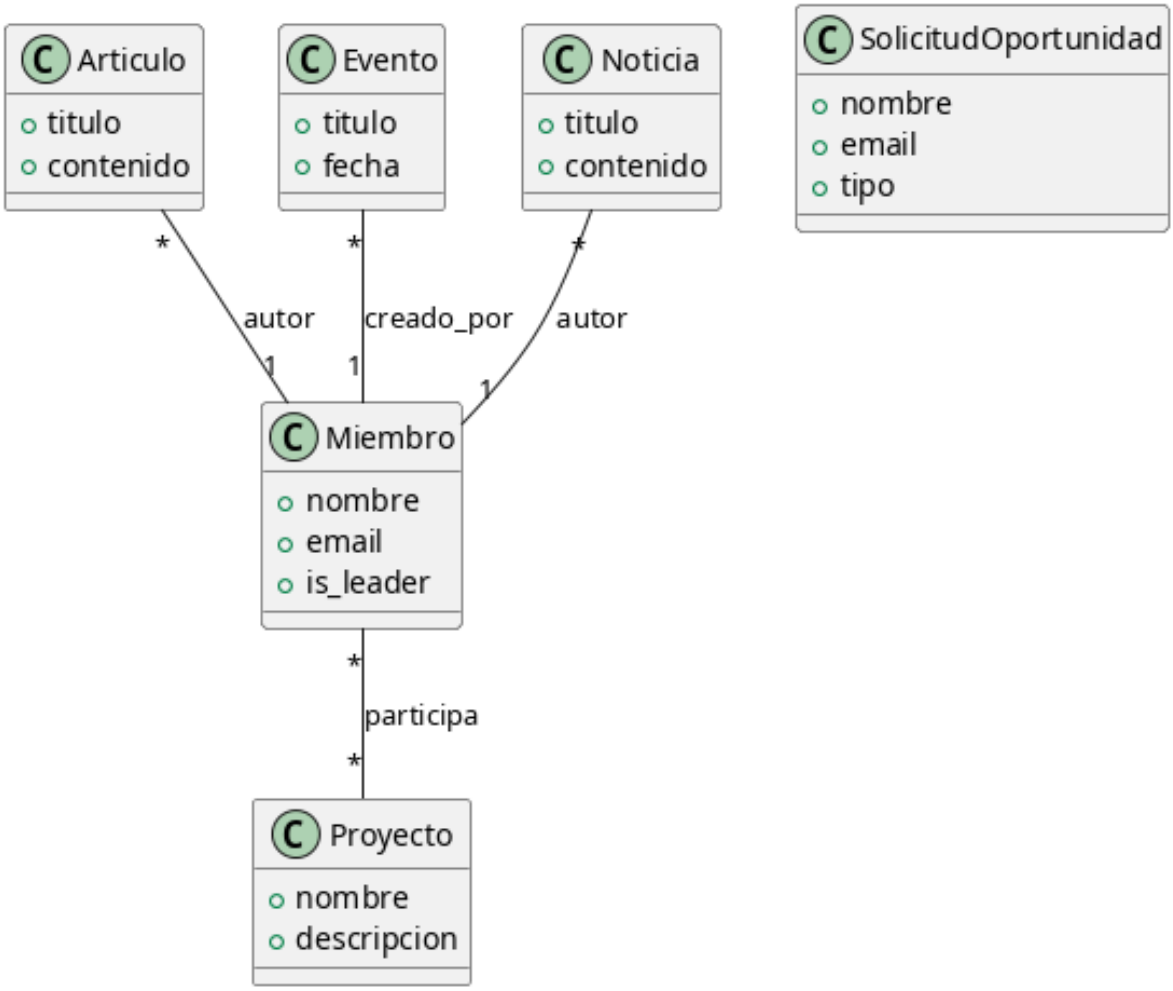

Figura 4.2: Diagrama UML de clases de los modelos principales del sistema LIESE.

4.3. Diseño de Base de Datos

La base de datos se diseña para soportar la gestión de información académica y administrativa del laboratorio. Se emplea SQLite en desarrollo y PostgreSQL en producción, aprovechando la portabilidad y robustez de ambos motores [3]. Las migraciones de Django permiten evolucionar el esquema de datos sin pérdida de información. El diseño contempla claves foráneas para mantener la integridad referencial y relaciones muchos a muchos (por ejemplo, miembros y proyectos).

