Informe de Django 2

Tema: Django 2

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
06	Django 2	2 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 4 de junio de 2024	Al 8 de junio de 2024

1. Introducción

Django es un potente framework de desarrollo web en Python que facilita la creación de aplicaciones web robustas y escalables. Entre sus características destacadas se encuentran el modelado de datos, la gestión de plantillas y las validaciones de formularios.

2. Objetivos

- Comprender el uso de blank=False: Asegurarse de que se entiende cómo y cuándo utilizar blank=False para garantizar la integridad de los datos en los modelos de Django.
- Manejar plantillas eficientemente: Aprender a utilizar el sistema de plantillas de Django, incluyendo el uso de bloques de contenido para crear interfaces de usuario dinámicas y reutilizables.
- Desarrollar aplicaciones web escalables: Aplicar los conocimientos sobre modelos y plantillas para construir aplicaciones web robustas y escalables con Django.

3. Tarea

- En esta tarea deberá seguir las diapositivas de DJango02, dentro de un proyecto git local.
- Deberá hacer un commit por cada paso y deberá usar branch para hacer sus experimentos o pruebas.

- La entrega de la tarea consistirá de texto con las respuestas a las preguntas de las diapositivas (incluya el enunciado de la diapositiva) y una captura de pantalla (deberá incrustar la captura de pantalla al inicio de su texto de respuesta) del siguiente comando:

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

- Cada commit debe ser realizado con un mensaje descriptivo del paso de la diapositiva que estuvo siguiendo

4. 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
```

- **Extra:** URL - Repositorio GitHub

5. 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
- Autenticación y Autorización de Usuarios
- Administración de Aplicaciones
- Control de Versiones
- Gestión de Ramas
- Resolución de Conflictos en Git

6. Respuestas a las Preguntas dejadas en las Diapositivas

1. **¿Qué pasa si añadimos un nuevo campo? los anteriores registros no lo tendrían, entonces ¿Con qué valor se actualizarán?**
   Respuesta: Si añadimos un nuevo campo en los modelos de nuestra aplicación, Al querer hacer las migraciones correspondientes, Django no nos dejará hacerlo, para ello se requiere crear ese campo con un valor por defecto uso de default. Lo que sucede con los anteriores registros es que no cambian sus datos correspondientes pero si se agrega el campo nuevo con el valor por defecto, osea el valor del nuevo campo se actualizara con el valor por defecto.

2. **¿Cómo crearía un campo que sea obligatorio?**
   Respuesta: Para crear un campo obligatorio se usa el parámetro 'blank=False' en la definición del campo del modelo, lo que significa que el campo no se puede dejar en blanco y es obligatorio proporcionar algún valor para dicho campo al crear una instancia del modelo correspondiente.

3. **¿Cuáles de estos elementos afectarían a la base de datos?¿Cuáles no?**
   Respuesta: Los elementos que afectán a la base de datos son 'null=True' y 'default=True' ya que estos tiene un impacto directo en la estructura y valores de la base de datos, mientras que 'blank=False' solo influye en la interfaz del usuario y en la validación de formularios.

7. Captura de Pantalla

```
Usuario@DESKTOP-G0S7RR6 MINGW64 /D/unsa/pw2/teoria/Django2/src (master)
$ git log --graph --pretty=oneline --abbrev-commit --all
*   61d8930 (HEAD -> master, origin/master) Finalizacion del proyecto -- Juntando la rama prueba
con master, se dio la ultima prueba que consta en crear una nueva plantilla y usar el tag include
 para incluirla en nav.html
|\
| * 675c47a (origin/pruebas, pruebas) Creando nueva plantilla contenedor.html y uso del tag inclu
de para anexarlo con nav.html
* | dd75c0e Implementando mas items en la lista y un header como prueba
|\|
| * d3e1066 Cambiando la plantilla nav.html
* | 0a3976a Agregando un item de la lista, para analizar funcionamiento de etiqueta tag
|\|
| * 0fc901f Error en paso de ramas, se solucionara posteriormente
* | 1a5c8e4 Colocando tag include para anexar la barra de navegacion con base.html
* | 87023a0 Se creo la plantilla nav con una estructura html basica
* | a69e700 Agregar tag extends en home.html para añadir etiquetas e información de base.home
* | 3b7b1b7 Añadir el archivo base.html como plantilla base para páginas HTML
* | 97c1cb6 Uso de variables de contexto en la plantilla home.html
* | 6bd18a0 Logrando que la vista myHomeView redirecciona a la pagina home.html, uso de request
* | 94a12ac Agregar carpeta templates, ademas e hizo estructura basica html en home.html
* | 3702708 Agregar la configuración de la carpeta de templates en settings.py
* | 6bed2dc Agregar en la vista myHomeView imresión de argumentos y usuario
* | dc166d4 Agregando importacion de la vista anotherView
* | 16f3517 Actualizar la configuración de URLs en urls.py para agregar la vista anotherView
* | 18cfca9 Añadir la vista anotherView en views.py para mostrar otra página
* | 7d7b336 Actualizar la configuración de URLs en urls.py para agregar una nueva ruta
* | 9a5b203 Haciendo migraciones, Añadiendo configuración de URLs en urls.py para la página de in
icio
* | c62bff5 Implementar la función myHomeView en views.py de la app inicio para mostrar un mensaj
e de saludo en la página principal
* | cce0903 Creando aplicación inicio, se configuro en settings.py del proyecto para definir la a
pp
```

Figura 1: Primera Parte commits

