REQUERIMIENTOS MEMORIA ESCRITA PRÁCTICA 1

Terminal de Buses

Grupo 1

Equipo 1

ASIGNATURA

Programación orientada a objetos

PROFESOR

Jaime Alberto Guzmán Luna

Integrantes:

Santiago Abelardo Salcedo Rodriguez

David Fernando Ceron Quintero

Juan Camilo Moreano Urresty

Luis Mario Cardona Ramirez

Stiven Brandon Rincon

Universidad Nacional de Colombia

Sede Medellín

2024

Índice

2. Descripción general de la solución.

El proyecto se centra en crear una herramienta con la cual se establezca una comunicación entre los usuarios de una terminal de buses y las empresas de dicha terminal de manera que se sistematice los procesos más comunes que involucran a ambos, junto a la sistematización de los procesos internos de las empresas para poder satisfacer las necesidades de sus clientes. Dentro de esto, se puede caracterizar que se tienen las siguientes necesidades:

- Garantizar la correcta compra de tiquetes por parte de los pasajeros y venta por parte de las empresas. Aquí vienen añadidos todos los procesos que esto trae consigo, como la correcta designación de asientos al usuario, la visibilidad de los buses que estén disponibles para la realización de la ruta, consistencia en las transacciones (Tanto del monto recibido como del monto enviado) o el respeto a las capacidades de los buses.
- Implementar un sistema que optimice los procesos respecto a la capacidad y distribución de los asientos. Esto es, que las empresas puedan modificar la disponibilidad de la ocupación de los buses para así generar una mayor satisfacción en los usuarios frente a una demanda alta de viajes, esto obviamente respetando la decisión de los usuarios a generar un cambio en su itinerario o no.
- Garantizar que los reembolsos se realicen adecuadamente. Esto quiere decir, que se respeten los horarios de devolución (Que no existan usuarios que obtengan sus reembolsos en fechas no permitidas según los parámetros), que la devolución de dinero se ejecute formalmente (Que la transacción sea visible y sea financieramente correcta) y que los cambios en la disponibilidad sea cambiada fielmente.
- Otorgar a las empresas la libertad de crear las rutas que vean necesarias ante las necesidades de los usuarios. Esto con la finalidad de que estos posean una mejor calidad de servicios en términos de la frecuencia de viajes entre distintas paradas, y tener variabilidad entre las opciones a la hora de escoger el trayecto total y el número de escalas a hacer.
- Evaluar el desempeño de los choferes considerando diversos factores clave. Analizando las dificultades específicas de las rutas asignadas, las encuestas de satisfacción de los usuarios y otros indicadores relacionados. Con esta información, se hace una evaluación integral del rendimiento de cada chofer, identificando áreas de mejora y reconocer el buen desempeño. Se busca mejorar la calidad del servicio y la experiencia del cliente, fomentando el desarrollo profesional de los choferes.

Y claramente también se busca que el método de interacción con el usuario sea lo más amena posible para no generar dificultades en los procesos que lo involucren.

Requisitos Funcionales

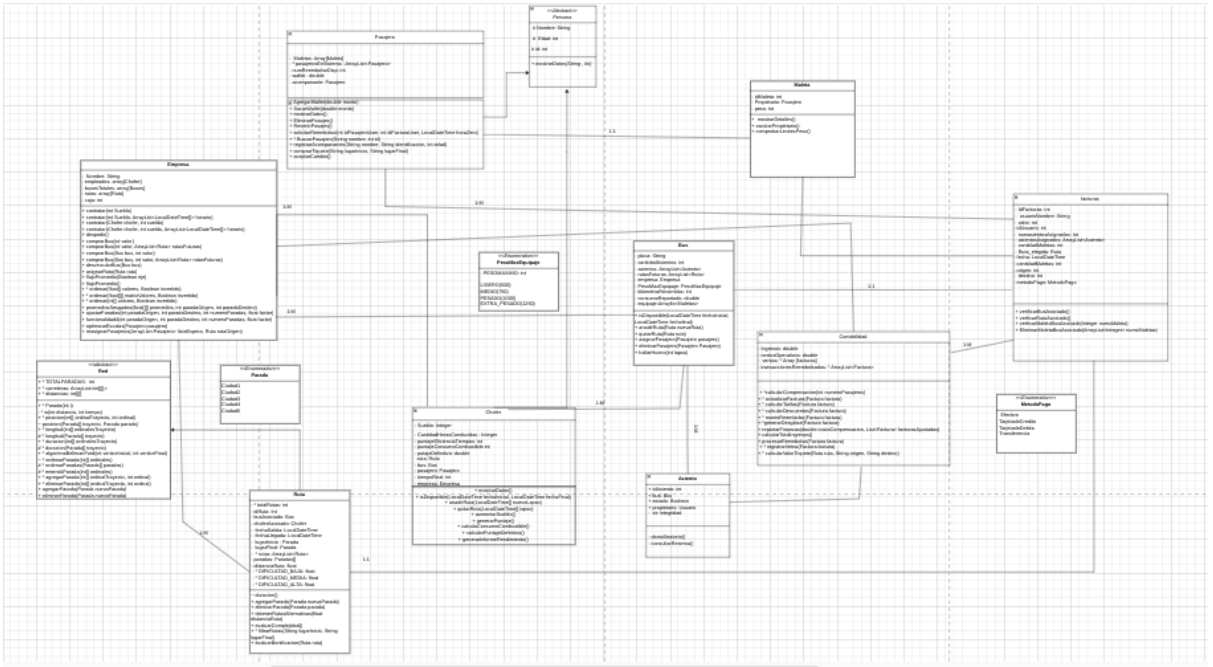
- **Gestión de usuarios:** Registro, autenticación.
- **Gestión de rutas:** Creación, modificación, eliminación de rutas, incluyendo paradas, horarios y distancias.
- **Gestión de buses:** Registro de buses, asignación a rutas.
- **Gestión de asientos:** Asignación de asientos a pasajeros, gestión de disponibilidad.
- **Venta de boletos:** Selección de ruta, elección de asientos, pago “en línea”.
- **Gestión de equipaje:** Registro de equipaje, cálculo de tarifas adicionales, verificación de límites de peso y dimensiones.
- **Reembolso de boletos:** Solicitud de reembolso, verificación de elegibilidad, procesamiento de reembolsos y actualización de registros.

- **Gestión de pagos:** reportes de ingresos.
- **Optimización de rutas:** Algoritmos para encontrar rutas más eficientes en términos de tiempo y distancia.
- **Predicción de demanda:** Análisis de datos históricos para predecir la demanda de pasajeros en diferentes rutas y horarios.

Requisitos No Funcionales

- **Disponibilidad:** El sistema debe estar disponible siempre y cuando se tenga el ejecutable a correr.
- **Tiempo de respuesta:** El sistema debe responder a las solicitudes de los usuarios en un tiempo razonable.
- **Escalabilidad:** El sistema debe ser capaz de manejar un aumento en el número de usuarios y transacciones.
- **Intuitividad:** La interfaz de usuario debe ser fácil de usar y comprender.
- **Consistencia:** La interfaz de usuario debe ser consistente en todo el sistema.
- **Documentación:** El sistema debe estar bien documentado para facilitar su comprensión y mantenimiento.

3. Descripción del diseño estático del sistema en la especificación UML.



Clase Persona (Abstracta)

Descripción: Incorpora los atributos básicos que tienen las personas involucradas (Tanto choferes como pasajeros).

Relación: Únicamente es heredada por las clases Pasajero y Chofer.

Clase Pasajero

Descripción: Es la clase que maneja toda la información del usuario que efectúa la compra-reembolso de tiquetes.

Relación: Hereda el comportamiento de la clase Persona. Puede poseer muchas facturas para que estas lleven el registro tanto de la información del pasajero como de los viajes, asientos y servicios que compró.

Clase Chofer

Descripción: Es la clase que indica todo lo referente a la calidad del viaje y la optimización

Relación: Hereda el comportamiento de la clase Persona. Está asociado a una única empresa y un único Bus, los cuales dictaminan su salario (En términos de sueldo y prestaciones por cantidad de viajes) y los horarios que este debe manejar.

