Procesamiento de datos a gran escala (26 de Noviembre, 2024)

Participantes: Johan Yesid Ramírez Torres, Néstor Iván Castellanos Merchán

Resumen: En el presente documento se realiza la respectiva documentación en Formato IEEE del punto B de laboratorio 2, el fin de este documento es presentar la metodología de dicho ejercicio.

I. INTRODUCCIÓN

En el presente formato encontraremos la solución del Punto B de laboratorio 2. El presente punto es referente al proceso de base de datos de entrada y salida de usuarios según corresponda.

II. ANÁLISIS

Se tomará en cuenta la programación modular como parte de la estructura para el desarrollo de la codificación, seguido de esto en la documentación una serie de pasos para lograr su respectiva solución al punto B de laboratorio 2 con la metodología de resolución de problemas.

- Contexto: Se realiza el punto B de laboratorio que es consistente en uso de logs como base de datos de usuarios que se le medirán entrada y salidas del personal.
- Población: Para usuarios que deseen tener un análisis del personal por ejemplo de empleados de una empresa y realizar gestión de entrada y salida del mismo.
- Limitación y Alcance: Usuarios interesados en implementar la gestión de ingreso del personal.

PROGRAMACION MODULAR

Figura 1: ProgramacionModular (Imagen fuente)

En la figura 1 se muestra la el mapa de programacion modular, la cual sera la estructura incial para trabajar sobre el ejercico B de laboratorio 2.

III. OBJETIVOS

- General:
  Realizar la documentación correspondiente a la codificación del punto B con su respectiva solución.

- Específicos:
  - Presentar un código preciso y sencillo de entender.
  - Presentar cada fase de la metodología en cuanto su documentación y la de codificación.
  - Presentar respectivos cambios en la codificación para dar con los códigos respectivos para la solución del ejercicio.

IV. CODIFICACIÓN

A. *Tipos de aplicación*

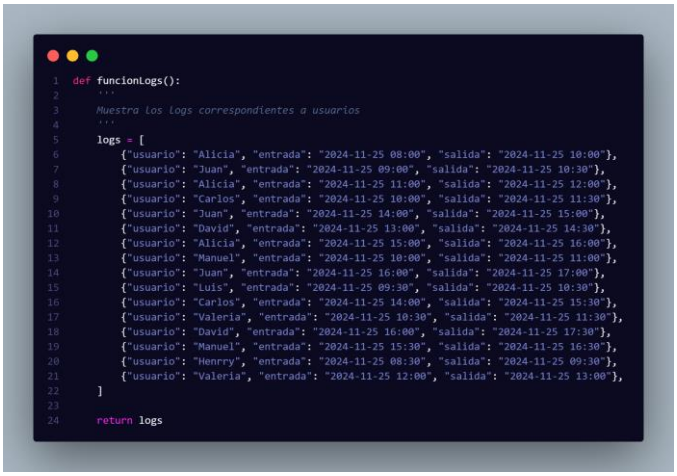
En el presente software a realizar se tendrá en cuenta en la fase de Análisis de la Programación Modular y el uso de los bucles while y for conjunto a las listas.

B. *Instrucciones de uso*

- Para dar con la solución del ejercicio vamos a utilizar los logs, que serían semejantes a los datos de los usuarios he ira registrada entrada y salida.

- En la codificación se utilizará ciclos for para saber los ingresos de cada usuario.

