
	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Marzo 2020 – Julio 2020

		<b>FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES</b>	
<b>CARRERA:</b> COMPUTACIÓN/INGENIERÍA DE SISTEMAS		<b>ASIGNATURA:</b> PROGRAMACIÓN APLICADA	
<b>NRO. PROYECTO:</b>	1.1	<b>TÍTULO PROYECTO:</b> Proyecto Integrador Inter ciclo Desarrollo e implementación de un sistema de gestión de datos de un parqueadero para la empresa EMOV-EC con persistencia de datos en PostgreSQL	
<b>OBJETIVO:</b> Reforzar los conocimientos adquiridos en clase sobre la programación aplicada (POO, Interfaz gráfica, jpa, etc.) en un contexto real.			
<b>INSTRUCCIONES:</b>		1. Revisar el contenido teórico y práctico del tema	
		2. Profundizar los conocimientos revisando los libros guías, los enlaces contenidos en los objetos de aprendizaje Java y la documentación disponible en fuentes académicas en línea.	
		3. Deberá desarrollar un sistema informático para la gestión de parqueaderos con persistencia de datos JPA.	
		4. Deberá generar un informe empleando una herramienta Web 2.0 o Prezi (Tutorial o manual técnico).	
		5. Tomar en consideración que la evaluación del trabajo a realizarse de forma <b>individual</b> y dependerá de los siguientes parámetros: Nivel de precisión, mejoramiento y explicación de la propuesta planteada del sistema informático. <b>50%</b> Tutorial o manual técnico del sistema <b>25%</b> (Pagina Web o Prezi) Exposición, funcionamiento, base de datos y validación del sistema <b>25%</b> . <b>Puntos extras:</b> Funcionalidad o librería no vista en clase serán valorados como puntos adicionales al Inter ciclo.	
		6. <b>Fecha de entrega:</b> El sistema debe ser subido al avac y presentado el día lunes <b>08 de febrero del 2021</b> .	
<b>ACTIVIDADES POR DESARROLLAR</b>			

1. Investigue, diseñe y desarrolle e implemente un sistema informático que permita gestionar los espacios de parqueadero de la empresa EMOV-EC del cancho del parque de la Madre que tiene una capacidad de 50 estacionamientos, esta información se debe ser almacenada dentro de una base de datos (PostgreSQL), finalmente desarrollar una simulación del ingreso del carro al parqueadero utilizando hilos.

#### **DEFINICIÓN DEL PROBLEMA:**

El ofrecer una atención cordial y eficiente a sus clientes es un objetivo de vital importancia para una empresa de este tipo. No es tarea fácil gestionar espacios de un estacionamiento ya que se debe de tomar en cuenta diferentes factores como son: el número de espacios con que se cuenta, los espacios que ya se encuentran ocupados en la actualidad, así como las que ya han sido reservadas para una fecha determinada, también hay que tomar en cuenta los servicios que se han contratado para cada vehículo y el precio que esta presenta.

#### **OBJETIVOS DEL SISTEMA:**

El sistema debe de ser desarrollado en el modo monousuario para que pueda:

- 1.3.1. Establecer usuarios en el sistema con dos niveles de operación: Administrador (Todas las operaciones) y Usuario simple (Solo gestiona el parqueadero), con ello mantiene y distingue las posibilidades de operación del usuario correspondiente.
- 2.3.2. Hacer el ingreso y egreso de vehículos desde un solo punto o puesto emitiendo su comprobante de entrada (para el cliente) y luego el de cobro.
- 3.3.3. Hacer ingreso y egreso de vehículos que se estacionan por un determinado tiempo (horario fijo) o que utilizan el espacio mediante un contrato de arrendamiento preestablecido emitiendo el correspondiente recibo de entrada (registro) y salida (cobro) si se quiere.
- 4.3.4. Emitir diversos reportes de ingresos y espacios disponibles.
- 5.3.1 Gestionar la base de datos PostgreSQL
- 6.3.1 Aplicar técnicas de persistencia de datos JPA para el acceso y manipulación de la información.
- 7.3.1 Realizar procesos de simulación utilizando Hilos (Thread).


#### **ESPECIFICACIÓN DE REQUERIMIENTOS:**

##### **2.1 Requerimientos No Funcionales**

- 2.1.1. Aprendizaje: El sistema debe permitir el aprendizaje fluido del usuario.
- 2.1.2. Facilidad de uso: El sistema debe poseer una interfaz visual para facilidad del usuario final.

##### **2.2 Requerimientos Funcionales:**

- 2.2.1. Hacer contrato de espacio: El sistema debe gestionar la información correspondiente a las reservas del estacionamiento.
- 2.2.2. Realizar las entradas y salidas de los vehículos: ya sea de vehículos con control de tiempo o de vehículos con control de espacio.
- 2.2.3. Consultar Importe total: El sistema calculará la cuenta total del cliente por los servicios prestados, se deberá tener una tabla para gestionar el valor por hora del parqueadero permitiendo asignar descuentos a clientes.
- 2.2.4. Consultar el Precio de cada espacio: El sistema deberá registrar y mostrar el precio de los espacios disponibles.
- 2.2.5. Ver listado de espacios disponibles: El sistema deberá mostrar la lista de espacios disponibles con que cuenta el estacionamiento de modo **gráfico**.
- 2.2.6 Permitirá consultar si alguno de los espacios contratados cuenta con algún tipo de servicio de arrendamiento o multa (Después de una semana de no pago se debe calcular multiplicando el valor de la deuda por 10%).
- 2.2.7 Se debe generar una simulación de la entrada y salida de vehículos.

	<b>Computación</b>	<b>Docente: Diego Quisi Peralta</b>
	Programación Aplicada	<b>Período Lectivo:</b> Marzo 2020 – Julio 2020

## 2. Tutorial técnico del uso (Manual técnico):

- Generar una página web o presentación que contenga lo siguiente:
  - Planteamiento y descripción del problema.
  - Proceso de solución.
  - Diagramas de Clases.
  - Diagrama de Base de Datos
  - Arquitectura del sistema.
  - Descripción de la solución y pasos seguidos.
  - Tutorial del uso del sistema (básico).
  - Requerimientos de HW y SF (Java).
  - Conclusiones y recomendaciones.
  - Resultados.

### **RESULTADO(S) OBTENIDO(S):**

Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.  
Identifica correctamente qué herramientas de programación se pueden aplicar.

### **CONCLUSIONES:**

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
- Los estudiantes implementan soluciones graficas en sistemas.
- Los estudiantes están en la capacidad de implementar la persistencia en base de datos.

### **RECOMENDACIONES:**

- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.
- **Consultar con el docente las dudas que puedan surgir al momento de realizar la práctica.**

### **BIBLIOGRAFIA:**

[1]: <https://www.ups.edu.ec/evento?calendarBookingId=98892>

**Docente / Técnico Docente:** Ing. Diego Quisi Peralta Msc.

**Firma:** \_\_\_\_\_

**CARRERA:** Computation

**ASIGNATURA:** Programacion Aplicada

**NRO. PRÁCTICA:** 01 **TÍTULO PRÁCTICA:** Proyecto Final de ciclo

**OBJETIVO ALCANZADO:**

- Reforzar los conocimientos adquiridos en clase sobre la programación aplicada (POO, Interfaz gráfica, jpa, etc.) en un contexto real.

**ACTIVIDADES DESARROLLADAS**

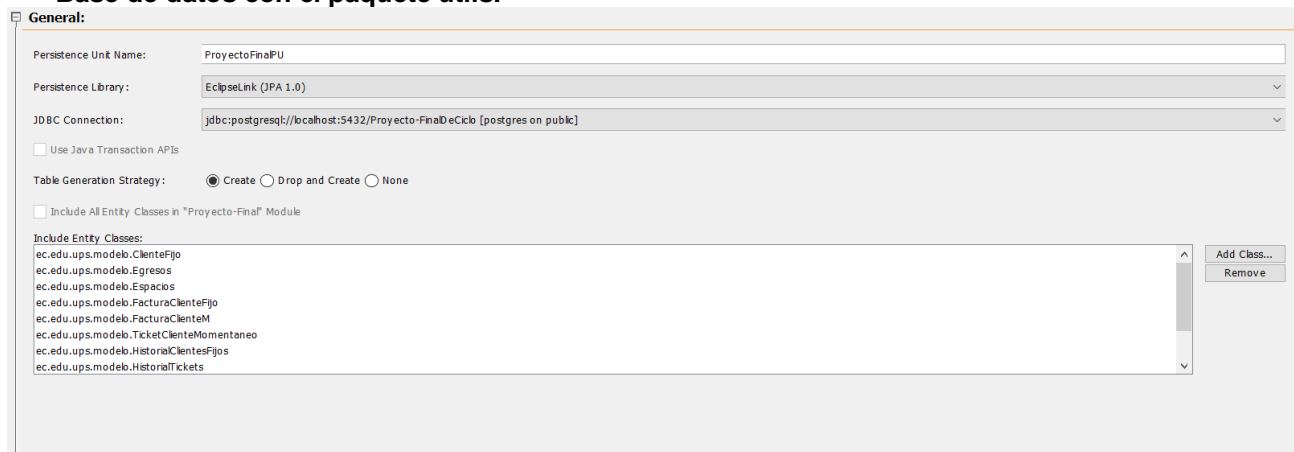
**1- Planteamiento y descripción del problema.**

El ofrecer una atención cordial y eficiente a sus clientes es un objetivo de vital importancia para una empresa de este tipo. No es tarea fácil gestionar espacios de un estacionamiento ya que se debe de tomar en cuenta diferentes factores como son: el número de espacios con que se cuenta, los espacios que ya se encuentran ocupados en la actualidad, así como las que ya han sido reservadas para una fecha determinada, también hay que tomar en cuenta los servicios que se han contratado para cada vehículo y el precio que esta presenta.

**2- Proceso de solucion.**

Para resolver el siguiente problema debemos ayudarnos de una arquitectura y usaremos la MVC, también nos ayudaremos de la base de datos de Postgress.

**- Base de datos con el paquete utils.**



**JPAUTILS**

```
package ec.edu.ups.utils;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author paul_
 */
public class JPAUtils {

    private static final EntityManagerFactory emf = Persistence.createEntityManagerFactory("ProyectoFinalPU");

    public static EntityManager getEntityManager () {
        return emf.createEntityManager ();
    }
}
```

**- Ec.edu.ups.modelo**

Cliente Fijo

```
package ec.edu.ups.modelo;

import java.io.Serializable;
import java.util.Calendar;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Temporal;

/**
 *
 * @author paul_
 */
@Entity
public class ClienteFijo implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    @Column (name="cedula")
    private String cedula;
    @Column (name="nombre")
    private String nombre;
    @Column (name="apellido")
    private String apellido;
    @Column (name="direccion")
    private String direccion;
    @Column (name="tlf")
    private String tlf;
    @Column (name="tipoVehiculo")
    private String tipoVehiculo;
    @Column (name="tipoTarifa")
    private String tipoTarifa;
    @Column (name="abono")
    private Double abono;
    @Column (name="fechaExpiracion")
    @Temporal(javax.persistence.TemporalType.DATE)
    private Calendar fechaExpiracion;
    @Column (name="espacioParqueo")
    private int espacioParqueo;
    @Column (name="multa")
    private double multa;
    @Column (name="estado")
    private boolean estado;

    public String getCedula() {
        return cedula;
    }

    public void setCedula(String cedula) {
        this.cedula = cedula;
    }

    public String getNombre() {
        return nombre;
    }
}
```

```
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

public String getDireccion() {
    return direccion;
}

public void setDireccion(String direccion) {
    this.direccion = direccion;
}

public String getTlf() {
    return tlf;
}

public void setTlf(String tlf) {
    this.tlf = tlf;
}

public String getTipoVehiculo() {
    return tipoVehiculo;
}

public void setTipoVehiculo(String tipoVehiculo) {
    this.tipoVehiculo = tipoVehiculo;
}

public String getTipoTarifa() {
    return tipoTarifa;
}

public void setTipoTarifa(String tipoTarifa) {
    this.tipoTarifa = tipoTarifa;
}

public Double getAbono() {
    return abono;
}

public void setAbono(Double abono) {
    this.abono = abono;
}

public Calendar getFechaExpiracion() {
    return fechaExpiracion;
}

public void setFechaExpiracion(Calendar fechaExpiracion) {
    this.fechaExpiracion = fechaExpiracion;
}
}
```

```
public int getEspacioParqueo() {
    return espacioParqueo;
}

public void setEspacioParqueo(int espacioParqueo) {
    this.espacioParqueo = espacioParqueo;
}

public double getMulta() {
    return multa;
}

public void setMulta(double multa) {
    this.multa = multa;
}

public boolean isEstado() {
    return estado;
}

public void setEstado(boolean estado) {
    this.estado = estado;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not
set
    if (!(object instanceof ClienteFijo)) {
        return false;
    }
    ClienteFijo other = (ClienteFijo) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "ec.edu.ups.modelo.ClienteFijo[ id=" + id + " ]";
}
}
```

## Egresos

```
package ec.edu.upson.modelo;

import java.io.Serializable;
import java.util.Calendar;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Temporal;

/**
 *
 * @author paul_
 */
@Entity
public class Egresos implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    @Column (name="fechaHora")
    @Temporal (javax.persistence.TemporalType.DATE)
    private Calendar fechaHora;
    @Column (name="descripcion")
    private String descripcion;
    @Column (name="egreso")
    private double egreso;

    public Calendar getFechaHora() {
        return fechaHora;
    }

    public void setFechaHora(Calendar fechaHora) {
        this.fechaHora = fechaHora;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

    public double getEgreso() {
        return egreso;
    }

    public void setEgreso(double egreso) {
        this.egreso = egreso;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
```



```

        this.id = id;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not
set
        if (!(object instanceof Egresos)) {
            return false;
        }
        Egresos other = (Egresos) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "ec.edu.ups.modelo.Egresos[ id=" + id + " ]";
    }
}

Espacios
package ec.edu.ups.modelo;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

/**
 *
 * @author paul_
 */
@Entity
public class Espacios implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    @Column (name="nombre")
    private String nombre;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {

