

Informe de Django 05

Tema: Django 5

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
09	Django 5	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 teoría de Django5. Ir haciendo commits cada avance. Compartirlo con el docente (CarloCorrales010)

2. Entregables

- Informe hecho en Latex.
- 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
- Uso de Vistas Genericas
- Archivos estaticos

4. Captura de Pantalla

```
Usuario@DESKTOP-G0S7RR6 MINGW64 /D/UNSA/pw2/teoria/Django2/src (master)
$ git log --graph --pretty=oneline --abbrev-commit --all
* 2da4421 (HEAD -> master, origin/master) Finalizacion de Django 5, se agrego una imagen en la plantilla bas
e.html, ademas se configuro en settings.py la ubicacion de los archivos static
* c8186af Modificando vista PersonaQueryView, donde se logra retornar un Json, se importo JsonResponse en ve
z de HttpResponse, ademas se uso un filtro que trata sobre obtener todos los objetos que tienen menor o igua
l a 40 años
* a95be56 Definiendo vista PersonaQueryView importando view y HttpResponse, además se definio rutas para dic
ha vista
* 597774f Se agrego vista PersonaDeleteView, se importo reverse_lazy y DeleteView, ademas se implemento la p
lantilla persona_confirm_delete.html donde se visualizara el objeto que se desea eliminar.Se definio las rut
as correspondientes
* 230110e Se implemento vista PersonaUpdateView, se importo UpdateView, de igual manera se definio las rutas
correspondientes y se reutilizo el formulario de createView
* 9b2f674 Se crea una nueva vista PerdonaCreateView, se importo CreateView, además se creo la plantilla de p
ersona_form.html donde se visualizara el formulario, se definio las rutas correspondientes para la vista
* 86c4562 Se creo nuevo template persona_detail.html donde se visualizara algunos datos de los objetos como
la edad o si es donador o no, además se corrigio errores de identacion en el template de persona_list.html
* e30bb44 Agregando vista PersonaDetailView, donde se definieron las rutas correspondientes para dicha vista
, ademas se modifiko el Modelo persona en la funcion get_absolute_url
* b668ce7 Aplicando filtros en la vista PersonalListView para poder extraer aquellos objetos que tengan edad
menor o igual a 10
* 8d4175b Modificando funcion get_absolute_url del modelo Persona, la funcion reverse tiene como argumento p
ersonas:persona-list
* 3c47007 Modificando plantilla base.html, donde se le quito su contenido anterior, y se colocotag block tit
le y tag block body, además se renombro la plantilla personasList a persona_list.html
* 8b0ca13 Iniciando Django 5, se hizo uso de views.generic, ademas de definir las rutas de la vista Personal
istView, se elimino las demas rutas para centrarnos es dicha vista
```