Python:
En el presente lenguaje de programación “Python” se dará la realización del Punto B de laboratorio 2, se utilizarán funciones, logs y bucles para su respectiva solución.



```python
def funcionLogs():
    '''
    Muestra los logs correspondientes a usuarios
    '''
    logs = [
        {"usuario": "Alicia", "entrada": "2024-11-25 08:00", "salida": "2024-11-25 10:00"},
        {"usuario": "Juan", "entrada": "2024-11-25 09:00", "salida": "2024-11-25 10:30"},
        {"usuario": "Alicia", "entrada": "2024-11-25 11:00", "salida": "2024-11-25 12:00"},
        {"usuario": "Carlos", "entrada": "2024-11-25 10:00", "salida": "2024-11-25 11:30"},
        {"usuario": "Juan", "entrada": "2024-11-25 14:00", "salida": "2024-11-25 15:00"},
        {"usuario": "David", "entrada": "2024-11-25 13:00", "salida": "2024-11-25 14:30"},
        {"usuario": "Alicia", "entrada": "2024-11-25 15:00", "salida": "2024-11-25 16:00"},
        {"usuario": "Manuel", "entrada": "2024-11-25 10:00", "salida": "2024-11-25 11:00"},
        {"usuario": "Juan", "entrada": "2024-11-25 16:00", "salida": "2024-11-25 17:00"},
        {"usuario": "Luis", "entrada": "2024-11-25 09:30", "salida": "2024-11-25 10:30"},
        {"usuario": "Carlos", "entrada": "2024-11-25 14:00", "salida": "2024-11-25 15:30"},
        {"usuario": "Valeria", "entrada": "2024-11-25 10:30", "salida": "2024-11-25 11:30"},
        {"usuario": "David", "entrada": "2024-11-25 16:00", "salida": "2024-11-25 17:30"},
        {"usuario": "Manuel", "entrada": "2024-11-25 15:30", "salida": "2024-11-25 16:30"},
        {"usuario": "Henrry", "entrada": "2024-11-25 08:30", "salida": "2024-11-25 09:30"},
        {"usuario": "Valeria", "entrada": "2024-11-25 12:00", "salida": "2024-11-25 13:00"},
    ]

    return logs
```

Figura 2: Codificacion_Python1 (Imagen fuente)

- En la figura 2 encontraremos la función logs, en esta encontramos la información de los usuarios donde se implementaron un total de 15 para la realización de la información.

```python
def procesarUsuarios(logs):
    '''
    Procesa los logs y extrae los usuarios
    '''

    usuariosProcesados = []
    for registro in logs:
        usuariosProcesados.append(registro["usuario"])
    return usuariosProcesados

def contarNumAccesos(usuariosProcesados):
    '''
    Cuenta cuántos accesos tiene cada usuario
    '''

    resultados = []
    usuarios_unicos = set(usuariosProcesados)
    for usuario in usuarios_unicos:
        numeroDeAccesos = usuariosProcesados.count(usuario)
        resultados.append([usuario, numeroDeAccesos])
    return resultados

def imprimirResultados(resultados):
    '''
    Imprime los resultados del conteo de accesos
    '''

    print("\nTotal de veces que usuarios ingresaron:")
    for resultado in resultados:
        print(f"Usuario: {resultado[0]}, Número de accesos: {resultado[1]}")
```

Figura 3: Codificacion_Python2 (Imagen fuente)

- En la figura 3 encontramos los procesos para:
    1. Extraer cada usuario.
    2. Contar los ingresos.
    3. Imprimir resultados.

Para extraer cada usuario utilizaremos ciclos for para ingresarlos a otra lista denominada usuariosProcesados.

Para el conteo de ingresos se tendrá en cuenta las listas con ayuda de los for y almacenarlos en una lista y verificar sus ingresos con .count donde estas iran almacenadas a variable resultado.

Por último imprimir cada los resultados obtenidos en las funciones anteriores con resultados e imprimirlas en un print.

```python
import os
os.system('cls')

from logs import funcionLogs
from processUsuarios import procesarUsuarios, contarNumAccesos, imprimirResultados

def mostrar_menu():
    '''
    Muestra el menu principal para el usuario
    '''

    print("\nMenú de opciones:")
    print("1. Procesar los usuarios y contar accesos")
    print("2. Mostrar resultados de accesos")
    print("3. Salir")

def menu():
    '''
    Funcion del menu principal
    '''

    usuariosProcesados = []
    resultados = []

    while True:
        mostrar_menu()
        opcion = input("Elija una opción (1-3): ")

        if opcion == '1':
            # Obtener los logs y procesar los usuarios
            logs = funcionLogs()  # Llamamos a la función que retorna los logs
            usuariosProcesados = procesarUsuarios(logs)
            resultados = contarNumAccesos(usuariosProcesados)
            print("\nDatos procesados con éxito.")

        elif opcion == '2':
            # Mostrar resultados de accesos
            if resultados:
                imprimirResultados(resultados)
            else:
                print("\nPrimero debe procesar los datos (opción 1).")

        elif opcion == '3':
            # Salir del programa
            print("Hasta pronto.")
            break

        else:
            print("\nOpción no válida. Por favor, elija una opción entre 1 y 3.")

# Llamar a menu
if __name__ == "__main__":
    menu()
```

