

# PROYECTO BIMESTRAL

CARLOS VALLADARES  
MARTIN RUIZ

# GESTION DE BUSES

Desarrollar un sistema de gestión que permita monitorear, registrar y optimizar los recorridos de los autobuses de la Universidad Técnica Particular de Loja (UTPL) en toda la ciudad. El sistema ayudará a administrar eficientemente las paradas, horarios y rutas de los autobuses para mejorar la calidad del servicio ofrecido a los usuarios.

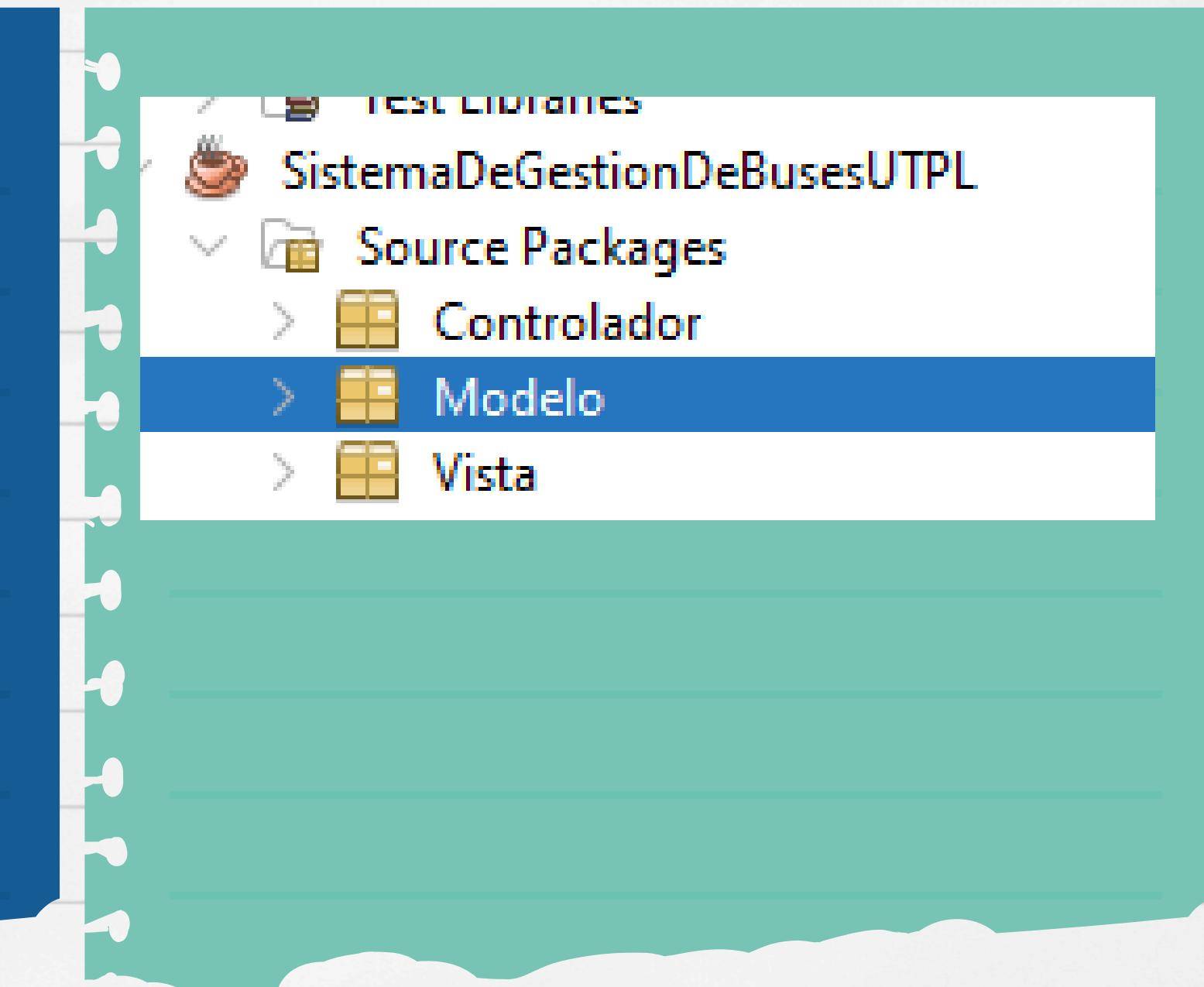
## Características

- Registro de paradas:
- Registro de horarios:
- Gestión de rutas
- Optimización de rutas
- Interfaz de usuario:

