Informe de Laboratorio 06

Tema: Django

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 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 9 de abril de 2024	Al 29 de mayo de 2024

1. Introducción

Django es un framework de desarrollo web en Python que facilita la creación rápida de aplicaciones web seguras y escalables. Ofrece un conjunto de herramientas integradas como ORM, administrador de Django y enrutamiento de URLs, lo que simplifica el desarrollo y mejora la seguridad de las aplicaciones web.

2. Objetivos

- Entender y configurar el entorno de trabajo para Django
- Crear un primer proyecto en Django
- Crear apps e integrarlas al framework

3. Tarea

Implementa un Sistema en Django que maneje una tabla de Alumnos, una de Cursos y una de NotasAlumnosPorCurso y que permita ingresar a nuevos alumnos, nuevos cursos y finalmente permita ingresar las notas por curso.

4. Entregables

- Informe en Latex
- URL: Repositorio GitHub
- Link del vÍdeo (Youtube)

5. Equipos, materiales y temas utilizados

- Python
- Pip
- Django
- Entornos virtuales en python
- Proyectos de Django
- Aplicaciones en Django
- Modelos, Vistas, Templates y Formularios en Django

6. URL de Repositorio Github

- URL del Repositorio GitHub
- `https://github.com/Victor-Gonzalo-Maldonado-Vilca/Lab06_Django.git`

7. Link de Video

- Link del Vídeo Explicativo
- `https://www.youtube.com/watch?v=unhmxDkoSDY`

8. Metodología

8.1. Creación del entorno de Trabajo

8.1.1. Carpeta de trabajo

Listing 1: Creación del Directorio

```
mkdir lab06 && cd lab06
```

8.1.2. Creación del Entorno Virtual

Listing 2: Creación Entorno virtual

```
virtualenv -p python3 .
```

8.1.3. Activar entorno Virtual

Listing 3: Activar Entorno Virtual

```
Scripts/activate
```

8.1.4. Instalar Django en el entorno Virtual

Listing 4: Instalar Django

```
pip install Django
```

8.1.5. Creación carpeta del proyecto

Listing 5: Carpeta src

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

8.1.6. Inicializar git

Listing 6: Inicializar git

```
git init
```

8.1.7. Crear .gitignore

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

8.2. Creacion del proyecto y apps

8.2.1. Crear Proyecto

Listing 7: Crear proyecto

```
django-admin startproject mi_proyecto
```

8.2.2. Crear App

Listing 8: Crear App

```
python manage.py startapp gestion_Universitario
```

9. Desarrollo del trabajo

9.1. Configuramos el proyecto en settings.py

9.1.1. Cambiamos el idioma y zona del tiempo

```
107  LANGUAGE_CODE = 'es'
108
109  TIME_ZONE = 'America/Lima'
110
111  USE_I18N = True
112
113  USE_TZ = True
```

Figura 1: Configuración a nuestra región

9.1.2. Agregamos la aplicación en el archivo settings.py

```
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'gestion_Universitaria',
41  ]
```

Figura 2: Agregar App

9.2. Definir Modelos en models.py en la App gestionUniversitaria

9.2.1. Modelo de Alumno

Esta parte del código define un modelo de Django llamado Alumno con campos como CUI, nombre, apellidos, edad y DNI, cada uno con validaciones específicas para garantizar la integridad de los datos. El método str devuelve una representación legible del objeto Alumno, concatenando el nombre y apellidos del alumno.

Listing 9: Modelo Alumno

```
class Alumno(models.Model):
  cui = models.IntegerField(
    validators=[
      MinValueValidator(10000000, message="Se requiere 8 digitos"),
      MaxValueValidator(99999999, message="Se requiere 8 digitos")
    ],
    unique=True
  )
  nombre = models.CharField(max_length=100)
  apellidos = models.CharField(max_length=100)
```

```
  edad = models.IntegerField(validators=[MinValueValidator(0), MaxValueValidator(100)])
  dni = models.IntegerField(
    validators=[
      MinValueValidator(10000000, message="Se requiere 8 digitos"),
      MaxValueValidator(99999999, message="Se requiere 8 digitos")
    ]
  )

  def __str__(self):
    return f"{self.nombre} {self.apellidos}"
```

9.2.2. Modelo de Curso

Esta parte del código está diseñado para manejar información sobre cursos universitarios, incluyendo su identificación única, nombre y cualquier otra información relacionada que pueda añadirse en futuras modificaciones del modelo.