```

```

        this.nombre = nombre;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not
set
        if (!(object instanceof Espacios)) {
            return false;
        }
        Espacios other = (Espacios) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "ec.edu.ups.modelo.Espacios[ id=" + id + " ]";
    }
}

```

#### Factura Cliente Fijo

```

package ec.edu.ups.modelo;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;

/**
 *
 * @author paul_
 */
@Entity
public class FacturaClienteFijo implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)

```

```
private Integer id;

@Column(name="descripcion")
private String descripcion;
@JoinColumn(name="clienteFijo")
@OneToOne
private ClienteFijo clientefijo;

public String getDescripcion() {
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}

public ClienteFijo getClientefijo() {
    return clientefijo;
}

public void setClientefijo(ClienteFijo clientefijo) {
    this.clientefijo = clientefijo;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not
set
    if (!(object instanceof FacturaClienteFijo)) {
        return false;
    }
    FacturaClienteFijo other = (FacturaClienteFijo) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "ec.edu.ups.modelo.FacturaClienteFijo[ id=" + id + " ]";
}
}
```

## Factura Cliente Momentaneo

```
package ec.edu.upse.modelo;

import java.io.Serializable;
import java.util.Calendar;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Temporal;

/**
 *
 * @author paul_
 */
@Entity
public class FacturaClienteM implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    @Column (name="fecha")
    @Temporal (javax.persistence.TemporalType.DATE)
    private Calendar Fecha;
    @Column (name="nombres")
    private String nombres;
    @Column (name="direccion")
    private String direccion;
    @Column (name="cedula")
    private String cedula;
    @Column (name="telefono")
    private String telefono;
    @JoinColumn (name="ticket")
    @OneToOne
    private TicketClienteMomentaneo ticket;

    public Calendar getFecha() {
        return Fecha;
    }

    public void setFecha(Calendar Fecha) {
        this.Fecha = Fecha;
    }

    public String getNombres() {
        return nombres;
    }

    public void setNombres(String nombres) {
        this.nombres = nombres;
    }

    public String getDireccion() {
        return direccion;
    }
}
```

```
public void setDireccion(String direccion) {
    this.direccion = direccion;
}

public String getCedula() {
    return cedula;
}

public void setCedula(String cedula) {
    this.cedula = cedula;
}

public String getTelefono() {
    return telefono;
}

public void setTelefono(String telefono) {
    this.telefono = telefono;
}

public TicketClienteMomentaneo getTicket() {
    return ticket;
}

public void setTicket(TicketClienteMomentaneo ticket) {
    this.ticket = ticket;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not
    set
    if (!(object instanceof FacturaClienteM)) {
        return false;
    }
    FacturaClienteM other = (FacturaClienteM) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
    !this.id.equals(other.id))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "ec.edu.ups.modelo.FacturaClienteM[ id=" + id + " ]";
}
```

```

    }

}

    Historial Cliente Fijo
package ec.edu.upse.modelo;

import java.io.Serializable;
import java.util.Calendar;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Temporal;

/**
 *
 * @author paul_
 */
@Entity
public class HistorialClientesFijos implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    @Column (name="fechaHora")
    @Temporal(javax.persistence.TemporalType.DATE)
    private Calendar FechaHora;
    @Column (name="descripcion")
    private String descripcion;
    @JoinColumn (name="clienteF")
    @OneToOne
    private ClienteFijo clienteF;

    public Calendar getFechaHora() {
        return FechaHora;
    }

    public void setFechaHora(Calendar FechaHora) {
        this.FechaHora = FechaHora;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

    public ClienteFijo getClienteF() {
        return clienteF;
    }

    public void setClienteF(ClienteFijo clienteF) {
        this.clienteF = clienteF;
    }
}

```

```
public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not
set
    if (!(object instanceof HistorialClientesFijos)) {
        return false;
    }
    HistorialClientesFijos other = (HistorialClientesFijos) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "ec.edu.ups.modelo.HistorialClientesFijos[ id=" + id + " ]";
}
}

Historial Cliente Momentaneo
package ec.edu.ups.modelo;

import java.io.Serializable;
import java.util.Calendar;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Temporal;

/**
 *
 * @author paul_
 */
@Entity
public class HistorialTickets implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```

private Integer id;

@Column (name="FechaHora")
@Temporal(javax.persistence.TemporalType.DATE)
private Calendar FechaHora;
@Column (name="descripcion")
private String descripcion;
@JoinColumn (name="ticketClienteM")
@OneToOne
private TicketClienteMomentaneo clienteM;

public Calendar getFechaHora() {
    return FechaHora;
}

public void setFechaHora(Calendar FechaHora) {
    this.FechaHora = FechaHora;
}

public String getDescripcion() {
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}

public TicketClienteMomentaneo getClienteM() {
    return clienteM;
}

public void setClienteM(TicketClienteMomentaneo clienteM) {
    this.clienteM = clienteM;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not
set
    if (!(object instanceof HistorialTickets)) {
        return false;
    }
    HistorialTickets other = (HistorialTickets) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
    }

```



```
    }  
    return true;  
}  
  
@Override  
public String toString() {  
    return "ec.edu.ups.modelo.HistorialTickets[ id=" + id + " ]";  
}  
  
}  
  
Ingresos  
package ec.edu.ups.modelo;  
  
import java.io.Serializable;  
import java.util.Calendar;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.Temporal;  
  
/**  
 *  
 * @author paul_  
 */  
@Entity  
public class Ingresos implements Serializable {  
  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private Integer id;  
  
    @Column (name="fechaHora")  
    @Temporal(javax.persistence.TemporalType.DATE)  
    private Calendar fechaHora;  
  
    @Column (name="desccripcion")  
    private String descripcion;  
  
    @Column (name="ingreso")  
    private double ingreso;  
  
    public Calendar getFechaHora() {  
        return fechaHora;  
    }  
  
    public void setFechaHora(Calendar fechaHora) {  
        this.fechaHora = fechaHora;  
    }  
  
    public String getDescripcion() {  
        return descripcion;  
    }  
  
    public void setDescripcion(String descripcion) {  
        this.descripcion = descripcion;  
    }  
  
    public double getIngreso() {
```

```

        return ingreso;
    }

    public void setIngreso(double ingreso) {
        this.ingreso = ingreso;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not
set
        if (!(object instanceof Ingresos)) {
            return false;
        }
        Ingresos other = (Ingresos) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "ec.edu.ups.modelo.Ingresos[ id=" + id + " ]";
    }
}

Tarifa
package ec.edu.ups.modelo;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

/**
 *
 * @author paul_
 */
@Entity
public class Tarifa implements Serializable {

    private static final long serialVersionUID = 1L;

```

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Integer id;

@Column (name="mediaHoraM")
private double mediaHoraM;
@Column (name="horaM")
private double horaM;
@Column (name=" mensualM")
private double mensualM;
@Column (name="mediaHoraA")
private double mediaHoraA;
@Column (name="horaA")
private double horaA;
@Column (name="mensualA")
private double mensualA;
@Column (name="mediaHoraP")
//Pesados
private double mediaHoraP;
@Column (name="horaP")
private double horaP;
@Column (name="mensualP")
private double mensualP;

public double getMediaHoraM() {
    return mediaHoraM;
}

public void setMediaHoraM(double mediaHoraM) {
    this.mediaHoraM = mediaHoraM;
}

public double getHoraM() {
    return horaM;
}

public void setHoraM(double horaM) {
    this.horaM = horaM;
}

public double getMensualM() {
    return mensualM;
}

public void setMensualM(double mensualM) {
    this.mensualM = mensualM;
}

public double getMediaHoraA() {
    return mediaHoraA;
}

public void setMediaHoraA(double mediaHoraA) {
    this.mediaHoraA = mediaHoraA;
}

public double getHoraA() {
    return horaA;
}

public void setHoraA(double horaA) {
```

```

        this.horaA = horaA;
    }

    public double getMensualA() {
        return mensualA;
    }

    public void setMensualA(double mensualA) {
        this.mensualA = mensualA;
    }

    public double getMediaHoraP() {
        return mediaHoraP;
    }

    public void setMediaHoraP(double mediaHoraP) {
        this.mediaHoraP = mediaHoraP;
    }

    public double getHoraP() {
        return horaP;
    }

    public void setHoraP(double horaP) {
        this.horaP = horaP;
    }

    public double getMensualP() {
        return mensualP;
    }

    public void setMensualP(double mensualP) {
        this.mensualP = mensualP;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not
set
        if (!(object instanceof Tarifa)) {
            return false;
        }
        Tarifa other = (Tarifa) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
            return false;

```

```
    }  
    return true;  
}  
  
@Override  
public String toString() {  
    return "ec.edu.ups.modelo.Tarifa[ id=" + id + " ]";  
}  
  
}  
  
Usuario  
package ec.edu.ups.modelo;  
  
import java.io.Serializable;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
  
/**  
 *  
 * @author paul_  
 */  
@Entity  
public class Usuario implements Serializable {  
  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private Integer id;  
  
    @Column (name="cedula")  
    private String cedula;  
    @Column (name="nombre")  
    private String nombre;  
    @Column (name="apellido")  
    private String apellido;  
    @Column (name="direccion")  
    private String direccion;  
    @Column (name="tlf")  
    private String tlf;  
    @Column (name="rol")  
    private String rol;  
    @Column (name="correo")  
    private String correo;  
    @Column (name="pass")  
    private String pass;  
  
    public String getCedula() {  
        return cedula;  
    }  
  
    public void setCedula(String cedula) {  
        this.cedula = cedula;  
    }  
  
    public String getNombre() {  
        return nombre;  
    }  
}
```

```
public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

public String getDireccion() {
    return direccion;
}

public void setDireccion(String direccion) {
    this.direccion = direccion;
}

public String getTlf() {
    return tlf;
}

public void setTlf(String tlf) {
    this.tlf = tlf;
}

public String getRol() {
    return rol;
}

public void setRol(String rol) {
    this.rol = rol;
}

public String getCorreo() {
    return correo;
}

public void setCorreo(String correo) {
    this.correo = correo;
}

public String getPass() {
    return pass;
}

public void setPass(String pass) {
    this.pass = pass;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}
```

@Override

```

public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not
set
    if (!(object instanceof Usuario)) {
        return false;
    }
    Usuario other = (Usuario) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "ec.edu.ups.modelo.Usuario[ id=" + id + " ]";
}
}

- Ec.edu.ups.controlador
Abstract Controlador
package ec.edu.ups.controlador;

import ec.edu.ups.utils.JPAUtils;
import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.persistence.EntityManager;

/**
 *
 * @author paul_
 * @param <E>
 */
public abstract class AbstractControlador<E> {

    private List<E> lista;
    private Class<E> clase;
    private EntityManager em;

    public AbstractControlador() {
        lista= new ArrayList<>();
        Type t = getClass().getGenericSuperclass();
        ParameterizedType pt =(ParameterizedType) t;
        clase= (Class) pt.getActualTypeArguments()[0];
        em=JPAUtils.getEntityManager();
    }
}

```

```

public boolean crear (E objeto){
    try {
        if(this.validar(objeto)){
            em.getTransaction().begin();
            em.persist(objeto);
            em.getTransaction().commit();
            lista.add(objeto);
            return true;
        } catch (Exception ex) {
            Logger.getLogger(AbstractControlador.class.getName()).log(Level.SEVERE,
null, ex);
        }

        return false;
    }

    public boolean eliminar (E objeto){
        em.getTransaction().begin();
        em.remove(em.merge(objeto));
        em.getTransaction().commit();
        lista.remove(objeto);
        return true;
    }

    public boolean actualizar (E objeto){
        try {
            if(this.validar(objeto)){
                em.getTransaction().begin();
                objeto=em.merge(objeto);
                em.getTransaction().commit();
                this.lista=buscarTodo();
                return true;
            }
        } catch (Exception ex) {
            Logger.getLogger(AbstractControlador.class.getName()).log(Level.SEVERE,
null, ex);
        }

        return false;
    }

    public E buscar (Object id){
        return (E) em.find(clase, id);
    }

    public List<E> buscarTodo () {

        return em.createQuery("Select t from "+ clase.getSimpleName()+ "
t").getResultList();
    }

    public abstract boolean validar(E objeto) throws Exception;

    public List<E> getLista() {
        return lista;
    }

    public void setLista(List<E> lista) {

```



```
        this.lista = lista;
    }

    public Class<E> getClase() {
        return clase;
    }

    public void setClase(Class<E> clase) {
        this.clase = clase;
    }

    public EntityManager getEm() {
        return em;
    }

    public void setEm(EntityManager em) {
        this.em = em;
    }
}

Controlador Cliente Fijo
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.ClienteFijo;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author paul_
 */
public class ControladorClienteFijo extends AbstractControlador<ClienteFijo> {

    @Override
    public boolean validar(ClienteFijo objeto) {
        return true;
    }

    public List<ClienteFijo> clientesFijos() {

        List<ClienteFijo> lista = new ArrayList();
        ClienteFijo u;
        Iterator i = super.getList().iterator();
        while (i.hasNext()) {
            u = (ClienteFijo) i.next();
            lista.add(u);
        }
        return lista;
    }
}
```

```

    }

    public double generarMulta(ClienteFijo cliente){

        double valorMulta=0;

        Calendar fechaExpiracion =cliente.getFechaExpiracion();
        Calendar fechaActual = new GregorianCalendar();
        int dias=-1;

        while(fechaExpiracion.before(fechaActual)|| fechaExpiracion.equals(fechaActual)){
            dias++;
            fechaExpiracion.add(Calendar.DATE,dias);
        }
        //si han transcurrido 7 dias ( una semana) de la fecha, se le aumenta un 10%
        al bono a pagar
        if(dias==7){
            valorMulta= cliente.getAbono()+(cliente.getAbono()*0.1);
        }
        return valorMulta;

    }

    public ClienteFijo buscarCliente(String cedula){

        ClienteFijo u;
        Iterator i = super.buscarTodo().iterator();
        while (i.hasNext()) {
            u = (ClienteFijo) i.next();
            if(u.getCedula().trim().equals(cedula.trim())){
                return u;
            }

        }
        return null;

