**Proyecto para el challenge "AB InBev MLOps Challenge v7"**

**Por. Diego Rivera**

**Descripción general del proyecto**

**Para este desafío se desarrolló un sistema de predicción de diabetes con una red neuronal profunda con el objetivo de ser utilizado a manera preventiva en las personas, por eso este modelo utiliza información de hábitos cotidianos o información básica acerca del estado de salud. Es pensado para prevenir y no para diagnosticar, pues el diagnóstico se deja a cargo de los especialistas de la salud y las entidades competentes.**

**Este desarrollo presenta dos opciones de uso, una de modo individual y otra de modo masivo o por lotes.**

**Para realizar la predicción se debe ingresar la información que se solicita, si se hace de modo individual. Si se utiliza el modo por lotes se ha cargado previamente un archivo tipo Excel con información anonimizada de pacientes, el cual servirá para simular una base de datos.**

**Al final se produce una salida en archivos .json para ser visualizada en tablas y en una gráfica de barras si se está en modo individual.**

**El punto de partida es un modelo de clasificación entrenada con una red neuronal para predecir la diabetes llamado “model\_diabetes.h5”, este modelo es de autoría propia.**

**Para la creación de la API se utilizó la librería Streamlit por su facilidad para la creación de aplicaciones relacionadas con data** esta es el sitio web: <https://streamlit.io>.

**El repositorio de este desarrollo se encuentra en esta dirección** <https://github.com/diegoriver/ABIChallenge_Diego-Rivera>

1. **Arquitectura**:

**Este sistema desarrollado se ha estructurado en tres pasos y dos modas de ejecución que son de modo individual y el otro de modo por lotes.**

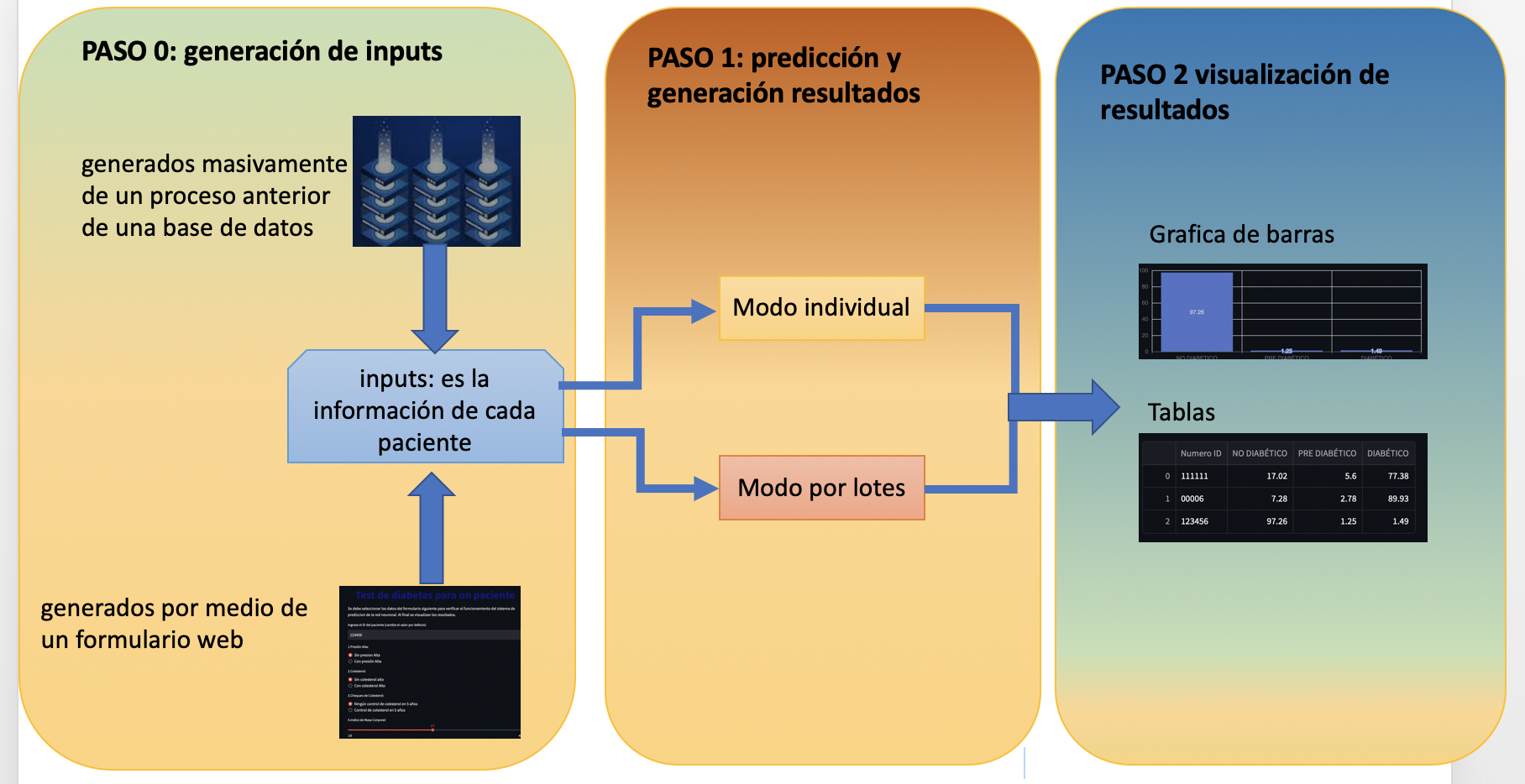
**Cada paso marca las etapas en las cuales se presenta la solución del objetivo que es la predicción de diabetes para pacientes.**