```
* | 1ea14b3 ¿Cuáles de estos elementos afectarían  a la base de datos?¿Cuáles no? El unico elemen
to que afecta a la db es null=True, ya que permite que el campo tenga valores nulos, mientras que
 default solo hace que exista un dato por defecto en caso de que este no se ingrese y por ultimo
blank logra que el campo sea obligario pero no afecta la db
* | 99f989f Merge branch 'pruebas'
|\|
| * daa4795 Se hizo más pruebas para cambiar el dato por defecto en donador de False a True y uso
 de null=True que hace que la base de datos permita un valor nulo
| * 13c17bb ¿Cómo crearía un campo que sea obligatorio?, Se hizo una prueba donde los campos nomb
re y apellido son obligatorios, entonces la respuesta es: se crearía un campo obligatorio con bla
nk=False
* | 205ded7 Cambios en la base de datos antes de hacer un merge, se regresa antes de colocar null
=True,etc
* | bb66088 Solucionando errores de merge, git no puedo solucionar estos errores, consta en confl
ictos de versiones
* | 6bc614a Juntando rama pruebas con master, Implementando prueba de añadir un nuevo campo en es
te caso telefono
|\|
| * aa27610 Añadiendo nuevo campo telefono, Prueba para confirmar que si se requiere un dato por
defecto sino no se podra actualizar los cambios en la base de datos
* | 8955681 Agregando nuevo campo dni, se comprobo que Django requiere que coloquemos un dato por
 defecto al crear un nuevo campo en un modelo previamente implementado
* | 46c2a49 ¿Qué pasa si añadimos un nuevo campo?  los anteriores registros no lo tendrían, enton
ces ¿Con qué valor se actualizarán? Al añadir un nuevo campo Django no permitira que hagamos migr
aciones, al menos hasta que coloquemos un dato por defecto, a los anteriores registros también se
 les colocaria el campo pero con el dato seleccionado por defecto.(Añadir nuevo campo donador de
tipo boolean)
* | 20b25fb Agregando nuevo objeto en Persona, implementando fila en la base de datos llenando lo
s campos con los datos correspondientes
* | aba8f7f Aplicar migraciones para sincronizar la base de datos con los cambios en los modelos
* | 484a982 Agregar registro de modelo Persona en panel de administración
* | c3eca03 (origin/errores, errores) Solucionando errores en el modelo Persona
* | c78ab57 Configurando settings.py del proyecto para cambiar el lenguaje, zona horaria y agrega
ndo la app personas para utilizar las funcionalidades proporcionadas por esta aplicacion
|/
```

Figura 2: Segunda Parte commits

```
* 86c38ac Agregando nuevo campo en el modelo persona, en este caso es email
* 09c8cfb Creando super usuario, se añadio este usuario a la base de datos
* 861ac4c Detectar cambions y aplicar migraciones, para implementar cambios en la base de Datos c
omo la creación de tablas
* cfbbf72 Añadiendo modelo Persona con los campos nombre, apellidos y edad
* 31f6b46 Agregando archivo .gitignore para ignorar archivos o directorios que deben ser ignorado
s por el sistema de control de versiones.
* d0b1301 Creando la aplicación personas en el proyecto listaContactos
* 5e7cc8d Iniciando un proyecto en Django llamado lista de Contactos
* 968875c Colocando en el readme.md la actividad que se tendrá que hacer en Teoría PW2 - Django2
* dde8384 Primer commit
(END)
```

Figura 3: Tercera Parte commits

8. URL de Repositorio Github

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

9. Metodología

9.1. Creación del entorno de Trabajo

9.1.1. Carpeta de trabajo

Listing 1: Creación del Directorio

```
mkdir Django2 && cd Django2
```

9.1.2. Creación del Entorno Virtual

Listing 2: Creación Entorno virtual

```
virtualenv -p python3 .
```

9.1.3. Activar entorno Virtual

Listing 3: Activar Entorno Virtual

```
Scripts/activate
```

9.1.4. Instalar Django en el entorno Virtual

Listing 4: Instalar Django

```
pip install Django
```

9.1.5. Creación carpeta del proyecto

Listing 5: Carpeta src

```
mkdir src && cd src
```

9.1.6. Inicializar git

Listing 6: Comandos Iniciales

```
echo "# Django2" >> README.md
git init
git add README.md
git commit -m "Primer commit"
git branch -M master
git remote add origin https://github.com/Victor-Gonzalo-Maldonado-Vilca/Django2.git
git push -u origin master
```

9.1.7. Uso de ramas

Se usarán las siguientes ramas:

```
git branch pruebas
git branch errores
```

9.1.8. Crear .gitignore

Seguimos el siguiente Repositorio `https://github.com/django/django/blob/main/.gitignore`

9.2. Creacion del proyecto y apps

9.2.1. Crear Proyecto

Listing 7: Crear proyecto

```
django-admin startproject listaContactos
```

9.2.2. Crear Apps

Listing 8: Crear App

```
python manage.py startapp personas
python manage.py startapp inicio
```

10. Desarrollo del trabajo

Continuamos a partir del proyecto realizado en Django 1

10.1. Configuración settings.py del proyecto

10.1.1. Templates

```
57  TEMPLATES = [
58      {
59          'BACKEND': 'django.template.backends.django.DjangoTemplates',
60          'DIRS': [os.path.join(BASE_DIR, "templates")],
61          'APP_DIRS': True,
62          'OPTIONS': {
63              'context_processors': [
64                  'django.template.context_processors.debug',
65                  'django.template.context_processors.request',
66                  'django.contrib.auth.context_processors.auth',
67                  'django.contrib.messages.context_processors.messages',
68              ],
69          },
70      },
71  ]
```

Figura 4: Configuración de la carpeta templates

10.1.2. Install Apps

```
34  INSTALLED_APPS = [
35      'django.contrib.admin',
36      'django.contrib.auth',
37      'django.contrib.contenttypes',
38      'django.contrib.sessions',
39      'django.contrib.messages',
40      'django.contrib.staticfiles',
41      'personas',
42      'inicio',
43  ]
44
```

Figura 5: Definir aplicación inicio en el proyecto

10.2. Modelos – Aplicación personas

El modelo `Persona` en Django define los atributos que se utilizarán para almacenar información sobre una persona en la base de datos. Cada atributo tiene un tipo de datos específico y ciertas reglas asociadas:

- `nombre`: Se trata de un campo de texto que puede almacenar hasta 100 caracteres y es obligatorio.
- `apellidos`: Otro campo de texto que también puede contener hasta 100 caracteres y debe ser completado.
- `edad`: Este campo guarda un número entero que representa la edad de la persona.
- `email`: Aquí se almacena la dirección de correo electrónico de la persona, con un límite de 100 caracteres.
- `donador`: Un campo booleano que indica si la persona es donante o no. Por defecto, se establece como `True` si no se especifica lo contrario.
- `dni`: Se utiliza para guardar el número de documento nacional de identidad (DNI) de la persona, aunque este campo puede quedar vacío si no se proporciona.
- `telefono`: Este campo guarda el número de teléfono de la persona y también puede quedar sin completar si no se desea proporcionar esta información.

Estos atributos permiten almacenar información relevante sobre una persona de manera organizada en la base de datos de la aplicación Django. Después se harán las migraciones correspondientes.

Listing 9: Modelo Persona