    }

}

Controlador Egreso
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Egresos;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author paul_
 */
public class ControladorEgreso extends AbstractControlador<Egresos> {

    @Override
    public boolean validar(Egresos objeto) {
        return true;
    }

}

```

```
public List<Egresos> egresos() {  
  
    List<Egresos> lista = new ArrayList();  
    Egresos u;  
    Iterator i = super.getList().iterator();  
    while (i.hasNext()) {  
        u = (Egresos) i.next();  
        lista.add(u);  
    }  
    return lista;  
}  
}
```

### Controlador Espacios

```
package ec.edu.ups.controlador;  
  
import java.util.ArrayList;  
import ec.edu.ups.modelo.Espacios;  
import java.util.List;  
  
/**  
 *  
 * @author paul_  
 */  
public class ControladorEspacios extends AbstractControlador<Espacios> {  
  
    @Override  
    public boolean validar(Espacios objeto) {  
        return true;  
    }  
  
    public boolean AsignarEspacio(int id, String cedula){  
  
        Espacios e = new Espacios();  
  
        e=super.buscar(id);  
  
        e.setNombre(cedula);  
  
        return super.actualizar(e);  
    }  
  
    public List<Espacios> espaciosDisponibles() {  
        List<Espacios> esp = new ArrayList();  
        for(Espacios e : super.buscarTodo()){  
            if(e.getNombre()==null){  
                esp.add(e);  
            }  
        }  
    }  
}
```

```

        return esp;

    }

}

Controlador Facturas Cliente Fijo
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.FacturaClienteFijo;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author paul_
 */
public class ControladorFacturaClienteFijo extends AbstractControlador<FacturaClienteFijo> {

    @Override
    public boolean validar(FacturaClienteFijo objeto) {
        return true;
    }

}

Controlador Facturas Cliente Momentaneo
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.FacturaClienteM;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author user
 */
public class ControladorFacturaClienteM extends AbstractControlador<FacturaClienteM>
{

    @Override
    public boolean validar(FacturaClienteM objeto) {
        return true;
    }

}

```

## Controlador Historial Cliente Fijo

```
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.HistorialTickets;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author paul_
 */
public class ControladorHistorialClienteM extends AbstractControlador<HistorialTickets> {

    @Override
    public boolean validar(HistorialTickets objeto) {
        return true;
    }

    public List<HistorialTickets> listaHistorialCM() {

        List<HistorialTickets> lista = new ArrayList();
        HistorialTickets u;
        Iterator i = super.getList().iterator();
        while (i.hasNext()) {
            u = (HistorialTickets) i.next();
            lista.add(u);
        }
        return lista;
    }
}
```

## Controlador Historial Cliente Momentaneo

```
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.HistorialClientesFijos;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author paul_
 */
public class ControladorHistorialClientesF extends AbstractControlador<HistorialClientesFijos> {
```

```

@Override
public boolean validar(HistorialClientesFijos objeto) {
    return true;
}

public List<HistorialClientesFijos> listaHistorialCF() {

    List<HistorialClientesFijos> lista = new ArrayList();
    HistorialClientesFijos u;
    Iterator i = super.getList().iterator();
    while (i.hasNext()) {
        u = (HistorialClientesFijos) i.next();
        lista.add(u);
    }
    return lista;
}
}

```

#### Controlador Ingreso

```

package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Ingresos;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author paul_
 */
public class ControladorIngreso extends AbstractControlador<Ingresos> {

    @Override
    public boolean validar(Ingresos objeto) {
        return true;
    }

    public List<Ingresos> ingresos() {

        List<Ingresos> lista = new ArrayList();
        Ingresos u;
        Iterator i = super.getList().iterator();
        while (i.hasNext()) {
            u = (Ingresos) i.next();
            lista.add(u);
        }
        return lista;
    }
}

```

```
}  
  
Controlador Tarifa  
package ec.edu.ups.controlador;  
  
import ec.edu.ups.modelo.Tarifa;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
  
/**  
 *  
 * @author paul_  
 */  
public class ControladorTarifa extends AbstractControlador<Tarifa>{  
  
    private Tarifa tarifa;  
  
    @Override  
    public boolean validar(Tarifa objeto) {  
        return true;  
    }  
  
    public Tarifa getTarifa() {  
        return tarifa;  
    }  
  
    public void setTarifa(Tarifa tarifa) {  
        this.tarifa = tarifa;  
    }  
    public List<Tarifa> tarifas() {  
  
        List<Tarifa> lista = new ArrayList();  
        Tarifa u;  
        Iterator i = super.getList().iterator();  
        while (i.hasNext()) {  
            u = (Tarifa) i.next();  
            lista.add(u);  
        }  
        return lista;  
    }  
}
```

```
Controlador Ticket Cliente  
package ec.edu.ups.controlador;  
  
import ec.edu.ups.modelo.ClienteFijo;  
import ec.edu.ups.modelo.Tarifa;  
import ec.edu.ups.modelo.TicketClienteMomentaneo;  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.GregorianCalendar;
```

```

import java.util.Iterator;
import java.util.List;

/**
 *
 * @author paul_
 */
public class ControladorTicketCliente extends AbstractControlador<TicketClienteMomentaneo> {

    private final ControladorTarifa controladorTarifa;

    public ControladorTicketCliente( ControladorTarifa controladorTarifa) {

        this.controladorTarifa = controladorTarifa;
    }

    @Override
    public boolean validar(TicketClienteMomentaneo objeto) {
        return true;
    }

    public Double calcularTotal(TicketClienteMomentaneo ticket) {
        Tarifa tarifal=null;
        for (Tarifa tarifa : controladorTarifa.buscarTodo()) {
            tarifal=tarifa;
        }

        double total=0;
        int fraccion;

        // Valores de las tarifas
        double mediaHoraT;

        //Asignamos las tarifas segun el tipo de vehiculo del cliente Momentaneo
        if (ticket.getTipoVehiculo().equalsIgnoreCase("Motocicleta")) {
            mediaHoraT=tarifal.getMediaHoraM();
        }else if(ticket.getTipoVehiculo().equalsIgnoreCase("Automovil")){
            mediaHoraT=tarifal.getMediaHoraA();

        }else {
            // Tarifa para vehivulos pesados
            mediaHoraT=tarifal.getMediaHoraP();

        }

        //obtenenemos horas y minutos

        double minutos = ( ticket.getFechaHoraSalida().getTimeInMillis()-ticket.getFechaHoraIngreso().getTimeInMillis()) / 60000;
        fraccion = (int) (minutos / 60);
        if(fraccion<1){
            total=mediaHoraT;
            fraccion=1;
        }
    }
}

```



```
        }else{
            total = (minutos*mediaHoraT)/30;
        }

        return total;
    }
    public int calcularIntervaloTiempo(TicketClienteMomentaneo ticket){

        int fraccion;

        Date in = ticket.getFechaHoraIngreso().getTime();
        Date sa=ticket.getFechaHoraSalida().getTime();

        int minutos = (int) (( sa.getTime()-in.getTime())/ 60000);

        return (int) minutos;

    }
    public List<TicketClienteMomentaneo> clientesM() {

        List<TicketClienteMomentaneo> lista = new ArrayList();
        TicketClienteMomentaneo u;
        Iterator i = super.getList().iterator();
        while (i.hasNext()) {
            u = (TicketClienteMomentaneo) i.next();
            lista.add(u);
        }
        return lista;
    }
    public double generarMulta(TicketClienteMomentaneo cliente){

        double valorMulta=0;

        Calendar fechaExpiracion =cliente.getFechaHoraSalida();
        Calendar fechaActual = new GregorianCalendar();
        int dias=-1;

        while(fechaExpiracion.before(fechaActual)|| fechaExpiracion.equals(fechaActual)){
            dias++;
            fechaExpiracion.add(Calendar.DATE,1);
        }
        //si han transcurrido 7 dias ( una semana) de la fecha, se le aumenta un 10%
        al bono a pagar
        if(dias==7){
            valorMulta= cliente.getTarifa()+(cliente.getTarifa()*0.1);
        }
        return valorMulta;
    }
}

Controlador Usuario
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Usuario;
```

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author paul_
 */
public class ControladorUsuario extends AbstractControlador<Usuario> {

    private Usuario usuario;

    @Override
    public boolean validar(Usuario objeto) {

        return true;

    }

    public List<Usuario> usuarios() {

        List<Usuario> lista = new ArrayList();
        Usuario u;
        Iterator i = super.buscarTodo().iterator();
        while (i.hasNext()) {
            u = (Usuario) i.next();
            lista.add(u);

        }
        return lista;

    }

    public Usuario getUsuario() {
        return usuario;
    }

    public boolean iniciarSesion(String correo, String pass) {

        for (Usuario usu : super.buscarTodo()) {
            Usuario u = (Usuario) usu;
            if (u.getCorreo().equals(correo) && u.getPass().equals(pass)) {
                this.usuario = u;
                return true;
            }
        }
        return false;

    }

}
```

- **Ec.edu.ups.vista**

Ventana Principal

```
package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorClienteFijo;
import ec.edu.ups.controlador.ControladorEgreso;
import ec.edu.ups.controlador.ControladorEspacios;
import ec.edu.ups.controlador.ControladorFacturaClienteFijo;
import ec.edu.ups.controlador.ControladorFacturaClienteM;
import ec.edu.ups.controlador.ControladorHistorialClienteM;
import ec.edu.ups.controlador.ControladorHistorialClientesF;
import ec.edu.ups.controlador.ControladorIngreso;
import ec.edu.ups.controlador.ControladorTarifa;
import ec.edu.ups.controlador.ControladorTicketCliente;
import ec.edu.ups.controlador.ControladorUsuario;
import ec.edu.ups.modelo.Espacios;
import javax.swing.JDesktopPane;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JMenuItem;
import javax.swing.JMenuItem;

/**
 *
 * @author paul_
 */

public final class VentanaPrincipal extends javax.swing.JFrame {

    private final VentanaIniciarSesion ventanaIniciarSesion;
    private final GestionUsuarios gestionUsuarios;
    private final Tarifas tarifas;
    private final GestionClientesFijos gestionClienteFijo;
    private final Tickets tickets;
    private final ConsultaMulta consultaMulta;
    private final ClientesFijosPuesto clientesFijosPuesto;
    private final Reportes reportes;
    private final FacturaClienteMo facturaClienteM;
    private final FacturaClienteFijos facturaClienteFijo;
    private final ControladorUsuario controladorUsuario;
    private final ControladorTarifa controladorTarifa;
    private final ControladorClienteFijo controladorClienteFijo;
    private final ControladorEspacios controladorEspacios;
    private final ControladorTicketCliente controladorTicketCliente;
    private final ControladorHistorialClienteM controladorHistorialClienteM;
    private final ControladorHistorialClientesF controladorHistorialClienteF;
    private final ControladorFacturaClienteFijo controladorFacturaClienteFijo;
    private final ControladorFacturaClienteM controladorFacturaClienteM;
    private final ControladorIngreso controladorIngresos;
    private final ControladorEgreso controladorEgreso;

    public VentanaPrincipal() {
        initComponents();

        controladorEspacios = new ControladorEspacios();

        controladorIngresos = new ControladorIngreso();
        controladorEgreso = new ControladorEgreso();

        controladorFacturaClienteM = new ControladorFacturaClienteM();
```

```

        controladorFacturaClienteFijo = new ControladorFacturaClienteFijo();
        controladorHistorialClienteF = new ControladorHistorialClientesF();
        controladorHistorialClienteM = new ControladorHistorialClienteM();
        controladorClienteFijo = new ControladorClienteFijo();
        controladorUsuario = new ControladorUsuario();
        controladorTarifa = new ControladorTarifa();
        controladorTicketCliente = new ControladorTicketCliente(controladorTarifa);
        ventanaIniciarSesion = new VentanaIniciarSesion(this, controladorUsuario);
        gestionUsuarios = new GestionUsuarios(controladorUsuario);
        tarifas = new Tarifas(controladorTarifa);
        gestionClienteFijo = new GestionClientesFijos(controladorClienteFijo, controladorTarifa, controladorEspacios, controladorIngresos);
        tickets = new Tickets(controladorTicketCliente, controladorEspacios, controladorTarifa);
        consultaMulta = new ConsultaMulta(controladorEspacios, controladorClienteFijo, controladorTicketCliente);
        clientesFijosPuesto = new ClientesFijosPuesto(controladorClienteFijo, controladorHistorialClienteF);
        reportes = new Reportes(controladorIngresos, controladorEgreso);
        facturaClienteM = new FacturaClienteMo(controladorFacturaClienteM, controladorTicketCliente);
        facturaClienteFijo = new FacturaClienteFijos(controladorFacturaClienteFijo, controladorClienteFijo);

        //cargarEspaciosDefault();

        menuFacturas.setVisible(false);
        RegistrosMonetariosMenuItem.setVisible(false);
        VerTablasMenu.setVisible(false);
        HistorialMenu.setVisible(false);
        CerrarSesionMenuItem.setVisible(false);

    }

    public void cargarEspaciosDefault() {
        for (int i = 1; i <= 50; i++) {
            Espacios e = new Espacios();
            e.setId(i);
            controladorEspacios.crear(e);
        }
    }

    private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void FacturaClienteFijoMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(facturaClienteFijo);
        facturaClienteFijo.setVisible(true);
    }

    private void CerrarSesionMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        menuFacturas.setVisible(false);
        RegistrosMonetariosMenuItem.setVisible(false);
        VerTablasMenu.setVisible(false);
        HistorialMenu.setVisible(false);
        CerrarSesionMenuItem.setVisible(false);
        IniciarSesionMenuItem.setVisible(true);
        exitMenuItem.setVisible(true);
    }