**PASOS DEL PROCESO**

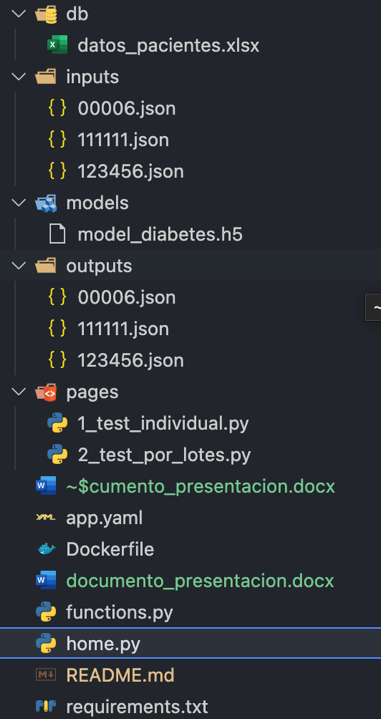
* **PASO 0: En este paso se realiza la creación de los inputs, estos son archivos .json de la carpeta de “/inputs” con la información de cada paciente.**

**Cabe mencionar que en un modelo de producción se supone que los inputs provienen de un proceso previo, el cual simulamos en este paso, tanto en modo individual como en modo por lotes se crean los inputs. Se hace la suposición que estos inputs provienen de un proceso anterior generado desde una base de datos con la información de los pacientes.**

* **PASO 1: En este paso, los resultados de la predicción se obtienen utilizando el modelo entrenado y los archivos .json de salida se crean en la carpeta de “/outputs” con los resultados obtenidos.**
* **PASO 2: En este paso, se cargan y se muestran los resultados de la carpeta de “/outputs” y se visualizan por medio de tablas, también, para el modo individual se utilizan gráficas de barras.**



