Trabajo especial: Análisis y Diseño de Algoritmos I

Rodríguez, Sofía
rodriguez.sofi23@gmail.com
Artaza, Sheila
sheilaartaza6@gmail.com
Grupo: 3

Índice:

Introducción:

En este informe desarrollaremos los siguientes Tipos de Datos Abstractos (TDA) necesarios para llevar a cabo el Trabajo Especial. Los cuales están especificados en el lenguaje Nereus e implementados en el lenguaje C++.

A su vez, detallaremos las estructuras utilizadas, como así también su complejidad temporal utilizando la notación BIG - OH

Especificación Nereus

Los tipos de datos abstractos identificados son los siguientes:

- Lista
- Biblioteca Unicen
- Servicios Biblioteca

Utilizamos las estructuras de lista y registros.

Las abreviaturas que se observan al lado de las operaciones de los TDA's son las clasificaciones de las mismas:

- ❖ CB : constructoras básicas que son aquellas que crean y agregan elementos a la estructura.
- ❖ O: Las observadoras son las que nos brindan la información que necesitemos.

TDA LISTA

Class Lista [any:element]
Imports natural
Basic Constructors Lista, agregarLista
Effective
Type Lista
Operations

(CB) **Lista :** → Lista () ;
(CB) **AgregarLista:** Lista(l) * e(element) → L;
(O) **Vacía: Lista(l)** → boolean;
(O) **DevolverElemento:** Lista(l) → e;
pre: notVacia(l)**;**
(O) **Tamaño:** Lista(l) → n;
(O) **Pertenece:** Lista(l) * e(element) → boolean;
pre: notVacia(l);
(O) **Insertar:** Lista(l) * e(element) * n(pos) → L;
(O) **DevolverPos:** Lista(l) * n(pos) → n;

END_CLASS

Header de la lista:

```
4   template <class L>
5
6   class Lista
7   {
8       public:
9           Lista();
10          virtual ~Lista();
11          void agregarlista(L datoagregar);
12          bool vacia() const;
13          L devolverelem() const;
14          void avanzarsig();
15          void volveralprincipio();
16          int lenght() const;
17          bool pertenece(L datobuscado) const;
18          void insertar(L nuevo, int pos);
19          L devolverpos (int pos);
20
21      protected:
22
23      private:
24          struct nodo {
25             L elem;
26             nodo * sig;
27          };
28          nodo * primero;
29          nodo * ult;
30          nodo * cursorpublico;
31          int longitud;
32  };
33
```

TDA BIBLIOTECAUNICEN

Class BibliotecaUnicen
Imports natural, string, lista[string];
Basic Constructors BibliotecaUnicen, AggBibliotecaUnicen
Effective
Type BibliotecaUnicen
Operations

(CB) **BibliotecaUnicen:** → BibliotecaUnicen;
(CB) **AggBibliotecaUnicen:** n(ID) * s(titulo) * s(autor) * s(editorial) * n(año) * n(edicion) * lista(listaGeneros) * n(nroPag) * n(cantEjV) * n(precio) → BibliotecaUnicen;
(O) **ConsultarID:** BibliotecaUnicen(B) → n(ID);
(O) **ConsultarTitulo:** BibliotecaUnicen(B) → s(título);
(O) **ConsultarAutor:** BibliotecaUnicen(B) → s(autor);
(O) **ConsultarEditorial:** BibliotecaUnicen(B) → s(editorial);
(O) **ConsultarAño:** BibliotecaUnicen(B) → n(año);
(O) **ConsultarEdicion:** BibliotecaUnicen(B) → n(edicion);
(O) **ConsultarNroPag:** BibliotecaUnicen(B) → n(nroPag);
(O) **ConsultarCantEjV:** BibliotecaUnicen(B) → n(cantEjV);
(O) **ConsultarPrecio:** BibliotecaUnicen(B) → n(precio);
(O) **ConsultarGenero:** BibliotecaUnicen(B) * s(género) → L;

END_CLASS;

