

MEMORIA ESCRITA

Sistema gestión de transporte

Julian Ricardo Salazar Duarte

Michael Garcia Quincos

Grupo 1 - Equipo 1

Práctica 1

Profesor:

Jaime Alberto Guzmán Luna

25 de mayo de 2023



Universidad Nacional de Colombia

Descripción general de la solución.

La aplicación se basa en un sistema de gestión para una empresa de transporte de productos entre ciudades de diferentes países (Colombia, Ecuador y Panamá).

El análisis se basó en diseñar un sistema para gestionar la solicitud de envío de diferentes tipos de productos de una ciudad a otra en diferentes países, además de la posibilidad de realizar seguimiento de este. Para representar todos los objetos necesarios para llevar a cabo dicha aplicación, se hizo la división por medio de paquetes; el paquete camion para representar los vehículos que iban a ser usados para transportar la mercancía solicitada; el paquete entidades para almacenar y representar toda la información referente a el administrador, empleados y usuario; el paquete país para representar diferentes países y las ciudades y el paquete producto para gestionar todo lo referente a una solicitud para realizar un envío, productos, pedido y factura.

Diagrama UML.

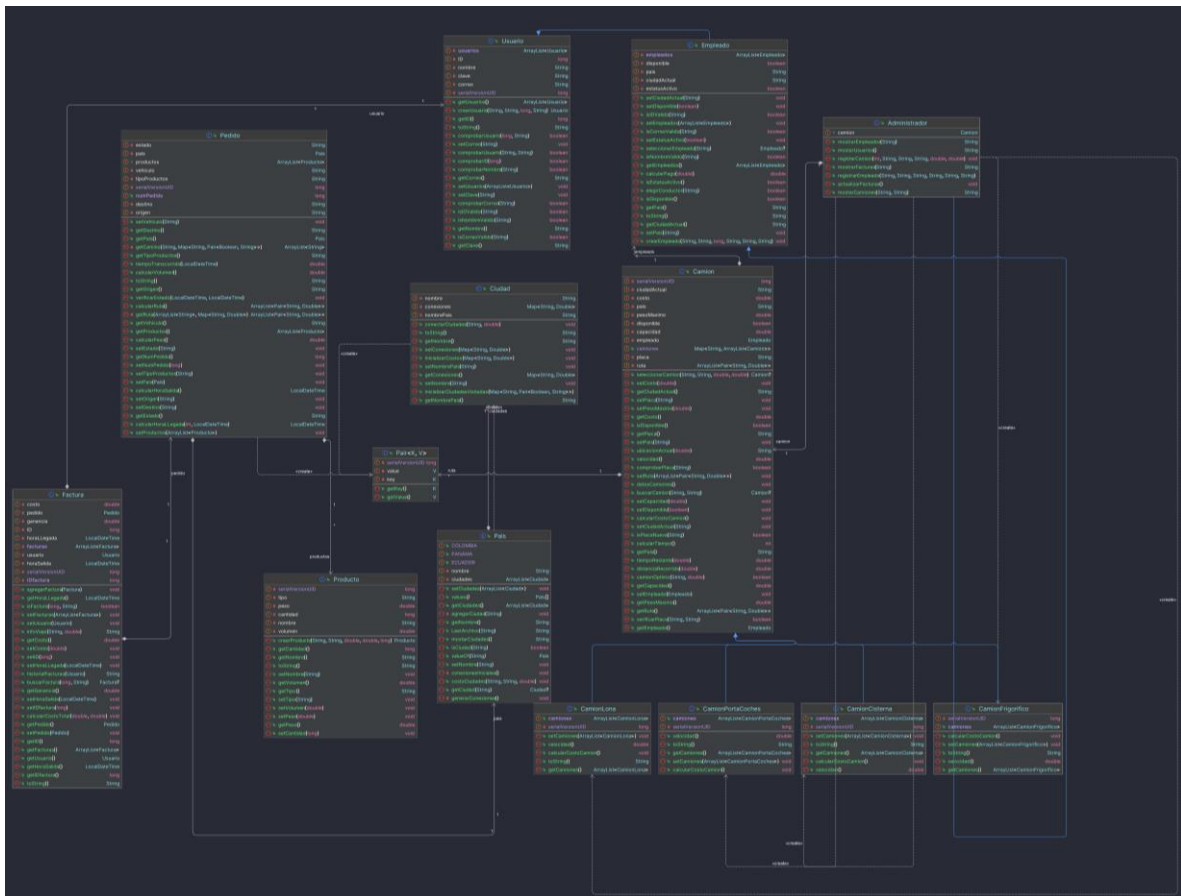
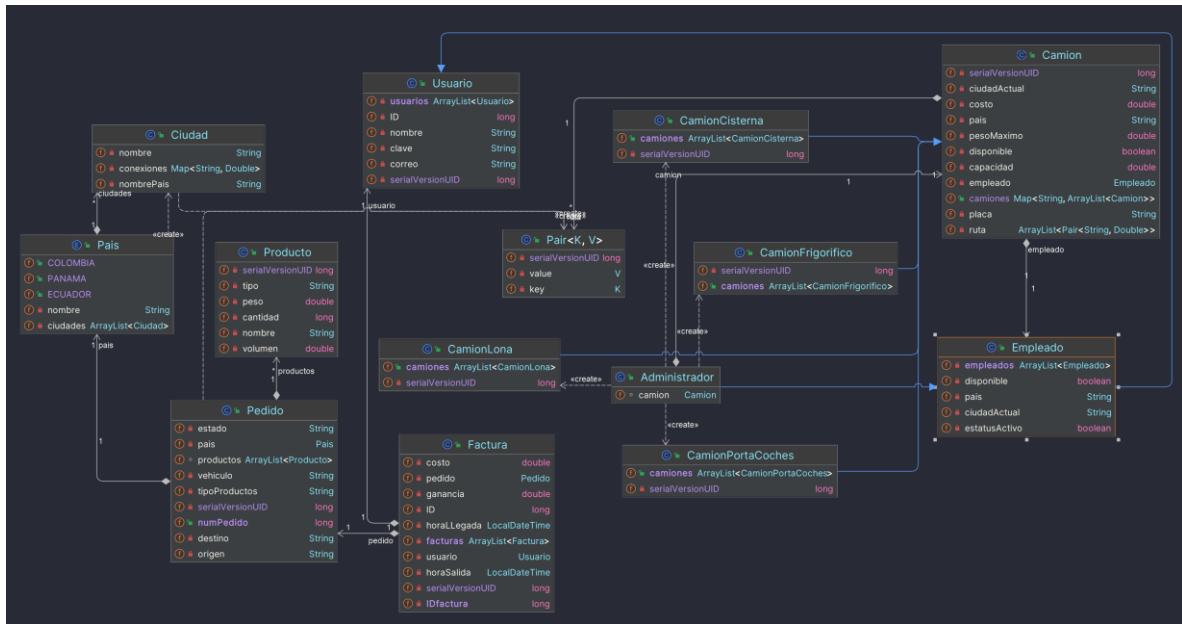