Listing 10: Modelo Curso

```
class Curso(models.Model):
  codigo = models.IntegerField(
    validators=[
      MinValueValidator(1000000, message="Se requiere 7 digitos"),
      MaxValueValidator(9999999, message="Se requiere 7 digitos")
    ]
  )
  nombre = models.CharField(max_length=100)

  def __str__(self):
    return self.nombre
```

9.2.3. Modelo de NotasAlumnosPorCurso

El modelo NotasAlumnosPorCurso en Django almacena las notas de los alumnos por curso, relacionando cada registro con un alumno y un curso específicos. Utiliza un campo decimal para representar las notas con precisión de 2 decimales.

Listing 11: Modelo NotasAlumnosPorCurso

```
class NotasAlumnosPorCurso(models.Model):
  alumno = models.ForeignKey(Alumno, on_delete=models.CASCADE)
  curso = models.ForeignKey(Curso, on_delete=models.CASCADE)
  notas = models.DecimalField(max_digits=5, decimal_places=2)

  def __str__(self):
    return f"La nota del Alumno: {self.alumno.nombre} {self.alumno.apellidos}, es =
        {self.notas}"
```

9.3. Crear y Aplicar Migraciones

Listing 12: Crear Migraciones

```
python manage.py makemigrations gestion_Universitaria
```

Listing 13: Aplicar Migraciones

```
python manage.py migrate
```

9.4. Crear Formularios

Nos encontramos en forms.py de la aplicación gestionUniversitaria

9.4.1. Formulario AlumnoForm

Formulario asociado al modelo Alumno, permite ingresar información sobre un alumno como CUI, nombre, apellidos, edad y DNI.

Listing 14: Formulario AlumnoForm

```
class AlumnoForm(forms.ModelForm):

  class Meta:
    model = Alumno
    fields = ['cui', 'nombre', 'apellidos', 'edad', 'dni']
```

9.4.2. Formulario CursoForm

Formulario asociado al modelo Curso, utilizado para ingresar información sobre un curso como código y nombre.

Listing 15: Formulario CursoForm

```
class CursoForm(forms.ModelForm):

  class Meta:
    model = Curso
    fields = ['codigo', 'nombre']
```

9.4.3. Formulario NotasAlumnosPorCursoForm

Formulario vinculado al modelo NotasAlumnosPorCurso, permite registrar las notas de un alumno en un curso específico. Incluye campos para seleccionar el alumno, el curso y la nota.

Listing 16: Formulario NotasAlumnosPorCursoForm

```
class NotasAlumnosPorCursoForm(forms.ModelForm):
  alumno = forms.ModelChoiceField(queryset=Alumno.objects.all(), label='Alumno')
  curso = forms.ModelChoiceField(queryset=Curso.objects.all(), label='Curso')

  class Meta:
    model = NotasAlumnosPorCurso
    fields = ['alumno', 'curso', 'notas']
```

9.5. Crear Vistas

Nos encontramos en views.py de la aplicación gestionUniversitaria

9.5.1. Vista agregarAlumno

Esta vista maneja la creación de un nuevo alumno. Si el método de la solicitud es POST, crea un formulario AlumnoForm a partir de los datos de la solicitud y, si es válido, guarda el formulario y redirige a la misma vista para mostrar el formulario vacío. Si el método de la solicitud no es POST, simplemente muestra un formulario vacío para agregar un alumno y una lista de todos los alumnos existentes.

