



# PROGRAMACIÓN WEB 2



**Profesor(a):**

Carlo Jose Luis Corrales Delgado

**Estudiantes:**

Mamani Anahua, Victor Narciso

**Repositorio GitHub:**

<https://github.com/VictorMA18/Lab07-Django-Telusko>

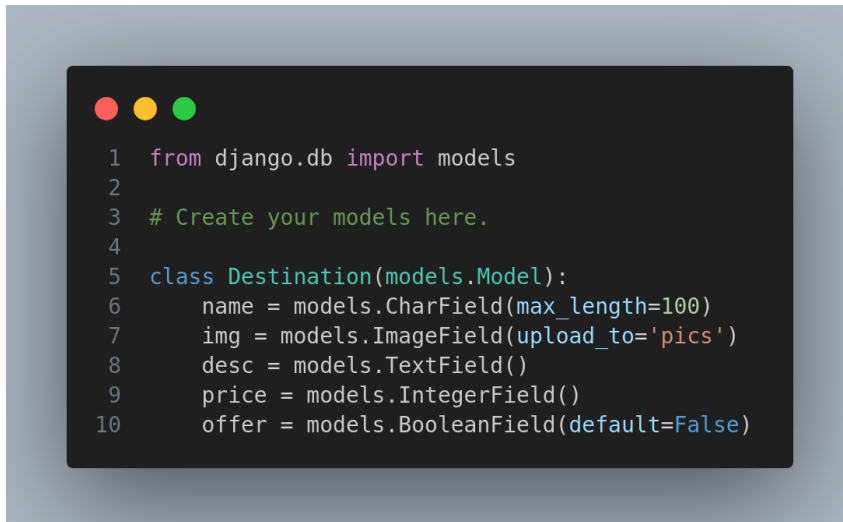
**Video:**

5 de junio, 2024

## TRAVELLO

### Clases en Models.py

Este código define un modelo en Django llamado Destination, que representa un destino turístico. Cada destino tiene un nombre, una imagen, una descripción, un precio y un indicador de oferta. El nombre se almacena como texto, la imagen se guarda en un directorio específico, la descripción es un texto largo, el precio es un número entero, y la oferta es un valor booleano que indica si hay una oferta especial para ese destino.

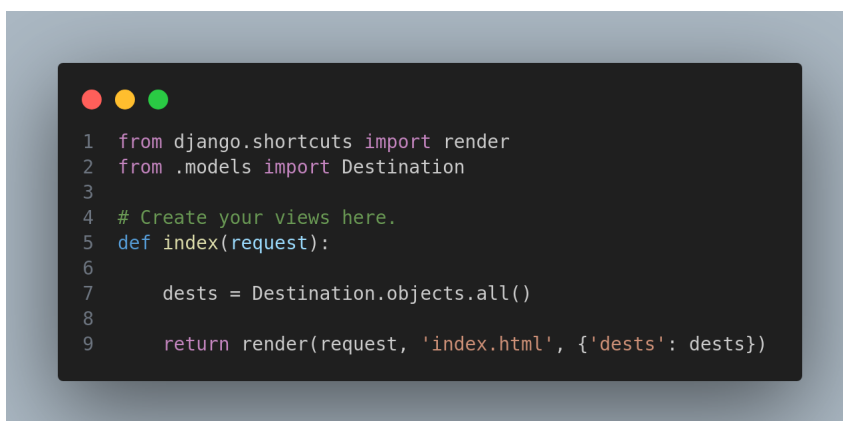


```
1 from django.db import models
2
3 # Create your models here.
4
5 class Destination(models.Model):
6     name = models.CharField(max_length=100)
7     img = models.ImageField(upload_to='pics')
8     desc = models.TextField()
9     price = models.IntegerField()
10    offer = models.BooleanField(default=False)
```

Figura 1: Código y Ejecución

### Funciones en Views.py

Este código define una vista en Django que renderiza una plantilla HTML llamada index.html. La vista obtiene todos los objetos Destination del modelo y los pasa a la plantilla como contexto, bajo el nombre dests. Cuando un usuario accede a la página asociada a esta vista, verá una lista de destinos turísticos que se han recuperado de la base de datos.



```
1 from django.shortcuts import render
2 from .models import Destination
3
4 # Create your views here.
5 def index(request):
6
7     dests = Destination.objects.all()
8
9     return render(request, 'index.html', {'dests': dests})
```

Figura 2: Código y Ejecución

## Funciones en Urls.py

Este archivo define las URL para la aplicación Django. La URL vacía (") está asociada a la vista index definida en el archivo views.py de la misma aplicación. Cuando un usuario visita la página principal del sitio, se llama a la función index en views.py, que renderiza la plantilla index.html.

