$-2233$

# STOCK MANAGEMENT TOOL 1.0

Generado por Doxygen 1.9.1

# Contents

# Chapter 1

# Gestión de Stock de Materiales

Esta aplicación en C permite gestionar un inventario de materiales, registrar su estado (operativo, en reparación, repuesto), y generar informes y copias de seguridad. Es ideal para un sistema de mantenimiento que necesita seguimiento detallado de los elementos.

## 1.1 Funcionalidades

1. **Agregar Materiales:**
   - Registra nuevos materiales con un ID único, nombre, estado y observaciones.
2. **Eliminar Materiales:**
   - Permite borrar materiales del sistema.
3. **Modificar Estado de Materiales:**
   - Actualiza el estado de los materiales (operativo, en reparación, repuesto) y añade observaciones.
4. **Generar Informe de Materiales en Reparación:**
   - Crea un archivo de texto con los detalles de los materiales en reparación, útil para su uso en correos electrónicos.
5. **Búsqueda de Materiales por ID:**
   - Permite consultar información de un material específico ingresando su ID.
6. **Generación de Copia de Seguridad en CSV:**
   - Exporta los datos a un archivo CSV para respaldo o análisis externo.
7. **Log de Actualizaciones con Fecha y Hora:**
   - Registra todas las modificaciones realizadas sobre los materiales para auditorías. (logUpdates.txt)

## 1.2 Compilación y Ejecución

Puedes compilar el proyecto usando GCC y ejecutar el programa desde la terminal. Aquí hay un ejemplo de cómo hacerlo:

```
1gcc -Wall -o gestor main.c -I ./includes ./lib/libGui.c ./lib/libPersist.c ./lib/libBaseUtils.c
1
1./gestor
```

Para configurar cantidad de materiales editar config.ini, no tiene que haber database.dat creado.

Para ingreso de materiales por lote, revisar formato en muestra-lote.txt : ID,NOMBRE_ELEMENTO,ESTADO(0=OPERATIVO,1=REPU
_REPARACION),OBSERVACIONES Y guardar en un archivo que se llame lote.txt

# Chapter 2

# Índice de estructura de datos

## 2.1 Estructura de datos

Lista de estructuras con una breve descripción:

# Chapter 3

# Indice de archivos

## 3.1 Lista de archivos

Lista de todos los archivos con descripciones breves:

# Chapter 4

# Documentación de las estructuras de datos

## 4.1 Referencia de la Estructura Elemento

`#include <baseUtils.h>`

### Campos de datos

- int id
- char nombre [31]
- Estado estado
- char observaciones [51]

### 4.1.1 Documentación de los campos

#### 4.1.1.1 estado

`Estado Elemento::estado`

#### 4.1.1.2 id

`int Elemento::id`

#### 4.1.1.3 nombre

```
char Elemento::nombre[31]
```

#### 4.1.1.4 observaciones

```
char Elemento::observaciones[51]
```

La documentación para esta estructura fue generada a partir del siguiente fichero:

- includes/baseUtils.h

# Chapter 5

# Documentación de archivos

## 5.1 Referencia del Archivo includes/baseUtils.h

Gráfico de los archivos que directa o indirectamente incluyen a este archivo:

## 5.2 Referencia del Archivo includes/gui.h

Gráfico de los archivos que directa o indirectamente incluyen a este archivo:

### Funciones

- int menuPrincipal ()

  *INTERFAZ DEL USUARIO.*
- void menuBorrar ()
- void menuEditar ()
- void menuAgregar ()
- void menuLeer ()
- void menuPendientes ()
- void actualizarEstado ()
- void menuOrdenar ()

  *Menu de ordenamiento con 3 opciones: ID, NOMBRE, ESTADO.*

### 5.2.1 Documentación de las funciones

#### 5.2.1.1 actualizarEstado()

```
void actualizarEstado ( )
```

#### 5.2.1.2 menuAgregar()

```
void menuAgregar ( )
```

#### 5.2.1.3 menuBorrar()

```
void menuBorrar ( )
```

#### 5.2.1.4 menuEditar()

```
void menuEditar ( )
```

#### 5.2.1.5 menuLeer()

```
void menuLeer ( )
```

#### 5.2.1.6 menuOrdenar()

```
void menuOrdenar ( )
```

Menu de ordenamiento con 3 opciones: ID, NOMBRE, ESTADO.

