Tarea 1 Big Data - Documentación Detallada

Estudiante: Marco Ferraro

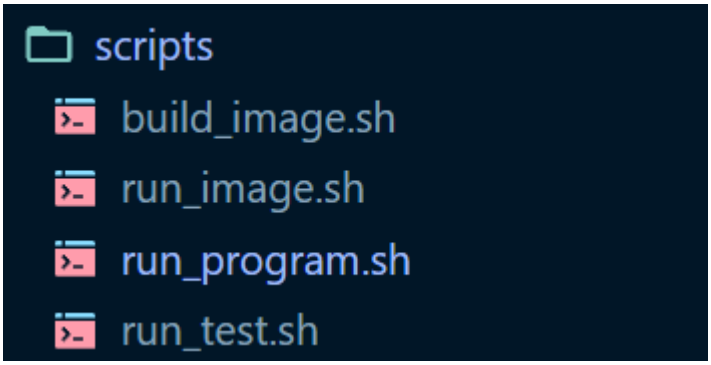
Objetivo del Documento

El propósito de este documento es proporcionar una guía detallada del proceso y los archivos necesarios para llevar a cabo la tarea relacionada con Big Data. Se detallarán los pasos para configurar el ambiente, ejecutar el programa y realizar pruebas unitarias.

Estructura de Archivos

Carpeta: scripts

Esta carpeta contiene varios scripts que facilitan diferentes aspectos del proceso. A continuación, se describen cada uno de ellos:



1. `build_image.sh` Descripción: Este script se encarga de construir la imagen del contenedor Docker con la configuración necesaria.
2. `run_image.sh` Descripción: Contiene las instrucciones para iniciar el contenedor con la imagen construida.
3. `run_program.sh` Descripción: Incluye las instrucciones para ejecutar el programa dentro del contenedor. Debe ser invocado dentro del ambiente.
4. `run_test.sh` Descripción: Llama a las pruebas unitarias para garantizar el correcto funcionamiento del programa.

Carpeta: assets

Documentos relacionados con la implementation de la tarea (i.e. enunciado, etc)

Datos

Para que el programa funcione correctamente, se requieren tres archivos CSV ubicados en la raíz del directorio, y deben seguir las convenciones mencionadas en el enunciado de la tarea:

ciclista.csv

- Formato específico mencionado en el enunciado.

- Debe contener la información relevante sobre los ciclistas.

actividad.csv

- Formato específico mencionado en el enunciado.
- Debe contener la información sobre las actividades de los ciclistas.

ruta.csv

- Formato específico mencionado en el enunciado.
- Debe contener información sobre las rutas seguidas por los ciclistas.

Estos archivos son cruciales para el correcto funcionamiento del programa, ya que contienen los datos necesarios para realizar las operaciones descritas en la tarea.

Implementacion del programa

Archivos Principales

Archivo: main.py

- **Descripción:** Este archivo maneja la lógica de inicialización del programa. Para invocar al programa se usa el siguiente comando
- **Uso:**

```
spark-submit --master local[2] main.py actividad.csv ruta.csv ciclistas.csv
```

Archivo: controller.py

- **Descripción:** Contiene lógica relacionada con el control y la coordinación del flujo del programa.
- **Uso:** Este archivo se importa y utiliza en `main.py` para orquestar la ejecución del programa.

Archivo: functions.py

- **Descripción:** Todas las funciones de procesamiento de datos se encuentran en este archivo.
- **Uso:** Este archivo se importa en `controller.py` para acceder a las funciones de procesamiento de datos.

Visualización de Resultados

Después de ejecutar el programa, los resultados se mostrarán en la consola. Además, se generará una carpeta llamada `results` que contendrá información más detallada sobre los resultados. A continuación, se proporcionan instrucciones para acceder y explorar estos resultados:

```
None
bash-5.0# ls
Dockerfile           actividad.csv        conftest.py          image.png            ruta.csv
Manual De Usuario.md assets               controller.py        main.py              scripts
__pycache__          ciclista.csv         functions.py         results              test_functions.py
```

1. **Acceso a la Carpeta de Resultados:**
    - Utilice el siguiente comando para navegar a la carpeta de resultados:

```
cd results
```

2. **Exploración de Resultados:**
    - Dentro de la carpeta `results`, encontrará un archivo con un identeficador que tendra los resultados en csv. Puede explorar estos archivos según sus necesidades.

