Manual Técnico del Sistema de Renta de Activos

Introducción

Este manual describe la implementación y el funcionamiento del sistema de "Renta de Activos". El sistema fue desarrollado en **C++20**, utilizando **Makefile** para la compilación y gestión del proyecto, y **Graphviz** para la generación de gráficas. El sistema permite a los administradores y usuarios interactuar con funcionalidades específicas orientadas a la gestión y renta de activos.

Tecnologías utilizadas

- **C++20**: Lenguaje principal de desarrollo.
- **Makefile**: Herramienta para la compilación automatizada del proyecto.
- **Graphviz**: Software para la creación de gráficas basadas en datos del sistema.

Funciones del administrador

El administrador tiene privilegios exclusivos en el sistema y puede realizar las siguientes tareas:

1. **Gestión de usuarios:**
    - Crear nuevos usuarios con sus respectivas credenciales.
    - Administrar permisos y datos básicos de los usuarios.
2. **Verificación de reportes:**
    - Consultar reportes en formato de matriz y árbol, representando la estructura y estado de los activos y usuarios.
3. **Generación de gráficas:**
    - Crear gráficas visuales con Graphviz para representar:
        - Departamentos y sus empresas.
        - Empresas, usuarios y sus activos.

Funciones del usuario

El sistema permite a los usuarios registrados realizar las siguientes operaciones:

1. **Inicio de sesión:**
    - Acceder al sistema con credenciales válidas.
2. **Gestión de activos:**

- Crear nuevos activos asociados a su cuenta.
- Modificar información de los activos existentes.
- Eliminar activos ya no requeridos.

3. **Renta de activos:**
    - Consultar la disponibilidad de activos para renta.
    - Realizar el proceso de renta de activos, incluyendo la generación de comprobantes si aplica.

Estructura del proyecto

El proyecto está organizado en varios módulos para mantener un diseño limpio y modular. Cada componente se encarga de una funcionalidad específica:

1. **Módulo de autenticación:**
    - Validación de credenciales para administradores y usuarios.
2. **Módulo de gestión de activos:**
    - CRUD (Crear, Leer, Actualizar, Eliminar) de activos.
3. **Módulo de reportes y gráficas:**
    - Generación de reportes en formato de matriz y árbol.
    - Creación de gráficas visuales con Graphviz.
4. **Módulo de renta:**
    - Gestión de la disponibilidad y renta de activos.

Compilación y ejecución

El proyecto utiliza un archivo **Makefile** para simplificar la compilación. A continuación, se describen los pasos:

1. **Compilación:** Ejecute el siguiente comando en la terminal:

   ```
   make
   ```

2. **Ejecución:** Una vez compilado, ejecute el binario generado:

   ```
   ./RenAct
   ```

3. **Limpieza:** Para eliminar archivos temporales o binarios generados, use:

   ```
   make clean
   ```

Uso de Graphviz

Graphviz se utiliza para representar datos complejos del sistema de manera visual. Las gráficas generadas incluyen:

1. **Departamentos y empresas:**
    - Representa la jerarquía entre departamentos y empresas.
2. **Usuarios y activos:**
    - Muestra la relación entre los usuarios, los activos registrados y su estado.

Ejemplo de un archivo DOT

El siguiente ejemplo ilustra cómo se generan las gráficas usando Graphviz:

```
digraph G {
    rankdir=LR;
    node [shape=box];

    Departamento1 -> Empresa1;
    Empresa1 -> Usuario1;
    Usuario1 -> Activo1 [label="Disponible"];
}
```

Para generar la gráfica, ejecute el comando:

`dot -Tpng -o grafica.png grafica.dot`

Seguridad

El sistema implementa medidas básicas de seguridad, como:

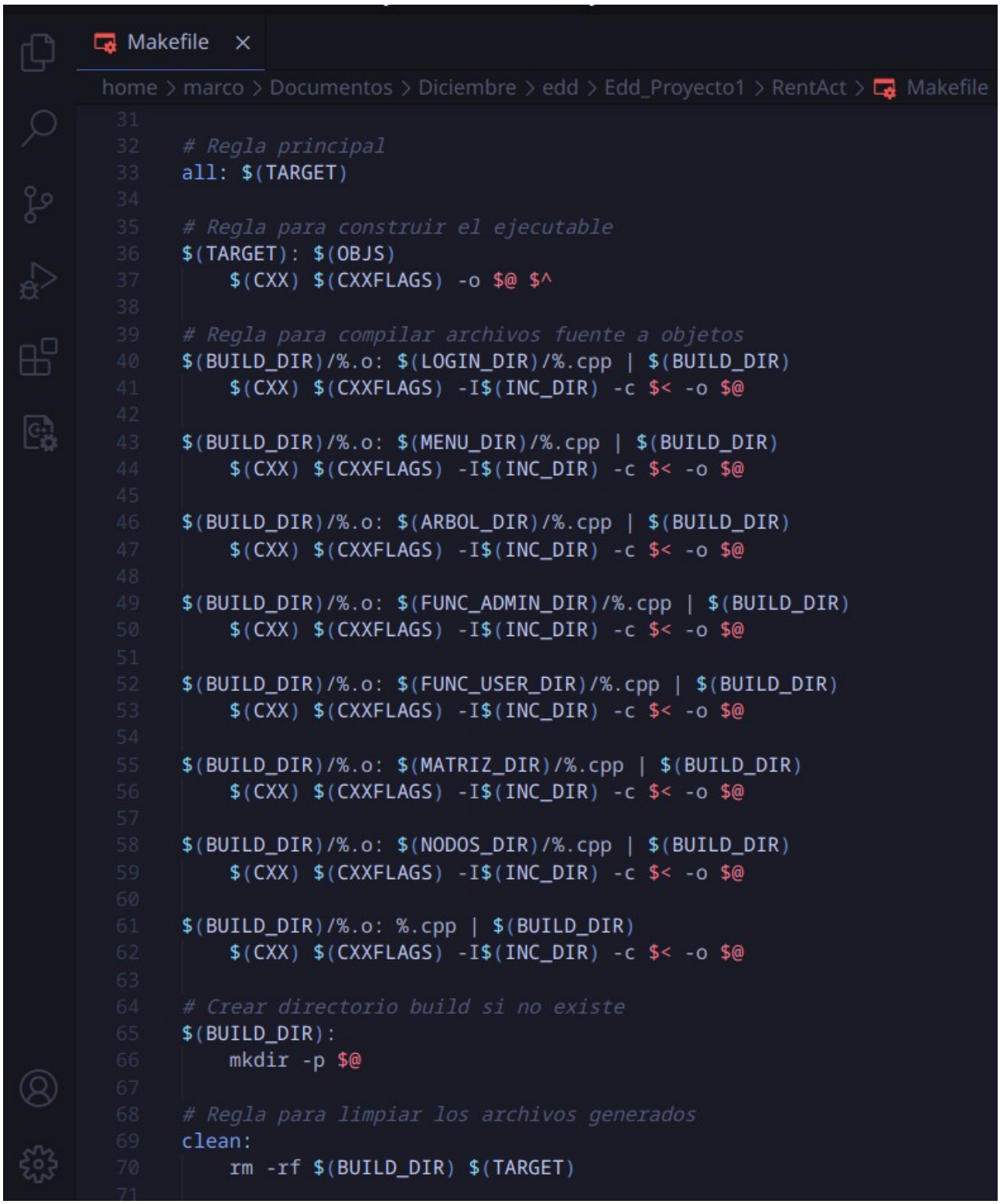
- **Autenticación por contraseña:**
    - El administrador tiene una contraseña única para acceder al sistema.
- **Validación de sesiones:**
    - Controla que los usuarios autenticados tengan acceso limitado según sus permisos.

