

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y sistemas
Estructuras de Datos

Ingenieros:

- Ing. Luis Espino
- Ing. Edgar Ornelis
- Ing. Álvaro Hernández

Auxiliares:

- Kevin Martinez
- Carlos Castro
- José Montenegro



MANUAL TECNICO

1. Introducción

Este manual describe la arquitectura, la lógica de implementación y los componentes del software de una aplicación desarrollada en C++ utilizando la biblioteca de Qt. El proyecto consiste en la simulación de una red social con gestión de usuarios, publicaciones y comentarios, y utiliza estructuras de datos complejas como árboles AVL, listas doblemente enlazadas, y un árbol binario de búsqueda (ABB).

2. Requisitos Técnicos

- **Lenguaje:** C++ (C++17 o superior recomendado)
- **Framework:** Qt 6.x (para la interfaz gráfica)
- **Sistema Operativo:** Windows o cualquier otro sistema que soporte Qt y MinGW o equivalente.
- **Compilador:** MinGW 64-bit
- **Entorno de Desarrollo:** Qt Creator

3. Arquitectura del Sistema

La aplicación tiene una arquitectura basada en ventanas que gestionan las interfaces de usuarios y administradores:

1. Ventana Principal (MainWindow):

- Proporciona opciones de inicio de sesión y registro.
- Identifica al usuario y carga la interfaz correspondiente (usuario o administrador).

2. Ventana de Usuario (formUser):

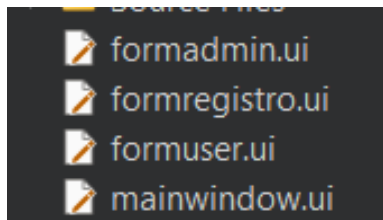
- Utilizada para gestionar las acciones del usuario, como crear publicaciones, comentarios y acceder a la información personal.

3. Ventana de Administrador (formAdmin):

- Gestiona el acceso a la administración global de la aplicación, como usuarios y estadísticas.

4. Ventana de Registro (formregistro):

- Permite a los nuevos usuarios registrarse y ser añadidos al árbol AVL de usuarios.



4. Estructuras de Datos Utilizadas

4.1 Árbol AVL (arbolAVL)

- **Propósito:** Almacenar y gestionar información sobre los usuarios.
- **Implementación:** El árbol AVL mantiene los usuarios ordenados por correo electrónico, lo cual facilita búsquedas, inserciones y eliminaciones balanceadas.
- **Métodos Importantes:**
 - `insertU(user*)`: Inserta un nuevo usuario en el árbol.
 - `eliminar(const std::string&)`: Elimina un usuario en base al correo electrónico.

- searchU(const std::string&): Busca un usuario y retorna un puntero al objeto user.

```

1  #ifndef ARBOLAVL_H
2  #define ARBOLAVL_H
3  #include "nodoavl.h"
4
5  class arbolAVL
6  {
7  private:
8
9      NodoAVL* raiz;
10     int height(NodoAVL* node);
11     NodoAVL* insert(NodoAVL* node, user* u);
12     int getBalance(NodoAVL* node);
13     NodoAVL* rotateRight(NodoAVL* y);
14     NodoAVL* rotateLeft(NodoAVL* x);
15     NodoAVL* minValueNode(NodoAVL* node);
16     NodoAVL* deleteNode(NodoAVL* root, const std::string& correoE);
17     std::string preOrder(NodoAVL* node);
18     std::string inOrder(NodoAVL* node);
19     std::string postOrder(NodoAVL* node);
20     void generarDot(NodoAVL* node, std::ofstream& out);
21
22 public:
23     arbolAVL() : raiz(nullptr) {}
24     void insertU(user* usuario);
25     bool search(const std::string& correoE);
26     NodoAVL* searchNode(NodoAVL* node, const std::string& correoE);
27     bool credenciales(const std::string& correoE, const std::string& contrasena);
28     std::string mostrarInfo(const std::string& correoE);
29     user* searchU(const std::string& correoE);
30     void eliminar(const std::string& correoE);
31     std::string mostrarPreOrder();
32     std::string mostrarInOrder();
33     std::string mostrarPostOrder();
34     void graficar(const std::string& nombreArchivo);
35 };
36
37
38 #endif // ARBOLAVL_H
39

