Maestría
en Inteligencia
Analítica de Datos

Taller: Entendiendo repositorios en GitHub

La entrega de este taller consiste en un reporte y unos archivos de soporte. Cree el archivo de su reporte como un documento de texto en el que pueda fácilmente incorporar capturas de pantalla, textos y similares. Puede ser un archivo de word, libre office, markdown, entre otros.

Pre-requisitos

1. Para esta sesión va a requerir una cuenta de **Github**. Si no la ha creado, es el momento de hacerlo en `https://github.com`. Note que Github requiere autenticación de doble factor (2FA), actívela usando por ejemplo la app de Github o de una aplicación de autenticación.
2. Descargue el **cliente de Git** para su sistema operativo en `https://git-scm.com/downloads`.
3. Ejecute el cliente. **Atención**: deje todas las configuraciones por defecto, excepto el editor, para el cual se recomienda seleccionar VSCode.
4. En caso de que no tenga instalado aún VSCode, es el momento de hacerlo en `https://code.visualstudio.com/download`. Deje todas las configuraciones por defecto.
5. Descargue GitHub Desktop para facilitar el manejo de los repositorios virtuales. Lo puede hacer en `https://desktop.github.com/download/`.

1. Crear un repositorio Git local

1. En este taller haremos uso de Git como herramienta de versionamiento de código.

   Después de instalar Git en su máquina local (pre-requisitos), crearemos una carpeta local dedicada a todo el material correspondiente a este taller.
2. Después de crearla, abra VSCode, vaya a File y seleccione la carpeta creada.
   En el panel de la izquierda seleccione Source Control y Initialize Repository para iniciar el repositorio local.
3. Antes de comenzar a usar Git, realizaremos una configuración inicial con un nombre de usuario y su correo. Para esto emita los siguientes comandos en la terminal, reemplazando `'your-username'` por su nombre de usuario y `'your-email'` por su correo:


```
git config --global user.name 'your-username'
git config --global user.email 'your-email'
```

4. Verifique que su carpeta (local) incluya una carpeta oculta .git. De esta forma, verificamos que se creó correctamente nuestro repositorio. Tome un pantallazo e incluyalo en su reporte.
5. Ahora, utilizando el explorador de archivos, agregaremos todos los archivos necesarios para el funcionamiento del tablero a esta carpeta. Estos archivos los encontrará disponibles en Coursera. Esto incluye el archivo app.py, los datos en datos_energia.csv y la carpeta assets, que contiene dos archivos de tipo css.
6. Una vez estén todos los archivos en la carpeta, regrese a VSCode y agregue los archivos al control de versiones (staging).

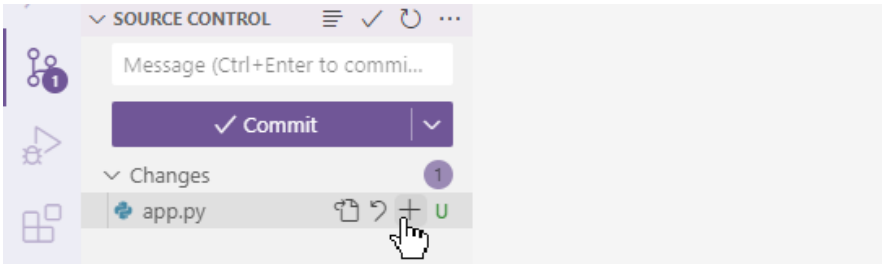

Figura 1: Staging.

7. Incluya un mensaje para el commit y realice el commit en la rama main. Tome un pantallazo e incluyalo en su reporte.
8. En la terminal, navegue hasta la ubicación de su repositorio local y use el siguiente comando:

   ```
   git log
   ```

   Este comando muestra el estado actual de su repositorio. Tome un pantallazo e incluyalo en su reporte.
9. Como siguiente paso, haremos un cambio en el archivo app.py. El código en este archivo define la lógica de un tablero que permite visualizar un pronóstico de producción energética. Para esto, el tablero debe cargar el archivo .csv que contiene los datos históricos de producción junto con el prónostico de energía. En el código encontrará la función `load_data()`, donde usted deberá desarrollar el código para cargar el archivo `datos_energia.csv` como un DataFrame. Para el funcionamiento correcto del tablero, tenga en cuenta que la columna con la fecha debe estar en formato `datetime` (para esto puede hacer uso de la función `to_datetime()` del paquete Pandas) y que el índice del DataFrame deberá ser la fecha. ***Hint:*** **Tenga en cuenta que la función deberá**

retornar el dataframe.

10. Para verificar el funcionamiento de su código, instale los paquetes necesarios en su máquina local y ejecute el código en su editor de código preferido (sin embargo, no lo ejecute en un cuaderno de Jupyter). Al ejecutarlo, en la terminal de su editor de código recibirá un mensaje que contiene la dirección en la que esta corriendo su tablero.

