



# Masterclass Django

Adrián Ezquerro

# ¿Quién soy?

**¡Bienvenidos!**



[linkedin.com/in/adrianezd](https://www.linkedin.com/in/adrianezd)



<https://github.com/adrianezd>



[adrianezd@gmail.com](mailto:adrianezd@gmail.com)



# ¿Qué vamos a aprender ?

---

## ¿Qué es Python?

Pequeña introducción a este lenguaje de alto nivel interpretado.

01



## Entornos en python

Ventajas, para qué son, cómo se despliegan...

02



## ¿Qué es Django ?

Que modelo usa, para qué sirve...

03



04

## Objetivos

Propuestos en esta masterclass



05

## Despliegue de servidor en Django

Ejemplo y ejercicio interactivo



06

## Expansión y posibilidades

Cosas útiles para Django



# Introducción

---

Python es un lenguaje de *alto nivel* de programación *interpretado* cuya filosofía hace hincapié en la facilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo, ejemplos: Instagram, Netflix, Spotify, Panda 3D, entre otros.

# ¿ Qué podemos desarrollar con Python ?

---



## **IA, MACHINE LEARNING...**

Programación de algoritmos capaces de aprender a realizar tareas.



## **CIENCIA DE DATOS**

Analizar datos y automatizar operaciones



## **SCRIPTING, APLICACIONES WEB**

Crear aplicaciones empresariales fiables y escalables

# Uso de python

---



## Muchos usos

IA, ML, Ciencia de datos, backend..



## Fácil aprendizaje

Sintaxis sencilla



## Gran expansión

Top 5 en los últimos años



# Entornos

---

Son directorios **de** instalación aislados. Este aislamiento te permite localizar la instalación **de** las dependencias **de** tu proyecto, sin obligarte a instalarlas en todo el sistema