Figura 1: commits - local

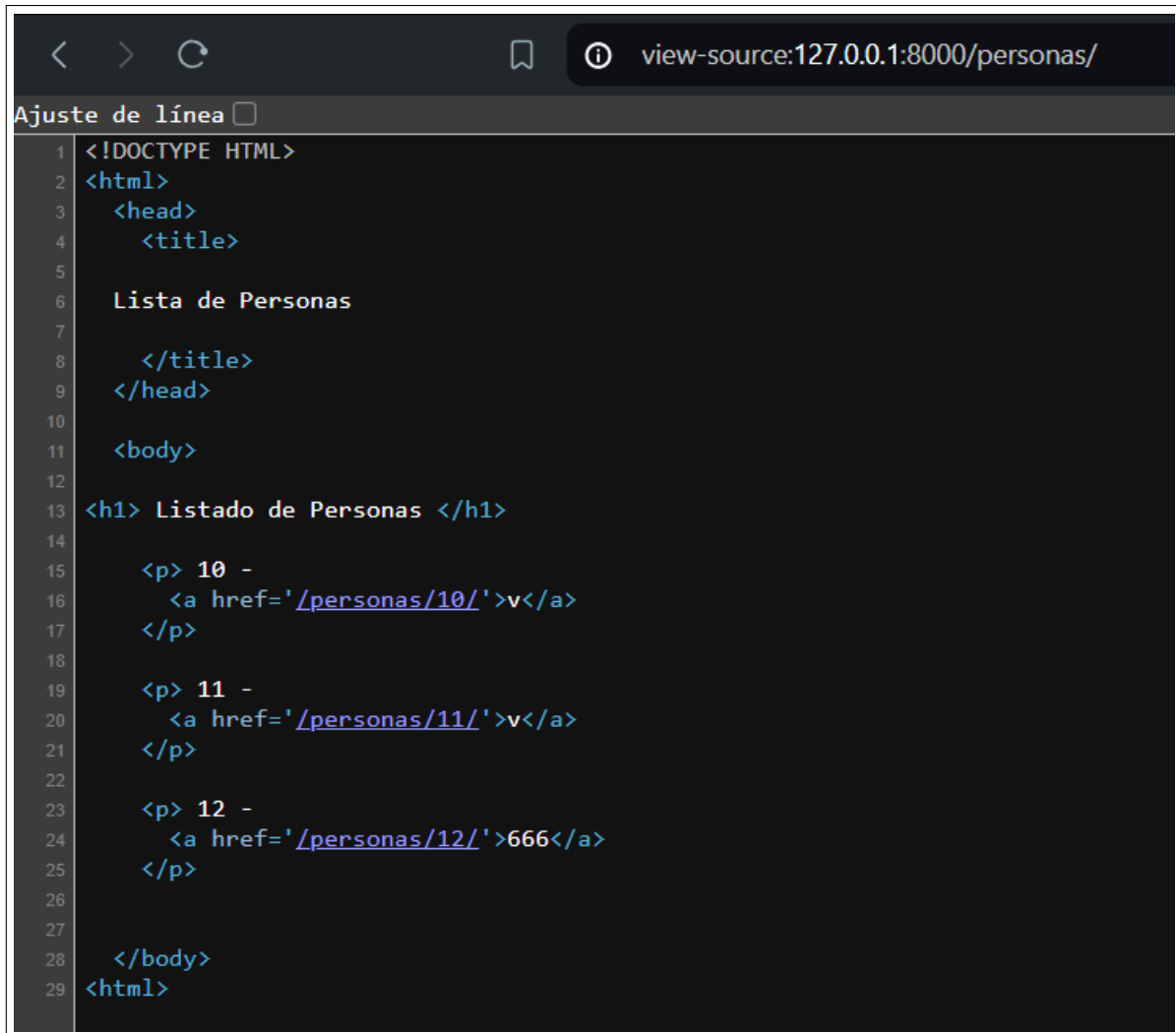
5. URL de Repositorio Github

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

6. Desarrollo del trabajo

Continuamos a partir del proyecto realizado en Django 4

6.1. Imágenes de Ejecución



```
1 <!DOCTYPE HTML>
2 <html>
3   <head>
4     <title>
5
6     Lista de Personas
7
8     </title>
9   </head>
10
11   <body>
12
13   <h1> Listado de Personas </h1>
14
15   <p> 10 -
16     <a href='/personas/10/'>v</a>
17   </p>
18
19   <p> 11 -
20     <a href='/personas/11/'>v</a>
21   </p>
22
23   <p> 12 -
24     <a href='/personas/12/'>666</a>
25   </p>
26
27
28   </body>
29 </html>
```