```
class Persona(models.Model):
  nombre   = models.CharField(max_length = 100, blank=False)
  apellidos = models.CharField(max_length = 100, blank=False)
  edad     = models.IntegerField()
  email    = models.CharField(max_length = 100)
  donador  = models.BooleanField(default=True)
  dni      = models.CharField(max_length = 8, null=True)
  telefono = models.CharField(max_length = 9, null=True)
```

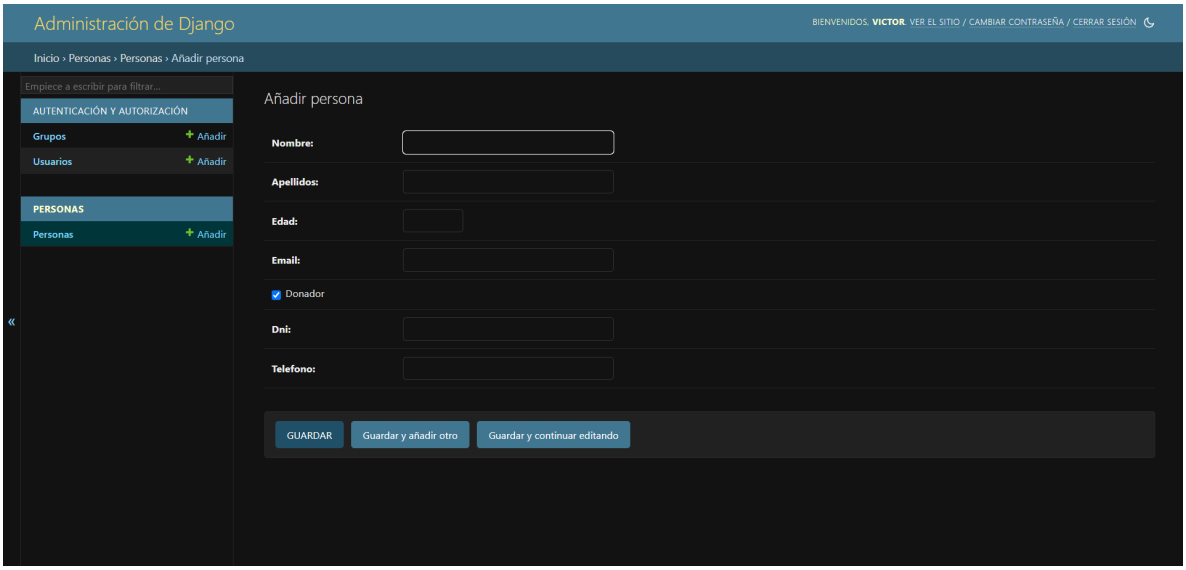

Figura 6: admin 1

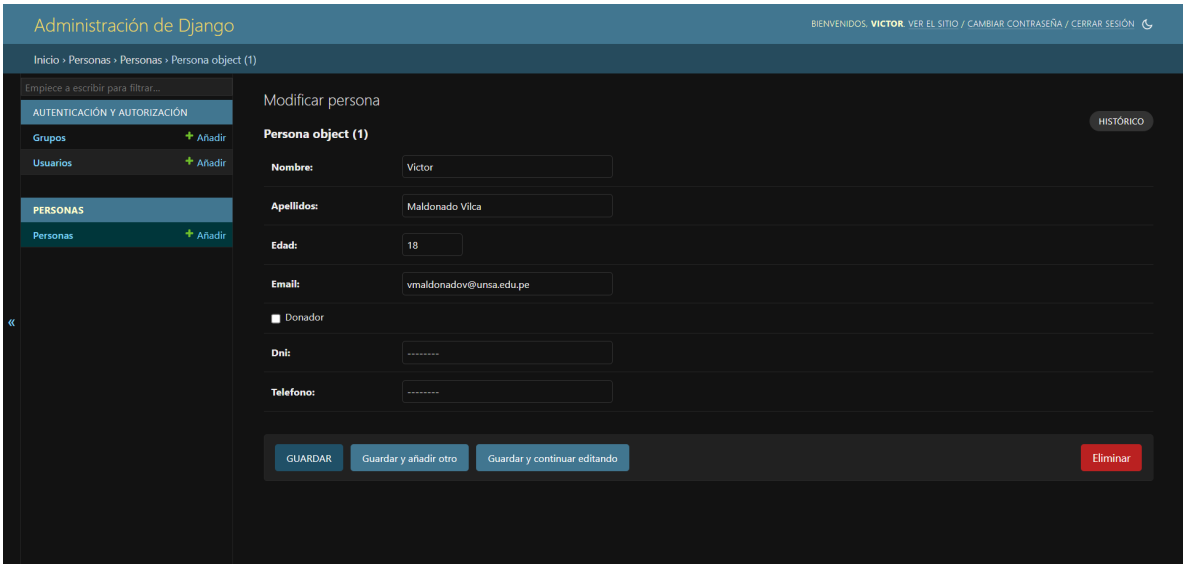

Figura 7: admin 2

10.3. Vistas – Aplicación inicio

El archivo de vistas en Django define dos vistas importantes para la aplicación web. La primera vista, llamada `myHomeView`, recibe un objeto `request` junto con argumentos adicionales, los cuales son impresos en la consola junto con el usuario asociado a la solicitud. Esta vista luego renderiza una plantilla llamada `home.html`, proporcionando una respuesta HTML basada en esta plantilla. La segunda vista, denominada `anotherView`, retorna una respuesta HTTP simple que contiene el texto "Solo otra página". Estas vistas permiten manejar diferentes rutas en la aplicación y generar respuestas dinámicas basadas en las solicitudes del usuario.

```
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.

def myHomeView(request, *args, **kwargs):
    print(args, kwargs)
    print(request.user)
    return render(request, "home.html", {})

def anotherView(request):
    return HttpResponse('<h1>Solo otra pagina</h1>')
```

10.4. URLs – proyecto listaContactos

El archivo de configuración de URLs en Django, define las rutas URL que la aplicación web manejará. Este archivo importa las vistas `myHomeView` y `anotherView` del módulo `inicio.views`, así como el módulo de administración de Django. La lista `urlpatterns` contiene las rutas definidas para la aplicación: la ruta raíz que utiliza `myHomeView` como la vista para la página de inicio, la ruta `another/` que utiliza `anotherView` para mostrar una página adicional, y la ruta `admin/` que proporciona acceso a la interfaz de administración de Django. Estas rutas permiten a la aplicación dirigir las solicitudes a las vistas correspondientes para generar las respuestas adecuadas.

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

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

11. Templates

11.1. Plantilla home.html

El siguiente código HTML define una plantilla base para una aplicación Django. Utiliza la sintaxis de plantillas de Django para extender una plantilla base llamada `base.html`. Dentro del bloque de contenido definido por 'block content', se inserta un mensaje "Hola Mundo desde Djangoz Çon Templates"dentro de las etiquetas h2 y h3, respectivamente. Adicionalmente, se muestra el usuario actual con 'request.user' y su estado de autenticación.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Template Base</title>
  </head>
  <body>
    {% extends 'base.html' %}
    {% block content %}
    <h2>Hola Mundo desde Django</h2>
    <h3>Con Templates</h3>
    {{ request.user }}
    <br>
    {{ request.user.is_authenticated }}
    {% endblock %}
  </body>
</html>
```