# Ventajas de usar un entorno

---



**AISLAMIENTO DE LIBRERÍAS**



**SEPARACIÓN DE PROYECTOS**

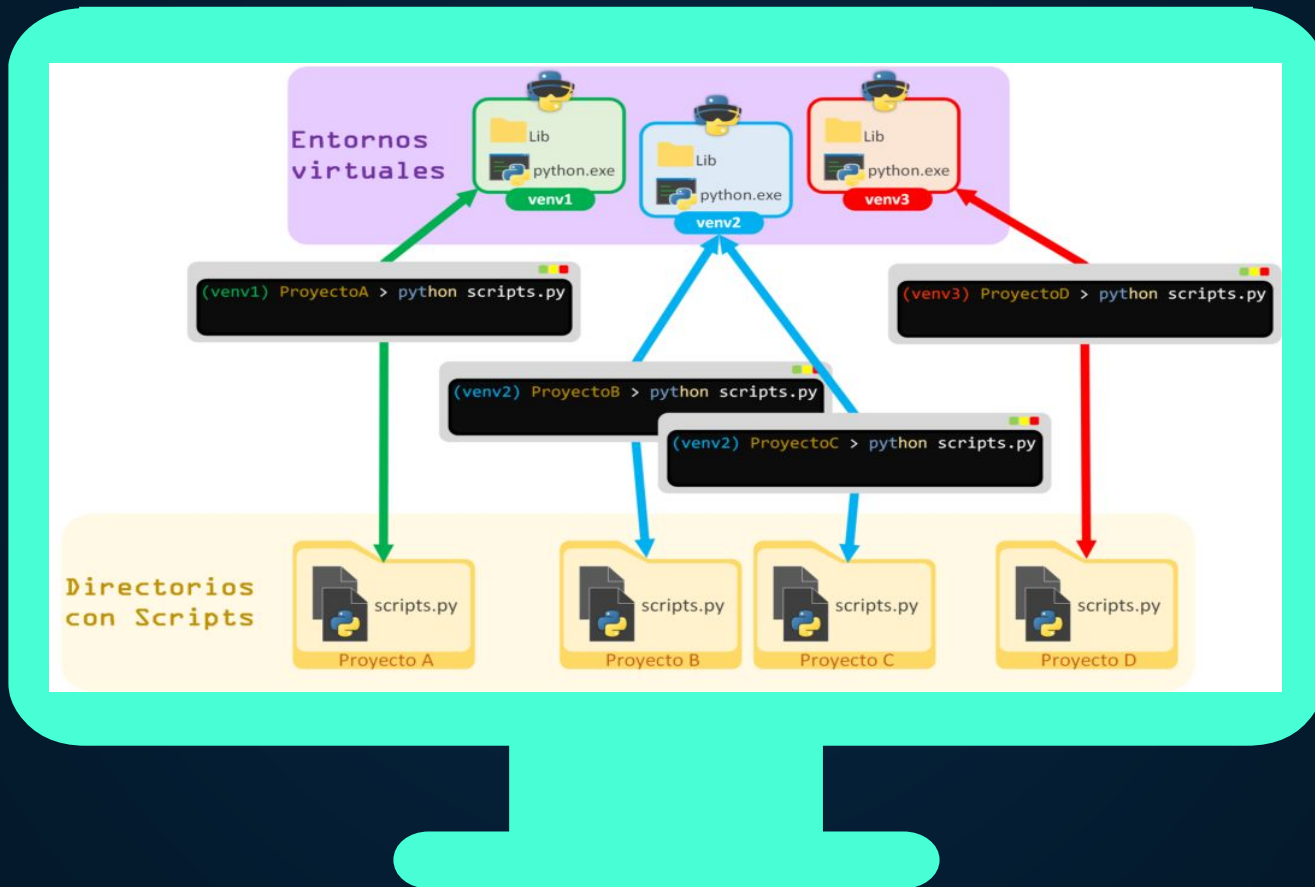


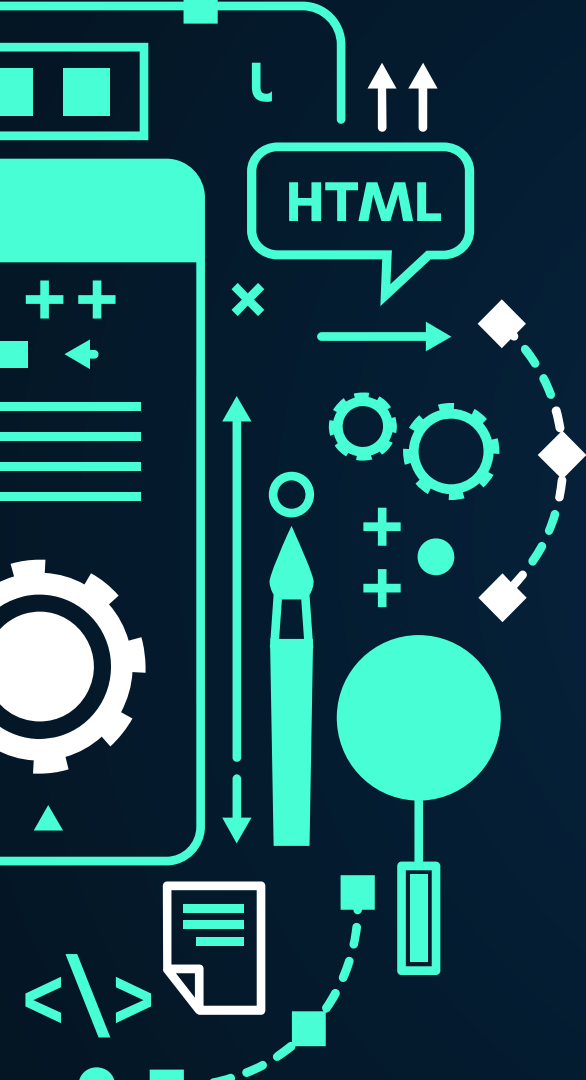
**PRUEBAS DE VERSIONES**





# VIRTUALENV





# DJANGO

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo–vista–controlador. (MVC)

