




PROYECTO 1

ESTRUCTURAS



Autores:
Gabriel Jaramillo Cuberos
Salomón Ávila Larrotta
Tomas Silva
Juan Pabón
Ángel Morales



de datos



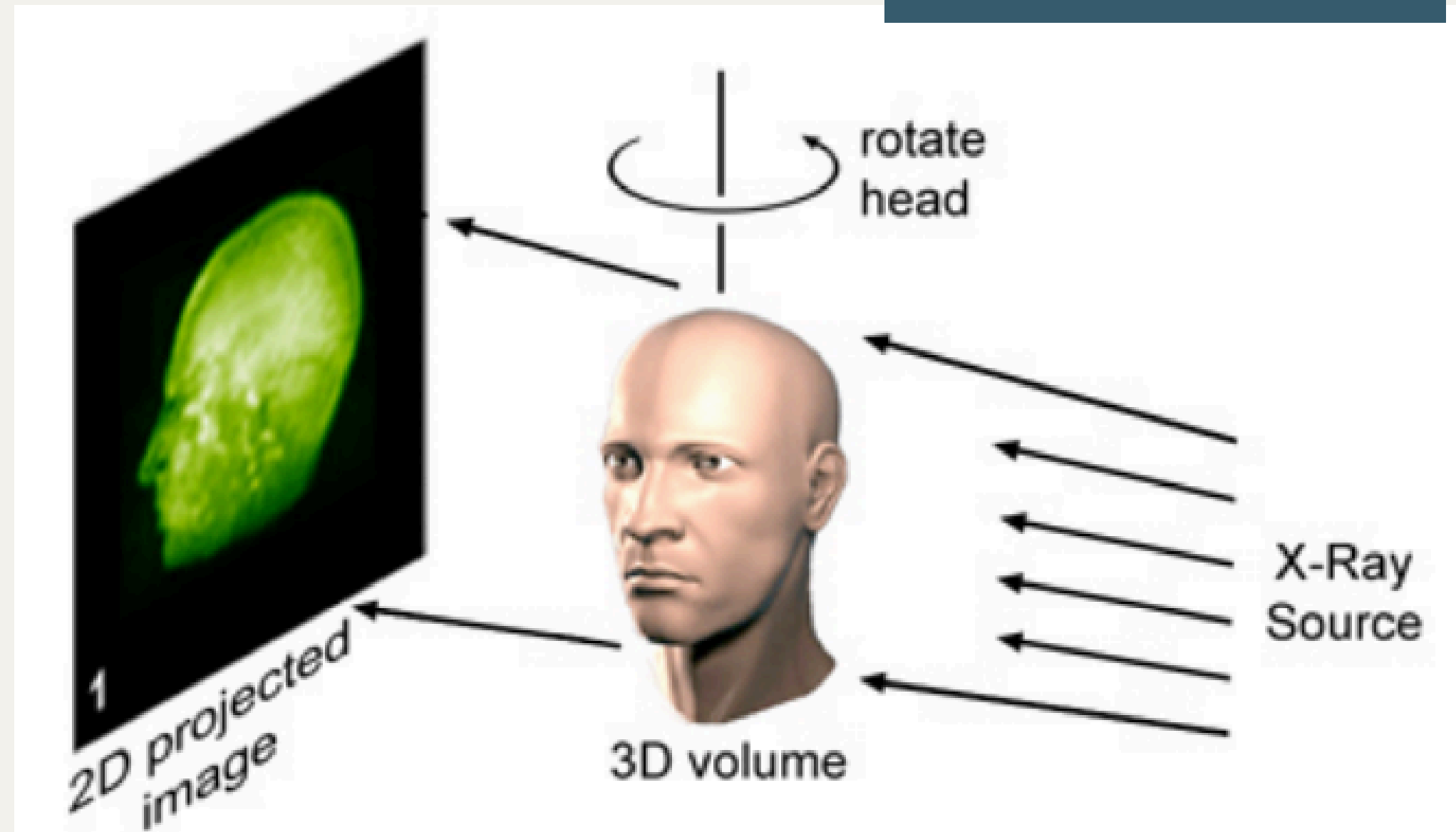


ÍNDICE

1. **Introducción**
2. **TADS**
3. **Diagrama de relacion**
4. **Main**
5. **Plan de pruebas**
6. **Conclusiones**

