Manual Tecnico

Proyecto: Sistema para la acreditación de Ingeniería en Software

Jesús Manuel Juárez Pasillas
Víctor Ubaldo Silva Luna

Diciembre, 2024

Introducción

Propósito

El propósito de este manual técnico es proporcionar una guía detallada para la configuración y operación del sistema de infraestructura de SEDESOL. Este documento está diseñado para ser utilizado por desarrolladores, administradores de sistemas y otros usuarios técnicos, con el fin de garantizar:

- **Comprensión Técnica**: Facilitar el entendimiento de la arquitectura, tecnologías y funcionalidades del sistema.
- **Configuración**: Ofrecer pasos precisos para la preparación del entorno y la implementación del sistema en diferentes escenarios.

Visión general del sistema

El sistema tiene como objetivo facilitar la organización, recolección y almacenamiento de evidencias fotográficas de las obras realizadas bajo los programas de SEDESOL, asegurando un seguimiento adecuado y verificable de los avances, calidad y cumplimiento de las obras por parte de los contratistas.

Alcance

El sistema permitirá organizar y controlar las evidencias fotográficas de obras públicas de infraestructura. Los contratistas podrán subir imágenes asociadas a etapas específicas del proyecto. Permitiendo organizar las evidencias de cada proyecto.

Descripción general del sistema

Arquitectura

La arquitectura de la aplicación sigue el patrón de **MTV (Model-Template-View)**:

- **Modelos (Model)**: Define la estructura de los datos y su lógica de negocio.
- **Plantillas (Template)**: Controla la presentación de los datos en la interfaz de usuario.
- **Vistas (View)**: Manejan la lógica de interacción entre modelos, plantillas y peticiones HTTP.

Modelo

El modelo se encarga de la representación y gestión de los datos del sistema. Define las estructuras de datos, sus relaciones y cómo interactúan con la base de datos.

Responsabilidades:

- **Definir las tablas de la base de datos**: Obras, contratistas, evidencias, estados de revisión, etc.
- **Validar datos** antes de almacenarlos.
- **Gestionar relaciones** entre entidades (e.g., una obra tiene varias evidencias, un contratista está asociado a varias obras).
- **Interactuar con la base de datos** para consultas, inserciones, actualizaciones y eliminaciones.

Vista

La vista actúa como un puente entre el modelo y la plantilla. Contiene la lógica de negocio y determina qué datos deben enviarse a la plantilla.

Responsabilidades:

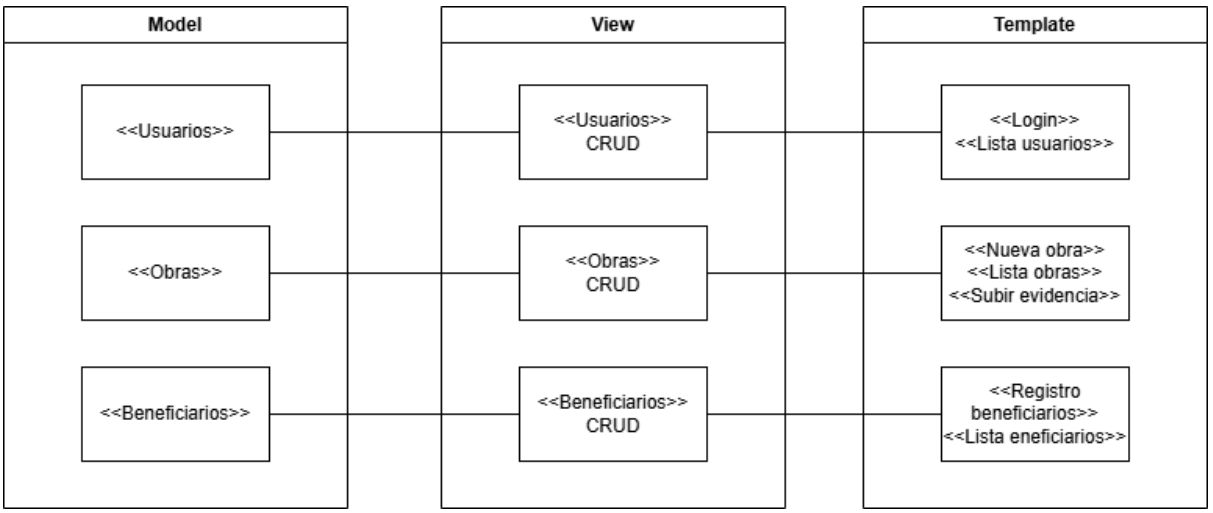
- **Gestionar las solicitudes HTTP** del usuario.
- **Consultar los datos del modelo** según lo necesario para las operaciones solicitadas.
- **Validar acciones del usuario**, como subir una evidencia o actualizar el estado.
- **Enviar los datos adecuados a la plantilla** para que sean renderizados.