#### 5.2.1.7 menuPendientes()

```
void menuPendientes ( )
```

#### 5.2.1.8 menuPrincipal()

```
int menuPrincipal ( )
```

INTERFAZ DEL USUARIO.

## 5.3 Referencia del Archivo includes/persist.h

`#include "../includes/baseUtils.h"`
Dependencia gráfica adjunta para persist.h: Gráfico de los archivos que directa o indirectamente incluyen a este archivo:

### Funciones

- int guardarElemento (Elemento ∗)

  *Guarda el Elemento elem en la base de datos.*
- void leerElemento (int, Elemento ∗)

  *Recibe un ID, y un puntero de Elemento, busca el id en la base y lo devuelve en el puntero aux.*
- void borrarElemento (int)

  *Borra un elemento de la base de datos.*
- void generarInformePendientes ()

  *Genera un Informe con los elementos pendientes de reparacion.*
- void generaCopiaCSV ()

  *Genera un archivo CSV, copia de la base de datos actual.*
- void logUpdate (Elemento ∗, Estado)
- int ingresoPorLote ()

### 5.3.1 Documentación de las funciones

#### 5.3.1.1 borrarElemento()

```
void borrarElemento (
            int id )
```

Borra un elemento de la base de datos.

**Parámetros**

| | |
|---|---|
| *int* | id : ID del elemento a borrar. |

#### 5.3.1.2 generaCopiaCSV()

```
void generaCopiaCSV ( )
```

Genera un archivo CSV, copia de la base de datos actual.

#### 5.3.1.3 generarInformePendientes()

```
void generarInformePendientes ( )
```

Genera un Informe con los elementos pendientes de reparacion.

#### 5.3.1.4 guardarElemento()

```
int guardarElemento (
            Elemento * elem )
```

Guarda el Elemento elem en la base de datos.

**Parámetros**

| | |
|---|---|
| *Elemento* | elem: Recibe el elemento a guardar. |

**Devuelve**

Int : Devuelve 0 en caso de error, 1 en caso de exito.

#### 5.3.1.5 ingresoPorLote()

```
int ingresoPorLote ( )
```

#### 5.3.1.6 leerElemento()

```
void leerElemento (
            int id,
            Elemento * aux )
```

Recibe un ID, y un puntero de Elemento, busca el id en la base y lo devuelve en el puntero aux.

**Parámetros**

| | |
|---|---|
| *Int* | id : ID del elemento a buscar, Elemento * aux: Puntero del espacio de memoria para el elemento. |

#### 5.3.1.7 logUpdate()

```
void logUpdate (
```

```
            Elemento * elem,
            Estado viejoEstado )
```

## 5.4 Referencia del Archivo lib/libBaseUtils.c

```
#include "stdio.h"
#include "stdlib.h"
#include "time.h"
#include "string.h"
#include "stdarg.h"
#include "../includes/baseUtils.h"
```

Dependencia gráfica adjunta para libBaseUtils.c:

### Funciones

- int defineMax (char ∗configFile)

  *Si no existe database, seteo el max_elements segun config.ini .*
- void inicializarArchivo ()

  *Si el archivo no existe, lo crea con la totalidad de los registros para acceso directo.*
- int cargarVector (Elemento elem[ ])

  *Vuelca la base de datos al vector pasado por referencia.*
- int validaId (int id)

  *Valida si una ID existe en la base o no, recibe una id por parametro, y devuelve:*
- void timer (char mode, char ∗tiempo)

  *Genera Strings con la fecha/hora actual.*
- int miscanf (char tipo,...)

