

«Talento Tech»

Inteligencia Artificial

Clase 10



Clase N° 10 | Deploy del ChatBot

Temario:

- Git
- GitHub
- Creación de la cuenta de github
- Comandos básicos.
- Nuestro Primer Commit.
- Repositorio en GitHub.
- Deploy con Streamlit Cloud.
- ¿Qué sigue después de esto?

Git

Git es un proyecto de código abierto que se inició en 2005 y creció hasta convertirse en uno de los Sistema sde control de versiones (VCS, por sus siglas en inglés) más populares del mercado: cerca del 87% de los desarrolladores utilizan Git para sus proyectos.

Se trata de un sistema de control de versiones distribuido. Esto significa que cualquier desarrollador del equipo que tenga acceso puede gestionar el código fuente y su historial de cambios utilizando las herramientas de línea de comandos de Git.

A diferencia de los sistemas de control de versiones centralizados, Git ofrece ramas de características. Esto significa que cada ingeniero de software en el equipo puede dividir una rama de características que proporcionará un repositorio local aislado para hacer cambios en el código.

Las ramas de características no afectan a la rama principal, también llamada rama maestra, que es donde se encuentra el código original del proyecto. Una vez que se hayan realizado los cambios y el código actualizado esté listo, la rama de características puede fusionarse de nuevo con la rama maestra, que es la forma en que se harán efectivos los cambios en el proyecto.



Algunas de las funciones clave de Git incluyen:

1. **Control de versiones:** Git permite a los desarrolladores mantener un historial completo de los cambios realizados en un proyecto, lo que facilita revertir a versiones anteriores si es necesario.
2. **Colaboración:** Git facilita el trabajo en equipo, permitiendo que varios desarrolladores trabajen en el mismo proyecto simultáneamente sin conflictos. Cada desarrollador puede trabajar en su propia copia del proyecto (llamada "branch") y luego fusionar sus cambios con los de otros.
3. **Seguimiento de cambios:** Git registra quién hizo qué cambios y cuándo, lo que es crucial para la revisión del código y la resolución de problemas.
4. **Despliegue seguro:** Git permite probar nuevas características o corregir errores en ramas separadas antes de fusionarlas con el código principal, reduciendo el riesgo de introducir errores en la versión principal del software.
5. **Distribuido:** A diferencia de otros sistemas de control de versiones, Git es distribuido, lo que significa que cada desarrollador tiene una copia completa del repositorio con todo su historial. Esto mejora la flexibilidad y la seguridad del desarrollo.

GitHub

GitHub es un servicio basado en la nube que aloja un sistema de control de versiones (VCS) llamado Git. Éste permite a los desarrolladores colaborar y realizar cambios en proyectos compartidos, a la vez que mantienen un seguimiento detallado de su progreso.

¿Por qué GitHub es tan popular?

GitHub aloja más de 100 millones de repositorios, la mayoría de los cuales son proyectos de código abierto. Esta estadística revela que GitHub se encuentra entre los clientes Git GUI más populares y es utilizado por varios profesionales y grandes empresas, como Hostinger.

Esto se debe a que **GitHub** es una plataforma de gestión y organización de proyectos basada en la nube que incorpora las funciones de control de versiones de Git. Es decir que todos los usuarios de **GitHub** pueden rastrear y gestionar los cambios que se realizan en el

código fuente en tiempo real, a la vez que tienen acceso a todas las demás funciones de Git disponibles en el mismo lugar.

Además, la interfaz de usuario de **GitHub** es más fácil de usar que la de Git, lo que la hace accesible para personas con pocos o ningún conocimiento técnico. Esto significa que se puede incluir a más miembros del equipo en el progreso y la gestión de un proyecto, haciendo que el proceso de desarrollo sea más fluido.

Para qué sirve GitHub

En el campo de la ingeniería y la programación de software, **GitHub** permite crear repositorios remotos y públicos en la nube, de manera gratuita. Los repositorios están constituidos de los archivos y el historial de revisiones de cada proyecto de programación.