Template

Propósito:
La plantilla se encarga de la presentación de los datos al usuario. Define cómo se muestran los datos obtenidos por la vista.

Responsabilidades:

- **Renderizar contenido dinámico** en HTML, como listas de obras o evidencias específicas.
- **Proveer interfaces amigables** para interactuar con el sistema, como formularios para subir imágenes o botones para revisar evidencias.
- **Separar el diseño visual de la lógica de negocio** para mantener un código limpio y organizado.



Requisitos del sistema

Los siguientes requisitos que se describen, es bajo carga mínima esperada, por lo tanto sí se espera más carga se recomienda aumentar estas especificaciones.

- **CPU**: 2 núcleos.
- **RAM**: 4 GB.
- **Almacenamiento**: 30 GB o más para bases de datos y archivos estáticos.

Tecnologías utilizadas

Para este proyecto se utilizaron las siguientes tecnologías y herramientas.

- **Python:** Lenguaje de programación principal del proyecto, conocido por su legibilidad y eficiencia. Python se utiliza para escribir la lógica del backend, incluyendo el manejo de la base de datos, la lógica de negocio, y la integración con otras aplicaciones y servicios.
- **Django:** Framework web de alto nivel para Python que fomenta el desarrollo rápido y diseño limpio y pragmático. Django se utiliza para estructurar el backend del sistema, proporcionando herramientas robustas para el desarrollo de URLs, vistas y configuraciones de seguridad integradas.
- **MySQL:** Sistema de gestión de base de datos relacional utilizada para almacenar y gestionar la data crítica de la aplicación, como información de usuarios, datos de transacción y registros históricos.
- **Docker:** Utilizado para contenerizar y gestionar los entornos de la aplicación, asegurando la consistencia entre los entornos de desarrollo pruebas y producción. Docker simplifica la configuración de despliegue, permitiendo que la aplicación se ejecute de manera aislada y escalable.
- **CSS:** Lenguaje de hojas de estilo utilizado para definir y diseñar la presentación del contenido HTML. CSS se emplea para estilizar las interfaces

de usuario, incluyendo colores, layout y tipografías, asegurando que la aplicación sea visualmente atractiva y fácil de usar.

- **HTML:** Lenguaje de marcado utilizado para construir la estructura de las páginas web en el sistema. HTML forma la base sobre la que se aplican estilos CSS y se ejecutan scripts, estructurando los contenidos y elementos en la interfaz de usuario.
- **Bootstrap:** Framework de HTML, CSS y JavaScript para desarrollar componentes de interfaz y grids responsive. Bootstrap se utiliza para asegurar que la aplicación sea accesible y eficaz en dispositivos móviles y de escritorio, proporcionando un sistema grid flexible y componentes prediseñados.

Configuración

Preparación del entorno

1. **Requisitos previos**
   - Se requiere tener instalado la herramienta **Docker.**
   - Se requiere la herramienta de control de versiones **git.**
2. **Clonar el repositorio**
   - git clone https://github.com/JesusJuarez1/infraestructura_SEDESOL
   - cd ifraestrucutura_SEDESOL/
3. **Configuración de variables de entorno**
   Crear un archivo llamado .env en la raíz del proyecto con el siguiente contenido:
   - DB_ENGINE=django.db.backends.mysql
   - DB_NAME=sedesol_database
   - DB_USER=sedesol_user
   - DB_PASSWORD=msedesol
   - DB_HOST=infraestructura_sedesol-db-1
   - DB_PORT=3306
   - DB_ROOT_PASSWORD=sedesolroot
4. **Construcción y Ejecución de Contenedores**
   - Construir los contenedores:
     i. docker-compose build
   - Levantar los servicios:
     i. docker-compose up -d
5. **Iniciar aplicación**
   - Entrar al contenedor
     i. docker exec -it infraestructura_sedesol-app-1 /bin/bash
   - Hacer las migraciones
     i. cd infraestructura_SEDESOL/
     ii. python3 manage.py makemigrations

