1. **Crear entorno virtual**

python -m venv venv

2. **Crear requirements.txt**

Creas en la carpeta del proyecto un archivo requirements.txt con django. Para desplegar en azure deberá contener esto:
django>=5.0,<6.0

django-crispy-forms>=2.0

crispy-bootstrap5>=2024.1

whitenoise>=6.0

gunicorn>=21.0

psycopg2-binary>=2.9.6

Así la persona que lo clone o el proceso de despliegue sabrá que librerías necesitan ser instaladas.

3. **Activar entorno virtual e instalar requirements**

venv/Scripts/activate

Si da error hacer esto primero:

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

Después, activar requirements: pip install -r requirements.txt

4. **Crear el proyecto django**

Comandos: django-admin startproject project .

django-admin startapp nombreApp, en el caso del video relecloud.

Luego, meter en los settings el apartado de installe_apps, relecloud.apps.RelecloudConfig, el nombre está en apps.py

5. **Vistas/Funciones que reciben una peticion http y devuelve respuesta**

Hay que crear en la app **relecloud/urls.py**

- from django.urls import path

- from . Import views

Creamos urlpatterns [

    path('', views.index, name='index'),

    path('about', views.about, name='about'),

    path('destinations', views.destinations, name='destinations'),

]

Y en el Global **Project/urls.py**

- From django.urls import path, include

- from django.contrib import admin

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('relecloud.urls')),

]

En relecloud/**views.py**, metemos:

```
from django.shortcuts import render

from django.urls import reverse_lazy

from . import models

from django.views import generic

from django.contrib.messages.views import SuccessMessageMixin


def index(request):

return render(request, 'index.html')

def about(request):

return render(request, 'about.html')

def destinations(request):

all_destinations = models.Destination.objects.all()

return render(request, 'destinations.html', { 'destinations': all_destinations})
```

6. Modelos/BBDD

from django.db import models

from django.urls import reverse

class Nombre(model.Model): **tipos CharField, Textfield, IntegerField**

variable = models.tipo(

unique = true, ejmplo

null = false, ejemplo

)

7. Estructura html

Base.html: contiene {% load static %} y el resto normal

Index y otras páginas:

{% extended “basic.html” %}

{% block title %} {% endblock title %}

{% block content %} aqui html normal {% endblock content %}

8. Estructura archivos

Para acceder a imagenes <img src="{% static 'res/img/rocket.jpg' %}>

Venv, project, relecloud, requerementes. Dentro de relecloud archivos y static con css e img y templates con html

9. Pasar a azure y migrar base de datos

import os

ALLOWED_HOSTS = [

  'practicasw.azurewebsites.net',

  'localhost',

  '127.0.0.1',

  '*' # Temporal para debugging

]

INSTALLED_APPS = [Añadir esas

  'crispy_forms',

  'crispy_bootstrap4',

  'relecloud.apps.RelecloudConfig'

]

DATABASES = {

  "default": {

    "ENGINE": "django.db.backends.postgresql",

    "NAME": "postgres",

    "USER": "nicolasitopertusato",

    "PASSWORD": os.getenv("DB_PASSWORD", "la contra que tenga!"),

    "HOST": "pertusato.postgres.database.azure.com",

    "PORT": "5432",

```
        "OPTIONS": {
            "sslmode": "require",
        },
    CRISPY_ALLOWED_TEMPLATE_PACKS = "bootstrap4"
    CRISPY_TEMPLATE_PACK = "bootstrap4"


    CSRF_TRUSTED_ORIGINS = ['https://practicasw.azurewebsites.net']
```

Crear super usuario

Si da problemas primero python manage.py migrate, este comando aplica cambios a la bd

Y ya luego python manage.py createsuperuser

Correr server

python manage.py runserver

Crear modelo

Creas el modelo en py y lo ejecutas con Python manage.py makemigrations, este comando genera un archivo con cambios. Ventajas no escribes SQL y usas objetos de Python.

Que es el entorno virtual: Carpeta aislada con intérprete de Python y librerías propias del proyecto. Evita conflictos de versiones entre proyectos, asegura reproducibilidad.

Modelos: Clases Python que representan tablas de BD. Makemigrations genera el archivo con cambios y el migrate aplica los cambios a la BD

Django: Framework de alto nivel de Python para crear aplicaciones web. Sigue el principio DRY para reutilizar código. Es seguro porque protege

Crispy forms: librería de django que sirve para mejorar la forma en la que se muestan los formularios.

Manage.py: es un script que django crea automáticamente en cada proyecto. Sirve como control remoto para ejecutar comandos relacionados con el projecto.

PostgresSQL: sistema de gestión de bbdd relacional de código abierto. Soporta consultas, JSON, funciones personalizadas y demás.