Instalación con Pipenv

- Creamos un directorio **cursodjango** para el proyecto.
- Lo abriremos en VSC para que se inicie la configuración.
- Abriremos la terminal e instalaremos **pipenv**:

```
pip install pipenv
```

- Creamos el entorno virtual instalando django directamente dentro del directorio del proyecto:

```
pipenv install django
```

Creando el proyecto

- Utilizaremos este comando para crear el proyecto:

```
pipenv run django-admin startproject tutorial
```

- Probaremos a poner el servidor en marcha:

```
cd tutorial
pipenv run python manage.py runserver
```

- Ya podremos acceder a la url y ver el proyecto:

```
http://127.0.0.1:8000/
```

- Crearemos un script en el **Pipfile** para simplificar la puesta en marcha:

```
Pipfile
[scripts]
server = "python manage.py runserver"
```

- Ahora podemos ejecutar el servidor con:

```
pipenv run server
```

Configuración básica

En esta lección vamos a aprender sobra la configuración del proyecto que se realiza en los ficheros del directorio con el mismo nombre que el proyecto:

- **__init__.py**: Sirve para inicializar un paquete en un directorio y también para ejecutar código durante su importarción. Desde Python 3.3 su uso no es obligatorio porque los paquetes se crean implícitamente.
- **settings.py**: Es el fichero principal de la configuración, volveremos a él después de ver los demás ficheros.

- **urls.py**: Aquí es donde se configuran las direcciones que escucharán peticiones en las URL del navegador para devolver las distintas páginas de la aplicación web.
- **wsgi.py**: Este último fichero contiene la configuración de la interfaz WSGI, una interfaz para realizar el despliegue del servidor en entornos de producción.

Bien, de vuelta al **settings.py** repasemos las variables de configuración. Si no os interesa profundizar podéis pasar de largo pero conocerlas os ayudará a entender cómo funciona Django:

- **BASE_DIR**: Esta variable genera la ruta al directorio del proyecto y se utiliza para generar rutas a otros ficheros del proyecto.
- **SECRET_KEY**: La clave secreta es una variable aleatoria y única generada automáticamente por Django que se utiliza principalmente para tareas criptográficas y generación de tokens.
- **DEBUG**: Cuando esta opción está activada se mostrará información de los errores como respuesta a las peticiones que fallan. Debido a que esta información es privada hay que desactivar el modo debug al publicar el proyecto en Internet.
- **ALLOWED_HOSTS**: Esta lista contiene las IP y dominios donde queremos permitir que Django se ejecute. Por ejemplo nuestro host local sería el '127.0.0.1' que se encuentra añadido por defecto.
- **INSTALLED_APPS**: Aquí tenemos otra lista importante que contiene las apps activadas en el proyecto. Las apps son como submódulos que controlan diferentes aspectos del proyecto y por defecto Django trae varias activadas para controlar funcionalidades como el panel de administrador, la autenticación de usuarios y las sesiones. Cuando nosotros hayamos creado nuestras propias apps o queramos utilizar apps externas creadas por la comunidad también tendremos que añadirlas a esta lista.
- **MIDDLEWARE**: Los middlewares son funciones de código que se añaden durante el procesamiento de las peticiones y respuestas a las URL. Django trae muchos por defecto aunque no todos son necesarios.
- **ROOT_URLCONF**: Esta variable contiene la ruta donde se encuentra el fichero con las URL, por

defecto **tutorial.urls** haciendo referencia al directorio **tutorial** y el fichero **urls**.

- **TEMPLATES**: Contiene una configuración completa para manejar los templates o plantillas de Django. Los templates son los ficheros HTML del proyecto y aquí se configura la forma de buscarlos, cargarlos y procesarlos.
- **WSGI_APPLICATION**: Otra variable que contiene la ruta a la variable **application** del fichero **wsgi.py** para usarla durante el despliegue.
- **DATABASES**: Aquí se define la configuración de acceso a la base de datos. Por defecto se utiliza **SQLite3** y no es necesario configurar nada porque se maneja todo en un fichero. SQLite está bien para el desarrollo y en proyectos pequeños pero en producción es recomendable usar una base de datos centralizada. Django tiene soporte para varias: PostgreSQL, MySQL, MariaDB y Oracle Database.
- **AUTH_PASSWORD_VALIDATORS**: Aqui se configuran distintas validaciones para las contraseñas de los usuarios, cosas como la longitud mínima, contraseñas demasiado comunes o numéricas y por tanto inseguras, etc.
- **LANGUAGE_CODE**: Una de las diferentes opciones de internacionalización de Django que sirven para configurar el idioma, la zona horaria del servidor para manejar fechas y horas, etc. Vamos a poner 'es' para que Django funcione en español.
- **STATIC_URL**: Finalmente una variable para controlar la ruta a la URL donde se irán a buscar los ficheros estáticos como CSS, JavaScript e imágenes.