<codoa codo/>

DJANGO
Clase 31

Django: Introducción

BA

Les damos la bienvenida

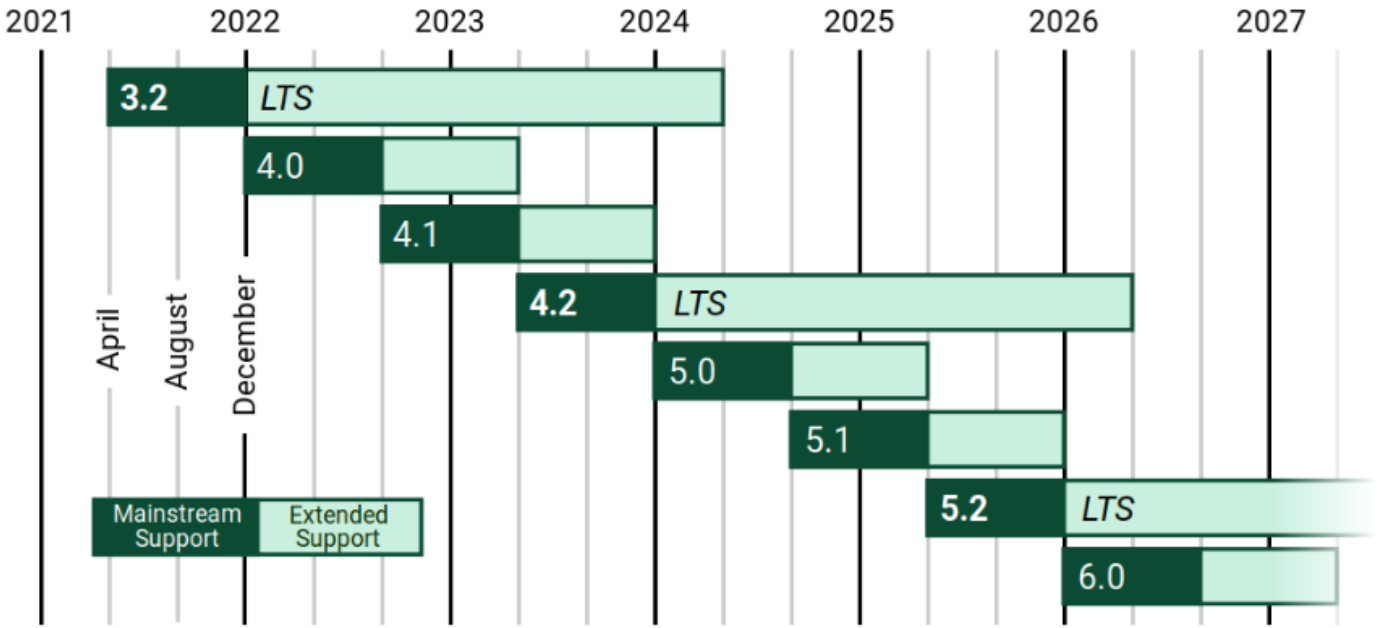
Vamos a comenzar a grabar la clase

Versión de Python

Release	Released	Security Support	Latest
3.11	5 months ago (24 Oct 2022)	Ends in 4 years and 7 months (24 Oct 2027)	3.11.2 (07 Feb 2023)
3.10	1 year and 5 months ago (04 Oct 2021)	Ends in 3 years and 6 months (04 Oct 2026)	3.10.10 (07 Feb 2023)
3.9	2 years and 5 months ago (05 Oct 2020)	Ends in 2 years and 6 months (05 Oct 2025)	3.9.16 (06 Dec 2022)
3.8	3 years and 5 months ago (14 Oct 2019)	Ends in 1 year and 6 months (14 Oct 2024)	3.8.16 (06 Dec 2022)
3.7	4 years and 9 months ago (26 Jun 2018)	Ends in 3 months (27 Jun 2023)	3.7.16 (06 Dec 2022)
3.6	6 years ago (22 Dec 2016)	Ended 1 year and 3 months ago (23 Dec 2021)	~~3.6.15~~

ref: Python | endoflife.date

Versión de Django



ref: [Download Django | Django (djangoproject.com)]

Django + Python

Django version	Python versions
3.2	3.6, 3.7, 3.8, 3.9, 3.10 (added in 3.2.9)
4.0	3.8, 3.9, 3.10
4.1	3.8, 3.9, 3.10, 3.11 (added in 4.1.3)
4.2	3.8, 3.9, 3.10, 3.11
5.0	3.10, 3.11, 3.12

