Informe de Django 04

Tema: Django 4

Nota

Estudiante	Escuela	Asignatura
Victor Gonzalo Maldonado Vilca vmaldonadov@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 1702122

Tarea	Tema	Duración
08	Django 4	2 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 23 de mayo de 2024	Al 24 de junio de 2024

1. Tarea

- Repetir las actividades vistas en teoria de Django4. Ir haciendo commits cada avance. Compartirlo con el docente (CarloCorrales010)

2. Entregables

- Informe hecho en Latex.
- **texto:** Respuestas a las preguntas de las diapositivas
- Captura de pantalla del siguiente comando:

```
git log --graph --pretty=oneline --abbrev-commit --all
```

- URL - Repositorio GitHub

3. Equipos, materiales y temas utilizados

- Configuración de Proyectos
- Modelado de Datos
- Migraciones de Base de Datos

- Desarrollo de Vistas y URLs
- Uso de Plantillas HTML
- Administración de Aplicaciones
- Control de Versiones
- Creacion y manejo de Formularios

4. Respuestas a las Preguntas dejadas en las Diapositivas

1. **¿El formulario tiene los mismos campos y tipos que el modelo?**
   Respuesta: Sí,debido a que en el formulario se integran los mismos campos y tipos de datos que encontramos en el modelo, es cierto que estos formularios ya no heredan de un modelo en particular, pero cumple con los campos ya establecidos en el modelo permitiendo ue el formulario capture y valide datos de acuerdo con la estructura definida en el modelo, aunque no esté vinculado de manera directa.

2. **¿Cuál es el valor para las opciones core by default?**
   Respuesta: Los valores por defecto de las opciones core son:

   - **required = True**: Se refiere a que el campo debe ser llenado obligatoriamente por el usuario; es otras palabras, no puede quedar vacío.
   - **widget=None**: Se refiere a que el campo utilizará el widget estándar adecuado para su tipo de datos, siendo seleccionado automáticamente si no se especifica.
   - **label=None**: Se refiere que el campo no tendrá una etiqueta predeterminada; en caso de ser None, se generará una etiqueta basada en el nombre del campo.
   - **initial=None**: Representa el valor inicial por defecto para el campo; si se omite, el campo estará vacío inicialmente.
   - **help_text=”**: Significa que se puede proporcionar un texto de ayuda opcional para el campo; si se deja vacío, no se mostrará ninguna ayuda.

5. Captura de Pantalla

```
Usuario@DESKTOP-G0S7RR6 MINGW64 /D/UNSA/pw2/teoria/Django2/src (master)
$ git log --graph --pretty=oneline --abbrev-commit --all
* 3659e2f (HEAD -> master, origin/master) Terminando Django 4, se creo nuevo archivo para las rutas de la aplic
acion, en las rutas del proyecto se importo include, asu vez en el modelo se hizo la importacion de reverse par
a poder usarlo
* 3efa7a2 Se definio metodo get_absolute_url() para poder calcular la url canonica de un objeto
* bc755ff Se agrego importacion get_object_or_404, ademas se implemento vista (personasDeleteView), definiendo
su ruta y creando una nueva plantilla para poder borrar o eliminar objetos
* 527a65b Se agrego una vista (personasListView), un template , se definio las rutas correspondientes, para pod
er listar todos los objetos que se encuentren en la base de datos solo se listara el nombre
* 8a52fa6 Se verifico el las url dinamicos, en caso de que se navegue en un objeto que no existe, uso de get_ob
ject_or_404
* d66837d Se implemento una vista (personasShowObject), y una ruta, ademas se modifico la plantilla description
.html -para poder navegar entre los elementos almacenados en nuestro modelo
* a3740d6 Se modifico la vista personaCreateView para  añadir valores iniciales al formulario en este caso en e
l campo nombre
* 834578e Modificando vistas personaCreateView para poder editar un objeto, en esta ocasion es el objeto de id
= 2
* ca68f2f Validacion extra de campos clean_<fieldname>(), la ruta agregar para la visualizacion de la validacio
n correspondiente
* baa7c14 Se agrego widget personalizado Textarea al formulario para el cambo nombre, se configuro placeholder,
 id, class y cols para el widget
* 77771fd En el formulario RawPersonaForm, se añadio opciones en el campo nombre y edad,(label = 'Your Name' e
initial = 20)
* 46a744f Agregando a la vista verificaciones que corresponden a imprimir los errores que se encuentren, ademas
 de imprimir los datos cuando sea valido el formulario - se implemente el guardar los datos enviados del formul
ario
* 0e85f7f Se sigue modificando la vista personasAnotherCreateView, para lograr hacer una comparación entre el m
etodo GET y POST
* 225528d Modificando vista personasAnotherCreateView, para hacer una llamada POST, donde se logra se muestre e
n el template que los campos son obligatorios
* dd0de3d Se definio ruta anoterAdd, además de movio RawPersonaForm de models.py a forms.py, se agrego las impo
rtaciones correspondientes en cada archivo(importacion de las vistas en las rutas)
* 0a68c32 creación de una nueva vista personasAnotherCreateView, donde se inicializa el formulario RawPersonaFo
rm, se pasa el formulario al contexto y renderizamos a la plantilla personasCreate.html
* ee8f5fb Inicio de Django 4 - Implementando nuevo modelo RawPersonaForm, que ya no hereda de ModelForm, es la
forma básica del form
```

