**Flujo de Migración de Bases de Datos con Git, GitHub y Flyway**

Contenido

[**introducción:** 3](#_Toc190874452)

[**Instalación git-githubDesktop** 3](#_Toc190874453)

[**Descarga e instalación de Git** 3](#_Toc190874454)

[**Descarga e instalación de GitHubDesktop** 5](#_Toc190874455)

[**Clonar repositorio github** 5](#_Toc190874456)

[**Clonar en GIT** 6](#_Toc190874457)

[**Clonar en GitHub Desktop** 7](#_Toc190874458)

[**Flujo de trabajo migraciones flyway** 8](#_Toc190874459)

[**Generalidades de Flyway** 8](#_Toc190874460)

[**Vinculación de proyecto a Flyway:** 9](#_Toc190874461)

[**Versión de migraciones con GitHub y flyway:** 10](#_Toc190874462)

[**Enviar cambios a la rama Dev -** **Git** 12](#_Toc190874463)

[**Enviar cambios a la rama Dev –** **Github Desktop** 14](#_Toc190874464)

# **introducción:**

Este manual aborda los aspectos clave del flujo de trabajo necesario para garantizar un uso adecuado de GitHub en la migración y versiona miento de bases de datos. Se explicará cómo lograrlo tanto desde la terminal Bash integrada de Git como a través de la aplicación GitHub Desktop.

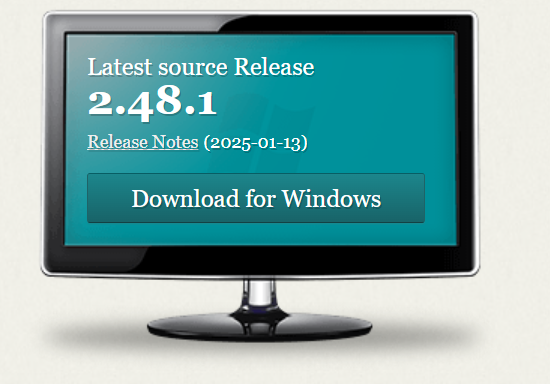
# **Instalación git-githubDesktop**

## **Descarga e instalación de Git**

Para obtener el controlador de versiones Git, es necesario acceder al sitio oficial a través del siguiente enlace:

🔗 [GIT](https://git-scm.com/)

Una vez en la página, haz clic en el botón Download y selecciona la versión correspondiente a tu sistema operativo (Windows, macOS o Linux).



La instalación es sencilla y no requiere modificar la mayoría de las opciones predeterminadas. Sin embargo, es importante asegurarse de seleccionar las siguientes opciones durante la configuración:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.✅ "Open Git Bash Here"  
✅ "Open Git GUI Here"

Estas opciones permiten abrir la terminal integrada de Git (Git Bash) y la interfaz gráfica (Git GUI) directamente desde el menú contextual de Windows, lo que facilita el acceso y uso de Git en el día a día.

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.**Al crear una nueva rama al iniciar un repositorio, asegúrate de seleccionar la opción **"Override the default branch name for new repositories"** y establecerla en **"main".** Esto se debe a que, en los nuevos repositorios y según el estándar actual de Git, la rama principal ya no se denomina “master”, sino “main”.

## **Descarga e instalación de GitHubDesktop**

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.Para descargar GitHub Desktop, puedes hacerlo en el siguiente enlace:  
🔗 [GithubDesktop](https://github.com/apps/desktop?ref_cta=download+desktop&ref_loc=installing+github+desktop&ref_page=docs)