Listing 17: Vista agregarAlumno

```
def agregar_alumno(request):
  if request.method == 'POST':
    form = AlumnoForm(request.POST)
    if form.is_valid():
      form.save()
      return redirect('agregar_alumno')
  else:
    form = AlumnoForm()

  alumnos = Alumno.objects.all()
  return render(request, 'gestion_Universitaria/agregar_alumno.html', {'form': form,
      'alumnos': alumnos})
```

9.5.2. Vista agregarCurso

Esta vista gestiona la creación de un nuevo curso. Similar a la vista agregarAlumno, crea un formulario CursoForm a partir de los datos de la solicitud y, si es válido, guarda el formulario y redirige a la misma vista para mostrar el formulario vacío. Si el método de la solicitud no es POST, simplemente muestra un formulario vacío para agregar un curso y una lista de todos los cursos existentes.

Listing 18: Vista agregarCurso

```
def agregar_curso(request):
  if request.method == 'POST':
    form = CursoForm(request.POST)
    if form.is_valid():
      form.save()
      return redirect('agregar_curso')
  else:
    form = CursoForm()

  cursos = Curso.objects.all()
  return render(request, 'gestion_Universitaria/agregar_curso.html', {'form': form,
      'cursos': cursos})
```

9.5.3. Vista agregarNotasAlumnosPorCurso

Esta vista maneja la creación de nuevas notas para los alumnos en un curso específico. Al igual que las vistas anteriores, crea un formulario NotasAlumnosPorCursoForm a partir de los datos de la solicitud y, si es válido, guarda el formulario y redirige a la misma vista para mostrar el formulario

vacío. Si el método de la solicitud no es POST, simplemente muestra un formulario vacío para agregar notas y una lista de todas las notas existentes.

Listing 19: Vista agregarNotasAlumnosPorCurso

```
def agregar_notas_alumnos_por_curso(request):
  if request.method == 'POST':
    form = NotasAlumnosPorCursoForm(request.POST)
    if form.is_valid():
      form.save()
      return redirect('agregar_nota')
  else:
    form = NotasAlumnosPorCursoForm()

  notas = NotasAlumnosPorCurso.objects.all()
  return render(request, 'gestion_Universitaria/agregar_nota.html', {'form': form, 'notas':
      notas})
```

9.6. Crear Plantillas

En esta parte se verán los HTML, no nos interesará fragmentos de código propios de este, sino las novedades de Django como bucles para iterar todos los datos como los formularios para ingresar datos.

9.6.1. Archivo HTML agregarAlumnos

- **Descripción:** Este archivo HTML representa una página web enfocada en la administración de alumnos universitarios. Está dividida en dos secciones clave:
  En la primera sección, se presenta un formulario que permite añadir nuevos alumnos. Este formulario cuenta con medidas de seguridad, como el token CSRF, y un botón "Guardar"para enviar la información al servidor. También incluye enlaces a otras áreas del sistema, como la lista de alumnos, cursos y las notas de los alumnos.
  La segunda parte de la página muestra una tabla que contiene los detalles de los alumnos ya registrados en la base de datos. Si no hay alumnos registrados, se muestra un mensaje indicando esta ausencia. La tabla presenta información como el CUI, nombres, apellidos, edad y DNI de cada alumno.

- **Código:**

Listing 20: HTML agregarAlumnos

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Agregar Alumnos</title>
    <meta charset="UTF-8"/>
    <meta name="author" content="Victor Gonzalo Maldonado Vilca"/>
    <meta name="description" content="Agregar alumnos universitarios"/>
    <meta name="keywords" content="alumnos, agregar, universidad"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  </head>
  <body>
    <div>
      <h1>Agregar Alumnos</h1>
      <form method="POST">
        {% csrf_token %}
```

```
          {{ form.as_p }}
          <button type="submit">Guardar</button>
        </form>
        <a href="#">Lista de Alumnos</a>
        <a href="{% url 'agregar_curso' %}">Lista de Cursos</a>
        <a href="{% url 'agregar_nota' %}">Notas de los Alumnos</a>
      </div>
      <div>
        <h1>Lista de Alumnos</h1>
        {% if alumnos %}
          <table border="1" style="text-align: center">
            <tr>
              <th>CUI</th>
              <th>Nombre</th>
              <th>Apellidos</th>
              <th>Edad</th>
              <th>DNI</th>
            </tr>
            {% for alumno in alumnos %}
              <tr>
                <td>{{ alumno.cui }}</td>
                <td>{{ alumno.nombre }}</td>
                <td>{{ alumno.apellidos }}</td>
                <td>{{ alumno.edad }}</td>
                <td>{{ alumno.dni }}</td>
              </tr>
            {% endfor %}

