**INFORME DE LABORATORIO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INFORMACIÓN BÁSICA** | | | | | |
| **ASIGNATURA:** | **Programacion Web 2** | | | | |
| **TÍTULO DE LA PRÁCTICA:** | ***Git de Django 4 y 5*** | | | | |
| **NÚMERO DE PRÁCTICA:** | ***8*** | **AÑO LECTIVO:** | ***2025*** | **NRO. SEMESTRE:** | ***1*** |
| **FECHA DE PRESENTACIÓN** | ***28/06/2025*** | **Repositorio** | *https://github.com/JesusFSP/Curso-PWeb2.git* | | |
| **INTEGRANTE (s):**  **Silva Pino Jesus Francisco** | | | | **NOTA:** |  |
| **DOCENTE(s):**  **CARLO JOSE LUIS CORRALES DELGADO** | | | | | |

|  |
| --- |
| **SOLUCIÓN Y RESULTADOS** |
| 1. **SOLUCIÓN DE EJERCICIOS/PROBLEMAS**  1. Introducción En este informe se describen los pasos realizados para implementar una aplicación web básica en Django, utilizando vistas basadas en clases (CBV) para gestionar un CRUD (Crear, Leer, Actualizar, Borrar) de personas. Este proyecto forma parte de la práctica del curso Programación Web 2, aplicando buenas prácticas de estructura de proyecto, plantillas y manejo de archivos estáticos. 2. Objetivos - Construir un proyecto Django organizado. - Implementar modelos, vistas y rutas para operar un CRUD. - Usar vistas genéricas basadas en clases. - Crear plantillas que extienden una base común. - Configurar archivos estáticos para aplicar estilos CSS. - Validar la funcionalidad completa de la aplicación. 3. Herramientas - Lenguaje: Python 3.11 - Framework: Django 5.2.3 - Entorno: Entorno virtual (venv) - Editor: VS Code (o similar) - Sistema operativo: Linux (Ubuntu) 4. Desarrollo4.1 Creación del entorno y del proyecto Instalación de Django:  python -m venv venv  source venv/bin/activate  pip install django  Creación del proyecto y la aplicación:  django-admin startproject listaContactos .  python manage.py startapp personas 4.2 Definición del modelo Se definió el modelo Persona en personas/models.py: class Persona(models.Model):  nombres = models.CharField(max\_length=100)  apellidos = models.CharField(max\_length=100)  edad = models.IntegerField()  donador = models.BooleanField(default=False) 4.3 Configuración de la base de datos Se aplicaron migraciones:  python manage.py makemigrations  python manage.py migrate 4.4 Creación de las vistas basadas en clases En personas/views.py se implementaron: - ListView: para listar personas con filtro por edad. - DetailView: para ver detalles (opcional). - CreateView: para crear nuevas personas. - UpdateView: para modificar datos. - DeleteView: para confirmar y borrar. 4.5 Configuración de URLs personas/urls.py:  path('', PersonaListView.as\_view(), name='persona-list')  path('create/', PersonaCreateView.as\_view(), name='persona-create')  path('<int:pk>/update/', PersonaUpdateView.as\_view(), name='persona-update')  path('<int:pk>/delete/', PersonaDeleteView.as\_view(), name='persona-delete')  listaContactos/urls.py:  path('personas/', include('personas.urls')) 4.6 Creación de plantillas - base.html: plantilla principal con título y hoja de estilos. - persona\_list.html: muestra título, lista de personas y botones de crear/modificar/borrar. - persona\_form.html: formulario para crear/modificar. - persona\_confirm\_delete.html: formulario para confirmar borrado.   4.7 Configuración de archivos estáticos Se creó la carpeta personas/static/personas/ con style.css. Se enlazó en base.html y se configuró en settings.py con STATIC\_URL. 4.8 Pruebas de funcionamiento **Se verificó /personas/ para mostrar solo: Mi sitio Lista de personas Listado de registros Botones de Crear, Modificar y Borrar.**    Modificar    Eliminar        Commits totales |
|  |
| 1. **CONCLUSIONES**   El proyecto implementa de forma correcta la estructura de un CRUD en Django usando vistas genéricas, plantillas base reutilizables y archivos estáticos para dar estilo. Se asegura una arquitectura limpia y escalable para mantener y extender el sistema. |

|  |
| --- |
| **RETROALIMENTACIÓN GENERAL** |
|  |

|  |
| --- |
| **REFERENCIAS Y BIBLIOGRAFÍA** |
|  |