Makefile

Makefile

home > marco > Documentos > Diciembre > edd > Edd_Proyecto1 > RentAct > Makefile

```
# Nombre del ejecutable
TARGET = RentAct

# Compilador y banderas
CXX = g++
CXXFLAGS = -std=c++20 -Wall -Wextra -O2

# Directorios
SRC_DIR = src
INC_DIR = include
BUILD_DIR = build
LOGIN_DIR = $(SRC_DIR)/login
MENU_DIR = $(SRC_DIR)/menu
ARBOL_DIR = $(SRC_DIR)/arbol
FUNC_ADMIN_DIR = $(SRC_DIR)/funciones_admin
FUNC_USER_DIR = $(SRC_DIR)/funciones_user
MATRIZ_DIR = $(SRC_DIR)/matriz
NODOS_DIR = $(SRC_DIR)/nodos

# Archivos fuente y objetos
SRCS = $(wildcard $(LOGIN_DIR)/*.cpp) \
       $(wildcard $(MENU_DIR)/*.cpp) \
       $(wildcard $(ARBOL_DIR)/*.cpp) \
       $(wildcard $(FUNC_ADMIN_DIR)/*.cpp) \
       $(wildcard $(FUNC_USER_DIR)/*.cpp) \
       $(wildcard $(MATRIZ_DIR)/*.cpp) \
       $(wildcard $(NODOS_DIR)/*.cpp) \
       main.cpp

OBJS = $(patsubst %.cpp,$(BUILD_DIR)/%.o,$(notdir $(SRCS)))

# Regla principal
all: $(TARGET)

```


Makefile
home > marco > Documentos > Diciembre > edd > Edd_Proyecto1 > RentAct > Makefile
31
32 # Regla principal
33 all: $(TARGET)
34
35 # Regla para construir el ejecutable
36 $(TARGET): $(OBJS)
37 $(CXX) $(CXXFLAGS) -o $@ $^
38
39 # Regla para compilar archivos fuente a objetos
40 $(BUILD_DIR)/%.o: $(LOGIN_DIR)/%.cpp | $(BUILD_DIR)
41 $(CXX) $(CXXFLAGS) -I$(INC_DIR) -c $< -o $@
42
43 $(BUILD_DIR)/%.o: $(MENU_DIR)/%.cpp | $(BUILD_DIR)
44 $(CXX) $(CXXFLAGS) -I$(INC_DIR) -c $< -o $@
45
46 $(BUILD_DIR)/%.o: $(ARBOL_DIR)/%.cpp | $(BUILD_DIR)
47 $(CXX) $(CXXFLAGS) -I$(INC_DIR) -c $< -o $@
48
49 $(BUILD_DIR)/%.o: $(FUNC_ADMIN_DIR)/%.cpp | $(BUILD_DIR)
50 $(CXX) $(CXXFLAGS) -I$(INC_DIR) -c $< -o $@
51
52 $(BUILD_DIR)/%.o: $(FUNC_USER_DIR)/%.cpp | $(BUILD_DIR)
53 $(CXX) $(CXXFLAGS) -I$(INC_DIR) -c $< -o $@
54
55 $(BUILD_DIR)/%.o: $(MATRIZ_DIR)/%.cpp | $(BUILD_DIR)
56 $(CXX) $(CXXFLAGS) -I$(INC_DIR) -c $< -o $@
57
58 $(BUILD_DIR)/%.o: $(NODOS_DIR)/%.cpp | $(BUILD_DIR)
59 $(CXX) $(CXXFLAGS) -I$(INC_DIR) -c $< -o $@
60
61 $(BUILD_DIR)/%.o: %.cpp | $(BUILD_DIR)
62 $(CXX) $(CXXFLAGS) -I$(INC_DIR) -c $< -o $@
63
64 # Crear directorio build si no existe
65 $(BUILD_DIR):
66 mkdir -p $@
67
68 # Regla para limpiar los archivos generados
69 clean:
70 rm -rf $(BUILD_DIR) $(TARGET)
71