Informe de Laboratorio 06

Tema: Django - Usando una plantilla para ver Destinos Turísticos

Nota

Estudiante	Escuela	Asignatura
Fernando Miguel Garambel Marín fgarambel@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Laboratorio de Programación Web 2 Semestre: III Código: 1701212

Laboratorio	Tema	Duración
06	Django - Usando una plantilla para ver Destinos Turísticos	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 24 de mayo 2024	Al 5 de junio 2024

1. Objetivos

- Implementar una aplicación en Django utilizando una plantilla profesional.
- Utilizar una tabla de Destinos turísticos para leer y completar la página web
- Utilizar los tags “if” y “for” en los archivos html para leer todos los registros de una tabla desde una base de datos.

2. Actividades

- Crear un proyecto en Django
- Siga los pasos del video para poder implementar la aplicación de Destinos turísticos
- Use git y haga los commits necesarios para manejar correctamente la aplicación.

3. Ejercicio Propuestos

- Deberán replicar la actividad del video que se encuentra en el AV de Teoría (Django Tutorial for Beginners - Telusko(`https://youtu.be/OTmQOjsl0eg`)donde se obtiene una plantilla de una aplicación de Destinos turísticos y adecuarla a un proyecto en blanco Django.

- Luego trabajar con un modelo de tabla DestinosTuristicos donde se guarden nombreCiudad, descripcionCiudad, imagenCiudad, precioTour, ofertaTour (booleano). Estos destinos turísticos deberán ser agregados en una vista dinámica utilizando tags for e if.
- Para ello crear una carpeta dentro del proyecto github colaborativo con el docente, e informar el link donde se encuentra.
- Crear formularios de Añadir Destinos Turísticos, Modificar, Listar y Eliminar Destinos.

4. Equipos, materiales y temas utilizados

- Sistema operativo de 64 bits, procesador basado en x64.
- Latex.
- git version 2.41.0.windows.1
- Cuenta en GitHub con el correo institucional.

5. URL Github, Video

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/FernandoGarambelM/Destinos_turisticos.git`
- URL para el video flipgrid
- `Video`

6. Replicando la actividad del video

- Primero ingresamos al entorno virtual

Listing 1: Activar el ambiente virtual

```
env\Scripts\activate
```

Listing 2: Clonar el repositorio

```
git clone https://github.com/navinreddy20/django-telusko-codes
```

- Luego instalamos postgresql y pgadmin
- Luego instalamos al entorno virtual psycopg2-binary y Pillow

Listing 3: Instalando psycopg2-binary y Pillow

```
pip install psycopg2-binary
pip install Pillow
```

- Despues de instalar, creamos un superusuario con el siguiente script

Listing 4: Script para crear un superusuario

```
#!/bin/sh

# Variables de entorno para el superusuario
DJANGO_SUPERUSER_USERNAME=admin
DJANGO_SUPERUSER_EMAIL=admin@example.com
DJANGO_SUPERUSER_PASSWORD=123456

# Ejecutar el comando createsuperuser sin interaccion
python manage.py shell -c "from django.contrib.auth import get_user_model; User =
    get_user_model(); User.objects.create_superuser('$DJANGO_SUPERUSER_USERNAME',
    '$DJANGO_SUPERUSER_EMAIL', '$DJANGO_SUPERUSER_PASSWORD') if not
    User.objects.filter(username='$DJANGO_SUPERUSER_USERNAME').exists() else
    print('Superusuario ya existe.')"
```

7. Instalar Django en el Ambiente Virtual

Listing 5: Instalar Django

```
pip install django
```

8. Crear un Directorio e Inicializar un Repositorio Git

Listing 6: Crear un directorio para el proyecto

```
mkdir mi_proyecto
cd mi_proyecto
```

Listing 7: Inicializar un repositorio Git

```
git init
```

Listing 8: Crear un proyecto Django

```
django-admin startproject mi_proyecto
```

9. Crear un Proyecto en GitHub y Enlazarlo

- Primero se crea un nuevo repositorio en GitHub

Listing 9: Enlazar el repositorio local con GitHub

```
git remote add origin https://github.com/Choflis/labDjango.git
```