Figura 1: commits - local

6. URL de Repositorio Github

- URL del Repositorio GitHub.
- `https://github.com/Victor-Gonzalo-Maldonado-Vilca/Django2.git`

7. Desarrollo del trabajo

Continuamos a partir del proyecto realizado en Django 3

7.1. Imágenes de Ejecución

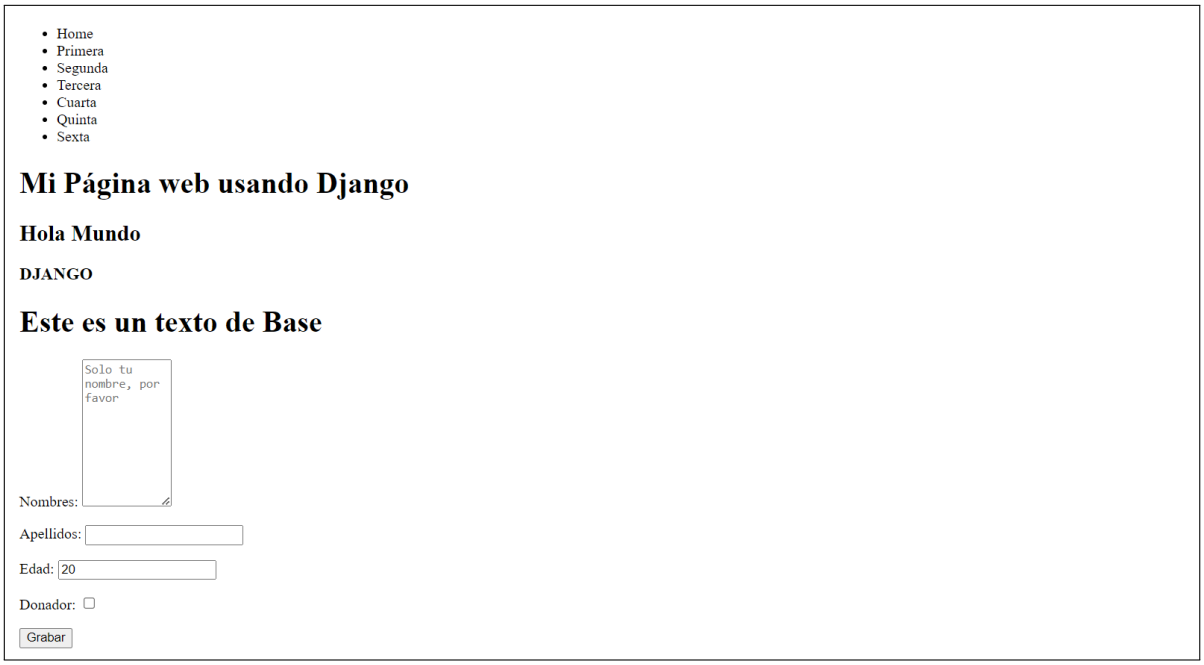

Figura 2: Agregar Objetos

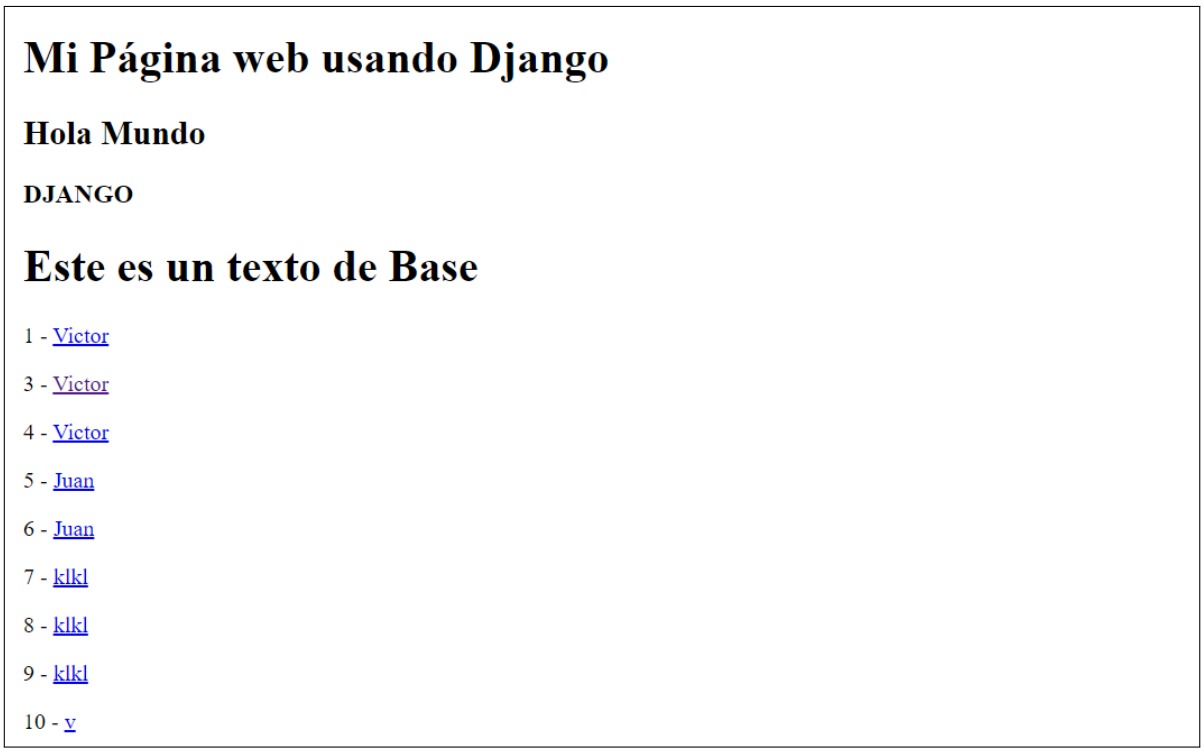

Figura 3: Listar objetos

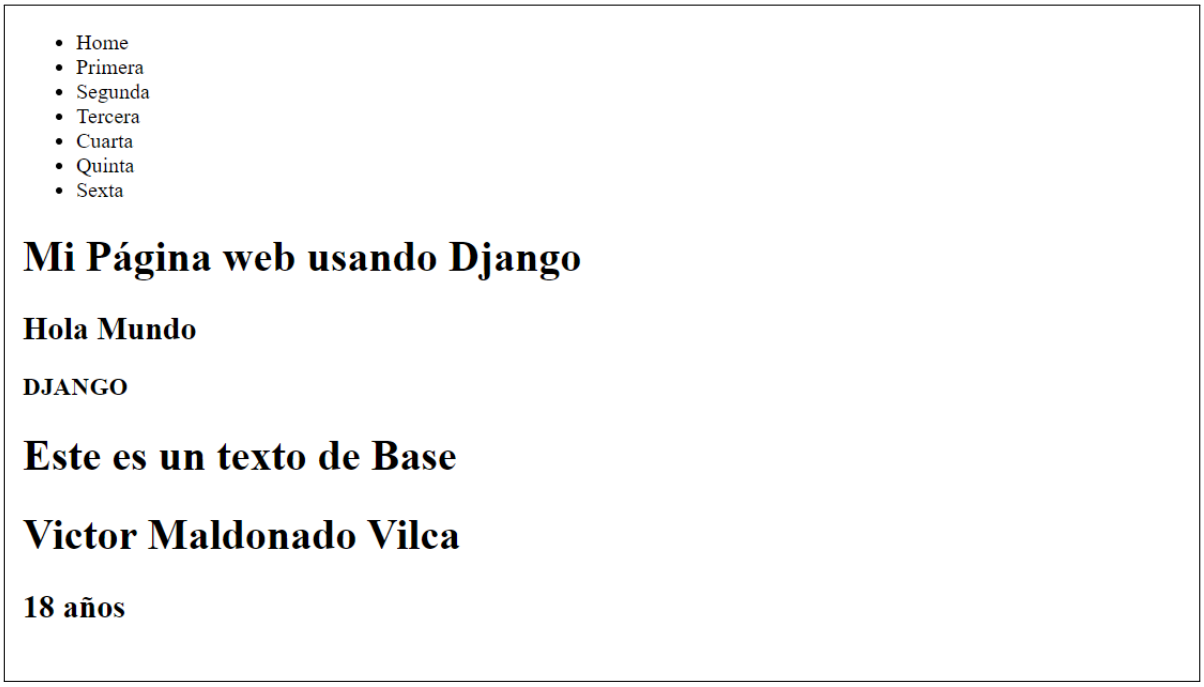

Figura 4: Visualizar Objetos Especificos

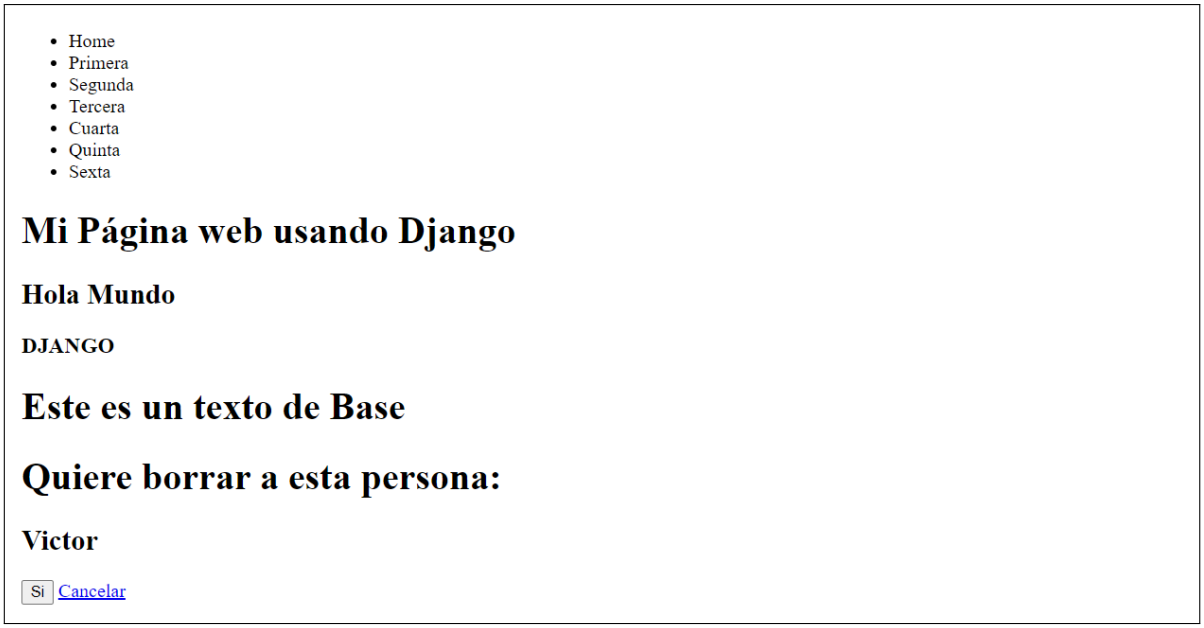

Figura 5: Eliminar objetos

7.2. Modelos – Aplicación personas

Se definio la función get_absolute_url en Django se utiliza para obtener la URL absoluta de una instancia de un modelo específico. En el siguiente código, la función devuelve la URL que apunta a la vista 'personas:browsing' con el parámetro myID igual al id de la instancia actual.

Listing 1: Modelo Persona

```
def get_absolute_url(self):
  return reverse('personas:browsing', kwargs={'myID': self.id})
