2022

Documentación MGUdb

Microbases de datos

Es un sistema para manejo de pequeñas bases de datos como datos de usuario en aplicaciones móviles, datos de configuración de aplicaciones, por ejemplo.

MGU
02/10/2022

Obtener y configurar

MGUdb es una librería para gestión de pequeñas base de datos, disponemos de una clase "MGUdb" desde donde se realizan todas las operaciones que requiera la base de datos.

Para usar el MGUdb primero debemos descargarlo de https://mau002g.github.io.

Una vez descargado tendremos un archivo .zip con un archivo .dll, un archivo .a y la carpeta "include".

Para instalar dependiendo de que compilador estés usando, debemos copiar la carpeta MGUdb que está dentro del include a la carpeta de includes del compilador.

Una vez hecho lo anterior no debería de haber problema al poner la línea "#include <MGUdb.hpp>", si hay algún error quizá no realizaste el paso anterior correctamente.

El archivo mgudb.dll lo colocas en la misma carpeta que el archivo.exe.

Finalmente, el archivo .a lo colocas en la lista de enlazado del compilador.

1)Crear una base de datos

Para crear una base de datos simplemente creamos un archivo vacio con cualquier extensión, por ejemplo, base.txt o datos.hsg.

Tambien puedes crear una base de datos desde la librería con la función Create del objeto MGUdb.

```
#include <MGUdb/MGUdb.hpp>

int main()

{

	mgu::MGUdb dtb;

	dtb.Create("miarchivo.txt");

	return 0;

}
```

Hay que tener en cuenta que la base de datos que se crea no se selecciona.

2)Seleccionando bases de datos

Para seleccionar bases de datos simplemente usamos la función UseDB

```
#include <MGUdb/MGUdb.hpp>

int main()

{

	mgu::MGUdb dtb;

	dtb.UseDB("nombre_base_datos.txt");

	return 0;

}
```

Esta función lo que hace es cargar los elementos de la base de datos, es en este punto donde pueden surgir errores si la integridad de los archivo esta comprometida o si el formato del archivo no es compatible. Esto selecciona la base de datos por lo que ya puedes modificar, crear, añadir datos a la base de datos.

3)Guardando el contenido

MGUdb trabaja en la memoria ram, por lo que debemos guardar en las modificaciones que hallamos hecho durante la ejecución.

```
#include <MGUdb/MGUdb.hpp>

int main()

{

	mgu::MGUdb dtb;

	dtb.UseDB(“nombre_base_datos.txt”); //Selecionamos

	dtb.CreateTag(“nombre”, std::string(“juan”); //Crea una etiqueta

	dtb.FlushDB(); //Guarda en el archivo lo que este en ram

	return 0;

}
```

La función toma lo que estaba en la memoria ram y lo almacena en el disco duro

4)Manejando etiquetas

Las etiquetas son elementos que pueden almacenar un solo dato con un tipo y un nombre.

El nombre de la etiqueta es lo que lo identifica en la base de datos, ¡no pueden haber dos etiquetas con el mismo nombre!.

Pero primero veamos como son los tipos de datos que maneja MGUdb:

- ENTERO
- DECIMAL

- CADENA
- BOOL

Como vemos son solo 4, entero para números sin decimales, decimal para numero con decimales, cadenas para los que sea y bool para valores booleanos.

MGUdb tiene unas reglas de almacenamiento un poco estrictas:

1. Puedes guardar cualquier cosa en formato de cadena, menos las palabra “true o false”.
2. Si la etiqueta tiene formato ENTERO no permitirá guardar cosa como esta. “texto”, “263y23”, “22.343”. Como vemos el tipo debe respetarse.
3. Las etiquetas en formato bool solo pueden guardar true o false y nada mas.

Ahora veamos como crearlas desde las librería:

Para crear una etiqueta se usa la función CreateTag(“nombre de etiqueta”, dato)

```
#include <MGUdb/MGUdb.hpp>

int main()

{

	mgu::MGUdb dtb;

	dtb.UseDB(“nombre_base_datos.txt”); //Selecionamos

	dtb.CreateTag(“nombre”, std::string(“juan”); //Crea una etiqueta tipo CADENA

	dtb.CreateTag(“edad”, 19);//Crea una etiqueta de tipo ENTERO

	dtb.FlushDB(); //Guarda en el archivo lo que este en ram

	return 0;

}
```

Para modificar datos de la etiqueta se usa la función ModTag(“nombre de etiqueta”, nuevo dato);

```
#include <MGUdb/MGUdb.hpp>

int main()

{
```

```
	mgu::MGUdb dtb;
	dtb.UseDB("nombre_base_datos.txt"); //Selecionamos
	dtb.CreateTag("edad",  3); //Crea una etiqueta tipo ENTERO
	dtb.Mod Tag("edad", 19);//Modifica el dato de edad
	dtb.FlushDB(); //Guarda en el archivo lo que este en ram
	return 0;
}
```

Debemos estar pendientes de que el tipo de dato sea el mismo que cuando se creó la etiqueta.

Como vemos las etiquetas usan los tipos de c++, para las etiquetas.

Ahora veamos como borrar una etiqueta, para borrar usamos la función DeleteTag("nombre de etiqueta")

```
#include <MGUdb/MGUdb.hpp>
int main()
{
	mgu::MGUdb dtb;
	dtb.UseDB("nombre_base_datos.txt"); //Selecionamos
	dtb.DeleteTag("edad");//Elimina la etiqueta edad
	dtb.FlushDB(); //Guarda en el archivo lo que este en ram
	return 0;
}
```

Ahora veamos como recuperar los datos desde una etiqueta

Tenermos 4 Funciones:

```
std::string getStringFromTag(std::string nombre);
 int getIntFromTag(std::string nombre);
```

```
        double getDecimalFromTag(std::string nombre);

        bool getBooleanFromTag(std::string nombre);
```

```
#include <MGUdb/MGUdb.hpp>

#include <iostream>


int main()

{

        mgu::MGUdb dtb;

        std::string dato;

        dtb.UseDB("nombre_base_datos.txt"); //Selecionamos

        dato = dtb.getStringFromTag("nombre"); //Vemos y guardamos en la variable
"dato" lo que contenia esa etiqueta

        dtb.FlushDB(); //Guarda en el archivo lo que este en ram

        return 0;

}
```

Como vemos los tipos de retorno debe coincidir con los de las etiquetas.

Manejando tablas