Listing 10: Agregar archivos, hacer un commit inicial y hacer push al repositorio remoto

```
git add .
git commit -m "Initial commit"
git push -u origin master
```

Listing 11: Creando un archivo .gitignore según el repositorio dado

```
# If you need to exclude files such as those generated by an IDE, use
# $GIT_DIR/info/exclude or the core.excludesFile configuration variable as
# described in https://git-scm.com/docs/gitignore

*.egg-info
*.pot
*.py[co]
.tox/
__pycache__
MANIFEST
dist/
docs/_build/
docs/locale/
node_modules/
tests/coverage_html/
tests/.coverage*
build/
tests/report/
tests/screenshots/
```

10. Crear un proyecto en Django que maneje una tabla de Productos y una tabla de Ventas

4. **Agregar la aplicación al archivo `settings.py` del proyecto:**

```
INSTALLED_APPS = [
    ...
    'tienda',
]
```

Paso 2: Definir los Modelos

1. **Definir el modelo `Producto` y `Venta` en `models.py` de la aplicación `tienda`:**

```
from django.db import models

class Producto(models.Model):
        codigo = models.CharField(max_length=100, unique=True)
        nombre = models.CharField(max_length=200)
        descripcion = models.TextField()
        precio = models.DecimalField(max_digits=10, decimal_places=2)

        def __str__(self):
        return self.nombre

class Venta(models.Model):
        producto = models.ForeignKey(Producto, on_delete=models.CASCADE)
        cantidad = models.PositiveIntegerField()
        fecha = models.DateTimeField(auto_now_add=True)
```

```
        def __str__(self):
        return f"Venta de {self.cantidad} unidades de {self.producto.nombre}"
```

2. **Crear y aplicar las migraciones:**

```
python manage.py makemigrations tienda
python manage.py migrate
```

Paso 3: Crear Formularios y Vistas

1. **Crear un formulario para ingresar ventas en `forms.py` de la aplicación `tienda`:**

```
from django import forms
from .models import Venta, Producto

class VentaForm(forms.ModelForm):
    codigo_producto = forms.CharField(max_length=100, label="Código del Producto")

    class Meta:
    model = Venta
    fields = ['codigo_producto', 'cantidad']

    def clean_codigo_producto(self):
    codigo = self.cleaned_data.get('codigo_producto')
    if not Producto.objects.filter(codigo=codigo).exists():
            raise forms.ValidationError("El producto con este código no existe.")
    return codigo

    def save(self, commit=True):
    codigo = self.cleaned_data.get('codigo_producto')
    producto = Producto.objects.get(codigo=codigo)
    venta = super().save(commit=False)
    venta.producto = producto
    if commit:
            venta.save()
    return venta
```

2. **Crear vistas para manejar el formulario de ventas en `views.py` de la aplicación `tienda`:**

```
from django.shortcuts import render, redirect
from .forms import VentaForm
```

```python
def crear_venta(request):
    if request.method == 'POST':
        form = VentaForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('lista_ventas')
    else:
        form = VentaForm()
    return render(request, 'tienda/crear_venta.html', {'form': form})

def lista_ventas(request):
    ventas = Venta.objects.all()
    return render(request, 'tienda/lista_ventas.html', {'ventas': ventas})
```

Paso 4: Crear las Plantillas

1. **Crear una plantilla** `crear_venta.html` **en el directorio** `templates/tienda/`**:**

```
<!DOCTYPE html>
<html>
<head>
    <title>Crear Venta</title>
</head>
<body>
    <h1>Crear Venta</h1>
    <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Guardar</button>
    </form>
    <a href="{% url 'lista_ventas' %}">Ver todas las ventas</a>
```

```
</body>

</html>
```

2. **Crear una plantilla `lista_ventas.html` en el directorio `templates/tienda/`:**

```
<!DOCTYPE html>

<html>

<head>

    <title>Lista de Ventas</title>

</head>

<body>

    <h1>Lista de Ventas</h1>

    <ul>

    {% for venta in ventas %}

    <li>{{ venta.fecha }}: {{ venta.cantidad }} unidades de {{
venta.producto.nombre }}</li>

    {% endfor %}

    </ul>

    <a href="{% url 'crear_venta' %}">Crear nueva venta</a>

</body>

</html>
```