```

4.2 Lista Doble Enlazada (ListaDoble)

- **Propósito:** Gestionar las publicaciones de los usuarios y sus comentarios.
- **Implementación:** Cada nodo contiene una publicación y la lista permite navegar entre publicaciones hacia adelante y hacia atrás.
- **Métodos Importantes:**
 - insertarFinal(...): Inserta una publicación al final de la lista.
 - eliminarPublicacionMasReciente(const std::string&): Elimina la publicación más reciente de un usuario específico.
 - buscarPorFecha(const std::string&): Busca publicaciones por fecha.

```

1  #ifndef LISTADOBLE_H
2  #define LISTADOBLE_H
3
4  #include "nodoListadoble.h"
5
6  class ListaDoble
7  {
8  private:
9      nodoListadoble* cabeza;
10     nodoListadoble* cola;
11
12 public:
13     ListaDoble();
14     ~ListaDoble();
15     void insertarFinal(std::string correo, std::string contenido, std::string fecha, std::string hora, std::string Imagen);
16     std::string mostrarAdelante();
17     std::string mostrarAtras();
18     std::string buscarPorFecha(std::string fecha);
19     void eliminarPublicacion(std::string correo, std::string fecha);
20     void agregarComentario(std::string correo, std::string fecha, std::string hora, Comentario comentario);
21     std::string mostrarComentarios(std::string fecha, std::string hora);
22     std::string mostrarPublicaciones();
23     std::string buscarCorreoPorHora(std::string fecha, std::string hora);
24     void generarGraphviz(std::string nombreArchivo);
25     int contarPublicaciones();
26     std::vector<std::pair<nodoListadoble*, int>> obtenerPublicacionesConComentarios();
27     std::string top3PublicacionesConMasComentarios();
28
29     bool eliminarPublicacionMasReciente(std::string correo);
30
31     // Eliminar todas las publicaciones de un usuario
32     void eliminarPublicacionesPorCorreo(std::string correo);
33
34
35     nodoListadoble* obtenerCabeza() {
36         return cabeza; // Retornar el puntero al primer nodo
37     }
38 };
39
40 #endif // LISTADOBLE_H
41

```

4.3 Árbol Binario de Búsqueda (ABB)

- **Propósito:** Almacenar publicaciones de acuerdo con la fecha de creación, permitiendo consultas eficientes.
- **Implementación:** Las fechas son utilizadas como clave para organizar las publicaciones en un ABB.
- **Métodos Importantes:**
 - insertarPublicacion(...): Inserta una nueva publicación en el árbol.
 - mostrarInordenLimitado(int), mostrarPreordenLimitado(int), mostrarPostordenLimitado(int): Permiten mostrar un número limitado de publicaciones según el tipo de recorrido.
 - top3FechasConMasPublicaciones(): Retorna las tres fechas con más publicaciones.

```

1  #ifndef ABB_H
2  #define ABB_H
3
4  #include "NodoABB.h"
5  #include "vector"
6
7
8  class ABB
9  {
10
11  private:
12
13      NodoABB* raiz;
14      NodoABB* insertarRecursivo(NodoABB* nodo, std::string fecha);
15      NodoABB* buscarNodo(NodoABB* nodo, std::string fecha);
16      void mostrarPublicacionesPorFecha(NodoABB* nodo, std::string fecha);
17      std::string inordenLimitado(NodoABB* nodo, int& contador, int limite);
18      std::string preordenLimitado(NodoABB* nodo, int& contador, int limite);
19      std::string postordenLimitado(NodoABB* nodo, int& contador, int limite);
20      std::vector<std::pair<std::string, int>> contarPublicacionesPorFecha(NodoABB* nodo);
21      static bool compararPorCantidad(const std::pair<std::string, int>& a, const std::pair<std::string, int>& b);
22      NodoABB* buscarNodoPorFecha(NodoABB* nodo, const std::string& fecha);
23      void generarGraphvizRec(NodoABB* nodo, std::ofstream& archivo, std::string fechaEspecific);
24      void eliminarPublicacionesPorCorreoRec(NodoABB* nodo, std::string correo);
25
26
27  public:
28
29      ABB() {
30          raiz = nullptr;
31      }
32
33      bool eliminarPublicacion(std::string correo);
34      void eliminarPublicacionesPorCorreo(std::string correo);
35
36      void insertarPublicacion(std::string correo, std::string contenido, std::string fecha, std::string hora, std::string pathImagen);
37      void mostrarPublicacionesPorFecha(std::string fecha);
38      std::string mostrarInordenLimitado(int limite);
39      std::string mostrarPreordenLimitado(int limite);
40      std::string mostrarPostordenLimitado(int limite);
41      std::string top3FechasConMasPublicaciones();
42      void graficarABBConListaFechaEspecific(std::string nombreArchivo, std::string fechaEspecific);
43
44  };
45
46  #endif // ABB_H
47