Diagrama UML completo



Relación entre clases

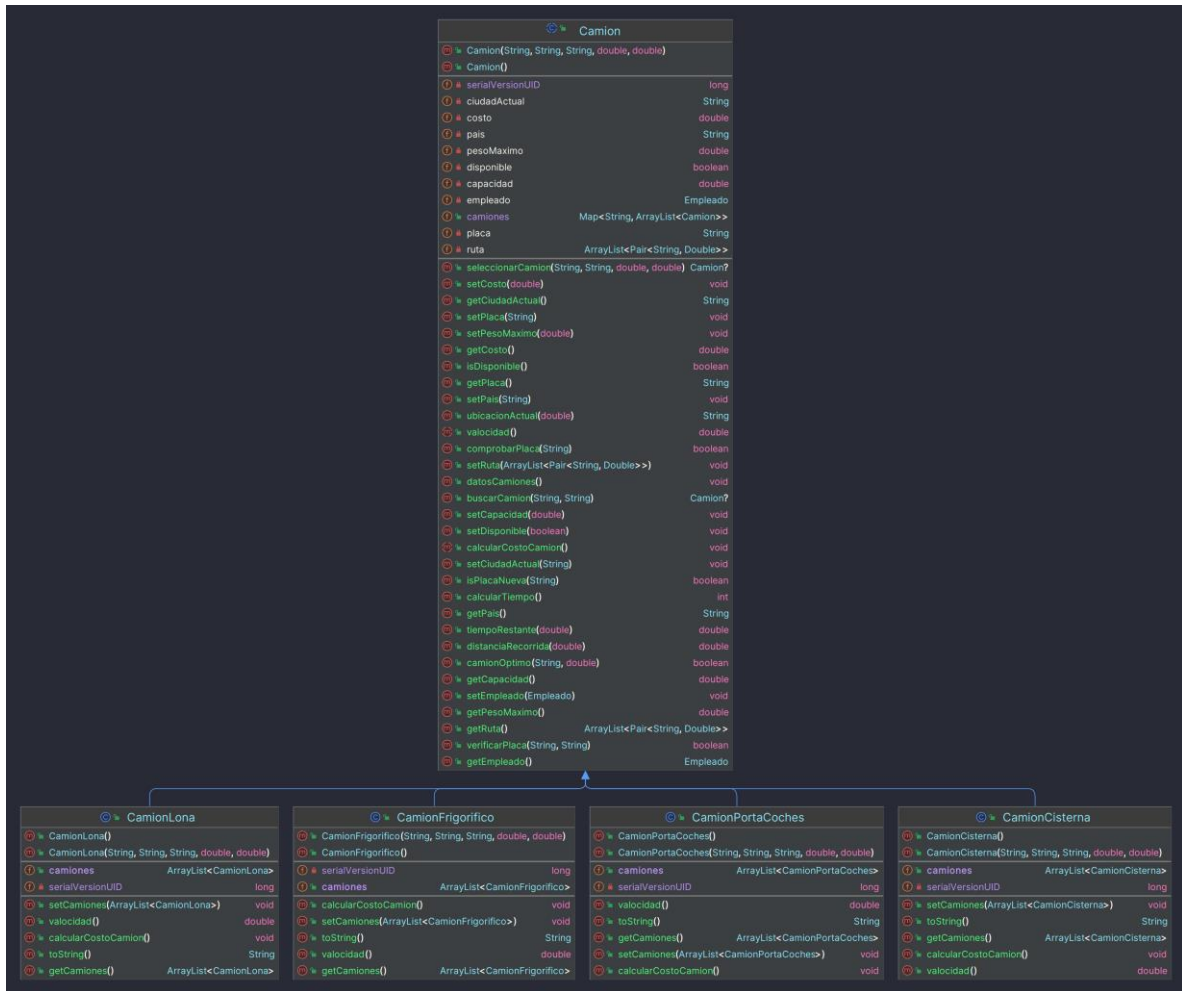


Diagrama UML paquete camion

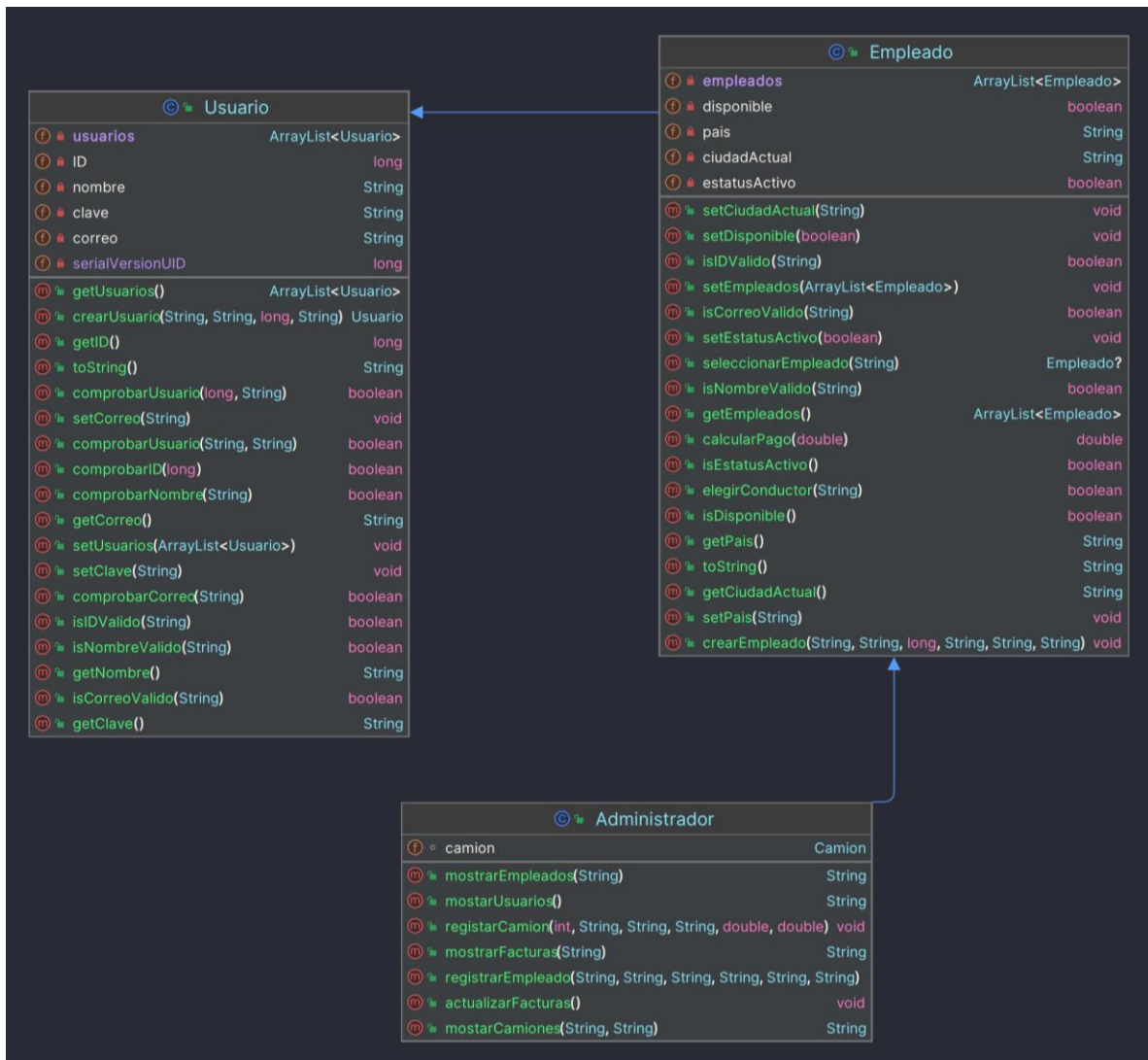


Diagrama uml paquete entidades

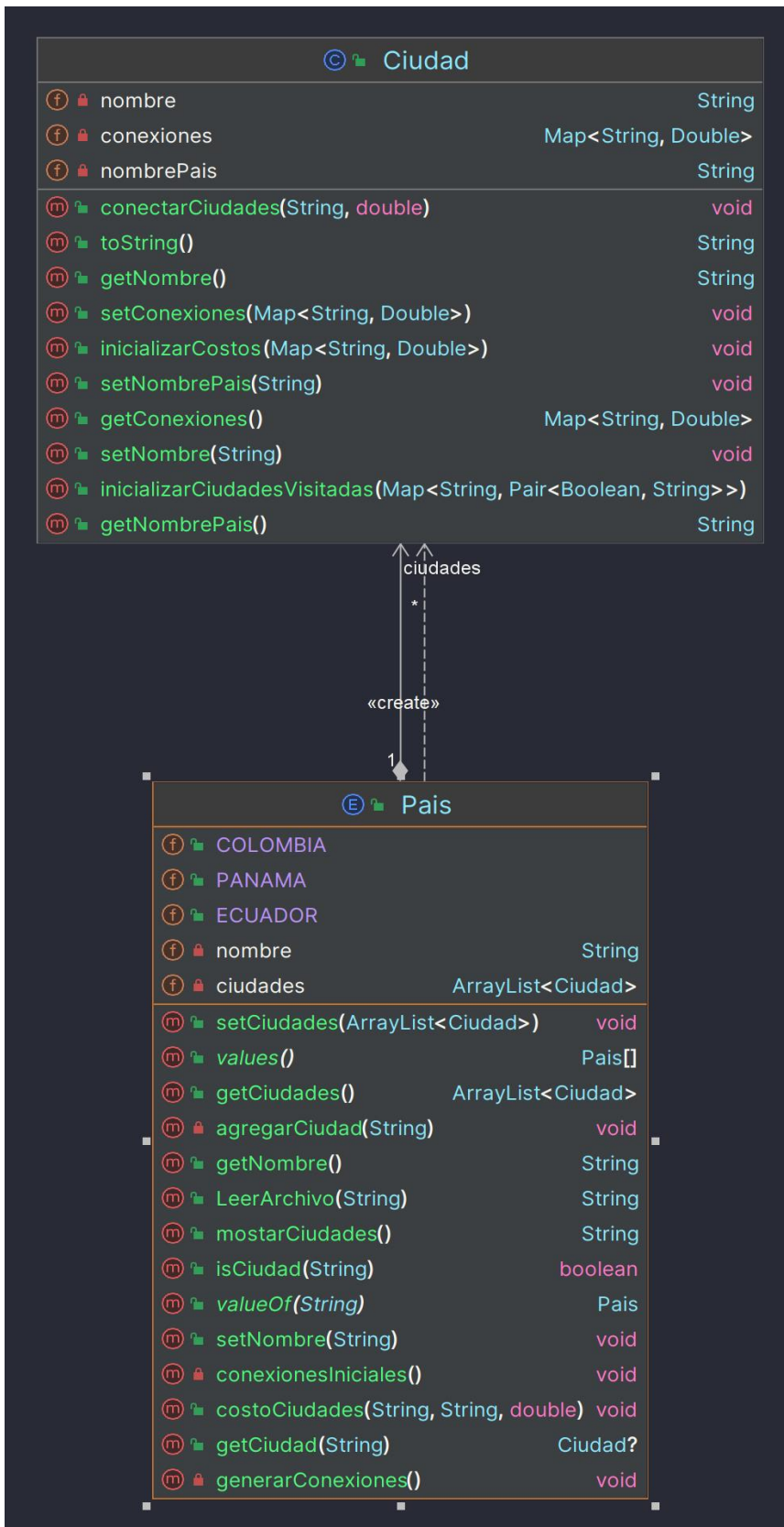


Diagrama uml paquete país

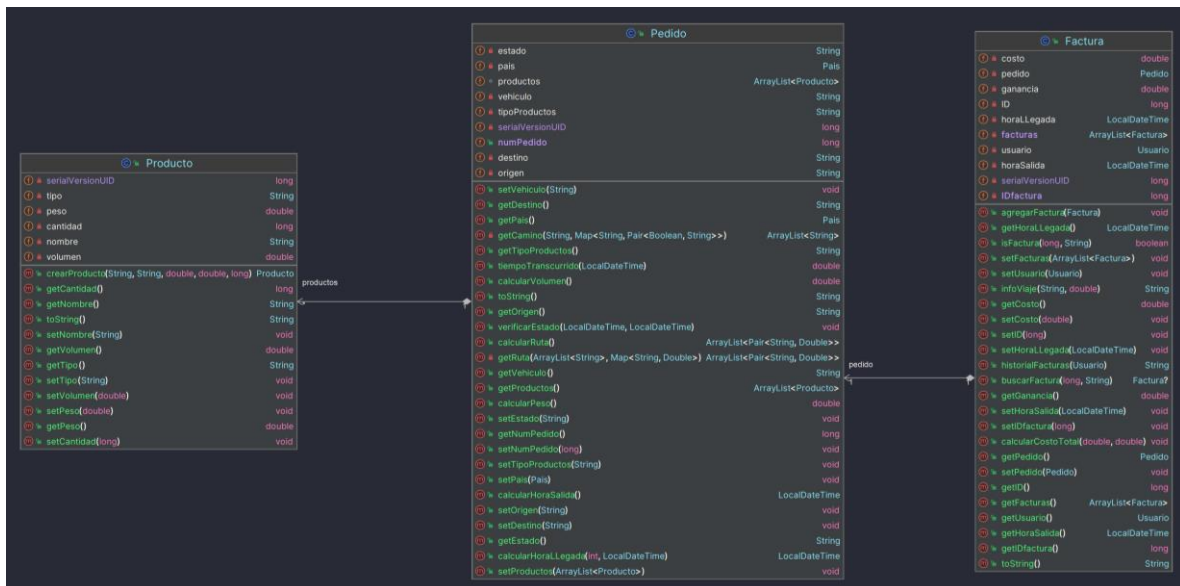


Diagrama uml paquete producto

Descripción de la implementación de características de programación orientada a objetos

A. Clases Abstracta (1) y Métodos Abstractos (1):

Ruta del archivo: *SistemaGestionTransporte/src/gestorAplicaciones/camion/Camion.java*

La clase *Camion* es abstracta. Además, es la clase que se tomará como base para la creación de los diferentes tipos de camiones (camión cisterna, camión frigorífico, camión lona y camión porta coches)

Línea 28

```

12  /** Camion es una clase abstracta que implementa la interfaz Serializable. Sirve como clase base ...*/
13  38 usages 4 inheritors Michael Garcia +1
28  public abstract class Camion implements Serializable {
29      /** ...
30
31      public final static Map<String,ArrayList<? extends Camion>> camiones = new HashMap<>();
32
  
```

Asimismo, al ser una clase abstracta debe de poseer como mínimo un método abstracto. Por ende, la clase *Camion* tiene los siguientes métodos abstractos:

Método abstracto para el calculo del costo del camión

Línea 138

```

134
135  /** debe ser implementado en las subclases para calcular el costo del camion. */
136  1 usage 4 implementations Julian Salazar
138  public abstract void calcularCostoCamion();
139  /** ...*/
  
```

Método abstracto para obtener la velocidad del camión

Línea 153

```
153 2 usages 4 implementations Julian Salazar  
154 public abstract double velocidad();
```

B. Interfaces (1) diferentes a los utilizados para serializar los objetos en el punto de la persistencia. Deberá ser propio del dominio a implementar.

La interfaz Sesion tendrá los métodos abstractos que se implementaran en los diferentes tipos de sesiones (usuario, administrador y empleado).

Ruta del archivo: *SistemaGestionTransporte/src/uiMain/sesion/Sesion.java*

Línea 8 - 29

```
package uiMain.sesion;  
  
import gestorAplicaciones.entidades.Usuario;  
  
/**  
 * @author Julian Salazar, Michael Garcia  
 */  
3 usages 3 implementations Julian Salazar  
8 public interface Sesion {  
9     /**  
10      * Método para iniciar la sección.  
11     */  
12     3 usages 3 implementations Julian Salazar  
13     public void Inicio();  
14     /**  
15      * Método para mostrar el menu de opciones de la sección.  
16     */  
17     4 usages 3 implementations Juan S  
18     public void showMenu();  
19     /**  
20      * Método para pedir información necesaria para iniciar sesión.  
21     */  
22     3 usages 3 implementations Julian Salazar  
23     public void ingresar();  
24     /**  
25      * Método para validar la información del usuario.  
26     */  
27     * @param usuario El nombre de usuario.  
28     * @param clave La contraseña del usuario.  
29     * @return El objeto Usuario correspondiente si la información es válida, o null en caso contrario.  
30     */  
31     2 usages 3 implementations Michael Garcia  
32     public Usuario validarInformacion(String usuario, String clave);  
33 }
```

Implementación en las clases:

Sesión administradora

Ruta del archivo: *SistemaGestionTransporte/src/uiMain/sesion/SesionAdministrador.java*

Línea 18