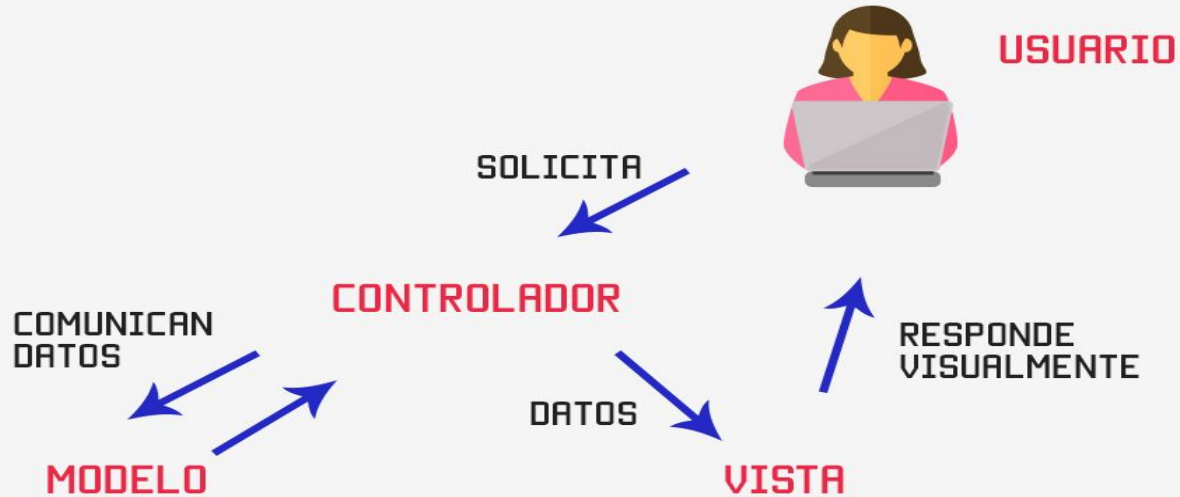


# MVC

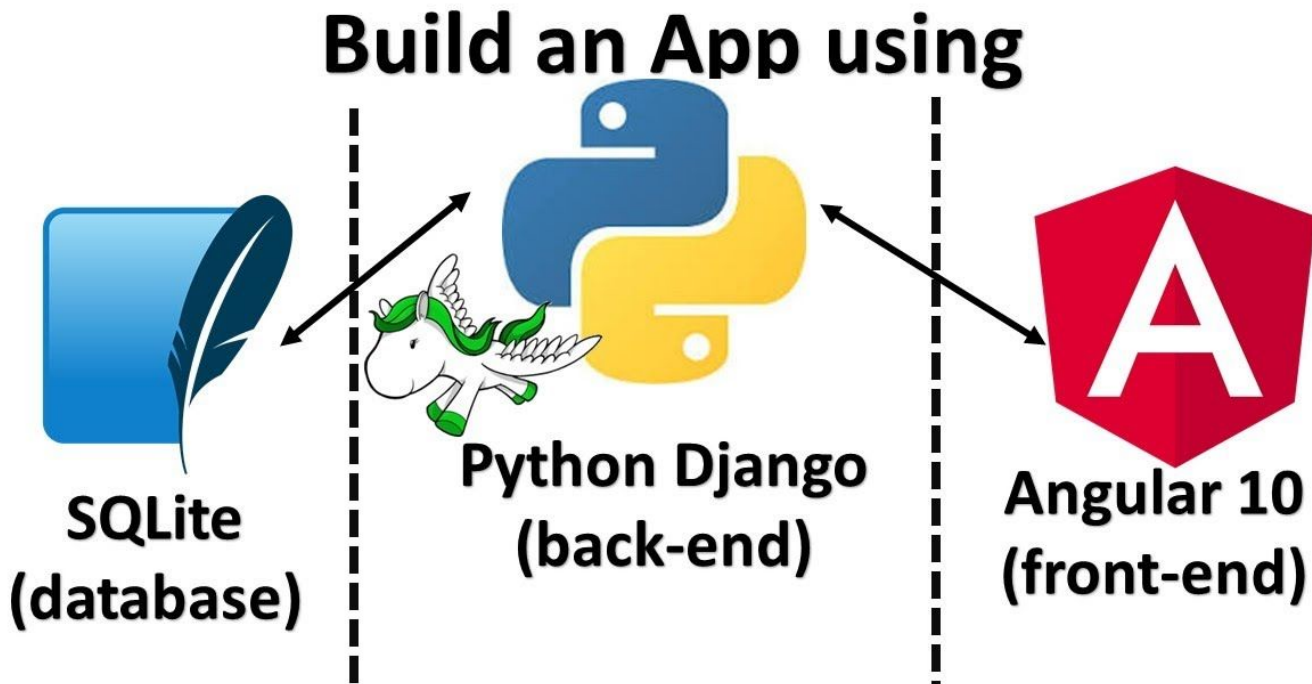
---



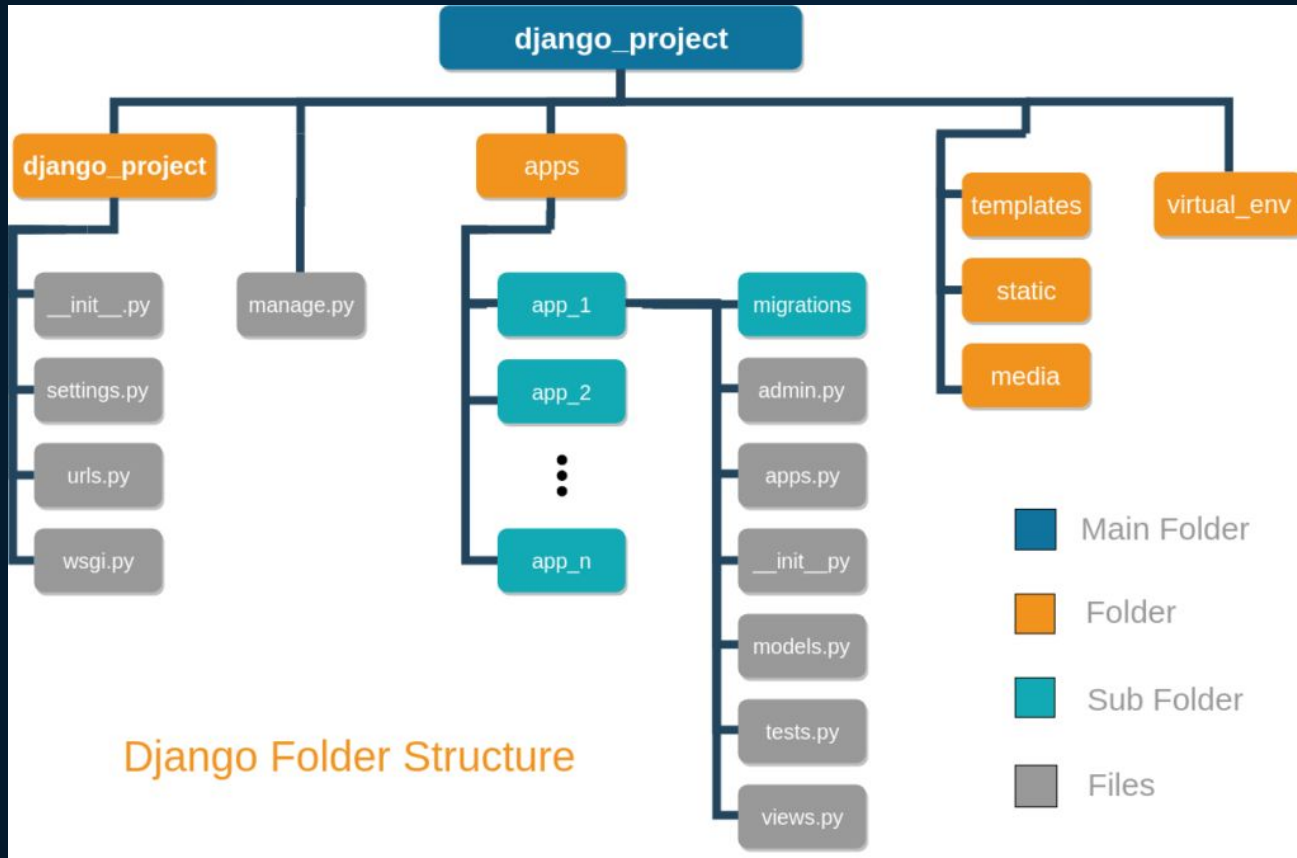
# MVC



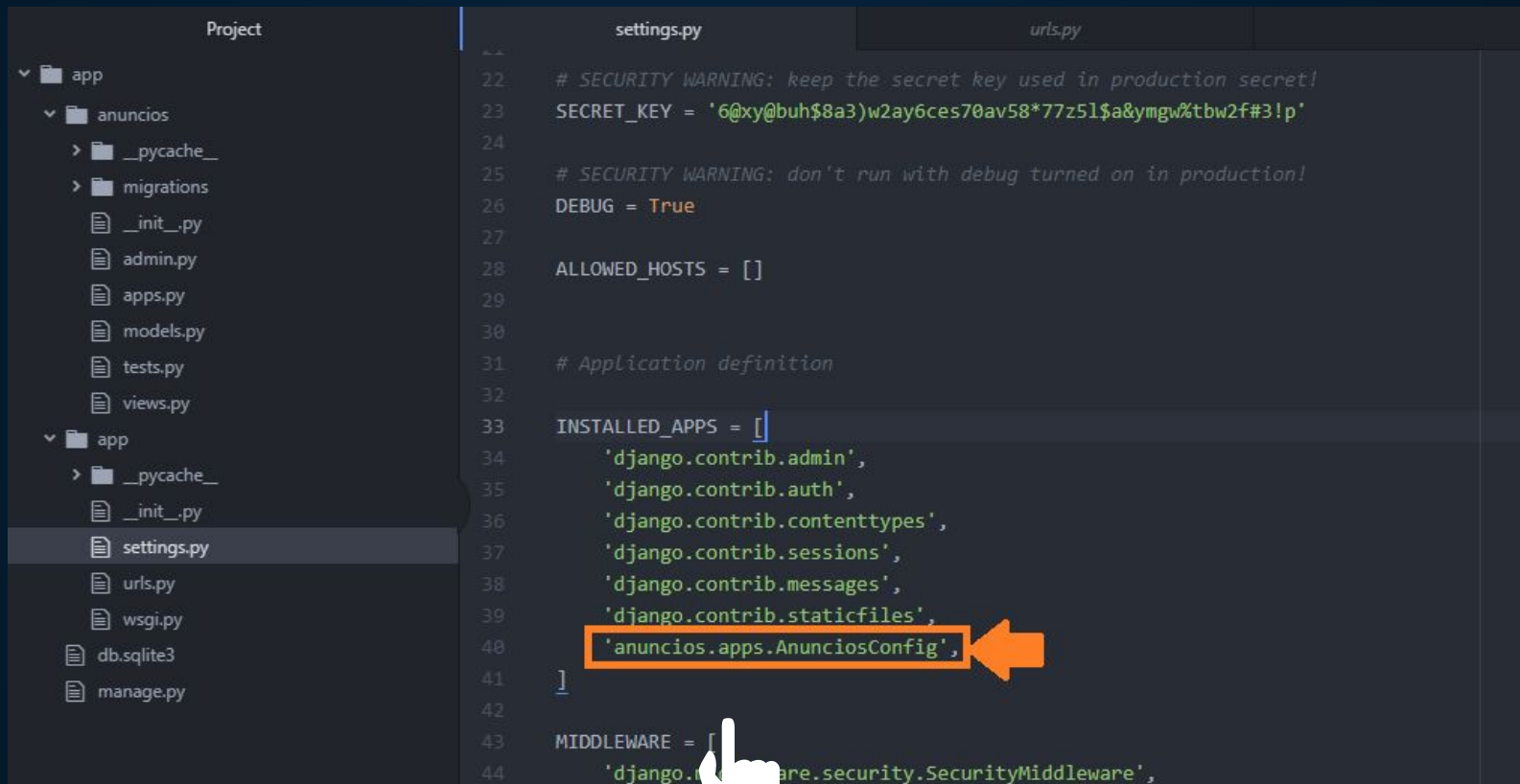
# MVC



# DJANGO



# DJANGO (Arbol)



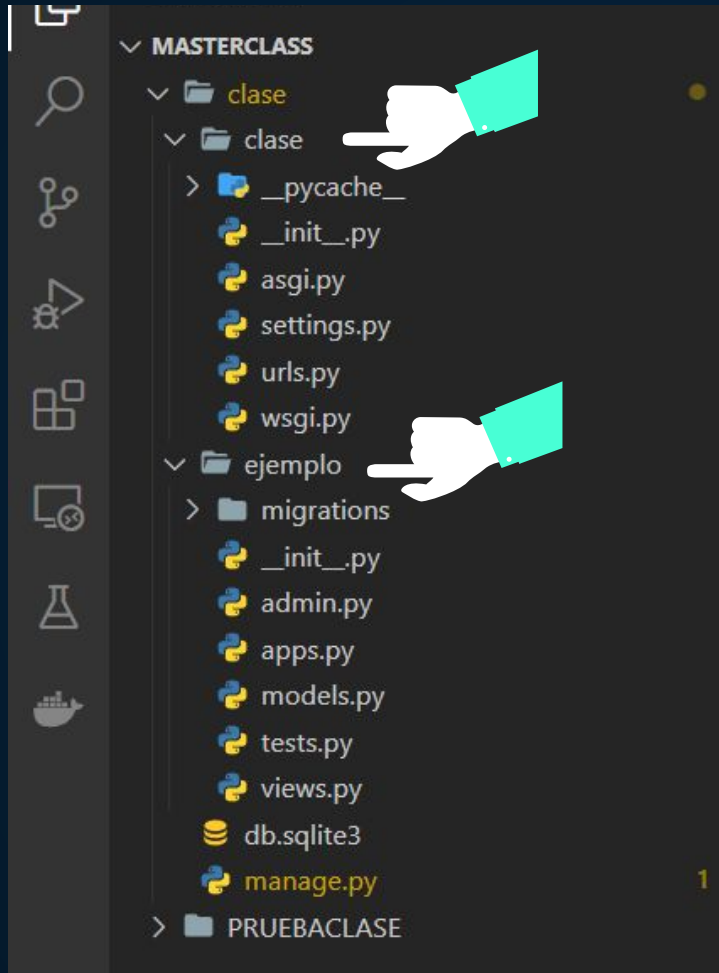
The image shows a code editor with a project structure on the left and the `settings.py` file on the right. The project structure includes a folder named `app` containing a subfolder `anuncios` and several Python files. The `settings.py` file contains configuration for a Django project, including the `SECRET_KEY`, `DEBUG` mode, `ALLOWED_HOSTS`, and the `INSTALLED_APPS` list. The `INSTALLED_APPS` list includes `'anuncios.apps.AnunciosConfig'`, which is highlighted with an orange box and a red arrow. A hand cursor is pointing at the `INSTALLED_APPS` list.

**Project Structure:**

- Project
  - app
    - anuncios
      - \_\_pycache\_\_
      - migrations
      - \_\_init\_\_.py
      - admin.py
      - apps.py
      - models.py
      - tests.py
      - views.py
    - app
      - \_\_pycache\_\_
      - \_\_init\_\_.py
      - settings.py
      - urls.py
      - wsgi.py
      - db.sqlite3
      - manage.py

**settings.py:**

```
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = '6@xy@buh$8a3)w2ay6ces70av58*77z5l$a&ymgw%tbw2f#3!p'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
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     'anuncios.apps.AnunciosConfig',
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
```



# DJANGO



# DJANGO (Settings.py)

```

Iniciar settings.py x
clase > clase > settings.py > ...
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-aumn6#uit76@@_b(nuo2j5nn55vxinkbcy6&($d$^ysoys0g%e'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
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 ]
41
```

# DJANGO (Settings.py)

---

```
WSGI_APPLICATION = 'clase.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases
```

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': BASE_DIR / 'db.sqlite3',  
    }  
}
```

```
# Password validation
```

```
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators
```



# DJANGO (Settings.py)

---