          </table>
        {% else %}
          <p>No hay alumnos en la base de datos.</p>
        {% endif %}
      </div>
    </body>
  </html>
```

9.6.2. Archivo HTML agregarCursos

- **Descripción:** Este código HTML representa una página web para la gestión de cursos universitarios. Incluye un formulario para agregar cursos, con un campo para ingresar información sobre nuevos cursos, un botón Guardar para enviar el formulario al servidor y enlaces a otras secciones del sistema. También muestra una lista de cursos existentes en una tabla, mostrando el código y el nombre de cada curso si hay cursos registrados, o un mensaje indicando la ausencia de cursos en caso contrario.
- **Código:**

Listing 21: HTML agregarCursos

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Agregar Cursos</title>
    <meta charset="UTF-8"/>
    <meta name="author" content="Victor Gonzalo Maldonado Vilca"/>
    <meta name="description" content="Agregar cursos universitarios"/>
```

```
    <meta name="keywords" content="cursos, agregar, universidad"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  </head>
  <body>
    <div>
      <h1>Agregar Cursos</h1>
      <form method="POST">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Guardar</button>
      </form>
      <a href="{% url 'agregar_alumno' %}">Lista de Alumnos</a>
      <a href="#">Lista de Cursos</a>
      <a href="{% url 'agregar_nota' %}">Notas de los Alumnos</a>
    </div>
    <div>
      <h1>Lista de Cursos</h1>
      {% if cursos %}
        <table border="1" style="text-align: center">
          <tr>
            <th>Codigo</th>
            <th>Nombre del Curso</th>
          </tr>
          {% for curso in cursos %}
            <tr>
              <td>{{ curso.codigo }}</td>
              <td>{{ curso.nombre }}</td>
            </tr>
          {% endfor %}

        </table>
      {% else %}
        <p>No hay cursos en la base de datos.</p>
      {% endif %}
    </div>
  </body>
</html>
```

9.6.3. Archivo HTML agregarNotas

- **Descripción:** Este código HTML crea una página web para agregar y mostrar notas de alumnos en cursos universitarios. Incluye un formulario para ingresar nuevas notas, enlaces a la lista de alumnos y cursos, y una tabla para mostrar las notas existentes. La tabla presenta información como el nombre del alumno, el curso y las notas obtenidas. Si no hay notas registradas, se muestra un mensaje indicando esta situación.
- **Código:**

Listing 22: HTML agregarNotas

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Agregar Notas</title>
    <meta charset="UTF-8"/>
    <meta name="author" content="Victor Gonzalo Maldonado Vilca"/>
```

```
    <meta name="description" content="Agregar notas"/>
    <meta name="keywords" content="notas, agregar, universidad"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  </head>
  <body>
    <div>
      <h1>Agregar Notas</h1>
      <form method="POST">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Guardar</button>
      </form>
      <a href="{% url 'agregar_alumno' %}">Lista de Alumnos</a>
      <a href="{% url 'agregar_curso' %}">Lista de Cursos</a>
      <a href="#">Notas de los Alumnos</a>
    </div>
    <div>
      <h1>Lista de Alumnos - curso - nota</h1>
      {% if notas %}
        <table border="1" style="text-align: center">
          <tr>
            <th>Alumno</th>
            <th>Curso</th>
            <th>Notas</th>
          </tr>
          {% for nota in notas %}
            <tr>
              <td>{{ nota.alumno}}</td>
              <td>{{ nota.curso }}</td>
              <td>{{ nota.notas }}</td>
            </tr>
          {% endfor %}