INTRODUCCIÓN

El proyecto es un sistema que busca procesar imágenes en escala de grises. Este sistema integra tres componentes: proyección 2D de volúmenes de imágenes, codificación y decodificación mediante el algoritmo de Huffman, y segmentación de imágenes usando grafos. Esta entrega se centra en la proyección de imágenes a partir de una serie ordenada de imágenes que conforman un volumen 3D.



ión del proceso de generación de una proyección planar por medio de ray
(www.cs.cmu.edu/classes/ESM4714/methods/FuncExt.html)

TADS

Imagen



TAD Imagen

-Datos minimos:

---vecImagen: vector de vectores de enteros, representa cada píxel de la imagen.

---MaxClaro: entero, representa el máximo de la intensidad de un pixel.

---DimensionX: entero, define el ancho de la imagen.

---DimensionY: entero, define la altura de la imagen.

---Formato: cadena de caracteres, dice el tipo de formato de la imagen.

---Nombre: cadena de caracteres, dice el nombre de la imagen.

-Operaciones:

---getVecImagen(): retorna el vector de vectores de vecImagen.

---setVecImagen(vecImg): asigna a vecImagen un vector de vectores de enteros.

---getMaxClaro(): retorna el valor máximo de claridad (brillo) permitido en la imagen.

---setMaxClaro(maximo): asigna al valor de "maxClaro" un valor entero que entra de parámetro.

---getDimensionX(): retorna el entero de dimensionX.

---setDimensionX(dimension): asigna al valor de "dimensionX" un valor entero que entra de parámetro.

---getDimensionY(): retorna el entero de DimensionY.

---setDimensionY(dimension): asigna al valor de "dimensionY" un valor entero que entra de parámetro.

---getFormato(): retorna el string de Formato.

---setFormato(form): asigna al valor de "Formato" un string que entra de parámetro.

---getNombre(): retorna el string de Nombre.

---setNombre(nombre2): asigna al valor de "Nombre" un string que entra de parámetro.

---cargarDesdePGM(ruta): carga imagen .PMG usando una ruta de archivo.

---guardarComoPGM(ruta): guarda imagen como .PMG a una ruta específica.

---mostrarInfo(): muestra la información relacionada a la Imagen.

TADS

Volumen de imágenes



TAD VolumenImágenes

-Datos mínimos:

---CantidadImágenes: entero, define la cantidad de imágenes que se van a cargar

---Imágenes: vector de imágenes, contiene las imágenes ordenadas según el orden de la imagen en el archivo

---NombreBase: cadena de caracteres, nombre base de la colección de imágenes

-Operaciones:

---GetCantidadImágenes(): retorna el número de cantidad de imágenes

---GetImágenes(): retorna el vector de imágenes

---SetCantidadImágenes(Cantidad): asigna al valor de "CantidadImágenes" un valor entero que entra de parámetro.

---SetImágenes(Imágenes): asigna al vector de vectores "Imágenes" un vector de mismo tipo que entra como parámetro de entrada.

---SetNombre(nombre_base): asigna un nombre a la colección de imágenes.