Figura 2: View Source

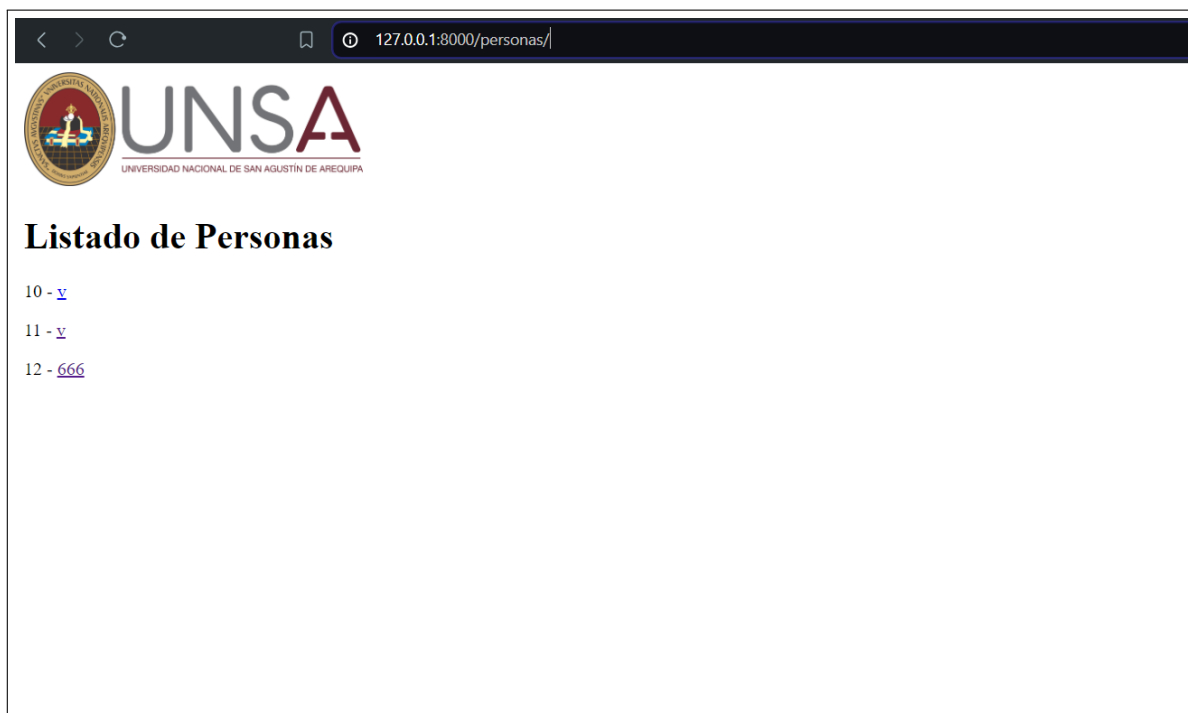


Figura 3: Listar Objetos

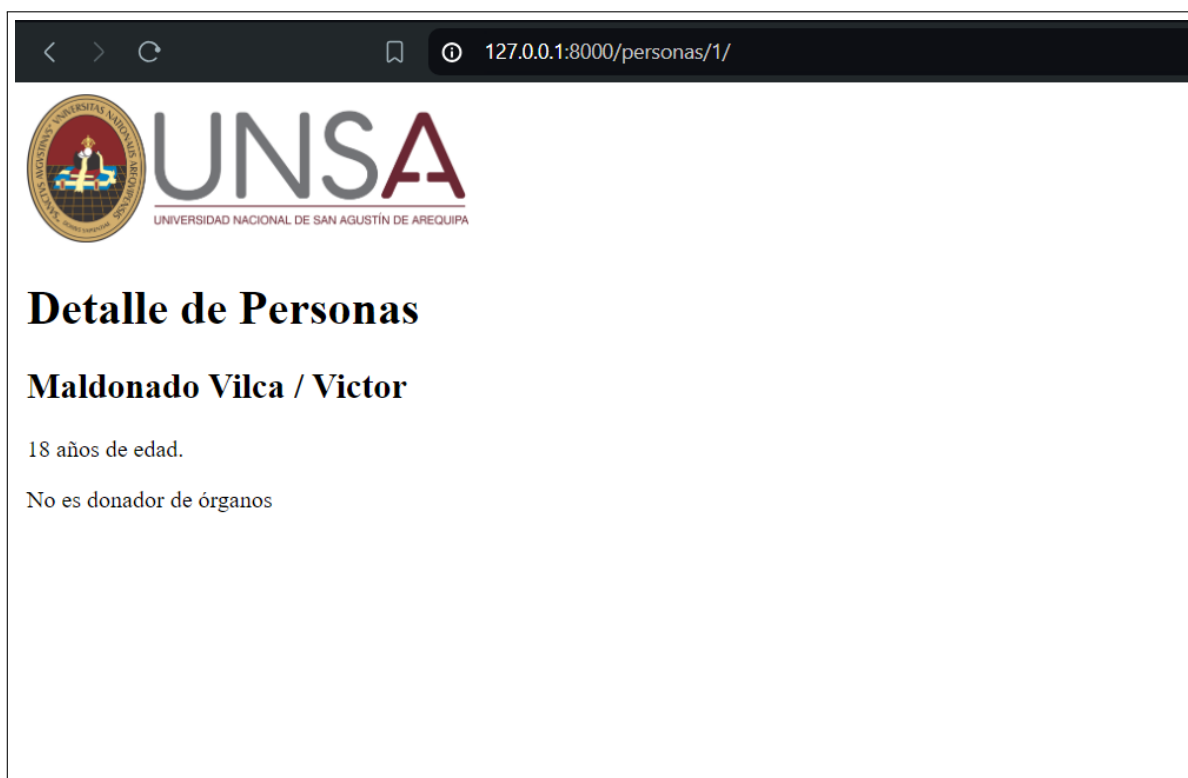
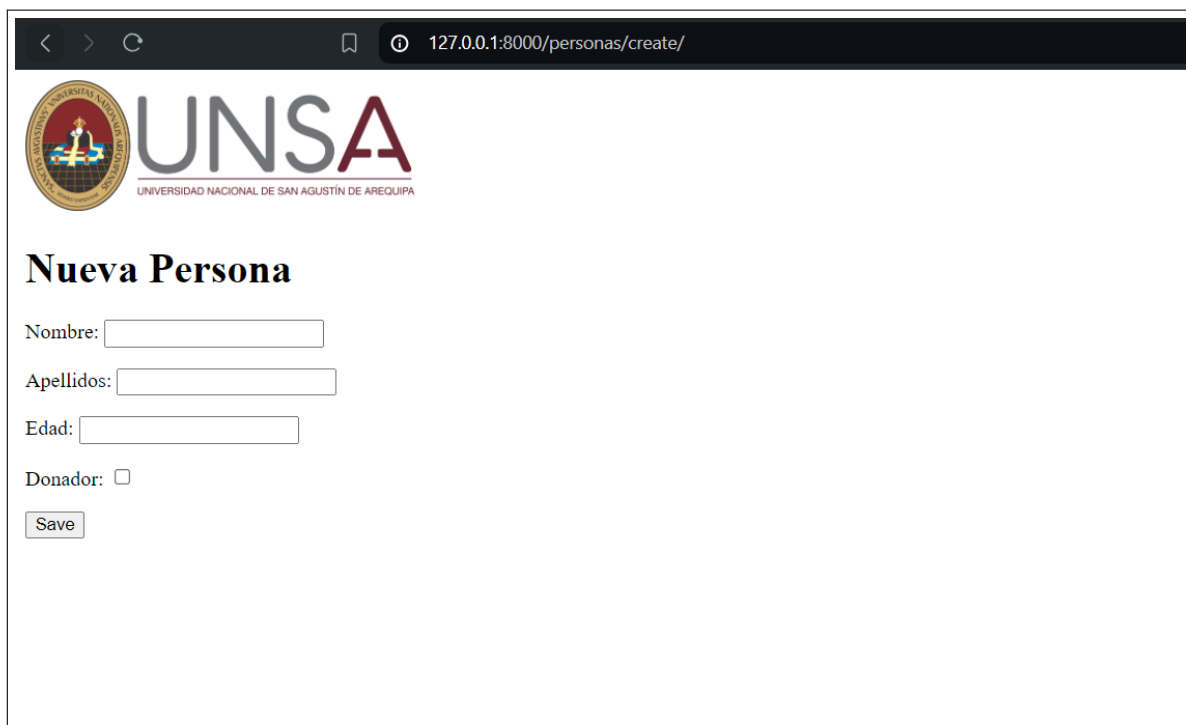