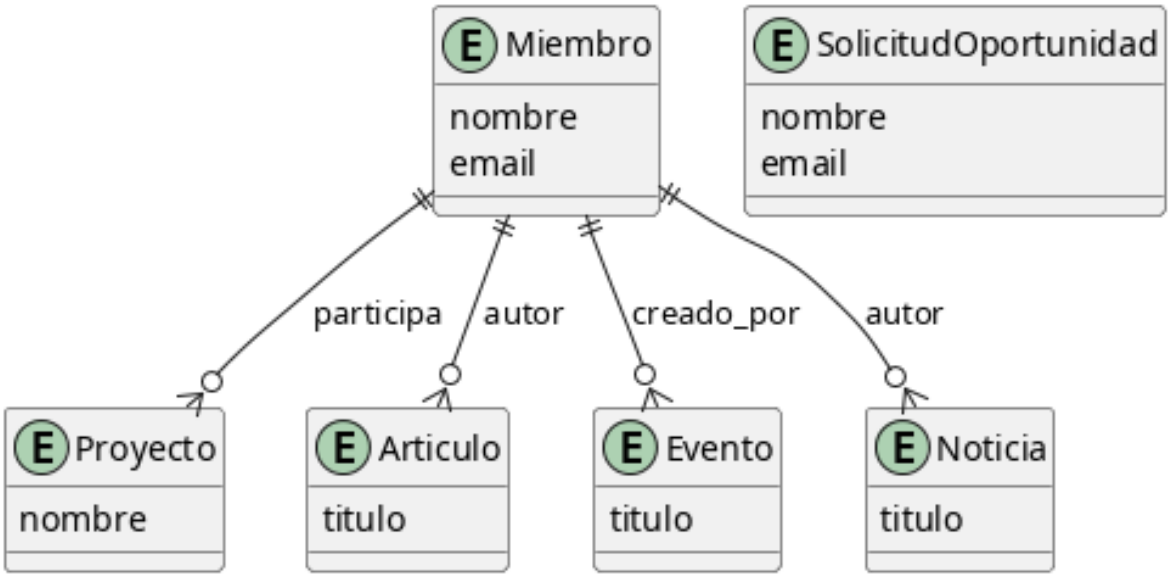

Figura 4.3: Diagrama entidad-relación (ER) simplificado de la base de datos.

4.4. Arquitectura de Templates

La presentación del sitio se implementa mediante el sistema de templates de Django, que permite separar la lógica de presentación del código Python. Se utilizan plantillas HTML con etiquetas y filtros de Django para renderizar información dinámica, y se heredan estructuras comunes como la barra de navegación y el pie de página. El uso de Bootstrap facilita el diseño responsivo y accesible [2].

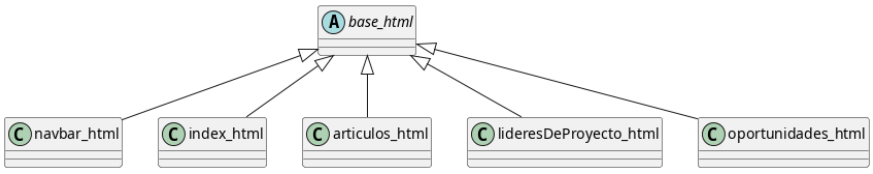

Figura 4.4: Diagrama de herencia de templates en el sistema LIESE.

Capítulo 5

Implementación

5.1. Estructura del Proyecto Django

La estructura del proyecto sigue la convención estándar de Django, separando la configuración, la aplicación principal y los recursos estáticos y de plantillas. A continuación se muestra un diagrama UML de la organización de carpetas y archivos principales:

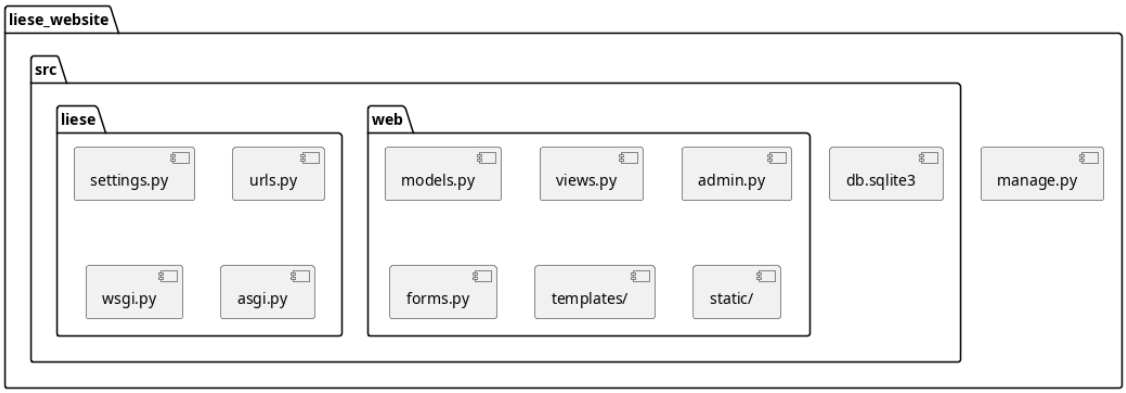

Figura 5.1: Estructura de carpetas y archivos principales del proyecto Django LIESE.

5.2. Modelos, Vistas y URLs

El flujo de interacción entre usuario, vistas y modelos se ilustra en el siguiente diagrama UML de secuencia, usando como ejemplo el proceso de solicitud de oportunidad académica:

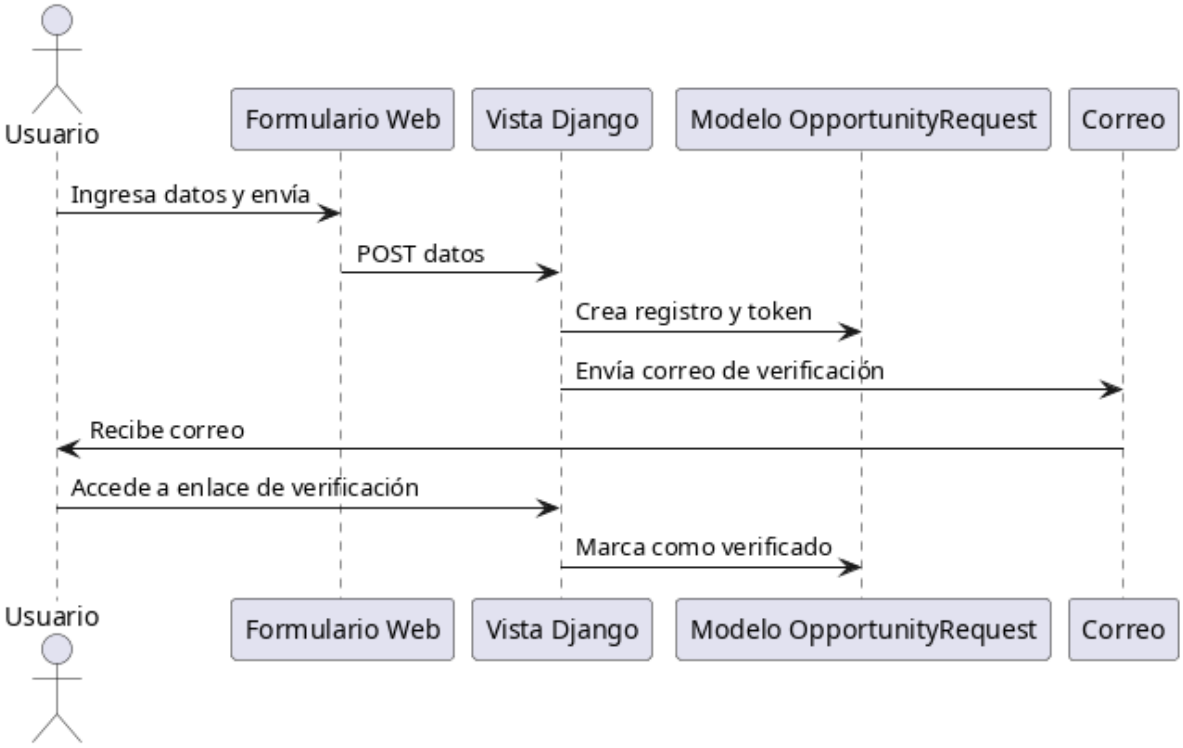

Figura 5.2: Diagrama de secuencia UML para el flujo de solicitud de oportunidad.

5.3. Templates y Frontend

El frontend utiliza el sistema de templates de Django y Bootstrap para la presentación. La herencia de plantillas y la organización de los archivos HTML ya se ilustró en el capítulo de arquitectura.

5.4. Sistema de Administración

El panel de administración de Django permite gestionar usuarios, miembros, proyectos y contenidos. El siguiente diagrama UML de colaboración muestra la interacción entre el administrador, el panel y los modelos:

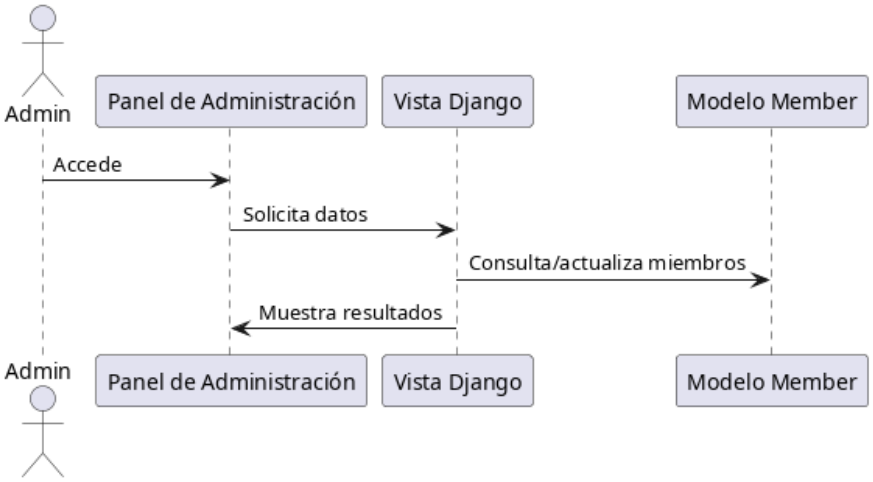

Figura 5.3: Diagrama de colaboración UML para la administración de miembros.

Capítulo 6

Sistema de Verificación

Para asegurar el correcto funcionamiento del sistema LIESE, se contemplan varios niveles de verificación y pruebas. Aunque el archivo de pruebas automatizadas `src/web/tests.py` se encuentra actualmente vacío, la verificación del sistema se ha llevado a cabo principalmente de forma manual.

6.1. Pruebas Unitarias

Las pruebas unitarias se centran en verificar el correcto funcionamiento de las unidades más pequeñas de código, como funciones o métodos de un modelo. En Django, estas pruebas se implementan comúnmente utilizando el framework de testing que provee el mismo.