```

4.4 Árbol B de Orden 5 (Comentarios)

- **Propósito:** Almacenar los comentarios realizados en publicaciones, ordenándolos por correo, fecha y hora.
- **Implementación:** El árbol B de orden 5 permite almacenar y buscar los comentarios de manera eficiente.

```

1  #ifndef ARBOLB_H
2  #define ARBOLB_H
3  #include "nodoComentario.h"
4  #include "Comentario.h"
5
6
7
8  class ArbolB
9  {
10
11  private:
12      nodoComentario* raiz;
13      int orden;
14
15      void insertarNolLeno(nodoComentario* nodo, Comentario comentario);
16      void dividirHijo(nodoComentario* nodoPadre, int indice);
17      int compararFechaHora(const Comentario& c1, const Comentario& c2);
18      std::string imprimirInorden(nodoComentario* nodo);
19      int contarComentariosRecursivo(nodoComentario* nodo);
20
21  public:
22      ArbolB(int _orden) {
23          raiz = nullptr;
24          orden = _orden;
25      }
26
27      void insertar(Comentario comentario);
28      std::string imprimir();
29      int contarComentarios();
30
31  };
32
33  #endif // ARBOLB_H
34

```

5. Descripción del Código

5.1 Archivos de Cabecera y Dependencias

- mainwindow.h y ui_mainwindow.h: Define la ventana principal, responsable de gestionar la interfaz gráfica del inicio de sesión.
- formuser.h, formadmin.h, formregistro.h: Define las interfaces y funcionalidades para cada tipo de usuario.
- QMessageBox y QFileDialog: Utilizados para mostrar mensajes al usuario y abrir archivos (imágenes para publicaciones).

5.2 Usuarios :

```
1  #ifndef USER_H
2  #define USER_H
3
4  #include <string>
5
6  class user{
7  private:
8      std::string nombre;
9      std::string apellido;
10     std::string fechaN;
11     std::string correoE;
12     std::string pass;
13
14 public:
15
16     // Constructor del usuario
17     user(const std::string& nombre, const std::string& apellido, const std::string& fechaN, const std::string& correoE, const std::string& pass);
18
19     std::string getnombre() const;
20     std::string getapellido() const;
21     std::string getfechaN() const;
22     std::string getcorreoE() const;
23     std::string getpass() const;
24
25     void setnombre(const std::string& nombre);
26     void setapellido(const std::string& apellido);
27     void setfechaN(const std::string& fechaN);
28     void setcorreoE(const std::string& correoE);
29     void setpass(const std::string& pass);
30     std::string mostrarInfo();
31     std::string mostrarInfoG();
32 };
33
34 #endif // USER_H
35
```