Proyecto de Estructura de Datos Grupo 4

Zúñiga Calderón Francini, Barrantes Chaves Jorge Fabian, Hernández Espinoza Kendal, Rojas Madrigal Frander
Ingeniería en Sistemas, Universidad Fidélitas
Costa Rica
fzuniga30280@ufide.ac.cr
jbarrantes70305@ufide.ac.cr
khernandez00143@ufide.ac.cr
frojas20190@ufide.ac.cr

Abstrac- **Documentation and explanation of the project developed by group number four of the Data Structure course, it focuses on the creation of a Java program that stores and modifies the information of clients and vehicles of a Rent a Car.**

Resumen

Documentación y explicación del proyecto desarrollado por el grupo número cuatro del curso Estructura de Datos, el mismo se enfoca en la creación de un programa en Java que almacena y modifica la información de clientes y vehículos de un Rent a Car.

I. INTRODUCCION

Este documento permitirá al lector comprender el proyecto realizado por estudiantes del curso Estructura de Datos y el mismo va enfocado en la creación de un programa que permite gestionar la información de un Rent a car, tanto de clientes como de los vehículos que poseen.

Lo principal es emplear correctamente el uso de las estructuras de datos utilizadas a lo largo del curso para llevar el desarrollo correcto del programa con base a lo solicitado en el anunciado y lograr cumplir con todos los requerimientos que serán necesarios para el desarrollo del código y elaborar una búsqueda extra de información para completar procesos como lo es la creación de una interfaz interactiva con el usuario y siempre pensando como si el código llegaría a ser utilizado por una empresa y un cliente donde tenga la necesidad de rentar un carro utilizando nuestro código y se demuestre el uso correcto y eficaz de las estructuras de datos como lo son los Árboles, Pilas, Colas y listas doblemente enlazadas donde este tipo de estructura de datos son de suma importancia para la elaboración correcta del código y lograr así cumplir las funciones solicitadas para un desarrollo completo del código y todo el recorrido del código se pueda ejecutar correctamente como lo son el alquiler del vehículo, devoluciones del vehículo, categorías de usuarios y no olvidar el registro correcto del cliente a la plataforma. Con esto queremos elaborar un código donde el cliente tenga todas las funciones que requiera funcionando correctamente para lograr alquilar un vehículo satisfactoriamente.

II. PROCEDIMIENTO PARA EL ENVIÓ DEL TRABAJO

II. PROBLEMA

Como parte de la reactivación económica que se está viviendo en Costa Rica posterior a las consecuencias sufridas por la pandemia, se han generado iniciativas para promover el turismo y que más nómadas digitales decidan venir a trabajar durante un periodo de tiempo en nuestro país. Una de las implicaciones de estas iniciativas, es que las personas requieran los servicios de renta de autos para desplazarse por el país. Este proyecto tiene el objetivo de implementar un sistema que permita gestionar los servicios que ofrece una empresa de renta de automóviles. A continuación, se detallan las funcionalidades que se espera que tenga el sistema.

III. OBJETIVOS

OBJETIVO GENERAL:

- Desarrollar un proyecto de software que permita cubrir la problemática planteada

OBJETIVOS ESPECÍFICOS:

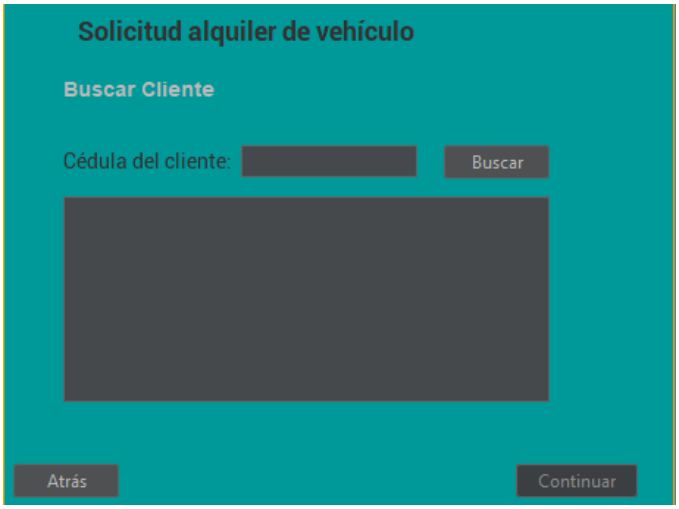
- Definir las estructuras de datos a utilizar en el programa de software
- Manipular los tipos de estructura de datos vistos en clases para la creación del programa.
- Simular el funcionamiento de una renta de autos mediante el software desarrollado.

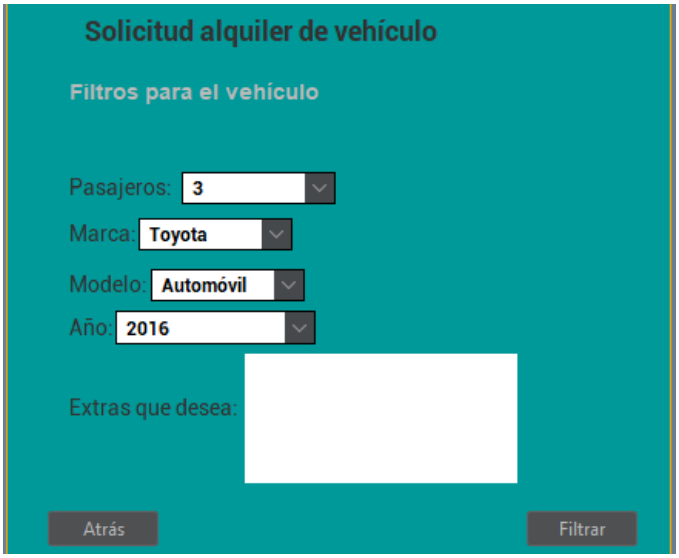
IV. MANUAL DE USUARIO

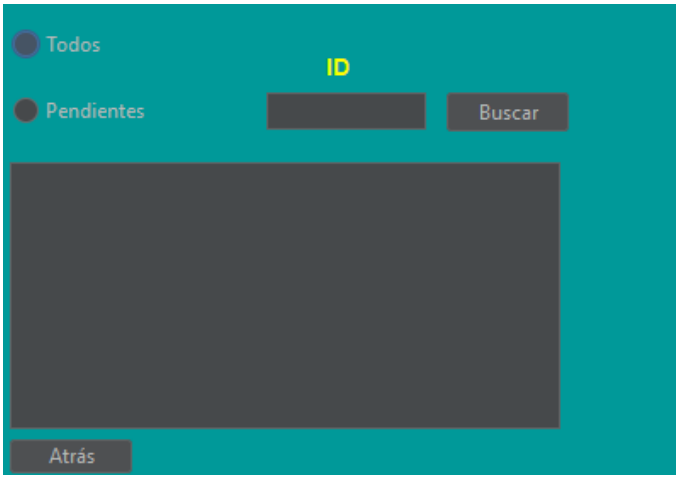
El aplicativo funciona a de la siguiente manera y estructura:

1) *Menu principal:* Contiene diferentes botones para acceder a las diferentes secciones (clientes, vehiculos, alquileres, estadisticas) así como botones para realizar funciones tales como atender alquiler, devolución, cargar y guardar.
2) Clientes: Poseerá opción de registrar un cliente o consultar cliente. Registrar despliega el formulario para ingresar los datos del cliente y registrarlo. Consultar permite ver todos los clientes, buscar uno en específico y poder editarlo o eliminarlo.
3) Vehículos: Poseerá opción de registrar un vehículo o consultar vehículo. Registrar despliega el formulario para ingresar los datos del vehículo y registrarlo. Consultar permite ver todos los vehículos, buscar uno en específico y poder editarlo o eliminarlo.
4) *Alquileres:* Poseerá opción de generar un alquiler o consultar alquiler. Generar despliega el formulario para generar alquiler, donde al asociar cliente, permite buscar un cliente seleccionarlo y de la misma manera con el vehículo a alquilar, posteriormente se genera el alquiler con todos los datos ingresados. Consultar permite ver todos los alquileres, buscar uno en específico o ver todos los alquileres pendientes por atender.
5) *Estadísticas:* Permite ver las estadísticas de los alquileres, como top 5 de clientes o vehículos con más alquileres, así como los montos totales y promedios según categorías de usuarios.
6) *Atender alquiler:* Permite procesar un alquiler recien registrado (acción de otorgar vehiculo al cliente)
7) *Devolución:* Permite recibir vehiculo y finalizar el alquiler
8) Cargar: Permite cargar los datos guardados de las estructuras de datos clientes, vehículos y alquileres.
9) Guardar: Permite guardar las estructuras de datos en txt fuera del aplicativo.

V. PRUEBAS DE FUNCIONALIDAD





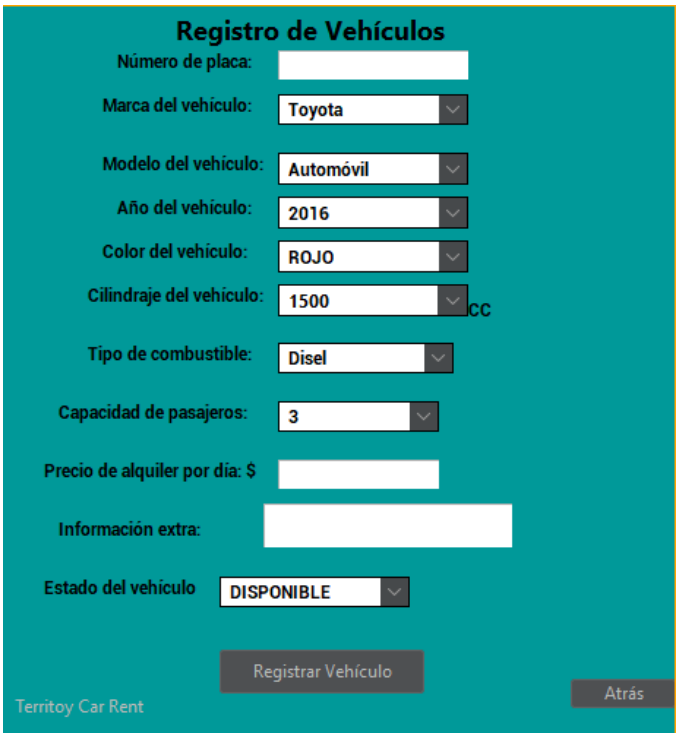


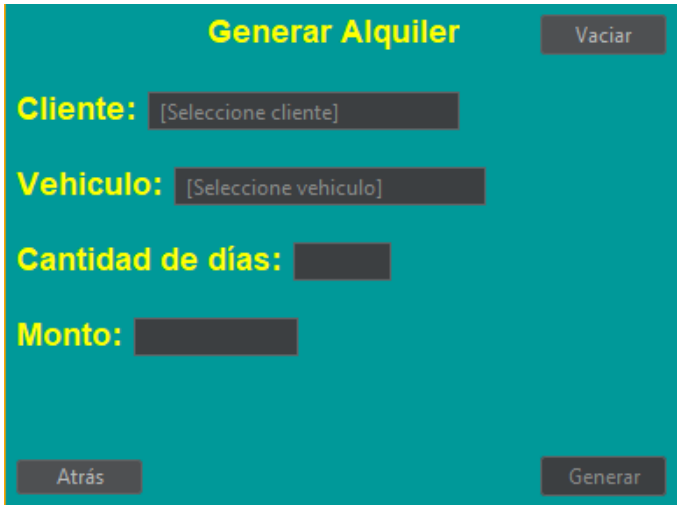

Clientes Registrados
Cédula
BUSCAR
Editar
Eliminar
Salir


Vehículos Registrados
Placa
BUSCAR
Editar
Eliminar
Salir


Cedula del cliente:
Placa de vehiculo:
Ingresar
Atrás
Devolucion


Editar Cliente
Cédula:
Nombre Completo:
Fecha de nacimiento:
Correo electrónico:
Categoría
Categoría
Guardar Cambios
Territoy Car Rent
Atrás


Registro de Vehículos
Número de placa:
Marca del vehículo:
Toyota
Modelo del vehículo:
Automóvil
Año del vehículo:
2016
Color del vehículo:
ROJO
Cilindraje del vehículo:
1500
CC
Tipo de combustible:
Disel
Capacidad de pasajeros:
3
Precio de alquiler por día: $
Información extra:
Estado del vehículo
DISPONIBLE
Registrar Vehículo
Territoy Car Rent
Atrás