Header BibliotecaUnicen:

```
class BibliotecaUnicen
{
    public:
        BibliotecaUnicen();
        virtual ~BibliotecaUnicen();
        void AggBibliotecaUnicen(string IDLibro, string tituloLibro, string Autor, string Editorial, int anio, int Edicion, Lista <string> listaGenero
        string consultarIDLibro() const;
        string consultartitulodelibro() const;
        string consultarautor() const;
        string consultareditorial() const;
        int consultaranio() const;
        int consultaredicion() const;
        int consultarnropaginas() const;
        int consultarcantejemplaresvendidos() const;
        int consultarprecio() const;
        Lista<string> consultarGenero() const;
        bool operator==(BibliotecaUnicen libro);

        void mostrarDatos();

    protected:

    private:
        string IDLibro;
        string tituloLibro;
        string Autor;
        string Editorial;
        int anio;
        int Edicion;
        Lista<string> listaGeneros;
        int nroPag;
        int CantEjV;
        float Precio;
};
```

TDA SERVICIOSBIBLIOTECA

Class ServiciosBiblioteca
Imports natural, arreglo, Lista;
Basic Constructors
Effective
Type ServiciosBiblioteca
Operations

- (O) **BuscarTitulo:** BibliotecaUnicen(B) * s(titulo) → Boolean;
- (O) **esmenor:** nat(año1) * nat(año2) → nat(año2);
- (O) **Devolver_Lista_Rango:** Libro * nat(año1) * nat(año2) * Lista(libros) → ListaRango;
- (O) **Devolver_Lista_Genero:** Libro * S(generoBuscado) → L(ListaGenero);
- (O) **Lista_vendidos:** L(listaGenero) → L(masVentas);
- (O) **menu:** Libro → menu;

END_CLASS;

Header ServiciosBiblioteca:

```
#ifndef SERVICIOSBIBLIOTECA_H
#define SERVICIOSBIBLIOTECA_H
#include "BibliotecaUnicen.h"
#include "Lista.h"

using namespace std;

//bool usaComodin(string tituloIngresado);
bool buscarTitulo(Lista<BibliotecaUnicen>& Libreria, string tituloIngresado);
void esmenor(int Anio_1,int & Anio_2);
void Devolver_Lista_Rango(Lista<BibliotecaUnicen> Libreria,int Anio_Min,int Anio_Max, Lista<BibliotecaUnicen> & ListadoLibros);
Lista<BibliotecaUnicen> Devolver_Lista_Genero(Lista<BibliotecaUnicen> Libreria, string generoBuscado);
Lista<BibliotecaUnicen> Lista_vendidos(Lista<BibliotecaUnicen> & lstGenero);
void menu (Lista<BibliotecaUnicen> & libreria);

#endif // SERVICIOSBIBLIOTECA_H

```

DESCRIPCIÓN Y COMPLEJIDAD DE LAS OPERACIONES

TDA LISTA
n = cantidad de elementos.

Nombre	Descripción	Complejidad
Lista	Crea la lista	O(1)
agregarLista	añade elementos a la lista	O(n)
Vacía	pregunta si la lista está vacía	O(1)
DevolverElemento	Devuelve un elemento de la lista	O(1)
Tamaño	indica el tamaño que tiene la lista	O(n)
Pertenece	indica si pertenece o no un elemento en la lista	O(1)
InsertarOrdenado	inserta los elementos de manera ordenada	O(n)
DevolverPos	devuelve la posición donde se encuentra el elemento.	O(n)

TDA BIBLIOTECAUNICEN
n = cantidad de géneros.

Nombre	Descripción	Complejidad
BibliotecaUnicen	Inicializa el libro.	O(1)
AggBibliotecaUnicen	Agrega un libro con sus atributos.	O(1)
ConsultarID	Dado un libro, nos devuelve el ID.	O(1)
ConsultarTitulo	Dado un libro, nos devuelve el título.	O(1)
ConsultarAutor	Dado un libro, nos devuelve el autor.	O(1)
ConsultarEditorial	Dado un libro, nos devuelve la editorial.	O(1)
ConsultarAño	Dado un libro, nos devuelve	O(1)

	el año.	
ConsultarEdicion	Dado un libro, nos devuelve la edición.	O(1)
ConsultarGenero	Dado un libro y un género, nos devuelve si se pertenece o no a ese género.	O(n)
ConsultarNroPag	Dado un libro, nos devuelve el número de página.	O(1)
ConsultarCantEjV	Dado un libro, nos devuelve la cantidad de ejemplares vendidos.	O(1)
ConsultarPrecio	Dado un libro, nos devuelve el precio que tiene.	O(1)

TDA SERVICIOSBIBLIOTECA
n = cantidad de libros.