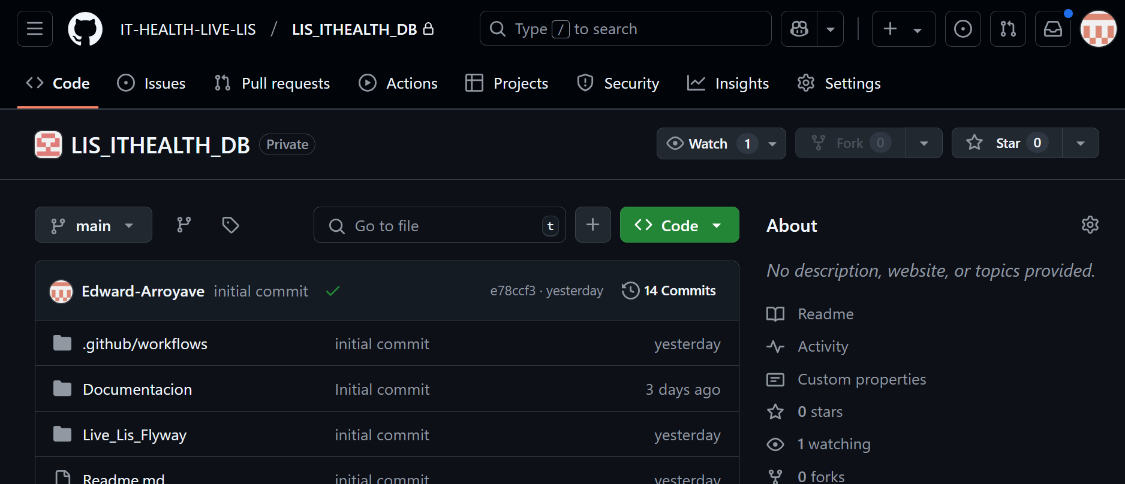
Una vez en la página, haz clic en el botón "Download for Windows" o "Download for macOS", según tu sistema operativo.

# **Clonar repositorio github**

En este ejemplo se explicará cómo clonar un repositorio utilizando la URL HTTPS, ya que es el método más accesible y seguro para la mayoría de los usuarios.

El protocolo HTTPS es preferido en entornos donde no se requiere autenticación con claves SSH, ya que permite clonar repositorios de manera sencilla utilizando credenciales de GitHub (usuario y token de acceso).

Para comenzar, accede al repositorio en GitHub y haz clic en el botón Code.

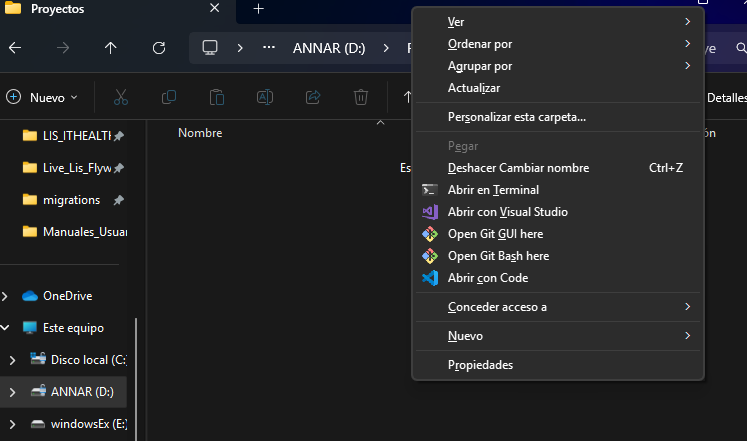


Interfaz de usuario gráfica, Texto, Sitio web

El contenido generado por IA puede ser incorrecto.Al hacer clic en el botón Code, se desplegarán varias opciones de clonación. Seleccionaremos la opción HTTPS y copiaremos la URL proporcionada.

## **Clonar en GIT**

Teniendo la URL del repositorio, dirígete a la carpeta donde deseas almacenar tu proyecto. Una vez dentro, haz clic derecho y selecciona "Git Bash Here" para abrir la terminal en esa ubicación.



Esto abrirá Git **Bash**, donde podremos ejecutar el siguiente comando:

**“git clone URL”**

Texto

El contenido generado por IA puede ser incorrecto.

## **Clonar en GitHub Desktop**

GitHub Desktop, al ser una interfaz gráfica, es más interactivo e intuitivo.

Para clonar el repositorio, primero debemos obtener la URL mencionada anteriormente.