Después de que se ha creado un repositorio en **GitHub**, es posible copiarlo en un dispositivo, añadir y modificar archivos de manera local, y luego subir esos cambios al repositorio para que todo el mundo pueda verlos.

Comandos básicos de Git.

Git es una herramienta de control de versiones que te permite gestionar el historial de cambios en tus proyectos de desarrollo. A continuación, se presentan algunos de los comandos básicos que necesitarás conocer para trabajar con Git. No usaremos todos para nuestro proyecto, pero es importante que los reconozcas, ya que te ayudarán a iniciar, modificar, y mantener el control de versiones de tus proyectos de manera efectiva.

1. git init

Inicializa un nuevo repositorio Git vacío en el directorio actual.

2. git clone <url>

Crea una copia local de un repositorio remoto especificado por la URL.

3. git add <archivo>

Añade cambios en el archivo especificado al área de preparación (staging area).

4. git commit -m "<mensaje>"

Guarda los cambios añadidos al área de preparación con un mensaje descriptivo.

5. **git status**

Muestra el estado actual del repositorio, incluyendo archivos modificados y archivos no añadidos.

6. **git log**

Muestra un historial de los commits realizados en el repositorio.

7. **git pull**

Descarga los cambios del repositorio remoto y los fusiona con la rama local actual.

8. **git push**

Envía los cambios de la rama local al repositorio remoto.

9. **git branch**

Lista todas las ramas en el repositorio, o crea una nueva rama si se especifica un nombre.

10. **git checkout <rama>**

Cambia a la rama especificada en el repositorio.

11. **git merge <rama>**

Fusiona la rama especificada con la rama actual.

12. **git remote add origin <url>**

Añade un repositorio remoto llamado "origin" con la URL especificada.

13. **git remote -v**

Muestra las URLs de los repositorios remotos asociados con el repositorio local.

14. **git rm <archivo>**

Elimina un archivo del repositorio y lo marca para eliminación en el próximo commit.

15. **git diff**

Muestra las diferencias entre los archivos modificados y el último commit.

¡Usemos GitHub para nuestro proyecto final!



Vamos a subir nuestro proyecto a un repositorio de GitHub para que podamos mostrarlo desde el hosting de streamlit Cloud. Lo primero que vamos a hacer es crearnos una cuenta de GitHub, eso lo logramos ingresando [ACA](#) desde el navegador. Una vez ahí, vamos a seguir los siguientes pasos:

1. Hacemos click en "Sign up" o "Sign up for GitHub", e ingresamos una dirección de correo electrónico. Recordá que GitHub es utilizado por profesionales y marca nuestra ruta de aprendizaje, por lo que es importante usar un mail principal. Luego seguimos las instrucciones donde nos pedirán **un nombre de usuario, una contraseña, y posiblemente resolver un captcha.**
2. **Confirmación:** Después de registrarte, GitHub enviará un correo de verificación a tu bandeja de entrada. Abre el correo y haz clic en el enlace para verificar tu cuenta.

Si estás desde tu computadora, vas a necesitar instalar git para poder subir tu código directamente desde tu editor de código.

Crear un repositorio en GitHub

Creando un nuevo repositorio: GitHub se maneja con repositorios, que podemos pensarlo como carpetas que contendrán todo nuestro código. Para crear un repositorio, vamos a hacer clic en el botón + en la esquina superior derecha y selecciona **"New repository"**.

- Le asignamos un **nombre**(ejemplo: *mi-proyecto*).
- Le agregamos una breve **descripción** contando nuestro proyecto
- Puedes elegir si el repositorio será **público o privado** (para esta clase, elige que sea público). Por lo general elegimos privado cuando el proyecto está en marcha y lo pasamos a público cuando hayamos terminado o queremos que alguien (algún alma generosa con más conocimientos que nosotros) colabore con el código.
- Podemos agregarle un README y una licencia.
- Haz clic en el botón **"Create repository"** para continuar.