Nombre	**Descripción**	**Complejidad**
BuscarTitulo	Busca el título ingresado dentro de la librería.	O(n)
EsMenor	Dado un año 1, obliga a el año2 a ser mayor.	O(1)
DevolverListaRango	Nos devuelve género determinado dentro de un rango establecido.	O(n)
DevolverListaGenero	Dada la librería y un género, pasa a una nueva lista todos los libros de dicho género.	O(n)
ListaVendidos	Dada la lista genero, ordena los libros por cantidad de ejemplares vendidos.	O(n)
Menú	Muestra todos los servicios que ofrece la Biblioteca.	O(1)

Descripción de las funciones de los servicios:

Para el servicio 1, pedimos que el usuario ingrese el título que desea encontrar, luego con ese string ingresado por el usuario comparamos con el campo de la librería que nos devuelve el título del mismo, de existir coincidencia nos devuelve que el libro si existe en la librería y en qué posición se encuentra.
La complejidad de este primer servicio es de O(n), ya que para poder dar una posible respuesta debe recorrer toda la lista libros.

En el servicio 2, para realizar dicho servicio implementamos la función DevolverListaRango, la cuál recibe por parámetros, la librería, los dos años(límites), y por referencia la lista con el nombre ListadosLibros.
Para los años implementamos un void, esMenor, el cual lo que realiza es chequear que el año 2 ingresado siempre sea mayor al año 1.
En el momento que se encuentre un libro dentro de este rango de años, se invoca el agregar en ListadoLibros.
La complejidad temporal de este servicio, es O(n), ya que debemos recorrer secuencialmente toda la librería, para saber cuales libros están dentro del rango ingresado por el usuario.

En el servicio 3, para la implementación del mismo utilizamos una primera función Devolver_Lista_genero, la cual dada la librería y un género de importancia para el usuario devuelve una listaGenero, donde se van a listar todos los libros de un mismo género.
A continuación, utilizamos la función lista_Vendidos, la cual recibe por parámetro la listaGenero, su funcion es recorrer la listaGenero y va consultando por la cantidad de ejemplares vendidos, y con este dato, los va insertando en una nueva lista Vendidos en orden de ejemplares vendidos.
La complejidad temporal de ese servicio es O(n) ya que siempre debe recorrer el total de la lista.

En cuanto a las estructuras utilizadas para la implementación de los servicios, creemos que son las más adecuadas ya que al tener implementada una lista podemos “reutilizar” código, y no hubo necesidad de implementar otras estructuras más sencillas como por ejemplo podría haber sido un arreglo.

Resto de código:

Lista.cpp

```
#include <iostream>
#include "Lista.h"
#include "BibliotecaUnicen.h"

using namespace std;

template <typename L>
Lista<L> :: Lista()
{
    this->primero = NULL;
    this->ult = NULL;
    this->cursorpublico = NULL;
    this->longitud = 0;
}

template <typename L>
Lista<L> :: ~Lista()
{
    nodo * aux;
    while (primero != NULL) {
        aux = primero->sig;
        primero = aux;
        delete aux;
    }
    primero = NULL;
}

template <typename L>
void Lista<L> :: agregarlista(L datoagregar)
```

```
{
    nodo * nuevo = new nodo;
    nuevo->elem = datoagregar;
    nuevo->sig = NULL;
    if (primero == NULL){
        primero = nuevo;
        ult = primero;
    }
    else{
        nodo * aux;
        aux = primero;
        while (aux->sig != NULL){
            aux = aux->sig;
        }
        aux->sig = nuevo;
    }
    longitud++;
}

template <typename L>
bool Lista<L> :: vacia() const
{
    if(longitud==0)
        return true;
    else
        return false;
}

template <typename L>
int Lista<L> :: lenght() const {
    return longitud;
}
```

```
template <typename L>
void Lista<L> :: avanzarsig()
{
    if (cursorpublico->sig != NULL){
        cursorpublico = cursorpublico->sig;
    }
}

template <typename L>
void Lista<L> :: volveralprincipio()
{
    cursorpublico = primero;
}

template <typename L>
L Lista<L> :: devolverelem() const
{
    if(cursorpublico != NULL){
        return cursorpublico->elem;
    }
}

template <typename L>
bool Lista<L> :: pertenece(L datobuscado) const
{
    nodo * aux;
    aux = primero;
    while (aux != NULL){
        if (aux->elem == datobuscado)
            return true;
        aux = aux->sig;
    }
    return false;
```

```
}

template <typename L>
void Lista<L> :: insertar(L nuevo, int pos)
{
   if ((pos <= longitud) && (pos>0)){
      nodo * ainsertar = new nodo;
      ainsertar->elem = nuevo;
      ainsertar->sig = NULL;
      nodo * cursor = primero;
      int contador = 1;
      while (contador<pos-1){
         contador++;
         cursor = cursor->sig;
      }
      if (pos == 1){
         ainsertar->sig = primero;
         primero = ainsertar;
      }
      else{

         ainsertar->sig = cursor->sig;
         cursor->sig = ainsertar;
      }
      longitud++;

   }
}

template <typename L>
L Lista<L> :: devolverpos(int pos)
{
   nodo * cursor;
```

```
    if ((pos<=longitud) && (pos>0)){
        int contador = 1;
        cursor = primero;
        while (contador < pos) {
            contador ++;
            cursor = cursor->sig;
        }
        return (cursor->elem);
    }
}

template class Lista <BibliotecaUnicen>;
template class Lista <string>;
```

