

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/OTmQQjs10eg>) 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
```

- Después 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.')"
```

- Realizamos migraciones para enlazar la base de datos

Listing 5: Código para realizar migraciones

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

- Después de seguir estos pasos la plantilla ya funciona por completo

7. Adecuar la plantilla a un proyecto en blanco de Django

Listing 6: Instalar Django

```
pip install django
```

8. Crear un Directorio e Inicializar un Repositorio Git

Listing 7: Crear un directorio para el proyecto

```
mkdir mi_proyecto  
cd mi_proyecto
```

Listing 8: Inicializar un repositorio Git

```
git init
```

Listing 9: 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 10: Enlazar el repositorio local con GitHub

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

Listing 11: 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 12: 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
```

```
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:

[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 13: Crear una aplicación Django

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

Listing 14: 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 15: Código de models.py

```
1 from django.db import models  
2  
3 class Alumno(models.Model):  
4     nombre = models.CharField(max_length=100)  
5     apellido = models.CharField(max_length=100)  
6     fecha_nacimiento = models.DateField()  
7  
8     def __str__(self):  
9         return f"{self.nombre} {self.apellido}"  
10  
11 class Curso(models.Model):  
12     nombre = models.CharField(max_length=100)  
13     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 16: 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 17: 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 18: Código de views.py

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

```
2 from .models import Alumno, Curso, NotasAlumnosPorCurso
3 from .forms import AlumnoForm, CursoForm, NotasAlumnosPorCursoForm
4
5 def gestionar_datos(request):
6     alumnos = Alumno.objects.all()
7     cursos = Curso.objects.all()
8     notas = NotasAlumnosPorCurso.objects.all()
9
10    context = {
11        'alumno_form': AlumnoForm(prefix='alumno'),
12        'curso_form': CursoForm(prefix='curso'),
13        'nota_form': NotasAlumnosPorCursoForm(prefix='nota'),
14        'alumnos': alumnos,
15        'cursos': cursos,
16        'notas': notas
17    }
18
19    return render(request, 'gestion_alumnos/gestionar_datos.html', context)
20 def agregar_alumno(request):
21     if request.method == "POST":
22         alumno_form = AlumnoForm(request.POST, prefix='alumno')
23         if alumno_form.is_valid():
24             alumno_form.save()
25             return redirect('gestionar_datos')
26
27 def agregar_curso(request):
28     if request.method == "POST":
29         curso_form = CursoForm(request.POST, prefix='curso')
30         if curso_form.is_valid():
31             curso_form.save()
32             return redirect('gestionar_datos')
33
34 def agregar_nota(request):
35     if request.method == "POST":
36         nota_form = NotasAlumnosPorCursoForm(request.POST, prefix='nota')
37         if nota_form.is_valid():
38             nota_form.save()
39             return redirect('gestionar_datos')
```

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

Listing 19: Código de urls.py de la aplicación

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.gestionar_datos, name='gestionar_datos'),
6     path('agregar_alumno/', views.agregar_alumno, name='agregar_alumno'),
7     path('agregar_curso/', views.agregar_curso, name='agregar_curso'),
8     path('agregar_nota/', views.agregar_nota, name='agregar_nota'),
9 ]
```

Listing 20: 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 21: Código del html de la aplicación

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Gestion de Alumnos</title>
6     <link rel="stylesheet"
7         href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
8     <style>
9         body {
10             padding: 20px;
11         }
12         .container {
13             max-width: 800px;
14             margin: 0 auto;
15         }
16     </style>
17 </head>
18 <body>
19     <div class="container">
20         <h1>Gestion de Alumnos</h1>
21         <div class="form-group">
22             <h3>Agregar Alumno</h3>
23             <form method="POST" action="{% url 'agregar_alumno' %}">
24                 {% csrf_token %}
25                 {{ alumno_form.as_p }}
26                 <button type="submit" class="btn btn-primary">Agregar Alumno</button>
27             </form>
28         </div>
29         <div class="form-group">
30             <h3>Agregar Curso</h3>
31             <form method="POST" action="{% url 'agregar_curso' %}">
32                 {% csrf_token %}
33                 {{ curso_form.as_p }}
34                 <button type="submit" class="btn btn-primary">Agregar Curso</button>
35             </form>
36         </div>
37         <div class="form-group">
38             <h3>Agregar Nota</h3>
39             <form method="POST" action="{% url 'agregar_nota' %}">
40                 {% csrf_token %}
41                 {{ nota_form.as_p }}
42                 <button type="submit" class="btn btn-primary">Agregar Nota</button>
43             </form>
44         </div>
45     </div>
```

```
44 <hr>
45 <h2>Alumnos</h2>
46 <table class="table table-striped">
47   <thead>
48     <tr>
49       <th>ID</th>
50       <th>Nombre</th>
51       <th>Apellido</th>
52       <th>Fecha de Nacimiento</th>
53     </tr>
54   </thead>
55   <tbody>
56     {% for alumno in alumnos %}
57     <tr>
58       <td>{{ alumno.id }}</td>
59       <td>{{ alumno.nombre }}</td>
60       <td>{{ alumno.apellido }}</td>
61       <td>{{ alumno.fecha_nacimiento }}</td>
62     </tr>
63     {% endfor %}
64   </tbody>
65 </table>
66 <h2>Cursos</h2>
67 <table class="table table-striped">
68   <thead>
69     <tr>
70       <th>ID</th>
71       <th>Nombre</th>
72       <th>Descripcion</th>
73     </tr>
74   </thead>
75   <tbody>
76     {% for curso in cursos %}
77     <tr>
78       <td>{{ curso.id }}</td>
79       <td>{{ curso.nombre }}</td>
80       <td>{{ curso.descripcion }}</td>
81     </tr>
82     {% endfor %}
83   </tbody>
84 </table>
85 <h2>Notas</h2>
86 <table class="table table-striped">
87   <thead>
88     <tr>
89       <th>ID</th>
90       <th>Alumno</th>
91       <th>Curso</th>
92       <th>Nota</th>
93     </tr>
94   </thead>
95   <tbody>
96     {% for nota in notas %}
97     <tr>
98       <td>{{ nota.id }}</td>
99       <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

Agregar Nota

Alumno: Curso: Nota:

Alumnos

ID	Nombre	Apellido	Fecha de Nacimiento
1	Luis Guillermo	Luque	Jan. 6, 2004
2	Fernando	Garambel	Nov. 2, 2005
3	Fernando	Miguel	Feb. 11, 2005

Cursos

ID	Nombre	Descripción
1	Programacion Web 2	Curso de segundo año, primer semestre

Notas

ID	Alumno	Curso	Nota
1	Luis Guillermo Luque	Programacion Web 2	15

12. Referencias

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