```

7.3. Vistas – Aplicación personas

7.3.1. Vista personasAnotherCreateView()

- **Descripción:** Se define una vista llamada personasAnotherCreateView, diseñada para gestionar la creación de instancias de objetos Persona utilizando un formulario personalizado denominado RawPersonaForm.
  Primero, se inicializa el formulario RawPersonaForm para recopilar los datos necesarios para crear una nueva persona. Luego, se verifica si la solicitud es de tipo POST. En caso afirmativo, se intenta procesar el formulario con los datos recibidos.
  Si el formulario es válido según la función is_valid(), se imprimen los datos limpios (cleaned_data) del formulario para su revisión y se procede a crear una nueva instancia de Persona utilizando estos datos validados. En caso de que el formulario no sea válido, se imprimen los errores generados por el formulario para su corrección.
  Finalmente, se prepara el contexto con el formulario y cualquier otra información necesaria, y se renderiza la plantilla 'personas/personasCreate.html' para que el usuario pueda interactuar con el formulario de creación de personas.

- **Código:**

```
def personasAnotherCreateView(request):
  form = RawPersonaForm()
  if request.method == "POST":
      form = RawPersonaForm(request.POST)
      if form.is_valid():
          print(form.cleaned_data)
          Persona.objects.create(**form.cleaned_data)
      else:
          print(form.errors)
  context = {
      'form': form,
  }
  return render(request, 'personas/personasCreate.html', context)
```

7.3.2. Vista personasShowObject()

- **Descripción:** La presente vista se encarga de mostrar los detalles de una instancia específica de Persona según el ID proporcionado en la solicitud. Primero, se utiliza get_object_or_404 para obtener la instancia de Persona correspondiente al ID especificado. Si no se encuentra ninguna instancia con ese ID, se muestra una página 404.
  Luego, se prepara el contexto con el objeto obj y se renderiza la plantilla 'personas/description.html', que probablemente contenga la estructura HTML para mostrar la información detallada de la persona en el frontend.

- **Código:**

```
def personasShowObject(request, myID):
  obj = get_object_or_404(Persona, id = myID)
  context = {
```

```
        'objeto': obj,
    }
    return render(request, 'personas/description.html', context)
```

7.3.3. Vista personasListView()

- **Descripción:** Esta vista obtiene todos los objetos Persona de la base de datos utilizando 'Persona.objects.all()'. Luego, prepara el contexto con la lista de objetos Persona y renderiza la plantilla 'personas/personasLista.html', que probablemente contenga la estructura HTML para mostrar la lista de personas en el frontend de la aplicación.
- **Código:**

```
def personasListView(request):
  queryset = Persona.objects.all()
  context = {
      'objectList': queryset,
  }
  return render(request, 'personas/personasLista.html', context)
```