Si formamos parte de una organización en GitHub Desktop, la aplicación puede mostrar directamente los repositorios disponibles para clonar.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

De no ver el repositorio que deseas en tus repositorios puedes clonarlo de la siguiente manera.

Debes presionar el botón clonar repositorio de internet

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Luego, selecciona la opción URL, agrega la URL copiada previamente e introdúcela en el campo correspondiente. A continuación, elige la ubicación donde deseas clonar la carpeta.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

# **Flujo de trabajo migraciones flyway**

## **Generalidades de Flyway**

Para iniciar el proceso de migraciones con **Flyway**, es fundamental comprender cómo deben nombrarse los archivos para garantizar una correcta interpretación y ejecución.

**Tipos de migraciones en Flyway**

1. **Migraciones con versiones**  
   Estas migraciones llevan la base de datos desde una versión anterior hasta la versión especificada en el archivo. Su nombre debe comenzar con el prefijo **V**.

**Ejemplo: V1.0.0\_\_Crear\_tabla\_usuarios.sql**

1. **Migraciones con versiones de deshacer** (*disponible solo en la versión Teams*)  
   Permiten revertir la base de datos desde la versión especificada en el archivo hasta la versión anterior. Estos archivos llevan el prefijo **U**.

**Ejemplo: U1.0.0\_\_Eliminar\_tabla\_usuarios.sql**

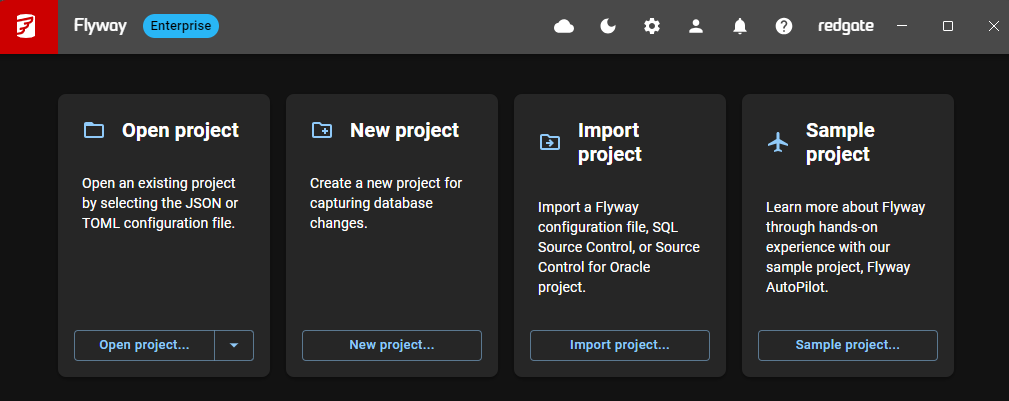
1. **Migraciones repetibles**
   * No están asociadas a una versión específica, sino a una descripción y una suma de comprobación. Su nombre debe comenzar con el prefijo **R**.
   * Se ejecutan nuevamente cada vez que su contenido cambia, ya que Flyway detecta la modificación y vuelve a aplicarlas.
   * Si hay varios archivos **R**, se ejecutan en orden ASCII según sus nombres.
   * Estos scripts deben ser **idempotentes**, lo que significa que pueden ejecutarse varias veces sin causar errores. Para evitar problemas, se recomienda usar declaraciones como **CREATE OR ALTER** o **CREATE OR REPLACE**, especialmente en vistas, procedimientos almacenados, funciones y paquetes. También son útiles para inserciones de datos en masa.
   * Siempre se aplican en último lugar, después de todas las migraciones versionadas.

Ejemplo: R\_\_Vista\_usuarios\_activos.sql

Con esta estructura clara, Flyway garantiza un control eficiente y seguro del versionamiento de bases de datos.

## **Vinculación de proyecto a Flyway:**

Luego de clonar el repositorio, abrimos la aplicación de **Flyway** y vinculamos nuestro proyecto presionando el botón **"Open Project"**.



Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.Seleccionaremos nuestro repositorio y elegiremos el archivo **flyway.toml**.

AL vincular el proyecto se nos abrirá el siguiente menú en el cuál podemos ver las versiones que se encuentren pendientes.

**Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.**

## **Versión de migraciones con GitHub y flyway:**

Con **GitHub**, controlaremos los entornos mediante ramas individuales. Por ello, es importante validar en qué rama te encuentras. Lo más común es que los cambios se realicen en la rama **Dev** y que los archivos de migración se vayan moviendo progresivamente a cada entorno.

Cada entorno se gestionará con las siguientes ramas:

* **Dev**
* **Test**
* **Demo**
* **Main** → Rama de producción

Para validar en que rama te encuentras puedes hacerlo en la terminal usando el comando “git status” o visualizando en final del encabezado de la terminal.  
Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

En **GitHub Desktop**, puedes verificar la rama actual en la esquina superior izquierda, dentro del menú desplegable **"Current Branch"**.  
Interfaz de usuario gráfica

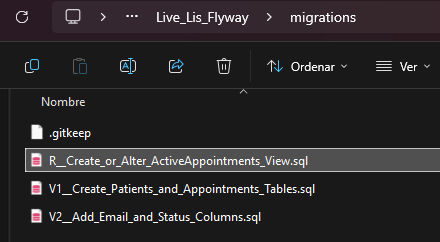
El contenido generado por IA puede ser incorrecto.

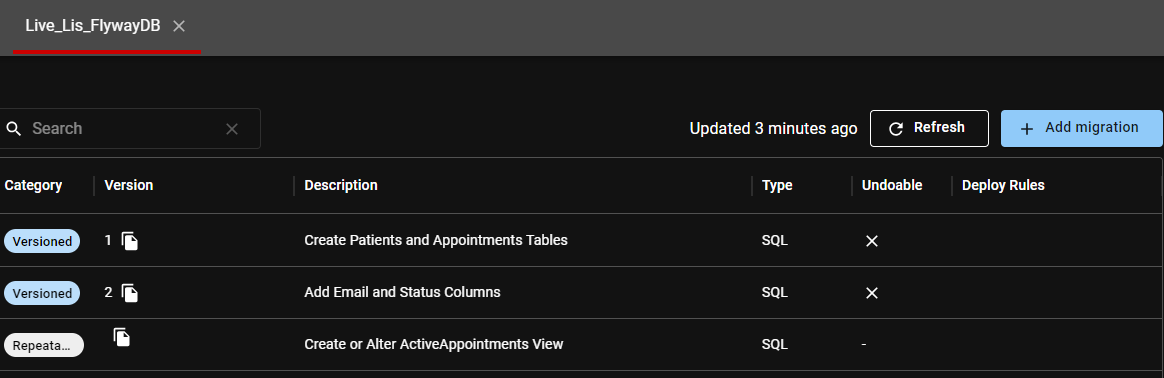
Estando en nuestro repositorio, vamos a la carpeta del proyecto donde se encuentra la carpeta de migración. Esta carpeta contendrá las migraciones que se enviarán a los diferentes entornos.

Texto

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Después de verificar que estamos en la rama correcta, procederemos a agregar el archivo de migración. Al hacerlo y actualizar Flyway, podremos visualizar los archivos que están en la carpeta de migraciones.



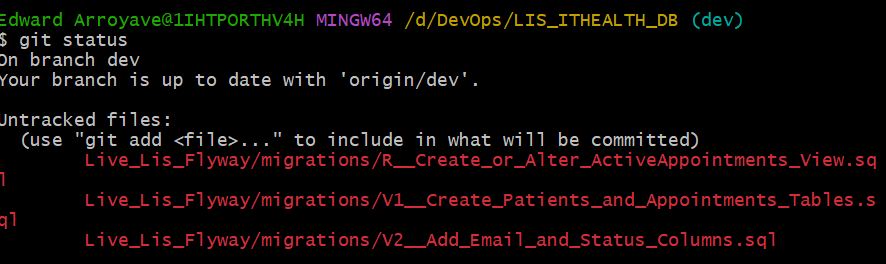
Como puedes ver, aunque el orden de los archivos en mi carpeta de migraciones dentro del repositorio es diferente, Flyway ejecuta primero las versiones antes que los archivos con encabezado R. Este orden es importante, ya que determina la secuencia en la que se aplicarán las migraciones.