```

```

    }

    private void IniciarSesionMenuItemActionPerformed(java.awt.event.ActionEvent
    evt) {
        desktopPane.add(ventanaIniciarSesion);
        ventanaIniciarSesion.setVisible(true);
    }

    private void RegistrosMonetariosMenuItemActionPerformed(java.awt.event.Action-
    Event evt) {
        desktopPane.add(reportes);
        reportes.setVisible(true);
    }

    private void facturaClienteMomentaneoMenuItemActionPerformed(java.awt.event.Ac-
    tionEvent evt) {
        desktopPane.add(facturaClienteM);
        facturaClienteM.setVisible(true);
    }

    private void ClientesFijosHistorialMenuItemActionPerformed(java.awt.event.Ac-
    tionEvent evt) {
        desktopPane.add(clientesFijosPuesto);
        clientesFijosPuesto.setVisible(true);
    }

    private void UsuariosMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(gestionUsuarios);
        gestionUsuarios.setVisible(true);
    }

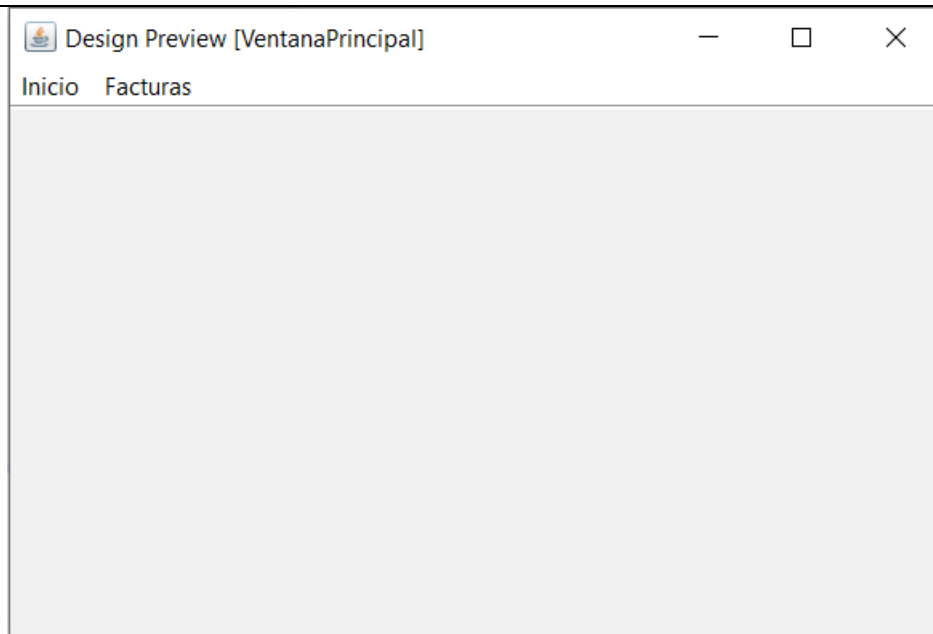
    private void TarifasMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(tarifas);
        tarifas.setVisible(true);
    }

    private void MultasMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(consultaMulta);
        consultaMulta.setVisible(true);
    }

    private void TicketsMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        desktopPane.add(tickets);
        tickets.setVisible(true);
    }

    private void ClientesFijosMenuItemActionPerformed(java.awt.event.ActionEvent
    evt) {
        desktopPane.add(gestionClienteFijo);
        gestionClienteFijo.setVisible(true);
    }

```



#### Ventana Iniciar Sesión

```
package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorUsuario;

import ec.edu.ups.modelo.Usuario;

import javax.swing.JOptionPane;

/**
 *
 * @author paul_
 */
public final class VentanaIniciarSesion extends javax.swing.JInternalFrame {

    private VentanaPrincipal ventanaPrincipal;
    private ControladorUsuario controladorUsuario;

    /**
     * Creates new form VentanaIniciarSesion
     *
     * @param ventanaPrincipal
     * @param controladorUsuario
     */
    public VentanaIniciarSesion(VentanaPrincipal ventanaPrincipal, ControladorUsua-
rio controladorUsuario) {
        initComponents();

        this.ventanaPrincipal = ventanaPrincipal;
        this.controladorUsuario = controladorUsuario;

        //generarAdmin();
    }
    private void btnIniciarSesionActionPerformed(java.awt.event.ActionEvent evt) {
        //
        String pass = "";
        char[] pass1 = txtPass.getPassword();
        for (int i = 0; i < pass1.length; i++) {
            pass = pass + pass1[i];
        }
    }
}
```

```

    }

    if (controladorUsuario.iniciarSesion(txtCorreo.getText().trim(), pass)) {
        ventanaPrincipal.getIniciarSesionMenuItem().setVisible(false);
        ventanaPrincipal.getCerrarSesionMenuItem().setVisible(true);
        ventanaPrincipal.getMenuFacturas().setVisible(true);
        ventanaPrincipal.getRegistrosMonetariosMenuItem().setVisible(true);
        ventanaPrincipal.getVerTablasMenu().setVisible(true);
        ventanaPrincipal.getHistorialMenu().setVisible(true);
        ventanaPrincipal.getExitMenuItem().setVisible(false);

        if (controladorUsuario.getUsuario().getRol().equals("user")) {
            ventanaPrincipal.getUsuariosMenuItem().setVisible(false);

        } else {

            ventanaPrincipal.getUsuariosMenuItem().setVisible(true);
        }
        this.dispose();
        Limpiar();
        JOptionPane.showMessageDialog(this, "Inicio de sesion exitoso");
    } else {

        JOptionPane.showMessageDialog(this, "Usuario o contrasena incorrecta ");
        Limpiar();
    }
}

//String cedula, String nombre, String apellido, String direccion, String telefono,
String rol, String correo, String pass
}
public void generarAdmin() {
    //Usuario admin = new Usuario("1303753618","PABLO","NIETO","EL VE-
CINO","0987645328","admin","pnieto@hotmail.com","admin");
    Usuario admin = new Usuario();
    admin.setCedula("1721286993");
    admin.setNombre("PAUL");
    admin.setApellido("GUAPUCAL");
    admin.setDireccion("RICAURTE");
    admin.setTlf("0983474289");
    admin.setRol("admin");
    admin.setCorreo("paulguapucal@gmail.com");
    admin.setPass("admin");

    controladorUsuario.crear(admin);
    System.out.println("ADMIN GENERADO CORRECTAMENTE");
}

private void formInternalFrameActivated(javax.swing.event.InternalFrameEvent
evt) {
}

public void Limpiar() {
    txtCorreo.setText("");
    txtPass.setText("");
}
}

```

INICIAR SESION

Correo:

Contraseña:

### Tickets

```
package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorEspacios;
import ec.edu.ups.controlador.ControladorTarifa;
import ec.edu.ups.controlador.ControladorTicketCliente;

import ec.edu.ups.modelo.TicketClienteMomentaneo;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Calendar;

import javax.swing.JOptionPane;
import com.lowagie.text.Document;
import com.lowagie.text.PageSize;
import com.lowagie.text.pdf.PdfContentByte;
import com.lowagie.text.pdf.PdfTemplate;
import com.lowagie.text.pdf.PdfWriter;
import ec.edu.ups.modelo.Espacios;
import java.awt.Desktop;
import java.awt.Graphics2D;
import java.io.File;

import javax.swing.table.DefaultTableModel;

/**
 *
 * @author paul_
 */
public class Tickets extends javax.swing.JInternalFrame {

    private int id;

    private ControladorTicketCliente controladorTicketCliente;
    private ControladorEspacios controladorEspacios;
    private ControladorTarifa controladorTarifa;

    public Tickets(ControladorTicketCliente controladorTicketCliente, ControladorEspacios controladorEspacios, ControladorTarifa controladorTarifa) {
```



```

initComponents();
this.controladorTicketCliente = controladorTicketCliente;
this.controladorEspacios = controladorEspacios;
this.controladorTarifa = controladorTarifa;

}

public void validarCampos() {

}

public void limpiarCbx() {
    cbxEspacioAsignar.removeAllItems();
}

private void cargarClientesMomentaneosTbl() {
    String fecha = "";
    DefaultTableModel modelo = (DefaultTableModel) tblClientesMomentaneos.getMo-
del();
    modelo.setRowCount(0);
    if (controladorTicketCliente.clientesM() != null) {
        for (TicketClienteMomentaneo cf : controladorTicketCliente.clientesM())
        {

            if (cf.isEstado() == false) {

                Calendar f = cf.getFechaHoraIngreso();
                int mes = (f.get(Calendar.MONTH)) + 1;
                fecha = f.get(Calendar.DAY_OF_MONTH) + "/" + mes + "/" +
f.get(Calendar.YEAR);
                String hora = "" + f.get(Calendar.HOUR_OF_DAY) + " : " +
f.get(Calendar.MINUTE) + " : " + f.get(Calendar.SECOND);
                Object[] rowData = {cf.getId(), cf.getNombre(), fecha, hora,
cf.getTipoVehiculo()};
                modelo.addRow(rowData);
                tblClientesMomentaneos.setModel(modelo);

            }

        } else {
            System.out.println("LISTA VACIA");
        }
    }

    /**
     *
     */
    @SuppressWarnings("empty-statement")
    public void cargarClientesTblEgresados() {
        String fecha = "";
        DefaultTableModel modelo = (DefaultTableModel) tblListaClientes.getModel();
        modelo.setRowCount(0);
        if (controladorTicketCliente.clientesM() != null) {
            for (TicketClienteMomentaneo cf : controladorTicketCliente.buscarTodo())
            {

                if (cf.isEstado()) {
                    Calendar f = cf.getFechaHoraIngreso();
                    int mes = f.get(Calendar.MONTH) + 1;
                    fecha = f.get(Calendar.DAY_OF_MONTH) + "/" + mes + "/" +
f.get(Calendar.YEAR);

```

```

        String hora = "" + f.get(Calendar.HOUR_OF_DAY) + " : " +
f.get(Calendar.MINUTE) + " : " + f.get(Calendar.SECOND);
        Object[] rowData = {cf.getId(), cf.getNombre(), fecha, hora,
cf.getTipoVehiculo(), cf.getTarifa(), cf.getMulta()};
        modelo.addRow(rowData);
        tblListaClientes.setModel(modelo);
    }

    }
} else {
    System.out.println("LISTA VACIA");
}
}

public void cargarCbxEspaciosDisp() {
    limpiarCbx();
    for (Espacios e : controladorEspacios.espaciosDisponibles()) {
        cbxEspacioAsignar.addItem(e.getId().toString());
    }
}

private void cbxEspacioAsignarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void tblListaClientesMouseClicked(java.awt.event.MouseEvent evt) {

    int index = tblListaClientes.getSelectedRow();
    id = Integer.parseInt("" + tblListaClientes.getValueAt(index, 0));
    String nombre = "" + tblListaClientes.getValueAt(index, 1);
    String FechaIngreso = "" + tblListaClientes.getValueAt(index, 2);
    String HoraIngreso = "" + tblListaClientes.getValueAt(index, 3);
    String tipoVehiculo = "" + tblListaClientes.getValueAt(index, 4);
    double tarifa = Double.parseDouble("" + tblListaClientes.getValueAt(index,
5));
    double multa = Double.parseDouble("" + tblListaClientes.getValueAt(index,
6));

    if (tipoVehiculo.trim().equals("Automovil")) {
        cbxAutomovil.setSelectedItem(0);
    } else if (tipoVehiculo.trim().equals("Motocicleta")) {
        cbxAutomovil.setSelectedItem(1);
    } else {
        cbxAutomovil.setSelectedItem(2);
    }

    txtNombre.setText(nombre);
    txtFechaIngreso.setText(FechaIngreso);

    TicketClienteMomentaneo c = controladorTicketCliente.buscar(id);
    txtEspActual.setText("" + c.getEspacioParqueo());
    txtValor.setText("" + tarifa);
}

private void tblClientesMomentaneosMouseClicked(java.awt.event.MouseEvent evt) {
    int index = tblClientesMomentaneos.getSelectedRow();
    id = Integer.parseInt("" + tblClientesMomentaneos.getValueAt(index, 0));
    String nombre = "" + tblClientesMomentaneos.getValueAt(index, 1);
    String FechaIngreso = "" + tblClientesMomentaneos.getValueAt(index, 2);
    String HoraIngreso = "" + tblClientesMomentaneos.getValueAt(index, 3);

```

```
String tipoVehiculo = "" + tblClientesMomentaneos.getValueAt(index, 4);

if (tipoVehiculo.trim().equals("Automovil")) {
    cbxAutomovil.setSelectedItem(0);

} else if (tipoVehiculo.trim().equals("Motocicleta")) {
    cbxAutomovil.setSelectedItem(1);
} else {
    cbxAutomovil.setSelectedItem(2);
}

txtNombre.setText(nombre);
txtFechaIngreso.setText.FechaIngreso);
txtEspActual.setText("");

TicketClienteMomentaneo c = controladorTicketCliente.buscar(id);

txtEspActual.setText("" + c.getEspacioParqueo());
}

private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {

    Calendar fechaIngreso = Calendar.getInstance();
    // String fecha = ""+fechaIngreso.get(Calendar.DAY_OF_MONTH)+"/"+fecha-
    Ingreso.get(Calendar.MONTH)+"/"+fechaIngreso.get(Calendar.YEAR);

    // TicketClienteMomentaneo ticket = new TicketClienteMomentaneo( "Cliente "
    , cbxAutomovil.getSelectedItem().toString().trim(), fechaIngreso, null, 0, 0, 0.0,
    false);
    //ticket.setEspacioParqueo(Integer.parseInt(cbxEspacioAsignar.getSelectedItem().toString().trim()));
    //ticket.setNombre("Cliente "+ticket.getId());
    TicketClienteMomentaneo ticket = new TicketClienteMomentaneo();
    ticket.setNombre("Cliente " + ticket.getId());
    ticket.setTipoVehiculo(cbxAutomovil.getSelectedItem().toString().trim());
    ticket.setFechaHoraIngreso(fechaIngreso);
    ticket.setEspacioParqueo(Integer.parseInt(cbxEspacioAsignar.getSelectedItem().toString().trim()));
    ticket.setEstado(false);
    if (controladorTicketCliente.crear(ticket)) {

        JOptionPane.showMessageDialog(this, "Cliente registrado con exito");

        if (controladorEspacios.AsignarEspacio(Integer.parseInt(cbxEspacioA-
        signar.getSelectedItem().toString().trim()),
            "" + ticket.getId())) {

            System.out.println("SE ASIGNO ESPACIO CORRECTAMENTE AL OBJETO " +
            ticket);

        }

        cargarClientesTblEgresados();
        cargarClientesMomentaneosTbl();
        cargarCbxEspaciosDisp();

    } else {
        JOptionPane.showMessageDialog(this, "Error al crear Cliente");
    }
}

public void limpiar() {
```

```

txtNombre.setText("");
txtFechaIngreso.setText("");
txtEspActual.setText("");
txtValor.setText("");

}
private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    // limpiar();
}

private void btnRetirarActionPerformed(java.awt.event.ActionEvent evt) {

    Calendar fechaSalida = Calendar.getInstance();
    // String fecha = ""+fechaIngreso.get(Calendar.DAY_OF_MONTH)+"/"+fecha-
Ingreso.get(Calendar.MONTH)+"/"+fechaIngreso.get(Calendar.YEAR);

    TicketClienteMomentaneo ticket = controladorTicketCliente.buscar(id);
    ticket.setFechaHoraSalida(fechaSalida);
    ticket.setMinutos(controladorTicketCliente.calcularIntervaloTiempo(ticket));

    double total = controladorTicketCliente.calcularTotal(ticket);
    ticket.setTarifa(total);
    ticket.setEstado(true);
    double multa = controladorTicketCliente.generarMulta(ticket);
    ticket.setMulta(multa);

    if (controladorTicketCliente.actualizar(ticket)) {

        JOptionPane.showMessageDialog(this, "Cliente actualizado con exito");

        controladorEspacios.AsignarEspacio(Integer.parseInt(txtEspActual.get-
Text().trim()), "");

        cargarClientesTblEgresados();
        cargarClientesMomentaneosTbl();
        cargarCbxEspaciosDisp();

    } else {
        JOptionPane.showMessageDialog(this, "Error al crear Cliente");
    }

}

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {

    int d = JOptionPane.showConfirmDialog(this, "Esta seguro de eliminar el " +
txtNombre.getText().trim());
    if (d == JOptionPane.YES_OPTION) {

        TicketClienteMomentaneo c = controladorTicketCliente.buscar(id);

        if (controladorTicketCliente.eliminar(c)) {
            controladorEspacios.AsignarEspacio(Integer.parseInt(txtEspAc-
tual.getText().trim()), "");

            JOptionPane.showMessageDialog(this, "eliminado exitosamente");

        } else {
            JOptionPane.showMessageDialog(this, "no se pudo eliminar ERROR");
        }
    }
}