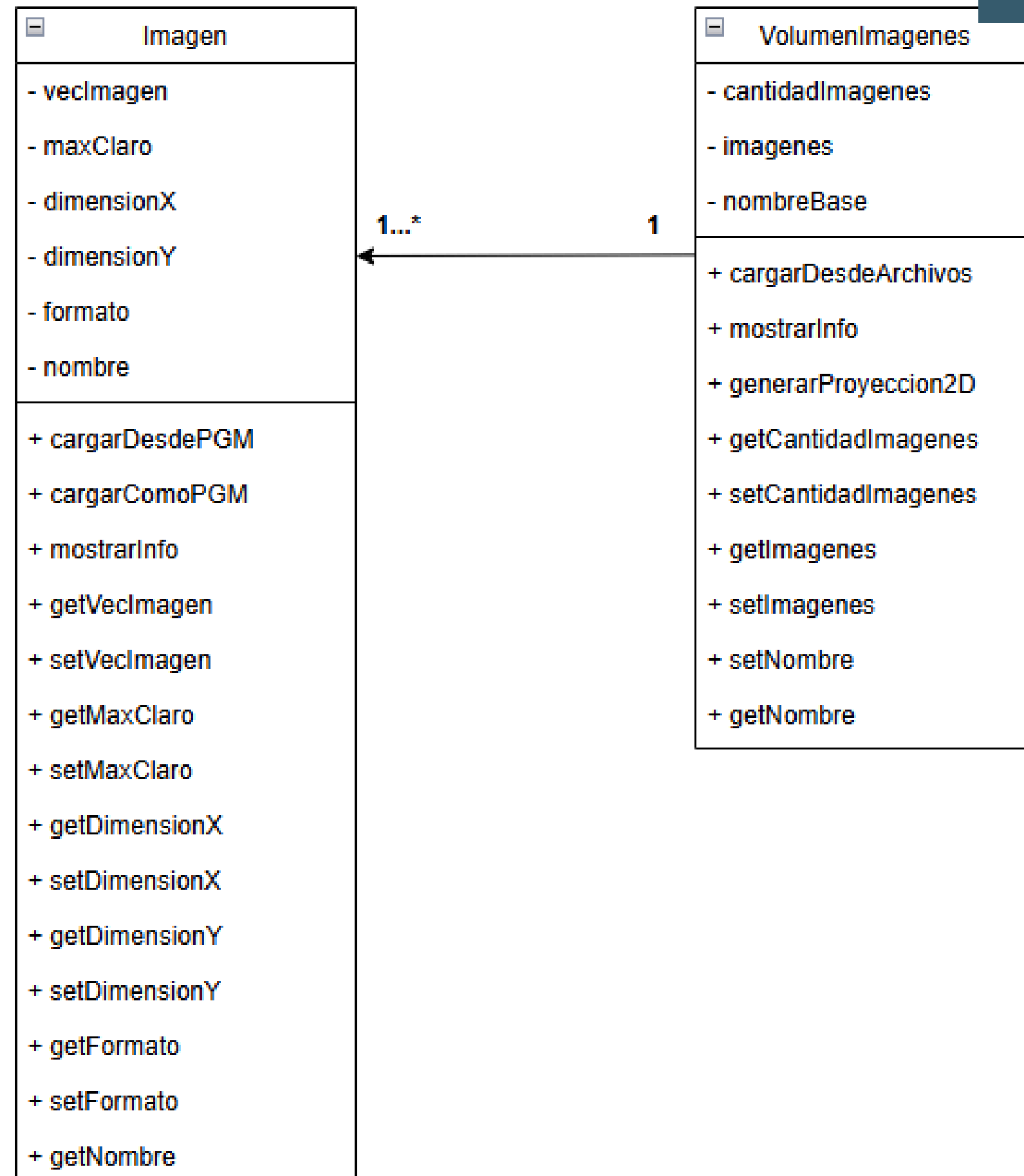
---GetNombre(): retorna el nombre de la colección de imágenes.

---CargarDesdeArchivos(base, numImágenes): Carga las imágenes desde la ruta e indica la cantidad de imágenes que se cargaron.

---MostrarInfo(): Muestra la información relacionada a la base y el número de imágenes cargadas.

---GenerarProyeccion2D(dirección, criterio, rutaFinal): Se genera una proyección **2D** del conjunto de imágenes dado la dirección de vista y el criterio especificado.

