Manual técnico

Para esta aplicación de consola se utilizó las siguientes clases:

Nodo

```
class Nodo
{
public:
    Transaccion* value;
    Nodo* next;
    Nodo* back;

    Nodo(Transaccion* value, Nodo* next, Nodo* back) {
        this->value = value;
        this->next = next;
        this->back = back;
    }


};
```

ListaDoble

```
class ListaDoble
{
public:
    Nodo* head;
    Nodo* end;
    int size;

    ListaDoble() {
        head = nullptr;
        end = nullptr;
        size = 0;
    }

    bool isEmpty();
    void addFirst(Transaccion* value);
    void addLast(Transaccion* value);
    Nodo* search(string value);
    void remove(string value);
    void show();
    void ordenar();
    void graficar();
    void usuario(Usuario* usuario);
};
```

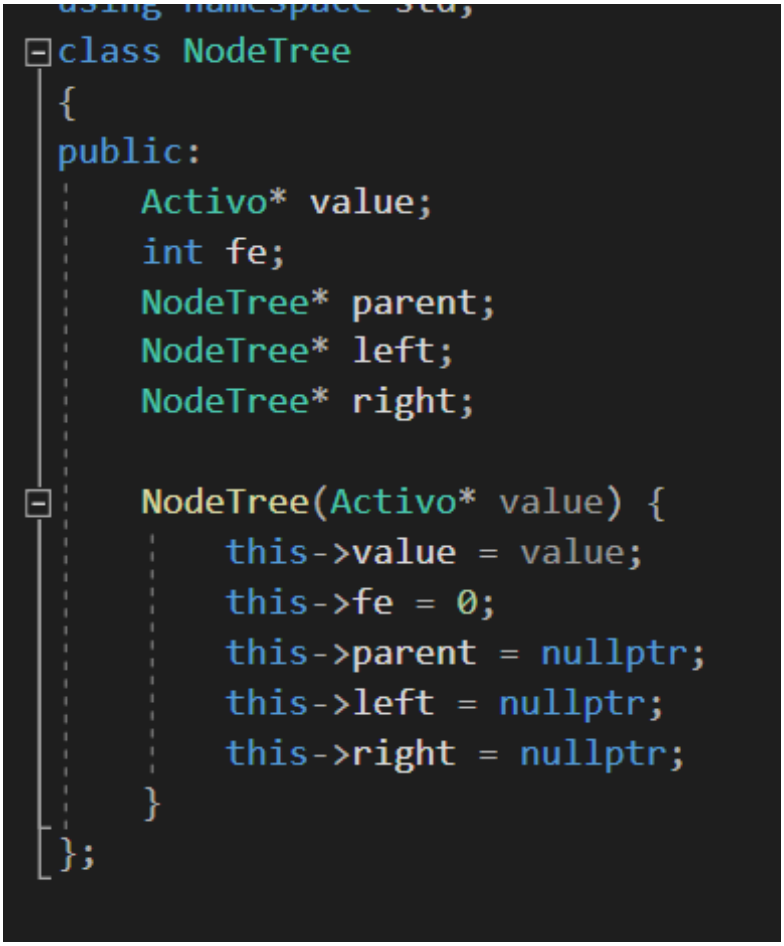
NodoMatriz

```
class NodoMatriz
{
public:
    Usuario* value;
    NodoMatriz* up;
    NodoMatriz* down;
    NodoMatriz* left;
    NodoMatriz* right;
    NodoMatriz* forward;
    NodoMatriz* back;
    string departamento;
    string empresa;

    NodoMatriz(Usuario* value, string departamento, string empresa) {
        this->value = value;
        up = nullptr;
        down = nullptr;
        left = nullptr;
        right = nullptr;
        forward = nullptr;
        back = nullptr;
        this->departamento = departamento;
        this->empresa = empresa;
    }
};
```

Matriz

```
class Matriz
{
public:
    NodoMatriz* head;
    NodoMatriz* departamento;
    NodoMatriz* empresa;
    ofstream archivo;
    Matriz() {
        head = new NodoMatriz(new Usuario("Admin","123","Gerber Colindres","","",""),"","");
        departamento = head;
        empresa = head;
    }

    NodoMatriz* createDepartamento(string departamento);
    NodoMatriz* createEmpresa(string empresa);
    void addUsuario(Usuario* data, string departamento, string empresa);
    NodoMatriz* searchDepartamento(string data);
    NodoMatriz* searchEmpresa(string data);
    NodoMatriz* searUsuario(string usuario, string empresa, string departamento);
    bool login(string usuario, string password, string empresa, string departamento);
    void graficar();
    void activoPorEmpresa(string empresa);
    void activoPorDepartamento(string departamento);
private:
    NodoMatriz* searchUsuario(NodoMatriz* fila, NodoMatriz* columna, string empresa, string departamento);
    bool insertUsuario(NodoMatriz* node, NodoMatriz* nuevo);
    NodoMatriz* searchUsuario(NodoMatriz* node, string usuario);
    string arriba(string cadena, NodoMatriz* node);
    string abajo(string cadena, NodoMatriz* node);
    string izquierda(string cadena, NodoMatriz* node);
    string derecha(string cadena, NodoMatriz* node);
    string generarID(string cadena, NodoMatriz* node);
    string adentro(string cadena, NodoMatriz* node);
    string label(string cadena, NodoMatriz* node);
    void generarArbol(NodoMatriz* usaurio, NodeTree* arbol);
    string generarID(string cadena, NodoMatriz* usaurio, NodeTree* arbol);
    string label(string caden, NodoMatriz* usaurio, NodeTree* arbol);
};
```

NodoTree



```
class NodeTree
{
public:
    Activo* value;
    int fe;
    NodeTree* parent;
    NodeTree* left;
    NodeTree* right;

    NodeTree(Activo* value) {
        this->value = value;
        this->fe = 0;
        this->parent = nullptr;
        this->left = nullptr;
        this->right = nullptr;
    }
};
```

Tree

```
class Tree
{
public:
    NodeTree* root;
    ofstream file;
    Tree() {
        this->root = nullptr;
    }

    bool isEmpty();
    NodeTree* getRoot();
    void add(Activo* data);
    void remove(string data);
    void search(string data);
    void inOrden();
    void preOrden();
    void postOrden();
    int getFE(NodeTree* node);
    NodeTree* leftRotate(NodeTree* node);
    NodeTree* rightRotate(NodeTree* node);
    NodeTree* doubleLeftRotate(NodeTree* node);
    NodeTree* doubleRightRotate(NodeTree* node);
    void addAVL(Activo* data);
    NodeTree* getNode(NodeTree* node, string data);
    void graficar(string usuario);
    void inOrden2(NodeTree* node);

private:
    void removeLeaf(NodeTree* node);
    void removeChild(NodeTree* node, int type);
    NodeTree* minSubTree(NodeTree* node);
    void remove(NodeTree* node);
    void preOrden(NodeTree* node);
    void inOrden(NodeTree* node);
    void postOrden(NodeTree* node);
    NodeTree* add(NodeTree* node, Activo* value);
    bool isRoot(NodeTree* node);
    bool isLeaf(NodeTree* node);
    bool isInternal(NodeTree* node);
    int height(NodeTree* node);
    int depth(NodeTree* node);
    NodeTree* addAVL(NodeTree* nuevo, NodeTree* subAr);
};
```