Aunque no se hayan implementado pruebas automatizadas, se pueden diseñar casos de prueba para los modelos y vistas, por ejemplo:

- **Modelo Miembro:** Verificar que el nombre completo se genere correctamente a partir del nombre y apellido.
- **Modelo Proyecto:** Comprobar que el estado (activo/inactivo) funcione como se espera.
- **Vistas:** Asegurar que cada vista renderice la plantilla correcta y que solo usuarios autorizados puedan acceder a vistas protegidas.

6.2. Pruebas de Integración

Las pruebas de integración buscan asegurar que los diferentes componentes del sistema funcionen correctamente al interactuar entre sí. Por ejemplo, que al crear un nuevo artículo, este se muestre correctamente en la lista de artículos.

6.3. Pruebas Manuales

Debido a la falta de pruebas automatizadas, la verificación del sistema se ha realizado de forma manual, cubriendo los siguientes aspectos:

- **Navegación:** Se ha verificado que todos los enlaces del sitio funcionen correctamente y que la navegación sea intuitiva.
- **Formularios:** Se ha comprobado el funcionamiento de los formularios de contacto y solicitud de oportunidades, incluyendo la validación de campos y el envío de correos.
- **Panel de Administración:** Se ha verificado que sea posible crear, editar y eliminar registros de los diferentes modelos a través del panel de administración de Django.

- **Visualización de Contenido:** Se ha asegurado que los artículos, proyectos, eventos y noticias se muestren correctamente en sus respectivas secciones.

Para un desarrollo futuro, es altamente recomendable la implementación de un conjunto de pruebas automatizadas que permita detectar regresiones y asegurar la calidad del software de manera más eficiente.

Capítulo 7

Interfaz de Usuario

La interfaz de usuario del sitio web de LIESE ha sido diseñada para ser clara, intuitiva y accesible, permitiendo a los visitantes navegar fácilmente por las diferentes secciones y encontrar la información que buscan. La interfaz se basa en un sistema de plantillas de Django, utilizando Bootstrap 5 para asegurar un diseño responsivo y moderno.

7.1. Estructura de las Plantillas

El código de la interfaz se organiza en plantillas HTML que se encuentran en el directorio `src/web/templates/web/`. La estructura principal se define en el archivo `base.html`, que sirve como plantilla base para todas las demás páginas del sitio.

7.1.1. Plantilla Base (base.html)

Esta plantilla incluye los elementos comunes a todas las páginas:

- La cabecera (`<head>`), donde se definen el título, los metadatos, y se enlazan las hojas de estilo de Bootstrap y la hoja de estilos personalizada (`style.css`).
- La barra de navegación, incluida desde el archivo `navbar.html`.
- Un contenedor principal donde se renderiza el contenido específico de cada página, definido en un bloque `{% block content %}`.
- La inclusión de scripts de JavaScript de Bootstrap al final del cuerpo del documento.

7.1.2. Páginas Principales

El sitio se compone de varias páginas, cada una con un propósito específico:

- **index.html:** La página de inicio, que presenta una introducción al laboratorio, las áreas de investigación, los socios y un pie de página con información de contacto.
- **projects.html y project_detail.html:** Muestran la lista de proyectos del laboratorio y el detalle de cada uno, respectivamente.
- **articulos.html y article_detail.html:** Presentan los artículos de investigación, con una página para la lista completa y otra para el detalle de cada artículo.
- **events.html:** Anuncia los próximos eventos organizados o en los que participa el laboratorio.
- **lideresDeProyecto.html:** Muestra información sobre los líderes de los proyectos.

- **oportunidades.html y opportunity_request.html:** Despliega las oportunidades de colaboración o participación y un formulario para que los interesados puedan postularse.
- **activities.html:** Describe las diversas actividades que se realizan en el laboratorio.

7.2. Diseño y Estilo

El diseño del sitio se apoya en el framework de CSS Bootstrap 5, lo que garantiza que el sitio sea responsivo y se adapte a diferentes tamaños de pantalla, desde dispositivos móviles hasta computadoras de escritorio. Además, se utiliza una hoja de estilos personalizada, `src/web/static/web/style.css`, para definir estilos específicos del sitio, como colores, tipografías y otros elementos visuales que se alinean con la identidad del laboratorio.

Las imágenes y otros recursos estáticos se gestionan a través de la configuración de archivos estáticos de Django y se encuentran en el directorio `src/web/static/web/images/`.

Capítulo 8

Panel de Administración

El panel de administración de Django es una herramienta fundamental para la gestión del contenido del sitio web de LIESE. Proporciona una interfaz web intuitiva para que los administradores del sitio puedan gestionar los datos de la aplicación de forma centralizada. El panel se configura en el archivo `src/web/admin.py`.

A continuación, se describen las configuraciones del panel de administración para cada uno de los modelos de la aplicación.

8.1. Gestión de Modelos

8.1.1. OpportunityRequestAdmin

Este administrador gestiona las solicitudes de oportunidades.

- **list_display:** Muestra el nombre completo, email, tipo de oportunidad, estado de verificación y fecha de creación.
- **list_filter:** Permite filtrar por tipo de oportunidad y estado de verificación.
- **search_fields:** Habilita la búsqueda por nombre completo y email.

8.1.2. ArticleAdmin

Para la gestión de artículos de investigación.