```

```

        cargarClientesTblEgresados();
        cargarClientesMomentaneosTbl();
        cargarCbxEspaciosDisp();
    } else {
        JOptionPane.showMessageDialog(this, "El cliente todavia no ha");
    }

}

private void btnListarActionPerformed(java.awt.event.ActionEvent evt) {
    cargarClientesTblEgresados();
    cargarClientesMomentaneosTbl();
    cargarCbxEspaciosDisp();

}

private void formInternalFrameActivated(javax.swing.event.InternalFrameEvent
evt) {
    cargarCbxEspaciosDisp();
}

public void imprimir() {
    float x = 10;
    float y = 20;
    Document document = new Document(PageSize.A4.rotate());
    try {
        PdfWriter writer = PdfWriter.getInstance(document, new FileOut-
putStream("datos/tblClientesMomentaneos" + id + ".pdf"));

        document.open();
        PdfContentByte cb = writer.getDirectContent();

        cb.saveState();

        PdfTemplate pdfTemplate = cb.createTemplate(tblClientesMomenta-
neos.getWidth(), tblClientesMomentaneos.getHeight());
        Graphics2D g2 = pdfTemplate.createGraphics(tblClientesMomenta-
neos.getWidth(), tblClientesMomentaneos.getHeight());
        /*g2.setColor(Color.BLACK);
g2.drawRect(x-2, y-2, table.getWidth()+2, table.getHeight()+2);*/
tblClientesMomentaneos.print(g2);
System.out.println("x=" + x + ", " + "y=" + y);
cb.addTemplate(pdfTemplate, x, y);
g2.dispose();
cb.restoreState();
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
    document.close();

}

/*public static void main(String[] args) {
JTable2Pdf frame = new JTable2Pdf();
frame.pack();
frame.setVisible(true);
frame.print();
}*/

public void cargarBusqueda(TicketClienteMomentaneo c) {
    DefaultTableModel modelo = (DefaultTableModel) tblClientesMomentaneos.getMo-
del();

```

```

        modelo.setRowCount(0);
        Calendar f = c.getFechaHoraIngreso();
        int mes = f.get(Calendar.MONTH) + 1;
        String fecha = f.get(Calendar.DAY_OF_MONTH) + "/" + f.get(Calendar.MONTH) +
"/" + f.get(Calendar.YEAR);
        String hora = "" + f.get(Calendar.HOUR_OF_DAY) + " : " + f.get(Calendar.MINUTE) + " : " + f.get(Calendar.SECOND);
        Object[] rowData = {c.getId(), c.getNombre(), fecha, hora, c.getTipoVehiculo()};
        modelo.addRow(rowData);
        tblClientesMomentaneos.setModel(modelo);
    }

    public void abrirarchivo(String archivo) {

        try {

            File objetofile = new File(archivo);
            Desktop.getDesktop().open(objetofile);

        } catch (IOException ex) {

            System.out.println(ex);

        }
    }

    private void btnImprimirTicketActionPerformed(java.awt.event.ActionEvent evt) {

        String respuesta = JOptionPane.showInputDialog(this, "Ingrese ID de ticket");
        int idTicket = Integer.parseInt(respuesta.trim());
        TicketClienteMomentaneo c = controladorTicketCliente.buscar(idTicket);
        System.out.println("" + c);
        cargarBusqueda(c);

        this.pack();
        this.imprimir();
        abrirarchivo("datos/tblClientesMomentaneos" + id + ".pdf");
    }

```

### DATOS DEL CLIENTE

Nombre:

Tipo de Vehículo:

Fecha de Ingreso:

### Espacio Asignado Actual

Espacio:

Espacios disponibles:

Total:

AGREGAR

ELIMINAR

LIMPIAR

LISTAR

RETIRAR

IMPRIMIR TICKET

### Salida de Tickets

ID	Cliente	Fecha Ingreso	Hora Ingreso	Tipo vehiculo

### Lista Clientes Momentaneos

ID	Cliente	Fecha Ingreso	Hora Ingreso	Tipo vehiculo	Tarifa	Muta

## Tarifas

```
package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorHistorialClienteM;
import ec.edu.ups.controlador.ControladorTarifa;
import ec.edu.ups.modelo.Tarifa;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author paul_
 */
public class Tarifas extends javax.swing.JInternalFrame {

    private ControladorTarifa controladorTarifa;

    public Tarifas(ControladorTarifa controladorTarifa) {
        initComponents();
        this.controladorTarifa = controladorTarifa;
    }

    public boolean validarCampos() {
        if (txtmediaHoraM.getText().isEmpty() || txtHoraM.getText().isEmpty() ||
            txtMensualM.getText().isEmpty() || txtmediaHoraA.getText().isEmpty() || txtHoraA.getText().isEmpty() ||
            txtMensualA.getText().isEmpty() || txtmediaHoraP.getText().isEmpty() || txtHoraP.getText().isEmpty() ||
            txtMensualP.getText().isEmpty()) {

            JOptionPane.showMessageDialog(this, "CAMPOS VACIOS");

            return false;
        } else {
```

```

        return true;

    }

}

public boolean validarTipoDato() {
    return false;
}

private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    if (validarCampos()) {
        if (controladorTarifa.tarifas().isEmpty()) {

            // Double mediaHoraM, Double horaM, Double mensualM, Double media-
            HoraA, Double horaA, Double mensualA, Double mediaHoraP, Double horaP, Double mensu-
            alP

            Tarifa tarifa = new Tarifa();
            tarifa.setMediaHoraM(Double.parseDouble(txtmediaHoraM.getText()));
            tarifa.setHoraM(Double.parseDouble(txtHoraM.getText()));
            tarifa.setMensualM(Double.parseDouble(txtMensualA.getText()));
            tarifa.setMediaHoraA(Double.parseDouble(txtmediaHoraA.getText()));
            tarifa.setHoraA(Double.parseDouble(txtHoraA.getText()));
            tarifa.setMensualA(Double.parseDouble(txtMensualA.getText()));
            tarifa.setMediaHoraP(Double.parseDouble(txtmediaHoraP.getText()));
            tarifa.setHoraP(Double.parseDouble(txtHoraP.getText()));
            tarifa.setMensualP(Double.parseDouble(txtMensualP.getText()));
            if (controladorTarifa.crear(tarifa)) {
                controladorTarifa.setTarifa(tarifa);
                JOptionPane.showMessageDialog(this, "Tarifa cargada con exito");
            } else {
                JOptionPane.showMessageDialog(this, "No se guardo la tarifa
error");
            }
        } else {
            JOptionPane.showMessageDialog(this, "Ya existe una tarifa asignada,
actualizela");
        }
    }
}

private void txtmediaHoraAActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void formInternalFrameActivated(javax.swing.event.InternalFrameEvent
evt) {

    if (controladorTarifa.tarifas() != null) {

        for (Tarifa tarifa : controladorTarifa.tarifas()) {
            txtmediaHoraM.setText("" + tarifa.getMediaHoraM());
            txtHoraM.setText("" + tarifa.getHoraM());
            txtMensualM.setText("" + tarifa.getMensualM());

            txtmediaHoraA.setText("" + tarifa.getMediaHoraA());
            txtHoraA.setText("" + tarifa.getHoraA());
            txtMensualA.setText("" + tarifa.getMensualA());

            txtmediaHoraP.setText("" + tarifa.getMediaHoraP());
            txtHoraP.setText("" + tarifa.getHoraP());
        }
    }
}

```



```

        txtMensualP.setText("" + tarifa.getMensualP());
    }

}

private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    if (validarCampos()) {
        Tarifa tarifa = null;
        for (Tarifa tarifa : controladorTarifa.tarifas()) {
            tarifa = tarifa;
        }
        // Double mediaHoraM, Double horaM, Double mensualM, Double mediaHoraA,
        Double horaA, Double mensualA, Double mediaHoraP, Double horaP, Double mensualP
        /* Tarifa tarifa = new Tarifa(
            Double.parseDouble(txtmediaHoraM.getText()), Double.parseDouble(txtHoraM.getText()),
            Double.parseDouble(txtMensualM.getText()), Double.parseDouble(txtmediaHoraA.getText()),
            Double.parseDouble(txtHoraA.getText()), Double.parseDouble(txtMensualA.getText()),
            Double.parseDouble(txtmediaHoraP.getText()), Double.parseDouble(txtHoraP.getText()),
            Double.parseDouble(txtMensualP.getText())); */
        Tarifa tarifa = new Tarifa();
        tarifa.setMediaHoraM(Double.parseDouble(txtmediaHoraM.getText()));
        tarifa.setHoraM(Double.parseDouble(txtHoraM.getText()));
        tarifa.setMensualM(Double.parseDouble(txtMensualA.getText()));
        tarifa.setMediaHoraA(Double.parseDouble(txtmediaHoraA.getText()));
        tarifa.setHoraA(Double.parseDouble(txtHoraA.getText()));
        tarifa.setMensualA(Double.parseDouble(txtMensualA.getText()));
        tarifa.setMediaHoraP(Double.parseDouble(txtmediaHoraP.getText()));
        tarifa.setHoraP(Double.parseDouble(txtHoraP.getText()));
        tarifa.setMensualP(Double.parseDouble(txtMensualP.getText()));

        if (controladorTarifa.actualizar(tarifa)) {

        }
    }
}

```

Tarifa para motocicletas

30 minutos

1 Hora

30 dias

Tarifa para automoviles

30 minutos

1 Hora

30 dias

Tarifa para pesados

30 minutos

1 Hora

30 dias

GUARDAR

ACTUALIZAR

## Reportes

```
package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorEgreso;
import ec.edu.ups.controlador.ControladorIngreso;
import ec.edu.ups.modelo.Egresos;
import ec.edu.ups.modelo.Ingresos;

import java.util.Calendar;

import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author paul_
 */
public final class Reportes extends javax.swing.JInternalFrame {

    private int idIngreso = 0;
    private int id = 0;

    private ControladorIngreso controladorIngreso;
    private ControladorEgreso controladorEgreso;

    public Reportes(ControladorIngreso controladorIngreso, ControladorEgreso controladorEgreso) {
        initComponents();
        this.controladorIngreso = controladorIngreso;
        this.controladorEgreso = controladorEgreso;

        cargarIngresosTbl();
        cargarEgresosTbl();
        calcularSaldo();
    }

    public void cargarIngresosTbl() {
        String fecha = "";
        String hora = "";
        DefaultTableModel modelo = (DefaultTableModel) tblIngresos.getModel();
        modelo.setRowCount(0);
        if (controladorIngreso.getIngresos() != null) {
            for (Ingresos ingreso : controladorIngreso.buscarTodo()) {
                Calendar f = ingreso.getFechaHora();
                int mes = f.get(Calendar.MONTH) + 1;
                fecha = f.get(Calendar.DAY_OF_MONTH) + "/" + mes + "/" + f.get(Calendar.YEAR);
                hora = f.get(Calendar.HOUR_OF_DAY) + ":" + (f.get(Calendar.MINUTE)) + ":" + f.get(Calendar.SECOND);
                Object[] rowData = {ingreso.getId(), fecha, hora, ingreso.getDescripcion(), ingreso.getIngreso()};
                modelo.addRow(rowData);
                tblIngresos.setModel(modelo);
            }
        } else {
            System.out.println("LISTA VACIA");
        }

        txtTotalIngresos.setText(" " + calcularTotalIngresos(controladorIngreso.buscarTodo()));
    }
}
```

```

public void cargarEgresosTbl() {
    String fecha = "";
    String hora = "";
    DefaultTableModel modelo = (DefaultTableModel) tblEgresos.getModel();
    modelo.setRowCount(0);
    if (controladorEgreso.egresos() != null) {
        for (Egresos egreso : controladorEgreso.buscarTodo()) {
            Calendar f = egreso.getFechaHora();
            int mes = (f.get(Calendar.MONTH)) + 1;
            fecha = f.get(Calendar.DAY_OF_MONTH) + "/" + mes + "/" + f.get(Calendar.YEAR);
            hora = f.get(Calendar.HOUR_OF_DAY) + ":" + (f.get(Calendar.MINUTE))
+ ":" + f.get(Calendar.SECOND);
            Object[] rowData = {egreso.getId(), fecha, hora, egreso.getDescripcion(), egreso.getEgreso()};
            modelo.addRow(rowData);
            tblEgresos.setModel(modelo);
        }
    } else {
        System.out.println("LISTA VACIA");
    }
    txtTotalEgresos.setText("" + calcularTotalEgresos(controladorEgreso.buscarTodo()));
}

public double calcularTotalIngresos(List<Ingresos> ingresos) {
    double acum = 0;
    for (Ingresos ingreso : ingresos) {
        acum = acum + ingreso.getIngreso();
    }
    return acum;
}

public void calcularSaldo() {
    if (txtTotalIngresos.getText().isEmpty() && txtTotalEgresos.getText().isEmpty()) {

    } else {

        String ingresos = txtTotalIngresos.getText().trim();
        String egresos = txtTotalEgresos.getText().trim();
        double total = 0;

        total = Double.parseDouble(ingresos) - Double.parseDouble(egresos);
        txtDisponible.setText("" + total);
    }
}

public double calcularTotalEgresos(List<Egresos> egresos) {
    double acum = 0;
    for (Egresos egreso : egresos) {
        acum = acum + egreso.getEgreso();
    }
    return acum;
}

private void btnEliminarIngresoActionPerformed(java.awt.event.ActionEvent evt) {
    if (idIngreso != 0) {