        </table>
      {% else %}
        <p>No hay notas en la base de datos.</p>
      {% endif %}
    </div>
  </body>
</html>
```

9.6.4. CSS

Los estilos usados son básicos simples cambios de color al igual que alineación de palabras, las clases y posiblemente id simplemente, se colocaron por buena práctiva.

9.7. Configurar URLS

9.7.1. Configurar las URLs en urls.py de la aplicación gestionUniversitaria

- **Descripción:** El archivo de configuración de urls en Django está diseñado para asignar rutas (URLs) a tres vistas diferentes en una aplicación web. La primera ruta está vinculada a la vista agregarAlumno, la segunda a la vista agregarCurso, y la tercera a la vista agregarNotasAlumnosPorCurso. Cuando un usuario accede a las URLs /alumnos/, /cursos/, o /notas/, Django activa la función correspondiente para manejar la solicitud. Los nombres asignados (name) a cada ruta

facilitan su referencia en otras partes del código, como enlaces (links) dentro de los templates HTML.

- **Código:**

Listing 23: urls.py de la App

```
from django.urls import path
from . import views

urlpatterns = [
  path('alumnos/', views.agregar_alumno, name='agregar_alumno'),
  path('cursos/', views.agregar_curso, name='agregar_curso'),
  path('notas/', views.agregar_notas_alumnos_por_curso, name='agregar_nota'),
]
```

9.7.2. Configurar las URLs en urls.py del proyecto

- **Descripción:** En el fragmento de código Python del archivo de configuración de URLs (urls.py) del proyecto Django, se importan las funciones necesarias para definir rutas de URL (path) y para incluir otro archivo de configuración de URLs (include). Posteriormente, se establecen las URLs principales del proyecto: la URL /admin/ se asigna a la interfaz de administración de Django, mientras que la URL raíz (") se asigna a las URLs definidas en el archivo urls.py de la aplicación gestionUniversitaria. Esta configuración implica que todas las URLs que comiencen con la raíz del sitio, como /, /alumnos/, /cursos/, etc., serán gestionadas por las URLs definidas en gestionUniversitaria.urls.

- **Código:**

Listing 24: urls.py del proyecto

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
  path('admin/', admin.site.urls),
  path('', include('gestion_Universitaria.urls')),
]
```

9.7.3. Ejecutar el Servidor

Listing 25: Servidor

```
python manage.py runserver
```

9.7.4. Ejecutando la Aplicación

- Agregar Alumnos:

Figura 3: Agregar Alumnos

Figura 4: Agregar Alumnos

- Agregar Cursos:

Figura 5: Agregar Cursos

Figura 6: Agregar Cursos

- Agregar Notas:

Figura 7: Agregar Notas

Figura 8: Agregar Notas

9.8. Uso de GitHub

9.8.1. Usuario de GitHub

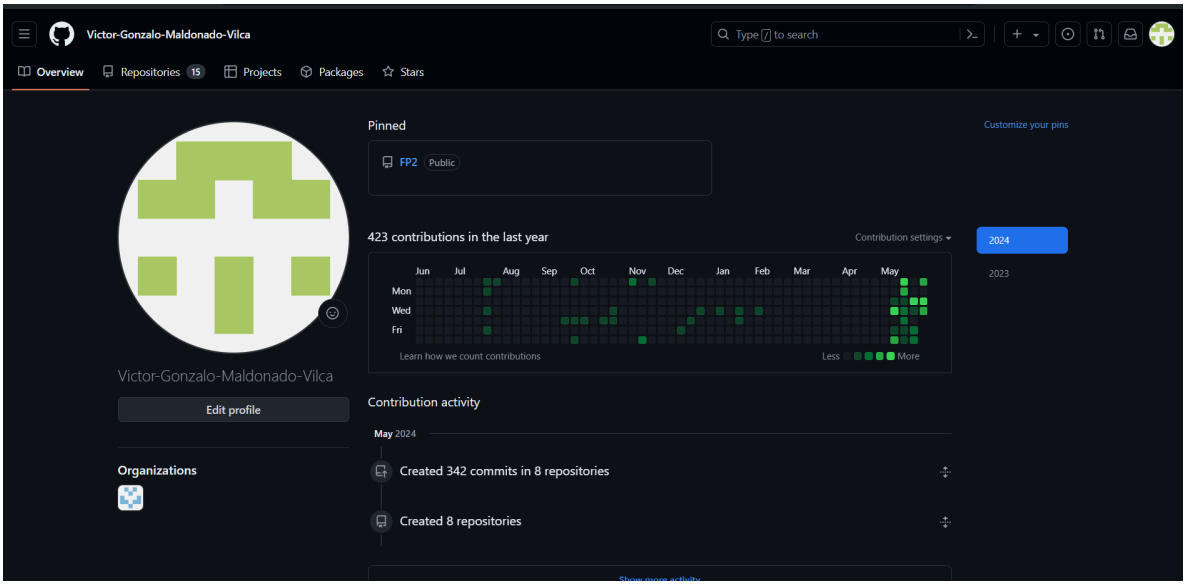

Figura 9: Usuario GitHub

9.8.2. Creación de un Nuevo Repositorio

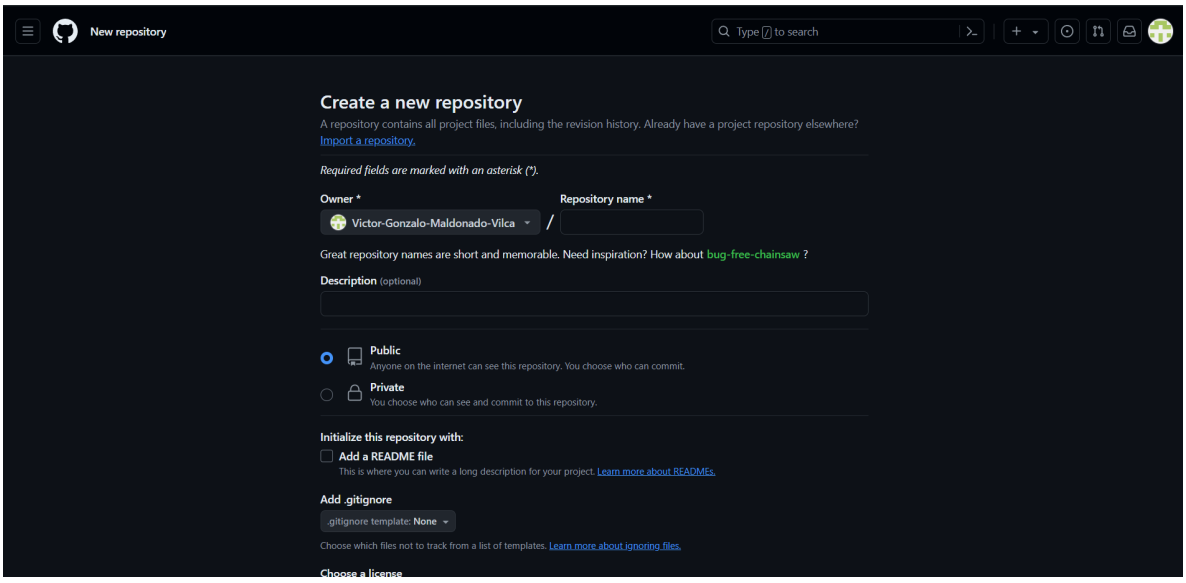

Figura 10: Creación Repositorio

9.8.3. Comandos de Configuración

Listing 26: Servidor

```
echo "# lab06" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin https://github.com/Victor-Gonzalo-Maldonado-Vilca/Lab06_Django.git
git push -u origin master
```