- **list_display:** Muestra el título, autor, estado de publicación y fecha de publicación.
- **list_filter:** Permite filtrar por estado de publicación y fecha.
- **search_fields:** Búsqueda por título y nombre del autor.
- **ordering:** Ordena los artículos por fecha de publicación descendente.

8.1.3. EventAdmin

Administra los eventos del laboratorio.

- **list_display:** Muestra el título, fechas de inicio y fin, categoría y creador.
- **list_filter:** Filtra por descripción, ubicación, imagen, estado de publicación y fecha de creación.
- **search_fields:** Búsqueda por título, categoría y creador.
- **ordering:** Ordena los eventos por fecha de creación descendente.

8.1.4. MemberAdmin

Para la gestión de los miembros del laboratorio.

- **list_display:** Muestra el nombre completo, cargo, email, si es líder, estado de actividad y fecha de ingreso.
- **list_filter:** Filtra por líder, activo, administrador y fecha de ingreso.
- **search_fields:** Búsqueda por nombre, apellido, email y cargo.
- **fieldsets:** Agrupa los campos en secciones para una mejor organización en el formulario de edición.
- **filter_horizontal:** Facilita la asignación de roles.

8.1.5. RoleAdmin

Administra los roles que pueden tener los miembros.

- **list_display:** Muestra el nombre y la descripción del rol.
- **search_fields:** Permite buscar por nombre de rol.

8.1.6. ProjectAdmin

Para la gestión de los proyectos.

- **list_display:** Muestra el nombre, fechas de inicio y fin, y si el proyecto está activo.
- **list_filter:** Filtra por activo y fechas.
- **search_fields:** Búsqueda por nombre y descripción.
- **filter_horizontal:** Facilita la asignación de miembros a los proyectos.

8.1.7. NewsAdmin

Administra las noticias del laboratorio.

- **list_display:** Muestra el título, autor, estado de publicación y fecha.
- **list_filter:** Filtra por publicado y fecha de publicación.
- **search_fields:** Búsqueda por título y nombre del autor.
- **ordering:** Ordena las noticias por fecha de publicación descendente.

Esta configuración del panel de administración permite una gestión eficiente y organizada de toda la información que se muestra en el sitio web, facilitando las tareas de actualización y mantenimiento de contenido.

Capítulo 9

Despliegue

El despliegue de la aplicación web de LIESE se ha configurado para ejecutarse como un servicio en un sistema Linux, utilizando `systemd`. Esta sección detalla la configuración actual del despliegue, así como algunas consideraciones importantes de seguridad y buenas prácticas.

9.1. Servicio de Systemd

Se ha creado un archivo de servicio de `systemd` llamado `liese-website.service` para gestionar la ejecución de la aplicación. Este archivo asegura que el servidor web se inicie automáticamente al arrancar el sistema y se reinicie en caso de fallo.

El contenido del archivo `liese-website.service` es el siguiente:

```
[Unit]
Description=Liese Django Runserver
After=network.target

[Service]
User=saul
Group=saul
WorkingDirectory=/home/saul/liese-website
ExecStart=/home/saul/liese-website/venv/bin/python \
/home/saul/liese-website/src/manage.py runserver 0.0.0.0:8000
Restart=always

[Install]
WantedBy=multi-user.target
```

Este servicio ejecuta el servidor de desarrollo de Django, `manage.py runserver`, que escucha en el puerto 8000. Es importante destacar que **este servidor no es adecuado para un entorno de producción**, ya que no es seguro ni tiene el mismo rendimiento que un servidor de aplicaciones WSGI como Gunicorn o uWSGI.

9.2. Dependencias

Las dependencias del proyecto están listadas en el archivo `requirements.txt`. Para desplegar la aplicación, es necesario instalar estas dependencias en un entorno virtual de Python. Las dependencias principales son:

- **Django:** El framework sobre el que está construida la aplicación.

- **Pillow:** Librería para el manejo de imágenes, utilizada para los campos de imagen en los modelos.

9.3. Configuración de Producción

El archivo de configuración `src/liese/settings.py` contiene varias opciones que deben ser ajustadas para un entorno de producción con el fin de garantizar la seguridad y el rendimiento del sitio.

9.3.1. DEBUG

La variable `DEBUG` está actualmente configurada como `True`. En producción, esta variable **debe estar en** `False`. Mantenerla en `True` expone información sensible de la aplicación en caso de error.

9.3.2. SECRET_KEY

La `SECRET_KEY` está hardcodeada en el archivo de configuración. Para producción, esta clave debe ser única y secreta, y se recomienda cargarla desde una variable de entorno o un sistema de gestión de secretos, en lugar de mantenerla en el código fuente.

9.3.3. Base de Datos

La aplicación utiliza SQLite como motor de base de datos. Si bien SQLite es adecuado para desarrollo y aplicaciones de bajo tráfico, para un entorno de producción con mayor concurrencia se recomienda utilizar un sistema de gestión de bases de datos más robusto, como PostgreSQL o MySQL.

9.3.4. Servidor Web

Como se mencionó anteriormente, el servidor de desarrollo de Django no es para producción. La configuración recomendada para producción incluye el uso de un servidor de aplicaciones WSGI como **Gunicorn** y un servidor web inverso como **Nginx**, que se encargue de servir los archivos estáticos y redirigir las peticiones a Gunicorn.