Paso 5: Configurar las URL

1. **Configurar las URLs en `urls.py` de la aplicación `tienda`:**

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


urlpatterns = [
    path('crear/', views.crear_venta, name='crear_venta'),
    path('', views.lista_ventas, name='lista_ventas'),
]
```

2. **Incluir las URLs de la aplicación `tienda` en el archivo `urls.py` del proyecto:**

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

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

Paso 6: Ejecutar el Servidor

1. **Ejecutar el servidor de desarrollo:**

```
python manage.py runserver
```

2. **Acceder a la aplicación en el navegador:**

   - Crear una venta en `http://127.0.0.1:8000/ventas/crear/`
   - Ver la lista de ventas en `http://127.0.0.1:8000/ventas/`

- Resultado de la ventana

Crear Venta

Código del Producto:

Cantidad:

Guardar

Ver todas las ventas 2. Crear una plantilla lista_ventas.html en el directorio templates/tienda/:

Lista de Ventas

Crear nueva venta

11. Ejercicio propuesto

- Primero creamos una aplicación Django

Listing 12: Crear una aplicación Django

```
python manage.py startapp gestion_alumnos
```

Listing 13: Agregar la aplicación al archivo settings.py del proyecto

```
INSTALLED_APPS = [
...
'gestion_alumnos',
]
```

- Luego definimos tres modelos en Django para una aplicación de gestión de alumnos, cursos y notas

Listing 14: Código de models.py

```
from django.db import models

class Alumno(models.Model):
    nombre = models.CharField(max_length=100)
    apellido = models.CharField(max_length=100)
    fecha_nacimiento = models.DateField()

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

class Curso(models.Model):
    nombre = models.CharField(max_length=100)
    descripcion = models.TextField()
```

```
14
15      def __str__(self):
16          return self.nombre
17
18  class NotasAlumnosPorCurso(models.Model):
19      alumno = models.ForeignKey(Alumno, on_delete=models.CASCADE)
20      curso = models.ForeignKey(Curso, on_delete=models.CASCADE)
21      nota = models.FloatField()
22
23      def __str__(self):
24          return f"{self.alumno} - {self.curso} - {self.nota}"
```

Listing 15: Crear y aplicar las migraciones

```
    python manage.py makemigrations gestion_alumnos
    python manage.py migratetien
```

- Luego creamos 3 formularios para cada modelo, cada formulario incluye todos los campos de su correspondiente modelo, permitiendo la creación y edición de instancias de estos modelos a través de formularios.

Listing 16: Código de forms.py

```
1  from django import forms
2  from .models import Alumno, Curso, NotasAlumnosPorCurso
3
4  class AlumnoForm(forms.ModelForm):
5      class Meta:
6          model = Alumno
7          fields = '__all__'
8
9  class CursoForm(forms.ModelForm):
10     class Meta:
11         model = Curso
12         fields = '__all__'
13
14 class NotasAlumnosPorCursoForm(forms.ModelForm):
15     class Meta:
16         model = NotasAlumnosPorCurso
17         fields = '__all__'
```

- Luego se definen las vistas para cada formulario que nos permiten:
- Mostrar una página con formularios para agregar alumnos, cursos y notas, y listas actuales de estos datos.
- Manejar las solicitudes POST para agregar nuevos alumnos, cursos y notas, validando y guardando los datos en la base de datos, y redirigiendo de vuelta a la página principal de gestión de datos después de cada operación.

Listing 17: Código de views.py