9.8.4. Implementación de Readme.md

Figura 11: Readme.md

9.8.5. Registro de cambios en mi código

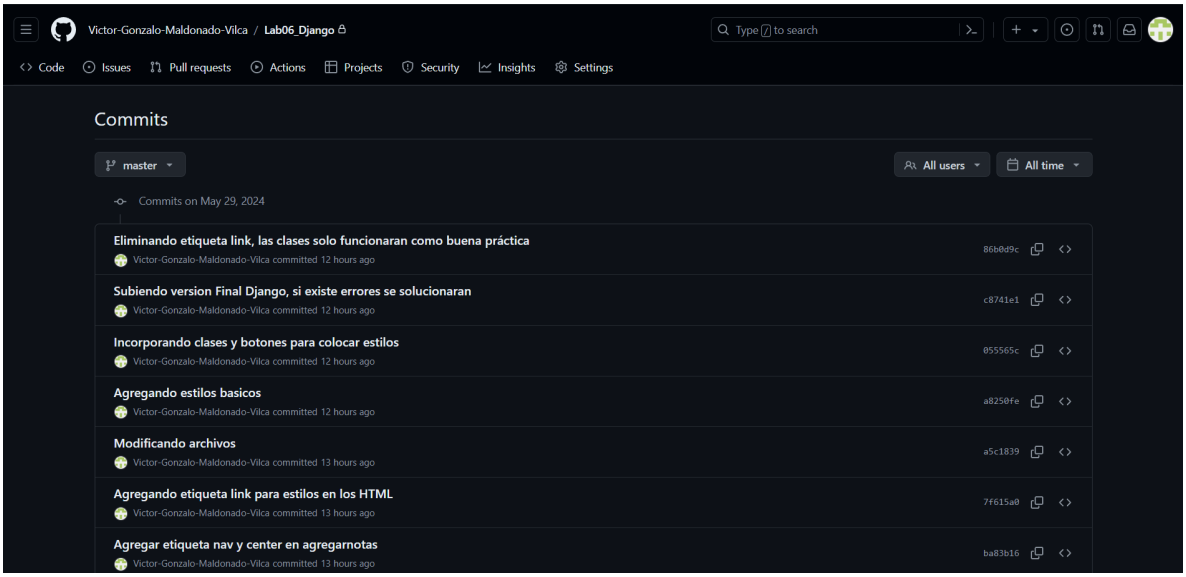

Figura 12: Commits

9.8.6. Repositorio

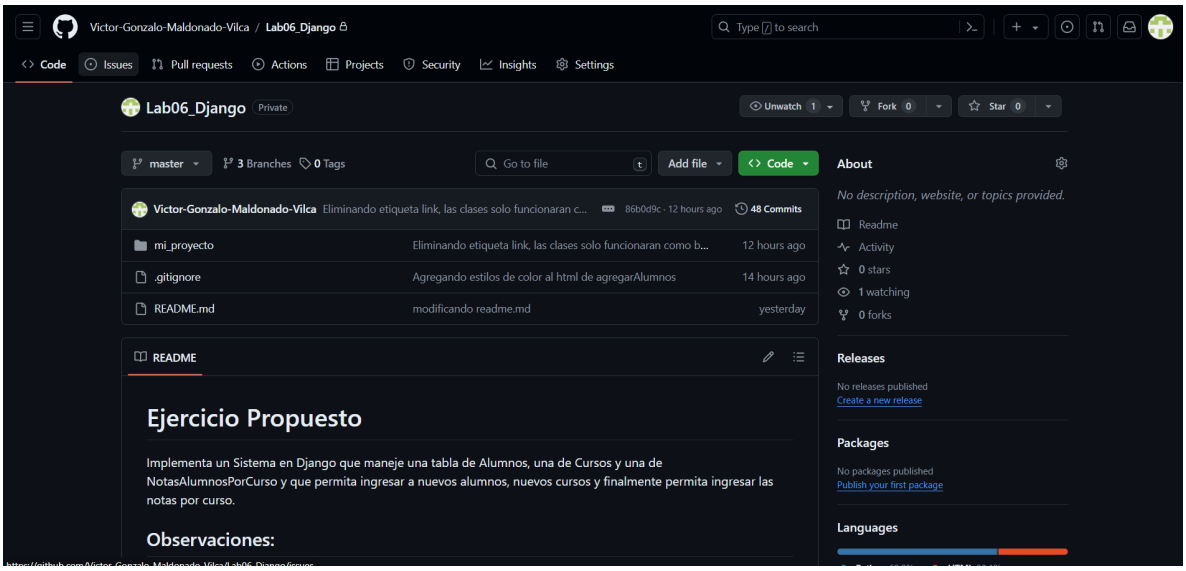

Figura 13: Repositorio

9.8.7. Uso de Ramas

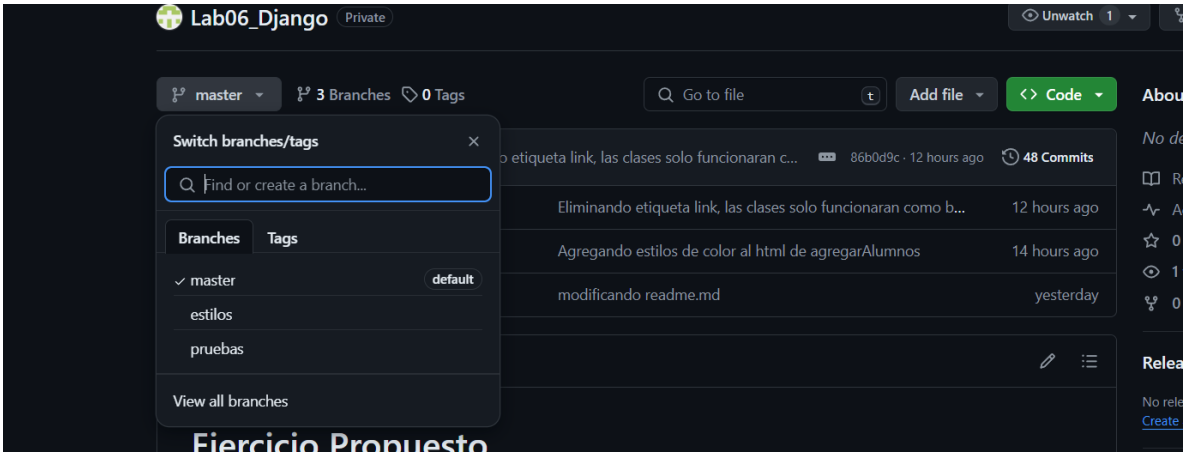

Figura 14: Ramas

9.8.8. Proyecto compartido con el profesor de github