```

2 usages  Julian Salazar +1
18  public class SesionAdministrador implements Sesion {
    24 usages
19      int opcion = 0; //toma el valor de la opcion ingresado por consola
    9 usages
20      Pais pais;
21
    9 usages
22      String p;
23
    8 usages
24      Administrador admin = new Administrador();
25

```

Sesión de empleado

Ruta del archivo: *SistemaGestionTransporte/src/uiMain/sesion/SesionTrabajador.java*

```

2 usages  Michael Garcia +1
15  public class SesionTrabajador implements Sesion {
    7 usages
16      int opcion = 0;
17
    6 usages
18      Empleado empleado;
19

```

Sesión de empleado

Ruta del archivo: *SistemaGestionTransporte/src/uiMain/sesion/SesionUsuario.java*

Línea 22

```

2 usages  Julian Salazar +1
22  public class SesionUsuario implements Sesion {
    17 usages
23      int opcion = 0;

```

C. Herencia (1)

Los diferentes tipos de camiones heredan de la clase *Camion*. Por lo cual, extienden sus funciones y atributos

Por ejemplo, en el caso de los camiones lona:

Ruta del archivo: *SistemaGestionTransporte/src/gestorAplicaciones/camion/CamionLona.java*

Línea 17

```

60 usages  Julian Salazar +1
17  public class CamionLona extends Camion implements Serializable {
18      //serializador
    no usages

```

D. Ligadura dinámica (2) asociadas al modelo lógico de la aplicación

Se utiliza ligaduras dinámicas con la clase padre *Camion* para la creación de sus clases hijas *CamionCisterna*, *CamionFrigorifico*, *CamionLona* y *CamionPortaCoches*

Ruta del archivo:

SistemaGestionTransporte/src/gestorAplicaciones/entidades/Administrador.java

Línea 17

```
3 usages  Michael Garcia +1
16 public class Administrador extends Empleado{
17     Camion camion;
18 }
```

Línea 123 - 127

```
1 usage  Julian Salazar +1
120 public void registrarCamion(int opcion, String placa, String pais, String ciudadActual, double pesoMaximo, double capacidad) {
121     //nueva instancia de Camion
122     switch (opcion) {
123         case 1 -> camion = new CamionCisterna(placa, pais, ciudadActual, pesoMaximo, capacidad);
124         case 2 -> camion = new CamionFrigorifico(placa, pais, ciudadActual, pesoMaximo, capacidad);
125         case 3 -> camion = new CamionLona(placa, pais, ciudadActual, pesoMaximo, capacidad);
126         case 4 -> camion = new CamionPortaCoches(placa, pais, ciudadActual, pesoMaximo, capacidad);
127         default -> System.out.println("opcion no valida");
128     }
129 }
130 }
```

E. Atributos de clase (1) y métodos de clase (1)

La clase *Factura* utiliza el atributo de clase facturas para almacenar instancias de *Factura*. Las cuales serian las facturas del usuario ingresado.

Ruta del archivo: *SistemaGestionTransporte/src/gestorAplicaciones/producto/Factura.java*

Línea 30

```
24 public class Factura implements Serializable{
25
26     //serializador
27     @Serial
28     private static final long serialVersionUID = 1L;
29     //atributos
30     private static ArrayList<Factura> facturas = new ArrayList<Factura>();
31 }
```

También, utiliza el método de clase *agregarFacturas* para agregar las facturas generadas por el usuario ingresado

Línea 151

```
1 usage  Michael Garcia
151 public static void agregarFactura(Factura factura) { Factura.facturas.add(factura); }
154 }
```

F. Uso de constante (1 caso)

En la clase *Factura* se tiene una constante con el valor de la ganancia que se obtiene por pedido.

