

1. Instalación de Git

1. Windows:

- Descarga Git desde el sitio oficial: git-scm.com.
- Durante la instalación, selecciona las opciones predeterminadas o ajusta según tus preferencias.
- Asegúrate de incluir la opción "Git Bash" como terminal.

2. Linux (Ubuntu/Debian):

bash

Copy code

sudo apt update

sudo apt install git

3. MacOS:

- Instala Git usando Homebrew:

bash

Copy code

brew install git

2. Verifica la instalación

Abre una terminal y escribe:

bash

Copy code

git --version

Deberías ver la versión instalada.

3. Configuración inicial de Git

Configura tu nombre y correo electrónico globales (estos se usarán para firmar tus commits):

bash

Copy code

```
git config --global user.name "Tu Nombre"
```

```
git config --global user.email "tuemail@example.com"
```

4. Configurar el editor predeterminado

Por defecto, Git usa vim. Si prefieres otro editor como VS Code:

```
bash
```

Copy code

```
git config --global core.editor "code --wait"
```

5. Configuración de las claves SSH (Opcional pero recomendado)

Esto permite conectarte a repositorios remotos (por ejemplo, GitHub) sin necesidad de introducir usuario/contraseña constantemente.

1. Generar una clave SSH:

```
bash
```

Copy code

```
ssh-keygen -t ed25519 -C "tuemail@example.com"
```

(Presiona Enter para usar la ubicación predeterminada).

2. Agregar la clave SSH al agente:

```
bash
```

Copy code

```
eval "$(ssh-agent -s)"
```

```
ssh-add ~/.ssh/id_ed25519
```

3. Copiar la clave pública:

```
bash
```

Copy code

```
cat ~/.ssh/id_ed25519.pub
```

Copia el contenido y agrégalo en tu cuenta de GitHub/GitLab bajo "SSH Keys".

6. Inicializar un repositorio

1. Ve al directorio de tu proyecto:

bash

Copy code

cd /ruta/del/proyecto

2. Inicializa Git:

bash

Copy code

git init

3. Añade un archivo .gitignore para excluir archivos/carpetas no deseados:

bash

Copy code

touch .gitignore

4. Añade los archivos y haz tu primer commit:

bash

Copy code

git add .

git commit -m "Primer commit"

7. Conexión con un repositorio remoto

1. Crea un repositorio en GitHub/GitLab/Bitbucket.

2. Vincula tu repositorio local:

bash

Copy code

git remote add origin https://github.com/usuario/repositorio.git

3. Sube los cambios:

bash

Copy code

```
git branch -M main
```

```
git push -u origin main
```

8. Instalación de librerías útiles (Opcional)

Estas herramientas pueden mejorar tu experiencia con Git:

- **Git GUI:** Para un uso gráfico de Git.
- **GitLens (para VS Code):** Extensión para rastrear cambios y autores en VS Code.
- **Tig:** Visualizador de historial Git en la terminal.

```
bash
```

Copy code

```
sudo apt install tig # Para Linux
```

9. Práctica básica

Algunas operaciones básicas que deberías dominar:

- Ver el estado del repositorio:

```
bash
```

Copy code

```
git status
```

- Ver historial de commits:

```
bash
```

Copy code

```
git log
```

- Crear una nueva rama:

```
bash
```

Copy code

```
git checkout -b nombre-de-la-rama
```

- Fusionar ramas:

bash

Copy code

git merge nombre-de-la-rama

Guía paso a paso para configurar un proyecto Django con su entorno virtual

1. Instalar Python

1. Asegúrate de tener Python instalado (versión 3.8 o superior).
 - **Windows:** Descarga desde python.org.
 - Durante la instalación, selecciona "Add Python to PATH".
 - **Linux/MacOS:** Verifica si ya está instalado:

bash

Copy code

python3 --version

Si no, instálalo:

bash

Copy code

sudo apt install python3 python3-pip

2. Instala pip (el gestor de paquetes de Python) si no viene incluido:

bash

Copy code

python3 -m ensurepip --upgrade

2. Crear un entorno virtual

Un entorno virtual permite mantener las dependencias de un proyecto aisladas.