**Esta es la estructura que se encuentra en el repositorio de Github:**



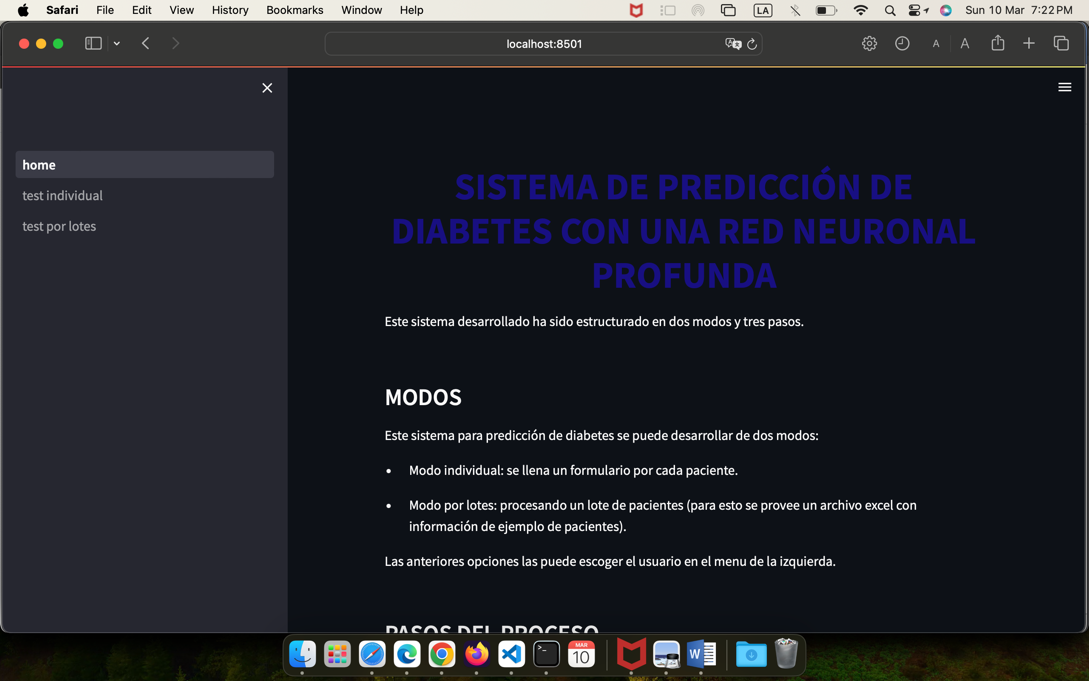
* **Home.py: es el archivo inicial que describe el proyecto. Se ejecuta en la línea de comandos así: streamlit run home.py**
* **Functions.py: contiene todas las funciones utilizadas.**
* **Page: es la carpeta que contiene las páginas para usar el modo individual o el modo por lotes de ejecución de la predicción.**
* **Models: esta carpeta contiene el modelo previamente entrenado.**
* **Inputs: esta carpeta contiene los archivos de entrada, estos se generan con el formulario de la pagina de modo individual o con el botón de “PASO 0” de la pagina de modo por lotes. Se dejan 3 archivos en inputs a manera de ejemplo.**
* **Outputs: esta carpeta contiene los archivos con los resultados de las predicciones. Se dejan 3 archivos en outputs a manera de ejemplo.**
* **Db: esta carpeta contiene un archivo Excel con 100 registros de pacientes para ser ejecutados a manera de prueba cuando se utilice la pagina del modo por lotes.**

**MODOS**

**Este sistema de predicción de diabetes se puede desarrollar de dos formas:**

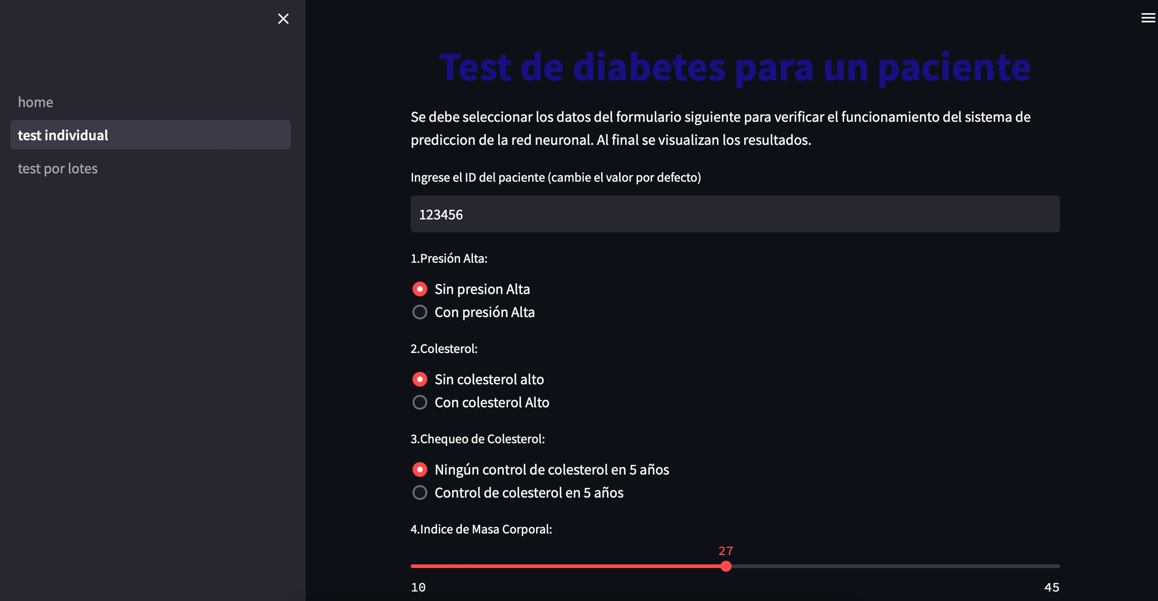
* **Modalidad individual: se rellena un formulario por cada paciente.**
* **Modo por lotes: procesar un lote de pacientes (para esto se proporciona un archivo Excel con información de paciente de ejemplo).**

