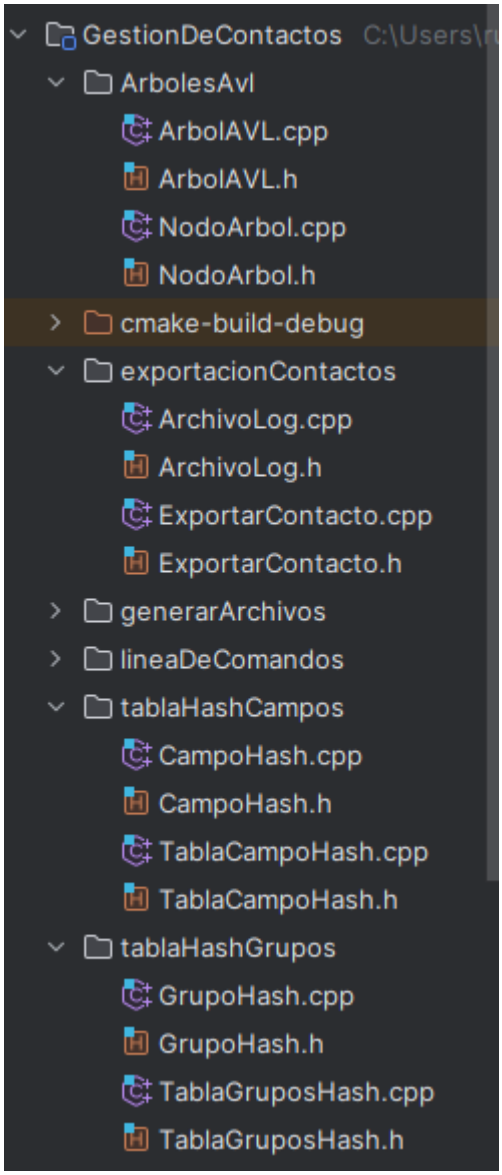
MANUAL TÉCNICO

1. Introducción y Propósito:

El siguiente proyecto contiene un software desarrollado en C++ cuya función es un sistema de gestión de contactos, básicamente el usuario ingresa contactos divididos por grupos.

Arquitectura y Diseño:



Esta es la estructura del proyecto cada función está divida por carpetas.

ArbolesAvl: Aquí se encuentran las clases para el manejo de árboles avl del sistema de campos de cada contacto.

exportacionContactos: Aquí se encuentran las clases que nos permiten exportar todos los contactos de un grupo en archivos de texto.

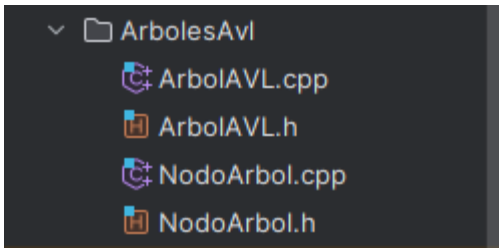
generarArchivos: En esta carpeta se encuentran las clases que nos permitirán exportar en imágenes la estructura completa de nuestro sistema, así como sus árboles.

lineaDeComandos: En esta carpeta se encuentran las clases que nos permiten extraer los datos de los comandos ingresados por el usuario para crear, ingresar y buscar contactos y grupos.

tablaHashCampos: Clases necesarias para construir la tabla hash de los campos de los contactos ingresados.

tablaHashGrupos: Clases necesarias para construir la tabla hash de los grupos en los que se dividirán los contactos.

Detalles de las clases:



Como ya se había mencionado anteriormente aquí se encuentran las clases que nos permitirán manejar los árboles correctamente en el sistema.

ArbolAVL.cpp/.h:

Estructura:

class ArbolAVL{

private:

int numeroDato; → Este atributo nos permitirá conocer el número total de nodos que tiene el árbol

NodoArbol* arbol; → Atributo de tipo NodoArbol

int altura(NodoArbol* nodo); → Funcion que obtiene la altura del arbol

int maxNum(int a, int b); → Función que obtiene el número máximo entre dos números

NodoArbol* nuevoNodo(string &valor); → Función que nos permite crear un nuevo nodo

NodoArbol* rotarDer(NodoArbol* y); → Función que nos permite rotar el árbol a la derecha

NodoArbol* rotarIzq(NodoArbol* x); → Función que nos permite rotar el árbol a la izquierda

int balance(NodoArbol *nodo);→ Función que nos permite verificar si el árbol está balanceado

NodoArbol* insertar(NodoArbol *nodo, string &valor); →Función que nos permite insertar un nodo

int valorNodoArbol(NodoArbol * raiz, string dato); → Función que nos permite obtener el valor de un nodo

string datoNodoArbol(NodoArbol * raiz, int valor); → Función que nos permite obtener el dato de un nodo

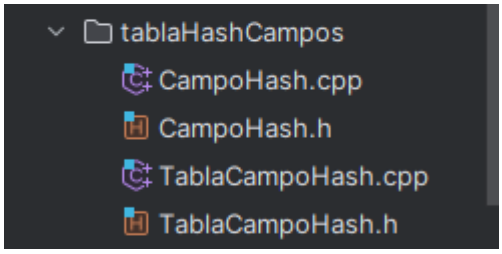
public:

ArbolAVL(); → Método constructor de la clase

void preOrden(NodoArbol* raiz); → Método para imprimir el árbol en preOrden

void inOrden(NodoArbol * raiz); → Método para imprimir el árbol en inOrden

void postOrden(NodoArbol* raiz);};→ Método para imprimir el árbol en postOrden



Estas clases nos permiten manejar lo que serían los campos de los contactos, cada campo tendrá consigo un árbol propio donde se ingresará cada campo de cada contacto.