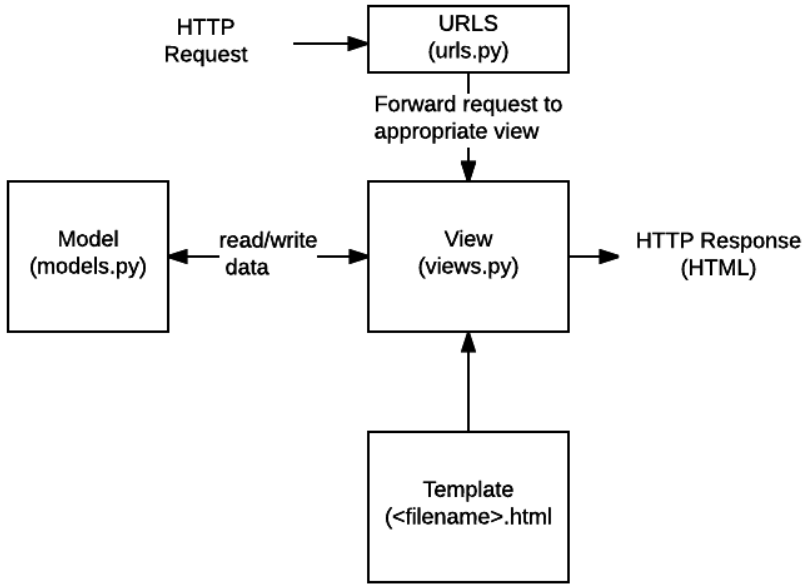
ref: FAQ: Installation | Django documentation | Django (djangoproject.com)

¿Qué es Django?

Django es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles.

- Gratuito.
- Código abierto.
- Gran comunidad y documentación.
- Incluye librerías extras.
- Escalable.
- Versátil.

Patrón MVT



Modelo: la capa de acceso a la base de datos (ORM).

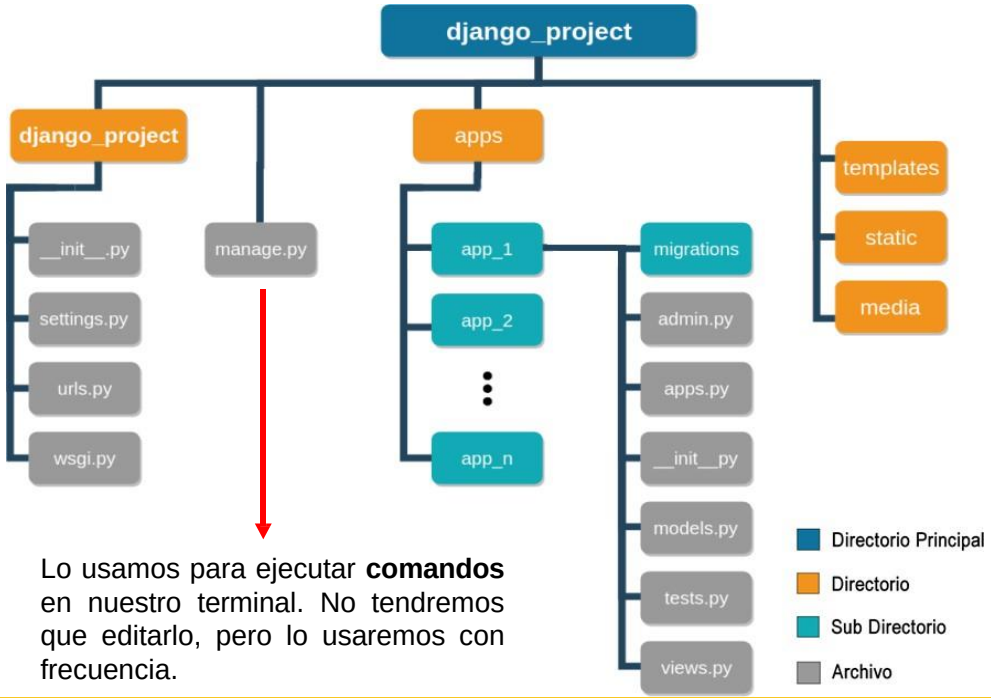
Vista: la capa de la lógica de negocios..

Template: (Plantilla), la capa de presentación.