Ruta del archivo: *SistemaGestionTransporte/src/gestorAplicaciones/producto/Factura.java*

Línea 32

```

30     private static ArrayList<Factura> facturas = new ArrayList<Factura>();
    4 usages
31     private static long IDfactura = 100000000;
    3 usages
32     private final double ganancia = 1.25;
    5 usages
33     private Pedido pedido;
    5 usages
34     private Usuario usuario;

```

G. Encapsulamiento (private, protected y public).

La clase *Camion* tiene atributos privados de los cuales el solo puede acceder, y tipo público, como: el constructor y el atributo de clase *camiones*.

```

28 public abstract class Camion implements Serializable {
29     //nota: capacidad de los camiones:(1 tn, 20 m3), (8 ton, 35 m3), (17 ton, 42 m3) y 24 (24 ton, 48m3);
30     //atributos
31     public final static Map<String,ArrayList<? extends Camion>> camiones = new HashMap<>();
32
33     @Serial
34     private static final long serialVersionUID = 1L;
    5 usages
35     private String placa;
    3 usages
36     private double pesoMaximo;
    3 usages
37     private double capacidad;
    2 usages
38     private double costo;
    3 usages
39     private String pais;
    3 usages
40     private boolean disponible;
    3 usages
41     private String ciudadActual;
    3 usages
42     private ArrayList <Pair<String, Double>> ruta;
    2 usages
43     private Empleado empleado;
44
45     //constructor
    4 usages Julian Salazar +1
46     public Camion(String placa,String pais, String ciudadActual,double pesoMaximo,double capacidad){
47         this.placa = placa;
48         this.pesoMaximo = pesoMaximo;
49         this.pais = pais;
50         this.ciudadActual = ciudadActual;
51         this.capacidad = capacidad;
52         this.disponible = true;
53     }

```

H. Los siguientes conceptos asociados a la POO:

i. Sobrecarga de métodos (1) y constructores (2)

Se realiza sobrecarga para el método *comprobarUsuario* de la clase *Usuario* con el fin de validar el inicio de sesión del usuario con ID o nombre.

Ruta del archivo:

SistemaGestionTransporte/src/gestorAplicaciones/entidades/Usuario.java

Línea 102 - 124

```
public boolean comprobarUsuario(String nombre, String clave){  
    return this.nombre.equals(nombre) && this.clave.equals(clave);  
}  
  
/**...*/  
2 usages  Julian Salazar +1  
public boolean comprobarUsuario(long id, String clave){  
    return this.ID == id && this.clave.equals(clave);  
}
```

También, se realiza sobrecarga en el constructor de la clase *Usuario* para así generar usuarios con contraseña predeterminada ("0000")

Línea 33 - 40

```
33  public Usuario (String nombre, String clave, long id, String correo) {  
34      this.nombre = nombre;  
35      this.clave = clave;  
36      this.ID = id;  
37      this.correo = correo;  
38  }  
39  }  
40  public Usuario (String nombre, long id, String correo) { this(nombre, clave: "0000", correo, id); }
```

Por otro lado. Igualmente se crea sobrecarga en el constructor de la clase *Empleado*.

Ruta del archivo:

SistemaGestionTransporte/src/gestorAplicaciones/entidades/Empleado.java

Línea 27 - 37

```
26  //constructor  
27  2 usages  Julian Salazar +1  
    public Empleado (String nombre, String clave, long id, String correo) { super(nombre, clave, id, correo); }  
30  
31  4 usages  Michael Garcia +1  
    public Empleado (String nombre, String clave, long id, String correo, boolean statusActivo, String pais, String ciudadActual){  
32      this(nombre, clave, id, correo);  
33      this.estatusActivo = statusActivo;  
34      this.pais = pais;  
35      this.ciudadActual = ciudadActual;  
36      this.disponible = true;  
37      Empleado.empleados.add(this);  
38  }
```

- ii. Manejo de referencias `this` para desambiguar y `this()` entre otras. 2 casos mínimo para cada caso

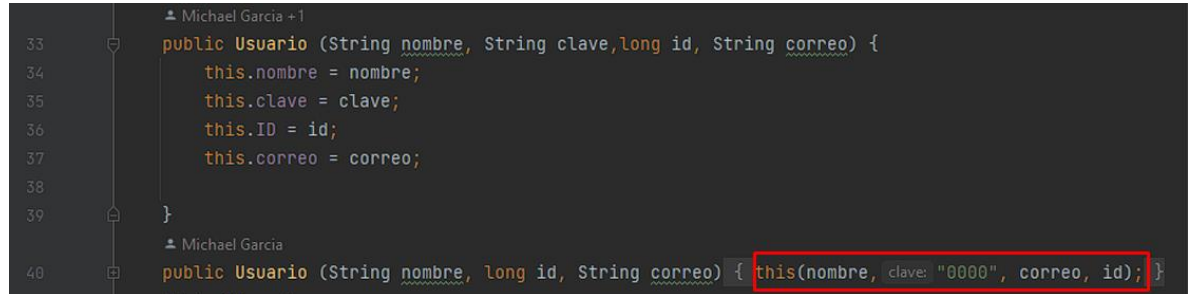
Referencias `this()`

Se utiliza el `this()` en la clase *Usuario* para hacer referencia al constructor *Usuario(String, String, long, String)*.

Ruta del archivo:

SistemaGestionTransporte/src/gestorAplicaciones/entidades/Usuario.java

Línea 40



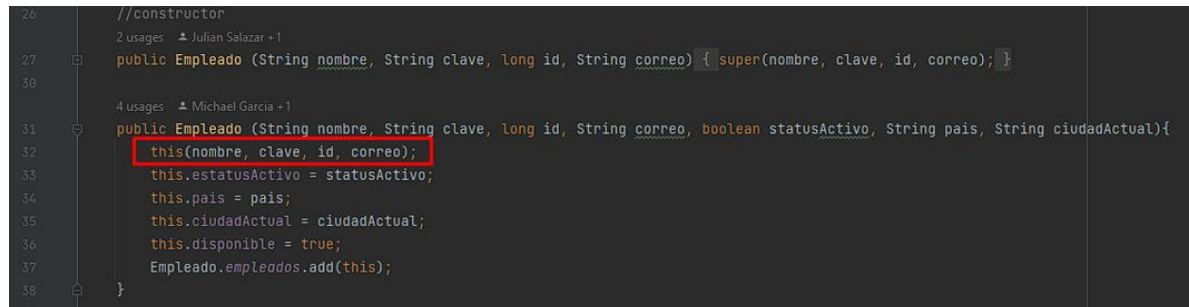
```
33 public Usuario (String nombre, String clave, long id, String correo) {
34     this.nombre = nombre;
35     this.clave = clave;
36     this.ID = id;
37     this.correo = correo;
38 }
39
40 public Usuario (String nombre, long id, String correo) { this(nombre, clave: "0000", correo, id); }
```

Se utiliza para hacer referencia al constructor *Empleado(String, String, long, String)*

Ruta del archivo:

SistemaGestionTransporte/src/gestorAplicaciones/entidades/Empleado.java

Línea 32



```
26 //constructor
27 public Empleado (String nombre, String clave, long id, String correo) { super(nombre, clave, id, correo); }
28
29
30
31 public Empleado (String nombre, String clave, long id, String correo, boolean statusActivo, String pais, String ciudadActual){
32     this(nombre, clave, id, correo);
33     this.estatusActivo = statusActivo;
34     this.pais = pais;
35     this.ciudadActual = ciudadActual;
36     this.disponible = true;
37     Empleado.empleados.add(this);
38 }
```

Referencias `this`

Se usa el `this` para desambiguar los atributos de la clase *Empleado*. (nombre, clave, ID y correo)

Ruta del archivo:

SistemaGestionTransporte/src/gestorAplicaciones/entidades/Usuario.java

Línea 34 - 37

```
33 public Usuario (String nombre, String clave, long id, String correo) {  
34     this.nombre = nombre;  
35     this.clave = clave;  
36     this.ID = id;  
37     this.correo = correo;  
38 }  
39  
40 public Usuario (String nombre, long id, String correo) { this(nombre, clave: "0000", correo, id); }
```

Lo mismo ocurre con la clase *Empleado* se utiliza para desambiguar los parámetros (estatusActivo, pais, ciudadActual, disponible)

Ruta del archivo:

SistemaGestionTransporte/src/gestorAplicaciones/entidades/Empleado.java

Línea 33 - 36

```
26 //constructor  
27 public Empleado (String nombre, String clave, long id, String correo) { super(nombre, clave, id, correo); }  
30  
31 public Empleado (String nombre, String clave, long id, String correo, boolean statusActivo, String pais, String ciudadActual){  
32     this(nombre, clave, id, correo);  
33     this.estatusActivo = statusActivo;  
34     this.pais = pais;  
35     this.ciudadActual = ciudadActual;  
36     this.disponible = true;  
37     Empleado.empleados.add(this);  
38 }
```

iii. Implementación de un caso de enumeración

Se utiliza esta enumeración para representar diferentes países de los cuales se tiene 3 (Colombia, Panamá y Ecuador). A partir del enum es posible Almacenar el nombre del país y la lista de sus ciudades.