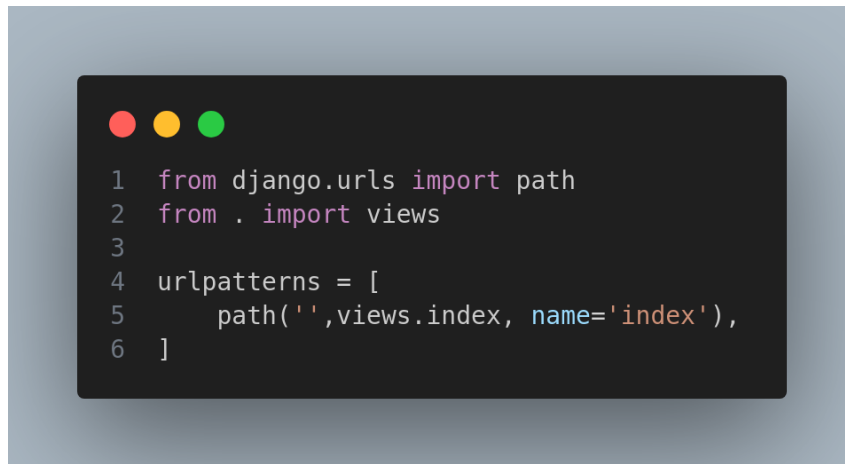


Figura 3: Código y Ejecución

## Funciones en Admin.py

En este código, se importa la clase Destination del archivo models.py de la misma aplicación Django. Luego, esta clase se registra en el panel de administración de Django utilizando admin.site.register(), lo que permite administrar los objetos de esta clase directamente desde el panel de administración. Esto facilita la gestión de los datos del modelo Destination a través de la interfaz de administración de Django.