Estructura de directorios Django

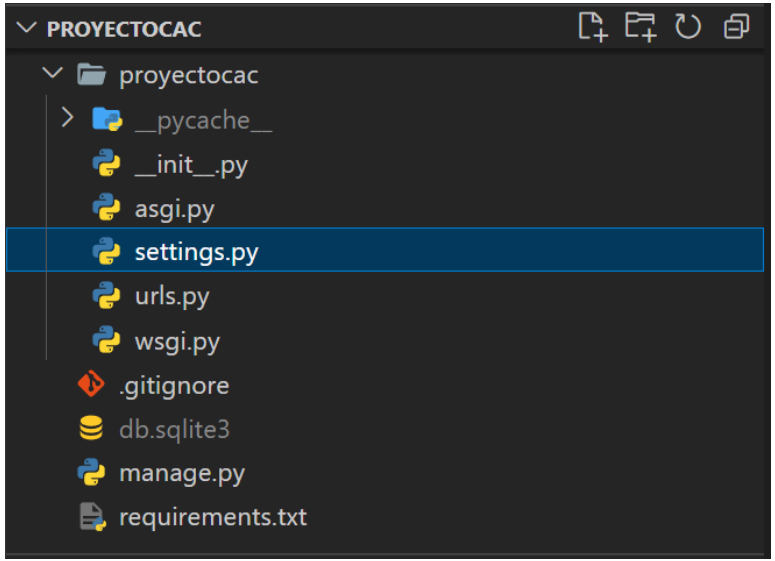
Contiene algunos **ajustes** de configuración importantes.
Es posible que deseemos cambiar algunos ajustes predeterminados de vez en cuando.

Contiene instrucciones sobre a dónde se debe dirigir a los usuarios después de navegar a una determinada **URL**.



Lo usamos para ejecutar **comandos** en nuestro terminal. No tendremos que editarlo, pero lo usaremos con frecuencia.

Estructura de directorios Django



manage.py: Una utilidad de la línea de comandos que le permite interactuar con este proyecto Django de diferentes formas.

__init__.py: Un archivo vacío que le indica a Python que este directorio debería ser considerado como un paquete Python.

settings.py: Ajustes/configuración para este proyecto Django.

urls.py: Las declaraciones URL para este proyecto Django.

wsgi.py: Un punto de entrada para que los servidores web compatibles con WSGI puedan servir su proyecto.

asgi.py: Un punto de entrada para que los servidores web compatibles con ASGI puedan servir su proyecto.

Instalación y configuración

Creación de entorno virtual

python –m venv mientornovirtual

Activación entorno virtual

source path\mientornovirutual\Scripts\activate

Instalación de Django

pip install Django==4.2.7

Verificación de Instalación de Django

```
python
>>> import django
>>> print(django.get_version())
```

Instalación y configuración

Creación del proyecto

django-admin startproject “nombreproyecto”

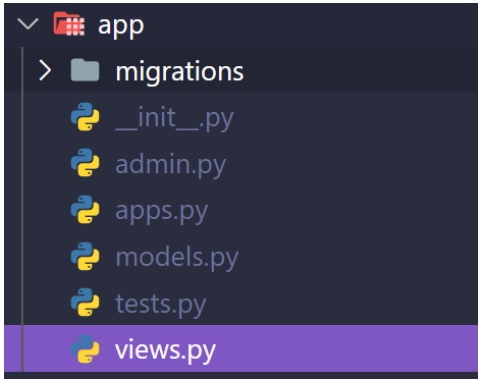
Verificación de creación del proyecto
Django provee un servidor web local para desarrollo, en el cual podremos probar lo que vamos modificando en el código de nuestro proyecto.

cd .\“nombreproyecto”
python manage.py runserver

Instalación y configuración

Creación de una aplicación

python manage.py startapp “app”



Registrar aplicación en settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app'
]
```

http://127.0.0.1:8000/

Instalación y configuración

Creación de archivo requirements.txt

pip freeze > requirements.txt

Este comando nos generará un archivo requirements.txt que contendrá el listado de todas las librerías y sus versiones instaladas en nuestro virtual env para el proyecto. Esto es útil para que posteriormente podamos instalarlas directamente por medio de este archivo en cualquier otro entorno virtual donde llevemos al proyecto.

pip install -r requirements.txt

Url dispatcher

Un esquema de URL limpio y elegante es un detalle importante en una aplicación web de alta calidad. Django te permite diseñar URL como quieras, sin limitaciones.

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

urlpatterns = [
    path('admin/', admin.site.urls),

]
```

Url dispatcher

Para cada aplicación creada es recomendable manejar un archivo urls.py. En el directorio de la app creada, creamos un archivo urls.py con el siguiente contenido:

```python
from django.urls import path, re_path, include
from . import views

urlpatterns = [
    path('', views.index, name='api_inicio'),
]
```