CODIGO DE SUS RESPECTIVAS CLASES

Clases de Nodos

```java
package Clases;


public class Nodo_Alquiler {
    private Alquiler Prestamo;
    private Nodo_Alquiler Next;


     public Nodo_Alquiler() {
    }

    public Nodo_Alquiler(Alquiler Prestamo) {
        this.Prestamo = Prestamo;
    }


    public void setPrestamo(Alquiler Prestamo) {
        this.Prestamo = Prestamo;
    }

    public void setNext(Nodo_Alquiler Next) {
        this.Next = Next;
    }




    public Alquiler getPrestamo() {
        return Prestamo;
    }

    public Nodo_Alquiler getNext() {
        return Next;
    }


}
```

```java
package Clases;


public class Nodo_Cliente {
    private Cliente Persona;
    private Nodo_Cliente R;
    private Nodo_Cliente L;

    public Nodo_Cliente(Cliente Persona) {
        this.Persona = Persona;
    }



    public Cliente getPersona() {
        return Persona;
    }

    public Nodo_Cliente getR() {
        return R;
    }

    public Nodo_Cliente getL() {
        return L;
    }

    public void setPersona(Cliente Persona) {
        this.Persona = Persona;
    }

    public void setR(Nodo_Cliente R) {
        this.R = R;
    }

    public void setL(Nodo_Cliente L) {
        this.L = L;
    }


}
```

```java
package Clases;


public class Nodo_Vehiculo {
    private Vehiculo Carro;
    private Nodo_Vehiculo Next;
   private Nodo_Vehiculo Back;

    public Nodo_Vehiculo(Vehiculo Carro) {
        this.Carro = Carro;
    }

    public Nodo_Vehiculo() {
    }


    public Vehiculo getCarro() {
        return Carro;
    }

    public Nodo_Vehiculo getNext() {
        return Next;
    }

    public Nodo_Vehiculo getBack() {
        return Back;
    }

    public void setCarro(Vehiculo Carro) {
        this.Carro = Carro;
    }

    public void setNext(Nodo_Vehiculo Next) {
        this.Next = Next;
    }

    public void setBack(Nodo_Vehiculo Back) {
        this.Back = Back;
    }

}
```

Clase cliente

```java
public class Cliente {

    private int cédula;
    private String Nombre_Completo;
    private String fecha;
    private String correo;
    private String categoria; //ZAFIRRO/ORO/PLATA/BRONCE

    public Cliente(int cédula, String Nombre_Completo, String fecha, String correo, String categoria) {
        this.cédula = cédula;
        this.Nombre_Completo = Nombre_Completo;
        this.fecha = fecha;
        this.correo = correo;
        this.categoria = categoria;
    }



    public Cliente(int cédula, String Nombre_Completo, String fecha, String correo) {
        this.cédula = cédula;
        this.Nombre_Completo = Nombre_Completo;
        this.fecha = fecha;
        this.correo = correo;
        this.categoria = "BRONCE";
    }
```