Figura 4: Codificacion_Python3 (Imagen fuente)

- En la figura 4 encontraremos el interfaz para el usuario, siendo este que toda la codificación se lleve a cabo, donde se importan los 2 archivos que ayudan a su funcionalidad utilizando los from e import.

Adicional se le importa el sistema operativo con fin de que únicamente a la hora de ejecutar el respectivo programa ingrese lo que se ha codificado, con el propósito de verse mas limpio. Para la utilización de este mimo se ingresa os.system(‘cls’’).

V. Requerimientos del sistema

- Para teléfono:
  Android: Versión 10
  RAM: 2GB
  Almacenamiento: 500Mb

- Para Ordenador:
  Windows 7 o superior
  RAM: 4GB
  Almacenamiento: 500Mb

VI. Diseño del algoritmo

En el presente punto se da a conocer el diseño de algoritmo teniendo presente su proceso y explicación, en concreto con la programación modular donde esta misma es sutil en cuanto la estructura en la codificación.

VII. Ejecución del programa

En la correspondiente ejecución del programa se establece sin novedad su funcionamiento como tal, siendo este logrando conseguir que se logre de antemano su finalidad bajo los parámetros.

```
Menú de opciones:
1. Procesar los usuarios y contar accesos
2. Mostrar resultados de accesos
3. Salir
Elija una opción (1-3): 1

Datos procesados con éxito.

Menú de opciones:
1. Procesar los usuarios y contar accesos
2. Mostrar resultados de accesos
3. Salir
Elija una opción (1-3): 2

Total de veces que usuarios ingresaron:
Usuario: Juan, Número de accesos: 3
Usuario: Manuel, Número de accesos: 2
Usuario: David, Número de accesos: 2
Usuario: Henrry, Número de accesos: 1
Usuario: Valeria, Número de accesos: 2
Usuario: Luis, Número de accesos: 1
Usuario: Alicia, Número de accesos: 3
Usuario: Carlos, Número de accesos: 2

Menú de opciones:
1. Procesar los usuarios y contar accesos
2. Mostrar resultados de accesos
3. Salir
Elija una opción (1-3): 3
Hasta pronto.
```

Figura 5: EjecucionPython1 (Imagen fuente)

- En la figura 5 encontramos la ejecución del programa.
  Se muestra el menú del usuario con 3 opciones.

1. Procesar los usuarios y contar accesos.
2. Mostrar resultados de accesos.
3. Salir.
4.

Este menú es totalmente funcional donde el mismo puede ser ejecutado si el usuario que lo este utilizando ingrese un símbolo no correspondiente se pueda repetir sin inconvenientes.

VIII. DEPURACION – DOCUMENTACION Y MANTENIMIENTO

En los presentes modelos de la codificación se han realizado sus respectivos cambios en la codificación hasta para poder dar con el código actual, se le ha dado al código sus respectivos tests para su solución y verificación de su funcionalidad.