```

```

        Ingresos i = controladorIngreso.buscar(id);

        int d = JOptionPane.showConfirmDialog(this, "Esta seguro de eliminar el
ingreso seleccionado?");

        if (d == JOptionPane.YES_OPTION) {
            if (controladorIngreso.eliminar(i)) {

                JOptionPane.showMessageDialog(this, "Ingreso eliminado correc-
tamente");

            }
        }
        cargarIngresosTbl();

    }

    private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
        cargarEgresosTbl();
        txtDescripcion.setText("");
        txtCantidadEgreso.setText("");
    }

    private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
        if (txtDescripcion.getText().isEmpty() || txtCantidadEgreso.getText().is-
Empty()) {
            JOptionPane.showMessageDialog(this, "Campos vacios");
        } else {

            Egresos egreso = controladorEgreso.buscar(id);

            egreso.setDescripcion(txtDescripcion.getText());
            egreso.setEgreso(Double.parseDouble(txtCantidadEgreso.getText()));
            if (controladorEgreso.actualizar(egreso)) {

                JOptionPane.showMessageDialog(this, "Egreso actualizado exitosa-
mente");

            }
        }
        cargarEgresosTbl();
    }

    private void btnGuardarEActionPerformed(java.awt.event.ActionEvent evt) {
        if (txtDescripcion.getText().isEmpty() || txtCantidadEgreso.getText().is-
Empty()) {
            JOptionPane.showMessageDialog(this, "Campos vacios");
        } else {

            Egresos egreso = new Egresos();
            Calendar fecha = Calendar.getInstance();
            egreso.setFechaHora(fecha);
            egreso.setDescripcion(txtDescripcion.getText());
            egreso.setEgreso(Double.parseDouble(txtCantidadEgreso.getText()));
            if (controladorEgreso.crear(egreso)) {

                JOptionPane.showMessageDialog(this, "Egreso guardado exitosamente");

            }
        }
    }

```

```
cargarEgresosTbl();  
}  
  
private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {  
    if (id != 0) {  
        Egresos egreso = controladorEgreso.buscar(id);  
  
        int d = JOptionPane.showConfirmDialog(this, "Esta seguro de eliminar el  
ingreso seleccionado?");  
  
        if (d == JOptionPane.YES_OPTION) {  
            if (controladorEgreso.eliminar(egreso)) {  
  
                JOptionPane.showMessageDialog(this, "Ingreso eliminado correcta-  
mente");  
  
            }  
        }  
    }  
    cargarIngresosTbl();  
}  
  
private void tblIngresosMouseClicked(java.awt.event.MouseEvent evt) {  
    int index = tblIngresos.getSelectedRow();  
    idIngreso = Integer.parseInt("" + tblIngresos.getValueAt(index, 0));  
}  
  
public double calcularValorDisponible() {  
    return 0;  
}  
  
private void tblEgresosMouseClicked(java.awt.event.MouseEvent evt) {  
    int index = tblIngresos.getSelectedRow();  
    id = Integer.parseInt("" + tblIngresos.getValueAt(index, 0));  
}  
  
private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {  
    cargarIngresosTbl();  
    cargarEgresosTbl();  
    calcularSaldo();  
}
```

**Ingresos**

ID	Fecha	Hora	Descripcion	Ingreso (\$)

ELIMINAR

ACTUALIZAR

Total:

**Egresos**

ID	Fecha	Hora	Descripcion	Egreso (\$)

GUARDAR

ACTUALIZAR

ELIMINAR

LIMPIAR

Total:

Disponible: (\$)

Descripcion

Cantidad (\$)

## Gestión de Usuarios

```

package ec.edu.upes.vista;

import ec.edu.upes.controlador.ControladorUsuario;
import ec.edu.upes.modelo.Usuario;
import java.util.Arrays;

import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author paul_
 */
public class GestionUsuarios extends javax.swing.JInternalFrame {

    private int id;

    private ControladorUsuario controladorUsuario;

    public GestionUsuarios(ControladorUsuario controladorUsuario) {
        initComponents();

        this.controladorUsuario = controladorUsuario;
    }

    public void limpiar() {
        id = 0;
        txtGcedula.setText("");
        txtGnombre.setText("");
        txtGapellido.setText("");
        txtGdireccion.setText("");
        txtGtelefono.setText("");
        txtGcorreo.setText("");
        txtGcontrasena.setText("");
    }

```

```

public void cargarUsuariosTbl() {
    DefaultTableModel modelo = (DefaultTableModel) tblUsuarios.getModel();
    modelo.setRowCount(0);
    if (controladorUsuario.usuarios() != null) {
        for (Usuario usuario : controladorUsuario.usuarios()) {
            Object[] rowData = {usuario.getId(), usuario.getCedula(), usuario.getNombre(), usuario.getApellido(), usuario.getDireccion(), usuario.getTlf(), usuario.getCorreo(), usuario.getPass()};
            modelo.addRow(rowData);
            tblUsuarios.setModel(modelo);
        }
    } else {
        System.out.println("LISTA VACIA");
    }
}

private void txtGcedulaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {
    this.dispose();
}

private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {

    /* txtGcedula.setText(cedula);
    txtGnombre.setText(nombre);
    txtGapellido.setText(apellido);
    txtGDireccion.setText(direccion);
    txtTelefono.setText(telefono);
    txtCorreo.setText(usuario);
    txtContrasena.setText(contrasena); */
    if (Arrays.equals(txtContrasena.getPassword(), txtConfirmarContrasena.getPassword())) {
        String pass = "";
        char[] pass1 = txtContrasena.getPassword();
        for (int i = 0; i < pass1.length; i++) {
            pass = pass + pass1[i];
        }
        // Usuario usuario = new Usuario(txtGcedula.getText().trim(), txtGnombre.getText().trim(), txtGapellido.getText().trim(),
        // , txtGDireccion.getText().trim(), txtTelefono.getText().trim(), "user", txtCorreo.getText().trim(), pass);

        Usuario usuario = new Usuario();
        usuario.setNombre(txtGnombre.getText().trim());
        usuario.setApellido(txtGapellido.getText().trim());
        usuario.setDireccion(txtGDireccion.getText().trim());
        usuario.setTlf(txtTelefono.getText().trim());
        usuario.setCorreo(txtCorreo.getText().trim());
        usuario.setPass(pass);
        usuario.setRol("user");

        if (controladorUsuario.crear(usuario)) {

            cargarUsuariosTbl();
            JOptionPane.showMessageDialog(this, "Registrado con exito");
        }
    }
}

```

```

        limpiar();
    } else {

        JOptionPane.showMessageDialog(this, "No se pudo registrar Usuario");

    }
} else {

    JOptionPane.showMessageDialog(this, "Las contraseñas no coinciden");
}

}

private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {

    if (Arrays.equals(txtContrasena.getPassword(), txtConfi-
marContrasena.getPassword())) {
        String pass = "";
        char[] pass1 = txtContrasena.getPassword();
        for (int i = 0; i < pass1.length; i++) {
            pass = pass + pass1[i];
        }
        //Usuario usuario = new Usuario(txtGcedula.getText().trim(), txtGnom-
bre.getText().trim(), txtGapellido.getText().trim()
        //, txtGDireccion.getText().trim(), txtTele-
fono.getText().trim(), "user", txtCorreo.getText().trim(), pass);

        Usuario usuario = new Usuario();
        usuario.setNombre(txtGnombre.getText().trim());
        usuario.setApellido(txtGapellido.getText().trim());
        usuario.setDireccion(txtGDireccion.getText().trim());
        usuario.setTlf(txtTelefono.getText().trim());
        usuario.setCorreo(txtCorreo.getText().trim());
        usuario.setPass(pass);
        usuario.setRol("user");

        if (controladorUsuario.actualizar(usuario)) {

            cargarUsuariosTbl();
            JOptionPane.showMessageDialog(this, "Actualizado con exito");

            cargarUsuariosTbl();
            limpiar();

        } else {
            JOptionPane.showMessageDialog(this, "No se pudo modificar Usuario");
        }
    } else {

        JOptionPane.showMessageDialog(this, "Las contraseñas no coinciden");
    }

}

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int d = JOptionPane.showConfirmDialog(this, "Esta seguro de eliminar el
usuario " + txtCorreo.getText().trim());
    if (d == JOptionPane.YES_OPTION) {

        Usuario usu = controladorUsuario.buscar(id);
    }
}

```



```
System.out.println("'" + usu);

    if (controladorUsuario.eliminar(usu)) {

        JOptionPane.showMessageDialog(this, "eliminado exitosamente");
        cargarUsuariosTbl();
    } else if (d == JOptionPane.NO_OPTION) {
        cargarUsuariosTbl();
    }

}

}

private void btnListarActionPerformed(java.awt.event.ActionEvent evt) {
    if (controladorUsuario.usuarios() != null) {

        cargarUsuariosTbl();

    }
}

private void txtConfirmarContraseñaActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
}

private void tblUsuariosMouseClicked(java.awt.event.MouseEvent evt) {
    int index = tblUsuarios.getSelectedRow();
    id = Integer.parseInt("'" + tblUsuarios.getValueAt(index, 0));
    String cedula = "'" + tblUsuarios.getValueAt(index, 1);
    String nombre = "'" + tblUsuarios.getValueAt(index, 2);
    String apellido = "'" + tblUsuarios.getValueAt(index, 3);
    String direccion = "'" + tblUsuarios.getValueAt(index, 4);
    String telefono = "'" + tblUsuarios.getValueAt(index, 5);
    String usuario = "'" + tblUsuarios.getValueAt(index, 6);

    txtGcedula.setText(cedula);
    txtGnombre.setText(nombre);
    txtGapellido.setText(apellido);
    txtGDireccion.setText(direccion);
    txtGtelefono.setText(telefono);
    txtGcorreo.setText(usuario);

}

private void formInternalFrameActivated(javax.swing.event.InternalFrameEvent
evt) {
    cargarUsuariosTbl();
}

private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    limpiar();
}
```

### GESTION DE USUARIO

Cedula:

Nombre:

Apellido:

Direccion:

Telefono

Correo

Contraseña

Confirmar contraseña

AGREGAR

MODIFICAR

ELIMINAR

LISTAR

LIMPIAR

SALIR

ID	Cedula	Nombre	Apellido	Direccion	Telefono	Usuario	Contraseña

### Gestión de Clientes Fijos

```

package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorClienteFijo;
import ec.edu.ups.controlador.ControladorEspacios;
import ec.edu.ups.controlador.ControladorIngreso;
import ec.edu.ups.controlador.ControladorTarifa;
import ec.edu.ups.modelo.ClienteFijo;
import ec.edu.ups.modelo.Espacios;
import ec.edu.ups.modelo.Ingresos;
import ec.edu.ups.modelo.Tarifa;
import java.io.IOException;
import java.util.Calendar;
import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author paul_
 */
public final class GestionClientesFijos extends javax.swing.JInternalFrame {

    private int id;
    private final ControladorEspacios controladorEspacios;
    private final ControladorClienteFijo controladorClienteFijo;
    private final ControladorTarifa controladorTarifa;
    private ControladorIngreso controladorIngreso;
    private final Calendar fechaRegistro;
    int anio;
    int mes;
    int dia;

