**Documentación del Proyecto**

**Índice**

1. Introducción
2. Requisitos del Sistema
3. Instalación
4. Configuración
5. Estructura del Proyecto
6. Descripción de Funcionalidades
7. Validaciones y Seguridad
8. Consideraciones de Despliegue
9. Tecnologías Utilizadas
10. Conclusiones

**Introducción**

Este proyecto consiste en una aplicación web desarrollada utilizando Django, Bootstrap y JavaScript. La aplicación permite la gestión de artículos con funcionalidades CRUD (Crear, Leer, Actualizar, Eliminar) y utiliza cifrado para la protección de datos sensibles.

**Requisitos del Sistema**

* Python 3.12.4
* Django 5.0.7
* Bootstrap 5.3.3
* cryptography 42.0.8

**Instalación**

Para instalar y configurar el proyecto, sigue los siguientes pasos:

1. **Clonar el repositorio:**

Copiar código

git clone <https://github.com/LuisOPavez/Examen.git>

cd proyecto

1. **Crear y activar un entorno virtual:**

Copiar código

python -m venv examen

source examen/bin/activate

# En Windows use `examen\Scripts\activate`

1. **Instalar las dependencias:**

Copiar código

pip install -r requirements.txt

1. **Realizar las migraciones:**

Copiar código

python manage.py migrate

1. **Cargar datos iniciales (si aplica):**

Copiar código

python manage.py loaddata initial\_data.json

1. **Iniciar el servidor de desarrollo:**

Copiar código

python manage.py runserver

**Configuración**

En el archivo settings.py, asegúrate de configurar las siguientes claves para el cifrado:

python

Copiar código

# settings.py

SECRET\_KEY = 'django-insecure-zbaf5!44-j5vii2zki+1#^9p^l0f#\*b5gd5r!b6zap\*a\_2p0u4'

FERNET\_KEY = os.getenv('FERNET\_KEY','TUFCR5btjwvytIS-UuyMsj4jr1Vl4XQNcC5ROCDrvhY=')

**Estructura del Proyecto**

proyecto/

│

├── items/

│ ├── templates/

│ │ └── items/

│ │ ├── item\_form.html

│ │ └── item\_list.html

│ ├── models.py

│ ├── views.py

│ └── urls.py

│

├── main/

│ ├── templates/

│ │ └── main/

│ │ └── index.html

│ ├── views.py

│ └── urls.py

│

├── static/

│ ├── css/

│ │ └── style.css

│ └── js/

│ └── index.js

│

├── proyectoweb/

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

│

└── manage.py

**Descripción de Funcionalidades**

**1. Gestión de Artículos**

Permite crear, leer, actualizar y eliminar artículos. Los formularios están validados tanto en el frontend como en el backend.

**2. Cifrado de Datos Sensibles**

Los datos sensibles se cifran utilizando la librería cryptography antes de ser almacenados en la base de datos.

python

Copiar código

# models.py (extracto)

from cryptography.fernet import Fernet

from django.conf import settings

from django.db import models

cipher\_suite = Fernet(settings.FERNET\_KEY)

class Item(models.Model):

# otros campos...

sensitive\_data = models.CharField(max\_length=500, default='')

def save(self, \*args, \*\*kwargs):

self.sensitive\_data = cipher\_suite.encrypt(self.sensitive\_data.encode()).decode()

super(Item, self).save(\*args, \*\*kwargs)

def get\_sensitive\_data(self):

return cipher\_suite.decrypt(self.sensitive\_data.encode()).decode()

**3. Validaciones y Funcionalidades en el Front-End**

Se utilizan JavaScript y Bootstrap para validar los formularios y mejorar la experiencia del usuario.

html

Copiar código

<!-- form\_validation.js (extracto) -->

document.querySelector('form').addEventListener('submit', function(event) {

var nombre = document.querySelector('input[name="nombre"]').value;

if (!nombre) {

alert('El nombre es obligatorio');

event.preventDefault();

}

});

**Validaciones y Seguridad**

* **Validaciones en el Front-End:** Utilizamos JavaScript para validar los formularios antes de su envío.
* **Cifrado de Datos Sensibles:** Se cifran los datos sensibles utilizando la librería cryptography antes de almacenarlos en la base de datos.
* **Autenticación y Autorización:** Utilizamos las funcionalidades de autenticación y autorización de Django para gestionar el acceso a diferentes partes de la aplicación.

**Consideraciones de Despliegue**

1. Configurar las variables de entorno para SECRET\_KEY y FERNET\_KEY.
2. Utilizar un servidor web como Nginx o Apache para servir la aplicación en producción.
3. Configurar una base de datos en producción, como PostgreSQL o MySQL.

**Tecnologías Utilizadas**

* **Django:** Framework de backend.
* **Bootstrap:** Framework de frontend para el diseño responsivo.
* **JavaScript:** Para validaciones y funcionalidades en el frontend.
* **cryptography:** Librería de cifrado para proteger datos sensibles.

**Conclusiones**

Este proyecto demuestra la capacidad de crear una aplicación web completa utilizando Django, con validaciones en el frontend, cifrado de datos sensibles, y un diseño responsivo utilizando Bootstrap. Además, se gestionan las versiones del código utilizando Git y GitHub.