Subir los archivos

Al momento de subir los archivos, podemos hacerlo de varias formas, siendo las dos mas conocidas las siguientes:



- a. Por línea de comandos desde la terminal de mi editor de código
- b. **Por la interfaz de GitHub web.**

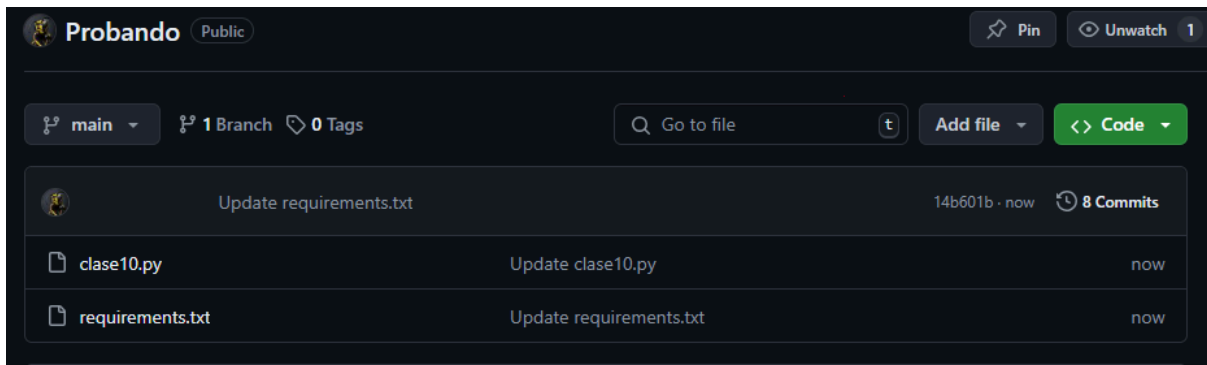
Nosotros nos vamos a centrar en la segunda opción, ya que nuestros archivos serán únicos y ya comprobamos localmente su funcionalidad.

1. Subo mi app.py y mis requirements.txt

Para hacer la carga de los archivos, necesitaremos tener nuestro repositorio de GitHub web abierto. Dentro del repositorio, hacemos click en **Add file => Upload files** y agregamos manualmente los dos archivos principales.

Hacemos click en **Commit changes**, le agremos un comentario al cambio, y presionamos nuevamente commit changes para guardar los cambios.

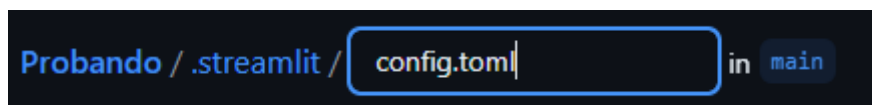
Nuestro repositorio se verá de esta forma:



2. Creo una carpeta para guardar mis configuraciones

Para crear una carpeta desde github, será necesario irnos nuevamente a **Add file => Create new file** y en nombre escribimos **.streamlit/config.toml**.

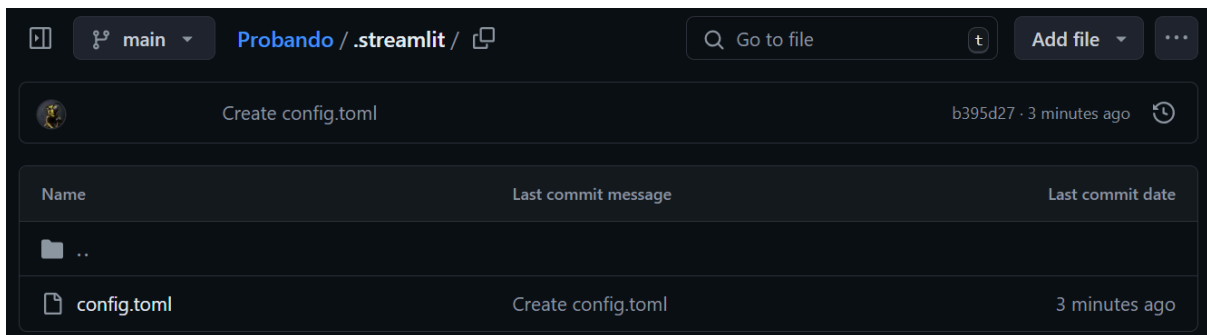
Es importante saber que cuando agregamos la barra (/) después de .streamlit, github ya reconoce que .streamlit será una carpeta y lo que nosotros agreguemos luego, irá dentro de la misma.