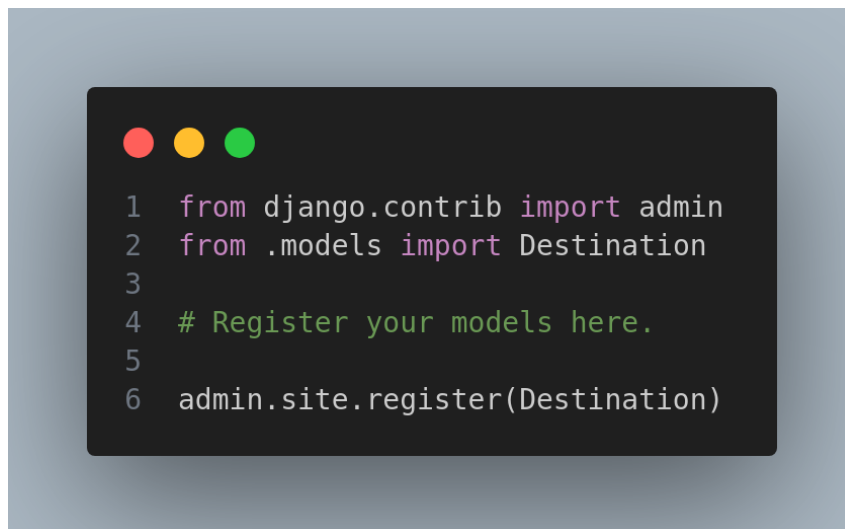


Figura 4: Código y Ejecución

## Los archivos HTML

Para poder visualizar el index nos ponemos en travello <http://127.0.0.1:8000>

```

1  {% load static%}
2  {% static "images" as baseUrl %}
3
4  <!DOCTYPE html>
5  <html lang="en">
6  <head>
7  <title>Travello</title>
8  <meta charset="utf-8">
9  <meta http-equiv="X-UA-Compatible" content="IE=edge">
10 <meta name="description" content="Travello template project">
11 <meta name="viewport" content="width=device-width, initial-scale=1">
12 <link rel="stylesheet" type="text/css" href="{% static 'styles/bootstrap4/bootstrap.min.css' %}">
13 <link href="{% static 'plugins/font-awesome-4.7.0/css/font-awesome.min.css' %}" rel="stylesheet" type="text/css">
14 <link rel="stylesheet" type="text/css" href="{% static 'plugins/OwlCarousel2-2.2.1/owl.carousel.css' %}">
15 <link rel="stylesheet" type="text/css" href="{% static 'plugins/OwlCarousel2-2.2.1/owl.theme.default.css' %}">
16 <link rel="stylesheet" type="text/css" href="{% static 'plugins/OwlCarousel2-2.2.1/animate.css' %}">
17 <link rel="stylesheet" type="text/css" href="{% static 'styles/main_styles.css' %}">
18 <link rel="stylesheet" type="text/css" href="{% static 'styles/responsive.css' %}">
19 </head>
20 <body>
21
22 <div class="super_container">
23
24 <!-- Header -->
25
26 <header class="header">
27 <div class="container">
28 <div class="row">
29 <div class="col">
30 <div class="header_content d-flex flex-row align-items-center justify-content-start">
31 <div class="header_content_inner d-flex flex-row align-items-end justify-content-start">
32 <div class="logo"><a href="index.html">Travello</a></div>
33 <nav class="main_nav">

```

Figura 5: Código y Ejecución

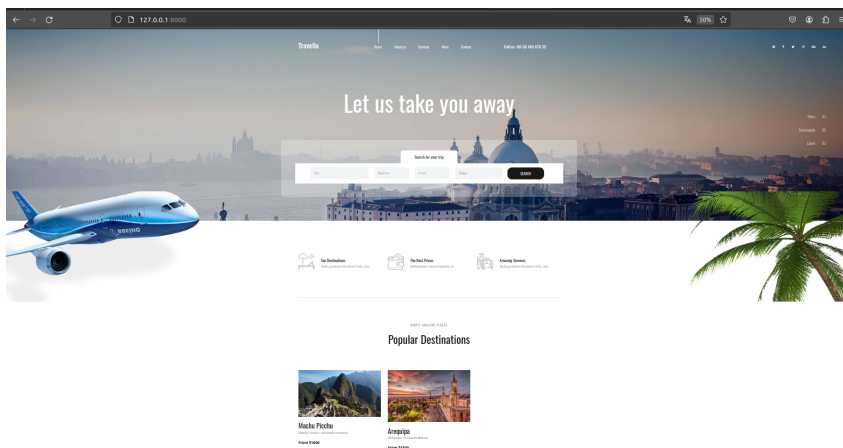


Figura 6: Código y Ejecución

## Base de Datos PostgradeSQL

### Entrar a Django Admin

Crear un destino en <http://127.0.0.1:8000/admin/login/?next=/admin/>

Cuando entremos hay que crear un superusuario para el admin y poder administrar la base de datos para esto hay crearlo desde la terminal

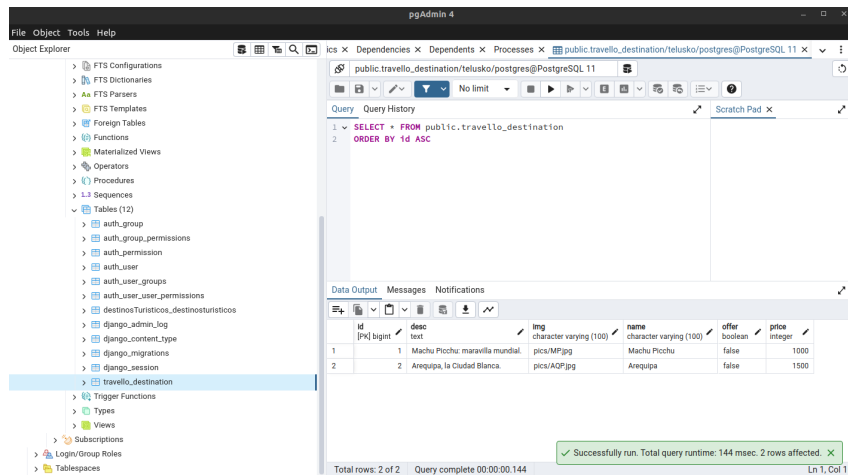


Figura 7: Código y Ejecución

## DESTINOSTURISTICOS

### Clases en Models.py

El modelo DestinosTuristicos define los atributos necesarios para representar un destino turístico en una base de datos Django. Cada destino tiene un nombre de ciudad, una imagen asociada que se carga en la carpeta 'pics' del directorio de medios, una descripción textual, un precio para el tour y un indicador de oferta, que por defecto es 'False'. Este modelo proporciona una estructura coherente para almacenar información sobre destinos turísticos en una aplicación web Django.

