



USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

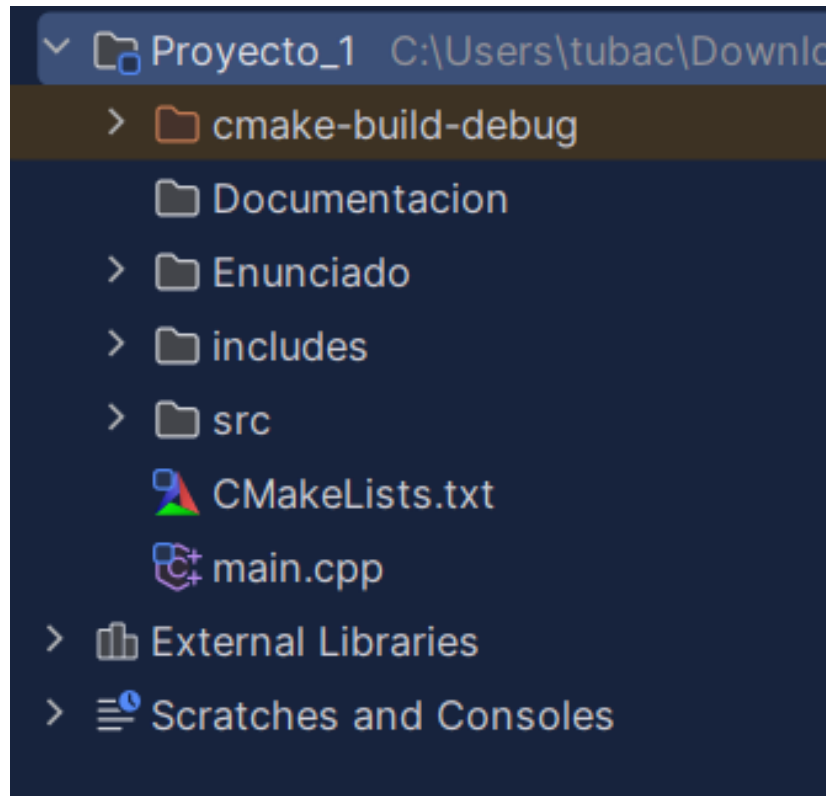
PROYECTO 1

MANUAL TÉCNICO

ESTRUCTURA DE DATOS

Miguel Adrian Tubac Agustin
202101927

Descripción de la solución



El programa resuelve el problema de un sistema de renta de activos entre usuarios. Dentro del sistema, se manejarán rentas, usuarios, un historial de transacciones y reportes. Todo el sistema es controlado mediante una aplicación en consola, trabajada con el lenguaje c++.

Lenguaje Utilizado:



Los requerimientos necesarios para la edición y ejecución del programa son la utilización del lenguaje de programación denominado C++, en el cual se crea la interfaz agradable con el usuario.



Clases

ClassAVL: en esta clase se definen las propiedades del árbol AVL, así mismo, con las funcionalidades de gráficas y modificación de los atributos.


```
ClassAVL::ClassAVL() {  
    this->raiz = nullptr;  
}
```

```
Nodo_AVL::Nodo_AVL(std::string id, Act  
    this->id = id;  
    this->activo = activo;  
    this->izq = nullptr;  
    this->der = nullptr;  
    this->factorEq = 0; //Inicializame  
    this->tiempoRenta = 0;  
}
```

Nodo_AVL: en esta clase se definen las propiedades de los nodos que integran el árbol AVL, en esta parte se inicializan los atributos de la clase.

Activos: en esta clase se definen las propiedades de los atributos del activo, este activo cuanta con las propiedades que describen el objeto almacenado en el mismo.

```
Activos::Activos(std::string nombre,  
    this->nombre = nombre;  
    this->descripcion = descripcion;  
}
```



```

Transacciones::Transacciones(std::string
    this->idActivo = idActivo;
    this->usuario = usuario;
    this->departamento = departamento;
    this->empresa = empresa;
    this->fecha = fecha;
    this->tiempoRenta = tiempoRenta;
}

```

Transacciones: en esta clase se definen las propiedades de los atributos que contendrá una transacción al momento de crear una.

Usuarios: en esta clase se definen las propiedades de los atributos que contendrá un atributo al momento de crear una instancia del mismo.

```

Usuarios::Usuarios(std::string
    this->usuar = usuar;
    this->password = password;
    this->nombre = nombre;
    this->arbol = arbol;
}


```

```

ListaEnlazadaDobleCircu::ListaEn
void ListaEnlazadaDobleCircu::agr
    Node3* newNode = new Node3(
    if (inicio == nullptr) {
        inicio = newNode;
        inicio->next = inicio;
        inicio->prev = inicio;
    } else {

```

Lista Curcular: en esta clase se definen las propiedades de la lista circular, así mismo como los métodos que se utilizan para graficar la lista.



MatrizDispersa: en esta clase se definen las propiedades de la matriz, así mismo como los métodos que se utilizan para graficar la matriz.

```
//Este es el constructor
MatrizDispersa::MatrizDispersa() {
    this->horizontal = nullptr;
    this->vertical = nullptr;
}

std::string nombre_usuario = "";
Usuarios *usar;
std::string nombre_departamento = ""
```

```
NodoMatriz::NodoMatriz(std::str
    this->cabecera = cabecera;
    this->valor = nullptr;

    this->siguiente = nullptr;
    this->anterior = nullptr;
```

NodoMatriz: en esta clase se definen las propiedades de los nodos de la matriz, en el cual se declaran los atributos que contiene un nodo de la matriz.

main: en esta clase se definen los menus de todo el sistema, así mismo como las creaciones de objetos pertenecientes a otras clases.

```
int main() {
    agregar_usuariosIniciales();
    int opcion = 0;

    do {
        cout << "\n-----"
        cout << "1. Iniciar Sesi
```

Métodos Y Funciones

```
void MatrizDispersa::insertarValor(Us
//Si esta vacia se incertan las d
NodoMatriz *cabeHorizontal = null
NodoMatriz *cabeVertical = nullptr
NodoMatriz *usuarioNuevo = new No

if (estaVacia()) {
```

insertarValor: este método se encuentra en la matriz y permite ingresar a un usuario dentro del sistema.

generarGrafica: con este método se crea la gráfica de la matriz dispersa, para esto toma en cuenta todos los casos que se encuentren en el mismo.

```
void MatrizDispersa::generarGrafica()
std::string dot = "digraph G {";
dot += "\n\tlabel=\"Matriz Disper";
dot += "\n\tlabelloc=\"t\"";
dot += "\n\tnode [shape=box width";

if (estaVacia()) {
    dot += "};";
    return;
```

```
void ClassAVL::insertar(Nodo
if (raiz == nullptr) {
    raiz = valor;
    raiz->factorEq = fact
    return;
}
```

insertar: en esta parte nos permite ingresar un nuevo activo al árbol avl, con las respectivas validaciones del mismo.