```

```
int numMeses=0;

public GestionClientesFijos(ControladorClienteFijo controladorClienteFijo, ControladorTarifa controladorTarifa, ControladorEspacios controladorEspacios, ControladorIngreso controladorIngreso) {
    initComponents();
    this.controladorEspacios=controladorEspacios;
    this.controladorClienteFijo=controladorClienteFijo;
    this.controladorTarifa=controladorTarifa;
    this.controladorIngreso=controladorIngreso;
    fechaRegistro = Calendar.getInstance();
    anio=fechaRegistro.get(Calendar.YEAR);
    mes=fechaRegistro.get(Calendar.MONTH)+1;
    dia=fechaRegistro.get(Calendar.DATE);

    //controladorEspacios.cargarEspaciosDefault();

    // cargarCbxEspaciosDisp();
}
/**
 * This method is called from within the constructor to initialize the
 * form. WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 * @param anio
 * @return
 */
@SuppressWarnings("unchecked")
public boolean isBisiesto(int anio){

    return ((anio%4)==0 && (anio%100)!=0 || (anio%400)==0);

}

public int dias_mes(int mes, int anio){

    int dias = 31;

    if(mes==4 || mes==6 || mes==9 || mes==11){
        dias=30;
    }else if(mes==2){
        if(isBisiesto(anio)){
            dias=29;
        }else{
            dias=28;
        }
    }

    return dias;

}

public void incrementarMeses(int numMeses){
    int cont=0;
    int dias=(int) (numMeses*30.4375);
    while (cont<=dias){

        if(dia>dias_mes(mes,anio)){
            dia=1;
            mes++;
            if(mes>12){
```

```

        mes=1;
        anio++;
    }
}
dia++;
cont++;
}
}

public Calendar calcularFechaExpiracionPago(double abono, ClienteFijo clienteFijo, Tarifa tarifa) {
    double valorTarifa=0;

    switch (clienteFijo.getTipoVehivulo()) {
        case "Motocicleta":
            valorTarifa=tarifa.getMensualM();
            break;
        case "Automovil":
            valorTarifa=tarifa.getMensualA();
            break;
        default:
            valorTarifa=tarifa.getMensualP();
            break;
    }
    numMeses = (int) (abono / valorTarifa);

    incrementarMeses(numMeses);
    Calendar fechaExp = Calendar.getInstance();
    fechaExp.set(anio, mes-1, dia);
    return fechaExp;
}

public double calcularSaldoAbono (double abono, ClienteFijo clienteFijo, Tarifa tarifa) {
    double valorTarifa=0;
    double saldo=abono;
    switch (clienteFijo.getTipoVehivulo()) {
        case "Motocicleta":
            valorTarifa=tarifa.getMensualM();
            break;
        case "Automovil":
            valorTarifa=tarifa.getMensualA();
            break;
        default:
            valorTarifa=tarifa.getMensualP();
            break;
    }

    while (saldo>=0) {
        if(saldo<valorTarifa) {
            valorTarifa=valorTarifa-saldo;
        }
        saldo=saldo-valorTarifa;
    }

    return saldo+valorTarifa;
}

public void limpiar() {

```

```

        //.setText("");
txtCedula.setText("");
txtNombre.setText("");
txtApellido.setText("");
txtDireccion.setText("");
txtTelefono.setText("");
cbxAutomovil.setSelectedItem(0);
cbxTarifa.setSelectedItem(0);
txtAbono.setText("");
txtEspActual.setText("Ninguno");
/*txtCedula.getText().isEmpty() ||
txtNombre.getText().isEmpty() ||
txtApellido.getText().isEmpty() ||
txtDireccion.getText().isEmpty() ||
txtTelefono.getText().isEmpty() ||
txtAbono.getText().isEmpty() */
}
public boolean validarCampos() {
    if( txtCedula.getText().isEmpty() ||
txtNombre.getText().isEmpty() ||
txtApellido.getText().isEmpty() ||
txtDireccion.getText().isEmpty() ||
txtTelefono.getText().isEmpty() ||
txtAbono.getText().isEmpty())

    {
        JOptionPane.showMessageDialog(this, "CAMPOS VACIOS");
        return false;

    }else{
        return true;
    }

}

public void limpiarCbX() {
    cbxEspacioAsignar.removeAllItems();

}

public void cargarCbxEspaciosDisp() {

    limpiarCbX();

    for(Espacios e: controladorEspacios.espaciosDisponibles()){
        cbxEspacioAsignar.addItem(""+e.getId());
    }
}

public void cargarClientesFijosTbl() {

    String fecha="";
    DefaultTableModel modelo = (DefaultTableModel) tblClientesFijos.getModel();
    modelo.setRowCount(0);
    if (controladorClienteFijo.clientesFijos() != null) {
        for (ClienteFijo cf : controladorClienteFijo.clientesFijos()) {
            Calendar f =cf.getFechaExpiracion();
            int mes1 =f.get(Calendar.MONTH)+1;

```



```

        Calendar fh= Calendar.getInstance();
        ingreso.setFechaHora(fh);
        ingreso.setDescripcion(" Paga multa : Cliente "+cliente.getNombre()+"
"+cliente.getApellido()+" con CI: "+cliente.getCedula()+"");
        ingreso.setIngreso(cliente.getMulta());
        controladorIngreso.crear(ingreso);
    }

    if(controladorEspacios.AsignarEspacio(Integer.parseInt(cbxEspacioAsig-
nar.getSelectedItem().toString().trim()), ""+cliente.getCedula())){

        if(controladorClienteFijo.crear(cliente)){
            Ingresos ingreso = new Ingresos();

            Calendar fh= Calendar.getInstance();
            ingreso.setFechaHora(fh);
            ingreso.setDescripcion("Nuevo Cliente "+cliente.getNombre()+"
"+cliente.getApellido()+" con CI: "+cliente.getCedula());
            ingreso.setIngreso(Double.parseDouble(txtAbono.getText().trim()));
            controladorIngreso.crear(ingreso);
            JOptionPane.showMessageDialog(this, "Cliente registrado con exito");

        }else{
            JOptionPane.showMessageDialog(this, "Error al crear Cliente");
        }
        }else{
            JOptionPane.showMessageDialog(this, "No se pueden ingresar abonos meno-
res al mensual");
        }
        cargarCbxEspaciosDisp();
        cargarClientesFijosTbl();

    }else {
        JOptionPane.showMessageDialog(this, "No existen registro de tarifas");
    }

    }

}

private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
if(validarCampos()){
    Tarifa tarifa=null;
    for (Tarifa tarifa : controladorTarifa.buscarTodo()) {
        tarifa=tarifa;
    }
    ClienteFijo cliente = new ClienteFijo();
    cliente.setTipoVehiculo(cbxAutomovil.getSelectedItem().toString().trim());
    cliente.setTipoTarifa(cbxTarifa.getSelectedItem().toString().trim());
    cliente.setAbono(Double.parseDouble(txtAbono.getText().trim()));
    cliente.setCedula(txtCedula.getText().trim());
    cliente.setNombre(txtNombre.getText().trim()); cliente.setApellido(txtA-
pellido.getText().trim());
    cliente.setDireccion(txtDireccion.getText().trim());
    cliente.setTlf(txtTelefono.getText().trim());
}
}

```

```

        cliente.setFechaExpiracion(calcularFechaExpiracionPago(Double.parseDouble(txtAbono.getText().trim()), cliente, tariffal));

        cliente.setAbono(calcularSaldoAbono(Double.parseDouble(txtAbono.getText().trim()), cliente, tariffal));

        cliente.setEspacioParqueo(Integer.parseInt(cbxEspacioAsignar.getSelectedItem().toString().trim()));

        double multa= controladorClienteFijo.generarMulta(cliente);
        cliente.setMulta(multa);

        if(cliente.getMulta()!=0){
            Ingresos ingreso = new Ingresos();
            Calendar fh= Calendar.getInstance();
            ingreso.setFechaHora(fh);
            ingreso.setDescripcion(" Paga multa : Cliente "+cliente.getNombre()+" "+cliente.getApellido()+" con CI: "+cliente.getCedula()+"");
            ingreso.setIngreso(cliente.getMulta());
            controladorIngreso.crear(ingreso);
        }
        if(controladorClienteFijo.actualizar(cliente)){
            Ingresos ingreso = new Ingresos();
            Calendar fh= Calendar.getInstance();
            ingreso.setFechaHora(fh);
            ingreso.setDescripcion(" Cliente Abona "+cliente.getNombre()+" "+cliente.getApellido()+" con CI: "+cliente.getCedula());
            ingreso.setIngreso(Double.parseDouble(txtAbono.getText().trim()));
            controladorIngreso.crear(ingreso);
            controladorEspacios.AsignarEspacio(Integer.parseInt(txtEspActual.getText().trim()), "");
            if(controladorEspacios.AsignarEspacio(Integer.parseInt(cbxEspacioAsignar.getSelectedItem().toString().trim()), txtCedula.getText().trim())){

            }

            JOptionPane.showMessageDialog(this, "Datos actualizados con exito");

        }
        cargarCbxEspaciosDisp();
        cargarClientesFijosTbl();
    }

    private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
        if(validarCampos()){
            int d = JOptionPane.showConfirmDialog(this, "Esta seguro de eliminar el cliente con cedula " + txtCedula.getText().trim());
            if (d == JOptionPane.YES_OPTION) {

                ClienteFijo c =controladorClienteFijo.buscar(id);

                if(controladorClienteFijo.eliminar(c)){

                    controladorEspacios.AsignarEspacio(Integer.parseInt(txtEspActual.getText().trim()), "");
                }
            }
        }
    }

```



```
JOptionPane.showMessageDialog(this, "eliminado exitosamente");
}else{
    JOptionPane.showMessageDialog(this, "no se pudo eliminar ERROR");
}
}

}
}
private void btnListarActionPerformed(java.awt.event.ActionEvent evt) {
    cargarCbxEspaciosDisp();
    cargarClientesFijosTbl();
}

private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    limpiar();
}

private void txtCedulaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void cbxEspacioAsignarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void tblClientesFijosMouseClicked(java.awt.event.MouseEvent evt) {

    int index = tblClientesFijos.getSelectedRow();
    id = Integer.parseInt("" + tblClientesFijos.getValueAt(index, 0));
    String cedula = "" + tblClientesFijos.getValueAt(index, 1);
    String nombre = "" + tblClientesFijos.getValueAt(index, 2);
    String apellido = "" + tblClientesFijos.getValueAt(index, 3);
    String direccion = "" + tblClientesFijos.getValueAt(index, 4);
    String telefono = "" + tblClientesFijos.getValueAt(index, 5);
    String tipoVehiculo = ""+tblClientesFijos.getValueAt(index, 6);
    String tipoTarifa = ""+tblClientesFijos.getValueAt(index, 7);
    double abono=Double.parseDouble(""+tblClientesFijos.getValueAt(index, 8));

    if(tipoVehiculo.trim().equals("Automovil")){
        cbxAutomovil.setSelectedItem(0);

    }else if(tipoVehiculo.trim().equals("Motocicleta")){
        cbxAutomovil.setSelectedItem(1);
    }else{
        cbxAutomovil.setSelectedItem(2);
    }
    if(tipoTarifa.trim().equals("Mensual")){
        cbxTarifa.setSelectedItem(0);
    }
    txtCedula.setText(cedula);
    txtNombre.setText(nombre);
    txtApellido.setText(apellido);
    txtDireccion.setText(direccion);
    txtTelefono.setText(telefono);
    txtAbono.setText(""+abono);

    ClienteFijo c = controladorClienteFijo.buscar(id);
    txtEspActual.setText(""+c.getEspacioParqueo());
}
```

```

    }

    private void formInternalFrameActivated(javax.swing.event.InternalFrameEvent
    evt) {
        cargarCbxEspaciosDisp();
        cargarClientesFijosTbl();
    }

```

### Factura de Clientes Momentáneos

```

package ec.edu.ups.vista;

import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.PageSize;
import com.lowagie.text.pdf.PdfContentByte;
import com.lowagie.text.pdf.PdfTemplate;
import com.lowagie.text.pdf.PdfWriter;
import ec.edu.ups.controlador.ControladorFacturaClienteM;
import ec.edu.ups.controlador.ControladorTicketCliente;
import ec.edu.ups.modelo.TicketClienteMomentaneo;
import ec.edu.ups.modelo.FacturaClienteM;
import java.awt.Desktop;
import java.awt.Graphics2D;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Calendar;

import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author paul

```

```

*/
public class FacturaClienteMo extends javax.swing.JInternalFrame {

    private int id;
    private ControladorTicketCliente controladorTicket;
    private ControladorFacturaClienteM controladorFacturaClienteM;

    public FacturaClienteMo(ControladorFacturaClienteM controladorFacturaClienteM,
        ControladorTicketCliente controladorTicket) {
        initComponents();

        this.controladorFacturaClienteM = controladorFacturaClienteM;
        this.controladorTicket = controladorTicket;
    }

    private void txtNombresActionPerformed(java.awt.event.ActionEvent evt) {

        private void btnImprimirActionPerformed(java.awt.event.ActionEvent evt) {
            //int id, Calendar Fecha, String nombres, String direccion, String cedula,
            String telefono, TicketClienteMomentaneo ticket
            FacturaClienteM factura = new FacturaClienteM();
            id++;
            factura.setCedula(txtCedula.getText().trim());
            factura.setNombres(txtNombres.getText().trim());
            factura.setDireccion(txtDireccion.getText().trim());
            factura.setTelefono(txtTelefono.getText().trim());

            TicketClienteMomentaneo ticket = controladorTicket.buscar(Integer.parseInt(txtNumTicket.getText().trim()));

            factura.setTicket(ticket);
            if (controladorFacturaClienteM.crear(factura)) {
                JOptionPane.showMessageDialog(this, "Factura generada exitosamente");
            }

            imprimir();
            abrirarchivo("datos/Factura" + id + ".pdf");
        }

        public void imprimir() {

            float x = 1;
            float y = 1;
            Document document = new Document(PageSize.A4.rotate());
            try {
                PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("datos/Factura" + id + ".pdf"));

                document.open();
                PdfContentByte cb = writer.getDirectContent();

                cb.saveState();

                PdfTemplate pdfTemplate = cb.createTemplate(this.getWidth(),
                    this.getHeight());
                Graphics2D g2 = pdfTemplate.createGraphics(this.getWidth(),
                    this.getHeight());
                /*g2.setColor(Color.BLACK);
                g2.drawRect(x-2, y-2, table.getWidth()+2, table.getHeight()+2);*/

```

```

        this.print(g2);
        System.out.println("x=" + x + "," + "y=" + y);
        cb.addTemplate(pdfTemplate, x, y);
        g2.dispose();
        cb.restoreState();
    } catch (DocumentException | FileNotFoundException e) {
        System.err.println(e.getMessage());
    }
    document.close();
}

public void abrirarchivo(String archivo) {

    try {

        File objetofile = new File(archivo);
        Desktop.getDesktop().open(objetofile);

    } catch (IOException ex) {

        System.out.println(ex);

    }
}

private void btnCargarDatosActionPerformed(java.awt.event.ActionEvent evt) {
    if (txtNumTicket.getText().isEmpty()) {

    } else {

        TicketClienteMomentaneo ticket = controladorTicket.buscar(Integer.parseInt(txtNumTicket.getText().trim()));

        if (ticket != null) {

            String fecha = "";
            DefaultTableModel modelo = (DefaultTableModel) tblFactura.getModel();
            modelo.setRowCount(0);

            Calendar f = ticket.getFechaHoraIngreso();
            int mes = f.get(Calendar.MONTH) + 1;
            fecha = f.get(Calendar.DAY_OF_MONTH) + "/" + mes + "/" + f.get(Calendar.YEAR);

            Object[] rowData = {ticket.getId(), " Tarifa por: " + controladorTicket.calcularIntervaloTiempo(ticket) + " minutos ", ticket.getTarifa(), ticket.getTarifa()};
            modelo.addRow(rowData);
            tblFactura.setModel(modelo);

        } else {
            System.out.println("LISTA VACIA");
        }
        double subtotal = ticket.getTarifa() - (ticket.getTarifa() * 0.12);

        double iva = (ticket.getTarifa() * 0.12);

        txtSubtotal.setText("" + subtotal);
        txtIva.setText("" + iva);
        txtTotalF.setText("" + ticket.getTarifa());
        Calendar f = Calendar.getInstance();
    }
}