Clase clientes

```java
public class Clientes {
    private Nodo_Cliente Root;

    public Nodo_Cliente getRoot() {
        return Root;
    }

    public void setRoot(Nodo_Cliente Root) {
        this.Root = Root;
    }


    public void insertar(Cliente persona){
        Nodo_Cliente newnodo = new Nodo_Cliente(persona);
        if(Root==null){
            Root=newnodo;
        }
        else{
            inserta(Root,persona);
        }
    }

    private void inserta(Nodo_Cliente node, Cliente value){
        Nodo_Cliente newnodo = new Nodo_Cliente(value);
        if(value.getCédula()<=node.getPersona().getCédula()){
            if(node.getL()==null){
                node.setL(newnodo);
            }
            else{
                inserta(node.getL(),value);
            }
        }
        else{
            if(node.getR()==null){
                node.setR(newnodo);
            }
            else{
                inserta(node.getR(),value);
            }
        }
    }
```

```java
public String  mostrar(){
        String datos="";
        ArrayList<Nodo_Cliente> Clientes=new ArrayList();
        if(Root != null){
            Clientes=inordenrecorrer(Root,Clientes);
            for(Nodo_Cliente node:Clientes){
                datos+="\nCédula: ["+node.getPersona().getCédula()+"]\n"
                    +"Nombre Completo: ["+ node.getPersona().getNombre_Completo()+"]\n"
                    +"Fecha de Nacimiento: ["+ node.getPersona().getFecha()+"]\n"
                    +"Correo Electrónico: ["+ node.getPersona().getCorreo()+"]\n"
                    +"Categoria: ["+ node.getPersona().getCategoria()+"]\n"
                    +"-------------------------------------------------------------------------------\n";
            }
        }
        else{
            System.out.println("Arbol vacio");
        }
        return "DATOS DE CLIENTES\n" + datos;
    }

    private ArrayList<Nodo_Cliente> inordenrecorrer(Nodo_Cliente node, ArrayList<Nodo_Cliente> clientes){

        if(node!=null){
            clientes=inordenrecorrer(node.getL(),clientes);
            clientes.add(node);
            clientes=inordenrecorrer(node.getR(),clientes);
            return clientes;
        }
        else{
            return clientes;
        }

    }

    public Cliente getPersona(int cedula){
        Nodo_Cliente node=Root;
        while(node!=null){
            int ced=node.getPersona().getCédula();
            if(cedula>ced){
            node=node.getR();
        }
        else if(cedula<ced){
            node=node.getL();
        }
        else if(cedula==ced){
            return node.getPersona();
        }
        }
        return null;
```

```java
public String printPersona(Cliente humano){
        String datos="DATOS DE CLIENTE";
        datos+="\nCédula: ["+humano.getCédula()+"]\n"
                    +"Nombre Completo: ["+ humano.getNombre_Completo()+"]\n"
                    +"Fecha de Nacimiento: ["+ humano.getFecha()+"]\n"
                    +"Correo Electrónico: ["+ humano.getCorreo()+"]\n"
                    +"Categoria: ["+ humano.getCategoria()+"]\n";
        return datos;
    }

    public void modificar(Cliente persona){
        int cedula=persona.getCédula();

        Nodo_Cliente node=Root;
        while(node!=null){
            int ced=node.getPersona().getCédula();
            if(cedula>ced){
            node=node.getR();
        }
        else if(cedula<ced){
            node=node.getL();
        }
        else if(cedula==ced){
            node.setPersona(persona);
            break;
        }
        }


    }
```

```java
public void eliminar(Cliente Humano){
        if(Root!=null){
            if(!General.Alquileres_Registrados.pendiente(Humano.getCédula())){
                Nodo_Cliente aux;
            ArrayList<Cliente> Clientes=new ArrayList();
            aux=getNodo(Humano);
            Clientes=eliminados(aux,Clientes);
            Clientes.remove(Clientes.size()-1);
            for(Cliente item:Clientes){
                General.Clientes_Registrados.insertar(item);
            }
            }
            else{
                JOptionPane.showMessageDialog(null,"El cliente aún tiene alquileres pendientes!", "ADVERTENCIA", JOptionPane.WARNING_MESSAGE);
            }
        }
    }
```

```
private void delete(int cedula){
    if(Root.getPersona().getCédula()==cedula){
        Root=null;
    }
    else{
        Nodo_Cliente node=Root;
    boolean fin=false;
    while(!fin && node!=null){
        int ced=node.getPersona().getCédula();
        if(cedula>ced){
            if(node.getR().getPersona().getCédula()==cedula){
                node.setR(null);
                fin=true;
            }
            else{
                node=node.getR();
            }

    }
    else if(cedula<ced){

        if(node.getL().getPersona().getCédula()==cedula){
                node.setL(null);
                fin=true;
            }
        else{
            node=node.getL();
        }

    }
    }
    }

}
private ArrayList<Cliente> eliminados(Nodo_Cliente node, ArrayList<Cliente> lista){
    if(node!=null){
        lista=eliminados(node.getL(),lista);

        lista=eliminados(node.getR(),lista);
        lista.add(node.getPersona());
        this.delete(node.getPersona().getCédula());
        return lista;
    }
    else{
        return lista;
    }
}
```

```
 private Nodo_Cliente getNodo(Cliente Humano){
     Nodo_Cliente node = Root;
     int cedula = Humano.getCédula();
     while(node!=null){
         int ced=node.getPersona().getCédula();
         if(cedula>ced){
         node=node.getR();
     }
     else if(cedula<ced){
         node=node.getL();
     }
     else if(cedula==ced){
         return node;
     }
     }
     return null;
 }
```

```
public void Guardar(){
    try{
       FileWriter fw=new FileWriter("Clientes.txt");
       BufferedWriter bw = new BufferedWriter(fw);
       PrintWriter pw = new PrintWriter(bw);
       ArrayList<Nodo_Cliente> Clientes = new ArrayList();
       String datos="";
       datos+=Root.getPersona().getCédula()+"\n"
               +Root.getPersona().getNombre_Completo()+"\n"
               +Root.getPersona().getFecha()+"\n"
               +Root.getPersona().getCorreo()+"\n"
               +Root.getPersona().getCategoria()+"\n";
       pw.println(datos);
       Clientes=inordenrecorrer(Root.getR(),Clientes);
       for(Nodo_Cliente item:Clientes){
           datos="";
           datos+=item.getPersona().getCédula()+"\n"
               +item.getPersona().getNombre_Completo()+"\n"
               +item.getPersona().getFecha()+"\n"
               +item.getPersona().getCorreo()+"\n"
               +item.getPersona().getCategoria()+"\n";
            pw.println(datos);
       }

       Clientes=inordenrecorrer(Root.getL(),Clientes);
       for(Nodo_Cliente item:Clientes){
           datos="";
           datos+=item.getPersona().getCédula()+"\n"
               +item.getPersona().getNombre_Completo()+"\n"
               +item.getPersona().getFecha()+"\n"
               +item.getPersona().getCorreo()+"\n"
               +item.getPersona().getCategoria()+"\n";
            pw.println(datos);
       }
       pw.flush();
       pw.close();
       }
       catch(Exception E){

    }

}
```

```
public void Cargar(){
     try{
            FileReader fr = new FileReader("Clientes.txt");
            BufferedReader br = new BufferedReader(fr);
            String texto="";
            while(texto!=null){
                int cédula=Integer.parseInt(br.readLine());
                String Nombre_Completo=br.readLine();
                String fecha=br.readLine();
                String correo=br.readLine();
                String categoria=br.readLine();
                Cliente humano = new Cliente(cédula,Nombre_Completo,fecha,correo,categoria);
                this.insertar(humano);
                texto=br.readLine();
            }
        }
        catch(Exception E){

        }
}
```

Clase Alquiler

```
public class Alquiler {
    private String ID;
    private Cliente Persona;
    private Vehiculo Carro;
    private int Dias;
    private String Estado;
    private double monto;
    private String Fecha;

    public Alquiler(String ID, Cliente Persona, Vehiculo Carro, int Dias, String Estado, double monto, String Fecha) {
        this.ID = ID;
        this.Persona = Persona;
        this.Carro = Carro;
        this.Dias = Dias;
        this.Estado = Estado;
        this.monto = monto;
        this.Fecha = Fecha;
    }



    public Alquiler(Cliente Persona, Vehiculo Carro, int Dias, double monto) {
        this.Persona = Persona;
        this.Carro = Carro;
        this.Dias = Dias;
        this.Estado = "REGISTRADO";
        this.monto = monto+(monto*0.13);
        boolean val=false;
        while(!val){
            Random n = new Random();
            this.ID="A-"+String.valueOf(n.nextInt(1000));
            val=validation(this.ID);
        }
        this.Fecha=new Date().toString();
    }
```

Clase Alquileres

```
public class Alquileres {
    private Nodo_Alquiler Head;

    public Nodo_Alquiler getHead() {
        return Head;
    }

    public void setHead(Nodo_Alquiler Head) {
        this.Head = Head;
    }

    public void Insertar(Alquiler alquiler){
        Nodo_Alquiler newnodo = new Nodo_Alquiler(alquiler);
        if(Head==null){
            Head=newnodo;
        }
        else{
            newnodo.setNext(Head);
            Head=newnodo;
        }
        if(alquiler.getDias()>=30){
            switch(alquiler.getPersona().getCategoria()){
                case "BRONCE":
                        alquiler.getPersona().setCategoria("PLATA");
                    break;
                case "PLATA":
                    alquiler.getPersona().setCategoria("ORO");

                    break;
                case "ORO":
                    alquiler.getPersona().setCategoria("ZAFIRO");
                    break;
                default:

                    break;
            }


        }
    }
```