BibliotecaUnicen.cpp

```
#include <iostream>
#include "BibliotecaUnicen.h"
#include "Lista.h"

using namespace std;

BibliotecaUnicen :: BibliotecaUnicen()
{
    // constructora
}

BibliotecaUnicen :: ~BibliotecaUnicen()
{
    //dtor
}

void BibliotecaUnicen :: AggBibliotecaUnicen(string IDLibro,
string tituloLibro, string Autor, string Editorial, int anio, int
Edicion, Lista<string> listaGeneros, int nroPag, int CantEjV,
float Precio)
{
    cout << "agrego el libro " << tituloLibro << endl;
    this->IDLibro = IDLibro;
    this->tituloLibro = tituloLibro;
    this->Autor = Autor;
    this->Editorial = Editorial;
    this->Edicion = Edicion;
    listaGeneros.volveralprincipio();
    //while ( != NULL) {
    this->listaGeneros = listaGeneros;
```

```
      //  listaGeneros.avanzarsig();
    //}
    this->anio= anio;
    this->nroPag = nroPag;
    this->CantEjV = CantEjV;
    this->Precio = Precio;
}

string BibliotecaUnicen :: consultarIDLibro() const
{
    return IDLibro;
}

string BibliotecaUnicen :: consultartitulodelibro() const
{
    return tituloLibro;
}

string BibliotecaUnicen :: consultarautor() const
{
    return Autor;
}

string BibliotecaUnicen :: consultareditorial() const
{
    return Editorial;
}

int BibliotecaUnicen :: consultaranio() const
{
    return anio;
}
```

```
int BibliotecaUnicen :: consultaredicion() const
{
    return Edicion;
}

int BibliotecaUnicen :: consultarnropaginas() const
{
    return nroPag;
}

int BibliotecaUnicen :: consultarcantejemplaresvendidos() const
{
    return CantEjV;
}

int BibliotecaUnicen :: consultarprecio() const
{
    return Precio;
}

Lista<string> BibliotecaUnicen :: consultarGenero() const
{
    return listaGeneros;

}

bool BibliotecaUnicen::operator==(BibliotecaUnicen libro)
{
    return IDLibro == libro.consultarIDLibro();
}

void BibliotecaUnicen :: mostrarDatos()
{
```

```
    cout << "El Id del libro es: " << this->IDLibro << endl;
    cout << "El titulo del libro es: " << this->tituloLibro << endl;
    cout << "El autor es: " << this->Autor << endl;
    cout << "La editorial es: " << this->Editorial << endl;
    cout << "El año de edición: " << this->anio << endl;
    cout << "El volumen del libro es: " << this->Edicion << endl;
    cout << "Generos: " << endl;
    listaGeneros.volveralprincipio();
    int longitud = 1;
    while (longitud <= listaGeneros.lenght()){
        listaGeneros.devolverpos(longitud);
        longitud++;
    }
    cout << "Tiene " << this->nroPag << "páginas" << endl;
    cout << "Tiene " << this->CantEjV << "ejemplares vendidos"
<< endl;
    cout << "$" << this->Precio << endl;
    cout << "-----------------------------------------" << endl;
}
```