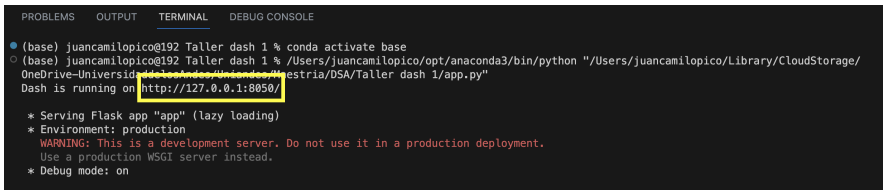

Figura 2: Dirección del tablero.

Entre a esta dirección utilizando su navegador para visualizar el tablero (Figura 3)
Tome un pantallazo del tablero funcionando correctamente e incluyalo en su reporte.

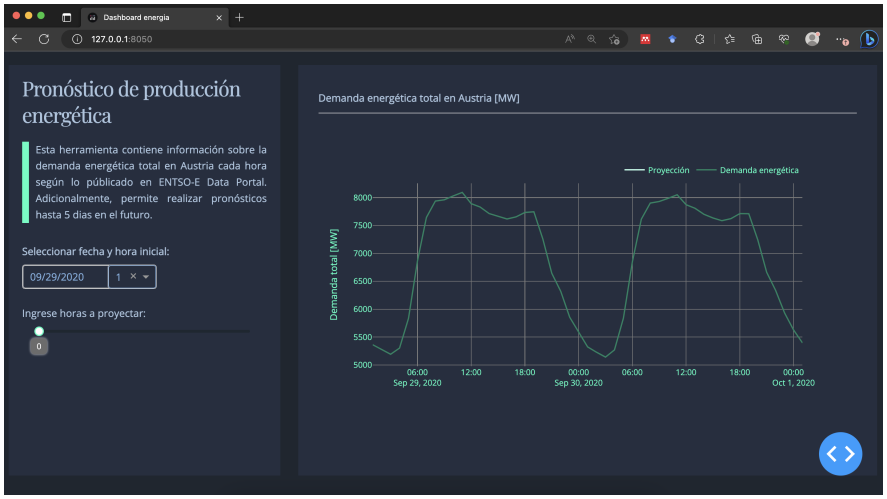

Figura 3: Tablero.

Hint: **Para correr su código desde la terminal puede utilizar el siguiente comando:**

```
python app.py
```

Recuerde haber instalado previamente todas las librerías necesarias.

11. Para terminar la ejecución del tablero, emita en la terminal donde se está ejecutando su código el comando `CTRL + c`.

12. Ahora que le realizamos cambios al archivo app.py, haremos un nuevo commit. Al editar los archivos, automáticamente se actualiza en source control.

    De nuevo, agregue los cambios al control de versiones y cree el commit.
13. Para observar el historial de commits, podemos utilizar el comando:

    ```
    git log
    ```

    Con este comando debería observar un historial similar a este:

```
(base) juancamilopico@192 Taller dash 1 % git log
commit 6f112f573ed9eda374e8193a77ce932faa3f293f (HEAD -> master)
Author: Juan Camilo Pico <juanca.pico@hotmail.com>
Date:   Wed Jul 19 00:42:42 2023 -0500

    nueva funcion para cargar datos

commit e506748ac267bf447110f9c6098448fc5e304490
Author: Juan Camilo Pico <juanca.pico@hotmail.com>
Date:   Wed Jul 19 00:21:38 2023 -0500

    mi primer commit
(base) juancamilopico@192 Taller dash 1 %
```

Figura 4: Historial de cambios.

Tome un pantallazo del historial de commits e incluyalo en su reporte.

2. Trabajar distintas ramas en un repositorio Git local

1. En VSCode, vista de control de versiones, use la opción Branch → Create Branch para crear una nueva rama. Use su primer nombre como parte del nombre de la rama, e.g., juan-viz.
   Las ramas permiten hacer cambios paralelamente sin afectar los archivos principales (rama main).
2. Puede hacer uso del comando `git log` para mostrar el estado actualizado de su repositorio con la nueva rama. Tome un pantallazo e incluyalo en su reporte.
3. Realice un nuevo cambio en los archivos. Agregue el cambio al control de versiones y realice el commit.
4. Use nuevamente el comando `git log` para mostrar el estado actualizado de su repositorio con la nueva rama y la modificación al archivo. Tome un pantallazo e incluyalo en su reporte.
   Note que estos cambios se hacen en su rama personal y no en la rama main.
5. Vuelva a la rama main. Esto lo puede hacer usando la opción Checkout to, en Source Control.
6. Use el comando `git log` para mostrar el estado actualizado de su repositorio en la rama main. Tome un pantallazo e incluyalo en su reporte.