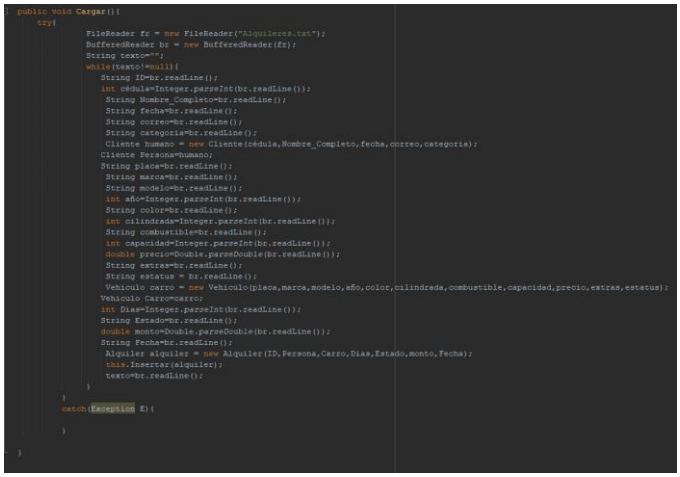
```
public void modificar(String ID, String estatus){
    Nodo_Alquiler aux = Head;
    boolean modificado = false;
    if(Head!=null){
        while(aux!=null & !modificado){
            if(aux.getPrestamo().getID().equals(ID)){
                aux.getPrestamo().setEstado(estatus);
                modificado = true;
            }
            else{
                aux=aux.getNext();
            }
        }
        if(modificado){
            //ALQUILER PROCESADO Y ACTUALIZADO
        }
        else{
            //ERROR
        }
    }
}
public String mostrar(){
    String datos="";
    if(Head!=null){
        Nodo_Alquiler aux=Head;
        while(aux!=null){
            Alquiler a=aux.getPrestamo();
            datos+="\nID: ["+a.getID()+"]\n"
                    +"Cliente: [" + a.getPersona().getNombre_Completo() +"]\n"
                    +"Vehiculo: [" + a.getCarro().getPlaca()+"]\n"
                    +"Dias de Alquiler: ["+a.getDias()+"]\n"
                    +"Estado de Alquiler:  "+ a.getEstado()+"]\n"
                    +"Precio a Paga: ["+a.getMonto()+"]\n"
                    +"Fecha: ["+a.getFecha()+"]\n"
                    +"----------------------------------------------------------------------";
            aux=aux.getNext();
        }
    }
    else{
        datos="No hay alquileres registrados";
    }
    return datos;
}
```

```
public String TodosDe(int cedula){
    String datos="\nALQUILERES\n";
     if(Head!=null){
        Nodo_Alquiler aux=Head;
        while(aux!=null){
            Alquiler a=aux.getPrestamo();
            if(a.getPersona().getCédula()==cedula){
                datos+="\nID: ["+a.getID()+"]\n"
                      +"Cliente: [" + a.getPersona().getNombre_Completo() +"]\n"
                      +"Vehiculo: [" + a.getCarro().getPlaca()+"]\n"
                      +"Dias de Alquiler: ["+a.getDias()+"]\n"
                      +"Estado de Alquiler:  "+ a.getEstado()+"]\n"
                      +"Precio a Paga: ["+a.getMonto()+"]\n"
                      +"Fecha: ["+a.getFecha()+"]\n"
                      +"----------------------------------------------------------------------";

            }

                aux=aux.getNext();

        }
    }
    return datos;
}

public boolean pendiente(int cedula){
     if(Head!=null){
        Nodo_Alquiler aux=Head;
        while(aux!=null){
            Alquiler a=aux.getPrestamo();
            if(a.getPersona().getCédula()==cedula && !"FINALIZADO".equals(a.getEstado())){
                return true;
            }

                aux=aux.getNext();

        }
    }
     return false;
}
```

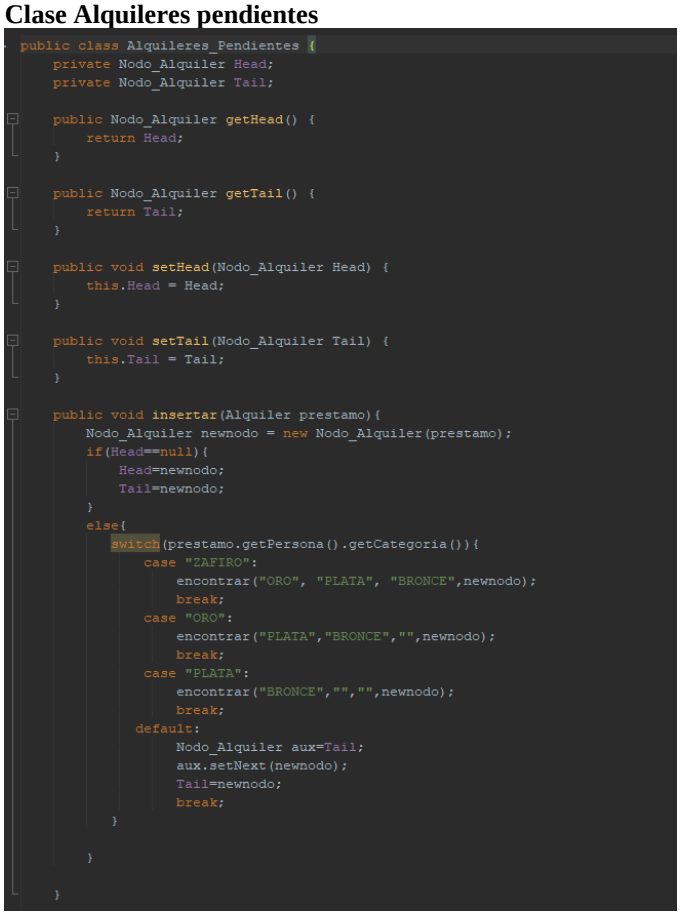
```
public Nodo_Alquiler buscar(String ID){
    Nodo_Alquiler aux;
    boolean encontrado = false;
    if(Head!=null){
        aux = Head;
        while(aux!=null & !encontrado){
            if(aux.getPrestamo().getID().equals(ID)){
                return aux;
            }
            else{
                aux=aux.getNext();
            }
        }
    }
    else{
        aux=null;
    }
    return aux;
}

public Alquiler devolution(int ced, String placa){
    if(Head!=null){
        Nodo_Alquiler aux=Head;
        while(aux!=null){
            if(aux.getPrestamo().getPersona().getCédula()==ced && placa.equals(aux.getPrestamo().getCarro().getPlaca()) &&"PROCESADO".equals(aux.getPrestamo()).getEstado()){
                return aux.getPrestamo();
            }
            else{
                aux=aux.getNext();
            }
        }
        return null;
    }
    else{
        return null;
    }
}
```