  *Funcion de ingreso de datos string(permite espacios en blanco) y enteros.*
- void parseElemento (Elemento ∗temp, char ∗cadena)
- int cmpNombre (const void ∗a, const void ∗b)
- int cmpEstado (const void ∗a, const void ∗b)

### Variables

- char ∗ ESTADOS [ ] = {"OPERATIVO", "REPUESTO", "EN REPARACION"}
- int MAX_ELEMENTOS

### 5.4.1 Documentación de las funciones

#### 5.4.1.1 cargarVector()

```
int cargarVector (
            Elemento elem[ ] )
```

Vuelca la base de datos al vector pasado por referencia.

**Parámetros**

| | |
|---|---|
| *Elemento* | elem[] : Puntero del vector a cargar con los elementos del archivo. |

**Devuelve**

Int : En caso satisfactorio, cantidad de elementos cargados, devuelve -1 en caso de error.

#### 5.4.1.2 cmpEstado()

```
int cmpEstado (
             const void * a,
             const void * b )
```

#### 5.4.1.3 cmpNombre()

```
int cmpNombre (
             const void * a,
             const void * b )
```

#### 5.4.1.4 defineMax()

```
int defineMax (
             char * configFile )
```

Si no existe database, seteo el max_elements segun config.ini .

#### 5.4.1.5 inicializarArchivo()

```
void inicializarArchivo ( )
```

Si el archivo no existe, lo crea con la totalidad de los registros para acceso directo.

#### 5.4.1.6 miscanf()

```
int miscanf (
             char tipo,
              ...  )
```

Funcion de ingreso de datos string(permite espacios en blanco) y enteros.

**Parámetros**

| | |
|---|---|
| *char* | * tipo : 'c' : para el ingreso de string, 'i': para el ingreso de enteros. |
| *void* | * segun modo elegido puntero correspondiente |
| *limite* | int: en caso de string : limite de caracteres |

**Devuelve**

int: 0: Error, !=0: Cantidad de caracteres o digitos.

#### 5.4.1.7 parseElemento()

```
void parseElemento (
            Elemento * temp,
            char * cadena )
```

#### 5.4.1.8 timer()

```
void timer (
            char mode,
            char * tiempo )
```

Genera Strings con la fecha/hora actual.

**Parámetros**

| | |
|---|---|
| *tiempo* | : puntero string con el espacio de memoria, |
| *mode* | : Opcion deseada: 'h':hora, 'd':dia, 'b': ambas. |

#### 5.4.1.9 validaId()

```
int validaId (
            int id )
```

Valida si una ID existe en la base o no, recibe una id por parametro, y devuelve:

**Parámetros**

| | |
|---|---|
| *Int* | id : ID a validar si existe o no. |

**Devuelve**

Int : -1: Error de lectura, 0: ID existente 1: ID válida.

### 5.4.2 Documentación de las variables

#### 5.4.2.1 ESTADOS

```
char* ESTADOS[] = {"OPERATIVO", "REPUESTO", "EN REPARACION"}
```

#### 5.4.2.2 MAX_ELEMENTOS

```
int MAX_ELEMENTOS
```

## 5.5 Referencia del Archivo lib/libGui.c

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "../includes/baseUtils.h"
#include "../includes/gui.h"
#include "../includes/persist.h"
```

Dependencia gráfica adjunta para libGui.c:

### Funciones

- int menuPrincipal ()

  *INTERFAZ DEL USUARIO.*
- void menuLeer ()
- void menuAgregar ()
- void menuBorrar ()
- void menuEditar ()
- void menuPendientes ()
- void actualizarEstado ()
- void menuOrdenar ()

  *Menu de ordenamiento con 3 opciones: ID, NOMBRE, ESTADO.*

### Variables

- char * ESTADOS [ ]
- int MAX_ELEMENTOS

### 5.5.1 Documentación de las funciones

#### 5.5.1.1 actualizarEstado()

```
void actualizarEstado ( )
```

#### 5.5.1.2 menuAgregar()

```
void menuAgregar ( )
```

#### 5.5.1.3 menuBorrar()