Ruta del archivo: *SistemaGestionTransporte/src/gestorAplicaciones/pais/Pais.java*

```
11 usages  Julian Salazar +1
public enum Pais {
    1 usage
    COLOMBIA( nombre: "Colombia"),
    1 usage
    PANAMA( nombre: "Panama"),
    1 usage
    ECUADOR( nombre: "Ecuador");

    //atributos
    6 usages
    private String nombre;
    12 usages
    private ArrayList<Ciudad> ciudades;
```

Implementación:

Se implementa en la función *Main* para obtener el nombre y ciudad del país desde el cual se realizar el pedido.

Ruta del archivo: *SistemaGestionTransporte/src/uiMain/Main.java*

Línea 158 - 163

```
6 usages  Julian Salazar
142 @
143 public static Pais seleccionarPais() {
144     //seleccionar pais donde se realizara el transporte del pedido.
145     int opcion;
146     do{
147         System.out.println("""
148             Ingrese:
149             1. Colombia.
150             2. Ecuador.
151             3. Panama.
152             0. Salir.""");
153
154         opcion = Main.getOption();
155         switch (opcion){
156             case 0:
157                 break;
158             case 1:
159                 return Pais.COLOMBIA;
160             case 2:
161                 return Pais.ECUADOR;
162             case 3:
163                 return Pais.PANAMA;
164             default:
165                 System.out.println("Opcion no valida.");
166         }
167     }while (opcion != 0);
168     return null;
169 }
170
171 }
```

Descripción de cada una de las 5 funcionalidades implementadas

Precondición

Registro del cliente como usuario:

Para la creación y gestión de pedidos, es necesario que el cliente se encuentre registrado como usuario.

Descripción de proceso:

El proceso comienza desde la clase *Main* donde se despliega el menú con las opciones iniciales del aplicativo, al seleccionar la opción "1" se crea un objeto tipo *SesionUsuario* del cual se utiliza el método abstracto *Inicio*. Luego, se muestra un submenú referente al registro e inicio de sesión del usuario, al digitar la opción "2" se ejecuta el método *registrarUsuario* de la clase *SesionUsuario*. Se inicia la rutina para la creación de usuario, se solicitará el nombre, ID, correo y clave del cliente, al finalizar correctamente el procedimiento se llegará a la pantalla principal del usuario mostrada por el método *showMenu*

Nota: Se realiza la verificación de cada parámetro a la hora de realizar el registro. Por ejemplo: No se admiten correos que no contenga el símbolo "@" o se encuentren registrados previamente.

Vista final de proceso:

```
Bienvenido/a TestUser
Ingrese:
1. Realizar pedido.
2. Seguir pedido.
3. historial de pedido.
0. salir.
Elija una opcion:
|
```

FUNCIONALIDADES

Gestión de pedido

Transportes Ltda brinda al usuario un conjunto de opciones para la gestión de sus pedidos. En donde será posible contratar nuestros servicios desde nuestras sedes disponibles (Colombia, Ecuador y Panamá) y hacer uso de la variedad de camiones de transporte, con el fin de generar al cliente escenarios que se adapten a su necesidad.

Descripción de proceso:

Al seleccionar la opción "Realizar pedido" se ejecutará el método *gestionarPedido* de la clase *SesionUsuario*. Se comenzará con la creación de un objeto tipo *Pedido* del cual se ira poblando poco a poco de información.

Primero se mapeará el país del cual se realizará el envío, será seleccionado desde el método *seleccionarPais* de la clase *Main* del cual se retornará un enum con la información del país elegido. Luego, se seleccionará la ciudad de origen y destino del pedido. (información desplegada del enum). Ahora, se iniciará el proceso de construcción del pedido. Es decir, la selección de productos y el tipo de carga que llevara el camión. Es posible escoger entre 5 tipos de cargas y dependiendo de esta se elige el tipo de camión:

- Carga perecedera: Camion de la clase *CamionFrigorifico*
- Carga frágil: Camion de la clase *CamionLona*
- Carga ADR: Camion de la clase *CamionCisterna*
- Carga de coches: Camion de la clase *CamionPortaCoches*
- Carga general: Camion de la clase *CamionLona*

Esta se opción se almacenará en la variable *tipoCarga*, la cual será pasada como parámetro para el método *seleccionarProductos* encargado de la creación, visualización, confirmación y eliminación de productos, dichos productos son almacenados en un *ArrayList* que es retornado para ser agregado al atributo de productos del objeto *pedido*. Por último, se seleccionará el camión para el viaje con el método *seleccionarCamion* de la clase *Camion* y brindar un vehículo que utilice los recursos óptimos. Es decir, que dependiendo del peso de los productos se asigne un camión que pueda llevar los elementos sin sobrepasar su máximo de carga. Si el método anteriormente mencionado retorna *null* se realiza la búsqueda de un camión cercano que pueda cumplir con las necesidades a partir del método *camionCercano* de la clase *Camion*. Para finalizar se agrega la placa del vehículo al pedido y se asigna un empleado que se encuentre disponible en la ciudad de origen al camión. Asimismo, se procesará las tareas finales, tales como:

- Cambiar la disponibilidad del camión y empleado
- Agregar el estado "Por confirmar" al pedido
- Agregar la factura al método de clase *agregarFactura*
- Mensaje final (Pedido realizado)

Vista final de proceso:

```
Factura Nro: 1000000000
Desde: Monteria
hasta: Armenia
Portatil      x1
Vaso          x2
Estado: Por confirmar
Vendido a:
nombre: testUser
id: 1
correo: a@b.com
Hora Salida: 2023-05-25 06:00:00
Hora llegada: 2023-05-25 11:00:00
Costo: $51.0
```

```
Seleccione.
1. Confirmar pedido.
2. Cancelar pedido.
```

```
Eliga una opcion:
1
Pedido realizado
```

```
Ingrese:
1. Realizar pedido.
2. Seguir pedido.
3. historial de pedido.
0. salir.
Eliga una opcion:
```

Seguimiento del pedido

Funcionalidad planteada para brindar al usuario seguimiento en tiempo real de su petición. Es decir, el cliente puede consultar en cualquier momento el estado de su orden y esta variara dependiendo del tiempo transcurrido desde que se realizó el pedido.

Actualmente, se tiene cuatro tipos de estados:

- Confirmado
- Enviado
- Entregado

Descripción de proceso:

Al seleccionar la opción "Seguir pedido" se ejecutará el método *seguirPedido* de la clase *SesionUsuario*. En donde se solicitará el ID de la factura para realizar la búsqueda en el método de clase *buscarFactura* de la clase *Factura*, Si no encuentra la factura no realiza ningún otro proceso. Caso contrario, al obtener la factura realiza la actualización de información y obtiene el pedido relacionado a la factura.

Por último, verifica si el pedido se encuentra en estado "Enviado". Si es así, muestra un resumen del tiempo faltante del camión.

Nota: No se especifica temáticas como el cambio de estado y el cálculo de tiempo. Debido a que pertenecen a otras funcionalidades.

Vista final de proceso:

```
Ingrese ID factura:
1000000000
Factura Nro: 1000000000
Desde: Monteria
hasta: Armenia
Portatil      x1
Vaso          x2
Estado: Confirmado
Vendido a:
nombre: testUser
id: 1
correo: a@b.com
Hora Salida: 2023-05-25 06:00:00
Hora llegada: 2023-05-25 11:00:00
Costo: $51.0

Ingrese:
1. Realizar pedido.
2. Seguir pedido.
3. historial de pedido.
0. salir.
Elija una opcion:
|
```

Tarifa dinámica

La tarifa dinámica consiste en el calculo de valor del pedido a partir de las diferentes condiciones propuestas por el cliente. Tales como:

- Tipo de carga
- Ruta
- Peso Vehiculo

Descripción de proceso:

La funcionalidad comienza desde el método *calcularTarifa* de la clase *SesionUsuario*. En esta se crear un objeto tipo *Factura* tomando como parámetros el pedido y el usuario, a partir del pedido se calcula la ruta con el método *calcularRuta* el cual implementa el algoritmo de Dijkstra para encontrar el camino corto entre ciudades de origen y destino (la información de estos se obtiene a partir de un .txt almacenados en el proyecto), todo esto se realiza a partir del entendimiento de grafos (clase implicada *Pair*). Al calcular la ruta esta es asignada al camión, donde calcula el costo de camión a partir del método abstracto *calcularCostoCamion* de la clase *Camion*.

Costos fijos por tipo de camión:

- *Camion Refrigerico: factor = 0.01*
- *Camion Lona: factor = 0.005*
- *Camion Cisterna: factor = 0.005*
- *Camion Porta Coches: factor = 0.008*

El calculo se hace a partir de la ecuación: $\text{km} * \text{capacidad} * \text{factor}$

Al tener el costo del camión se procede a calcular el costo del empleado con el método *calcularPago* de la clase *Empleado* da como resultado la sumatoria del costo del empleado mas el costo del camión

El cálculo se realiza mediante la ecuación: $\text{costoCamion} + \text{costoCamion} * 0.2$

Por último, se genera el costo total de la factura, calculada con el método *calcularCostoTotal* de la clase *Factura*. Toma como parámetros el costo del pedido (costo camión + costo empleado) y la capacidad del camión.