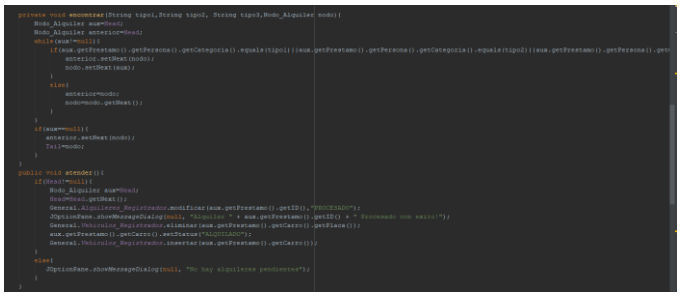
```
public void Guardar(){
        try{
            FileWriter fw=new FileWriter("Alquileres.txt");
            BufferedWriter bw = new BufferedWriter(fw);
            PrintWriter pw = new PrintWriter(bw);

            Nodo_Alquiler nodo = Head;

            while(nodo!=null){
                String datos="";
                Alquiler a = nodo.getPrestamo();
                datos+=a.getID()+"\n"
                        +a.getPersona().getCédula()+"\n"
                      +a.getPersona().getNombre_Completo()+"\n"
                      +a.getPersona().getFecha()+"\n"
                      +a.getPersona().getCorreo()+"\n"
                      +a.getPersona().getCategoria()+"\n"
                          +a.getCarro().getPlaca()+"\n"
                          +a.getCarro().getMarca()+"\n"
                          +a.getCarro().getModelo()+"\n"
                          +a.getCarro().getAño()+"\n"
                          +a.getCarro().getColor()+"\n"
                          +a.getCarro().getCilindrada()+"\n"
                          +a.getCarro().getCombustible()+"\n"
                          +a.getCarro().getCapacidad()+"\n"
                          +a.getCarro().getPrecio()+"\n"
                          +a.getCarro().getExtras()+"\n"
                          +a.getCarro().getStatus()+"\n"
                          +a.getDias()+"\n"
                          +a.getEstado()+"\n"
                          +a.getMonto()+"\n"
                          +a.getFecha()+"\n";
                pw.println(datos);
                nodo=nodo.getNext();
                }
                pw.flush();
                pw.close();
                }
                catch(Exception E){

            }
    }
```

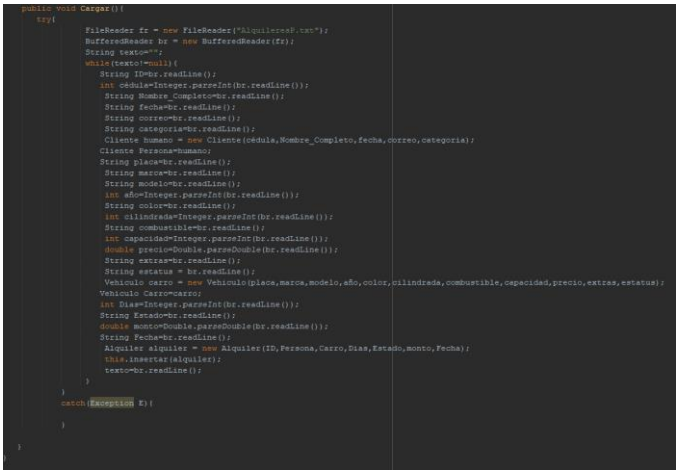
```
public int cantidad(Object object){
        int pepe=0; //Contador
        if(Head!=null){
            Nodo_Alquiler aux = Head;
            while(aux!=null){
                if(aux.getPrestamo().getCarro()==object || aux.getPrestamo().getPersona()==object){
                    pepe+=1;
                }
                aux=aux.getNext();
            }
        }

        return pepe;
    }


public void montopromedio(){
    if(Head!=null){
        ArrayList<Double> montos=new ArrayList();
         ArrayList<Double> promedios=new ArrayList();
         ArrayList<String> categorias=new ArrayList();
         categorias.add("ZAFIRO");
         categorias.add("ORO");
         categorias.add("PLATA");
         categorias.add("BRONCE");
         Interfaz.Estadisticas.Categorias=categorias;
     for(String item:categorias){
         double monto=0.000;
         double promedio=0.000;
         Nodo_Alquiler aux=Head;
         while(aux!=null){
             if(aux.getPrestamo().getPersona().getCategoria().equals(item)){
                 monto+=aux.getPrestamo().getMonto();
                 promedio+=1;
             }
             aux=aux.getNext();
         }
         promedio=monto/promedio;
         montos.add(monto);
         promedios.add(promedio);
     }
     Interfaz.Estadisticas.Montos=montos;
     Interfaz.Estadisticas.Promedios=promedios;

    }

}
```