Figura 4: Detallar Objetos



127.0.0.1:8000/personas/create/

 **UNSA**
UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

Nueva Persona

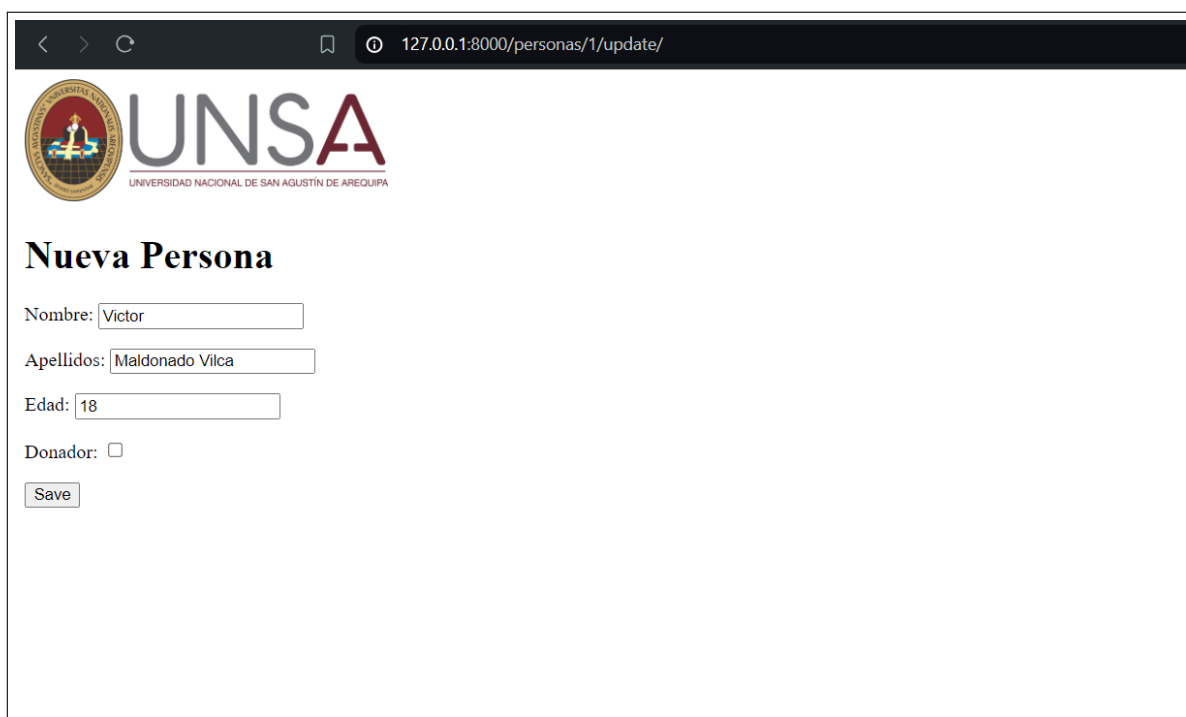
Nombre:

Apellidos:


Edad:

Donador: ☐

Figura 5: Agregar Objetos



127.0.0.1:8000/personas/1/update/

 **UNSA**
UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

Nueva Persona

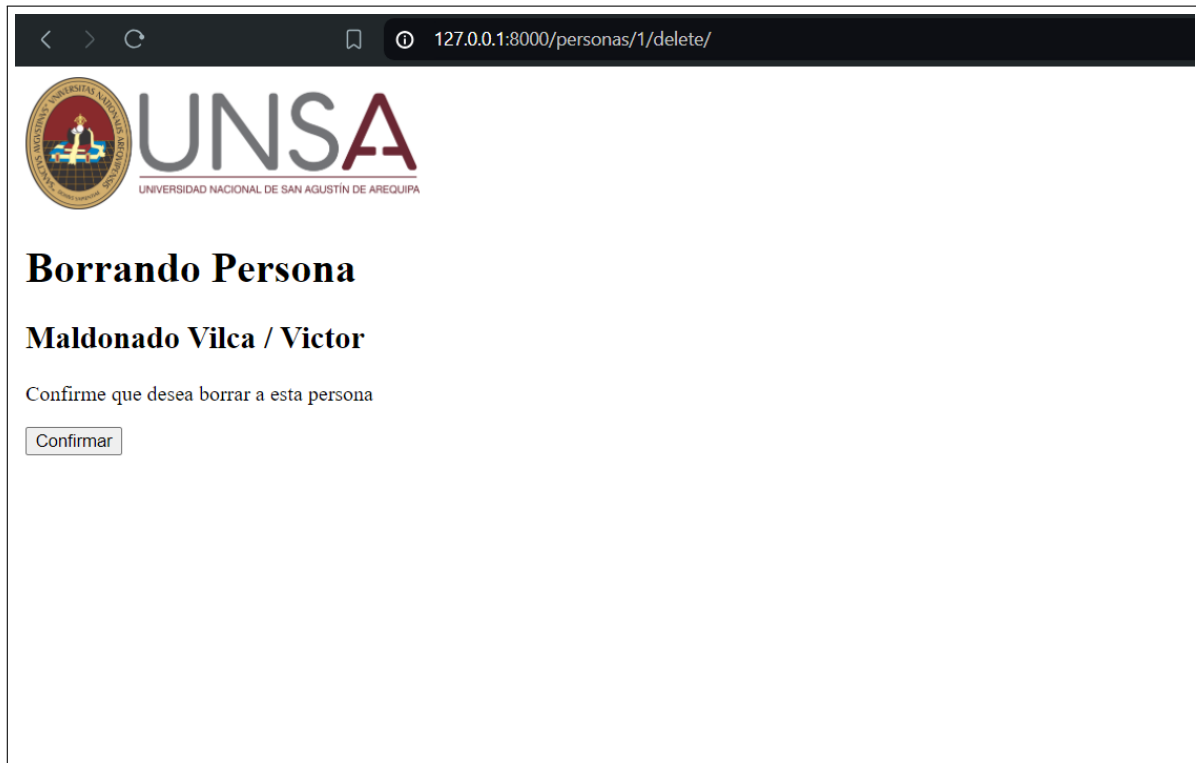
Nombre:

Apellidos:


Edad:

Donador: ☐

Figura 6: Actualizar Objetos



127.0.0.1:8000/personas/1/delete/

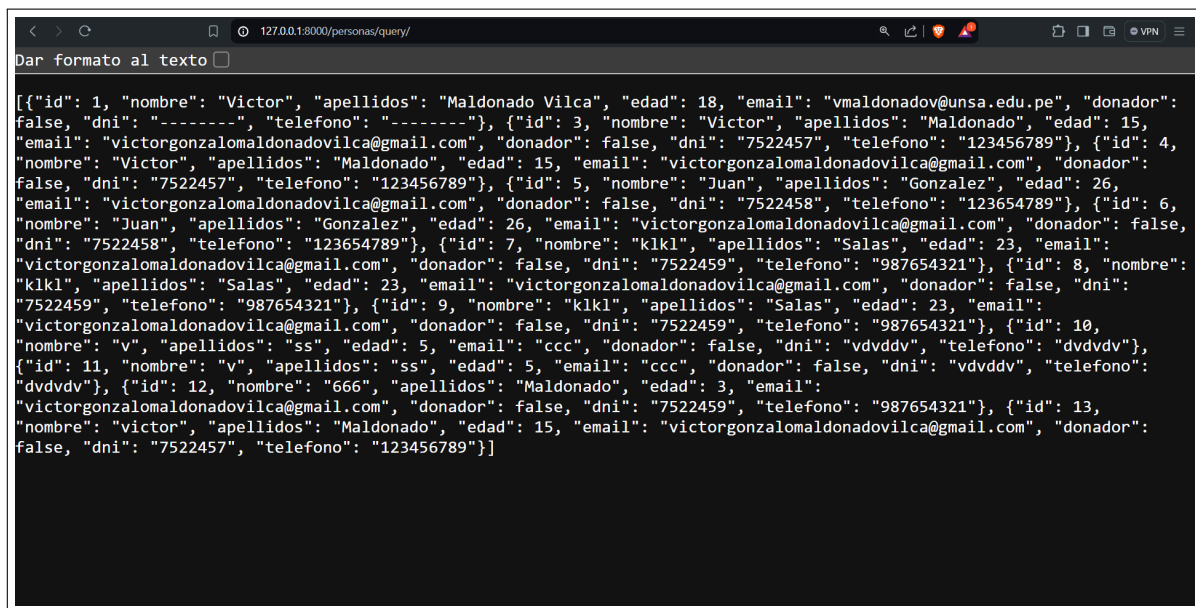
 **UNSA**
UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