```
# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

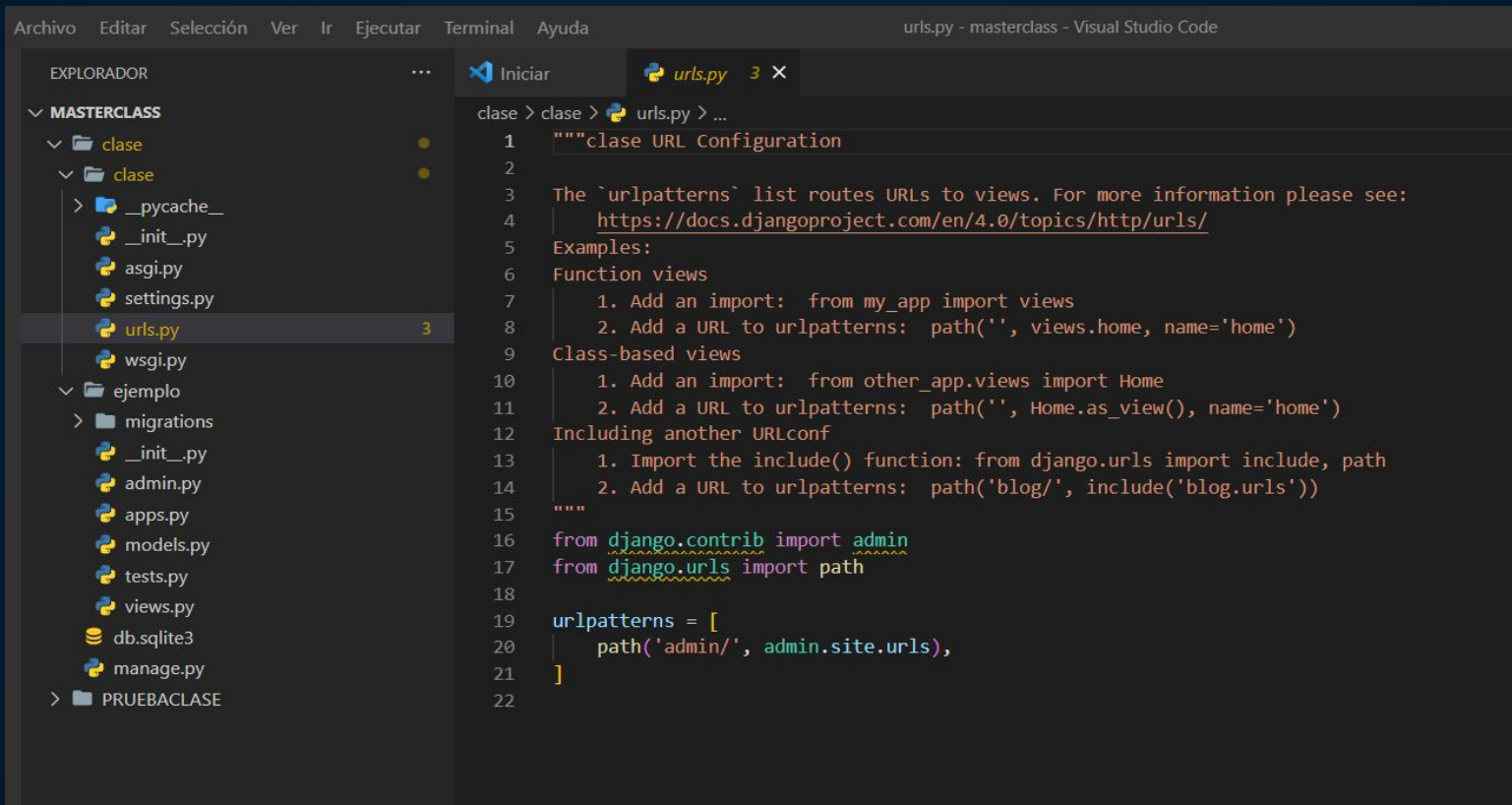

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC_URL = 'static/'


# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

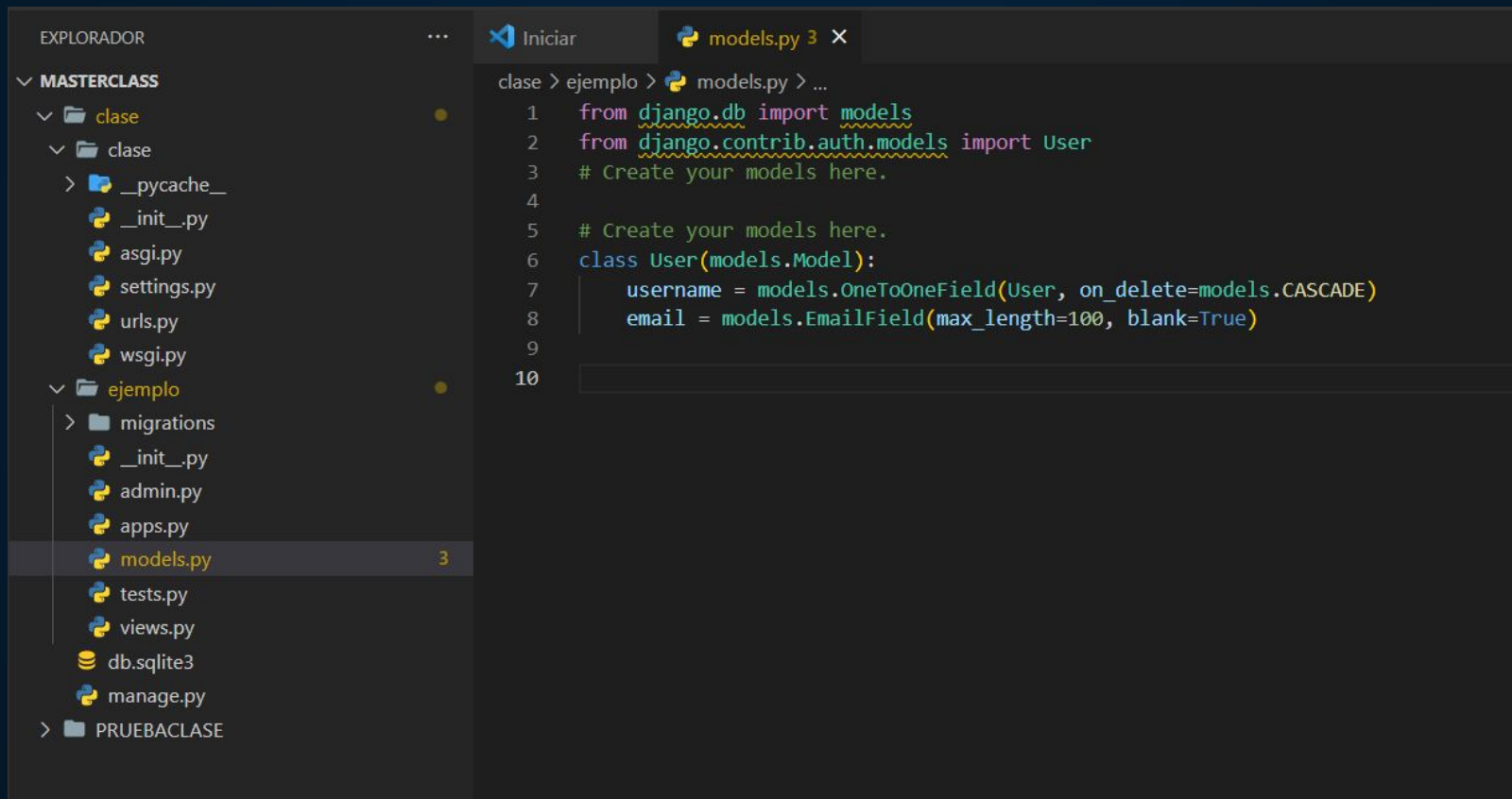
# DJANGO (Urls.py - Proyecto)



The screenshot displays the Visual Studio Code interface with a Django project named 'masterclass'. The Explorer sidebar on the left shows the project structure, including a 'class' folder containing 'urls.py' (which is selected and highlighted). Other files in the 'class' folder include '\_\_pycache\_\_', '\_\_init\_\_.py', 'asgi.py', 'settings.py', and 'wsgi.py'. Below the 'class' folder is an 'ejemplo' folder containing 'migrations', '\_\_init\_\_.py', 'admin.py', 'apps.py', 'models.py', 'tests.py', 'views.py', 'db.sqlite3', and 'manage.py'. At the bottom of the Explorer is a folder named 'PRUEBA CLASE'. The main editor window shows the content of 'urls.py', which includes a docstring explaining the 'urlpatterns' list, examples of function and class-based views, and the actual URL configuration code at the bottom.

```
class > class > urls.py > ...
1  """class URL Configuration
2
3  The `urlpatterns` list routes URLs to views. For more information please see:
4  | https://docs.djangoproject.com/en/4.0/topics/http/urls/
5  | Examples:
6  | Function views
7  |     1. Add an import: from my_app import views
8  |     2. Add a URL to urlpatterns: path('', views.home, name='home')
9  | Class-based views
10 |     1. Add an import: from other_app.views import Home
11 |     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 | Including another URLconf
13 |     1. Import the include() function: from django.urls import include, path
14 |     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 | """
16 from django.contrib import admin
17 from django.urls import path
18
19 urlpatterns = [
20 |     path('admin/', admin.site.urls),
21 | ]
22
```

# DJANGO ( Models.py )



# DJANGO ( Views.py )

views.py sensores 4, M

views.py blog 1 X

Untitled-1

blog > views.py > ...

```
1  from django.shortcuts import render, redirect
2  from blog.models import Categoria, Post
3
4  # Create your views here.
5
6  def blog(request):
7      if not request.user.is_authenticated:
8          return redirect('autenticacion:loguear')
9      posts=Post.objects.all()
10     return render(request, "blog/blog.html",{"posts":posts})
11
12 def categoria(request, categoria_id):
13     if not request.user.is_authenticated:
14         return redirect('autenticacion:loguear')
15     categoria=Categoria.objects.get(id=categoria_id)
16     posts=Post.objects.filter(categorias=categoria)
17     return render(request, "blog/categoria.html",{"categoria":categoria, "posts":posts})
18
19
20
```