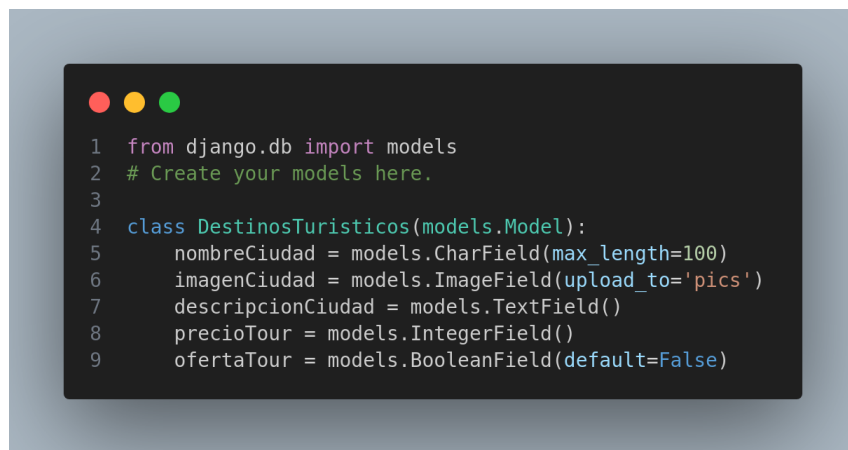


Figura 8: Código y Ejecución

### Funciones en Views.py

Este conjunto de vistas en Django se encarga de gestionar las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para los destinos turísticos. La vista creardestino maneja la creación de nuevos destinos, listardestino muestra todos los destinos disponibles, editardestino permite modificar destinos existentes y eliminardestino facilita la eliminación de destinos. Cada vista procesa las solicitudes HTTP correspondientes, ya sea guardando datos nuevos, mostrando información existente o eliminando registros según lo especificado en la lógica de la aplicación.

```

1 from django.shortcuts import render, get_object_or_404
2 from django.http import HttpResponseRedirect, HttpResponseBadRequest
3 from .forms import *
4 from .models import *
5 # Create your views here.
6 def crear_destino(request):
7     if request.method == 'POST':
8         form = DestinoForm(request.POST, request.FILES)
9         if form.is_valid():
10            form.save()
11            return HttpResponseRedirect("Se guardo el destino")
12        else:
13            form = DestinoForm()
14            return render(request, '../templates/crear_destino.html', {'form': form})
15
16 def listar_destino(request):
17     destinos = DestinosTuristicos.objects.all()
18     return render(request, '../templates/listar_destino.html', {'destinos': destinos})
19
20 def editar_destino(request, id=None):
21     if id:
22         destino = get_object_or_404(DestinosTuristicos, pk=id)
23         if request.method == 'POST':
24             form = DestinoForm(request.POST, request.FILES, instance=destino)
25             if form.is_valid():
26                 form.save()
27                 return HttpResponseRedirect("Se edito el destino")
28             else:
29                 form = DestinoForm(instance=destino)
30         else:
31             form = None
32
33     destinos = DestinosTuristicos.objects.all()
34     return render(request, '../templates/editar_destino.html', {'destinos': destinos, 'form': form, 'selected_destino': id})
35
36 def eliminar_destino(request):
37     if request.method == 'POST':
38         nombre_ciudad = request.POST.get('nombreCiudad')
39         destinos = DestinosTuristicos.objects.filter(nombreCiudad=nombre_ciudad)
40         if destinos.exists():
41             destino = destinos.first()
42             destino.delete()
43             return HttpResponseRedirect("Se eliminó el destino con éxito")
44         else:
45             return HttpResponseBadRequest("No se encontró ningún destino con el nombre de ciudad proporcionado")
46     else:
47         destinos = DestinosTuristicos.objects.all()
48         return render(request, '../templates/eliminar_destino.html', {'destinos': destinos})