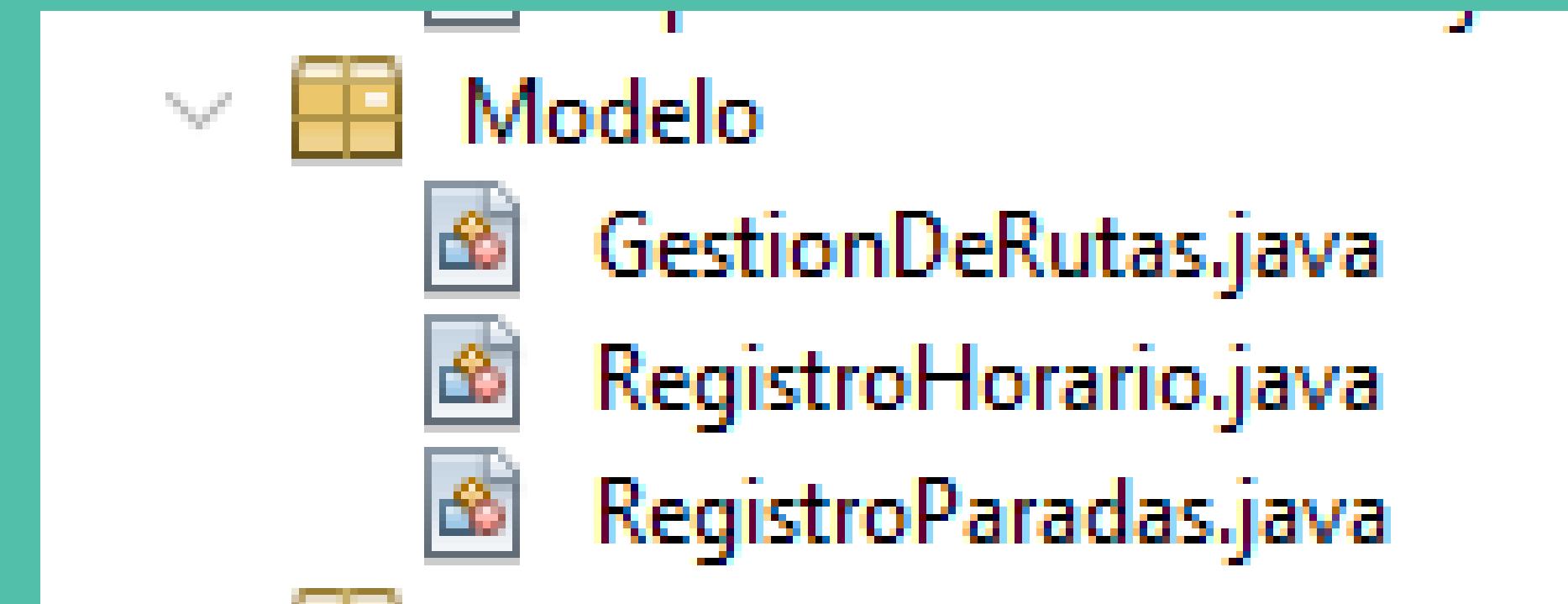


# ARQUITECTURA MVC

MODELO  
VISTA  
CONTROLADOR



# MODELO



# GESTION RUTAS

## Variables

```
public class GestionDeRutas implements Serializable {  
  
    private String nombreRuta;  
    private ArrayList<RegistroParadas> paradas;  
    private ArrayList<RegistroHorario> horarios;
```

## Constructor

```
public GestionDeRutas(String nombreRuta, ArrayList<RegistroParadas> paradas,  
                      ArrayList<RegistroHorario> horarios) {  
    this.nombreRuta = nombreRuta;  
    this.paradas = paradas;  
    this.horarios = horarios;  
  
}  
  
public GestionDeRutas(String nombreRuta) {  
    this.nombreRuta = nombreRuta;  
}
```

## Metodos getter y setter

```
public GestionDeRutas(String nombreRuta) {  
    this.nombreRuta = nombreRuta;  
}  
  
public ArrayList<RegistroParadas> obtenerParadas() {  
    return paradas;  
}  
  
public ArrayList<RegistroHorario> obtenerHorarios() {  
    return horarios;  
}  
  
public String getNombreRuta() {  
    return nombreRuta;  
}
```