El calculo se realiza mediante la siguiente ecuación: $\text{capacidad} * 0.05 * \text{costoPedido} * \text{ganancia}$

En donde ganancia es una constante de la clase *Factura* que es igual a 1.25

Vista final de proceso:

```
Factura Nro: 100000000
Desde: Monteria
hasta: Armenia
Portatil      x1
Vaso          x2
Estado: Confirmando
Vendido a:
nombre: testUser
id: 1
correo: a@b.com
Hora Salida: 2023-05-25 06:00:00
Hora Llegada: 2023-05-25 11:00:00
Costo: $51.0
```

Calcular hora de salida y llegada

Funcionalidad creada para el calculo de tiempos entre salida y llegada del pedido. Se tiene mayor control acerca del proceso de transporte de la orden y generar al usuario información sobre las fechas estipuladas en las que puede llegar su petición.

Descripción de proceso:

El usuario en el proceso de generación de pedido ejecuta el método *calcularTiempo* de la clase *SesionUsuario*. Consiste en el calculo de tiempo de salida y llegada del pedido. Dichos cálculos se desean establecer en los atributos *setHoraSalida* y *setHoraLlegada* de la clase *Factura*.

Para el parámetro *setHoraSalida* se utiliza el método *calcularHoraSalida* del objeto pedido que hemos creado. Realiza el calculo a partir de la fecha actual con el método *now* de la clase *LocalDateTime* obteniendo las horas y comparándolas con las siguientes condiciones:

- Si $14 > \text{hora} > 5$. Entonces la hora de salida será 15:00:00
- Si $14 < \text{hora}$. Entonces la hora de salida será a las 06:00:00 del siguiente día
- Si ninguna de las anteriores cumple. Entonces la hora de salida será a las 06:00:00 del mismo día

Nota: Las fechas se devuelven en formato "yyyy-MM-dd HH:mm:ss"

Luego. Para el calculo de la hora de llegada se utiliza el método *calcularHoraLlegada* del pedido donde toma como parámetro el tiempo que se demora el camión en horas y la hora de salida relacionada a la factura. Su resultado simplemente es la sumatoria de las horas que se demora el camión más la hora de salida.

Vista final de proceso:

```
Desde: Monteria
hasta: Armenia
Portatil      x1
Vaso          x2
Estado: Confirmando
Vendido a:
nombre: testUser
id:           1
correo: a@b.com
Hora Salida: 2023-05-25 06:00:00
Hora Llegada: 2023-05-25 11:00:00
Costo: $51.0
```

Actualizar información envió

Actualizar información envió

La actualización de información de envió plantea generar escenarios dinámicos que dan alusión al seguimiento de un pedido en tiempo real. Estableciendo distintos estados dependiendo del tiempo que ha transcurrido desde la solicitud de una orden.

¿Cuándo hay cambios de estado?:

El estado inicial del pedido al realizarse siempre será generado como: “Por confirmar”. Al utilizar por primera vez la funcionalidad de pedido el estado se actualizará a “Confirmado”. Luego, cuando se supere el tiempo de salida de la orden su estado cambiara a “Enviado” y por último si ya ha transcurrido el tiempo de la hora de llega del pedido. Entonces, se pasará a su estado final “Entregado”.

Descripción del proceso:

La funcionalidad inicia desde el método actualizarInformacion de la clase Main la cual recibe como parámetro una factura tipo Factura. Primer se declara una variable para guardar el estado anterior del pedido en la variable estadoAnterior. Luego pedido usa el método verificarEstado para actualizar el atributo estado del pedido si este edido difiere de la variable estadop anterior y es igual a “Entregado” los objetos camion pasaran su atributo disponible a true, ciudadActual a la ciudad de destino del pedido y los mismo con los atributos disponible y ciudadActual de empleado.

```

/**
 *
 * @param factura parametro una instancia de Factura que va a ser modificado
 *
 *actualiza el atributo estado de la instancia Pedido, atribtu de factura.
 */
4 usages  ▴ Julian Salazar +1
public static void actualizarInformacion(Factura factura){

    //funcionalidad 5: actualizar informacion de envio

    String estadoAnterior;

    if(!factura.getPedido().getEstado().equals("Entregado"))) {
        Pedido pedido = factura.getPedido();
        estadoAnterior = pedido.getEstado();
        pedido.verificarEstado(factura.getHoraSalida(), factura.getHoraLlegada());
        if (!pedido.getEstado().equals(estadoAnterior) && pedido.getEstado().equals("Entregado")) {
            Camion camion = Camion.buscarCamion(pedido.getTipoProductos(), pedido.getVehiculo());
            camion.setDisponible(true);
            camion.setCiudadActual(pedido.getDestino());
            camion.getEmpleado().setDisponible(true);
            camion.getEmpleado().setCiudadActual(pedido.getDestino());
        }
    }
}

```

Funcionalidad 5

Manual de usuario

Menú principal: Contamos con un menú principal de bienvenida en el que se debe ingresar un número del 0 al 3 para realizar diferentes operaciones dentro de nuestra aplicación.

```
-----Bienvenidos a Transportes ltda.-----
Presione:
1. Ingresar como usuario
2. Ingresar como empleado
3. Ingresar como administrador
0. Salir
Eliga una opcion:
|
```

MODULO INGRESAR COMO USUARIO

Al seleccionar esta opción se despliega el submenú de usuario. El cual nos muestra lo siguiente:

```
Ingrese:
1. iniciar Seccion.
2. Registrarse.
0. salir.
Eliga una opcion:
```

Opción 1 - Iniciar sesión:

Al ingresar el numero 1 nos pedirá los datos de acceso necesario para poder ingresar a la cuenta registrada en la compañía. Solicitará lo siguiente:

- Usuario/ID: Puedes elegir entre dos maneras de identificarte. Por ID o por nombre de usuario.
- Clave: Contraseña con la que creaste tu usuario

Si ingresas un usuario valido te enviara al menú de bienvenida.

```
Ingrese:
1. iniciar Seccion.
2. Registrarse.
0. salir.
Eliga una opcion:
1
USuario/ID:
1
clave:
2
```

NOTA: Si ingresar algún campo incorrecto saldrá un mensaje mencionándote que algún parámetro no es valido

Algunos usuarios previamente registrados:

nombre	id	contraseña
Juan	1001	0000
Maria	1002	0000
Pedro	1003	0000
Laura	1004	0000

Carlos	1005	0000
Ana	1006	0000
Luis	1007	0000
Marta	1008	0000
Andres	1009	0000
Sofia	1010	0000
Javier	1011	0000

Opción 2 - Registrarse:

Al ingresar el numero 2 comienza el proceso de registro de usuario, se solicitará el nombre, ID, correo, clave. Si se ha ingresado todos los datos correctamente. Te llevara al menú de bienvenida de usuario.

NOTA: Si se ingresa en algún campo un valor no valido o ya se encuentra registrado el ID, se generará un mensaje

Menú de registro

```
Ingrese:
1. iniciar Seccion.
2. Registrarse.
0. salir.
Eliga una opcion:
2
nombre:
test
ID:
2023
correo:
test@gmail.com
Clave:
123
```

MODULO MENU DE BIENVENIDAD OPCION 1

```
Bienvenido/a test

Ingrese:
1. Realizar pedido.
2. Seguir pedido.
3. historial de pedido.
0. salir.
Eliga una opcion:
```

Opción 1 - Realizar pedido

Al seleccionar el proceso de realizar un pedido se desplegará un submenú para elegir el país en el cual se desea realizar el envío.

```
Ingrese:
1. Colombia.
2. Ecuador.
3. Panama.
0. Salir.
Eliga una opcion:
```

NOTA: Cada opción tiene consigo diferentes ciudades e información

Al elegir cualquiera de las opciones anteriores nos desplegara el siguiente menú:

Ciudad de origen

```
Ingrese:
1. Ingresar ciudad de Origen
2. ver lista de ciudades.
0. Salir.
Eliga una opcion:
```

Opción 1 – Ingresar ciudad de Origen

Si eliges esta opción, luego debes de ingresar la ciudad de origen del pedido. Teniendo en cuenta que debe ser escrita tal cual y como se muestra en la lista de ciudades de la opción 2. Si has realizado bien el proceso iras al menú de ciudad de destino.

Opción 2 – ver lista de ciudades

Muestra la lista de ciudades disponibles para ese país. Se ve de esta manera:

```
Ingrese:
1. Ingresar ciudad de Origen
2. ver lista de ciudades.
0. Salir.
Elija una opcion:
2
ciudades de Colombia
Riohacha
    Valledupar
    Barranquilla
    Cucuta
    Monteria
    Bucaramanga
    Irinida
    Medellin
    Boyaca
    Mitu
    Bogota
    Armenia
    Quibdo
    Villavicencio
    Florencia
    Neiva
    Cali
    Pasto
```

Ciudad de destino

```
Ingrese:
1. Ingresar ciudad de Destino
2. ver lista de ciudades.
0. Salir.
Elija una opcion:
```

Opción 1 – Ingresar ciudad de destino

Si eliges esta opción, luego debes de ingresar la ciudad de destino del pedido. Teniendo en cuenta que debe ser escrita tal cual y como se muestra en la lista de ciudades de la opción 2. Si has realizado bien el proceso iras al menú de selección de tipo de producto.