7.3.4. Vista personasDeleteView()

- **Descripción:** Esta vista elimina una instancia específica de Persona según el ID proporcionado en la solicitud POST. Primero, se obtiene la instancia de Persona o se muestra una página 404 si no se encuentra. Luego, si la solicitud es POST, se elimina la instancia y se redirige al usuario a la página principal. En caso contrario, se muestra un formulario de confirmación en la plantilla ''personas/personasBorrar.html''.
- **Código:**

```
def personasDeleteView(request, myID):
  obj = get_object_or_404(Persona, id = myID)
  if request.method == 'POST':
      print("lo borro")
      obj.delete()
      return redirect('../')
  context = {
      'objeto': obj,
  }
  return render(request, 'personas/personasBorrar.html', context)
```

7.4. Formularios - Apicación personas

7.4.1. Formulario PersonaForm()

- **Descripción:** Se define el método clean_nombres que verifica si el valor ingresado en el campo 'nombre' tiene la primera letra en mayúscula. Si es así, el método devuelve el nombre sin realizar ninguna modificación. En caso contrario, se lanza una excepción ValidationError con el mensaje 'La primera letra en Mayuscula'.

- **Código:**

```
def clean_nombres(self, *args, **kwargs):
  print('paso')
  name = self.cleaned_data.get('nombre')
  if name.istitle():
      return name
  else:
      raise forms.ValidationError('La primera letra en Mayuscula')
```

7.4.2. Formulario RawPersonaForm()

- **Descripción:** El formulario RawPersonaForm se utiliza para recopilar información sobre una persona. Incluye campos para el nombre, apellidos, edad y la opción de indicar si la persona es donadora. El campo de nombres utiliza un widget de Textarea con características adicionales como marcador de posición, identificador, clase y número de columnas. El campo de edad tiene un valor inicial predeterminado de 20 años. Este formulario proporciona una interfaz sencilla para que los usuarios ingresen información básica sobre una persona.

- **Código:**

```
class RawPersonaForm(forms.Form):
  nombres  = forms.CharField(
      widget = forms.Textarea(
          attrs={
              'placeholder': 'Solo tu nombre, por favor',
              'id': 'nombreID',
              'class': 'special',
              'cols': 10,
          }
      )
  )
  apellidos = forms.CharField()
  edad     = forms.IntegerField(initial = 20)
  donador  = forms.BooleanField()
```

7.5. URLs

7.5.1. urls del proyecto listaContactos

- **Descripción:** El archivo urls.py importa las funciones necesarias para manejar las rutas URL, como path y include, junto con las vistas myHomeView y anotherView desde el módulo views de la aplicación inicio. Las rutas URL se configuran de la siguiente manera: la ruta 'personas/' incluye las URLs de la aplicación personas, la ruta 'admin/' está asignada a la interfaz de administración de Django, la ruta raíz ('') dirige a la vista myHomeView con el nombre 'Página de Inicio', y la ruta 'another/' se relaciona con la vista anotherView y se le asigna el nombre 'otro'.

- **Código:**

```
from django.contrib import admin
from django.urls import path, include
from inicio.views import myHomeView, anotherView

urlpatterns = [
```

```
    path('personas/', include('personas.urls')),
    path('admin/', admin.site.urls),
    path('', myHomeView, name='Pagina de Inicio'),
    path('another/', anotherView, name='otro'),
]
```

7.5.2. urls de la aplicación personas

- **Descripción:** El archivo urls.py maneja las rutas URL relacionadas con las vistas de personas. Incluye rutas como ''persona/'' para personaTestView, 'agregar/' para personaCreateView, ''personas/¡int:myID¿/'' para personasShowObject, la ruta raíz ('') para personasListView, y ''personas/¡int:myID¿/delete'' para personasDeleteView.
- **Código:**

```
from django.urls import path
from personas.views import personasDeleteView, personasListView, personaTestView,
    personaCreateView, searchForHelp, personasAnotherCreateView,
    personasShowObject

app_name = 'personas'
urlpatterns = [
    path('persona/', personaTestView, name='testViewPersona'),
    path('agregar/', personaCreateView, name='createPersona'),
    path('personas/<int:myID>/', personasShowObject, name = 'browsing'),
    path('', personasListView, name = 'listing'),
    path('personas/<int:myID>/delete', personasDeleteView, name = 'deleting'),
]
```