11.2. Plantilla base.html

El siguiente código HTML define una plantilla base para una aplicación Django. La plantilla incluye el archivo `nav.html` utilizando la sintaxis de inclusión de Django include 'nav.html'. Dentro del cuerpo de la página, se muestra un encabezado de nivel 1 con el texto .Este es un texto de Base". Además, se define un bloque de contenido utilizando la sintaxis de bloques de Django block content que inicialmente contiene el texto Reemplazame". Este bloque puede ser sobrescrito en plantillas que extiendan esta plantilla base.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Codigo para estudiantes desde Django</title>
  </head>
  <body>
    {% include 'nav.html' %}
    <h1> Este es un texto de Base</h1>
    {% block content %}
      Reemplazame
    {% endblock %}
  </body>
<html>
```

11.3. Plantilla nav.html

El siguiente código HTML define una estructura básica para una página web con un título y un menú de navegación. En la sección `<head>`, se establece el título de la página como "Navegación". En el cuerpo de la página (`<body>`), se incluye un elemento `<nav>` que contiene una lista desordenada (`<ul>`) con varios elementos de lista (`<li>`) representando diferentes secciones de la página: Home, Primera, Segunda, Tercera, Cuarta, Quinta y Sexta. Debajo del menú de navegación, se encuentra un encabezado de nivel 1 (`<h1>`) con el texto "Mi Página web usando Django". Finalmente, se incluye el contenido del archivo `contenedor.html` utilizando la sintaxis de inclusión de Django 'include 'contenedor.html".

```
<!DOCTYPE HTML>
<html>
```

```
    <head>
      <title>Navegacion</title>
    </head>
    <body>
      <nav>
        <ul>
          <li>Home</li>
          <li>Primera</li>
          <li>Segunda</li>
          <li>Tercera</li>
          <li>Cuarta</li>
          <li>Quinta</li>
          <li>Sexta</li>
        </ul>
      </nav>
      <h1>Mi Pagina web usando Django</h1>
      {% include 'contenedor.html' %}
    </body>
  </html>
```

11.4. Plantilla contenedor.html

El siguiente código HTML define una estructura simple de una página web que incluye un contenedor de contenido. En la sección `<head>`, se establece el título de la página como Çontenedor". En el cuerpo de la página (`<body>`), se encuentra un elemento `<div>` que contiene dos encabezados. El primer encabezado es de nivel 2 (`<h2>`) con el texto "Hola Mundo", y el segundo es de nivel 3 (`<h3>`) con el texto "DJANGO".

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Contenedor</title>
  </head>
  <body>
    <div>
      <h2>Hola Mundo</h2>
      <h3>DJANGO</h3>
    </div>
  </body>
</html>
```