DIAGRAMA DE RELACIÓN



MÓDULOS

Imagen.h	Representacion de una imagen
Imagen.cpp	Definicion de metodos de representacion de una imagen
VolumenImagenes.h	Representacion de un volumen de imagenes
VolumenImagenes.cpp	Definicion de volumen de imagenes junto con proyeccion2D
Menu.h	Representacion de un sistema gestor de imagenes
menu.cpp	Implementacion de funciones de gestion de imagenes

```

// Declaración del TAD "Imagen":
class Imagen {
private:
    // Matriz que almacena los valores de los píxeles de la imagen en escala de grises
    vector<vector<int>> vecImagen;

    // Valor máximo de claridad en la imagen (escala de grises 0-255)
    int maxClaro;

    // Dimensiones de la imagen en píxeles
    int dimensionX; // Ancho de la imagen
    int dimensionY; // Alto de la imagen

    // Formato de la imagen (por ejemplo, "P2" para PGM ASCII)
    string formato;

    // Nombre de la imagen (nombre del archivo)
    string nombre;

public:
    // Constructor: inicializa la imagen con valores por defecto
    Imagen();

    // Retorna la matriz de píxeles de la imagen
    vector<vector<int>> getVecImagen();

    // Asigna una nueva matriz de píxeles a la imagen
    void setVecImagen(vector<vector<int>> vecImg);

    // Retorna el valor máximo de claridad (brillo) permitido en la imagen
    int getMaxClaro();

    // Asigna el valor máximo de claridad de la imagen
    void setMaxClaro(int maxClaro);

    // Retorna el ancho de la imagen en píxeles
    int getDimensionX();

    // Asigna el ancho de la imagen en píxeles
    void setDimensionX(int dimX);

    // Retorna el alto de la imagen en píxeles
    int getDimensionY();

    // Asigna el alto de la imagen en píxeles
    void setDimensionY(int dimY);

    // Retorna el formato de la imagen (por ejemplo, "P2" para PGM)
    string getFormato();

    // Asigna el formato de la imagen
    void setFormato(string &form);

    // Retorna el nombre del archivo de la imagen
    string getNombre();

    // Asigna el nombre del archivo de la imagen
    void setNombre(string &nombre);

    // Carga una imagen en formato PGM desde una ruta específica
    void cargarDesdePGM(string &ruta);

    // Guarda la imagen en formato PGM en una ruta específica

```

```

}
int Imagen::getDimensionY()
{
    return dimensionY;
}
int Imagen::getDimensionX()
{
    return dimensionX;
}
void Imagen::setFormato(string &form)
{
    formato = form;
}
void Imagen::cargarDesdePGM(string &nombre)
{
    string rutaDeArchivo = "src/";
    rutaDeArchivo += nombre;
    setNombre(nombre);
    ifstream archivo;
    archivo.open(rutaDeArchivo);
    if (!archivo.is_open())
    {
        cout << "La imagen " << rutaDeArchivo << " no pudo ser cargada" << endl;
        return;
    }
    if (archivo.is_open())
    {
        string linea;
        archivo >> linea;
        setFormato(linea);
        archivo >> linea;
        setDimensionX(stoi(linea));
        archivo >> linea;
        setDimensionY(stoi(linea));
        archivo >> linea;
        setMaxClaro(stoi(linea));
        for (int i = 0; i < dimensionY; i++)
        {
            vector<int> vi;
            vecImagen.push_back(vi);
            for (int j = 0; j < dimensionX; j++)
            {
                int tmp;
                archivo >> tmp;
                vecImagen[i].push_back(tmp);
                cout<<tmp<<endl;
            }
        }
        archivo.close();
        if (!vecImagen.empty())
        {
            cout << "La imagen " << nombre << " ha sido cargada" << endl;
        }
        else
        {
            cout << "La imagen " << nombre << " no ha podido ser cargada" << endl;
        }
    }
    string salida2 = "salida.pgm";
    guardarComoPGM(salida2);
}

string Imagen::getFormato()
{
    return formato;
}

int Imagen::getMaxClaro()
{
    return maxClaro;
}
void Imagen::guardarComoPGM(string &ruta)
{
    ofstream archivo;
    archivo.open(ruta);
    if (archivo.is_open())
    {
        cout<<"P2\n";
        cout<<getDimensionX()<<"\n";
        cout<<getDimensionY()<<"\n";
        cout<<getMaxClaro()<<"\n";
        archivo << getFormato() << "\n";
    }
}

```


MAIN

La consolidacion del sistema utilizando todos los modulos anteriormente mencionados y el procesamiento de el input de usuario para el correcto uso del sistema.