En mi caso, mi repositorio se llama **Probando**, que tiene una carpeta llamada **.streamlit**, y dentro creo el archivo **config.toml**.



¡importante! En esta carpeta solamente tendremos el archivo `config.toml`. Por lo tanto copiamos y pegamos el código que vemos en nuestro editor y lo pegamos dentro del recién creado en github. Dejamos el `secrets.toml` para más adelante.
¡No elimines tu API Key!



Este archivo recordá que tiene que tener el tema de color que le asignamos en la clase 7.

¡Listo ya subimos nuestro proyecto final a nuestro repositorio de GitHub!.

De GitHub a Streamlit Cloud

¡Excelente!, ya que subimos nuestro proyecto a GitHub, ahora nos faltaría desplegarlo para que nuestro ChatBot esté en internet, y para ellos vamos a usar **Streamlit Cloud**, ¿qué es **Streamlit Cloud**?, te lo explico de manera simple y breve

Streamlit Cloud es un hosting que permite a los desarrolladores y científicos de datos compartir y desplegar sus aplicaciones, con muy pocas configuraciones

Características clave de Streamlit Cloud:

1. **Despliegue fácil:** Con un par de click ya está la app en la web.
2. **Actualizaciones automáticas:** Cada vez que se hacen nuevos cambios a GitHub, Streamlit Cloud puede actualizar automáticamente la aplicación desplegada.
3. **Colaboración:** Nos permite compartir fácilmente las aplicaciones con otros usuarios mediante un enlace, lo que facilita la colaboración en proyectos de análisis de datos, prototipos de machine learning, y más.

4. **Escalabilidad:** Proporciona recursos para ejecutar aplicaciones, incluyendo CPU y memoria, y puede escalar para manejar múltiples usuarios simultáneos.
5. **Administración de secretos:** Ofrece una forma segura de gestionar variables sensibles, como claves API y contraseñas, sin tener que incluirlas directamente en el código fuente.

Creandonos una cuenta

Para poder subir nuestro proyecto a internet vamos a crearnos una cuenta en Streamlit Cloud. Para eso vamos a seguir estos pasos:

1. Visitar el sitio de Streamlit Cloud en <https://streamlit.io/cloud>.
2. Nos aseguramos de tener nuestra cuenta de GitHub abierta y nuestra aplicación subida.
3. Iniciamos sesión con GitHub haciendo clic en "Sign in with GitHub" o "Log in" y serás redirigido a GitHub para autorizar a Streamlit Cloud a acceder a tus repositorios.
4. Autorizar Streamlit Cloud: Una vez iniciada la sesión en GitHub, se te pedirá que autorices a Streamlit Cloud para acceder a tus repositorios. Haz clic en "Authorize Streamlit".

Después de la autorización, Streamlit Cloud puede pedirte algunos detalles adicionales como tu nombre o preferencias. Completa estos pasos para finalizar la creación de tu cuenta.

Deployando nuestra app

Una vez que tengamos nuestra cuenta creada y autorizada, vamos a subir nuestro proyecto. Sigamos los siguientes pasos para asegurarnos de que nuestra aplicación corra correctamente.

1. **Creamos una nueva app:** Desde la página principal de Streamlit Cloud hacemos clic en "Create App", en la parte superior derecha y cuando se nos solicite seleccionamos donde dice **"Yup, I have an app"**.

2. **Buscando nuestra aplicación:** Streamlit Cloud te mostrará una lista de tus repositorios en GitHub, y nosotros vamos a seleccionar el repositorio que contiene tu proyecto de Streamlit.
3. **Seleccionando la rama:** Después de seleccionar el repositorio, podemos elegir la rama (**branch**) que deseamos desplegar, en nuestro caso será **main**. (recordá que podemos tener también varias ramas si trabajamos en conjunto a otras personas)
4. **Seleccionamos nuestro archivo principal:** Este es el archivo con extensión .py donde hicimos todo el código de nuestro ChatBot.