## toString

## toString

```
@Override  
public String toString() {  
    String acumulador = "+ " + nombreRuta;  
  
    if (paradas == null) {  
        return nombreRuta;  
    }  
  
    for (int i = 0; i < paradas.size(); i++) {  
        acumulador = String.format(format: "%s\n - Parada: %s\n - Horario: %s\n",  
                                    args: acumulador,  
                                    args: paradas.get(index: i),  
                                    args: horarios.get(index: i));  
    }  
    return acumulador;  
}
```

# REGISTRO HORARIO

## VARIABLES

```
private double horaLlegada;  
private double horaSalida;
```

## CONSTRUCTOR

```
public RegistroHorario(double horaSalida, double horaLlegada) {  
    this.horaSalida = horaSalida;  
    this.horaLlegada = horaLlegada;  
}
```

## METODOS GETTER Y SETTER

```
public void setHoraLlegada(double horaLlegada) {  
    this.horaLlegada = horaLlegada;  
}  
  
public void setHoraSalida(double horaSalida) {  
    this.horaSalida = horaSalida;  
}
```

## TOSTRING

```
@Override  
public String toString() {  
    return "[Salida: " + horaSalida + ", Llegada: " + horaLlegada + "]";  
}
```

## Variables

```
private String nombre;  
private String ubicacion;  
private ArrayList<RegistroHorario> horarios;  
private ArrayList<GestionDeRutas> rutas;
```

## Constructores

```
public RegistroParadas(String nombre, String ubicacion,ArrayList<RegistroHorario> horarios,  
ArrayList<GestionDeRutas> rutas) {  
  
    this.nombre = nombre;  
    this.ubicacion = ubicacion;  
    this.horarios = horarios;  
    this.rutas = rutas;  
}
```

## Variables

## Métodos getter y setter

```
public ArrayList<RegistroHorario> obtenerHorarios() {  
    return horarios;  
}  
  
public ArrayList<GestionDeRutas> obtenerRutas() {  
    return rutas;
```

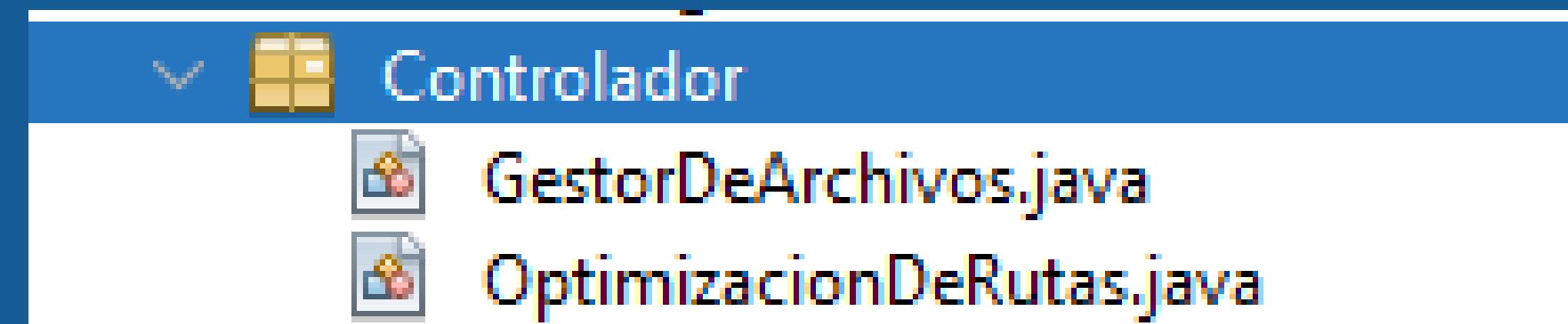
# REGISTRO PARADAS

```
public RegistroParadas(String nombre, String ubicacion) {  
    this.nombre = nombre;  
    this.ubicacion = ubicacion;  
}
```

## toString

```
@Override  
public String toString() {  
    String acumulador = "+ " + nombre + ", ubicado en " + ubicacion;  
    if (rutas == null) {  
        return nombre + ", ubicado en " + ubicacion;  
    }  
    for (int i = 0; i < rutas.size(); i++) {  
        acumulador = String.format(format: "%s\n - Ruta: %s\n - Horario: %s\n",  
        args: acumulador,  
        args: rutas.get(index: i),  
        args: horarios.get(index: i));  
    }  
    return acumulador;  
}
```