## **Enviar cambios a la rama Dev -** **Git**

Luego de haber agregado nuestra migración y querer enviarla a la rama dev, debemos seguir estos pasos:

Debemos usar los siguientes comandos:

git status — Validaremos el estado del repositorio. Los archivos en rojo no estarán en el staging area.



Usa git add . para agregar todos los cambios al staging area.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

"Una vez que los cambios estén en el staging area, procederemos a realizar el commit utilizando el siguiente formato:"

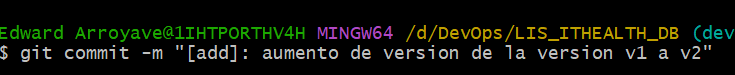
git commit -m "[verbo]: descripción del cambio"

"Es importante seguir esta convención para mantener un historial de cambios claro y organizado. Los verbos permitidos para los mensajes de commit son los siguientes:"

[add]: Para añadir nuevas funcionalidades.

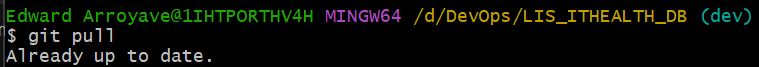
* **[fix]**: Para solucionar errores.
* **[update]**: Para actualizar funcionalidades existentes.
* **[remove]**: Para eliminar funcionalidades.
* **[refactor]**: Para cambios de código que no afectan el comportamiento del sistema (refactorización).
* **[docs]**: Para cambios relacionados con la documentación.
* **[test]**: Para agregar o modificar pruebas.
* **[chore]**: Para tareas generales, como la actualización de dependencias, configuración del entorno, etc.
* **[initial commit]**: Para el primer commit del proyecto.

"No seguir esta convención puede generar inconsistencias en el historial del repositorio y dificultar la colaboración entre desarrolladores."



Para evitar conflictos antes de enviar los cambios al remoto, debemos actualizar la versión local.

“git pull”

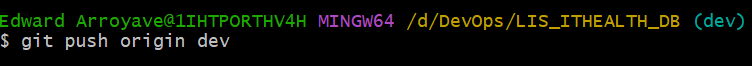


Después de confirmar los cambios, los enviaremos al repositorio remoto con el siguiente comando:

**“git push”**

Si queremos asegurarnos de que los cambios se envíen al origen correcto, podemos especificarlo de la siguiente manera:

**git push origin dev**



## **Enviar cambios a la rama Dev –** **Github Desktop**

Luego de haber agregado nuestra migración y antes de enviarla a la rama **dev**, debemos actualizar nuestro repositorio para evitar conflictos. Para ello, hacemos clic en el botón **Fetch origin** y verificamos si hay cambios.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

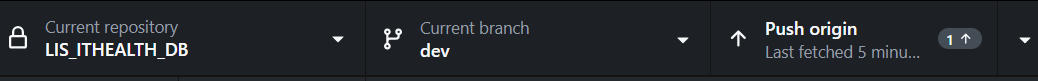
El contenido generado por IA puede ser incorrecto.

Si al presionar el botón **Fetch origin** se detectan cambios que no existen en nuestro repositorio local, debemos presionarlo nuevamente para actualizar nuestro repositorio local.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto. Luego de haber actualizado el repositorio local, confirmamos los cambios con un **commit**, que en **GitHub Desktop** se realiza en la parte inferior izquierda.

Luego de haber confirmado los cambios con el **commit**, enviamos los cambios al origen presionando el botón que aparece en la barra superior, el mismo que utilizamos anteriormente para hacer **pull ahora Push origin**.

  
Con esto, los cambios se enviarán al repositorio remoto.