```

Figura 9: Código y Ejecución

## Funciones en Urls.py

Estas son las URL configuradas para las vistas en la aplicación de destinos turísticos en Django. Cada URL está asociada a una función de vista específica. Por ejemplo, la URL `crearDestino/` está vinculada a la vista `creardestino`, que maneja la creación de nuevos destinos. Además, hay una URL `editarDestino/int:id/` que permite editar un destino específico identificado por su ID. Estas URL proporcionan puntos de acceso a las diferentes funcionalidades de la aplicación de destinos turísticos.

```

1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('crearDestino/', views.crear_destino, name='crear_destino'),
6     path('listarDestino/', views.listar_destino, name='listar_destino'),
7     path('editarDestino/', views.editar_destino, name='editar_destino'),
8     path('editarDestino/<int:id>/', views.editar_destino, name='editar_destino_id'),
9     path('eliminarDestino/', views.eliminar_destino, name='eliminar_destino'),
10 ]

```

Figura 10: Código y Ejecución

## Funciones en Forms.py

El formulario `DestinoForm` se define utilizando la clase `ModelForm` de Django, que se importa del módulo `django.forms`. Este formulario está asociado al modelo `DestinosTuristicos`. Las propiedades del formulario coinciden con los campos del modelo, y se especifican en el atributo `fields`. Además, se ha definido un método `save()` personalizado para guardar los datos del formulario en la base de datos.

Dentro de este método, se crea una instancia del modelo DestinosTuristicos con los datos del formulario y se guarda en la base de datos si commit es verdadero.

```
1 from django import forms
2 from .models import *
3
4 class DestinoForm(forms.ModelForm):
5     nombreCiudad = models.CharField(max_length = 100)
6     imagenCiudad = models.ImageField(upload_to = 'pics')
7     descripcionCiudad = models.TextField()
8     precioTour = models.IntegerField()
9     ofertaTour = models.BooleanField(default=False)
10
11 class Meta:
12     model = DestinosTuristicos
13     fields = ('nombreCiudad', 'imagenCiudad', 'descripcionCiudad', 'precioTour', 'ofertaTour')
14
15 def save(self, commit=True):
16     destinos_turisticos = super().save(commit=False)
17     if commit:
18         destinos_turisticos.save()
19     return destinos_turisticos
```

Figura 11: Código y Ejecución

## Funciones en Admin.py

Este fragmento de código Django importa la clase DestinosTuristicos del archivo models.py de la misma aplicación Django. Luego, registra esta clase en el panel de administración de Django utilizando `admin.site.register()`. Esto permite administrar los objetos de la clase DestinosTuristicos directamente desde el panel de administración de Django, lo que facilita la gestión de los datos relacionados con los destinos turísticos en la aplicación.

```
1 from django.contrib import admin
2 from .models import DestinosTuristicos
3
4 # Register your models here.
5
6 admin.site.register(DestinosTuristicos)
```