Borrando Persona

Maldonado Vilca / Victor

Confirme que desea borrar a esta persona

Figura 7: Borrar Objetos



127.0.0.1:8000/personas/query/

Dar formato al texto ☐

```
[{"id": 1, "nombre": "Victor", "apellidos": "Maldonado Vilca", "edad": 18, "email": "vmaldonadov@unsa.edu.pe", "donador": false, "dni": "-----", "telefono": "-----"}, {"id": 3, "nombre": "Victor", "apellidos": "Maldonado", "edad": 15, "email": "victorgonzalomaldonadovilca@gmail.com", "donador": false, "dni": "7522457", "telefono": "123456789"}, {"id": 4, "nombre": "Victor", "apellidos": "Maldonado", "edad": 15, "email": "victorgonzalomaldonadovilca@gmail.com", "donador": false, "dni": "7522457", "telefono": "123456789"}, {"id": 5, "nombre": "Juan", "apellidos": "Gonzalez", "edad": 26, "email": "victorgonzalomaldonadovilca@gmail.com", "donador": false, "dni": "7522458", "telefono": "123654789"}, {"id": 6, "nombre": "Juan", "apellidos": "Gonzalez", "edad": 26, "email": "victorgonzalomaldonadovilca@gmail.com", "donador": false, "dni": "7522458", "telefono": "123654789"}, {"id": 7, "nombre": "klkl", "apellidos": "Salas", "edad": 23, "email": "victorgonzalomaldonadovilca@gmail.com", "donador": false, "dni": "7522459", "telefono": "987654321"}, {"id": 8, "nombre": "klkl", "apellidos": "Salas", "edad": 23, "email": "victorgonzalomaldonadovilca@gmail.com", "donador": false, "dni": "7522459", "telefono": "987654321"}, {"id": 9, "nombre": "klkl", "apellidos": "Salas", "edad": 23, "email": "victorgonzalomaldonadovilca@gmail.com", "donador": false, "dni": "7522459", "telefono": "987654321"}, {"id": 10, "nombre": "v", "apellidos": "ss", "edad": 5, "email": "ccc", "donador": false, "dni": "vddv", "telefono": "vddv"}, {"id": 11, "nombre": "v", "apellidos": "ss", "edad": 5, "email": "ccc", "donador": false, "dni": "vddv", "telefono": "vddv"}, {"id": 12, "nombre": "666", "apellidos": "Maldonado", "edad": 3, "email": "victorgonzalomaldonadovilca@gmail.com", "donador": false, "dni": "7522459", "telefono": "987654321"}, {"id": 13, "nombre": "victor", "apellidos": "Maldonado", "edad": 15, "email": "victorgonzalomaldonadovilca@gmail.com", "donador": false, "dni": "7522457", "telefono": "123456789"}]
```

Figura 8: JSON de los Objetos - query

6.2. Modelos – Aplicación personas

Esta función `get_absolute_url` genera la URL absoluta para una instancia del modelo utilizando el nombre de la ruta `personas:persona-detail` y el identificador primario (`pk`) del objeto. Esto permite obtener una URL directa para visualizar los detalles de la instancia correspondiente.

Listing 1: Modelo Persona

```
def get_absolute_url(self):  
    return reverse('personas:persona-detail', kwargs = {'pk': self.id})
```

6.3. Vistas – Aplicación personas

6.3.1. Importaciones

Se importa `render` desde `django.shortcuts` para renderizar plantillas HTML. Se importa `reverse_lazy` desde `django.urls` para la construcción de URLs reversibles y `JsonResponse` desde `django.http` para devolver respuestas JSON. Las vistas genéricas `ListView`, `DetailView`, `CreateView`, `UpdateView`, `DeleteView` y `View` se importan desde `django.views.generic` para manejar las diferentes operaciones CRUD y vistas personalizadas. Finalmente, se importa el modelo `Persona` desde `.models` para interactuar con los datos del modelo en las vistas.

```
from django.shortcuts import render  
from django.urls import reverse_lazy  
from django.http import JsonResponse  
from django.views.generic import (  
    ListView,  
    DetailView,  
    CreateView,  
    UpdateView,  
    DeleteView,  
    View,  
)  
from .models import Persona
```

6.3.2. Vista `PersonaQueryView()`

■ Descripción:

- Esta vista personalizada hereda de `View` y define el método `get`.
- Filtra las instancias del modelo `Persona` cuya edad es menor o igual a 40.
- Retorna un `JsonResponse` con los datos de las instancias filtradas en formato JSON.

■ Código:

```
class PersonaQueryView(View):  
    def get(self, request, *args, **kwargs):  
        queryset = Persona.objects.filter(edad__lte='40')  
        return JsonResponse(list(queryset.values()), safe = False)
```

6.3.3. Vista PersonaDetailView()

■ Descripción:

- Esta vista hereda de DetailView y se usa para mostrar los detalles de una instancia específica del modelo Persona.
- Django automáticamente busca una plantilla llamada personaí_detail.html y pasa la instancia del modelo a esta plantilla.