Clase Empresa

Descripción: Hace el control sobre los buses y choferes, en su compra y venta, o contratación y despido, evaluándolos para garantizar los servicios a los pasajeros. También es la encargada de creación y supervisión de las rutas.

Relación: Puede poseer varios empleados (Choferes) y buses, sobre los cuales establece sus horarios y pagos para su correcto funcionamiento. También puede poseer varias rutas, las cuales puede crear, cambiar sus horarios o cambiar su estructura de paradas, dependiendo de las necesidades que posea.

Clase Red (Abstracta)

Descripción: Esta clase posee la información acerca de las paradas existentes y cómo vienen relacionadas entre sí (El grafo que lo determina), y el cómo controlar trayectos con estas paradas para optimizar los parámetros de recorrido.

Relación: Únicamente es heredada por la clase Ruta.

Clase Ruta

Descripción: Manifiesta los recorridos que deben realizar los buses, mostrando las paradas que deben realizar y el horario en el que deben ser recorridas.

Relación: Está asociada a una única empresa, por la cual es creada y maneja a su conveniencia para generar los recorridos necesarios.

Clase Bus

Descripción: Realiza las rutas especificadas por la empresa.
Relación: Está asociado a un único chofer, con el cual realizan los viajes. También posee varios asientos los cuales están vinculados con los pasajeros que va a llevar.

Clase Asiento

Descripción: Determina la posición del pasajero en el bus al que está vinculado.

Relación: Está asociado a un único bus, el cual determina su estado y disponibilidad.

Clase Contabilidad

Descripción: La clase contabilidad en el contexto del sistema de gestión de transporte, actúa como el corazón financiero de la aplicación. Su principal responsabilidad es llevar un registro detallado de todas las transacciones monetarias relacionadas con las operaciones de la empresa, desde la venta de boletos hasta los reembolsos y los gastos operativos.

Relación: Una instancia de contabilidad puede tener muchas facturas, contabilidad utiliza información de muchas rutas, pasajeros y buses

Clase Facturas

Descripción: Recopila la información del cliente, establece un registro entre las compras efectuadas por el cliente y las transacciones que se efectúan a partir de este, para llevar un control de cuentas de la empresa.

Relación: Una factura puede tener uno o más asientos asignados, una factura está asociada a una única ruta, un usuario puede tener muchas facturas y un bus estar asignado a una única factura

Clase Maleta

Descripción: Solo posee la información referente a las proporciones y especificaciones de las maletas del cliente para manejarlo en el equipaje del bus.

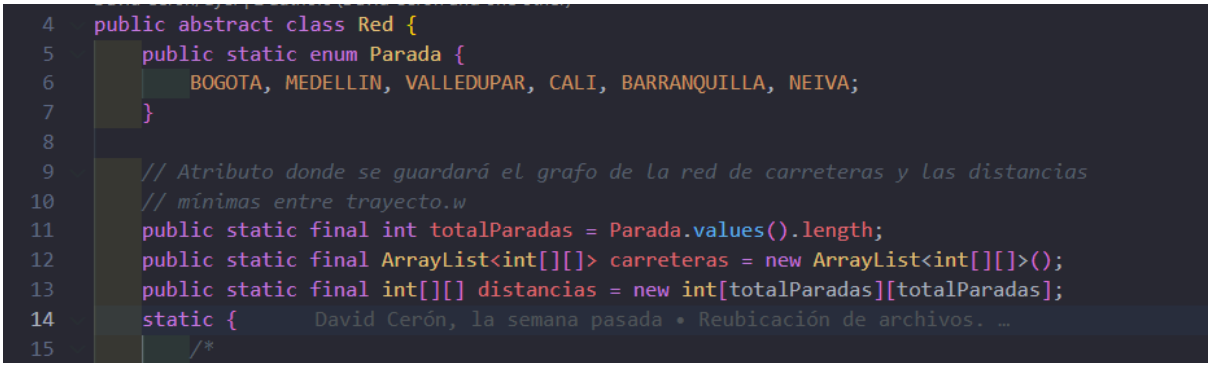
Relación: Una instancia de maleta únicamente va a tener un dueño.

Enlace del UML:
https://drive.google.com/file/d/1HWU2raOnXIBURkaLB0ptP8dFF4y-wHl6/view?usp=drive_link

4. Descripción de la implementación de características de Programación orientada a objetos en el proyecto.

Clase abstracta y Método abstracto.

La clase abstracta Red definida en la **línea 4 clase Red** es una clase abstracta que es padre de Ruta, y posee 2 métodos abstractos agregarParada() y eliminarParada() **en las líneas 618 y 620 respectivamente de la clase Red.** Estas son definidas en su interior por la clase Ruta que serán de vital importancia para la clase Ruta para modificar el conjunto de paradas que posee y su ordenamiento.



```
606         if(estaEnElArray){
607             nuevoTrayecto[numeroParadas - 2] = ordinalTrayecto[numeroParadas - 1];
608         }
609
610         // Preguntando si la parada se encontraba en el array.
611         if (!estaEnElArray && ordinalTrayecto[numeroParadas - 1] != ordinal) {
612             return ordinalTrayecto;
613         }
614
615         return nuevoTrayecto;
616     }
617
618     public abstract void agregarParada(Parada nuevaParada);
619
620     public abstract void eliminarParada(Parada nuevaParada);
621 }
```

Herencia.

Caso 1 : En las clases Persona, Pasajero y Chofer se está utilizando el concepto de herencia pues no necesitan definir de nuevo los atributos de nombre **línea 4 clase Persona**, edad **línea 5 clase Persona** e id **línea 6 clase Persona** en sus propias clases, ya que son heredados de su padre Persona. Esto es muy útil e imprescindible para la reutilización eficiente de código.

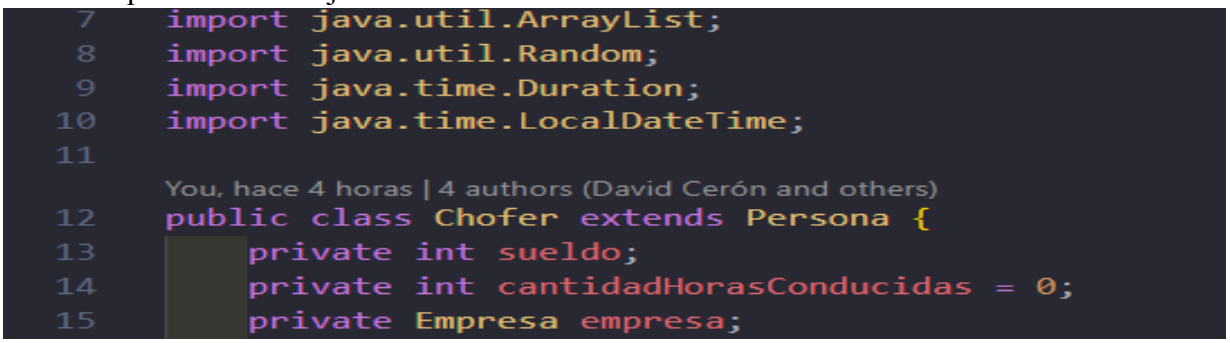
pr-ctica-1-grupo-1-equipo-1 > src > gestorAplicacion > operacion > individuos > Persona.java > Persona

```
David Cerón, hace 2 semanas | 1 author (David Cerón)
1   package gestorAplicacion.operacion.individuos;
2
David Cerón, hace 2 semanas | 1 author (David Cerón)
3   public abstract class Persona{
4       protected String nombre;
5       protected int edad;
6       protected int id;
7       abstract String mostrarDatos();
8   }       David Cerón, hace 2 semanas • Código de la mitad de las clases …
```

Podemos ver que Pasajero si es hijo de la clase Persona **en la línea 15 de la clase Pasajero.**

```
13  import java.time.Duration;
14
You, hace 1 hora | 4 authors (LuismallitoDev and others)
15  public class Pasajero extends Persona {
16
17      // Atributos //
18      private static ArrayList<Pasajero> pasajerosEnSistema = new ArrayList<>();;
19      private ArrayList<Maleta> maletas;
20      private double wallet;
21      private Factura factura;
22      private int numReembolsoDisp;
23      private Pasajero acompanante;
24
```