Figura 12: Código y Ejecución

## Los archivos HTML

Para poder visualizar el index nos ponemos en travello `http://127.0.0.1:8000`

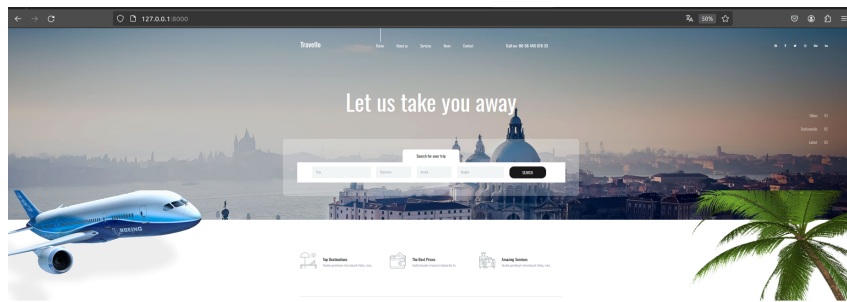


Figura 13: Código y Ejecución

Crear un destino en <http://127.0.0.1:8000/crearDestino/>

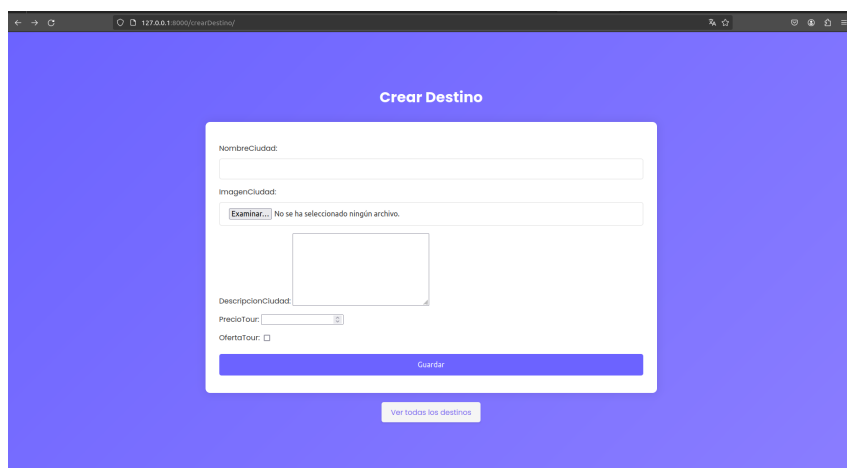


Figura 14: Código y Ejecución

Listar destinos en <http://127.0.0.1:8000/listarDestino/>



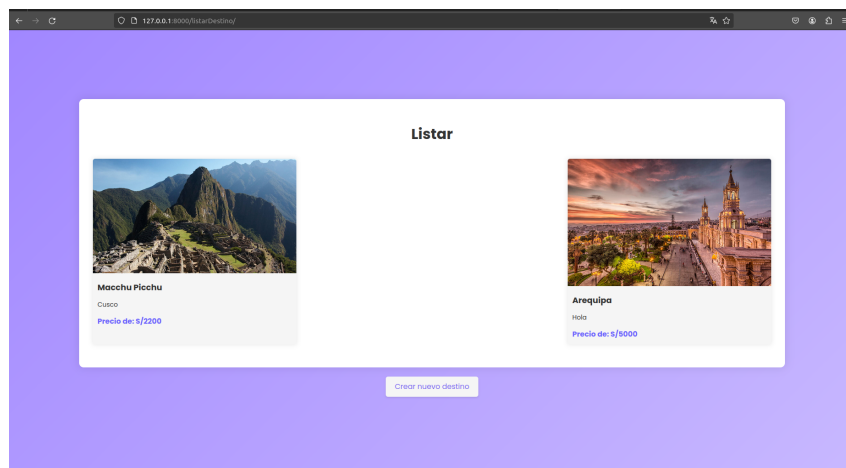


Figura 15: Código y Ejecución

Editar un destino en <http://127.0.0.1:8000/editarDestino/>

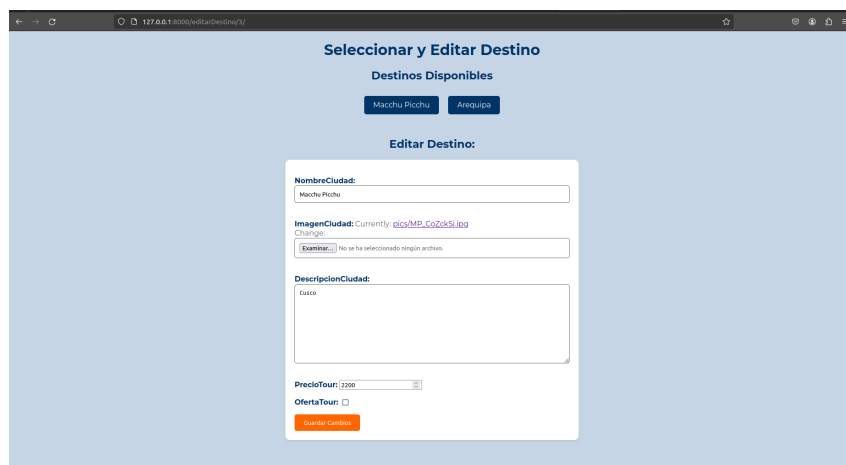


Figura 16: Código y Ejecución

Eliminar un destino en <http://127.0.0.1:8000/eliminarDestino/>

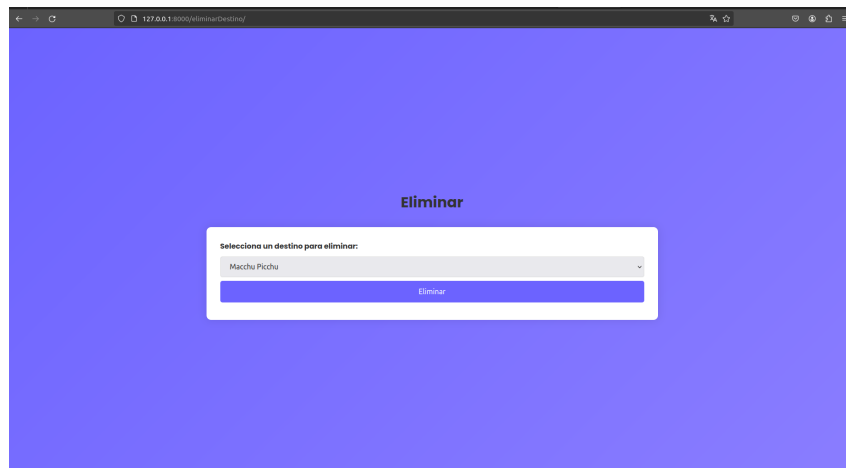


Figura 17: Código y Ejecución

## Base de Datos PostgradeSQL

### Entrar a Django Admin

Crear un destino en <http://127.0.0.1:8000/admin/login/?next=/admin/>

Cuando entremos hay que crear un superusuario para el admin y poder administrar la base de datos para esto hay crearlo desde la terminal

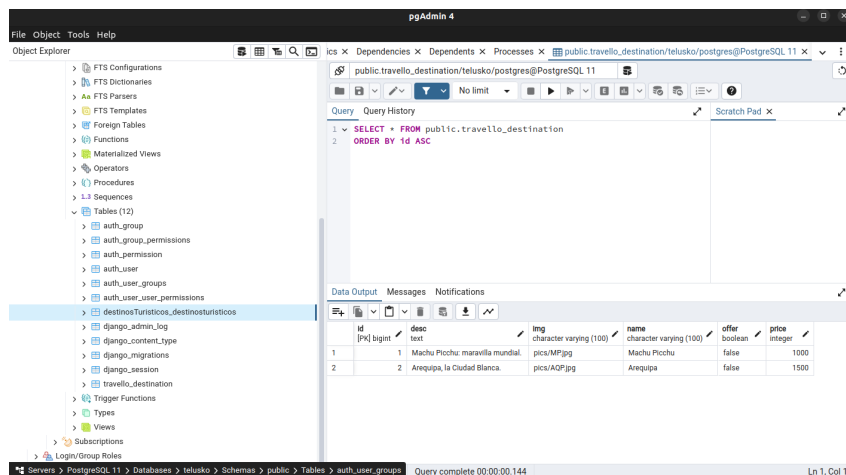


Figura 18: Código y Ejecución

## El proyecto es cual es TELUSKO

### Funciones en Urls.py en el proyecto

Este archivo configura las URL para el proyecto telusko. El patrón de URL vacío (") está incluido para las aplicaciones travello y destinosTuristicos, lo que significa que las URL definidas en los archivos urls.py de esas aplicaciones estarán disponibles en la raíz del sitio. Además, se incluye la URL para el panel de administración de Django en admin/.

```
1 """
2 URL configuration for telusko project.
3
4 The 'urlpatterns' list routes URLs to views. For more information please see:
5     https://docs.djangoproject.com/en/5.0/topics/http/urls/
6 Examples:
7 Function views
8     1. Add an import: from my_app import views
9     2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19 from django.conf import settings
20 from django.conf.urls.static import static
21
22 urlpatterns = [
23     path('', include('travello.urls')),
24     path('', include('destinosTuristicos.urls')),
25     path('admin/', admin.site.urls),
26 ]
27
28 urlpatterns = urlpatterns + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Figura 19: Código y Ejecución

Además, se agrega la configuración para servir archivos multimedia en desarrollo, agregando las URL MEDIA-URL y MEDIAROOT definidas en la configuración de Django. Esto permite que los archivos multimedia, como imágenes cargadas, se sirvan correctamente durante el desarrollo del proyecto.

## Funciones en Settings.py

El fragmento proporcionado configura las aplicaciones y la conexión a la base de datos en un proyecto Django. En la lista INSTALLED\_APPS, se enumeran las aplicaciones instaladas, incluidas las específicas del proyecto (destinosTuristicos y travello) y las aplicaciones predeterminadas de Django para funcionalidades como la administración, la autenticación y el manejo de sesiones. Por otro lado, en la sección DATABASES, se define la configuración para conectarse a una base de datos PostgreSQL, especificando el nombre de la base de datos, el nombre de usuario, la contraseña y el host. Estos elementos son esenciales para establecer el entorno de desarrollo y la infraestructura de almacenamiento de datos para el proyecto Django.