ServiciosBiblioteca.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include "Lista.h"
#include "BibliotecaUnicen.h"
#include "ServiciosBiblioteca.h"

using namespace std;

/*-------------------------------------Servicio
1---------------------------------*/
/*bool usaComodin(string tituloIngresado)
//retorna si usa comodin o no
{
    bool usa = true;
    unsigned int i =0;
    while((i < tituloIngresado.size()) && (tituloIngresado[i] != '*'
&& tituloIngresado[i] != '?'))
    {
        i= i +1;
    }
    if (i == tituloIngresado.size())
        usa = false;
    return usa;
}*/

bool buscarTitulo(Lista<BibliotecaUnicen>& Libreria, string
tituloIngresado)
// Si no usa comodin, entra aca y retorna si existe o no el libro
buscado
```

```
{
    bool existe = false;
    int i = 1;
    while ((i <= Libreria.lenght()) && (existe == false))
    {
        if (Libreria.devolverpos(i).consultartitulodelibro() ==
tituloIngresado){
            existe = true;
        }
        else{
            i++;
        }
    }
    if (existe == true)
    {
        cout << endl;
        cout << "-----------------------------------------------" << endl;
        cout << "El titulo " << tituloIngresado << ". Existe en la
biblioteca" << " y se encuentra en la posicion: " << i << endl;
        return true;
    }
    else
        {
        cout << endl;
        cout << "-----------------------------------------------" << endl;
        cout << "El titulo ingresado no esta disponible en la
biblioteca" << endl;
        return false;
    }
}

/*----------------------------------Servicio
2----------------------------------*/
```

```
Lista<BibliotecaUnicen>
Devolver_Lista_Rango(Lista<BibliotecaUnicen> & libreria,int
Anio1, int Anio2)
{
   int contador = 1;
   Lista<BibliotecaUnicen> ListadoLibros;
   libreria.volveralprincipio();
   BibliotecaUnicen libro;
   while (contador <= libreria.lenght()){
      libro = libreria.devolverpos(contador);
      if ((libro.consultaranio() >= Anio1) && (libro.consultaranio()
<= Anio2)){ //si el libro cuenta con un año en el rango indicado,
se agrega al listado pedido
          ListadoLibros.agregarlista(libro);
      }
      contador++;//se incrementa en 1 la posicion para seguir
con la busqueda secuencial sobre el arreglo
   }
   return ListadoLibros;
}

void esmenor(int & Anio_1, int & Anio_2) // esta funcion obliga
al usuario a poner el segundo año ingresado mas grande que
el primer año
{
   while(Anio_1 > Anio_2) {
      cout << " Debe ingresar un año mayor al año1 " << Anio_1
<< endl;
      cin>> Anio_2;
   }
}
```

```
/*---------------------------------Servicio
3-------------------------------------*/
Lista<BibliotecaUnicen>
Devolver_Lista_Genero(Lista<BibliotecaUnicen> libreria, string
generoBuscado)
{
    int contador = 1;
    bool pertenece = false;
    Lista<BibliotecaUnicen> lstGenero;
    libreria.volveralprincipio();
    BibliotecaUnicen libro;
    while (contador <= libreria.lenght()) {
        libro = libreria.devolverpos(contador);
        if (libro.consultarGenero().pertenece(generoBuscado)){
            lstGenero.agregarlista(libro);
        }
        contador++;
    }
    return lstGenero;
}

Lista<BibliotecaUnicen>
agregarOrdenado(Lista<BibliotecaUnicen> & lstGenero)
{
    int longitud=1;
    int contador;
    Lista<BibliotecaUnicen> lstVendidos;
    lstGenero.volveralprincipio();
    while(longitud <= lstGenero.lenght()){
        lstVendidos.volveralprincipio();
        if (lstVendidos.vacia()){
            lstVendidos.agregarlista(lstGenero.devolverpos(1));
        }
```

```
        else{
            contador = 1;
            while((contador <= lstVendidos.lenght()) &&
(lstVendidos.devolverpos(contador).consultarcantejemplaresve
ndidos() >
lstGenero.devolverelem().consultarcantejemplaresvendidos())){
                lstVendidos.avanzarsig();
                contador++;
            }
            if (contador > lstVendidos.lenght()){
                lstVendidos.agregarlista(lstGenero.devolverelem());
            }
            else{

lstVendidos.insertar(lstGenero.devolverelem(),contador);
            }
        }
        lstGenero.avanzarsig();
        longitud++;
    }
    return lstVendidos;
}