Capítulo 10

Configuración de Ubuntu Server

Para desplegar el sitio web de LIESE, se recomienda utilizar una distribución de Linux estable y ampliamente soportada como Ubuntu Server. Esta sección describe los pasos generales para configurar un servidor Ubuntu desde cero para alojar la aplicación Django.

10.1. Actualización del Sistema

El primer paso después de instalar Ubuntu Server es asegurarse de que todos los paquetes del sistema estén actualizados. Esto se hace con los siguientes comandos:

```
sudo apt update
sudo apt upgrade
```

10.2. Instalación de Python y Dependencias

La aplicación está desarrollada en Python, por lo que es necesario instalar Python y las herramientas necesarias para gestionar los paquetes y entornos virtuales.

```
sudo apt install python3-pip python3-dev python3-venv
```

10.3. Clonación del Repositorio

Suponiendo que el código fuente del proyecto está alojado en un repositorio de Git, el siguiente paso es clonar el repositorio en el servidor.

```
git clone <URL_DEL_REPOSITORIO> liese-website
cd liese-website
```

10.4. Creación del Entorno Virtual

Es una buena práctica aislar las dependencias de cada proyecto en su propio entorno virtual. Para crear y activar un entorno virtual para el proyecto, se utilizan los siguientes comandos dentro del directorio del proyecto:

```
python3 -m venv venv
source venv/bin/activate
```

Una vez activado el entorno, el prompt de la terminal cambiará para indicar que se está trabajando dentro del entorno virtual.

10.5. Instalación de Dependencias del Proyecto

Con el entorno virtual activado, se pueden instalar las dependencias del proyecto, que se encuentran listadas en el archivo `requirements.txt`.

```
pip install -r requirements.txt
```

10.6. Configuración de la Base de Datos

Antes de poder ejecutar la aplicación, es necesario aplicar las migraciones de la base de datos. Esto creará las tablas necesarias en la base de datos para los modelos de Django.

```
python src/manage.py migrate
```

10.7. Creación de un Superusuario

Para acceder al panel de administración de Django, es necesario crear un superusuario.

```
python src/manage.py createsuperuser
```

Se solicitará un nombre de usuario, una dirección de correo electrónico y una contraseña.

10.8. Configuración del Servicio

Finalmente, como se describió en el capítulo anterior (Capítulo 9), se debe configurar el servicio de `systemd` para que la aplicación se ejecute de forma automática y persistente. Esto implica crear el archivo `.service`, habilitarlo e iniciarlo.

Con estos pasos, el servidor Ubuntu estaría configurado para servir la aplicación web de LIESE. Para un entorno de producción, se requerirían pasos adicionales, como la configuración de un servidor web Nginx y un servidor de aplicaciones Gunicorn.

Capítulo 11

Conexión Remota por SSH

La conexión remota a un servidor es una tarea fundamental para la administración y el despliegue de aplicaciones. El protocolo estándar para realizar esta tarea de forma segura es SSH (Secure Shell).

11.1. ¿Qué es SSH?

SSH es un protocolo de red criptográfico que permite la operación de servicios de red de forma segura sobre una red insegura. La aplicación más conocida es el acceso remoto a sistemas operativos tipo Unix, aunque también se puede utilizar para tunelizar otros protocolos, transferir archivos (usando SFTP o SCP) y gestionar repositorios de Git de forma remota.

La principal ventaja de SSH es que toda la comunicación entre el cliente y el servidor está cifrada, lo que protege la integridad y confidencialidad de los datos, incluyendo las credenciales de acceso.

11.2. Conexión a un Servidor

Para conectarse a un servidor remoto a través de SSH desde una terminal de Linux or macOS, se utiliza el comando `ssh`. La sintaxis básica es la siguiente:

```
ssh usuario@direccion_del_servidor
```

Donde:

- **usuario:** Es el nombre del usuario en el servidor remoto al que se desea acceder.
- **direccion_del_servidor:** Puede ser una dirección IP (ej. `192.168.1.100`) o un nombre de dominio (ej. `servidor.ejemplo.com`).

La primera vez que se conecte a un servidor, se mostrará una advertencia sobre la autenticidad del host y se pedirá confirmación para agregar la huella digital de la clave del servidor a la lista de hosts conocidos. Después de confirmar, se solicitará la contraseña del usuario.

11.3. Autenticación con Claves SSH

Aunque la autenticación con contraseña es común, se considera más seguro y conveniente utilizar un par de claves SSH (una clave pública y una privada). La clave pública se instala en el servidor, mientras que la clave privada se mantiene segura en el cliente. De esta forma, el cliente puede autenticarse sin necesidad de introducir una contraseña cada vez.

Los pasos generales para configurar la autenticación con claves son:

1. Generar un par de claves SSH en la máquina cliente con el comando `ssh-keygen`.
2. Copiar la clave pública (generalmente el archivo `~/.ssh/id_rsa.pub`) al servidor, en el archivo `~/.ssh/authorized_keys` del usuario con el que se desea conectar.

