SISTEMA DE MONITOREO DE SENSORES TÉRMICOS (Noviembre de 2024)

Autores: Torres Pachón Juan Esteban, González Gonzáles Dinalut y Ruiz Rincon Jessica Jineth estudiantes de la foundation universitaria de San gil unisangil (SEDE-CHIQUINQUIRA)

***Resumen*-En el presente documento se ha diseñado un algoritmo capaz de detectar y recolectar datos de un sistema de sensores térmicos de una fábrica.**

Palabras clave-Sistemas, Monitoreo, Detección, Recolección.

INTRODUCCIÓN

En el presente documento se busca desarrollar un algoritmo capaz de interactuar con la información otorgada por los sensores térmicos la cual, no permitirá saber que valores bajo ciertas condiciones debemos darles importancia.

ANÁLISIS

A. *Contexto*

Una fábrica solicitó a un algoritmo que trabajara de la mano con una red de sensores ubicados de manera estratégica, para la captura de las variaciones térmicas de la fábrica.

B. Población

Fábricas las cuales necesitan un sistema para monitorear la temperatura mediante sensores.

C. Limitaciones y alcance

Las limitaciones y alcances que se pudieron identificar son las siguientes:

Limitaciones

- El usuario portador de este sistema debe tener una fábrica para implementarlo.
- El sistema de sensores tiene actividades limitadas las cuales son: Detectar y recolectar información.

Alcances

- El usuario podrá ingresar los datos requeridos.
- El usuario puede visualizar los cambios térmicos en su fábrica.
- El usuario puede imprimir reportes de los cambios generados, en un rango temporal establecido.

III. OBJETIVOS

Objetivo general

Diseñar un sistema de monitoreo mediante sensores térmicos.

B. *Objetivos específicos*

- Determinar el lenguaje de programación que se va a usar para crear el algoritmo.
- Identificar las áreas donde se implementará el sistema.
- Generar la salida correspondiente de los datos.
- Presentar al usuario un programa que pueda manejar de forma comprensible.

IV. ESPECIFICACIÓN DE REQUERIMIENTOS.

B. *CON RESPECTO AL USUARIO.*

Requisitos mínimos

- *El usuario debe tener una instalación a la cual sea factible la adición de sensores térmicos.*

- *El usuario debe contar con una necesidad puntual la cual es: gestionar la temperatura de algunas partes de la fábrica.*

V. DISEÑO DEL ALGORITMO

Se utilizó la aplicación generadora de pseudocódigo denominado PseInt la cual permitió una guía de cómo el producto final debía ser ofrecido al usuario final.

```
Definir matrixSensores Como Lista
Definir matrixTemps Como Lista
Definir row, column Como Entero
row ← 4
column ← 4

Escribir "Monitoreo de temperatura mediante sensores."
Escribir "Medición de valores de temperatura:"
Escribir " - Bajos 0 > T < 10°"
Escribir " - Medios ⩾ 10° T < 25°"
Escribir " - Altos 25° ⩾ T < 55°"
Escribir " - Críticos T ⩾ 55°"

Subproceso createMatrixRandomValues(row, column)
    Definir z, w, randValue Como Entero
    Definir values Como Lista

    Para z ← 0 Hasta row - 1 Hacer
        values ← CrearLista()  // Lista vacía para cada fila

        Para w ← 0 Hasta column - 1 Hacer
            randValue ← Aleatorio(0, 100)
            Agregar values, ConvertirATexto(randValue) + " °C"
        Fin Para

        Agregar matrixSensores, values
    Fin Para
Fin Subproceso

Subproceso show(matrix)
    Definir z, w Como Entero
    Para z ← 0 Hasta Longitud(matrix) - 1 Hacer
        Escribir "F", z + 1, ": "

        Para w ← 0 Hasta Longitud(matrix[z]) - 1 Hacer
            Si w % 2 = 0 Entonces
                Escribir "| ", matrix[z][w], " |", Sin Saltar
            Sino
                Si w = Longitud(matrix[z]) - 1 Entonces
                    Escribir " ", matrix[z][w], " |", Sin Saltar
                Sino
                    Escribir " ", matrix[z][w], " ", Sin Saltar
                Fin Si
            Fin Si
        Fin Para
        Escribir ""  // Salto de línea después de cada fila
    Fin Para
Fin Subproceso
```

```
Subproceso search()
    Definir z, w, down, middle, high, critical, number Como Entero
    Definir chain Como Cadena
    down ← 0
    middle ← 0
    high ← 0
    critical ← 0

    Para z ← 0 Hasta Longitud(matrixSensores) - 1 Hacer
        Para w ← 0 Hasta Longitud(matrixSensores[0]) - 1 Hacer
            chain ← ""
            number ← matrixSensores[z][w]

            // Extraer el número eliminando "°C"
            Para cada caracter En number Hacer
                Si caracter ≠ "°" Y caracter ≠ "C" Entonces
                    chain ← chain + caracter
                Fin Si
            Fin Para

            chain ← ConvertirANumero(chain)

            Si chain ≥ 0 Y chain < 10 Entonces
                down ← down + 1
            Sino Si chain ≥ 10 Y chain < 25 Entonces
                middle ← middle + 1
            Sino Si chain ≥ 25 Y chain < 55 Entonces
                high ← high + 1
            Sino Si chain ≥ 55 Entonces
                critical ← critical + 1
            Fin Si
        Fin Para
    Fin Para

    Agregar matrixTemps, down
    Agregar matrixTemps, middle
    Agregar matrixTemps, high
    Agregar matrixTemps, critical
Fin Subproceso

Subproceso printText()
    Escribir "Valores totales detectados:"
    Escribir "[", matrixTemps[0], "] → valores Bajos."
    Escribir "[", matrixTemps[1], "] → Valores Medios."
    Escribir "[", matrixTemps[2], "] → Valores Altos."
    Escribir "[", matrixTemps[3], "] → Valores Criticos."
Fin Subproceso

// Ejecución principal
createMatrixRandomValues(row, column)
show(matrixSensores)
search()
printText()
```