# DJANGO (Urls.py - App)

views.py 4, M

urls.py 1, M

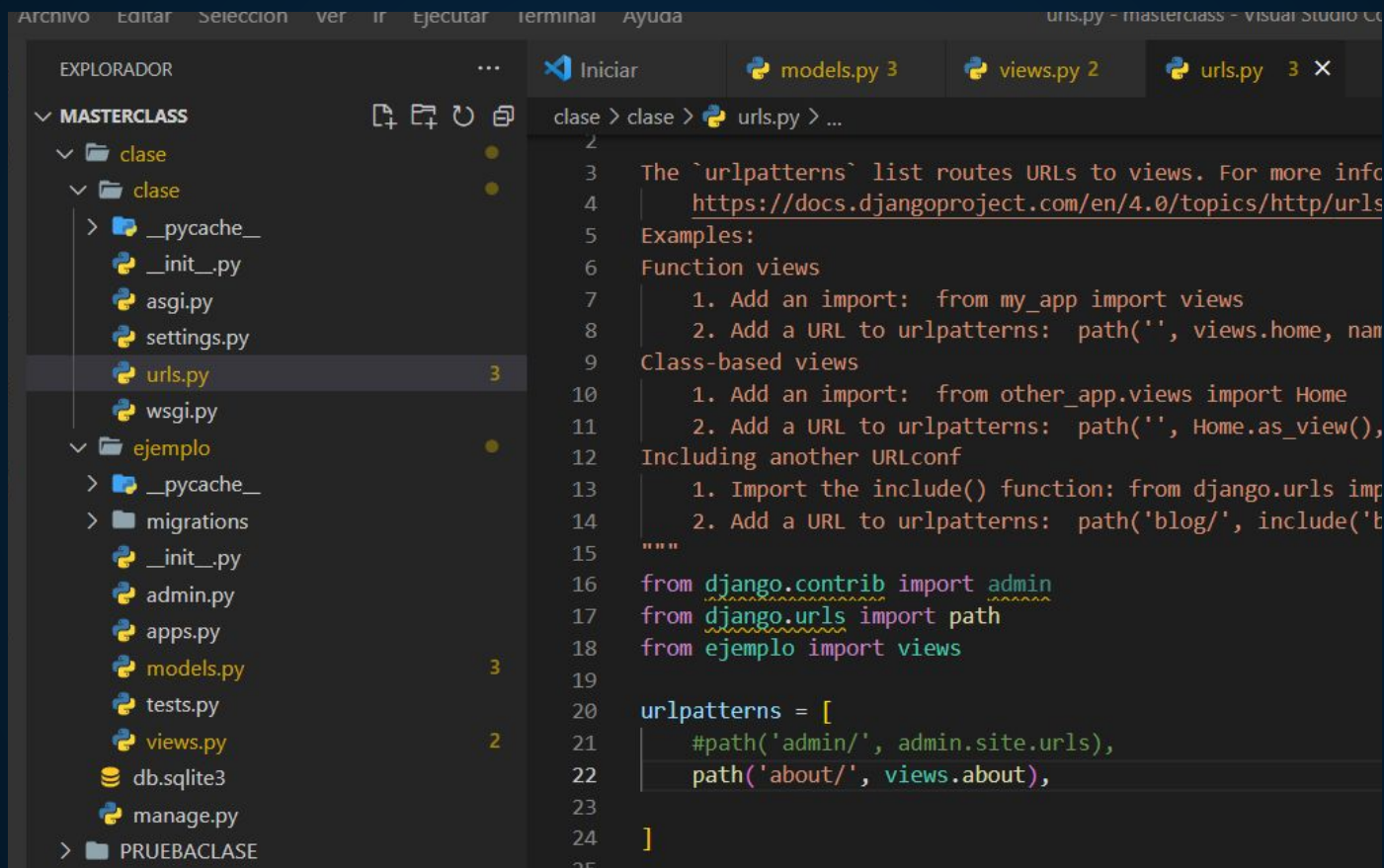
Untitled-1

autenticacion > urls.py > ...

```
1 from django.urls import path
2 from .views import VRegistro, cerrar_sesion, logear, logear2
3
4 app_name='autenticacion'
5
6 urlpatterns = [
7     path('cerrar_sesion', cerrar_sesion, name="cerrar_sesion"),
8     path('', logear2, name="logear"),
9 ]
```



# DJANGO (Urls.py - Proyecto)





# PARTE PRÁCTICA

---



# Expansión y posibilidades

---



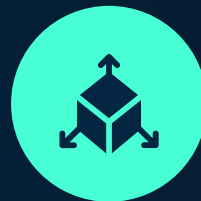
## ESCALABILIDAD

Capacidad de adaptación y respuesta frente a un aumento significativo de los recursos



## PROYECTOS CON BBDD GRANDES

Spotify, Youtube....  
Guardan grandes bases de datos



## BACKEND IMPORTANTE

El MVC que tiene es muy beneficioso, y Django es perfectamente combinable con Vue, Angular...



# PREGUNTAS

---

¿ SENCILLEZ ?



MVC O MTV



¿ FLASK ES MEJOR ?