■ Código:

```
class PersonaDetailView(DetailView):  
    model = Persona
```

6.3.4. Vista PersonaListView()

■ Descripción:

- Hereda de ListView y se utiliza para mostrar una lista de instancias del modelo Persona
- Filtra las instancias cuya edad es menor o igual a 40.
- Django busca una plantilla llamada persona_list.html y pasa la lista de instancias a esta plantilla.

■ Código:

```
class PersonaListView(ListView):  
    model = Persona  
    queryset = Persona.objects.filter(edad__lte='40')
```

6.3.5. Vista PersonaCreateView()

■ Descripción:

- Hereda de CreateView y se utiliza para crear nuevas instancias del modelo Persona.
- Define los campos del modelo que estarán disponibles en el formulario de creación.
- Django busca una plantilla llamada persona_form.html para renderizar el formulario.

■ Código:

```
class PersonaCreateView(CreateView):  
    model = Persona  
    fields = [  
        'nombre',  
        'apellidos',  
        'edad',  
        'donador',  
    ]
```


6.3.6. Vista `PersonaUpdateView()`

■ Descripción:

- Hereda de `UpdateView` y se utiliza para actualizar instancias existentes del modelo `Persona`.
- Define los campos del modelo que estarán disponibles en el formulario de actualización.
- Django busca una plantilla llamada `persona_form.html` para renderizar el formulario de actualización.

■ Código:

```
class PersonaUpdateView(UpdateView):  
    model = Persona  
    fields = [  
        'nombre',  
        'apellidos',  
        'edad',  
        'donador',  
    ]
```

6.3.7. Vista `PersonaDeleteView()`

■ Descripción:

- Hereda de `DeleteView` y se utiliza para eliminar instancias del modelo `Persona`.
- Define `success_url` para redirigir a la lista de personas después de eliminar una instancia.
- Django busca una plantilla llamada `persona_confirm_delete.html` para confirmar la eliminación.

■ Código:

```
class PersonaDeleteView>DeleteView):  
    model = Persona  
    success_url = reverse_lazy('personas:persona-list')
```

6.4. URLs

6.4.1. urls del proyecto `listaContactos`

- **Descripción:** Este archivo `urls.py` principal de un proyecto Django define las rutas URL que gestionan las solicitudes. Importa módulos necesarios y vistas, aunque no se usan en este archivo. La lista `urlpatterns` contiene rutas que emparejan solicitudes con vistas: `path('admin/', admin.site.urls)` para la interfaz de administración accesible a través de `/admin/`, y `path('personas/', include('personas.urls'))` para incluir las rutas definidas en la aplicación `personas`, accesibles a través de `/personas/`

■ Código:

```
from django.contrib import admin  
from django.urls import path, include  
from inicio.views import myHomeView, anotherView  
  
urlpatterns = [  
    path('admin/', admin.site.urls),
```

```
path('personas/', include('personas.urls')),  
]
```

6.4.2. urls de la aplicación personas

- **Descripción:** Se define las rutas URL específicas para la aplicación personas. Se importan las vistas necesarias desde `personas.views` y se asigna el nombre de la aplicación como `personas`. La lista `urlpatterns` contiene las siguientes rutas: `path("")` para la lista de personas con `PersonaListView`, `path('int:pk/')` para el detalle de una persona con `PersonaDetailView`, `path('create/')` para la creación de una nueva persona con `PersonaCreateView`, `path('int:pk/update/')` para la actualización de una persona existente con `PersonaUpdateView`, `path('int:pk/delete/')` para la eliminación de una persona con `PersonaDeleteView`, y `path('query/')` para consultar personas según ciertos criterios con `PersonaQueryView`.

- **Código:**

```
from django.urls import path  
from personas.views import (  
    PersonaListView,  
    PersonaDetailView,  
    PersonaCreateView,  
    PersonaUpdateView,  
    PersonaDeleteView,  
    PersonaQueryView,  
)  
  
app_name = 'personas'  
urlpatterns = [  
    path('', PersonaListView.as_view(), name = 'persona-list'),  
    path('<int:pk>', PersonaDetailView.as_view(), name = 'persona-detail'),  
    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'),  
    path('query/', PersonaQueryView.as_view(), name = 'persona-query'),  
]
```

7. Templates

En esta actividad se realizó 5 templates

7.1. Plantilla modificada base.html

- **Descripción:** La siguiente plantilla HTML utiliza `load static` para cargar archivos estáticos como CSS, imágenes, etc. El bloque `block title` define el título de la página, que puede ser reemplazado en las plantillas que heredan de esta. El bloque `block body` se utiliza para el contenido principal de la página, que también puede ser reemplazado en las plantillas que heredan de esta. El código `static "personas/logo-unsas.png"` se utiliza para cargar la imagen del logo de la UNSA desde los archivos estáticos.