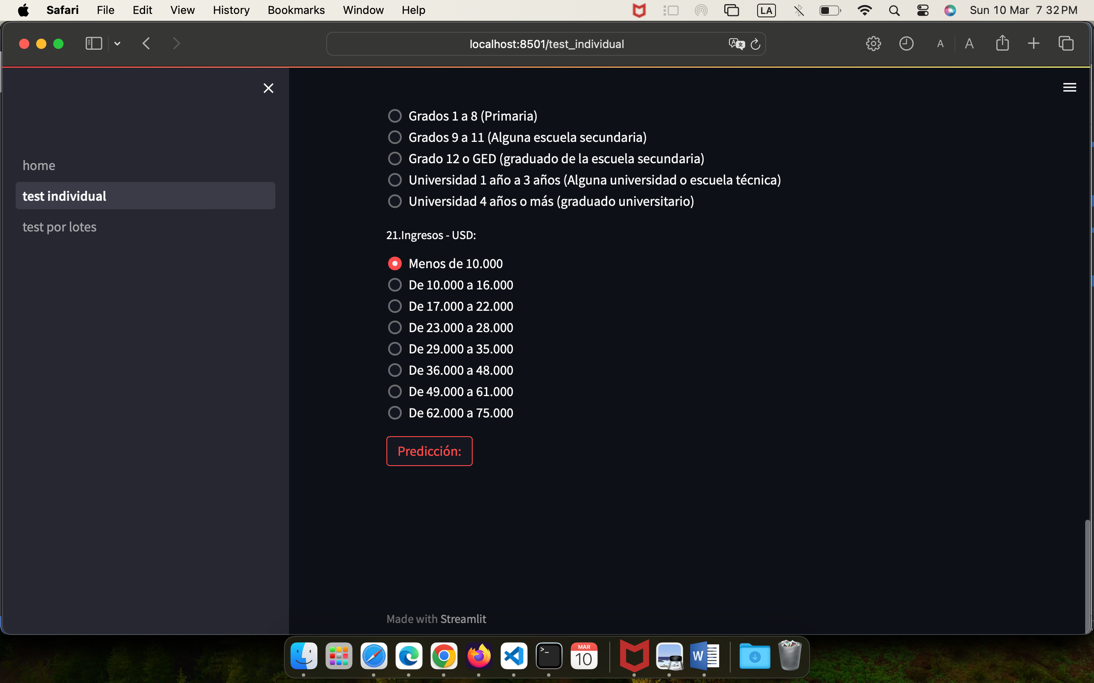
**El usuario puede elegir las opciones anteriores en el menú de la izquierda.**