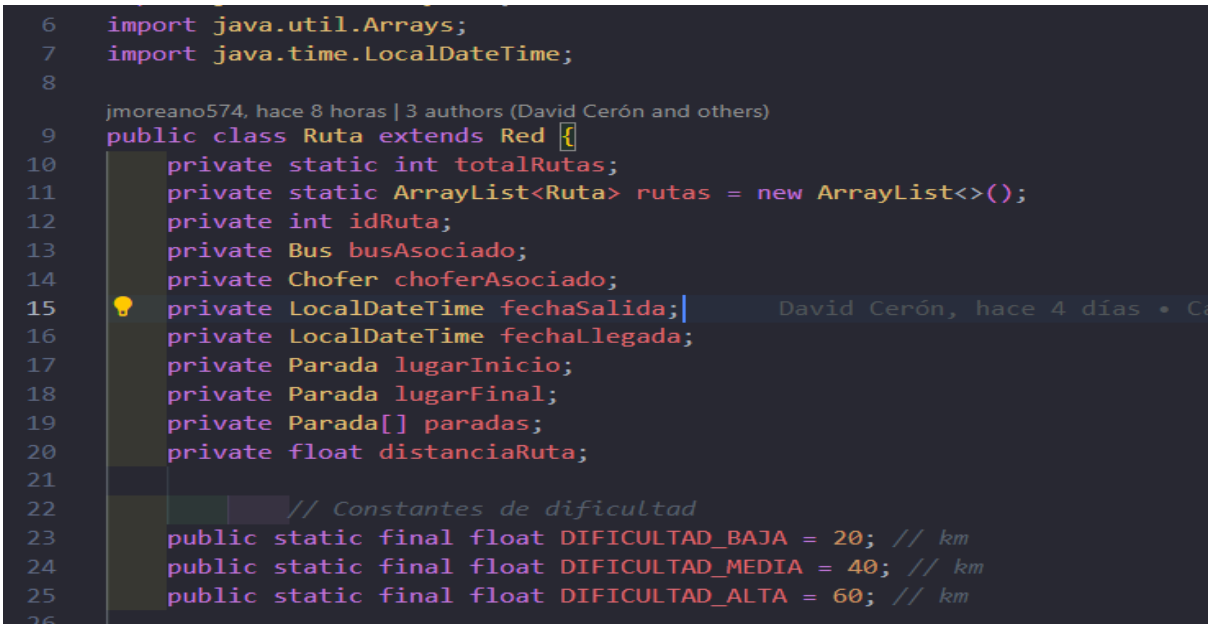
También que Chofer es hijo de la clase Persona **línea 12 de la clase Chofer.**



```
7   import java.util.ArrayList;
8   import java.util.Random;
9   import java.time.Duration;
10  import java.time.LocalDateTime;
11
    You, hace 4 horas | 4 authors (David Cerón and others)
12  public class Chofer extends Persona {
13      private int sueldo;
14      private int cantidadHorasConducidas = 0;
15      private Empresa empresa;
```

Atributo y Método Estático:

En la captura podemos observar el atributo estático como totalRutas y rutas **en la linea 11 de la clase Ruta** este es vital que exista sin tener la necesidad de ser utilizado por una instancia de tipo Ruta creados ya que luego en la ejecucion del codigo serán almacenados para un proximo analisis de distintos metodos tambien un metodo estatico filtrarRutas() es encontrado en **la clase ruta linea 146** y es encargado de evaluar en una lista de rutas las que son convenientes a ser filtradas



```
6   import java.util.Arrays;
7   import java.time.LocalDateTime;
8
    jmoreano574, hace 8 horas | 3 authors (David Cerón and others)
9   public class Ruta extends Red {
10      private static int totalRutas;
11      private static ArrayList<Ruta> rutas = new ArrayList<>();
12      private int idRuta;
13      private Bus busAsociado;
14      private Chofer choferAsociado;
15      private LocalDateTime fechaSalida;      David Cerón, hace 4 días • C
16      private LocalDateTime fechaLlegada;
17      private Parada lugarInicio;
18      private Parada lugarFinal;
19      private Parada[] paradas;
20      private float distanciaRuta;
21
22          // Constantes de dificultad
23      public static final float DIFICULTAD_BAJA = 20; // km
24      public static final float DIFICULTAD_MEDIA = 40; // km
25      public static final float DIFICULTAD_ALTA = 60; // km
```

```java
public static ArrayList<Ruta> filtrarRutas(String lugarInicio, String lugarFinal) {
    ArrayList<Ruta> rutasFiltradas = new ArrayList<>();
    for (Ruta ruta : Ruta.rutas) {
        Parada[] paradas = ruta.getParadas();
        if (paradas[0].equals(Parada.valueOf(lugarInicio))
                && paradas[paradas.length - 1].equals(Parada.valueOf(lugarFinal))) {
            rutasFiltradas.add(ruta);
        }
    }
    return rutasFiltradas == null ? null : rutasFiltradas;
}
```

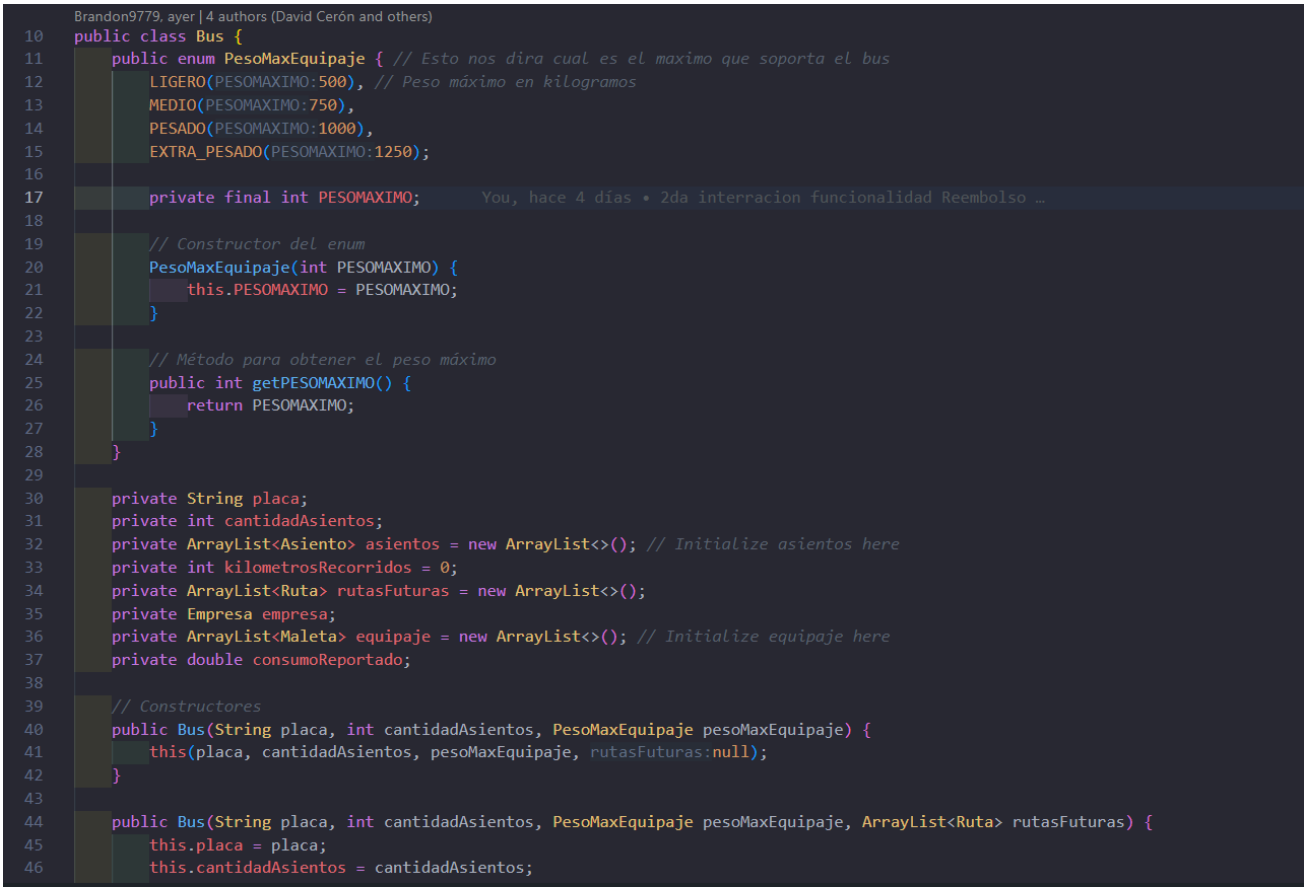
Constante:

En la clase Red Podemos encontrar distintas constantes de todo tipos, enumerados, Arraylists, int **en las línea 5, 11, 12 y 13 de la clase Red** utilizadas para guardar el grafo de la red de carreteras y las distancias entre el mínimo trayecto.

```java
4   public abstract class Red {
5       public static enum Parada {
6           BOGOTA, MEDELLIN, VALLEDUPAR, CALI, BARRANQUILLA, NEIVA;
7       }
8
9       // Atributo donde se guardará el grafo de la red de carreteras y las distancias
10      // mínimas entre trayecto.w
11      public static final int totalParadas = Parada.values().length;
12      public static final ArrayList<int[][]> carreteras = new ArrayList<int[][]>();
13      public static final int[][] distancias = new int[totalParadas][totalParadas];
14      static {       David Cerón, la semana pasada • Reubicación de archivos. …
15          /*
16           * Creación de la red de carreteras:
17           * Esta vendrá representada por un grafo.
18           * Los vértices serán las trayecto (Representadas por su ordinal).
19           * Las aristas serán representadas por una matriz de enteros 2x2, cuyas
20           * entradas:
```

Encapsulamiento:

A lo largo de todo el proyecto se ha utilizado este concepto tan importante, utilizando las distintas restricciones de privacidad según sean necesarias manteniendo lo más seguros y encapsulados posibles los objetos, un buen ejemplo sería en **la clase Bus tenemos varios tipos de modificadores de acceso en la línea 40** dejando el constructor público para poder ser accedido desde donde sea necesario **atributos privados en líneas 30-37** para no ser modificados de manera intrusiva etc.



Sobrecarga de métodos y constructores:

Para la Sobrecarga de métodos tenemos el método flujoPromedio() el cual es sobrecargado en **la clase Empresa en las líneas 420 y 508,** cambiando la firma de este en su parámetro, primero recibiendo un Boolean y luego ninguno.

```java
// Creación de rutas con paradas muy concurridas.
public float[] flujoPromedio(Boolean eje) {// Posiblemente no nos sirva a futuro
    /*
     * Calcula el promedio de personas por viaje desde una parada a otra.
     *
     * Parámetros:
     * - eje: bool,
     * Determina si el promedio se calcula en las filas (False) o columnas (True).
     *
     * Retorna:
     * - promedios: int[# Paradas][# Paradas],
     * Matriz de cantidad de personas que compran este viaje.
     * La entrada i, j representa el flujo desde la ciudad con ordinal i hasta la
     * ciudad con ordinal j.
     */

    // Definición de variables

    // Matriz de personas que realizan el viaje.
    int[][] flujoDemanda = new int[Red.totalParadas][Red.totalParadas];
    // Matriz de viajes totales realizados.
    int[][] flujoOferta = new int[Red.totalParadas][Red.totalParadas];
    // Contador de rutas ya analizadas.
    ArrayList<Ruta> rutasCopia = new ArrayList<Ruta>();

    // Generando la copia del atributo rutas.
    for (Ruta ruta : rutas) {
        rutasCopia.add(ruta);
    }
```

```java
public float[][] flujoPromedio() {
    /*
     * Calcula el promedio de personas por viaje desde una parada a otra.
     *
     * Retorna:
     * - promedios: int[# Paradas][# Paradas],
     * Matriz de cantidad de personas que compran este viaje.
     * La entrada i, j representa el flujo desde la ciudad con ordinal i hasta la
     * ciudad con ordinal j.
     */

    // Definición de variables

    // Matriz de personas que realizan el viaje.
    int[][] flujoDemanda = new int[Red.totalParadas][Red.totalParadas];
    // Matriz de viajes totales realizados.
    int[][] flujoOferta = new int[Red.totalParadas][Red.totalParadas];
    // Contador de rutas ya analizadas.
    ArrayList<Ruta> rutasCopia = new ArrayList<Ruta>();

    // Generando la copia del atributo rutas.
    for (Ruta ruta : rutas) {
        rutasCopia.add(ruta);
    }

    // Contando el número de personas que viajan y viajes realizados (En la
    // empresa).
    ArrayList<Factura> facturas = Contabilidad.getVentas(); // Las facturas llevan registro de esto.
    for (Factura factura : facturas) {
        // Definiendo el origen y destino de la ruta.
        Ruta rutaAsociada = factura.getRutaElegida();
        int origen = factura.getOrigen().ordinal();
        int destino = factura.getDestino().ordinal();
```

En sobrecarga de constructores en la **clase Bus** sus constructores tienen distintas firmas según la cantidad de parámetros que son necesarios, **en la líneas 40 y 44** podemos observar dicha sobrecarga, también en la **clase Chofer en las líneas 29, 35 y 38.** Se estaría utilizando este concepto para una mayor versatilidad a la hora de crear instancias de dichas clases.

```java
    // Constructores
    public Bus(String placa, int cantidadAsientos, PesoMaxEquipaje pesoMaxEquipaje) {
        this(placa, cantidadAsientos, pesoMaxEquipaje, rutasFuturas:null);
    }

    public Bus(String placa, int cantidadAsientos, PesoMaxEquipaje pesoMaxEquipaje, ArrayList<Ruta> rutasFuturas) {
        this.placa = placa;
        this.cantidadAsientos = cantidadAsientos;
        // Removed duplicate line: this.cantidadAsientos = cantidadAsientos;
        if (rutasFuturas != null) {
            this.setRutasFuturas(rutasFuturas);
        }
    }

    // Métodos get-set
    public ArrayList<Maleta> getEquipaje() {
        return equipaje;
    }

    public String getPlaca() {
        return placa;
    }

```

```java
    // Fecha fin}.
    private ArrayList<LocalDateTime[]> horario = new ArrayList<LocalDateTime[]>(); // Cambiar int por fechas

    // Constructores
    public Chofer(String nombre, int edad, int id, int sueldo) {
        this.nombre = nombre;
        this.edad = edad;
        this.id = id;
        this.sueldo = sueldo;
    }
    public Chofer(int sueldo){

    }
    public Chofer(int sueldo, ArrayList<LocalDateTime[]> horario) {
        this.sueldo = sueldo;
        if (horario != null) {
            this.setHorario(horario);
        }
    }

    // Métodos get-set
```

Manejo de referencias this para desambiguar:

En el ejemplo anterior veíamos en la **clase Bus** las siguientes características en sus constructores, **en la línea 41** utilizamos this() para llamar a otro constructor reutilizando así lineas de codigo tambien en las **líneas 45 y 46** utilizamos la referencia this, para así desambiguar los nombres de los parámetros recibidos por el constructor y el atributo como tal al cual se le está asignando el valor, esto es muy útil para tener un código organizado legible y mantenible a lo largo del tiempo (Ver abajo la captura de nuevo) .

```java
// Constructores
public Bus(String placa, int cantidadAsientos, PesoMaxEquipaje pesoMaxEquipaje) {
    this(placa, cantidadAsientos, pesoMaxEquipaje, rutasFuturas:null);
}

public Bus(String placa, int cantidadAsientos, PesoMaxEquipaje pesoMaxEquipaje, ArrayList<Ruta> rutasFuturas) {
    this.placa = placa;
    this.cantidadAsientos = cantidadAsientos;
    // Removed duplicate line: this.cantidadAsientos = cantidadAsientos;
    if (rutasFuturas != null) {
        this.setRutasFuturas(rutasFuturas);
    }
}

// Métodos get-set
public ArrayList<Maleta> getEquipaje() {
    return equipaje;
}

public String getPlaca() {
    return placa;
}
```

Implementación de un caso de enumeración:

Aquí mostramos distintos casos en el cual fue de gran utilidad el utilizar enumeraciones para guardar constantes utilizadas a lo largo de todo el proyecto, como es el caso de las clases **Factura, Red en las líneas 13 y 5 respectivamente**. Esto nos permite gran versatilidad y mejor manejo del código en mantener la cantidad y selección de ciudades en un conjunto deseable, e igualmente aplica esta razón en los métodos de pago.

```java
Brandon9779, ayer | 4 authors (LuismallitoDev and others)
public class Factura implements Serializable {
    // Atributos
    public enum MetodoPago {
        Efectivo, TarjetadeCredito, TarjetadeDebito, Transferencia
    }

    private int idFactura;
    private String usuarioNombre;
```

```java
David Ceron, ayer | 2 authors (David Ceron and one other)
public abstract class Red {
    public static enum Parada {
        BOGOTA, MEDELLIN, VALLEDUPAR, CALI, BARRANQUILLA, NEIVA;
    }

    // Atributo donde se guardará el grafo de la red de carreteras y las distancias
    // mínimas entre trayecto.w
    public static final int totalParadas = Parada.values().length;
```

5. Descripción de cada una de las 5 funcionalidades implementadas.

Funcionalidad 1:Compra de Tiquete con Gestión de Equipaje

Descripción:

El pasajero selecciona una ruta válida y el sistema muestra las empresas disponibles. Tras elegir una empresa, se validan los autobuses disponibles junto con los asientos (asientos vip y normales)y precios. Una vez que el pasajero selecciona un asiento, el sistema solicita detalles sobre el usuario y el equipaje, verificando edad, identificación, peso y límite permitido. Si el equipaje supera el límite, el sistema notifica al usuario e intenta asignar otro autobús compatible.

Interacción 1: Selección de Ruta y Empresa

Clases involucradas: Pasajero, Ruta, Empresa, Bus.

Descripción:

1. El Pasajero selecciona un lugar de origen y destino.
2. El sistema consulta las Empresas que ofrecen rutas entre estos lugares, considerando:

- Tipo de asiento (estándar o VIP).
- Cantidad de escalas.
- El Bus encargado de cada ruta.

3. La clase Ruta devuelve una lista de opciones disponibles.
4. El Pasajero elige la Ruta que más se ajuste a sus necesidades.

Interacción 2: Selección de Asiento y Registro del Pasajero

1. **Clases involucradas: Pasajero, Asiento, Bus.**
2. **Descripción:**
    1. Una vez elegida la Ruta, el sistema muestra los Asientos disponibles en el Bus asignado.
    2. El Pasajero selecciona un asiento, ya sea estándar o VIP.
    - Si el Pasajero es menor de edad:
        - Se solicita ingresar los datos de un acompañante mayor de edad.
        - El sistema registra tanto al Pasajero como al acompañante.
    - Los datos registrados incluyen nombre, identificación y edad.

Interacción 3: Gestión de Equipaje y Generación de Factura

1. **Clases involucradas: Pasajero, Bus, Factura.**
2. **Descripción:**
    1. El Pasajero indica la cantidad de equipaje que llevará.
    - El sistema:
        - Calcula el peso total del equipaje.
        - Verifica si el peso excede la capacidad máxima del Bus.
    - Si se supera el límite:
        - Ofrece cambiar de Ruta a una que cumpla con las especificaciones de peso.

        - O bien, eliminar parte del equipaje.
    - Una vez confirmado, se genera una Factura con:
        - Costo del pasaje (según tipo de asiento).
        - Costo adicional por el equipaje.(teniendo en cuenta peso)

Funcionalidad 2: Reasignación de asientos por alta demanda.

Descripción:

Se estima los pasajeros presentes en cada una de las paradas. Si el bus encargado de la ruta está cerca a su límite de capacidad, se notifica a los pasajeros si desearían hacer un cambio de horario con compensación. Se implementa una lista con los pasajeros que desean hacer el cambio, y se escogen al azar {Número} de pasajeros para hacer el cambio.

Se buscan rutas que tengan la misma parada y destino, y se buscan buses con mayor disponibilidad. Se procede a hacer los cambios de asientos con aquellos buses. Con esto, se cambian las facturas de los pasajeros.

Se hace un cómputo de la financiación hecha con los bonos y se notifica el reporte de contabilidad a la empresa (imprimir términos y condiciones)

El sistema optimiza las escalas en tiempo real durante la ejecución del programa. Permite al pasajero decidir si ajustar su itinerario en función de nuevas paradas intermedias y actualiza automáticamente la información de costos y capacidades.

Interacción 1: Monitoreo de Capacidad y Notificación a los Pasajeros

1. **Clases involucradas:** Ruta, Bus, Pasajero, Contabilidad.
2. **Descripción:**
    - Se recorre la lista de **Rutas** activas.
    - Para cada **Ruta**, se consulta el número de **Pasajeros** presentes en cada parada.
    - Si el **Bus** asignado a una **Ruta** se encuentra cerca de su capacidad máxima:
        - Se notifica a los **Pasajeros** ofreciéndoles la opción de cambiar de horario con compensación.
        - Los **Pasajeros** que acepten se agregan a una lista de espera para el cambio.
    - Se registra un cálculo preliminar de los posibles costos de compensación en la clase Contabilidad.

Interacción 2: Reasignación de Pasajeros a Nuevas Rutas y Buses

1. **Clases involucradas:** Ruta, Bus, Pasajero, Asiento, Factura.
2. **Descripción:**
    - El sistema busca **Rutas** alternativas que compartan la misma parada y destino.

- Dentro de estas **Rutas**, se identifican **Buses** con mayor disponibilidad de asientos.
- Se selecciona un número de **Pasajeros** de la lista de espera para reasignarlos.
  - Se asignan nuevos **Asientos** en los **Buses** con mayor capacidad.
  - Las **Facturas** de los **Pasajeros** afectados se actualizan para reflejar el cambio de horario y la compensación otorgada.
- Los asientos liberados en el bus original se actualizan en la clase Bus.

Interacción 3: Ajuste Financiero y Reporte a la Empresa

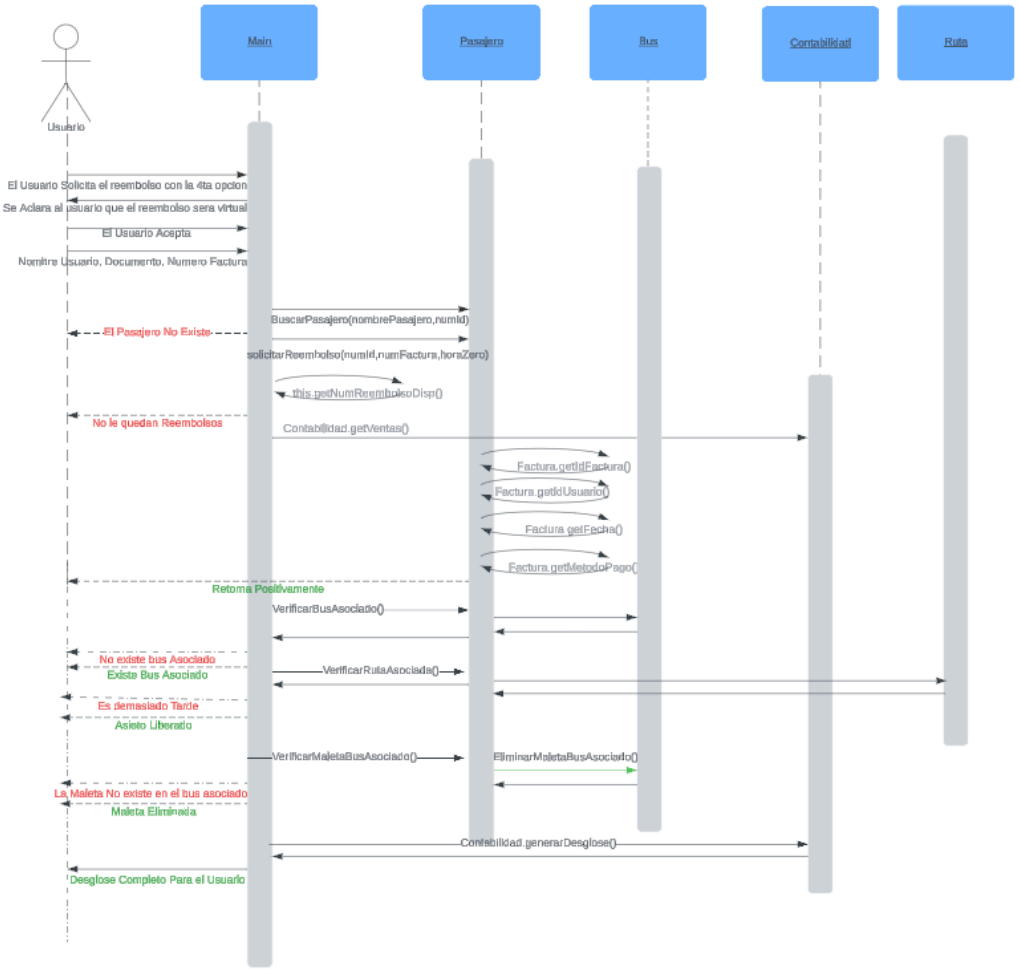
1. **Clases involucradas:** Contabilidad, Factura, Empresa.
2. **Descripción:**
   - Se realiza un cómputo de los costos asociados a las compensaciones ofrecidas (bonos) y los cambios realizados.
   - Se reajustan los costes operativos basados en los asientos liberados y la capacidad utilizada en los nuevos **Buses**.
   - Se genera un reporte detallado en la clase Contabilidad que incluye
     - El costo total de las compensaciones.
     - Los ajustes realizados a las **Facturas**.
     - El impacto en los costos operativos.
   - El reporte financiero se notifica a la **Empresa**.