1. Crea el entorno virtual:

bash

Copy code

```
python3 -m venv venv
```

Esto creará una carpeta llamada venv en el directorio actual.

2. Activa el entorno virtual:

- **Windows:**

```
bash
```

Copy code

```
venv\Scripts\activate
```

- **Linux/MacOS:**

```
bash
```

Copy code

```
source venv/bin/activate
```

3. Verifica que el entorno virtual está activado:

```
bash
```

Copy code

```
which python
```

El resultado debe apuntar al directorio dentro de venv.

3. Instalar Django

Con el entorno virtual activado, instala Django:

```
bash
```

Copy code

```
pip install django
```

4. Crear un proyecto Django

1. Crea un nuevo proyecto llamado, por ejemplo, mi_proyecto:

```
bash
```

Copy code

```
django-admin startproject mi_proyecto
```

2. Esto generará una estructura básica:

bash

Copy code

mi_proyecto/

├─ manage.py # Script para ejecutar comandos de Django

└─ mi_proyecto/

├─ __init__.py # Indica que es un paquete Python

├─ asgi.py # Configuración para ASGI (opcional)

├─ settings.py # Configuración global del proyecto

├─ urls.py # Rutas del proyecto

└─ wsgi.py # Configuración para WSGI

5. Ejecutar el servidor de desarrollo

1. Ve al directorio del proyecto:

bash

Copy code

cd mi_proyecto

2. Inicia el servidor:

bash

Copy code

python manage.py runserver

3. Abre un navegador y visita: <http://127.0.0.1:8000/>. Deberías ver la página de bienvenida de Django.
-

6. Crear una aplicación Django

Las aplicaciones son componentes reutilizables dentro de un proyecto.

1. Crea una nueva aplicación llamada, por ejemplo, mi_app:

bash

Copy code

```
python manage.py startapp mi_app
```

2. Esto generará una nueva carpeta:

```
bash
```

Copy code

```
mi_app/
```

```
├── admin.py    # Configuración del panel de administración
├── apps.py     # Configuración de la app
├── migrations/ # Archivos de migraciones para la base de datos
├── models.py   # Definición de modelos de datos
├── tests.py    # Pruebas unitarias
└── views.py    # Lógica de las vistas
```

3. Registra la aplicación en settings.py: Abre mi_proyecto/settings.py y añade 'mi_app', a la lista de INSTALLED_APPS.

7. Migraciones de base de datos

Django usa SQLite como base de datos predeterminada.

1. Crea las tablas iniciales:

```
bash
```

Copy code

```
python manage.py migrate
```

2. Crea un superusuario para acceder al panel de administración:

```
bash
```

Copy code

```
python manage.py createsuperuser
```

8. Buenas prácticas de estructura

Tu proyecto Django puede crecer y necesitar una estructura más organizada:

csharp

Copy code

mi_proyecto/

```
├── mi_proyecto/      # Configuración del proyecto
|   ├── __init__.py
|   ├── settings.py
|   ├── urls.py
|   └── wsgi.py
├── mi_app/           # Aplicación principal
|   ├── migrations/
|   ├── templates/    # Archivos HTML
|   ├── static/       # Archivos estáticos (CSS, JS)
|   └── views.py
├── manage.py
└── venv/             # Entorno virtual
```

9. Instalar dependencias comunes

1. Instala django-environ para manejar variables de entorno:

bash

Copy code

pip install django-environ

2. Instala herramientas de depuración:

bash

Copy code

pip install django-debug-toolbar

10. Documenta y exporta dependencias

1. Genera un archivo requirements.txt:

bash

Copy code

```
pip freeze > requirements.txt
```

2. En otro entorno o máquina, instala estas dependencias:

bash

Copy code

```
pip install -r requirements.txt
```

Guía paso a paso para configurar un proyecto Flutter

1. Instalar Flutter

1. Descarga Flutter:

- Visita flutter.dev y descarga el SDK correspondiente a tu sistema operativo.

2. Extrae el archivo descargado:

- **Windows:** Extrae el contenido en una carpeta, por ejemplo, C:\flutter.
- **MacOS/Linux:** Extrae el archivo en tu directorio de usuario o en /opt.