11.5. 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
```

Figura 8: Path ”

Figura 9: Path ’another/’

11.6. Uso de GitHub

11.6.1. Usuario de GitHub

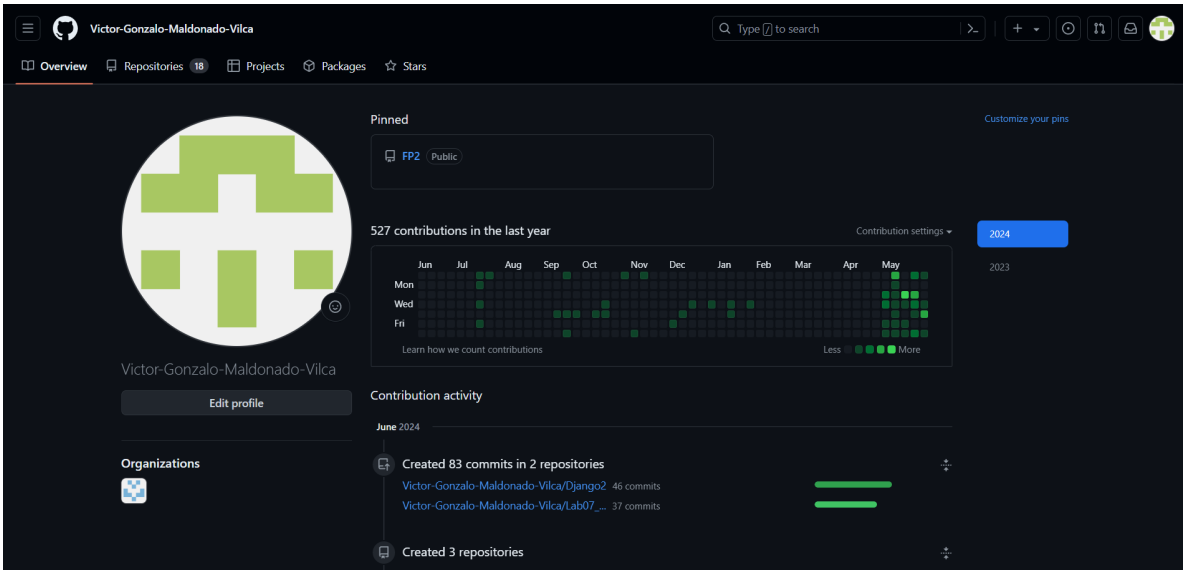

Figura 10: Usuario

11.6.2. Creación de un Nuevo Repositorio

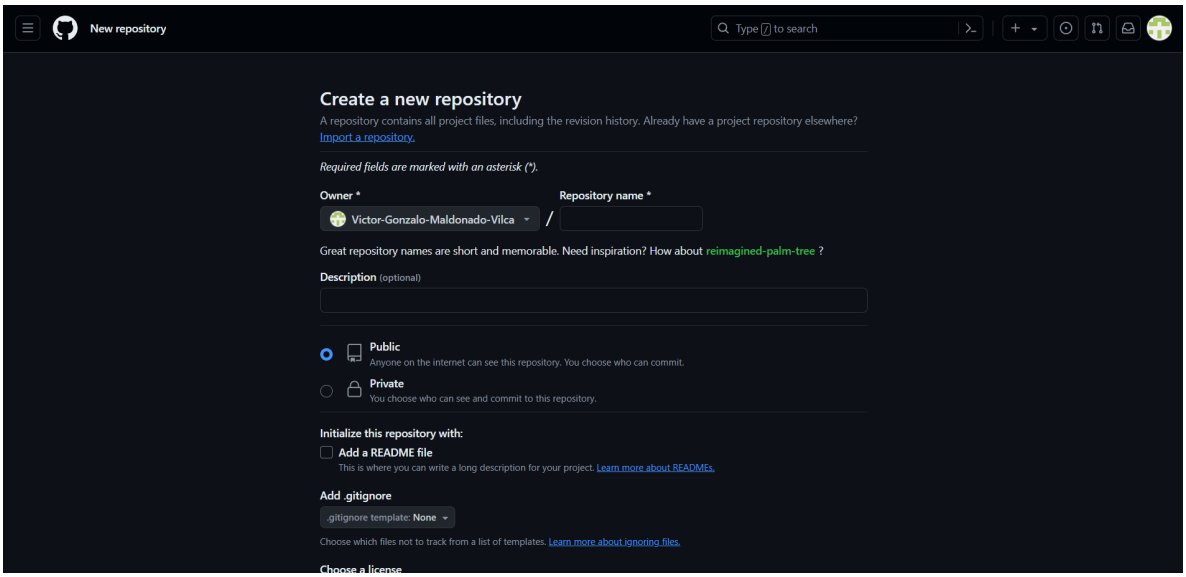

Figura 11: Crear Repositorio

11.6.3. Implementación de Readme.md

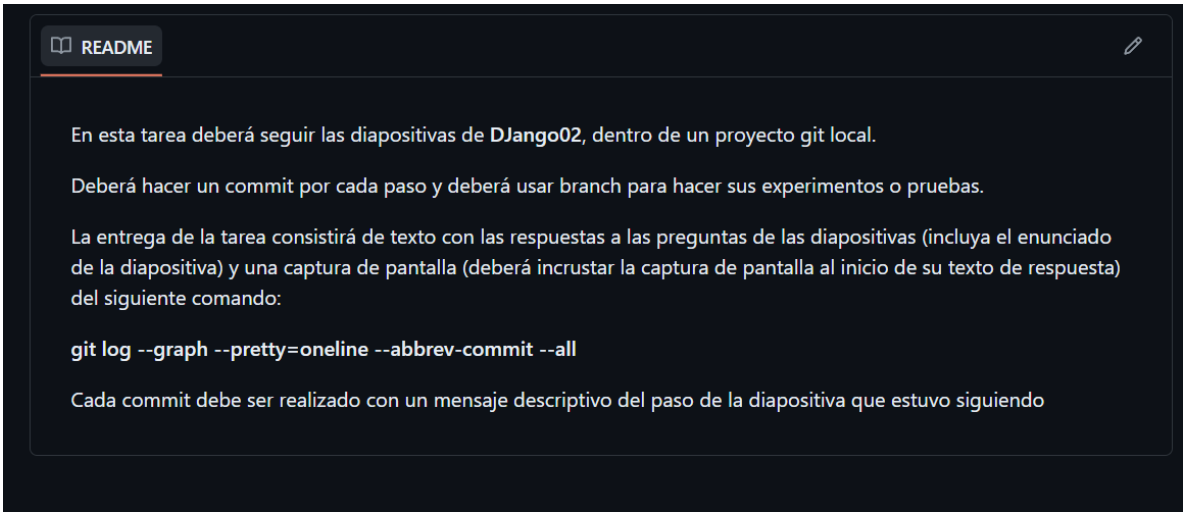

Figura 12: README.md

11.6.4. Registro de cambios en mi código