Opción 2 – ver lista de ciudades

Muestra la lista de ciudades disponibles para ese país. Se ve de esta manera:

```
ciudades de Colombia:
    Riohacha
    Valledupar
    Barranquilla
    Cucuta
    Monteria
    Bucaramanga
    Irinida
    Medellin
    Boyaca
    Mitu
    Bogota
    Armenia
    Quibdo
    Villavicencio
    Florencia
    Neiva
    Cali
    Pasto

Ingrese:
1. Ingresar ciudad de Destino
2. ver lista de ciudades.
0. Salir.
Elija una opcion:
|
```

Tipo de producto:

Seleccione el tipo de producto a transportar

```
Ingrese:
1. Carga perecedera.
2. Carga fragil.
3. Carga ADR.
4. Carga de coches.
5. Carga general.
0. Salir.
Elija una opcion:
```

Puedes elegir cualquiera de nuestras 5 opciones que se adapten a las necesidades.

Si eliges una opción valida debe de aparecer el menú de creación de productos.

Creación de productos

```
1. Ingresar producto.  
2. Ver productos ingresados.  
3. Confirmar productos.  
4. Eliminar producto.  
0. Descartar productos.  
Elija una opcion:
```

Opción 1 – Ingresar producto

Se ingresa el producto que se desea llevar. Se pediría como parámetros: El nombre del producto, peso de producto (kg), Volumen del producto (m3) y cantidad de ese producto.

```
Ingrese:  
1. Ingresar producto.  
2. Ver productos ingresados.  
3. Confirmar productos.  
4. Eliminar producto.  
0. Descartar productos.  
Elija una opcion:  
1  
nombre del producto:  
TV  
peso del producto (kg):  
30  
volumen del producto (m3):  
1  
Cantidad de ese producto:  
1
```

Opción 2 – Ver productos ingresados

Muestra los productos ingresados hasta el momento, como muestra a continuación:

```
Ingrese:  
1. Ingresar producto.  
2. Ver productos ingresados.  
3. Confirmar productos.  
4. Eliminar producto.  
0. Descartar productos.  
Elija una opcion:  
2  
TV      x1
```

Opción 3 – Confirma producto

Confirma la orden para la creación de la factura con los productos registrados hasta el momento. Como se muestra a continuación:

```
Ingresa:
1. Ingresar producto.
2. Ver productos ingresados.
3. Confirmar productos.
4. Eliminar producto.
0. Descartar productos.
Elija una opcion:
3
Factura Nro: 100000000
Desde: Riohacha
hasta: Armenia
TV      x1
Estado: Por confirmar
Vendido a:
nombre: testuser
id:     2
correo: j@gmail.com
Hora Salida: 2023-05-25 15:00:00
Hora llegada: 2023-05-26 03:00:00
Costo: $133.350000000000002

Seleccione.
1. Confirmar pedido.
2. Cancelar pedido.
```

Confirmar pedido

Confirma el pedido, genera la factura y almacena los datos relacionados con el usuario. Regresa el menú de bienvenida de usuario

```
Seleccione.
1. Confirmar pedido.
2. Cancelar pedido.

Elija una opcion:
1
Pedido realizado
```

Cancelar pedido

Cancela el pedido y finalizar el proceso. Vuelve al menú de bienvenida de usuario

```
Seleccione.
1. Confirmar pedido.
2. Cancelar pedido.

Elija una opcion:
2

Ingrese:
1. Realizar pedido.
2. Seguir pedido.
3. historial de pedido.
0. salir.
Elija una opcion:
|
```

Opción 4 – Eliminar producto

Se solicita el nombre del producto del cual se quiere eliminar. Al digitar correctamente el nombre, este procederá a eliminarse.

NOTA: Genera un mensaje notificando la eliminación del producto

```
Ingrese:
1. Ingresar producto.
2. Ver productos ingresados.
3. Confirmar productos.
4. Eliminar producto.
0. Descartar productos.
Elija una opcion:
4
nombre del producto:
TV
Producto eliminado
```

MODULO MENU DE BIENVENIDA OPCION 2

```
Bienvenido/a test

Ingrese:
1. Realizar pedido.
2. Seguir pedido.
3. historial de pedido.
0. salir.
Elija una opcion:
```

Opción 2 – Seguir pedido

Al seleccionar esta opción solicita el valor del número de factura. Y muestra en pantalla los datos acerca del pedido.

```
Ingrese:
1. Realizar pedido.
2. Seguir pedido.
3. historial de pedido.
0. salir.
Elija una opcion:
2
Ingrese ID factura:
100000000
Factura Nro: 100000000
Desde: Medellin
hasta: Bogota
Machete x4
Estado: Confirmado
Vendido a:
nombre: testuser
id: 1
correo: test@gmail.com
Hora Salida: 2023-05-25 15:00:00
Hora llegada: 2023-05-25 17:00:00
Costo: $19.35
```

MODULO MENU DE BIENVENIDAD OPCION 3

```
Bienvenido/a test

Ingrese:
1. Realizar pedido.
2. Seguir pedido.
3. historial de pedido.
0. salir.
Elija una opcion:
```

Opción 3 – Historial de pedido

Muestra todas las facturas realizadas por el momento por el usuario.

```
Ingrese:
1. Realizar pedido.
2. Seguir pedido.
3. historial de pedido.
0. salir.
Elija una opcion:
3
Factura Nro: 100000000
Desde: Medellin
hasta: Bogota
Machete x4
Estado: Confirmado
Vendido a:
nombre: testuser
id: 1
correo: test@gmail.com
Hora Salida: 2023-05-25 15:00:00
Hora llegada: 2023-05-25 17:00:00
Costo: $19.35
```

MODULO INGRESAR COMO EMPLEADO

```
-----Bienvenidos a Transportes ltda.-----
Presione:
1. Ingresar como usuario
2. Ingresar como empleado
3. Ingresar como administrador
0. Salir
Elija una opcion:
|
```

Al seleccionar la opción 2. Se obtiene la siguiente vista:

```
Ingrese:
1. iniciar Seccion.
0. salir.
Elija una opcion:
|
```

Opción 1 - iniciar sesión:

Al ingresar el numero 1 nos pedirá los datos de acceso necesario para poder ingresar a la cuenta del empleado registrado en la compañía. Solicitará lo siguiente:

- Usuario/ID: Puedes elegir entre dos maneras de identificarte. Por ID o por nombre de empleado.
- Clave: Contraseña con la que se creó el empleado

NOTA: Los empleados no pueden registrarse por su cuenta. Deben de ser registrados por el usuario administrador.

Información empleados ya creados:

Empleados de Colombia:

nombre	id	contraseña
Juan Garcia	10001	xxxx
Juan Rodriguez	10002	xxxx
Juan Martinez	10003	xxxx
Juan Lopez	10004	xxxx
Juan Perez	10005	xxxx
Juan Gonzalez	10006	xxxx
Maria Garcia	10007	xxxx
Maria Rodriguez	10008	xxxx
Maria Martinez	10009	xxxx
Maria Lopez	10010	xxxx
Maria Perez	10011	xxxx
Maria Gonzalez	10012	xxxx
Pedro Garcia	10013	xxxx
Pedro Rodriguez	10014	xxxx
Pedro Martinez	10015	xxxx
Pedro Lopez	10016	xxxx
Pedro Perez	10017	xxxx
Pedro Gonzalez	10018	xxxx
Ana Garcia	10019	xxxx
Ana Rodriguez	10020	xxxx
AnaMartinez	10021	xxxx
Ana Lopez	10022	xxxx
Ana Perez	10023	xxxx
Ana Gonzalez	10024	xxxx
Luis Garcia	10025	xxxx
Luis Rodriguez	10026	xxxx
Luis Martinez	10027	xxxx
Luis Lopez	10028	xxxx
JLuis Perez	10029	xxxx
Luis Gonzalez	10030	xxxx
Carolina Garcia	10031	xxxx
Carolina Rodriguez	10032	xxxx
Carolina Martinez	10033	xxxx
Carolina Lopez	10034	xxxx
Carolina Perez	10035	xxxx
Carolina Gonzalez	10036	xxxx

Empleados de Ecuador:

nombre	id	contraseña
Andres Hernandez	20001	xxxx
Andres Torres	20002	xxxx
Andres Rizo	20003	xxxx
Laura Hernandez	20004	xxxx
Laura Torres	20005	xxxx
Laura Rizo	20006	xxxx
Carlos Hernandez	20007	xxxx
Carlos Torres	20008	xxxx
Carlos Rizo	20009	xxxx

Sofia Hernandez	20010	xxxx
SofiaTorres	20011	xxxx
Sofia Rizo	20012	xxxx
Javier Hernandez	20013	xxxx
Javier Torres	20014	xxxx
Javier Rizo	20015	xxxx

Empleados de Panamá:

nombre	id	contraseña
Marcela Medina	30001	xxxx
Marcela Vargas	30002	xxxx
Marcela Picon	30003	xxxx
Julian Medina	30004	xxxx
Julian Vargas	30005	xxxx
Julian Picon	30006	xxxx
Camila Medina	30007	xxxx
Camila Vargas	30008	xxxx
Camila Picon	30009	xxxx
Camilo Medina	30010	xxxx
Camilo Vargas	30011	xxxx
Camilo Picon	30012	xxxx