# CONTROLADOR



# OPTIMIZACION DE RUTAS

```
public static String editarHorarios(String nombreRuta, ArrayList<GestionDeRutas> todasLasRutas,
    ArrayList<RegistroParadas> todasLasParadas) {
    Scanner entrada = new Scanner(System.in);
    double nuevaHoraSalida;
    double nuevaHoraLlegada;
    int n;

    for (int i = 0; i < todasLasParadas.size(); i++) {
        for (int j = 0; j < todasLasParadas.get(index(i).obtenerHorarios().size(); j++) {
            if (todasLasParadas.get(index(i)).obtenerRutas().get(index(j)).getNombreRuta().equals(index(nombreRuta))) {
                System.out.println("Ingrese la nueva hora de salida (HH,MM):");
                nuevaHoraSalida = entrada.nextDouble();
                System.out.println("Ingrese la nueva hora de llegada (HH,MM):");
                nuevaHoraLlegada = entrada.nextDouble();

                todasLasParadas.get(index(i)).obtenerHorarios().get(index(j)).setHoraSalida(hourSalida: nuevaHoraSalida);
                todasLasParadas.get(index(i)).obtenerHorarios().get(index(j)).setHoraLlegada(hourLlegada: nuevaHoraLlegada);
            }
        }
    }

    for (int i = 0; i < todasLasRutas.size(); i++) {
        for (int j = 0; j < todasLasRutas.get(index(i)).obtenerHorarios().size(); j++) {
            if (todasLasRutas.get(index(i)).getNombreRuta().equals(index(nombreRuta))) {
                System.out.println("Ingrese la nueva hora de salida (HH,MM):");
                nuevaHoraSalida = entrada.nextDouble();
                System.out.println("Ingrese la nueva hora de llegada (HH,MM):");
                nuevaHoraLlegada = entrada.nextDouble();

                todasLasRutas.get(index(i)).obtenerHorarios().get(index(i)).setHoraSalida(hourSalida: nuevaHoraSalida);
                todasLasRutas.get(index(i)).obtenerHorarios().get(index(i)).setHoraLlegada(hourLlegada: nuevaHoraLlegada);
            }
        }
    }
}
```

# GESTOR ARCHIVOS

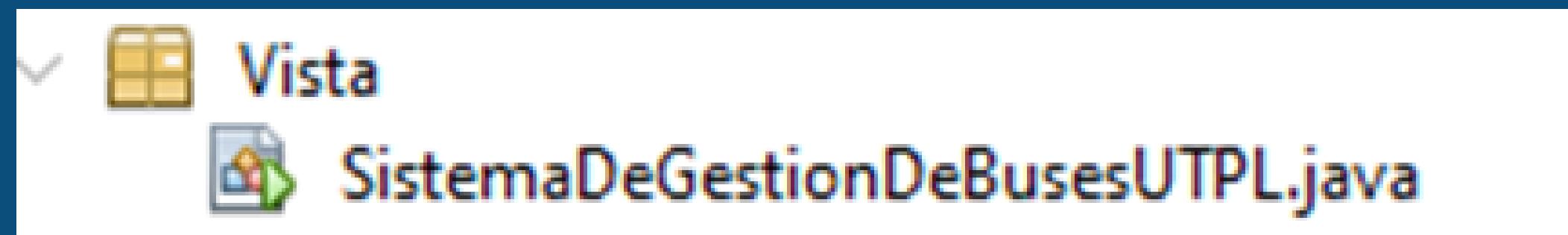
## GUARDAR EN ARCHIVOS

```
public class GestorDeArchivos {  
    // Guardar lista de objetos en archivo .dat  
    public static void guardarEnArchivo(String nombreArchivo, ArrayList<?> lista) {  
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nombreArchivo))) {  
            oos.writeObject(lista);  
            System.out.println("Datos guardados en " + nombreArchivo);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

## CARGAR DESDE ARCHIVO

```
@SuppressWarnings("unchecked")  
public static <T> ArrayList<T> cargarDesdeArchivo(String nombreArchivo) {  
    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nombreArchivo))) {  
        return (ArrayList<T>) ois.readObject();  
    } catch (IOException | ClassNotFoundException e) {  
        e.printStackTrace();  
    }  
    return new ArrayList<>();  
}
```

# VISTA



# SISTEMA DE GESTION DE BUSES UTPL

```
public static void main(String[] args) {
    Scanner entrada = new Scanner(source: System.in);

