––––––

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

**DE APLICACIONES WEB**

Manual para Desplegar una Aplicación Django con PostgreSQL

**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](#_Toc200617148)

[1. Objetivo: 2](#_Toc200617149)

[Este manual tiene como finalidad guiar paso a paso el proceso de despliegue de una aplicación Django que utiliza PostgreSQL como sistema de gestión de bases de datos. Está enfocado en una configuración inicial para entornos gratuitos o de pruebas, como **Render**, **Railway**, **Fly.io** o **Heroku (legacy)**, y es extensible a servidores VPS o alojamiento profesional si se desea hacer el despliegue definitivo en el futuro. 2](#_Toc200617150)

[2. Requisitos previos 2](#_Toc200617151)

[3. Configuración del Proyecto para Producción 2](#_Toc200617152)

[3.1. Instalar dependencias necesarias 2](#_Toc200617153)

[3.2. Configurar el archivo settings.py 3](#_Toc200617154)

[3.3. Crear el archivo .env 3](#_Toc200617155)

[4. Archivos de configuración para el despliegue 3](#_Toc200617156)

[4.1. Procfile (necesario en Heroku o Render) 3](#_Toc200617157)

[4.2. runtime.txt (si usas una versión concreta de Python) 3](#_Toc200617158)

[5. Paso a paso según la plataforma 4](#_Toc200617159)

[Opción A: Render (más recomendada actualmente) 4](#_Toc200617160)

[Opción B: Railway 4](#_Toc200617161)

[6. Migraciones y archivos estáticos 4](#_Toc200617162)

[7. Verificación 4](#_Toc200617163)

[8. Siguientes pasos 5](#_Toc200617164)

[9. Conclusión 5](#_Toc200617165)

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

## Objetivo:

### Este manual tiene como finalidad guiar paso a paso el proceso de despliegue de una aplicación Django que utiliza PostgreSQL como sistema de gestión de bases de datos. Está enfocado en una configuración inicial para entornos gratuitos o de pruebas, como **Render**, **Railway**, **Fly.io** o **Heroku (legacy)**, y es extensible a servidores VPS o alojamiento profesional si se desea hacer el despliegue definitivo en el futuro.

## Requisitos previos

Antes de iniciar el despliegue, asegúrate de tener:

* Una cuenta en una plataforma de despliegue (por ejemplo: Render, Railway o Fly.io).
* Repositorio del proyecto en GitHub.
* Archivo requirements.txt actualizado.
* Archivo Procfile (en caso de usar plataformas como Heroku o Render).
* Configuración de variables de entorno (.env).
* Uso de whitenoise para servir archivos estáticos si no hay un bucket externo.
* Base de datos configurada para PostgreSQL.
* DEBUG=False y ALLOWED\_HOSTS correctamente definidos en settings.py.

## Configuración del Proyecto para Producción

## Instalar dependencias necesarias

En tu entorno local, asegúrate de tener:

*pip install gunicorn*

*pip install whitenoise*

*pip install psycopg2-binary*

Luego actualiza tu archivo requirements.txt:

*pip freeze > requirements.txt*

## Configurar el archivo settings.py

Archivos estáticos (Añade o asegúrate de incluir)

*STATIC\_URL = '/static/'*

*STATIC\_ROOT = BASE\_DIR / 'staticfiles'*

*STATICFILES\_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'*

*MIDDLEWARE.insert(1, 'whitenoise.middleware.WhiteNoiseMiddleware')*

Seguridad para producción

*DEBUG = False*

*ALLOWED\_HOSTS = ['tu-dominio.render.com', '127.0.0.1'] # Cambiar según la plataforma*

*CSRF\_TRUSTED\_ORIGINS = ['https://tu-dominio.render.com']*

Configuración de la base de datos PostgreSQL

En lugar de usar SQLite, cambia tu base de datos en settings.py por:

*import dj\_database\_url*

*DATABASES = {*

*'default': dj\_database\_url.config(default=os.environ.get('DATABASE\_URL'))*

*}*

## Crear el archivo .env

Ejemplo:

*SECRET\_KEY=tu\_clave\_secreta*

*DEBUG=False*

*DATABASE\_URL=postgres://usuario:contraseña@host:puerto/basededatos*

No olvides añadir .env a tu .gitignore.

## Archivos de configuración para el despliegue

## Procfile (necesario en Heroku o Render)

web: gunicorn nombre\_proyecto.wsgi:application

## runtime.txt (si usas una versión concreta de Python)

## Paso a paso según la plataforma

### Opción A: Render (más recomendada actualmente)

1. Entra en https://render.com y crea una cuenta.
2. Crea un nuevo servicio web → conecta tu repositorio de GitHub.
3. En el campo Build Command, pon:
   1. pip install -r requirements.txt && python manage.py collectstatic --noinput && python manage.py migrate
4. En Start Command, pon: *gunicorn nombre\_proyecto.wsgi:application*
5. En Environment, selecciona "Environment Variables" y añade:
   1. SECRET\_KEY
   2. DEBUG=False
   3. DATABASE\_URL (Render puede crear una base de datos PostgreSQL automáticamente)
   4. Otros valores necesarios para tu configuración.

Render detectará el servicio y lo desplegará automáticamente.

### Opción B: Railway

1. Entra en https://railway.app y crea una cuenta.
2. Crea un nuevo proyecto y conecta tu repositorio de GitHub.
3. Ve a la pestaña Plugins y añade PostgreSQL.
4. Railway generará automáticamente una DATABASE\_URL que puedes usar como variable de entorno.
5. En Settings > Variables, añade también SECRET\_KEY, DEBUG, etc.
6. Railway se encargará del build y despliegue automáticamente.

## Migraciones y archivos estáticos

Después del despliegue, asegúrate de ejecutar:

*python manage.py migrate*

*python manage.py collectstatic --noinput*

Algunas plataformas como Render permiten configurar esto automáticamente en el Build Command.

## Verificación

1. Accede a la URL que te ofrece la plataforma (por ejemplo: <https://miapp.onrender.com>).
2. Comprueba que:
   1. El sitio carga sin errores.
   2. La base de datos está funcionando (puedes registrarte, iniciar sesión, etc.).
   3. Los archivos estáticos se cargan correctamente (CSS, JS).
3. En caso de error 500, revisa los logs desde la plataforma.

## Siguientes pasos

* Conecta un dominio personalizado (opcional).
* Añade certificado SSL (la mayoría lo hace automáticamente).
* Realiza pruebas con varios usuarios.
* Configura un sistema de backups si el uso se vuelve intensivo.

## Conclusión

El despliegue de aplicaciones Django con PostgreSQL puede presentar ciertas dificultades, sobre todo en entornos gratuitos. Sin embargo, plataformas modernas como Render o Railway han simplificado este proceso significativamente. Aunque este proyecto no se desplegará de momento por decisión estratégica, este manual proporciona una base sólida para que el despliegue pueda realizarse en cualquier momento futuro.