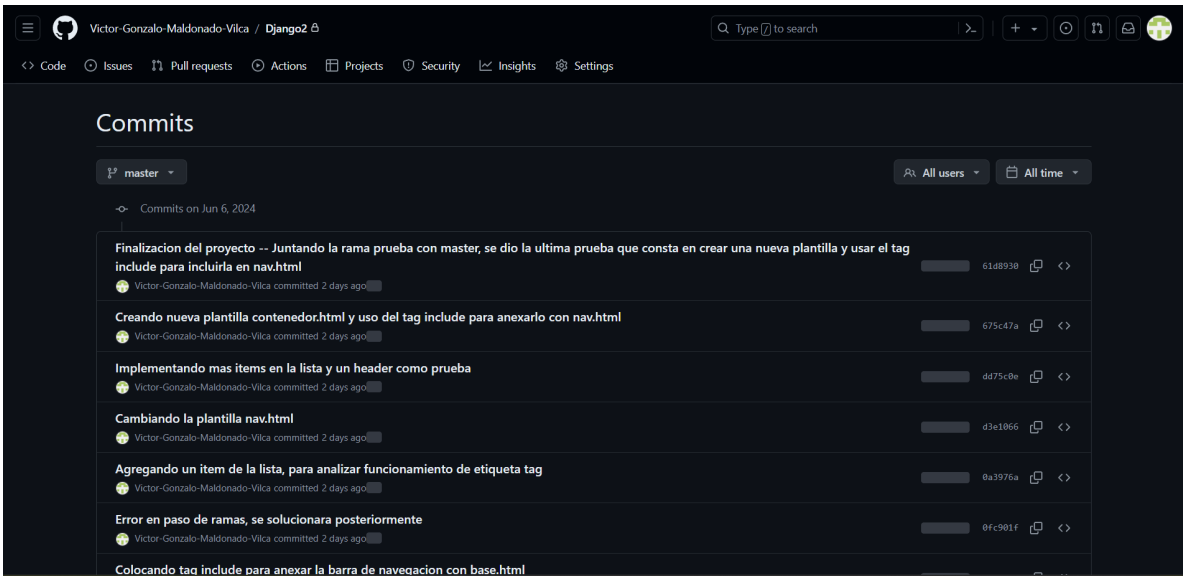

Figura 13: Commits

11.6.5. Uso de Ramas

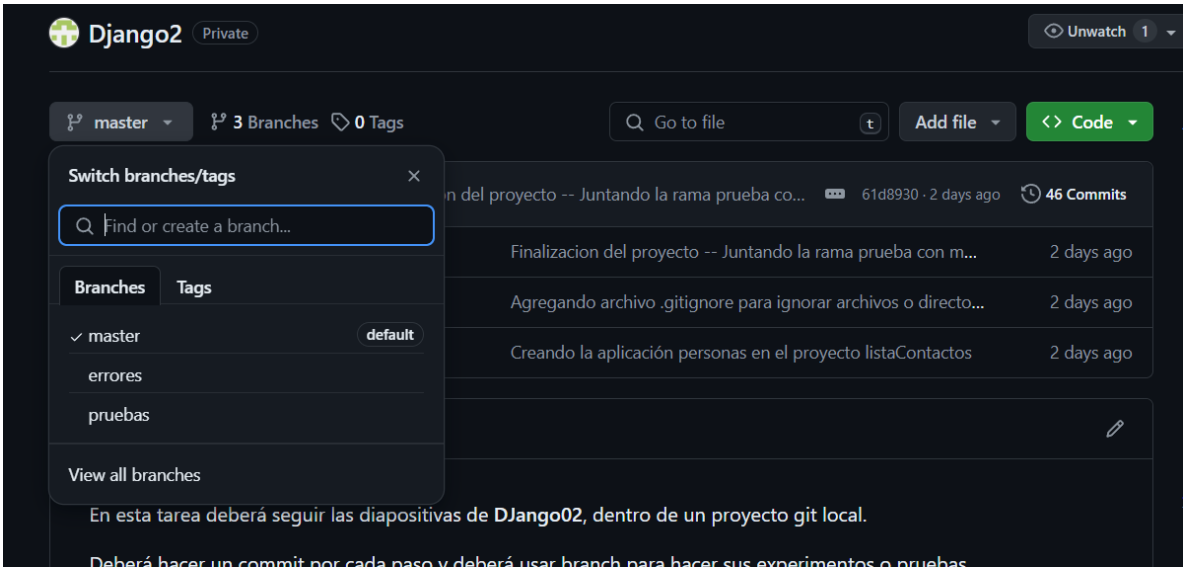

Figura 14: Ramas

11.6.6. Repositorio

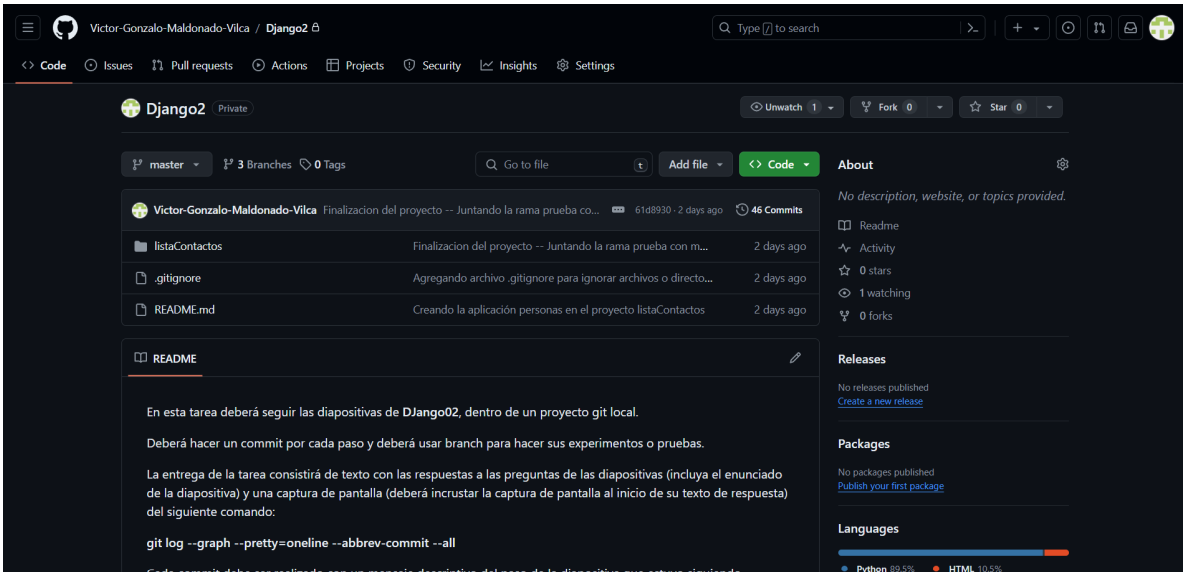

Figura 15: Repositorio

11.6.7. Proyecto compartido con el profesor de github

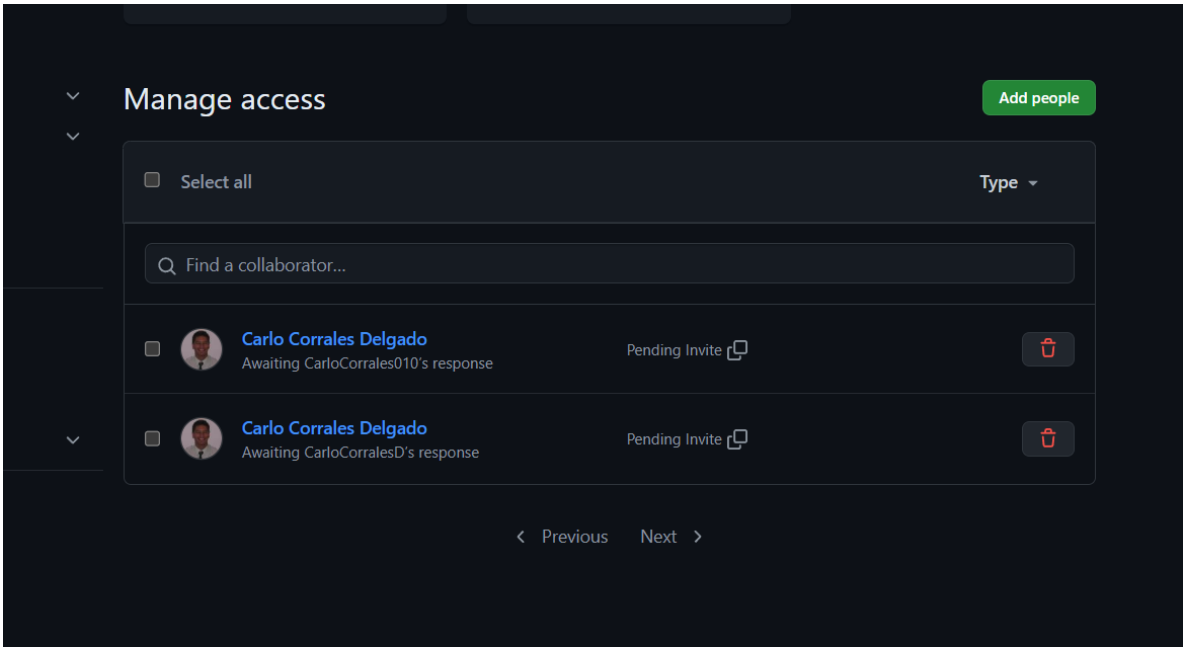

Figura 16: Compartir con el Docente

12. Recomensaciones

- Seguir las mejores prácticas de desarrollo de Django, como la separación de la lógica de negocios en vistas y la creación de plantillas reutilizables.
- Realizar pruebas unitarias y de integración para garantizar la calidad del código, y asegúrate de implementar medidas de seguridad en tu aplicación.

13. Conclusiones

- 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.

13.1. 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	

14. Referencias

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