7. Ahora, actualizaremos la rama main, incorporando su rama personal a la rama main. Esto trae los cambios realizados en su rama hacia la rama principal. Esto se hace usando la opción Branch → Merge Branch.
8. Use nuevamente el comando `git log` para mostrar el estado actualizado de su repositorio en la rama main. En este punto se puede observar los cambios realizados en la rama main, donde se observan los cambios realizados en la rama auxiliar y el HEAD debe apuntar a las dos ramas (main y auxiliar). Tome un pantallazo e incluyalo en su reporte.

3. Publicar un repositorio en Github

1. En VSCode>Source Control, asegúrese de estar en el repositorio creado anteriormente en la rama main.
2. Use el botón Publish Branch para publicar la rama main. Aquí debe autorizar a VSCode para enlazar con su cuenta de GitHub. Para este taller, asegúrese de publicar en un repositorio **público.**
3. Use el comando `git log` para mostrar el estado actualizado de su repositorio, donde se debe observar ahora también la rama main asociada al repositorio remoto. Tome un pantallazo e incluyalo en su reporte.
4. Realice ahora un cambio sobre el archivo, localmente, usando VSCode.
5. Agregue el cambio al control de versiones en VSCode y realice el commit. En este caso, es necesario realizar una sincronización (Sync Changes) que ejecuta un push y un pull a la ubicación remota.
6. Use el comando `git log` para mostrar el estado actualizado de su repositorio. En este se debe observar el cambio realizado sobre la rama main asociada al repositorio remoto. Tome un pantallazo e incluyalo en su reporte.
7. Asegúrese de que en GitHub su repositorio incluya el cambio realizado.

4. Colaborar a través de repositorios

1. En GitHub incluya como colaboradores a su compañero de grupo.
   Para agregar colaboradores en su repositorio siga los siguientes pasos:
   a) Ingrese a su cuenta en GitHub.
   b) Acceda a su repositorio e ingrese a Settings o Configuración.
   c) En Configuración>Colaboradores y en esta ventana podrá agregar a su compañero. Al correo de su compañero llegará la invitación a colaborar, siga las instrucciones del correo para finalizar

el proceso.

2. Permita acceso restringido a través de pull request a su rama principal (master/main). Para hacerlo siga los siguientes pasos:
   a) Ubiquese en Configuración>Ramas>Agregar regla de ramas.
   b) Pongale un nombre a la regla en Ruleset Name.
   c) Active el enforcement status
   d) Asegúrese de aplicar la regla en su rama main
   e) Finalmente, en rules, deje las configuraciones por default y agregue "Require a pull request before merging"
3. Clone el repositorio de su compañero localmente. Esto lo puede hacer desde GitHub Desktop en File>Clone Repository y siguiendo las instrucciones para clonar. Como se muestra en la siguiente imagen:

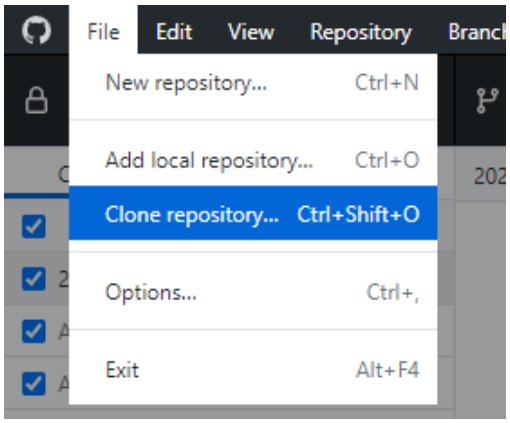

Figura 5: Clonar un repositorio en GitHub Desktop.

4. Sobre este nuevo repositorio, en VSCode cree (Create Branch) y haga Checkout de una rama.
5. Sobre esta rama realice modificaciones al archivo de su compañero.
6. Publique la rama en el repositorio remoto.
7. Use el comando `git log` para mostrar el estado actual del repositorio, donde se debe observar la rama sobre la que está trabajando. Tome un pantallazo e incluyalo en su reporte.
8. En GitHub, en el repositorio de su compañero, cree un Pull Request en la que se solicite que se agregue su rama a la rama main. Tome un pantallazo e incluyalo en su reporte.
9. En GitHub, vaya ahora a su repositorio y verifique que su compañero haya solicitado el Pull Request.

10. Verifique los cambios y acéptelos. Revise que los cambios sean visibles en GitHub.
11. Vuelva a su repositorio localmente y realice un pull.
12. Use el comando `git log` para mostrar el estado actual del repositorio, donde se debe observar la rama main actualizada con los cambios realizados por su compañero en la rama incluida a través del Pull Request. Tome un pantallazo e incluyalo en su reporte.