iii. python3 manage.py migrate
- Crear un superusuario
  i. python3 manage.py createsuperuser
- Iniciar la aplicación
  i. python3 manage.py runserver 0.0.0.0:8000

6. **Verificación del Despliegue Local**
   - Accede al sistema desde el navegador en:
     i. http://localhost:8000/
7. **Apagar los Contenedores**
   - Si necesitas detener los servicios:
     i. docker-compose down

Estructura del proyecto

Nivel superior del proyecto

El directorio principal contiene archivos y carpetas clave para el funcionamiento global del proyecto, donde la estructura se ve así:

```
app
│   ├────manage.py
│   ├────requeriments.txt
│   ├────db.sqlite3
│   ├────infraestructura_SEDESOL
│   │           ├── asgi.py
│   │           ├── settings.py
│   │           ├──  urls.py
│   │           └── wsgi.py
│   ├────usuarios
│   ├────media
│   ├────home
│   ├────obras
│   ├────static
│   ├────authentication
├─.env
├─.gitignore
├─docker-compose.yml
├─Dockerfile
├─README.md
```

- **manage.py**: Herramienta para interactuar con el proyecto, como ejecutar el servidor o comandos administrativos.
- **core/**: Contiene la configuración principal del proyecto:
- **settings.py**: Configuración global (base de datos, aplicaciones instaladas, variables de entorno, etc.).

- **urls.py**: Puntos de entrada de las URL principales.
- **wsgi.py y asgi.py**: Configuración para servidores web y aplicaciones asíncronas.
- **app/**: Carpeta donde se agrupan las aplicaciones del proyecto, cada una con su propia lógica y organización interna.
- **static/**: Archivos estáticos compartidos (CSS, JavaScript, imágenes).
- **templates/**: Plantillas HTML compartidas.
- **media/**: Archivos cargados por los usuarios.
- **requirements.txt**: Dependencias necesarias del proyecto.
- **Dockerfile y docker-compose.yml**: Archivos para configurar y ejecutar contenedores Docker.
- **.env**: Variables de entorno sensibles (nunca debe subirse al repositorio).

Estructura de cada aplicación (módulo)

Cada aplicación tiene su propia carpeta con una disposición estándar que incluye los mismos archivos clave:

usuarios
│ admin.py
│ apps.py
│ forms.py
│ functions.py
│ models.py
│ tests.py
│ urls.py
│ views.py
├──migrations
├──templates

- **admin.py**: Configuración para la interfaz de administración de Django.
- **apps.py**: Configuración de la aplicación (nombre, etiquetas).
- **migrations/**: Migraciones de base de datos generadas por Django.
- **templates/**: Plantillas HTML del módulo.
- **models.py**: Modelos de datos (clases ORM que representan las tablas de la base de datos).
- **tests.py**: Pruebas unitarias o de integración específicas de la aplicación.
- **urls.py**: Rutas específicas de la aplicación.
- **views.py**: Lógica de las vistas (controladores de solicitudes y respuestas)

Cada módulo tiene por defecto esta estructura de carpetas y archivos, por lo que si se quiere agregar un nuevo módulo se crea con el comando **python manage.py startapp nombre_aplicación,** con lo que se creará la estructura.

Operación del sistema

Inicio y parada del sistema

Al levantar el contenedor de docker (explicado en el punto **Configuración - preparación del entorno**), el sistema correrá automáticamente. Igualmente para detenerlo se debe apagar el contenedor con el comando **docker compose down**.

Monitoreo del sistema

Para monitorear el sistema se puede hacer de la siguiente forma:

1. Conectarse a los logs del contenedor que corre la aplicación y verlos en tiempo real
    a. **docker logs -f infraestructura_sedesol-app-1**