Imágenes. Pseudocódigo. Autoría: Chat GPT. OPEN IA.

VI.CODIFICACIÓN.

Para la codificación del algoritmo utilizamos los siguientes programas:

- PSeInt.
- Visual Studio Code.
- python
- Git hub.

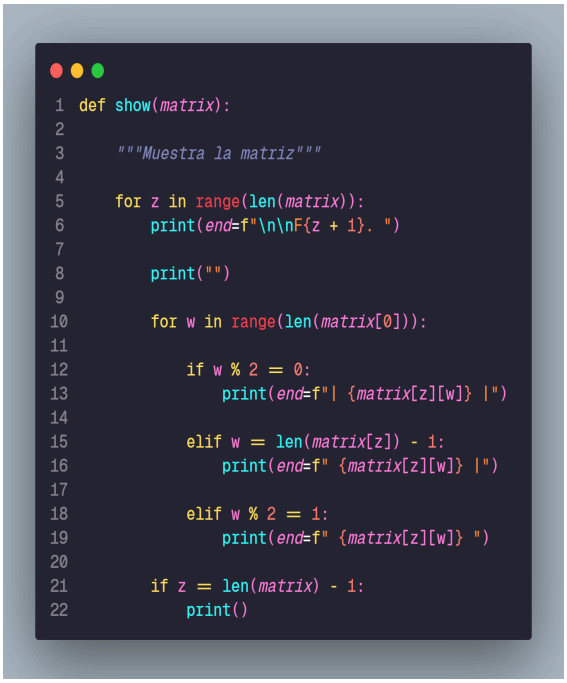
Instrucciones del código fuente

A. Código fuente

Dentro del código fuente trabajamos con scripts donde cada script se conectaba con una función a continuación se anexan imágenes del código seccionado por módulos que complementan el script

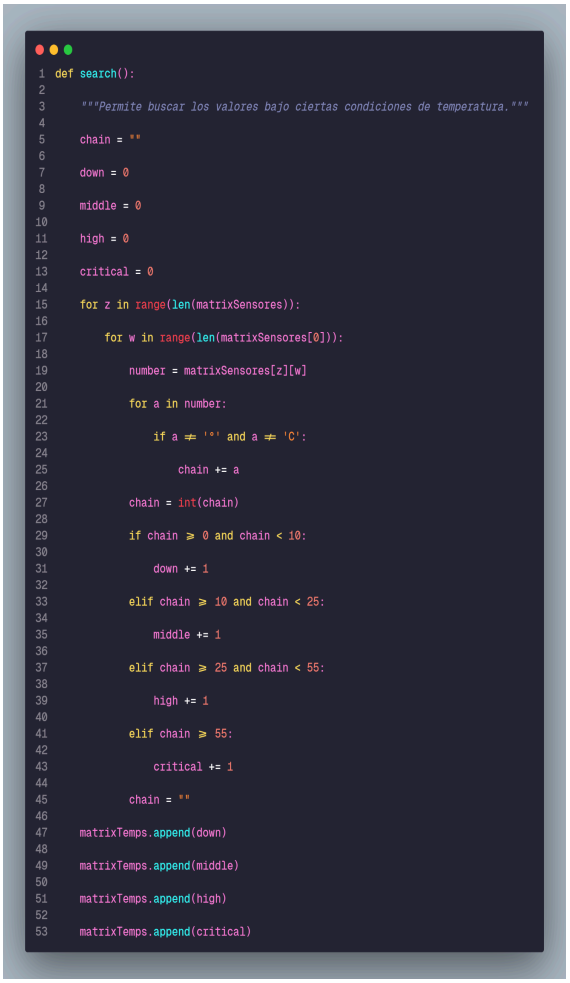
```python
def createMatrixRandomValues(row, column):

    """Función encargada de generar una matriz n x m y
    llenar las posiciones con valores aleatorios."""

    for z in range(row):

        values = []

        for w in range(column):

            randValue = random.randint(0, 100)

            newValue = str(randValue) + " °C"

            values.append(newValue)

        matrixSensores.append(values)
```