TablaCampoHash.cpp/.h:

Estructura:

```
class TablaCampoHash{
private:
   double factorCarga = 0.6; → Este atributo nos permite controlar el factor de carga del campo.
   CampoHash* tabla; → Arreglo de campos.
   int tamanoActual; → Tamaño actual del arreglo.
   int cantidadDatos; → Cantidad de campos ingresados en el arreglo.
   int funcionHash(string& nombreCampo); → Función hash.
   void rehash(); → Función que nos permite hacer el rehash del arreglo
public:
   TablaCampoHash(); → Método constructor
   ~TablaCampoHash() { → Método destructor.
      delete[] tabla;
   }
      void insertarGrupo(string nombreCampo, string tipoDato); → Función para insertar un nuevo campo.
   CampoHash buscarGrupo(string nombreCampo); → Función para buscar un campo.
   int obtenerPosicionGrupo(string nombreCampo); → Función para obtener la posición de un campo
   CampoHash* obtenerTabla(); →Función para obtener todo el arreglo de campos.
   void imprimirDatosTabla(); → Función para imprimir todos los campos de la tabla.
      void insertarDatosArbol(string nombreCampo, string valor); → Inserta los datos del contacto al arbol.
```

void insertarDatosArbolPosicion(int posicion, string valor); → Inserta los datos del contacto al arbol.

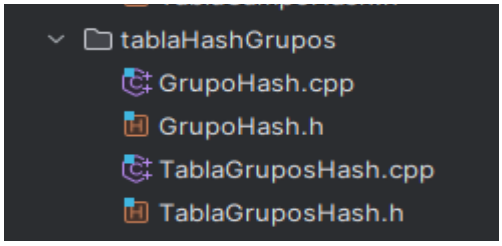
string obtenerNombreCampo(int valor); → Obtiene el nombre de un campo ingresando la posición .

int getTamanoTabla(); → Obtenemos el tamaño del arreglo.

int obtenerCantidadDatos(); → Obtenemos la cantidad de campos de la tabla.

int obtenerCantidadDatosArbol(); → Obtenemos la cantidad de contactos ingresados.

};



Estas clases son las que se encargan de manejar los grupos de contactos, cada grupo tendrá otra tabla hash asociada a los campos de cada contacto.

TablaGrupoHash.cpp/.h:

Estructura:

class TablaGruposHash{

private:

double factorCarga = 0.6; → Este atributo nos permite controlar el factor de carga del grupo.

GrupoHash* tabla; → Arreglo de grupos.

int tamanoActual; → Tamaño actual del arreglo.

int cantidadDatos; → Cantidad de grupos ingresados en el arreglo.

int funcionHash(string& nombreGrupo); → Función hash.

void rehash(); → Función que nos permite hacer el rehash del arreglo

public:

TablaGruposHash(); → Método constructor

~TablaGruposHash() { → Método destructor

delete[] tabla;

}

void insertarGrupo(GrupoHash grupo); → Función para insertar un nuevo grupo.

void insertarGrupoPorNombre(string nombreGrupo); → Función para insertar un nuevo grupo.

void insertarCamposGrupo(string nombreGrupo, string campo, string tipoDato); → Función para insertar los campos de un grupo de contactos.

void insertarDatosCampos(string nombreGrupo, string campo, string valor); → Función para insertar un contacto..

GrupoHash buscarGrupo(string nombreGrupo); → Función para buscar un grupo.

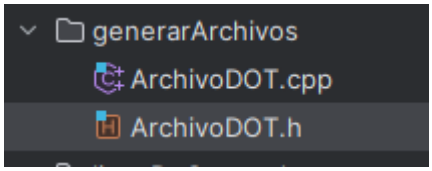
int obtenerPosicionGrupo(string nombreGrupo); → Función para obtener la posición de un grupo.

GrupoHash* obtenerTabla(); → Función para obtener el arreglo de grupos.

int obtenerTamanoActual(); → Función para obtener el tamaño del arreglo.

void imprimirDatosTabla(); → Función para imprimir los grupos

};



Estas clases nos permiten generar las graficas de todo el sistema, de las tablas hash y de los árboles.

ArchivoDOT.cpp/.h:

Estructura:

class ArchivoDOT{

public:

void generarArchivoDOT(NodoArbol* raiz, ofstream& archivoDot); →Funcion que permite generar el dot de los arboles.

void obtenerArchivoDOT(NodoArbol *raiz); →Función que nos permite crear el gráfico de los árboles.

void graficarArbolesGrupo(string nombreG,TablaGruposHash &tabla); →Funcion que nos permite graficar todos los árboles de los campos de un grupo.

void archivoDOTTablaCampos(ofstream& archivoDot, TablaGruposHash &tabla); → Función que nos permite graficar la tabla hash de los campos de un grupo.

void archivoDOTTablaHashGlobal(ofstream& archivoDot, TablaGruposHash &tabla); → Función que nos permite generar el archivo dot de la estructura general.

void graficarTablaHashGlobal(TablaGruposHash &tabla); → Función que nos permite graficar toda la estructura global del sistema.

void archivoDOTCamposGrupo(ofstream& archivoDot, string nombre, TablaGruposHash &tabla); →Funcion que nos permite generar el archivo dot de los campos de un grupo.

void graficarGrupoHash(string nombre, TablaGruposHash &tabla); →Funcion que nos permite graficar un grupo en especifico.

};