Funcionalidad 3: Reembolso de Tiquete

El usuario puede solicitar un reembolso si cumple con las siguientes condiciones: existencia de una factura asociada, identificación del usuario y que la solicitud se realice con más de 24 horas de antelación. Si se cumplen estas condiciones, el sistema invalida la factura, actualiza el estado del asiento y la capacidad del autobús, registra los cambios en contabilidad y transfiere el dinero al monedero virtual del usuario.

Link del diagrama de interacción:
https://lucid.app/lucidchart/2873788b-348e-4c2c-92fb-6b02cc3811e7/edit?viewport_loc=

638%2C1584%2C1776%2C815%2C0_0&invitationId=inv_4cab04cd-4e17-4c71-ac9c-f7dfc18459c2



Interacción 1: Validación de Reembolso y Notificación Inicial

1. **Clases involucradas:** Pasajero, Factura, Contabilidad, Asiento, Ruta.
2. **Descripción:**
    - El **Pasajero** inicia la solicitud de reembolso proporcionando su **Nombre, ID** y el número de **Factura**.
    - El sistema valida las condiciones necesarias:
        - La existencia de la **Factura** en el sistema.
        - Que el **ID** del **Pasajero** coincida con el registrado en la **Factura**.
        - Que el tiempo transcurrido desde la compra sea mayor a 24 horas.
        - Verificar si el reembolso solicitado está dentro de las políticas de la empresa, como:
            - Un límite de reembolsos permitidos por pasajero en un periodo determinado. 2 por año
            - Condiciones específicas para el método de pago utilizado.
    - Si alguna condición no se cumple:
        - Se notifica automáticamente al **Pasajero**, indicando el motivo del rechazo de forma clara y detallada y se regresa al menú principal.

- Si las condiciones son válidas:
    - Se notifica al usuario que la solicitud está en proceso.
    - La **Ruta** asociada se consulta para verificar cómo la liberación del asiento afectará las futuras asignaciones.

Interacción 2: Gestión del Reembolso y Actualización del Sistema

1. **Clases involucradas:** Factura, Asiento, Bus, Contabilidad.
2. **Descripción:**
    - El **Asiento** asociado se marca como disponible, quedando libre para futuros pasajeros.
    - La capacidad de equipaje en el **Bus** se actualiza, liberando el espacio previamente asignado al **Pasajero**.
    - La **Factura** se invalida para evitar su reutilización en el futuro.
    - **Impacto en otras clases :**
        - Si el equipaje liberado afecta la capacidad del **Bus**, se recalcula la distribución del peso para maximizar la eficiencia operativa.
    - **Manejo de Errores:**
        - Si ocurre un fallo al invalidar la factura, actualizar el estado del asiento o liberar la capacidad del equipaje:
            - El sistema revierte los cambios realizados hasta ese momento.
    - El sistema registra los cambios en la clase Contabilidad, actualizando:
        - La lista de transacciones realizadas.
        - El balance general del sistema.

Interacción 3: Transferencia de Fondos y Notificación Final

1. **Clases involucradas:** Pasajero, Contabilidad, Factura, Empresa.
2. **Descripción:**
    - El monto correspondiente al reembolso se transfiere al monedero virtual del **Pasajero**, asegurando que la transacción sea segura.
    - **Cálculo financiero adicional:**
        - Si el reembolso incluye tarifas administrativas o descuentos, estos se calculan y detallan en el desglose de la transacción.
        - El sistema genera un informe que evalúa el impacto financiero del reembolso en la operación general.
        - Impacto financiero en los balances.
    - El sistema notifica automáticamente al **Pasajero** que el proceso de reembolso ha sido completado, proporcionando un Desglose digital con los detalles de la transacción.

Funcionalidad 4 : Análisis de ruta concurrida una nueva ruta a partir de dos paradas muy concurridas

Analizar y crear una nueva ruta utilizando dos paradas que sean muy solicitadas por los usuarios. La nueva ruta debe ser eficiente en términos de distancia, agregando paradas adicionales solo cuando sea necesario, y sin incrementar excesivamente la distancia original o el número de paradas.

Interacción 1: Identificación de Paradas Más Concurridas

1. **Clases involucradas:** Empresa, Red, Factura.
2. **Descripción:**
    - Se analizan los registros históricos de las rutas realizadas, mediante la consulta de las facturas, evaluando el uso de cada parada en términos de número de pasajeros, frecuencia de trayectos y capacidad de buses asociados.
    - El sistema utiliza los datos de la clase Factura (registros de viajes) para calcular el nivel de concurrencia de cada parada.
    - Se ordenan las paradas por nivel de concurrencia y se seleccionan las dos con mayor frecuencia de uso.
    - **Resultado:** El sistema define las dos paradas más concurridas como el inicio y el destino de la nueva ruta (Si no son especificados).

Interacción 2: Análisis de Rutas Existentes y Selección de la Ruta Base

1. **Clases involucradas:** Red.
2. **Descripción:**
    - El sistema analiza las rutas existentes que conectan las dos paradas seleccionadas.
    - Se utiliza el algoritmo de Bellman-Ford para identificar el trayecto más directo entre ambas paradas, considerando una función de peso que involucre:
        - La distancia entre las paradas.
        - El tiempo estimado de viaje.
    - **Impacto en otras clases:**
        - La clase Red se ve involucrada en el cálculo de optimización de las paradas intermedias a considerar y la agregación o eliminación de estas respetando el orden y la minimización de recorrido.
    - **Criterios adicionales:**
        - La distancia total de la nueva ruta no debe superar en más de un porcentaje dado la distancia directa entre las dos paradas.
        - Si no existe una ruta directa, se analiza la posibilidad de agregar paradas intermedias estratégicas.

- **Resultado:** Se define una ruta base eficiente que conecta las dos paradas seleccionadas, minimizando la distancia y optimizando el tiempo de viaje.