```
1  from django.shortcuts import render, redirect
```

```python
from .models import Alumno, Curso, NotasAlumnosPorCurso
from .forms import AlumnoForm, CursoForm, NotasAlumnosPorCursoForm

def gestionar_datos(request):
    alumnos = Alumno.objects.all()
    cursos = Curso.objects.all()
    notas = NotasAlumnosPorCurso.objects.all()

    context = {
        'alumno_form': AlumnoForm(prefix='alumno'),
        'curso_form': CursoForm(prefix='curso'),
        'nota_form': NotasAlumnosPorCursoForm(prefix='nota'),
        'alumnos': alumnos,
        'cursos': cursos,
        'notas': notas
    }

    return render(request, 'gestion_alumnos/gestionar_datos.html', context)
def agregar_alumno(request):
    if request.method == "POST":
        alumno_form = AlumnoForm(request.POST, prefix='alumno')
        if alumno_form.is_valid():
            alumno_form.save()
        return redirect('gestionar_datos')

def agregar_curso(request):
    if request.method == "POST":
        curso_form = CursoForm(request.POST, prefix='curso')
        if curso_form.is_valid():
            curso_form.save()
        return redirect('gestionar_datos')

def agregar_nota(request):
    if request.method == "POST":
        nota_form = NotasAlumnosPorCursoForm(request.POST, prefix='nota')
        if nota_form.is_valid():
            nota_form.save()
        return redirect('gestionar_datos')
```

- Luego agregamos las urls que se utilizaran para procesar los formularios

Listing 18: Código de urls.py de la aplicación

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

urlpatterns = [
    path('', views.gestionar_datos, name='gestionar_datos'),
    path('agregar_alumno/', views.agregar_alumno, name='agregar_alumno'),
    path('agregar_curso/', views.agregar_curso, name='agregar_curso'),
    path('agregar_nota/', views.agregar_nota, name='agregar_nota'),
]
```

Listing 19: Código de urls.py del proyecto

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

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

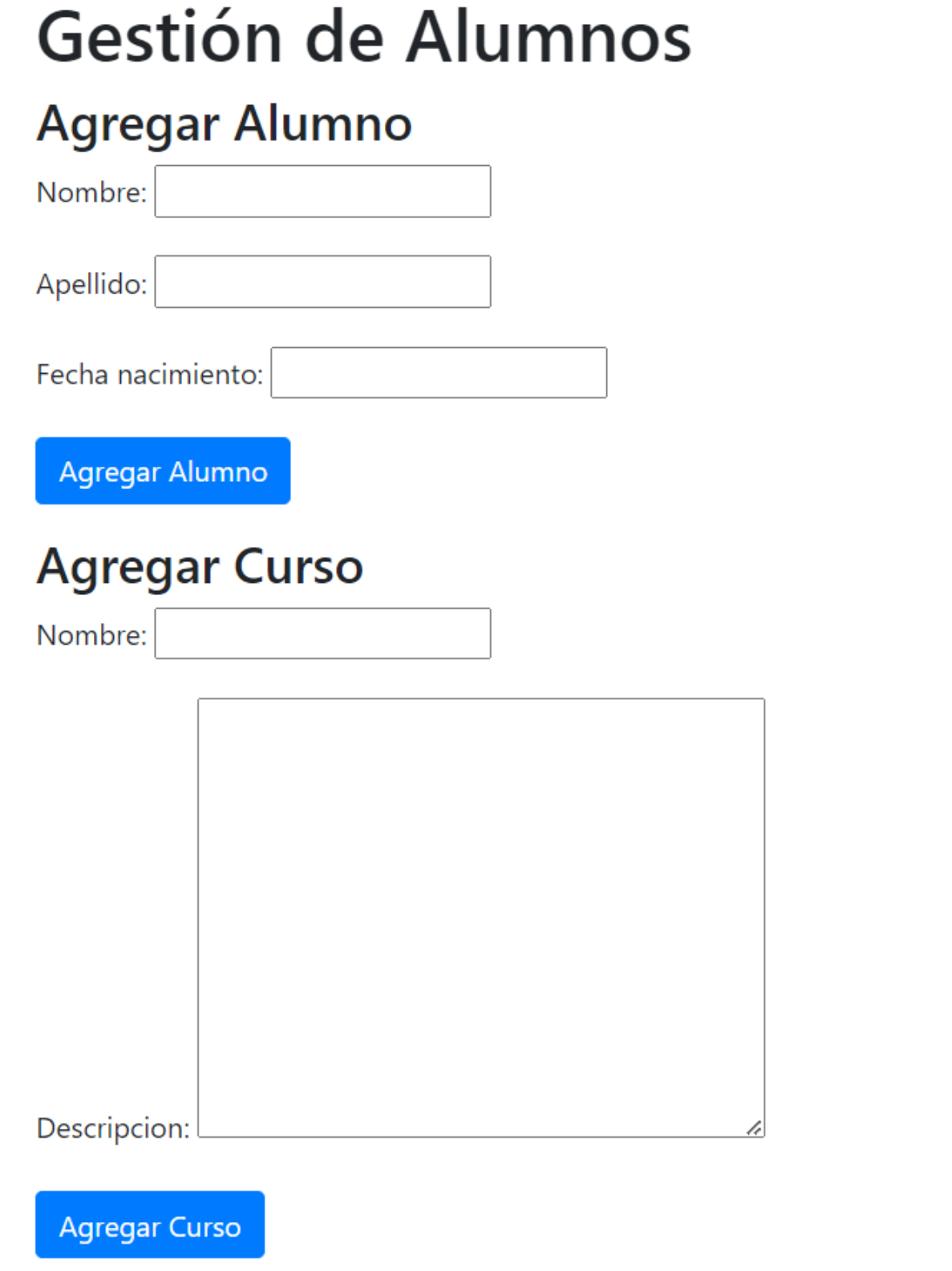
Listing 20: Código del html de la aplicación