Una vez llegado a este punto, vamos a hacer unas configuraciones presionando :

- **AppUrl:** Acá colocamos la URL que queremos que tenga nuestro proyecto cuando lo compartimos, sino se te ocurre nada puedes dejar el que te aparece por defecto. Siempre será este formato:

nombreQueLeDiALaURL.streamlitapp

5. **5. Advanced Settings:** Dejamos tal cual la versión de Python que nos aparece y después en secrets vamos a colocar la **groqAPIKey** que tenemos en nuestro archivo, *secrets.toml* y guardamos los cambios . Debemos copiarla tal cual la tenemos así te debería quedar algo así.

×

Advanced settings

Python version

3.12

Secrets

Provide environment variables and other secrets to your app using [TOML](#) format. This information is encrypted and served securely to your app at runtime. Learn more about Secrets in our [docs](#). Changes take around a minute to propagate.

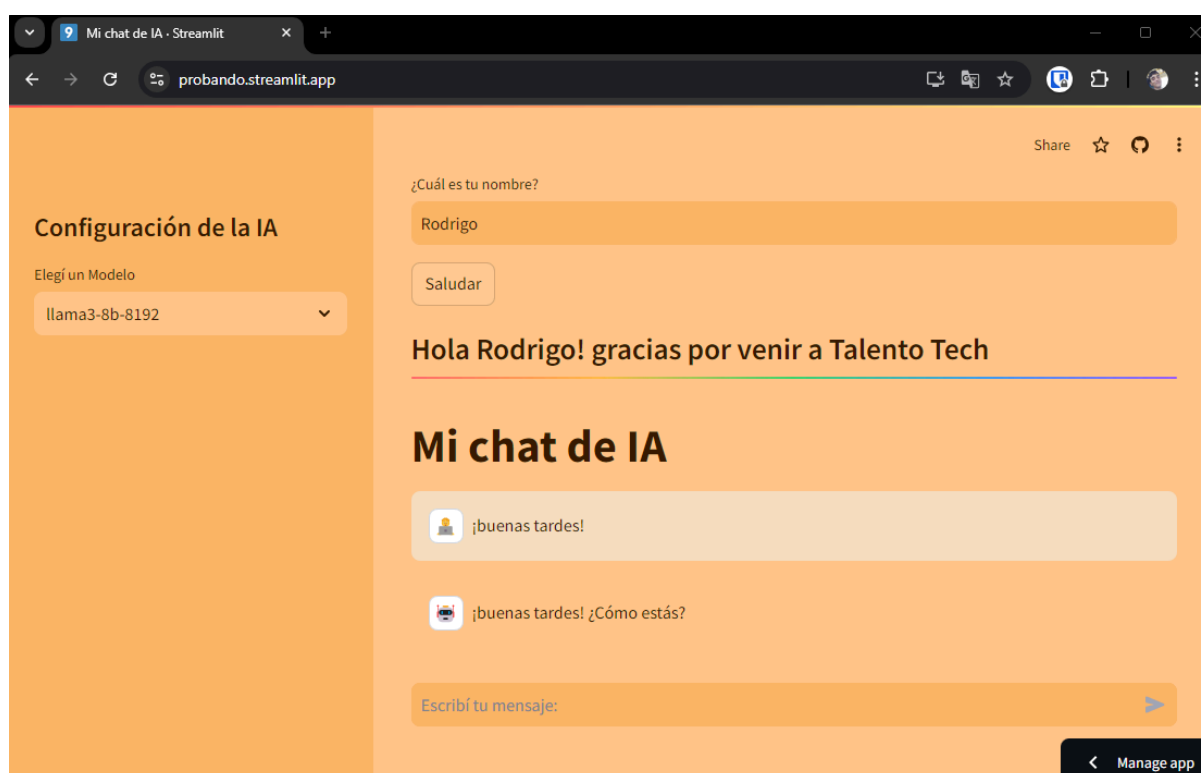
```
groqAPIKey
='gsk_Yk54464674421414140iDovDzTe4o41YWGdyb3FY0ptQm18fwSIZNXsZ7TbB6vbn '
```