3. Agrega Flutter al PATH:

- **Windows:**
 1. Abre "Editar las variables de entorno del sistema".
 2. Agrega la ruta de la carpeta bin del SDK (por ejemplo, C:\flutter\bin).
- **MacOS/Linux:** Edita el archivo ~/.bashrc, ~/.zshrc o ~/.bash_profile y agrega:

bash

Copy code

```
export PATH="$PATH:/ruta/a/flutter/bin"
```

Luego, recarga el terminal:

bash

Copy code

```
source ~/.bashrc
```

4. **Verifica la instalación:** En la terminal, escribe:

```
bash
```

Copy code

```
flutter doctor
```

Esto mostrará un resumen del estado de la instalación. Resuelve cualquier error o advertencia.

2. Configurar un editor de texto o IDE

1. Instala Visual Studio Code o Android Studio:

- Visual Studio Code: [Descargar](#).
- Android Studio: [Descargar](#).

2. Instala extensiones para Flutter:

- En Visual Studio Code, instala las extensiones:
 - Flutter
 - Dart
- En Android Studio, ve a Preferences > Plugins y busca "Flutter". Instala el complemento, que también instalará Dart.

3. Configuración de emuladores o dispositivos físicos

1. Android Studio:

- Abre Android Studio > Configure > AVD Manager.
- Crea un nuevo dispositivo virtual (elige un teléfono con Google APIs).
- Inicia el emulador.

2. Dispositivo físico (Android):

- Activa la depuración USB en tu teléfono (en las opciones de desarrollador).
- Conecta tu teléfono a la computadora.

- Verifica que el dispositivo sea detectado:

bash

Copy code

flutter devices

3. Dispositivo físico (iOS):

- Conecta tu iPhone o iPad a la computadora.
- Abre Xcode y acepta los permisos necesarios.
- Verifica que el dispositivo sea detectado:

bash

Copy code

flutter devices

4. Crear un proyecto Flutter

1. Crea un nuevo proyecto:

bash

Copy code

flutter create nombre_proyecto

Esto generará la estructura básica de Flutter:

bash

Copy code

nombre_proyecto/

```
├── android/      # Configuración específica de Android
├── ios/          # Configuración específica de iOS
├── lib/          # Código principal (Dart)
|   └── main.dart # Punto de entrada de la app
├── test/         # Pruebas unitarias
├── pubspec.yaml  # Archivo de configuración del proyecto
└── README.md
```

2. Ve al directorio del proyecto:

bash

Copy code

cd nombre_proyecto

3. Abre el proyecto en tu editor o IDE.
-

5. Ejecutar la aplicación

1. Asegúrate de tener un dispositivo conectado (físico o emulador).
2. Inicia la aplicación:

bash

Copy code

flutter run

6. Configuración del archivo pubspec.yaml

1. Este archivo administra las dependencias de tu proyecto.
 - Por ejemplo, para agregar una dependencia como http:

yaml

Copy code

dependencies:

flutter:

 sdk: flutter

 http: ^0.15.0

- Luego, ejecuta:

bash

Copy code

flutter pub get

2. Puedes agregar assets, como imágenes:

yaml

Copy code

flutter:

assets:

- assets/images/logo.png

7. Carpetas principales del proyecto

- **lib/main.dart:** Punto de entrada principal de tu aplicación.
 - **lib/:** Contiene el resto de tu código, como widgets y lógica de negocio.
 - **pubspec.yaml:** Define las dependencias y configuraciones del proyecto.
 - **android/ y ios/:** Configuraciones específicas para Android e iOS.
-

8. Ejecución y pruebas

1. **Hot Reload:** Para recargar cambios rápidamente mientras la app está en ejecución:

bash

Copy code

r # En la terminal donde ejecutaste `flutter run`

2. **Pruebas unitarias:**

- Añade pruebas en la carpeta test/ y ejecuta:

bash

Copy code

flutter test

9. Depuración

1. Usa el comando:

bash

Copy code

flutter doctor

Para resolver problemas con dependencias o configuraciones.

2. Abre herramientas de depuración en VS Code o Android Studio para inspeccionar widgets, errores y trazas.

10. Construir para producción

1. Android APK:

bash

Copy code

flutter build apk

2. iOS App:

bash

Copy code

flutter build ios

(Requiere Xcode configurado).