1. VERSION 1.0:
   En la Versión 1.0 se realizó una codificación de borrador para dar con la solución e implantación de los logs de manera directa sin la implementación de las funciones.

```python
logs = [
    {"usuario": "Alicia", "entrada": "2024-11-25 08:00", "salida": "2024-11-25 10:00"},
    {"usuario": "Juan", "entrada": "2024-11-25 09:00", "salida": "2024-11-25 10:30"},
    {"usuario": "Alicia", "entrada": "2024-11-25 11:00", "salida": "2024-11-25 12:00"},
    {"usuario": "Carlos", "entrada": "2024-11-25 10:00", "salida": "2024-11-25 11:30"},
    {"usuario": "Juan", "entrada": "2024-11-25 14:00", "salida": "2024-11-25 15:00"},
]

# Lista para almacenar los usuarios procesados
usuarios_procesados = []

# Usamos un índice para simular el procesamiento secuencial
indice = 0
while indice < len(logs):
    registro = logs[indice]
    usuarios_procesados.append(registro["usuario"])
    indice += 1

# Contar el número de accesos por usuario
resultados = []
usuarios_unicos = set(usuarios_procesados)

for usuario in usuarios_unicos:
    numero_de_accesos = usuarios_procesados.count(usuario)
    resultados.append([usuario, numero_de_accesos])

# Imprimir los resultados
print("Resultados del procesamiento de logs:")
for resultado in resultados:
    print(f"Usuario: {resultado[0]}, Número de accesos: {resultado[1]}")
```

Figura 6: Version1.0 Codificación (Imagen fuente)

TEST VERSION 1.0:
T1:
En el test de la versión 1.0 tenemos el resultado de la codificación de dicha versión, donde el programa arroja los resultados exigidos por el ejercicio B de laboratorio.

```
Resultados del procesamiento de logs:
Usuario: Carlos, Número de accesos: 1
Usuario: Alicia, Número de accesos: 2
Usuario: Juan, Número de accesos: 2
```

Figura 7: Version1.0 Test (Imagen fuente)

2. VERSION 1.1:
   En la versión 1.1 se busca realizar un código más completo, donde haya más usuarios a los que se van a imprimir y por lo tanto para que la codificación no sea tan obsoleta se le añade un pequeño menú que cumple de todas formas con los requerimientos del ejercicio y puede el mismo seguir funcionando a pesar de ingresar opciones no reconocidas en el menú.

```python
def mostrar_menu():
    '''
    Muestra el menu principal para el usuario
    '''

    print("\nMenú de opciones:")
    print("1. Procesar los usuarios y contar accesos")
    print("2. Mostrar resultados de accesos")
    print("3. Salir")

def menu():
    '''
    Funcion del menu principal
    '''

    usuariosProcesados = []
    resultados = []

    while True:
        mostrar_menu()
        opcion = input("Elija una opción (1-3): ")

        if opcion == '1':
            # Obtener los logs y procesar los usuarios
            logs = funcionLogs()  # Llamamos a la función que retorna los logs
            usuariosProcesados = procesarUsuarios(logs)
            resultados = contarNumAccesos(usuariosProcesados)
            print("\nDatos procesados con éxito.")

        elif opcion == '2':
            # Mostrar resultados de accesos
            if resultados:
                imprimirResultados(resultados)
            else:
                print("\nPrimero debe procesar los datos (opción 1).")

        elif opcion == '3':
            # Salir del programa
            print("Hasta pronto.")
            break

        else:
            print("\nOpción no válida. Por favor, elija una opción entre 1 y 3.")
```

Figura 8: Version1.1 Codificación (Imagen fuente)

TEST VERSION 1.1:
T1:
Acorde al respectivo menú integrado se realiza un test de los errores y verificación de que al momento de ser impreso siga en funcionamiento.

```
Menú de opciones:
1. Procesar los usuarios y contar accesos
2. Mostrar resultados de accesos
3. Salir
Elija una opción (1-3): Enayo error

Opción no válida. Por favor, elija una opción entre 1 y 3.

Menú de opciones:
1. Procesar los usuarios y contar accesos
2. Mostrar resultados de accesos
3. Salir
Elija una opción (1-3): 4

Opción no válida. Por favor, elija una opción entre 1 y 3.
```

Figura 9: Versión 1.1 Test (Imagen fuente)

IX. CONCLUSIÓN

- En la respectiva documentación se da por finalizado el formato IEEE referente al punto c de laboratorio 2, respectiva documentación se encuentra la programación modular y codificación completa dada en el lenguaje de programación de Python, donde se maneja simulación de logs en el programa.