Imagen. 1. Código fuente. Autoría Propia.



```python
def show(matrix):

    """Muestra la matriz"""

    for z in range(len(matrix)):
        print(end=f"\n\nF{z + 1}. ")

        print("")

        for w in range(len(matrix[0])):

            if w % 2 == 0:
                print(end=f"| {matrix[z][w]} |")

            elif w == len(matrix[z]) - 1:
                print(end=f" {matrix[z][w]} |")

            elif w % 2 == 1:
                print(end=f" {matrix[z][w]} ")

        if z == len(matrix) - 1:
            print()
```

Imagen. 2. Código fuente. Autoría propia.



```python
def search():

    """Permite buscar los valores bajo ciertas condiciones de temperatura."""

    chain = ""

    down = 0

    middle = 0

    high = 0

    critical = 0

    for z in range(len(matrixSensores)):

        for w in range(len(matrixSensores[0])):

            number = matrixSensores[z][w]

            for a in number:

                if a != '°' and a != 'C':

                    chain += a

            chain = int(chain)

            if chain >= 0 and chain < 10:

                down += 1

            elif chain >= 10 and chain < 25:

                middle += 1

            elif chain >= 25 and chain < 55:

                high += 1

            elif chain >= 55:

                critical += 1

            chain = ""

    matrixTemps.append(down)

    matrixTemps.append(middle)

    matrixTemps.append(high)

    matrixTemps.append(critical)
```

Imagen. 3. Código fuente. Autoría propia.

Imagen. 4. Código fuente. Autoría Propia.

VII. EJECUCIÓN DEL PROGRAMA.

En este caso comenzamos con la apertura de la terminal de Git Bash. Si ya se creó el repositorio remoto y se sincronizan las cuentas de los desarrolladores con el repositorio local, se envía el archivo al Staging área con el comando git add. Luego ejecutamos el git commit -m “Comentario” para confirmar los cambios, donde, posteriormente usamos git push origin o pb master o main dependiendo la rama que se esté usando con el fin de almacenar la información en el repositorio remoto de GitHub.

VIII. VERIFICACIÓN Y DEPURACIÓN

El versionamiento inicio desde la versión -V 0.1, inició el versionamiento con 4 funciones específicas las cuales permiten, crear la matriz de valores, Mostrar la matriz de valores creada respecto a las condiciones de temperatura que se designaron en los requerimientos del código las cuales son: Valores térmicos, Bajos, Medios, Altos, Críticos. Por último tenemos la función de imprimir la cual nos sirve para observar la cantidad de valores de los tipos de temperaturas, que se generaron en la red matricial de valores aleatorios, representando estas temperaturas de la fábrica.

Se pensó en un ordenamiento sencillo de las funciones, las cuales pudiésemos identificar cada actividad que para los programadores resulta necesario cambiar y/o, optimizar, respecto a los errores y mejoras subyacentes que pueden surgir en el proceso de reprogramación.

VERSIÓN -V 0.1:

1. En la versión -V 0.1 se pensó de forma ágil, el objetivo principal fue crear funciones lo suficientemente explícitas y puntuales capaces de poder realizar cambios y/o, optimizaciones, de forma eficiente

como se logra apreciar en las Imágenes: 1, 2 y 3.

A. Interna.

Toda la información relacionada con el desarrollo del código se encuentra almacenada en un repositorio en GitHub.
Enlace:
https://github.com/Jessica-Ruiz/Laboratorio2.git

Modelos de análisis:
Como modelo de análisis podemos identificar que se utilizó la programación modular para analizar el problema

Manual de usuario.

Para el manual de usuario, al usuario se le provee una red de sensores térmicos con estructura matricial. El portal de usuario será breve pues al encargado de monitorear las temperaturas solo tendrá que observar el rango de valores y validar cuántos valores están presentes en temperaturas: Bajas, Medias, Altas, Críticas.

VIII CONCLUSIONES.

- Implementamos un análisis situacional específico para captar con mayor detalle los requisitos del cliente,el pudimos utilizar para escribir el código de manera correcta

- Incorporamos la evaluación y pruebas para el programa, con ellas constatamos y verificamos que el código realizado por nosotros los desarrolladores esté a fin con los requerimientos del cliente.

- Aprendimos que es necesario aplicar una metodología que hace más comprensible y efectivo el proceso a la hora de desarrollar un algoritmo.

- Empleamos un generador de pseudocódigo denominado PSeInt como guía para estructurar de una forma clara el algoritmo en Python.

- Para realizar la codificación del código utilizamos varios programas como Visual Studio Code y GitHub los cuales facilitan el trabajo en grupo y la gestión de información.