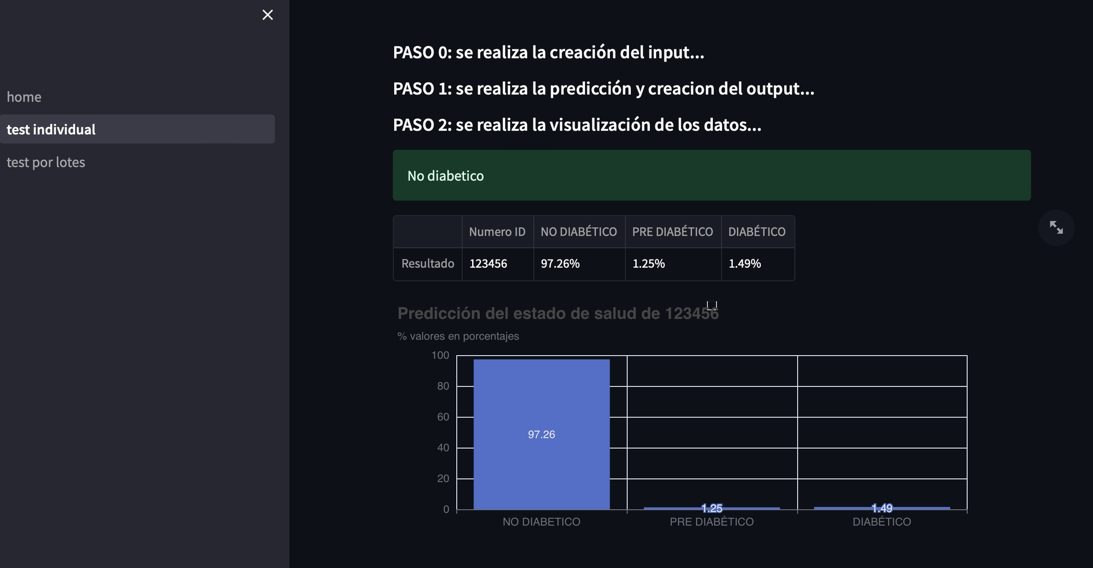


A continuación, se muestran unas pantallas de cada uno de los modos, con los pasos que se dan:

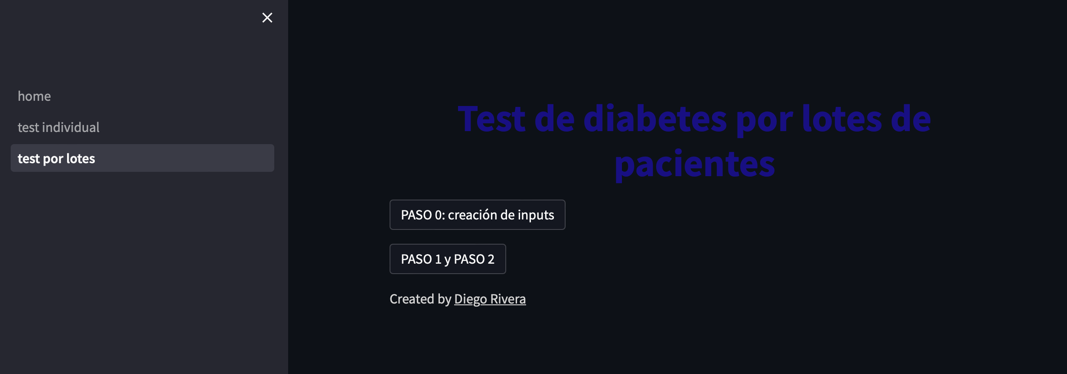
Modo individual:

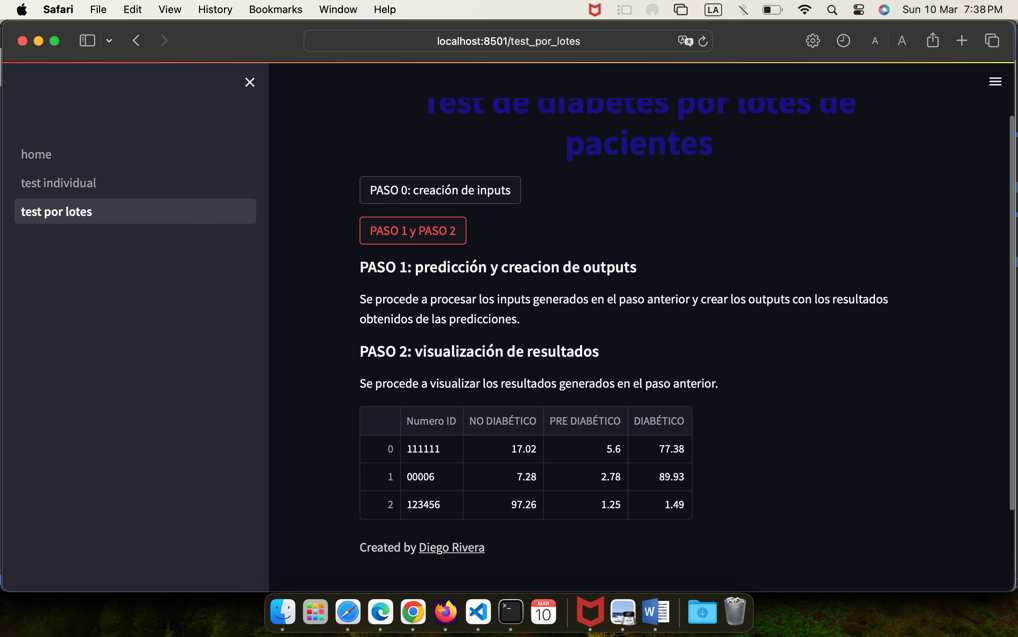






Modo por lotes:



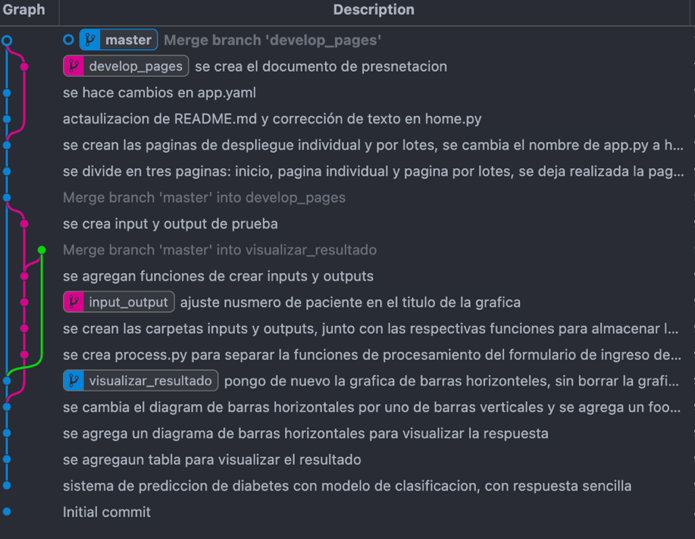


1. **Colaboración**: Para simular un flujo de trabajo colaborativo se creó un repositorio privado en GitHub con diferentes ramas y commits, esta es la dirección:

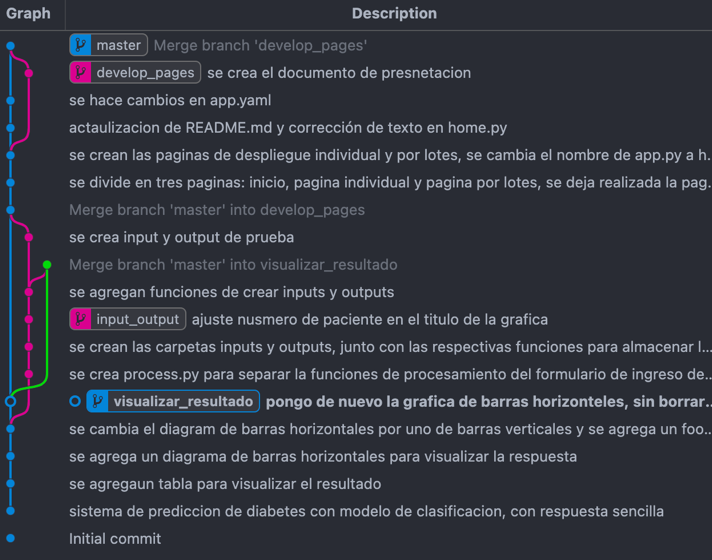
<https://github.com/diegoriver/ABIChallenge_Diego-Rivera>

En este repositorio se tienen las siguientes ramas:

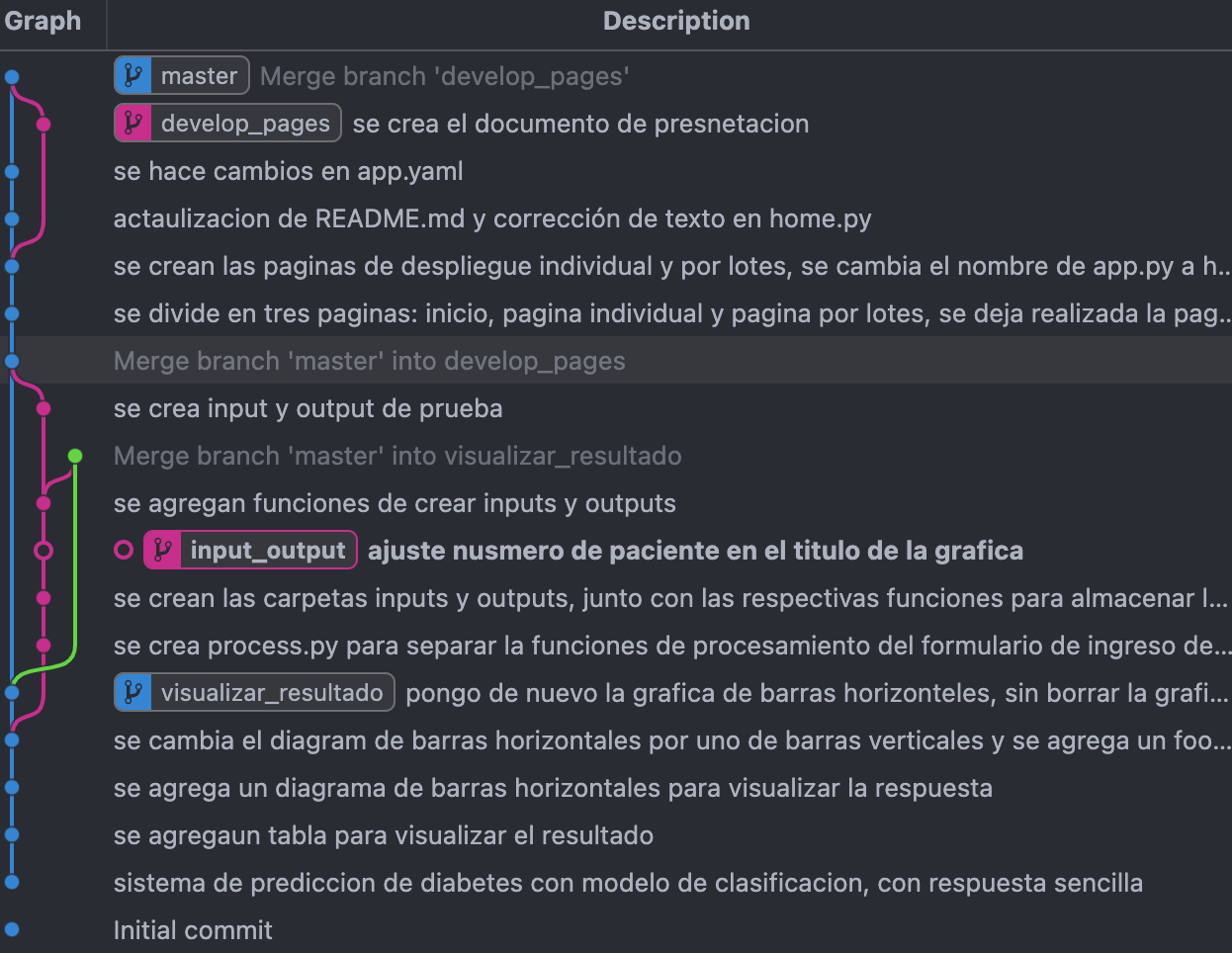
* + 1. Master: rama principal donde se hace el desarrollo principal del proyecto y se integran el trabajo de las otras ramas.