Save

6. **Desplegar la aplicación:** Hacemos clic en **"Deploy"** para iniciar el despliegue de la aplicación y Streamlit Cloud comenzará a construir la aplicación utilizando las dependencias especificadas en `requirements.txt`. Esto puede tardar unos minutos, dependiendo de la cantidad de librerías que tengas.
7. **Ver la aplicación en vivo:** Una vez desplegada, se te proporcionará un enlace que puedes compartir con otros para acceder a tu aplicación en línea. Puedes visitar este enlace para ver y probar tu aplicación en directo.



Si todo funcionó a la perfección debería aparecerte tu chatbot similar a este, solo nos quedaría, en caso de ser necesario, ir a settings y cambiarle el tema por que nosotros le dimos.:



Vemos que la URL es la que escribimos nosotros, por lo que nos confirma que el chatBot se desplegó de manera correcta y está funcionando a la perfección en internet,

si te da algún error no pierdas tiempo en pedirle ayuda a tu instructor y tutor para que lo puedas solucionar.



Tutorial sobre GitHub y Streamlit Cloud

Si no pudiste conseguir el deploy de tu chatBot, no te preocupes, hemos creado el siguiente tutorial paso a paso de cómo es todo el proceso de deploy desde nivel local,



subiendo nuestro proyecto a un repositorio de Github y luego desplegarlo con Streamlit Cloud.

```
<iframe width="854" height="480"
src="https://www.youtube.com/embed/ZiZXIfdBHgA?si=ekf0GTKKDxlyFJqY" title="YouTube
video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write;
encrypted-media; gyroscope; picture-in-picture; web-share"
referrerpolicy="strict-origin-when-cross-origin" allowfullscreen></iframe>
```

¿Y ahora qué sigue?

Felicitaciones por haber culminado exitosamente el curso de **Inteligencia Artificial con Python**, seguro ahora te preguntarás: ¿qué curso hago ahora? ¿Qué sigue? ¿Qué oportunidades tengo?, te dejo las respuestas:

INTELIGENCIA ARTIFICIAL CON PYTHON

¿Qué sigue después?



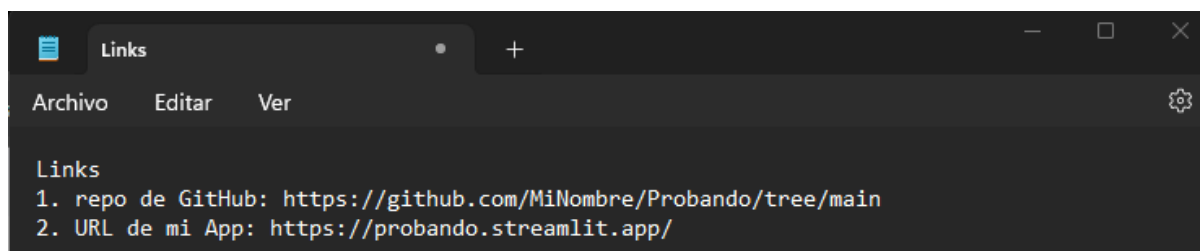


Si al terminar este curso, **ya sales de secundaria y tienes 18 años** nuestra invitación es a que te unas a la siguiente propuesta:

- Talento Tech Adultos.

Desafío N° 10:

1. Entrega el link de tu repositorio de GitHub.
2. Entrega la URL del hosting facilitada por Streamlit Cloud.



```
Links
1. repo de GitHub: https://github.com/MiNombre/Probando/tree/main
2. URL de mi App: https://probando.streamlit.app/
```



Buenos Aires
aprende
Agencia de Actividades para el Futuro

BA Buenos
Aires
Ciudad