––––––

**C.F.G.S DESARROLLO**

**DE APLICACIONES WEB**

MANUAL CONTROL DE VERSIONES EN GITHUB

**CANCIONERO DIGITAL SALESIANO**

Jerome Gamboa y Marco Batista

**COLEGIO NUESTRA SEÑORA DEL PILAR**

**ÍNDICE**

índice

[I. Manual para Trabajar en Equipo con GitHub Desktop en un Proyecto Django 2](#_Toc192758879)

[1. **Objetivo**: 2](#_Toc192758880)

[**Paso 1: Configuración Inicial del Repositorio** 2](#_Toc192758881)

[**Paso 2: Clonar el Repositorio en GitHub Desktop** 2](#_Toc192758882)

[**Paso 3: Crear y Trabajar con Ramas** 3](#_Toc192758883)

[**Paso 4: Mantenerse Sincronizados** 3](#_Toc192758884)

[**Paso 5: Trabajar en la Rama y Subir Cambios** 3](#_Toc192758885)

[**Paso 6: Crear un Pull Request (PR)** 4](#_Toc192758886)

[**Paso 7: Resolver Conflictos (Si Ocurren)** 4](#_Toc192758887)

[**Paso 8: Configuración de .gitignore y Dependencias** 4](#_Toc192758888)

[**2.** **Resumen del Flujo de Trabajo** 6](#_Toc192758889)

# Manual para Trabajar en Equipo con GitHub Desktop en un Proyecto Django

## **Objetivo**:

Este manual te guiará en el uso de **GitHub Desktop** para colaborar en un proyecto Django con otra persona. Cada miembro del equipo trabajará en su propia rama, sincronizando cambios con la rama principal (main), evitando conflictos y asegurando un flujo de trabajo eficiente.

### **Paso 1: Configuración Inicial del Repositorio**

**1. Crear un Repositorio en GitHub**

1. Inicia sesión en [GitHub](https://github.com/).
2. Crea un nuevo repositorio y ponle un nombr.
3. Asegúrate de marcar la opción **"Incluir .gitignore"** y selecciona **"Django"**.
4. Agrega un archivo README.md si deseas documentar el proyecto.
5. Copia la URL del repositorio, la necesitarás más adelante.

**2. Agregar a tu compañero como colaborador**

1. Ve a tu repositorio en GitHub.
2. Dirígete a **Settings > Collaborators & teams**.
3. En **Collaborators**, haz clic en **Add people**.
4. Ingresa el nombre de usuario de tu compañero y agrégalo.
5. Tu compañero recibirá una invitación para unirse al repositorio.

### **Paso 2: Clonar el Repositorio en GitHub Desktop**

**1. Descargar e instalar GitHub Desktop**

**2. Clonar el repositorio**

1. Abre **GitHub Desktop**.
2. Haz clic en **File > Clone repository**.
3. En la pestaña **GitHub.com**, selecciona tu repositorio o ingresa la URL copiada anteriormente.
4. Selecciona la carpeta donde deseas guardar el proyecto y haz clic en **Clone**.

### **Paso 3: Crear y Trabajar con Ramas**

Cada miembro del equipo debe crear su propia rama para trabajar en su parte del código sin interferir con la rama principal.

**1. Crear una rama en GitHub Desktop**

1. Ve a **Branch > New branch**.
2. Ponle un nombre a la rama.
3. Selecciona **Create branch**.
4. En la parte superior, asegúrate de que la rama activa sea la que creaste.

**2. Subir la nueva rama al repositorio remoto**

1. Ve a **Repository > Push origin**.
2. Esto enviará tu nueva rama al repositorio en GitHub.

### **Paso 4: Mantenerse Sincronizados**

**1. Obtener los últimos cambios de la rama principal (main)**

Si otros han realizado cambios en la rama main, debes actualizar tu copia local.

1. Cambia a la rama main.
2. Haz clic en **Fetch origin** y luego en **Pull origin**.
3. Regresa a tu rama de trabajo y haz clic en **Merge main into [La rama del main]** para actualizarla con los cambios recientes (Te dará la opción en tu rama para hacer un push para tenerlo también en tu rama).

### **Paso 5: Trabajar en la Rama y Subir Cambios**

**1. Hacer cambios en el código**

* Realiza las modificaciones necesarias en tu código.

**2. Confirmar y subir cambios**

1. Ve a **GitHub Desktop**.
2. Verás los archivos modificados en la ventana principal.
3. Escribe un mensaje de confirmación en **Summary**.
4. Haz clic en **Commit to [tu-rama]**.
5. Luego, haz clic en **Push origin** para subir los cambios a GitHub.

### **Paso 6: Crear un Pull Request (PR)**

Cuando termines tu trabajo en tu rama y quieras unirlo a main:

1. Abre GitHub Desktop y haz clic en **Branch > Create pull request**.
2. Se abrirá GitHub en el navegador.
3. Agrega una descripción de los cambios realizados.
4. Haz clic en **Create Pull Request**.
5. Tu compañero puede revisar los cambios y aprobar la fusión o tu con esta modificacion.

### **Paso 7: Resolver Conflictos (Si Ocurren)**

Si hay conflictos al fusionar tu rama con main, GitHub te avisará.

**1. Resolver conflictos en GitHub Desktop**

1. Abre GitHub Desktop y cambia a la rama en conflicto.
2. GitHub marcará los archivos en conflicto.
3. Edita manualmente los archivos en un editor de código para resolver los conflictos.

¡Tener en cuenta para quitar los conflictos escoger que versión quieras de las que te ofrecen separados por >>>>>>> , =======!

1. Guarda los cambios y confirma el commit.
2. Sube la solución a GitHub con **Push origin**.

### **Paso 8: Configuración de .gitignore y Dependencias**

**1. Configurar .gitignore**

Para evitar subir archivos innecesarios, asegúrate de que .gitignore tenga lo siguiente:

# Django

\*.log

\*.pot

\*.pyc

\_\_pycache\_\_/

# Entornos virtuales

venv/

env/

# Archivos locales

\*.sqlite3

# Archivos de configuración de IDEs

.vscode/

.idea/

# Variables de entorno

\*.env

**2. Generar requirements.txt**

Después de instalar las dependencias necesarias en Django, ejecuta:

1. pip freeze > requirements.txt

Para instalar las dependencias en otra máquina:

1. pip install -r requirements.txt

## **Resumen del Flujo de Trabajo**

1. **Agregar colaboradores** → Asegurar que ambos tienen acceso al repositorio.
2. **Crear una rama por usuario** → Cada persona trabaja en su propia rama (rama-usuario1, rama-usuario2).
3. **Trabajar y hacer commits** → Confirmar los cambios y subirlos a GitHub.
4. **Sincronización** → Obtener cambios recientes de main antes de hacer nuevas modificaciones.
5. **Pull Request (PR)** → Solicitar la fusión de la rama con main.
6. **Revisión y resolución de conflictos** → Resolver cualquier conflicto antes de fusionar.