Interacción 3: Creación de la Nueva Ruta y Ajustes Operativos

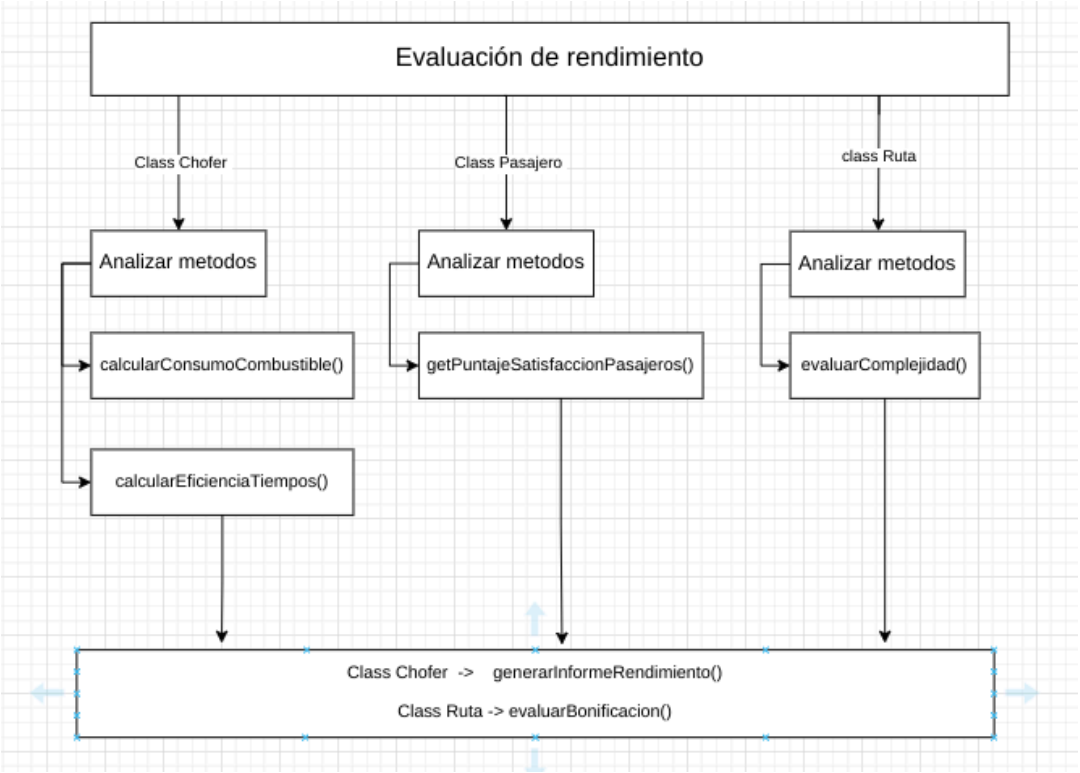
1. **Clases involucradas:** Ruta, Empresa, Bus, Chofer.
2. **Descripción:**
    - Con la ruta base definida, el sistema crea la nueva ruta, asignándole:
        - A partir de lo definido en la interacción 2, se calcula el tiempo total en realizar el trayecto haciendo el cómputo entre ciudades contiguas no necesariamente involucradas en la ruta original.
        - Mediante una búsqueda de horarios entre los buses existentes en la empresa, un horario que se adecúe con la duración total de la ruta, respetando los tiempos de espera entre viajes para los buses.
        - Igualmente se busca un chofer que sea capaz de satisfacer este horario, y en caso de no existir, se deberá contratar uno nuevo y generar el reporte financiero a la empresa para cubrirlo.
        - Un horario inicial que optimice la cobertura de las horas según los literales anteriores.
    - **Reporte avanzado:**
        - Se genera un informe detallado para la clase Empresa, que incluye:
            - Proyección de ingresos y costos.
            - Impacto esperado en la ocupación de otras rutas.
            - Recomendaciones sobre ajustes de tarifas si es necesario.
    - Finalmente, se activa la nueva ruta en el sistema.

Funcionalidad 5: Gestión de Conductores y Optimización de Rendimiento

Objetivo:

Desarrollar una funcionalidad que evalúe el rendimiento de los conductores, optimice su asignación a rutas según su desempeño y disponibilidad, y realice un análisis detallado para identificar oportunidades de mejora en la operación. La funcionalidad debe considerar factores como eficiencia de conducción, cumplimiento de horarios, consumo de combustible y satisfacción de los pasajeros.

Enlace Diagrama de interacción:
https://drive.google.com/file/d/1-sY7cexjJf7DiT5uApexpHn4gGNVOH2C/view?usp=sharing



Interacción 1: Evaluación de Rendimiento de los Conductores

1. **Clases involucradas:** Chofer, Ruta, Bus, Contabilidad, Pasajero.
2. **Descripción:**
    - **Recopilación de datos:** El sistema recopila información histórica del desempeño de los conductores:
        - **Eficiencia en tiempos:** Comparación entre el tiempo estimado de viaje y el tiempo real tomado por el conductor.
        - **Consumo de combustible:** Relación entre la distancia recorrida y el consumo reportado por el bus asignado.
        - **Satisfacción de pasajeros:** Análisis de encuestas realizadas a los pasajeros de las rutas operadas por cada conductor.(debatible)(idea david volverlo aleatorio :D)(simulando las calificaciones)
    - **Evaluación de rendimiento:**
        - Se calcula un puntaje para cada conductor basado en los criterios mencionados, ponderando cada aspecto según su importancia (por ejemplo, eficiencia en tiempos, consumo de combustible, satisfacción de pasajeros).
    - **Criterios adicionales:**
        - Conductores con desempeño por debajo de un umbral establecido (ejm: puntaje < 70%) son marcados para una revisión más detallada.
    - **Resultado:** Se genera un informe de rendimiento para cada conductor, destacando fortalezas y áreas de mejora.

Interacción 2: Asignación Óptima de Conductores a Rutas

1. **Clases involucradas:** Chofer, Ruta, Bus, Empresa.
2. **Descripción:**
    - **Análisis de requisitos de la ruta:**
        - Se evalúan las características de cada ruta, como:
            - Longitud del trayecto.
            - Tiempo estimado de viaje.
            - Complejidad (rutas con más tráfico o condiciones difíciles).
    - **Asignación basada en rendimiento:**
        - Conductores con mejor puntaje en eficiencia de tiempo son asignados a rutas críticas que exigen cumplimiento estricto de horarios.
        - Conductores con mejor puntaje en consumo de combustible son asignados a rutas largas para maximizar la eficiencia.
        - Conductores con mejor satisfacción de pasajeros son priorizados en rutas VIP.
    - **Impacto en otras clases:**
        - La clase Empresa es notificada con un plan de asignación que optimiza la operación.
        - La clase Bus recibe actualizaciones sobre los conductores asignados a cada unidad.
    - **Validaciones adicionales:**
        - Se verifica que los conductores no excedan el límite de horas laborales permitidas.
        - Se actualiza la disponibilidad de los conductores para reflejar su asignación.
    - **Resultado:** El sistema asigna automáticamente los conductores a las rutas, optimizando el rendimiento general.

Interacción 3: Análisis Operativo y Recomendaciones

1. **Clases involucradas:** Chofer, Contabilidad, Empresa.
2. **Descripción:**
    - **Cálculo financiero adicional:**
    -
        - Se analiza el impacto financiero del rendimiento de los conductores:
            - Costos asociados a consumo de combustible.
            - Ingresos generados por satisfacción de pasajeros en rutas VIP.
            - Costos operativos por incumplimientos de horarios.
        - Se generan métricas clave como:

        - Costo promedio por kilómetro recorrido.
        - Índice de satisfacción del servicio según conductor.
    - **Reporte avanzado:**
        - Se genera un reporte detallado para la clase Empresa, que incluye:
            - Conductores con mejor y peor desempeño.
            - Rutas que requieren ajustes operativos debido a problemas recurrentes.
            - Recomendaciones para capacitaciones específicas (por ejemplo, técnicas de manejo eficiente o servicio al cliente).
    - **Automatización de decisiones:**
        - Conductores con bajo rendimiento son asignados automáticamente a programas de capacitación.
        - Conductores con alto rendimiento reciben incentivos económicos o reconocimiento en el sistema.
    - **Resultado:** La empresa recibe un análisis completo que permite tomar decisiones estratégicas para mejorar la operación

6. Manual de usuario.

Introducción

Este manual tiene como objetivo guiar al usuario en el uso del sistema de gestión de terminal de buses. La aplicación permite realizar diversas funciones como la compra de tiquetes, gestión de equipaje, reasignación de asientos, solicitudes de reembolsos, análisis de rutas concurridas y gestión de conductores. Este documento describe cada funcionalidad y proporciona instrucciones detalladas para interactuar con el sistema.