Menú de bienvenida

```

Bienvenido/a Pedro Gonzalez

Ingrese:
1. Mostar estado en la empresa.
2. Cambiar estado en la empresa.
0. salir.
Elija una opcion:

```

Opción 1 - Mostrar estado en la empresa

Muestra en pantalla el estado actual del empleado. Activo – Inactivo.

```

Ingrese:
1. Mostar estado en la empresa.
2. Cambiar estado en la empresa.
0. salir.
Elija una opcion:
1
Estado: Activo

```

Opción 2 – Cambiar estado en la empresa

Cambia el estado del empleado y muestra en pantalla el estado actualizado. Además, de un mensaje mencionando el cambio.

```
Ingrese:
1. Mostar estado en la empresa.
2. Cambiar estado en la empresa.
0. salir.
Elija una opcion:
2
Estado: Inactivo
Tu estado en la empresa ha cambiado.
```

MODULO INGRESAR COMO ADMINISTRADOR

```
-----Bienvenidos a Transportes ltda.-----
```

```
Presione:
1. Ingresar como usuario
2. Ingresar como empleado
3. Ingresar como administrador
0. Salir
Elija una opcion:
|
```

Opción 1 – Ingresar como administrador

```
Ingrese:
1. ingresar como administrador.
0. salir.
Elija una opcion:
|
```

Al ingresar el numero 1 nos pedirá los datos de acceso necesario para poder ingresar a la cuenta administrador. Solicitará lo siguiente:

- Clave: Contraseña del usuario administrador

NOTA: La contraseña del usuario administrador es “Clave”

```
Ingrese:
1. ingresar como administrador.
0. salir.
Elija una opcion:
1
Clave:
Clave
```

Menú de bienvenida

```
Ingresando como administrador...

Ingrese:
1. Registrar nuevo empleado.
2. Registrar nuevo vehiculo.
3. Facturas.
4. Vehiculos.
5. Usuarios.
6. Empleados.
0. salir.
Elija una opcion:
```

Opción 1 – registrar nuevo empleado

Si seleccionas esta opción podrás registrar un nuevo empleado. Primero veras un submenú para elegir el país en donde vas a crear el empleado.

```
Ingrese:
1. Colombia.
2. Ecuador.
3. Panama.
0. Salir.
Elija una opcion:
```

Luego, se te solicitara los siguientes parámetros:

- Nombre: Nombre del empleado
- ID: Id del empleado
- Correo: Correo del empleado
- Clave: Contraseña del empleado
- Ciudad actual del empleado: Ciudad en donde se encuentra el empleado

NOTA: La ciudad actual del empleado debe de digitarse tal cual y como esta en la lista de ciudades del país seleccionado

```
Ingrese:
1. Registrar nuevo empleado.
2. Registrar nuevo vehiculo.
3. Facturas.
4. Vehiculos.
5. Usuarios.
6. Empleados.
0. salir.
```

Elija una opcion:

1

```
Ingrese:
1. Colombia.
2. Ecuador.
3. Panama.
0. Salir.
```

Elija una opcion:

1

nombre:

test

ID:

1023

correo:

a@gmail.com

Clave:

1

ciudad actual del empleado:

Medellin

Nuevo empleado agregado

Opción 2 – registrar nuevo vehículo

Si seleccionas esta opción podrás registrar un nuevo vehículo. Primero veras un submenú para elegir el país en donde vas a crear el vehiculo.

```
Ingrese:
1. Registrar nuevo empleado.
2. Registrar nuevo vehiculo.
3. Facturas.
4. Vehiculos.
5. Usuarios.
6. Empleados.
0. salir.
Elija una opcion:
2
```

```
Ingrese:
1. Colombia.
2. Ecuador.
3. Panama.
0. Salir.
Elija una opcion:
|
```

Luego, se te solicitara los siguientes parámetros:

- Placa: Placa del camión
- Tipo de carga: Se mostrará un submenú con los tipos de cargas
- Ciudad actual del camión: Ciudad en la que se encuentra el vehículo
- Tipo de vehículo: Se mostrará un submenú con el tipo de vehículo

Ingrese:

1. Registrar nuevo empleado.
2. Registrar nuevo vehiculo.
3. Facturas.
4. Vehiculos.
5. Usuarios.
6. Empleados.
0. salir.

Elija una opcion:

2

Ingrese:

1. Colombia.
2. Ecuador.
3. Panama.
0. Salir.

Elija una opcion:

1

Placa:

DJF231

Vehiculo tipo:

1. Liviano 1tn.
2. Mediano 8tn.
3. Semipesado 17tn.
4. Pesado 24tn.
0. cancelar.

Elija una opcion:

3

ciudad actual del Camion:

Medellin

Tipo de vehiculo:

1. Cisterna.
2. Frigorifico.
3. Lona.
4. PortaCoches.
0. Cancelar.

Elija una opcion:

1

Nuevo camion agregado

Opción 3 – Facturas

Si seleccionas esta opción podrás listar las facturas. Primero veras un submenú para elegir el país en donde vas a ver todas las facturas:

```
Ingrese:
1. Registrar nuevo empleado.
2. Registrar nuevo vehiculo.
3. Facturas.
4. Vehiculos.
5. Usuarios.
6. Empleados.
0. salir.
Elija una opcion:
3

Ingrese:
1. Colombia.
2. Ecuador.
3. Panama.
0. Salir.
Elija una opcion:
1
Factura Nro: 1000000000
Desde: Medellin
hasta: Bogota
Machete x4
Estado: Confirmado
Vendido a:
nombre: testuser
id: 1
correo: test@gmail.com
Hora Salida: 2023-05-25 15:00:00
Hora llegada: 2023-05-25 17:00:00
Costo: $19.35
```


Opción 4 – Vehículos

Si seleccionas esta opción podrás listar los vehículos. Primero veras un submenú para elegir el país en donde vas a ver todos los vehículos de ese lugar. Luego, otro submenú para seleccionar los tipos de camión que deseas visualizar.

```
Ingrese:
1. Registrar nuevo empleado.
2. Registrar nuevo vehiculo.
3. Facturas.
4. Vehiculos.
5. Usuarios.
6. Empleados.
0. salir.
Elija una opcion:
4
```

```
Ingrese:
1. Colombia.
2. Ecuador.
3. Panama.
0. Salir.
Elija una opcion:
1
```

```
Tipo de vehiculo:
1. Cisterna.
2. Frigorifico.
3. Lona.
4. PortaCoches.
0. Cancelar.
```

```
Elija una opcion:
1
```

```
Tipo: Cisterna
Placa: ABC123
pais: Colombia
Ciudad actual: Riohacha
Peso Maximo: 1.0
Capacidad: 20.0
Disponible: true
```

```
Tipo: Cisterna
Placa: DEF456
pais: Colombia
Ciudad actual: Valledupar
Peso Maximo: 8.0
Capacidad: 35.0
Disponible: true
```

```
Tipo: Cisterna
Placa: GHI789
pais: Colombia
```

Opción 5 – Usuarios

Si seleccionas esta opción podrás listar los usuarios registrados hasta el momento

```
Ingrese:
1. Registrar nuevo empleado.
2. Registrar nuevo vehiculo.
3. Facturas.
4. Vehiculos.
5. Usuarios.
6. Empleados.
0. salir.
Elija una opcion:
5
```

```
nombre: Juan
id:      1001
correo:  juan@example.com
```

```
nombre: Maria
id:      1002
correo:  maria@example.com
```

```
nombre: Pedro
id:      1003
correo:  pedro@example.com
```

```
nombre: Laura
id:      1004
correo:  laura@example.com
```

```
nombre: Carlos
id:      1005
correo:  carlos@example.com
```

```
nombre: Ana
id:      1006
correo:  ana@example.com
```

```
nombre: Luis
id:      1007
correo:  luis@example.com
```

```
nombre: Marta
id:      1008
correo:  marta@example.com
```

Opción 6 – Empleados

Si seleccionas esta opción podrás listar los empleados. Primero veras un submenú para elegir el país en donde se encuentren tus empleados:

```
Ingrese:
1. Registrar nuevo empleado.
2. Registrar nuevo vehiculo.
3. Facturas.
4. Vehiculos.
5. Usuarios.
6. Empleados.
0. salir.
Elija una opcion:
6

Ingrese:
1. Colombia.
2. Ecuador.
3. Panama.
0. Salir.
Elija una opcion:
1

nombre: Juan Garcia
id: 10001
correo: JuanGarcia@example.com
Pais:Colombia
Ciudad Actual: Riohacha
Activo: true
Disponible: true
Pais Colombia

nombre: Juan Rodriguez
id: 10002
correo: JuanRodriguez@example.com
Pais:Colombia
Ciudad Actual: Valledupar
Activo: true
Disponible: true
Pais Colombia

nombre: Juan Martinez
id: 10003
correo: JuanMartinez@example.com
Pais:Colombia
Ciudad Actual: Barranquilla
Activo: true
Disponible: true
Pais Colombia

nombre: Juan Lopez
id: 10004
```