```
void menuBorrar ( )
```

#### 5.5.1.4 menuEditar()

```
void menuEditar ( )
```

#### 5.5.1.5 menuLeer()

```
void menuLeer ( )
```

#### 5.5.1.6 menuOrdenar()

```
void menuOrdenar ( )
```

Menu de ordenamiento con 3 opciones: ID, NOMBRE, ESTADO.

#### 5.5.1.7 menuPendientes()

```
void menuPendientes ( )
```

#### 5.5.1.8 menuPrincipal()

```
int menuPrincipal ( )
```

INTERFAZ DEL USUARIO.

### 5.5.2 Documentación de las variables

#### 5.5.2.1 ESTADOS

```
char* ESTADOS[]  [extern]
```

#### 5.5.2.2 MAX_ELEMENTOS

```
int MAX_ELEMENTOS  [extern]
```

## 5.6 Referencia del Archivo lib/libPersist.c

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "../includes/persist.h"
```

Dependencia gráfica adjunta para libPersist.c:

### Funciones

- int guardarElemento (Elemento *elem)

  *Guarda el Elemento elem en la base de datos.*
- void leerElemento (int id, Elemento *aux)

  *Recibe un ID, y un puntero de Elemento, busca el id en la base y lo devuelve en el puntero aux.*
- void borrarElemento (int id)

  *Borra un elemento de la base de datos.*
- void logUpdate (Elemento *elem, Estado viejoEstado)
- void generaCopiaCSV ()

  *Genera un archivo CSV, copia de la base de datos actual.*
- void generarInformePendientes ()

  *Genera un Informe con los elementos pendientes de reparacion.*
- int ingresoPorLote ()

### Variables

- char * ESTADOS [ ]
- int MAX_ELEMENTOS

### 5.6.1 Documentación de las funciones

#### 5.6.1.1 borrarElemento()

```
void borrarElemento (
            int id )
```

Borra un elemento de la base de datos.

**Parámetros**

| | |
|---|---|
| *int* | id : ID del elemento a borrar. |

#### 5.6.1.2 generaCopiaCSV()

```
void generaCopiaCSV ( )
```

Genera un archivo CSV, copia de la base de datos actual.

#### 5.6.1.3 generarInformePendientes()

```
void generarInformePendientes ( )
```

Genera un Informe con los elementos pendientes de reparacion.

#### 5.6.1.4 guardarElemento()

```
int guardarElemento (
            Elemento * elem )
```

Guarda el Elemento elem en la base de datos.

**Parámetros**

| | |
|---|---|
| *Elemento* | elem: Recibe el elemento a guardar. |

**Devuelve**

Int : Devuelve 0 en caso de error, 1 en caso de exito.

#### 5.6.1.5 ingresoPorLote()

```
int ingresoPorLote ( )
```

#### 5.6.1.6 leerElemento()

```
void leerElemento (
            int id,
            Elemento * aux )
```

Recibe un ID, y un puntero de Elemento, busca el id en la base y lo devuelve en el puntero aux.

**Parámetros**

| | |
|---|---|
| *Int* | id : ID del elemento a buscar, Elemento * aux: Puntero del espacio de memoria para el elemento. |

#### 5.6.1.7 logUpdate()

```
void logUpdate (
            Elemento * elem,
            Estado viejoEstado )
```

### 5.6.2 Documentación de las variables

#### 5.6.2.1 ESTADOS

```
char* ESTADOS[]  [extern]
```

#### 5.6.2.2 MAX_ELEMENTOS

```
int MAX_ELEMENTOS  [extern]
```

## 5.7 Referencia del Archivo main.c

```
#include "stdio.h"
#include "includes/baseUtils.h"
#include "includes/gui.h"
#include "includes/persist.h"
```

Dependencia gráfica adjunta para main.c:

### Funciones

- int main (int argc, char const ∗argv[ ])

### 5.7.1 Documentación de las funciones

#### 5.7.1.1 main()

```
int main (
            int argc,
            char const * argv[] )
```

## 5.8 Referencia del Archivo README.md