3. **Comandos Útiles:**
    - Utilice comandos estándar de sistema operativo para visualizar o manipular archivos. Algunos ejemplos son:
        - Visualizar el contenido de un archivo de texto:

```
cat nombre_archivo.csv
```

        - Abrir un archivo con un editor de texto (por ejemplo, nano, vim, o emacs):

```
nano nombre_archivo.csv
```

        - Explorar archivos en un directorio:

```
ls
```

        - Moverse hacia atrás al directorio principal:

```
cd ..
```

```
bash-5.0# cd results/
bash-5.0# ls
_SUCCESS                                              part-00000-3d89a21f-9ba5-40c9-b83a-64ebe9c90c73-c000.csv
bash-5.0# cat part-00000-3d89a21f-9ba5-40c9-b83a-64ebe9c90c73-c000.csv
Alajuela,Sofía Jiménez,79.90,39.95
Alajuela,Silvia Solano,69.74,34.87
Alajuela,Mónica Castro,56.30,28.15
Alajuela,Óscar Sánchez,52.10,52.10
Alajuela,Marco Ferraro,50.20,50.20
Cartago,Pablo Navarro,91.30,45.65
Cartago,Laura Quesada,87.10,43.55
Cartago,Maria Fernández,69.74,34.87
Cartago,Hugo Chaves,33.20,33.20
Guanacaste,Diego Sánchez,100.40,50.20
Guanacaste,Juan Mora,83.20,41.60
Guanacaste,Sandra Céspedes,41.50,41.50
Guanacaste,Javier Ureña,28.60,28.60
Heredia,Roberto Chacón,100.40,50.20
Heredia,Carlos Rodríguez,83.20,41.60
Heredia,Amanda Castillo,56.30,28.15
Heredia,Natalia Zamora,45.04,45.04
Heredia,Feliciano Arce,23.70,23.70
Limón,Isabel Chaves,87.10,43.55
Limón,Manuel Vargas,60.50,30.25
```

Ejecución de Pruebas Unitarias

Para ejecutar las pruebas unitarias, utilice el siguiente comando en la terminal:

```
pytest test_functions.py -v
```

Este comando utilizará el archivo `test_functions.py`, que contiene 5 pruebas unitarias con funcionalidades y lógica de procesamiento de datos. Los resultados y un informe detallado se mostrarán en la consola. A continuación, se proporciona una descripción detallada de cómo interpretar los resultados de las pruebas unitarias:

1. **Ejecución del Comando:**
    - Al ejecutar el comando mencionado, pytest buscará y ejecutará las pruebas definidas en el archivo `test_functions.py`.
2. **Resultados en Consola:**
    - Cada prueba unitaria se ejecutará y mostrará su resultado en la consola. Los resultados pueden incluir "éxito", "falla" o "error". Asegúrese de revisar cualquier mensaje de error para identificar posibles problemas en la implementación.
3. **Informe Detallado:**
    - Después de ejecutar todas las pruebas, pytest proporcionará un informe detallado que incluye estadísticas sobre el número de pruebas ejecutadas, éxitos, fallos y errores. Esta información se mostrará al final de la salida en consola.
4. **Interpretación de Resultados:**
    - Un resultado exitoso de una prueba se indicará con un mensaje como "ok" o "passed". Por otro lado, un fallo o error proporcionará información adicional sobre la prueba que falló y la razón del

fallo.

5. **Recomendaciones:**
    - Si una prueba falla, revise el código correspondiente en `functions.py` y realice las correcciones necesarias. Vuelva a ejecutar las pruebas para asegurarse de que todas pasen correctamente.

Ejecutar pruebas unitarias es una práctica fundamental para garantizar la calidad y la confiabilidad del código. Asegúrese de abordar cualquier problema identificado durante las pruebas y realice ajustes según sea necesario en la implementación de las funciones en `functions.py`.