8. Templates

En esta actividad se realizo 2 templates

8.1. Plantilla personasLista.html

- **Descripción:** Se define una estructura HTML básica con metadatos y un título. Luego, extiende el contenido del archivo base.html utilizando la directiva extends 'base.html' . Dentro del bloque de contenido block content , utiliza un bucle for para iterar sobre cada instancia en la lista de objetos (objectList). Para cada instancia, muestra su ID y nombre como un enlace. Al hacer clic en el nombre, se redirige a la URL absoluta definida en el método get_absolute_url de la instancia.
- **Código:**

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="UTF-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>test</title>
  </head>
  <body>
    {% extends 'base.html' %}
    {% block content %}
```

```
        {% for instance in objectList %}
          <p> {{instance.id}}
          -
            <a href='{{instance.get_absolute_url}}'>
              {{instance.nombre}}
            </a>
          </p>
        {% endfor %}
        {% endblock %}
      </body>
    </html>
```

8.2. Plantilla personasBorrar.html

- **Descripción:** En este template se define un formulario que se envía utilizando el método POST y contiene un token CSRF para protección contra ataques de falsificación de solicitudes entre sitios. En el formulario, se muestra el nombre de la persona que se va a eliminar '(objeto.nombre)' junto con un mensaje preguntando si se desea borrar a esa persona. Se incluyen dos opciones para el usuario: un botón de enviar ('Si') para confirmar la eliminación y un enlace para cancelar la acción y regresar a la página anterior.
- **Código:**

```
    <!DOCTYPE HTML>
    <html>
      <head>
        <meta charset="UTF-8"/>
        <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
        <title>test</title>
      </head>
      <body>
        {% extends 'base.html' %}
        {% block content %}
        <form method='POST'> {% csrf_token %}
          <h1>Quiere borrar a esta persona: </h1>
          <h2>{{objeto.nombre}}</h2>
          <input type='submit' value='Si'/>
          <a href='../'>Cancelar</a>
        </form>
        {% endblock %}
      </body>
    </html>
```

8.3. Ejecución del servidor

Este siguiente comando inicia el servidor local de Django. Una vez ejecutado, el servidor estará disponible por defecto en la dirección `http://127.0.0.1:8000/`. Desde esta dirección, se puede acceder a las diferentes vistas y funcionalidades del proyecto Django.

```
python manage.py runserver
```

9. Conclusiones

- El aprendizaje sobre el desarrollo en Django ha sido significativo, abarcando aspectos clave como la configuración de rutas URL, definición de vistas y creación de formularios.
- La comprensión de la estructura de proyectos en Django, incluyendo la organización de aplicaciones, modelos y archivos de configuración, es esencial para un desarrollo eficiente.
- La utilización de herramientas como el servidor de desarrollo de Django y el sistema de plantillas facilitaron el desarrollo ágil y eficiente de la aplicación.
- Django es un framework web potente y versátil que permite desarrollar aplicaciones web de manera rápida y eficiente.
- La arquitectura MVC (Modelo-Vista-Controlador) de Django ayuda a organizar el código de manera estructurada y modular, lo que facilita la mantenibilidad y escalabilidad de las aplicaciones.
- Con un buen entendimiento de Django y siguiendo las mejores prácticas de desarrollo, se pueden crear aplicaciones web robustas y de alto rendimiento.

9.1. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 2: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		20	

10. Referencias

- https://docs.djangoproject.com/en/5.0/
- https://docs.github.com/es
- https://git-scm.com/doc