Requisitos del Sistema

- Archivo ejecutable principal: terminal_buses.jar.

Estructura del Sistema

Menú Principal

Al iniciar el programa, el usuario verá un menú con las siguientes opciones:

1. **Consultar Rutas**
2. **Comprar Pasaje**
3. **Ver Pasajes Comprados**
4. **Solicitar Reembolso**
5. **Panel Empresa**
6. **Salir.**

El usuario debe ingresar el número correspondiente a la opción deseada para navegar por el sistema.

Funciones del Programa.

1. **Comprar Pasaje**

Descripción

Permite al pasajero seleccionar una ruta, y asiento, además de gestionar su equipaje. El sistema valida las condiciones de peso del equipaje y genera una factura detallada.

Pasos para realizar una compra:

1. Seleccione la opción **1** en el menú principal.
2. Ingrese sus Nombres (Solo 1 en caso de tener 1 solo).
3. Ingrese sus Apellidos .
4. Ingrese su número de documento.
5. Ingrese su edad.

Nota: Estos datos son necesarios para poder ser registrados en nuestro sistema.

Si la edad de la persona es menor a 18 entonces leer esto, Sino ignorar:

- Usted deberá ingresar los datos de un acompañante de manera obligatoria,
- El sistema le pedira que ingrese los nombres del acompañante (en caso de tener solo uno escriba uno solo).
- Ingrese los apellidos del acompañante.
- Ingrese la edad del acompañante.
- Se registra el acompañante del menor de edad.

6. Ingrese el nombre del lugar de **origen** de acuerdo a las ciudades ofrecidas
7. Ingrese el nombre del lugar de destino entre las opciones.

   Si no Hay rutas disponibles entre las ciudades escogidas se mostrará un mensaje de error

8. El Sistema mostrará las rutas disponibles, seleccione una entre las opciones mostradas.

   En Caso de no haber seleccionado una ruta de las opciones se mostrará invalida

9. Elija el asiento dentro de las opciones disponibles que se le serán mostradas en el bus, tenga en cuenta que las X son los asientos del bus que se encuentran ocupados

   En caso de seleccionar un asiento ocupado el sistema lo notificará

10. Proporcione la información requerida: nombre, identificación, edad.
    - Si es menor de edad, ingrese los datos de un acompañante.
11. Indique el peso total de su equipaje.
    - Si el peso excede el límite, el sistema sugerirá alternativas (cambio de bus o reducción de equipaje).
12. Confirme la compra y revise la factura generada.

---

2. Reasignación de Asientos por Alta Demanda

Descripción

El sistema notifica a los pasajeros si un bus está cerca de su límite de capacidad. Los pasajeros pueden aceptar cambiar de horario a cambio de una compensación.

Pasos:

1. Seleccione la opción **2** en el menú principal.
2. El sistema monitoreará la capacidad de los buses y notificará a los pasajeros afectados.
3. Si desea participar, confirme su aceptación para ser incluido en la lista de cambios.
4. El sistema reasignará su asiento y le notificará los detalles del nuevo horario y compensación.

---

3. Ver Pasajes Comprados

---

4. Solicitud de Reembolso de Tiquete

Descripción

Permite al usuario solicitar un reembolso si cumple con las condiciones establecidas.

Pasos:

1. Seleccione la opción **4** en el menú principal.
2. El sistema asegurará que tenga presente que el dinero reembolsado será en su monedero virtual. Seleccione la opción 1 para continuar
3. Proporcione los siguientes datos:
    - Su nombre con exactitud a como se encuentra en la factura
    - Su número de documento
    - Número de factura.
4. El sistema validará las condiciones necesarias : En Caso de no ser validada alguna de las condiciones el sistema expresara el porqué y se le devolverá al menú
    - El Pasajero exista
    - El Bus asociado a la factura exista
    - Existencia de la factura.
    - Coincidencia de la identificación.
    - Solicitud realizada con más de 24 horas de antelación.
5. Ingrese el número de maletas registradas del Pasajero en la factura.
6. Se le pedirá que ingrese el identificador de la maleta repetidas veces en caso de ser varias maletas, esto con el fin de poder eliminarlas del bus
7. Recibirá una notificación con los detalles de la transacción, con el monto reembolsado entre otros datos importantes para usted. .

---

5. Panel de Empresa.

Descripción

En este apartado encontrarás una sección enfocada en los usuarios más administrativos y de empresa, en el cual podrás analizar 2 paradas muy frecuentes y encontrar otra nueva a partir de ella.

Pasos:

1. Seleccione la opción **5** en el menú principal.

Se le pedirán un conjunto de especificaciones para hacer el análisis de la ruta a crear.

2. Seleccione a qué empresa disponible desea hacer el análisis.

En caso de no haber seleccionado una opción disponible el sistema no lo permitirá avanzar hasta que escoja otra que sí esté disponible.

3. El sistema preguntará si desea escoger las paradas de origen o destino de la nueva ruta a analizar.

   En caso de querer escoger las paradas:

   - Se le mostrarán las ciudades disponibles a escoger.
   - Seleccione la ciudad de origen, en caso de no querer seleccionar, oprima -1.
   - Seleccione la ciudad de destino, en caso de no querer seleccionar, oprima -1.

En caso de haber seleccionado una ciudad inexistente o seleccionar la misma 2 veces deberás volver a ingresar ambas ciudades.

Si decidió no escoger alguna de las ciudades o ninguna, el sistema procederá a buscar aquellas que tengan la mayor demanda. En caso de no seleccionar la ciudad origen, pero sí la de destino, se buscará aquella ciudad que los usuarios suelen escoger con mayor frecuencia para subirse en ella y bajarse en la ciudad de destino especificada. En caso de no seleccionar la ciudad destino, pero sí la de origen, se buscará aquella ciudad que los usuarios suelen escoger con mayor frecuencia para bajarse en ella y subirse en la ciudad de origen especificada. En caso de no seleccionar ninguna, se buscarán las ciudades cuyo trayecto de origen y destino sean las más demandadas.

Con esto, el programa calculará con las ciudades establecidas, la ruta que minimiza el factor asociado a cada viaje entre ciudades (Un promedio ponderado entre tiempo y distancia). Luego se le mostrará el resultado sugerido para la nueva ruta. Se le solicitarán nuevos datos:

4. Defina las paradas totales que desea que tenga el recorrido. Si no desea que se modifique este parámetro, inserte la cantidad de paradas que tiene la ruta establecida.
5. Defina el factor de crecimiento máximo que desea tenga la ruta (Es decir, el porcentaje que no desea que aumente en proporción respecto a la ruta sugerida).

En caso de haber seleccionado una cantidad de paradas inferior a 2, tendrá que volver a repetir los puntos 4 y 5.

Una vez seleccionados estos parámetros, el sistema le mostrará cómo se añaden o eliminan paradas dependiendo de los criterios establecidos, los cuales también serán mostrados.

6. Revise los detalles de la nueva ruta, incluyendo horarios y buses asignados.
7. Disfrute su nueva ruta.

---

5. Gestión de Conductores y Optimización de Rendimiento

Descripción

Evalúa el desempeño de los conductores y optimiza su asignación a rutas.

Pasos:

1. Seleccione la opción **5** en el menú principal.
2. El sistema evaluará a los conductores según:
    - Eficiencia en tiempos.
    - Consumo de combustible.
    - Satisfacción de los pasajeros.
3. Revise el informe de rendimiento generado.
4. Los conductores serán reasignados automáticamente según su puntuación, y se enviarán recomendaciones para capacitaciones o incentivos.

---

6. Salir

Descripción: En caso de querer cerrar el programa Seleccione la opción **6**

Resolución de Problemas Comunes

1. **El sistema no inicia.**
    - Asegúrese de que el archivo terminal_buses.jar esté en la ubicación correcta.
2. **No se encuentra la ruta deseada.**
    - Verifique que ingresó correctamente el lugar de origen y destino.
    - Considere ajustar su itinerario con una nueva parada intermedia.
3. **El sistema rechaza la solicitud de reembolso.**
    - Asegúrese de cumplir con las condiciones (factura válida, tiempo de antelación).