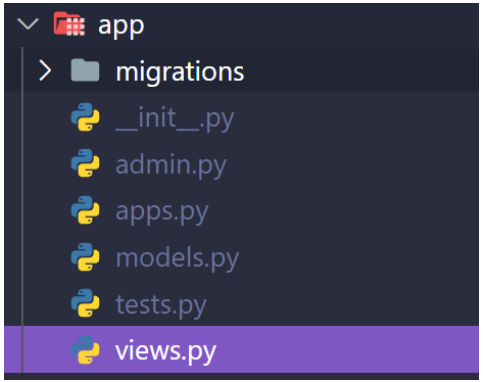
Por último, actualizamos el archivo urls.py del proyecto para que pueda buscar las rutas de la aplicación desde el archivo que acabamos de crear.

```python
from django.urls.conf import include

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

Nuestra primer vista

Para completar el ejemplo anterior, en el archivo views.py de la aplicación deberemos crear una nueva función index, que retorna una respuesta del tipo HTTP. Este método, permitirá al URL dispatcher detectar que método de la vista debe ejecutar cuando reciba un path específico, por ejemplo: http:\\127.0.0.1:8000\api




```python
from django.http import HttpResponse


# Create your views here.
def index(request):
    return HttpResponse("Hola gente")
```

Instalación y configuración – MySQL

Instalación librería para conectar el proyecto con MySQL
Se debe tener en cuenta que para instalar librerías se debe tener el entorno virtual activado, de otro modo la librería se instalará en el entorno global de Python.

pip install mysql

Verificación de conexión a Base de datos
En una primera instancia vamos a tener que crear una base de datos desde el cliente de base de datos MySQL que se este usando (Workbench, phpmyadmin,etc). Creamos por ejemplo una base de datos llamada: **db_cac_movies_dj**

Instalación y configuración – MySQL

Verificación de conexión a Base de datos

Una vez creada la base de datos, nos dirigimos al archivo settings.py del proyecto y modificamos las siguientes líneas. Donde 'pass' es la contraseña del usuario root de mysql.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'db_cac_movies_dj',
        'USER': 'root',
        'PASSWORD': 'pass',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}
```

Para corroborar que la conexión este bien configurada, ejecutaremos el siguiente comando. Este comando corre las migraciones de django, para crear tablas en la base de datos seleccionada a partir de los modelos creados.

python manage.py migrate

Instalación y configuración – Variables de entorno

Las variables de entorno se utilizan para almacenar los secretos de la aplicación y los datos de configuración, que son recuperados por tu aplicación en ejecución cuando se necesitan. El valor de estas variables puede proceder de diversas fuentes: archivos de texto, gestores secretos de terceros, scripts de llamada, etc.

Instalación librería

pip install django-environ

Siempre que se instala un nueva librería, recordar de actualizar el archivo requirements.txt

pip freeze > requirements.txt

ref: Variables de entorno

Instalación y configuración – Variables de entorno

Se debe crear un archivo .env en el mismo directorio donde se encuentra el archivo settings.py. Donde lo completaremos de la siguiente manera, la SECRET_KEY la obtenemos del archivo settings.py y el resto de los valores se corresponden a los datos de conexión a nuestra base de datos.

```
SECRET_KEY=django-insecure-+x82h-_3)2&--($_24yov@1+itld*@5%mllq@pj$=j$f*h5ah*
DATABASE_NAME=db_cac_movies_dj
DATABASE_USER=root
DATABASE_PASSWORD=pass
DATABASE_HOST=localhost
DATABASE_PORT=3306
```

Por último, modificaremos el archivo settings.py para poder tomar las variables de entorno y no dejar expuesta los datos de la secret_key y nuestra conexión a la base de datos

ref: Variables de entorno

Instalación y configuración – Variables de entorno

Agregamos estas líneas para que Django pueda cargar el archivo .evn

```
import os
import environ

env = environ.Env()
# reading .env file
environ.Env.read_env()
```

Reemplazamos el contenido, por la referencias a los nombres de las variables de entorno.

```
SECRET_KEY = env("SECRET_KEY")
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': env("DATABASE_NAME"),
        'USER': env("DATABASE_USER"),
        'PASSWORD': env("DATABASE_PASSWORD"),
        'HOST': env("DATABASE_HOST"),
        'PORT': env("DATABASE_PORT"),
    }
}
```

ref: Variables de entorno

No te olvides de completar la asistencia y consultar dudas

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

TODO EN EL AULA VIRTUAL