Interfaz de usuario por consola mediante comandos guiados.

```
#include <bits/stdc++.h>
#include "TADS/Imagen.h"
#include "TADS/VolumenImagenes.h"
#include "Utils/Menu.h"
using namespace std;
int main(){
    Imagen imagen;
    VolumenImagenes volumen;
    cout << "Bienvenido al sistema de manejo de imágenes de la Pontificia Universidad Javeriana" <<
endl;

    string opt;
    while (true){
        menu();
        getline(cin, opt);
        stringstream ss;
        ss << opt;
        vector<string> argumentos;
        while(ss){
            string temp;
            ss >> temp;
            argumentos.push_back(temp);
        }
        argumentos.pop_back();
        if(argumentos.size() > 0){
            if(argumentos[0] == "cargar_imagen" && argumentos.size() == 2){
                string tmp = argumentos[1];
                imagen.cargarDesdePGM(tmp);
            }
            else if(argumentos[0] == "cargar_volumen" && argumentos.size() == 3){
                volumen.cargarDesdeArchivos(argumentos[1], stoi(argumentos[2]));
            }
            else if(argumentos[0] == "info_imagen" && argumentos.size() == 1){
                imagen.mostrarInfo();
            }
            else if(argumentos[0] == "info_volumen" && argumentos.size() == 1){
                volumen.mostrarInfo();
            }
            else if(argumentos[0] == "proyeccion2D" && argumentos.size() == 4){
                volumen.generarProyeccion2D(argumentos[1], argumentos[2], argumentos[3]);
            }
            else if(argumentos[0] == "salir" && argumentos.size() == 1){
                cout << "Hasta luego" << endl;
                return 0;
            }
            else{
                cout << "Comando incorrecto" << endl;
            }
        }
    }

    return 0;
}
```

PLAN DE PRUEBAS

Prueba	Resultado esperado	Resultado real
Info_imagen (con imagen cargada)	Imagen cargada en memoria: imagen.pgm, ancho: W, alto: H.	
Info_imagen (sin imagen cargada)	No hay una imagen cargada en la memoria.	

Prueba	Resultado esperado	Resultado real
Info_volumen (con volumen cargado)	Volumen cargado en memoria: volumen, tamaño: n_im, ancho: W, alto: H.	
Info_volumen (sin volumen cargado)	No hay un volumen cargado en memoria.	

Carga imagenes

Carga volumenenes

Info imagenes

Info volumenenes

Proyeccion 2D

Prueba	Resultado esperado	Resultado real
cargar_imagen nombre_imagen.pgm	La imagen imagen.pgm ha sido cargada.	
cargar_imagen nombre_imagen.png	La imagen imagen.pgm no ha podido ser cargada.	
cargar_imagen	Falta de argumentos.	
cargar_imagen imagen_inexistente.pgm	La imagen imagen.pgm no ha podido ser cargada.	
cargar_imagen nombre_imagen.pgm (después de haber cargado otra imagen)	La imagen imagen.pgm ha sido cargada. (Sobreescribe la anterior)	

Prueba	Resultado esperado	Resultado real
cargar_volumen volumen 10	El volumen “volumen” ha sido cargado.	
cargar_volumen volumen -5	La cantidad de imágenes no es válida.	
cargar_volumen volumen	Falta de argumentos.	
cargar_volumen volumen 200	La cantidad de imágenes supera el límite permitido.	
cargar_volumen volumen 10 (con una imagen faltante)	El volumen “volumen” no ha podido ser cargado.	
cargar_volumen volumen 10 (después de haber cargado otro volumen)	El volumen “volumen” ha sido cargado. (Sobreescribe el anterior)	

Prueba	Resultado esperado	Resultado real
proyeccion2D x maximo salida.pgm (con volumen cargado)	La proyección 2D del volumen en memoria ha sido generada y almacenada en el archivo salida.pgm.	
proyeccion2D a maximo salida.pgm	El volumen aún no ha sido cargado en memoria.	
proyeccion2D z promedio salida.pgm (sin volumen cargado)	Dirección no válida. Las opciones son: x, y, z.	
proyeccion2D x suma salida.pgm	Criterio no válido. Las opciones son: minimo, maximo, promedio, mediana.	
proyeccion2D x maximo	Falta de argumentos.	

CONCLUSIONES

Despues de una implementacion completa del sistema se identificaron los siguientes puntos clave que llevan a la conclusion de la entrega del proyecto:

- Integración de conocimientos teóricos y prácticos
- Modularidad y escalabilidad
- Eficiencia en el manejo de datos
- Cambios respecto a la entrega anterior
- Importancia de la interacción entre el usuario y el sistema



MUCHAS GRACIAS

Marzo 2025