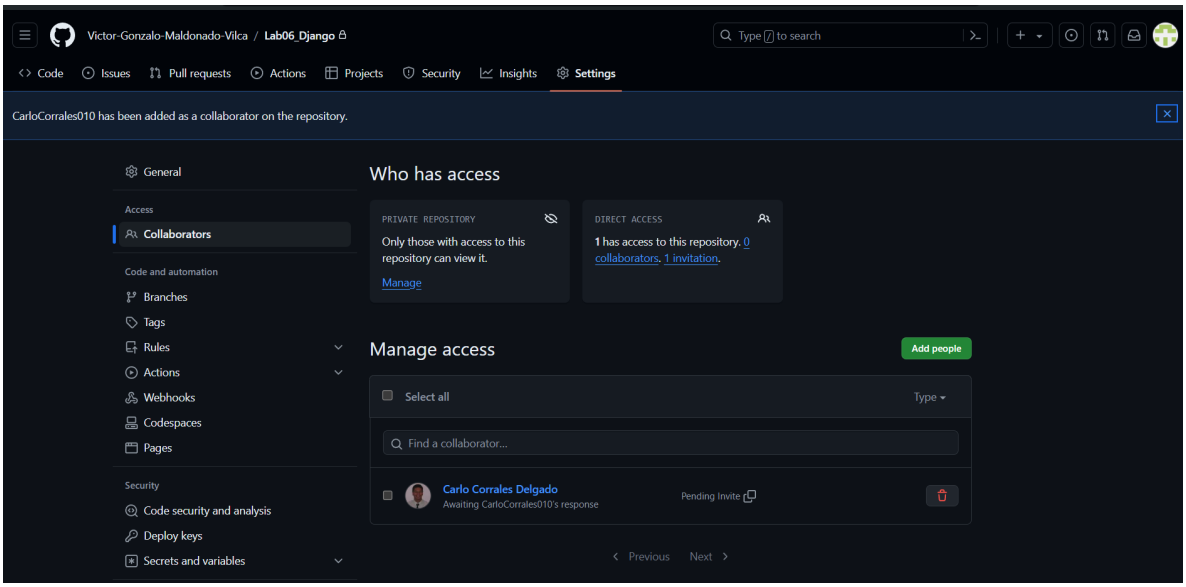

Figura 15: Compartir con el Docente

10. 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.

11. Conclusiones

- 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.

11.1. 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	

12. Referencias

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