Clase Alquileres pendientes







Clase Vehículo

Clase vehículos

```
public class Vehiculos {
    private Nodo_Vehiculo Head;
    private Nodo_Vehiculo Tail;

    public Nodo_Vehiculo getHead() {
        return Head;
    }

    public Nodo_Vehiculo getTail() {
        return Tail;
    }

    public void setHead(Nodo_Vehiculo Head) {
        this.Head = Head;
    }

    public void setTail(Nodo_Vehiculo Tail) {
        this.Tail = Tail;
    }
```

```
//Insertar/crear vehiculo
public void insertar(Vehiculo carro){
    Nodo_Vehiculo newnodo = new Nodo_Vehiculo(carro);
    if(Head ==  null){
        Head=newnodo;
    }
    else{
        if(Head.getNext()==null){
            Tail=newnodo;
            Tail.setBack(Head);
            Head.setNext(Tail);
        }
        else{
            if("DISPONIBLE".equals(carro.getStatus())){
                Nodo_Vehiculo aux=Head;
                aux.setBack(newnodo);
                newnodo.setNext(aux);
                Head=newnodo;
            }
            else{
                Nodo_Vehiculo aux=Tail;
                aux.setNext(newnodo);
                newnodo.setBack(aux);
                Tail=newnodo;
                    }
        }

    }
}
```

```
//Modificar vehiculo
public void modificar (Vehiculo carro){
    Nodo_Vehiculo Presente=Head;
    boolean encontrado=false;
    if(Head!=null){
    while(Presente!=null && !encontrado){
        if(carro.getPlaca().equals(Presente.getCarro().getPlaca())){

            Presente.setCarro(carro);

            encontrado=true;
        }
        else{
            Presente=Presente.getNext();
        }
            if(!encontrado){
            //No se encontró vehiculo
        }
        else{
            //Vehiculo encontrado y modificado
        }
        }
    }
}
```

```
//Eliminar vehiculo
public void eliminar (String placa){
    Nodo_Vehiculo Presente=Head;
    boolean encontrado=false;
    if(Head!=null){
    while(Presente!=null){
        if(placa.equals(Presente.getCarro().getPlaca())){
            if(Presente==Head){
                try{
                    Head=Head.getNext();
                    Head.setBack(null);
                }
                catch(NullPointerException ex){
                    Head=null;
                }
            }
            if(Presente==Tail){
                try{
                    Tail=Tail.getBack();
                    Tail.setNext(null);
                }
                catch(NullPointerException ex){
                    Tail=null;
                }

            }
            else{
                Presente.getBack().setNext(Presente.getNext());
                Presente.getNext().setBack(Presente.getBack());
            }
            Presente=null;
            encontrado=true;
        }
        else{
            Presente=Presente.getNext();
        }
        if(!encontrado){
            //No se encontró vehiculo
        }
        else{
            //Vehiculo encontrado y eliminado
        }

        }
    }
}
```

```
//Leer vehiculo
public String buscar(String placa){
    Nodo_Vehiculo Presente=Head;
    boolean encontrado=false;
    String datos="DATOS DE AUTO\n";
    if(Head!=null){
    while(Presente!=null  && !encontrado){
        if(placa.equals(Presente.getCarro().getPlaca())){
            Vehiculo carro = Presente.getCarro();
            datos+="\nEstado: ["+carro.getStatus()+"]\n"
                + "Placa: [" + carro.getPlaca() + "]\n"
                + "Marca: [" + carro.getMarca() + "]\n"
                + "Modelo: "+carro.getModelo() + "]\n"
                + "Año: [" + carro.getAño()+"]\n"
                + "Color: [" + carro.getColor() + "]\n"
                + "Cilindrada: ["+carro.getCilindrada()+"]\n"
                + "Combustible: ["+carro.getCombustible()+"]\n"
                + "Capacidad: ["+carro.getCapacidad()+"]\n"
                + "Precio: ["+carro.getPrecio()+"]\n"
                + "Extras: ["+carro.getExtras()+"]\n"
                +"-----------------------------------------------------------------";
            encontrado=true;
        }
        else{
            Presente=Presente.getNext();
        }
        if(!encontrado){
            datos="Vehiculo no encontrado";
        }

        }
    }
    else{
        datos="NO HAY VEHICULOS";
    }
    return datos;
}
```

```
public String mostrar(){
    Nodo_Vehiculo Presente =Head;
    String datos="\nDATOS DE VEHICULOS\n";
    if(Head!=null){
    while(Presente!=null){
        Vehiculo carro = Presente.getCarro();
        datos+="\nEstado: ["+carro.getStatus()+"]\n"
                + "Placa: [" + carro.getPlaca() + "]\n"
                + "Marca: [" + carro.getMarca() + "]\n"
                + "Modelo: "+carro.getModelo() + "]\n"
                + "Año: [" + carro.getAño()+"]\n"
                + "Color: [" + carro.getColor() + "]\n"
                + "Cilindrada: ["+carro.getCilindrada()+"]\n"
                + "Combustible: ["+carro.getCombustible()+"]\n"
                + "Capacidad: ["+carro.getCapacidad()+"]\n"
                + "Precio: ["+carro.getPrecio()+"]\n"
                + "Extras: ["+carro.getExtras()+"]\n"
                +"-----------------------------------------------------------------";
        Presente=Presente.getNext();
    }
    }
    else{
        datos="NO HAY VEHICULOS";
    }
    return datos;
}
    public Vehiculo getauto(String placa){
        Nodo_Vehiculo Presente=Head;
    boolean encontrado=false;
    Vehiculo auto=null;
    if(Head!=null){
    while(Presente!=null  && !encontrado){
        if(placa.equals(Presente.getCarro().getPlaca())){
            auto=Presente.getCarro();
            encontrado=true;
        }
        else{
            Presente=Presente.getNext();
        }
        if(!encontrado){
            auto=null;
        }

        }
    }
    return auto;
    }
```

```
public ArrayList<Nodo_Vehiculo> filtrar(String model, String branch, int year, int passage, Nodo_Vehiculo node, ArrayList<Nodo_Vehiculo> lista){
    if(node!=null){
        Vehiculo carro=node.getCarro();
        if("DISPONIBLE".equals(carro.getStatus())){
            if(!carro.getModelo().equals(model)&&!carro.getMarca().equals(branch)&&carro.getAño()!=year&&carro.getCapacidad()!=passage){
                filtrar(model,branch,year,passage,node.getNext(),lista);
            }
            else{
                lista.add(node);
                filtrar(model,branch,year,passage,node.getNext(), lista);
            }
        }
        else{
            filtrar(model,branch,year,passage,node.getNext(), lista);
        }

    }
    return lista;
}


public ArrayList<Nodo_Vehiculo>Top5() {
    ArrayList<Nodo_Vehiculo> top = new ArrayList();
    if(Head!=null){
        Nodo_Vehiculo aux=Head;
        while(aux!=null){
            if(top.isEmpty()){
                top.add(aux);
            }
            else{
                top=comparar(aux,top);
            }
            aux=aux.getNext();
        }
    }
    else{
        top=null;
    }
    return top;
}
```

```
private ArrayList<Nodo_Vehiculo> comparar(Nodo_Vehiculo aux,ArrayList<Nodo_Vehiculo> top){
                boolean fin=false;
                int pepe=0;
                int can2=General.Alquileres_Registrados.cantidad(aux);
                while(!fin){
                    int can1=General.Alquileres_Registrados.cantidad(top.get(pepe));

                    if(can2>can1){
                        if(top.size()==5){
                            for(int i=4;i>=top.indexOf(pepe);i--){
                                if(i>top.indexOf(pepe)){
                                    top.set(i, top.get(i-1));
                                }
                                else{
                                    top.set(i, aux);
                                }
                            }
                        }
                        else{
                            for(int i=top.size()-1;i>top.indexOf(pepe);i--){
                                if(i>top.indexOf(pepe)){
                                    if(i==top.size()-1){
                                        top.add(top.get(i));
                                    }
                                        top.set(i, top.get(i-1));
                                }
                                else{
                                    top.set(i, aux);
                                }
                            }
                        }
                        fin=true;
                    }
                    else{
                        pepe+=1;
                        if(pepe>=top.size()){
                            fin=true;
                            if(pepe<5){
                                top.add(aux);
                            }
                        }
                    }
                }
                return top;
    }
```

```
    public void Guardar(){
        try{
        FileWriter fw=new FileWriter("Vehiculos.txt");
        BufferedWriter bw = new BufferedWriter(fw);
        PrintWriter pw = new PrintWriter(bw);

        Nodo_Vehiculo nodo = Head;

        while(nodo!=null){
            String datos="";
            Vehiculo carro = nodo.getCarro();
            datos+=carro.getPlaca()+"\n"
                    +carro.getMarca()+"\n"
                    +carro.getModelo()+"\n"
                    +carro.getAño()+"\n"
                    +carro.getColor()+"\n"
                    +carro.getCilindrada()+"\n"
                    +carro.getCombustible()+"\n"
                    +carro.getCapacidad()+"\n"
                    +carro.getPrecio()+"\n"
                    +carro.getExtras()+"\n"
                    +carro.getStatus()+"\n";
        pw.println(datos);
        nodo=nodo.getNext();
        }
        pw.flush();
        pw.close();
        }
        catch(Exception E){

    }

    }
```

```
public void Cargar(){
    try{
        FileReader fr = new FileReader("Vehiculos.txt");
        BufferedReader br = new BufferedReader(fr);
        String texto="";
        while(texto!=null){
            String placa=br.readLine();
            String marca=br.readLine();
            String modelo=br.readLine();
            int año=Integer.parseInt(br.readLine());
            String color=br.readLine();
            int cilindrada=Integer.parseInt(br.readLine());
            String combustible=br.readLine();
            int capacidad=Integer.parseInt(br.readLine());
            double precio=Double.parseDouble(br.readLine());
            String extras=br.readLine();
            String estatus = br.readLine();
            Vehiculo carro = new Vehiculo(placa,marca,modelo,año,color,cilindrada,combustible,capacidad,precio,extras,estatus);
            this.insertar(carro);
            texto=br.readLine();
        }
    }
    catch(Exception E){

    }
}
```