11.4. Transferencia de Archivos

SSH también proporciona una forma segura de transferir archivos entre el cliente y el servidor a través del comando `scp` (Secure Copy). Por ejemplo, para copiar un archivo local a un servidor remoto:

```
scp /ruta/al/archivo/local.txt usuario@servidor:/ruta/remota/
```

Y para copiar un archivo desde el servidor al cliente:

```
scp usuario@servidor:/ruta/remota/archivo.txt /ruta/local/
```

Capítulo 12

Servicios de Linux

En un sistema operativo Linux, un servicio (o demonio) es un programa que se ejecuta en segundo plano, fuera del control interactivo de los usuarios. Para el despliegue del sitio web de LIESE, se utiliza el sistema de gestión de servicios `systemd` para asegurar que la aplicación se ejecute de forma continua.

12.1. Systemd y la Gestión de Servicios

`systemd` es el sistema de inicio y gestor de servicios estándar en la mayoría de las distribuciones modernas de Linux, incluyendo Ubuntu. Se encarga de arrancar el sistema y de gestionar los servicios que se ejecutan en él.

Los servicios se definen en archivos de unidad (con extensión `.service`), que describen cómo se debe iniciar, detener y gestionar un programa.

12.2. El Servicio de la Aplicación LIESE

Como se introdujo en el Capítulo 9, la aplicación se gestiona a través del servicio `liese-website.service`. Analicemos en detalle su configuración:

```
[Unit]
Description=Liese Django Runserver
After=network.target

[Service]
User=saul
Group=saul
WorkingDirectory=/home/saul/liese-website
ExecStart=/home/saul/liese-website/venv/bin/python ...
Restart=always

[Install]
WantedBy=multi-user.target
```

- **[Unit]:** Esta sección contiene metadatos sobre el servicio.
  - `Description`: Una breve descripción del servicio.
  - `After=network.target`: Indica que este servicio debe iniciarse después de que la red esté disponible.
- **[Service]:** Esta sección define cómo se ejecuta el servicio.

  - `User=saul` y `Group=saul`: Especifica que el servicio se ejecutará con los permisos del usuario y grupo `saul`.
  - `WorkingDirectory`: Establece el directorio de trabajo para el proceso.
  - `ExecStart`: El comando que se ejecutará para iniciar el servicio. En este caso, inicia el servidor de desarrollo de Django.
  - `Restart=always`: Indica a `systemd` que reinicie el servicio automáticamente si el proceso termina.
- **[Install]:** Esta sección define cómo se debe instalar el servicio.
  - `WantedBy=multi-user.target`: Habilita el servicio para que se inicie en el arranque del sistema para un entorno multiusuario.

12.3. Gestión del Servicio con systemctl

La herramienta de línea de comandos para interactuar con `systemd` es `systemctl`. A continuación se muestran los comandos más comunes para gestionar el servicio de la aplicación:

- **Iniciar el servicio:**

  ```
  sudo systemctl start liese-website.service
  ```
- **Detener el servicio:**

  ```
  sudo systemctl stop liese-website.service
  ```
- **Reiniciar el servicio:**

  ```
  sudo systemctl restart liese-website.service
  ```
- **Ver el estado del servicio:**

  ```
  sudo systemctl status liese-website.service
  ```
- **Habilitar el inicio automático en el arranque:**

  ```
  sudo systemctl enable liese-website.service
  ```
- **Deshabilitar el inicio automático:**

  ```
  sudo systemctl disable liese-website.service
  ```

12.4. Otros Servicios

En la configuración actual, la aplicación Django y la base de datos SQLite no requieren servicios adicionales. La base de datos SQLite es un archivo en el sistema de ficheros y no un servicio de red.

En un entorno de producción, se añadirían otros servicios, como:

- **Servidor de base de datos:** Como PostgreSQL o MySQL, que se ejecutan como sus propios servicios.
- **Servidor web Nginx:** Que actuaría como proxy inverso y serviría los archivos estáticos, también gestionado como un servicio de `systemd`.

Capítulo 13

Mejoras Futuras

El sitio web de LIESE es una plataforma robusta y funcional, pero siempre hay espacio para la mejora y la expansión. A continuación, se presenta una lista de posibles mejoras que podrían implementarse en el futuro para enriquecer la funcionalidad, la seguridad y la experiencia de usuario del sitio.