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Gestion de Alumnos</title>
    <link rel="stylesheet"
        href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        body {
            padding: 20px;
        }
        .container {
            max-width: 800px;
            margin: 0 auto;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Gestion de Alumnos</h1>
        <div class="form-group">
            <h3>Agregar Alumno</h3>
            <form method="POST" action="{% url 'agregar_alumno' %}">
                {% csrf_token %}
                {{ alumno_form.as_p }}
                <button type="submit" class="btn btn-primary">Agregar Alumno</button>
            </form>
        </div>
        <div class="form-group">
            <h3>Agregar Curso</h3>
            <form method="POST" action="{% url 'agregar_curso' %}">
                {% csrf_token %}
                {{ curso_form.as_p }}
                <button type="submit" class="btn btn-primary">Agregar Curso</button>
            </form>
        </div>
        <div class="form-group">
            <h3>Agregar Nota</h3>
            <form method="POST" action="{% url 'agregar_nota' %}">
                {% csrf_token %}
                {{ nota_form.as_p }}
                <button type="submit" class="btn btn-primary">Agregar Nota</button>
            </form>
        </div>
```

```
        <hr>
        <h2>Alumnos</h2>
        <table class="table table-striped">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Nombre</th>
                    <th>Apellido</th>
                    <th>Fecha de Nacimiento</th>
                </tr>
            </thead>
            <tbody>
                {% for alumno in alumnos %}
                <tr>
                    <td>{{ alumno.id }}</td>
                    <td>{{ alumno.nombre }}</td>
                    <td>{{ alumno.apellido }}</td>
                    <td>{{ alumno.fecha_nacimiento }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        <h2>Cursos</h2>
        <table class="table table-striped">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Nombre</th>
                    <th>Descripcion</th>
                </tr>
            </thead>
            <tbody>
                {% for curso in cursos %}
                <tr>
                    <td>{{ curso.id }}</td>
                    <td>{{ curso.nombre }}</td>
                    <td>{{ curso.descripcion }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        <h2>Notas</h2>
        <table class="table table-striped">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Alumno</th>
                    <th>Curso</th>
                    <th>Nota</th>
                </tr>
            </thead>
            <tbody>
                {% for nota in notas %}
                <tr>
                    <td>{{ nota.id }}</td>
                    <td>{{ nota.alumno }}</td>
```

```
100                     <td>{{ nota.curso }}</td>
101                     <td>{{ nota.nota }}</td>
102                 </tr>
103                 {% endfor %}
104             </tbody>
105         </table>
106     </div>
107 </body>
108 </html>
```

- Como se ve en la página:


Gestión de Alumnos
Agregar Alumno
Nombre:
Apellido:
Fecha nacimiento:
Agregar Alumno
Agregar Curso
Nombre:
Descripcion:
Agregar Curso

12. Referencias

- `https://docs.djangoproject.com/es/3.2/`
- `https://docs.djangoproject.com/es/3.2/ref/models/fields/#field-types`