VI. DISEÑO DEL PROGRAMA

A) Librerías utilizadas del proyecto

1) *Javax.Swing:* Utilizada para poder crear pequeñas y breves ventanas emergentes, para ahorrarnos trabajo de crear ventanas extras para cada pequeña interacción de los usuarios con el aplicativo, por lo que optamos por utilizar el JOptionPane de la librería para estos propósitos.

2) *Java.util:* Esta librería nos permite utilizar ArrayList<> que es una lista muy versátil y fácil de utilizar. Dado que el proyecto se centra específicamente en estructuras de datos desarrolladas por nosotros, ArrayList<> no almacena datos importantes del proyecto, solo es utilizado como una mera herramienta para facilitar algunas transacciones, más sin embargo seguimos explotando al máximo todas las ventajas de las estructuras de datos manuales.

3) Java.io: Esta librería permite interactuar con funciones de archivos, nosotros la empleamos para poder generar archivos txt para almacenar las estructuras de datos con la finalidad de que éstas perduren aun cuando el programa haya sido cerrado.

B) Estructuras de datos

1) *Vehículos:* decidimos optar por emplear listas doblemente enlazadas. Por iniciativa nuestra, decidimos ordenar los vehículos de la siguiente manera: Los vehículos con estado DISPONIBLE estarán al inicio de la lista, ya que serán añadidos al Head, de los contrario estos serán añadidos al Tail de la lista. También decidimos que fuera doblemente enlazada, dado que al eliminar un vehículo se debe interactuar con el nodo anterior y siguiente del vehículo por lo que era más sencillo si era doblemente enlazada.

2) *Clientes:* optamos por la estructura de datos tipo Árbol binario, para de esta manera ordenar clientes de acuerdo con su cedula, una variable entera no modificable fácilmente comparable para ordenar de acuerdo con cedula mayor o menor. A pesar de que eliminar elementos de un árbol es difícil, la estructura era perfecta para aplicarla a clientes dado que los clientes no se eliminan con frecuencia.

3) *Alquileres:* optamos por dos estructuras de datos, pila y cola. La pila es para almacenar todos los alquileres y la cola sirve para almacenar alquileres por procesar y otorgar vehículo al cliente. Lo decidimos de esa manera ya que los alquileres no son eliminables, y se guarda un historial de acuerdo con orden de llegada de más antiguo a más reciente o viceversa, por lo que la pila era la más adecuada para estas especificaciones. Mientras que para los alquileres pendientes sería una cola dado que el primero en entrar es el primero en salir, por ende, era provechoso para nosotros aplicar dicha estructura en el proyecto.

VII. ANÁLISIS DE RESULTADOS

Los resultados obtenidos a nivel general fueron satisfactorios, ya que logramos crear un programa funcional que cumpliera con las funcionalidades básicas según lo demandaba el problema. Sin embargo, existieron detalles que no se lograron implementar de la mejor manera, por ejemplo, los ranking de la sección estadísticas, a veces presentan errores y no funciona el 100% de las veces, pero el algoritmo si esta operativo y se consiguen resultados afines pero no son completo o precisos todo el tiempo

De la misma manera por temas de tiempo y organización algunos detalles no se lograron implementar, tales como algunas validaciones o interfaces más llamativas y atractivas, entre otros, solo logramos abarcar lo básico del proyecto, pero de igual manera concluimos que logramos unos resultados satisfactorios.

Concluimos que los resultados obtenidos cumplen en su mayoría con lo solicitado, más sin embargo no cumplen al 100%, existen detalles que se nos escaparon, por lo que hay todavía mucho margen de mejora, pero reiteramos que a pesar de ello obtuvimos un resultado satisfactorio desde nuestro punto de vista como desarrolladores.

VIII. CONCLUSIÓN

Tras la culminación de nuestro proyecto de desarrollo de software, hemos logramos poner en practica todos los fundamentos de estructuras de datos, durante el proceso hemos aprendido diversas maneras de aplicar las estructuras de datos en la búsqueda de soluciones a diferentes problemáticas. A pesar de los obstáculos que tuvimos durante el desarrollo, logramos llegar a una conclusión satisfactoria del proyecto y nos queda de aprendizaje y desarrollo profesional para próximos proyectos de programación