/*-----------------------------------------Menu-------------------------
--------------------*/
void menu(Lista<BibliotecaUnicen> & Libreria)
{
    int opcion = 1;
    while (opcion != 0) {
        cout << endl;
        cout << "Menú" << endl;
        cout << "Servicio 1: Buscar libro por titulo" << endl;
```

```
        cout << "Servicio 2: Listado de libros entre un rango de
años" << endl;
        cout << "Servicio 3: Primeras 10 ventas de un genero
determinado" << endl;
        cout << endl;
        cout << "-----------------------------------------------" << endl;
        cout << "Por favor ingrese el numero de servicio que
desea ejecutar, o en caso de querer salir ingrese 0: ";
        cin >> opcion;
        cout << "-----------------------------------------------" << endl;
        cout << endl;
        if (opcion < 0 || opcion > 3) {
            cout << "-----------------------------------------------" << endl;
            cout << "Ingreso un numero valido, por favor" << endl;
            cout << "-----------------------------------------------" << endl;
        }
        switch (opcion)
        {
           case 1:
            {
                string tituloIngresado;
                bool resultado;
                cout << "-----------------------------------------------" <<
endl;
                cout<< "Ingrese el titulo del libro que desea buscar: "
<< endl;
                cin.ignore();
                getline(cin, tituloIngresado);
                cout << "-----------------------------------------------" <<
endl;
                resultado = buscarTitulo(Libreria,tituloIngresado);
                break;
            }
```

```
        case 2:
        {
            cout << "-----------------------------------------" <<
endl;
            cout << "Año 1: " << endl;
            int Anio_1,Anio_2; //se pasa como mínimo el 1
debido a que no tiene sentido que un libro sea del año 0
            cin>> Anio_1;
            cout << "Año 2: " << endl;
            cin>> Anio_2;
            //esmenor(Anio_1,Anio_2);
            cout << "-----------------------------------------" <<
endl;
            cout << endl;
            Lista<BibliotecaUnicen> ListadoLibros =
Devolver_Lista_Rango(Libreria, Anio_1, Anio_2);
            cout << "-----------------------------------------" <<
endl;
            if (ListadoLibros.vacia())
                cout << "No hay ningun libro perteneciente al
rango " << Anio_1 << "-" << Anio_2 << endl;
            else {
                cout << "El listado de libros perteneciente al rango
" << Anio_1 << "-" << Anio_2 << " es el siguiente:" << endl;
                cout << " " << endl;
                ListadoLibros.volveralprincipio();
                int longitud = 1;
                while (longitud <= ListadoLibros.lenght()){

ListadoLibros.devolverpos(longitud).mostrarDatos();
                    longitud++;
                }
            }
```

```
            cout << "-----------------------------------------" <<
endl;
            break;
        }
        case 3:
        {
            string generoBuscado;
            cout << "-----------------------------------------" <<
endl;
            cout<< "Ingrese el genero del libro que desea buscar:
" << endl;
            cin.ignore();
            getline(cin, generoBuscado);
            cout << "-----------------------------------------" <<
endl;
            if (Libreria.vacia())
                cout << "No hay ningun libro con el genero
registrado " << endl;
            else {
                cout << "El listado de 10 ventas de libros con dicho
genero: " << generoBuscado << endl;
                cout << " " << endl;
                Libreria.volveralprincipio();
                Lista<BibliotecaUnicen> lstgenero =
Devolver_Lista_Genero(Libreria,generoBuscado);
                cout<<lstgenero.lenght()<<endl;
                Lista<BibliotecaUnicen> vendidos =
agregarOrdenado(lstgenero);
                int j = 1;
                while (j <= vendidos.lenght()){
                    vendidos.devolverpos(j).mostrarDatos();
                    j++;
                }
```

```
                break;
                }
            }
        }
    }
}
```

Conclusión:

En conclusión, en el informe realizamos punto por punto las distintas particularidades propias del código implementado para la resolución del ejercicio en C++, como también especificaciones algebraicas en Nereus y la complejidad temporal concluyendo que la estrategia que implementamos fue bastante correcta, resolviendo de manera eficiente y sin una alta complejidad temporal lo solicitado por la cátedra, combinación que es fundamental más allá de si la complejidad de los servicios implementados es la mejor posible.