- **Certificación SSL:** Implementar un certificado SSL/TLS (por ejemplo, a través de Let's Encrypt) para asegurar que toda la comunicación entre el cliente y el servidor esté cifrada (HTTPS).
- **Mejoras de Ciberseguridad:** Realizar una auditoría de seguridad exhaustiva y aplicar parches y configuraciones para fortalecer el sitio contra vulnerabilidades comunes.
- **Botón de Patrocinio:** Integrar una pasarela de pago como Stripe o PayPal para facilitar las donaciones y patrocinios al laboratorio.
- **CI/CD (Integración y Despliegue Continuos):** Configurar un pipeline de CI/CD utilizando herramientas como GitHub Actions o Travis CI para automatizar las pruebas y el despliegue de nuevas versiones del sitio.
- **Soporte Multiidioma:** Añadir soporte para múltiples idiomas (como el inglés) para ampliar el alcance del sitio a una audiencia internacional.
- **Dashboard Administrativo:** Desarrollar un panel de administración personalizado y más visual para la gestión de contenido, más allá del panel por defecto de Django.
- **Sistema de Búsqueda:** Implementar un motor de búsqueda más avanzado (por ejemplo, con Elasticsearch o Algolia) que permita a los usuarios encontrar rápidamente proyectos, artículos o noticias.
- **Integración con Google Calendar:** Sincronizar los eventos del laboratorio con un calendario de Google público para que los usuarios puedan suscribirse y recibir notificaciones.
- **Modo Claro/Oscuro:** Ofrecer a los usuarios la opción de cambiar entre un tema claro y uno oscuro para mejorar la accesibilidad y la comodidad visual.
- **Sistema de Comentarios:** Añadir una sección de comentarios en los artículos y noticias para fomentar la interacción y el debate.
- **Pruebas Unitarias y de Integración:** Desarrollar un conjunto completo de pruebas automatizadas para garantizar la estabilidad y la calidad del código en futuros desarrollos.
- **Dockerización:** Empaquetar la aplicación y sus dependencias en contenedores de Docker para simplificar el desarrollo, el despliegue y la escalabilidad.
- **Análisis y Monitoreo:** Integrar herramientas como Google Analytics para obtener métricas de uso del sitio y Sentry para el monitoreo de errores en tiempo real.

Capítulo 14

Conclusiones

El desarrollo del sitio web para el Laboratorio de Instrumentación Electrónica de Sistemas Espaciales (LIESE) ha culminado en la creación de una plataforma digital integral que no solo sirve como una vitrina para las actividades del laboratorio, sino que también funciona como una herramienta de gestión y comunicación estratégica.

14.1. Logros del Proyecto

El proyecto ha alcanzado exitosamente sus objetivos principales, entregando una solución tecnológica completa y funcional. Entre los logros más destacados se encuentran:

- **Plataforma Web Completa:** Se ha construido una aplicación web moderna y responsiva utilizando Django, proporcionando una base sólida, segura y escalable para la presencia digital del LIESE.
- **Centralización de la Información:** El sitio unifica la gestión de miembros, proyectos, artículos, eventos y noticias, proporcionando una fuente única y autorizada de información.
- **Portal de Oportunidades Funcional:** Se implementó un sistema robusto para que los estudiantes puedan solicitar tesis, servicio social o investigación, incluyendo un mecanismo de verificación por correo electrónico que previene el spam y valida la autenticidad de los solicitantes.
- **Autonomía en la Gestión de Contenido:** Gracias al panel de administración de Django, el personal del laboratorio puede actualizar el contenido del sitio de manera autónoma, sin necesidad de conocimientos técnicos de programación.

14.2. Lecciones Aprendidas

A lo largo del ciclo de vida del proyecto, se obtuvieron varias lecciones importantes que serán valiosas para futuros desarrollos:

- **Importancia de un Despliegue para Producción:** Se constató que el servidor de desarrollo de Django (`runserver`) es inadecuado para un entorno de producción. La transición a una configuración más robusta con Gunicorn y Nginx es un paso crucial para garantizar la estabilidad y seguridad.
- **Necesidad de Pruebas Automatizadas:** La ausencia de un conjunto de pruebas unitarias y de integración (el archivo `tests.py` está vacío) representa un riesgo para la mantenibilidad a largo plazo. La implementación de pruebas es fundamental para detectar regresiones y asegurar la calidad del código.
- **Gestión de Secretos:** La práctica de mantener claves secretas y credenciales directamente en el código (como en `settings.py`) es insegura. La lección es la necesidad de adoptar sistemas de gestión de secretos o variables de entorno, como se sugiere en la documentación del propio proyecto (README.md).

14.3. Impacto y Aplicaciones

El nuevo sitio web tendrá un impacto significativo y diversas aplicaciones prácticas para el LIESE:

- **Mejora de la Visibilidad y Vinculación:** La plataforma aumentará drásticamente la presencia digital del laboratorio, sirviendo como el principal canal de comunicación y atrayendo talento (estudiantes, investigadores) y potenciales socios industriales o académicos.
- **Diseminación del Conocimiento:** Facilitará la difusión de los resultados de investigación, publicaciones y proyectos del laboratorio a una audiencia global, fortaleciendo su reputación académica.
- **Herramienta de Reclutamiento:** El portal de oportunidades agilizará el proceso de captación de nuevos miembros para el laboratorio, centralizando las solicitudes y facilitando su gestión.
- **Fomento de la Comunidad:** Al centralizar noticias y eventos, el sitio ayuda a construir un sentido de comunidad tanto dentro del laboratorio como con el público externo interesado en las actividades de LIESE.

Bibliografía

[1] Django Software Foundation. (2024). Django documentation. https://docs.djangoproject.com/en/5.1/

[2] Bootstrap. (2024). Bootstrap 5.3 Documentation. https://getbootstrap.com/docs/5.3/getting-started/introduction/

[3] The PostgreSQL Global Development Group. (2024). PostgreSQL Documentation. https://www.postgresql.org/docs/

[4] Canonical Ltd. (2024). Ubuntu Documentation. https://ubuntu.com/server/docs

[5] Django Software Foundation. (2024). Django Email. https://docs.djangoproject.com/en/5.1/topics/email/

[6] Nginx, Inc. (2024). Nginx Documentation. https://nginx.org/en/docs/