```
1  INSTALLED_APPS = [  
2      'destinosTuristicos.apps.DestinosturisticosConfig',  
3      'travello.apps.TravelloConfig',  
4      'django.contrib.admin',  
5      'django.contrib.auth',  
6      'django.contrib.contenttypes',  
7      'django.contrib.sessions',  
8      'django.contrib.messages',  
9      'django.contrib.staticfiles',  
10 ]
```

Figura 20: Código y Ejecución

```
1  DATABASES = {  
2      'default': {  
3          'ENGINE': 'django.db.backends.postgresql',  
4          'NAME': 'telusko',  
5          'USER': 'postgres',  
6          'PASSWORD': '1234',  
7          'HOST': 'localhost'  
8      }  
9  }
```

Figura 21: Código y Ejecución

## Ejecutar el servidor

El comando `python manage.py makemigrations colegio` se utiliza en Django para crear nuevas migraciones basadas en los cambios realizados en los modelos del aplicativo colegio. Las migraciones son archivos que describen cómo modificar la estructura de la base de datos para mantenerla sincronizada con los modelos de Django. Este comando analiza los modelos en la aplicación colegio y genera los archivos de migración necesarios para aplicar esos cambios a la base de datos.

```
python manage.py makemigrations travello  
python3 manage.py sqlmigrate travello 0001  
python3 manage.py sqlmigrate travello 0002  
python manage.py makemigrations destinosTuristicos  
python manage.py sqlmigrate destinosTuristicos 0001  
python manage.py migrate
```

El comando `python manage.py runserver` se utiliza en Django para iniciar el servidor de desarrollo local. Una vez ejecutado, el servidor se activa y permite acceder a la aplicación web en desarrollo a través de un navegador en la dirección `http://localhost:8000/` por defecto. Es una herramienta fundamental

durante el proceso de desarrollo de aplicaciones web con Django, ya que proporciona un entorno de prueba para probar y depurar el código antes de desplegar la aplicación en un entorno de producción.

```
python manage.py runserver
```

## Acceder a la aplicacion en el navegador

travello `http://127.0.0.1:8000`

Crear un destino en `http://127.0.0.1:8000/crearDestino/`

Listar destinos en `http://127.0.0.1:8000/listarDestino/`

Editar un destino en `http://127.0.0.1:8000/editarDestino/`

Eliminar un destino en `http://127.0.0.1:8000/eliminarDestino/`

## Flip

No carga mi video en flip por ellos use un link de drive

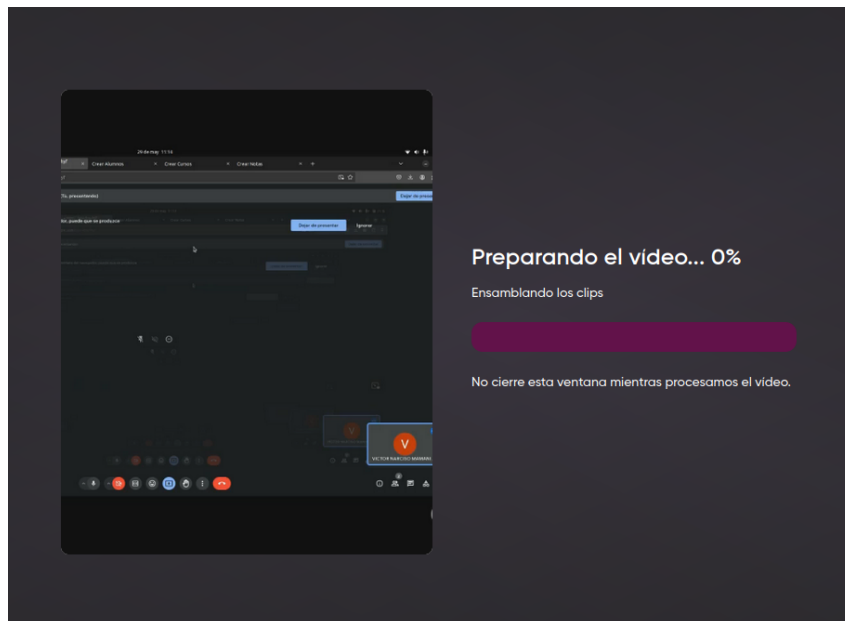


Figura 22: Código y Ejecución

**URL de video de explicación:**

**URL de repositorio de GitHub:** <https://github.com/VictorMA18/Lab07-Django-Telusko>