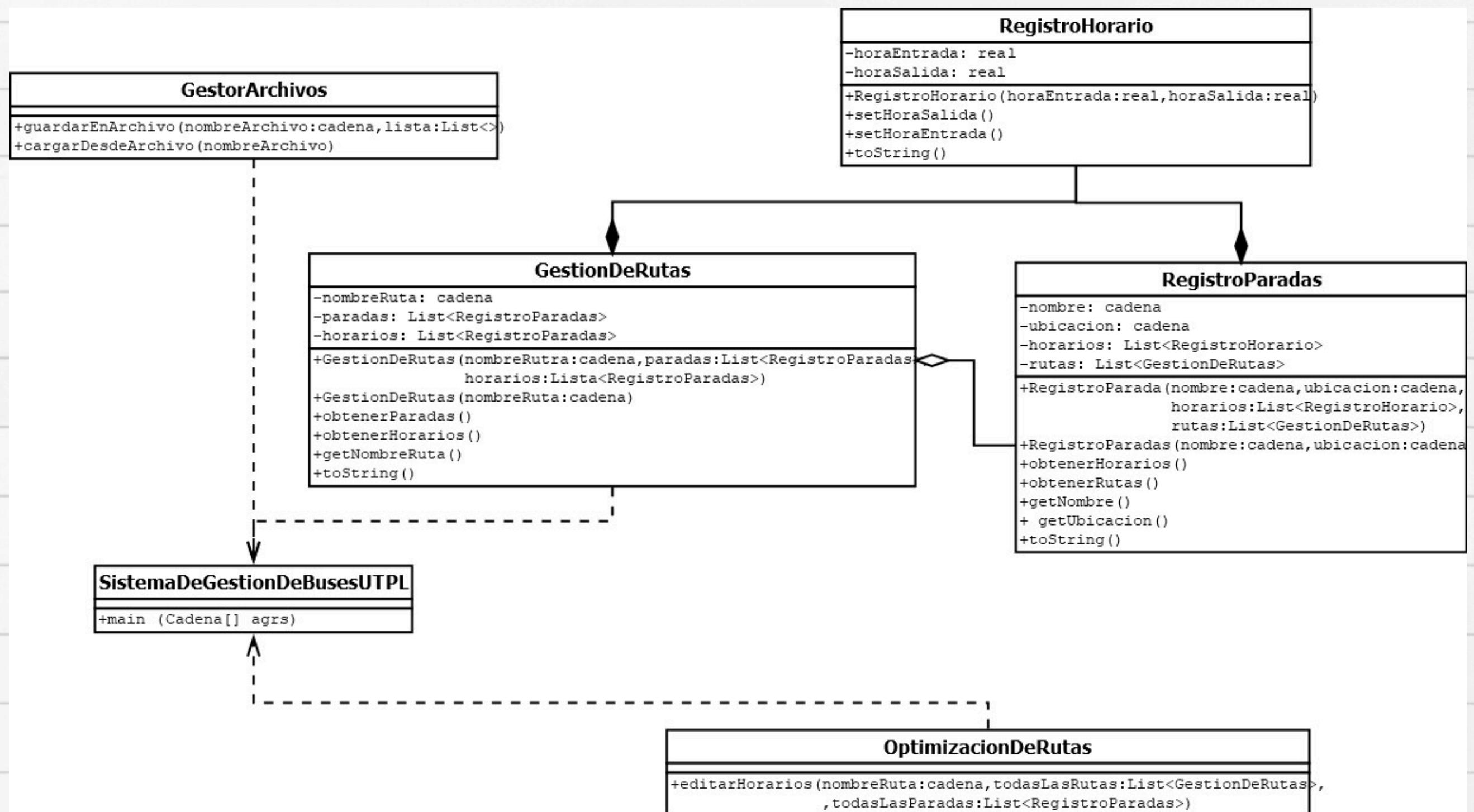
    ArrayList<RegistroParadas> todasLasParadas = new ArrayList<>();
    ArrayList<GestionDeRutas> todasLasRutas = new ArrayList<>();

    String agregarHorarios;
    int n = 0;
    int opcion = 0;

    do {
        System.out.println("""
            + "-----\n"
            + "\t\t\tSistema De Gestión De Buses UTPL\n"
            + "-----\n"
            + "-----");
        System.out.println("Seleccione una opción:");
        System.out.println("1. Registrar una nueva parada");
        System.out.println("2. Registrar una nueva ruta");
        System.out.println("3. Mostrar todas las paradas");
        System.out.println("4. Mostrar todas las rutas");
        System.out.println("5. Consultar horarios de una parada");
        System.out.println("6. Optimizar tiempo");
        System.out.println("7. Salir");
        opcion = entrada.nextInt();
        entrada.nextLine(); // Limpiar el buffer
    }
}
```



# DIAGRAMA UML



# GRACIAS