- **Código:**

```
{% load static %}  
<!DOCTYPE HTML>
```

```
<html>
  <head>
    <title>
      {% block title %}
      {% endblock %}
    </title>
  </head>

  <body>
    <img src='{% static "personas/logo-unsa.png" %}' width='300px' alt='logo
      unsa'>
    {% block body %}
    {% endblock %}
  </body>
</html>
```

7.2. Plantilla persona_list.html

- **Descripción:** Esta plantilla HTML extiende de 'base.html'. En el bloque block title, se define el título de la página como "Lista de Personas". En el bloque block body, se muestra un encabezado **h1** con el texto "Listado de Personas", seguido de un bucle for instance in object_list que itera sobre una lista de objetos instance y muestra su ID y nombre en párrafos **p**. Cada nombre es un enlace a la URL definida por get_absolute_url para ese objeto.

- **Código:**

```
{% extends 'base.html' %}
{% block title%}
  Lista de Personas
{% endblock %}
{% block body %}
<h1> Listado de Personas </h1>
{% for instance in object_list %}
  <p> {{instance.id}} -
    <a href='{instance.get_absolute_url}'>{{instance.nombre}}</a>
  </p>
{% endfor %}
{% endblock %}
```

7.3. Plantilla persona_detail.html

- **Descripción:** Este es un archivo de plantilla HTML que extiende de 'base.html'. En el bloque block title, se define el título de la página como "Detalle de Personas". En el bloque block body, se muestra un encabezado **h1** con el texto "Detalle de Personas", seguido de un encabezado **h2** que muestra los apellidos y nombres del objeto object. Luego, se muestra la edad del objeto y un mensaje que indica si es donador de órganos o no, dependiendo del valor de object.donador.

- **Código:**

```
{% extends 'base.html' %}
{% block title%}
  Detalle de Personas
{% endblock %}
{% block body %}
```

```
<h1> Detalle de Personas </h1>
<h2> {{ object.apellidos }} / {{ object.nombres }}</h2>
<p>{{ object.edad }} .... de edad.</p>
<p>
{% if object.donador %}
    Es
{% else %}
    No es
{% endif %}
    donador de organos
</p>
{% endblock %}
```

7.4. Plantilla persona_form.html

- **Descripción:** Este archivo de plantilla HTML extiende de 'base.html' y define el título de la página como "Nueva Persona.^{en} el bloque block title. En el bloque block body, se muestra un encabezado h1 con el texto "Nueva Persona", seguido de un formulario que utiliza el método POST y el token CSRF para seguridad. El formulario renderiza los campos del formulario form utilizando form.as_p y agrega un botón "Save" para enviar el formulario.
- **Código:**

```
{% extends 'base.html' %}
{% block title%}
    Nueva Persona
{% endblock %}
{% block body %}
<h1> Nueva Persona </h1>
<form method='post'>{% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Save">
</form>
{% endblock %}
```

7.5. Plantilla persona_confirm_delete.html

- **Descripción:** Se define una plantilla HTML que se extiende de 'base.html' y establece el título de la página como "Borrando Persona.^{en} el bloque block title. En el bloque block body, se muestra un encabezado h1 con el texto "Borrando Persona", seguido de un subtítulo h2 que muestra los apellidos y nombre de la persona a borrar. Luego, se presenta un formulario que utiliza el método POST y el token CSRF para seguridad, con un mensaje de confirmación y un botón "Confirmar" para proceder con la eliminación de la persona.
- **Código:**

```
{% extends 'base.html' %}
{% block title%}
    Borrando Persona
{% endblock %}
{% block body %}
<h1> Borrando Persona </h1>
<h2>{{ object.apellidos }} / {{ object.nombre }}</h2>
<form method='post'>{% csrf_token %}
```

```
<p>
  Confirme que desea borrar a esta persona
</p>
<input type="submit" value="Confirmar">
</form>
{% endblock %}
```

7.6. 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
```

8. Conclusiones

- El uso de archivos estáticos mejora la experiencia del usuario al permitir un diseño más atractivo y funcional.
- La compatibilidad con herramientas y frameworks como Django asegura una gestión eficiente de los archivos estáticos para una experiencia web sólida y profesional.
- Se ha adquirido un conocimiento significativo en el desarrollo con Django, cubriendo aspectos importantes como la configuración de URL, la definición de vistas y la creación de formularios.
- La comprensión de la estructura de los proyectos en Django, que incluye la organización de aplicaciones, modelos y archivos de configuración, resulta fundamental para lograr un desarrollo eficiente.
- La utilización de herramientas como el servidor de desarrollo de Django y el sistema de plantillas ha facilitado un desarrollo ágil y eficaz de la aplicación.
- Django se destaca como un framework web poderoso y versátil que permite la rápida y eficiente creación de aplicaciones web.
- La arquitectura MVC (Modelo-Vista-Controlador) de Django contribuye a organizar el código de manera estructurada y modular, lo que favorece la mantenibilidad y escalabilidad de las aplicaciones.
- Con un sólido conocimiento de Django y al seguir las mejores prácticas de desarrollo, es posible crear aplicaciones web robustas y de alto rendimiento.

8.1. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple 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 los ítems.
- 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 están 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	

9. Referencias

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