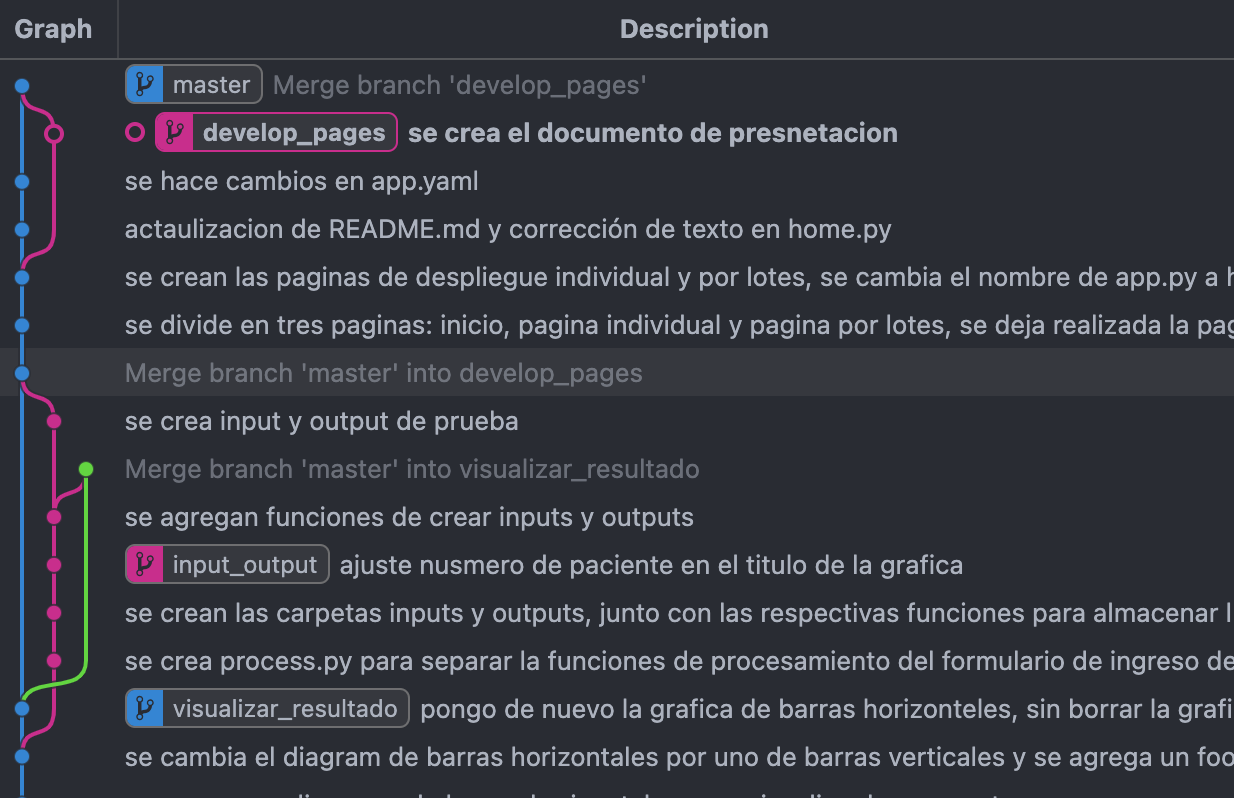
* + 1. Visualizar\_resultados: el trabajo en esta rama fue de crear la visualización de los resultados obtenidos acerca de la predicción de la diabetes realizada por un modelo de inteligencia artificial previamente entrenado, esta rama se ha dejado con un commit sin hacer merge con la rama master para simular que el trabajo en esa rama puede continuar.



* + 1. Input\_output: en esta rama se trabajaron las funciones para la creación de los inputs de los valores médicos de los pacientes para ser procesados y la creación de los outputs, es decir, los resultados obtenidos de las predicciones.



* + 1. Develop\_pages: en esta rama se trabajó la creación de las páginas: pagina inicial o Home, página para procesamiento individual de cada registro y página para procesamiento de un lote de registros de pacientes para ser evaluados si tenían diabetes, prediabetes o no tenían diabetes.



1. **Automatización**: Para configurar SonarCloud en tu repositorio y ejecutar pruebas para validar tu código, puedes seguir estos pasos:
   * Ve a la página de SonarCloud y haz clic en “Analyze new project”.
   * Selecciona tu repositorio de GitHub.
   * Sigue las instrucciones para configurar SonarCloud con tu repositorio.
   * Una vez configurado, SonarCloud analizará automáticamente tu código cada vez que hagas push a tu repositorio.
2. **Despliegue del Modelo de ML**:

Para exponer el modelo previamente entrenado se utilizó el marco de trabajo Streamlit el cual es basado en Python, en el archivo README.md se explica como se debe ejecutar el proyecto utilizando esta libreria.

De manera adicional se incluyó un archivo Dockerfile para la ejecución del proyecto en otros entornos de trabajo.

1. **Presentación (opcional)**: Para destacar los puntos clave del proceso, los desafíos principales y los hallazgos, puedes crear una presentación o un documento que incluya:
   * Una descripción general del proyecto.
   * Los desafíos que enfrentaste y cómo los superaste.
   * Los resultados que obtuviste.
   * Cualquier hallazgo interesante o inesperado.
   * Los próximos pasos o mejoras que podrías hacer en el futuro.