```

```
int mes = f.get(Calendar.MONTH);

String fechaA = "" + f.get(Calendar.DAY_OF_MONTH) + "/" + mes + "/" +
f.get(Calendar.YEAR);

txtFecha.setText(fechaA);
}

}
```

**DATOS DEL USUARIO**

Fecha

Sr. (a)

Dirección:

Cédula:

Teléfono

Num. Ticket

Cant	Descripción	Precio U.	Precio Total

**Subtotal:**

**I.V.A 12%:**

**Total:**

### Factura de Clientes Fijos

```
package ec.edu.ups.vista;

import com.lowagie.text.Document;
import com.lowagie.text.PageSize;
import com.lowagie.text.pdf.PdfContentByte;
import com.lowagie.text.pdf.PdfTemplate;
import com.lowagie.text.pdf.PdfWriter;
import ec.edu.ups.controlador.ControladorClienteFijo;
import ec.edu.ups.controlador.ControladorFacturaClienteFijo;
import ec.edu.ups.modelo.ClienteFijo;
import ec.edu.ups.modelo.FacturaClienteFijo;
import java.awt.Desktop;
import java.awt.Graphics2D;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Calendar;

import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author paul_
 */
public class FacturaClienteFijos extends javax.swing.JInternalFrame {
```

```

private int id;
private ControladorFacturaClienteFijo controladorFacturaClienteFijo;
private ControladorClienteFijo controladorClienteFijo;

public FacturaClienteFijos(ControladorFacturaClienteFijo controladorFactura-
ClienteFijo, ControladorClienteFijo controladorClienteFijo) {
    initComponents();
    this.controladorFacturaClienteFijo = controladorFacturaClienteFijo;
    this.controladorClienteFijo = controladorClienteFijo;
}

private void btnImprimirActionPerformed(java.awt.event.ActionEvent evt) {
    //int id, Calendar Fecha, String nombres, String direccion, String cedula,
    String telefono, TicketClienteMomentaneo ticket
    FacturaClienteFijo factura = new FacturaClienteFijo();

    id = id+1;
    ClienteFijo clienteFijo = controladorClienteFijo.buscarCliente(txtCe-
dula.getText().trim());
    factura.setClientefijo(clienteFijo);

    Calendar f = Calendar.getInstance();
    int mes = f.get(Calendar.MONTH);
    String fecha = f.get(Calendar.DAY_OF_MONTH) + "/" + mes + "/" + f.get(Calen-
dar.YEAR);
    factura.setDescripcion("Abono Mensual por espacio en el parqueadero" + "(" +
fecha + ")" + "" + clienteFijo.getAbono() + "" + clienteFijo.getAbono());

    if (controladorFacturaClienteFijo.crear(factura)) {
        JOptionPane.showMessageDialog(this, "Factura generada exitosamente");
    }

    imprimir();
    abrirarchivo("datos/Factura" + id + ".pdf");
}

public void imprimir() {

    float x = 1;
    float y = 1;
    Document document = new Document(PageSize.A4.rotate());
    try {
        PdfWriter writer = PdfWriter.getInstance(document, new FileOut-
putStream("datos/Factura" + id + ".pdf"));

        document.open();
        PdfContentByte cb = writer.getDirectContent();

        cb.saveState();

        PdfTemplate pdfTemplate = cb.createTemplate(this.getWidth(),
this.getHeight());
        Graphics2D g2 = pdfTemplate.createGraphics(this.getWidth(),
this.getHeight());
        /*g2.setColor(Color.BLACK);
g2.drawRect(x-2, y-2, table.getWidth()+2, table.getHeight()+2);*/
        this.print(g2);
        // System.out.println("x=" + x + "," + "y=" + y);
        cb.addTemplate(pdfTemplate, x, y);
        g2.dispose();
        cb.restoreState();
    } catch (Exception e) {

```

```
        System.err.println(e.getMessage());
    }
    document.close();

}

public void abrirarchivo(String archivo) {

    try {

        File objetofile = new File(archivo);
        Desktop.getDesktop().open(objetofile);

    } catch (IOException ex) {

        System.out.println(ex);

    }
}

private void btnCargarDatosActionPerformed(java.awt.event.ActionEvent evt) {
    if (txtCedula.getText().isEmpty()) {

    } else {
        ClienteFijo cliente;
        cliente = controladorClienteFijo.buscarCliente(txtCedula.getText().trim());

        if (cliente != null) {

            String fecha = "";
            DefaultTableModel modelo = (DefaultTableModel) tblFactura.getModel();
            modelo.setRowCount(0);

            Calendar f = Calendar.getInstance();

            fecha = f.get(Calendar.DAY_OF_MONTH) + "/" + f.get(Calendar.MONTH) +
            "/" + "/" + f.get(Calendar.YEAR);
            Object[] rowData = {cliente.getId(), "Abono Mensual por espacio en
            el parqueadero" + "(" + fecha + ")", cliente.getAbono(), cliente.getAbono()};
            modelo.addRow(rowData);
            tblFactura.setModel(modelo);

        } else {
            System.out.println("LISTA VACIA");
        }
        double subtotal = cliente.getAbono() - (cliente.getAbono() * 0.12);

        double iva = (cliente.getAbono() * 0.12);

        txtSubtotal.setText("" + subtotal);
        txtIva.setText("" + iva);
        txtTotalF.setText("" + cliente.getAbono());
        Calendar f = Calendar.getInstance();

        String fechaA = "" + f.get(Calendar.DATE) + "/" + f.get(Calendar.MONTH)
        + "/" + f.get(Calendar.YEAR);

        txtFecha.setText(fechaA);
    }
}
```

```

        txtNombres.setText(cliente.getNombre() + " " + cliente.getApellido());

        txtDireccion.setText(cliente.getDireccion());

        txtTelefono.setText(cliente.getTlf());
    }

}

private void txtNombresActionPerformed(java.awt.event.ActionEvent evt) {
}

```

**DATOS DEL CLIENTE**

**Fecha**

**Sr. (a)**

**Direccion:**

**Cedula:**

**Telefono**

Cant	Descripcion	Precio U.	Precio Total

**Subtotal:**  
**I.V.A 12%:**  
**Total:**

### Consulta de multa

```

package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorClienteFijo;
import ec.edu.ups.controlador.ControladorEspacios;
import ec.edu.ups.controlador.ControladorTicketCliente;
import ec.edu.ups.modelo.ClienteFijo;
import ec.edu.ups.modelo.Espacios;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author paul_
 */
public class ConsultaMulta extends javax.swing.JInternalFrame {

    private ControladorEspacios controladorEspacios;
    private ControladorClienteFijo controladorClienteFijo;
    private ControladorTicketCliente controladorTicketCliente;

    public ConsultaMulta(ControladorEspacios controladorEspacios, ControladorClien-
teFijo controladorClienteFijo, ControladorTicketCliente controladorTicketCliente) {
        initComponents();
    }
}

```



```

        this.controladorEspacios = controladorEspacios;
        this.controladorClienteFijo = controladorClienteFijo;
        this.controladorTicketCliente = controladorTicketCliente;
    }
    private void btnConsultarActionPerformed(java.awt.event.ActionEvent evt) {
        if (txtConsultal.getText().isEmpty()) {
            JOptionPane.showMessageDialog(this, "CAMPO VACIO");
        } else {
            Espacios e= controladorEspacios.buscar(Integer.parseInt(txtConsultal.getText().trim()));

            if (e.getNombre().equals("")) {
                JOptionPane.showMessageDialog(this, "El espacio solicitado no cuenta con algún tipo de servicio de arrendamiento o multa");
            } else {

                ClienteFijo a = controladorClienteFijo.buscarCliente(e.getNombre());

                if (a != null) {
                    JOptionPane.showMessageDialog(this, "El espacio solicitado esta ocupado por " + a.getNombre() + " " + a.getApellido());

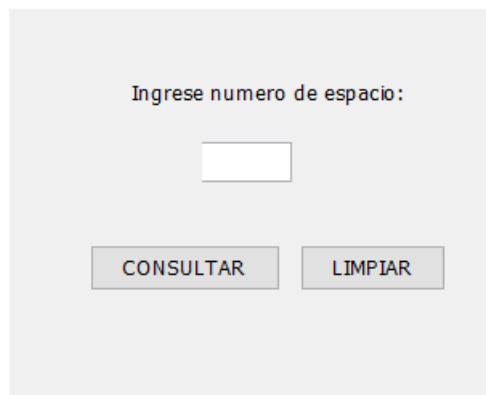
                    JOptionPane.showMessageDialog(this, "Tiene una multa de " + a.getMulta());
                } else {

                    JOptionPane.showMessageDialog(this, "El espacio solicitado esta ocupado por " + e.getNombre() + " ");
                }
            }
        }
    }

    private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void txtConsultalActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

```



Ingrese numero de espacio:

CONSULTAR LIMPIAR

## Cientes Fijos Puesto

```
package ec.edu.ups.vista;

import ec.edu.ups.controlador.ControladorClienteFijo;
import ec.edu.ups.controlador.ControladorHistorialClientesF;
import ec.edu.ups.modelo.ClienteFijo;
import ec.edu.ups.modelo.HistorialClientesFijos;
import java.io.IOException;
import java.util.Calendar;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author paul_
 */
public class ClientesFijosPuesto extends javax.swing.JInternalFrame {

    private ControladorClienteFijo controladorClienteFijo;

    private ControladorHistorialClientesF controladorHistorial;

    public ClientesFijosPuesto(ControladorClienteFijo controladorClienteFijo, ControladorHistorialClientesF controladorHistorial) {
        initComponents();
        this.controladorClienteFijo = controladorClienteFijo;
        this.controladorHistorial = controladorHistorial;
    }

    public void cargarClientesFijosTbl() {

        String fecha = "";
        DefaultTableModel modelo = (DefaultTableModel) tblPuestos.getModel();
        modelo.setRowCount(0);
        if (controladorHistorial.listaHistorialCF() != null) {
            for (HistorialClientesFijos cf : controladorHistorial.listaHistorialCF()) {
                Calendar f = cf.getFechaHora();
                int mes = (f.get(Calendar.MONTH)) + 1;
                fecha = f.get(Calendar.DAY_OF_MONTH) + "/" + mes + "/" + f.get(Calendar.YEAR);
                Object[] rowData = {cf.getId(), cf.getClienteF().getCedula(), cf.getClienteF().getNombre(), cf.getClienteF().getApellido(), cf.getClienteF().getTipoVehiculo(), cf.getDescripcion()};
                modelo.addRow(rowData);
                tblPuestos.setModel(modelo);
            }
        } else {
            System.out.println("LISTA VACIA");
        }
    }

    private void btnIngresarActionPerformed(java.awt.event.ActionEvent evt) {
        Calendar fecha = Calendar.getInstance();
        int mes = (fecha.get(Calendar.MONTH)) + 1;
        String f = "" + fecha.get(Calendar.DAY_OF_MONTH) + "/" + mes + "/" + fecha.get(Calendar.YEAR);
    }
}
```

```
String h = "" + fecha.get(Calendar.HOUR) + ":" + (fecha.get(Calendar.MI-
NUTE)) + ":" + fecha.get(Calendar.SECOND);
    if (txtCedula.getText().isEmpty()) {
        JOptionPane.showMessageDialog(this, "CAMPO VACIO");
    } else {
        ClienteFijo clienteFijo = controladorClienteFijo.buscarCliente(txtCe-
dula.getText().trim());
        if (clienteFijo != null) {
            clienteFijo.setEstado(false);
            HistorialClientesFijos historial = new HistorialClientesFi-
jos();//fecha,"Entra cliente "+ " con cedula:" +clienteFijo.getCedula()+" el "+f+"
"+h,clienteFijo);
            historial.setFechaHora(fecha);
            historial.setDescripcion("Entra cliente " + " con cedula:" + clien-
teFijo.getCedula() + " el " + f + " " + h);
            historial.setClienteF(clienteFijo);
            //int id, Calendar FechaHora, String descripcion, ClienteFijo clien-
teF

            if (controladorHistorial.crear(historial)) {

                JOptionPane.showMessageDialog(this, "El Cliente Ingreso al Par-
queadero");
                if (controladorClienteFijo.actualizar(clienteFijo)) {
                    System.out.println("SE ACTUALIZO ESTADO DEL CLIENTE CUANDO
INGRESO AL PARQUEADERO");
                }
            }
        }

        cargarClientesFijosTbl();
    }

private void btnIngresarActionPerformed(java.awt.event.ActionEvent evt) {
    Calendar fecha = Calendar.getInstance();
    int mes = (fecha.get(Calendar.MONTH)) + 1;
    String f = "" + fecha.get(Calendar.DAY_OF_MONTH) + "/" + mes + "/" +
fecha.get(Calendar.YEAR);
    String h = "" + fecha.get(Calendar.HOUR) + ":" + (fecha.get(Calendar.MI-
NUTE)) + ":" + fecha.get(Calendar.SECOND);
    if (txtCedula.getText().isEmpty()) {
        JOptionPane.showMessageDialog(this, "CAMPO VACIO");
    } else {
        ClienteFijo clienteFijo = controladorClienteFijo.buscarCliente(txtCe-
dula.getText().trim());
        if (clienteFijo != null) {
            clienteFijo.setEstado(true);
            // HistorialClientesFijos historial = new HistorialClientesFijos(fe-
cha,"Sale cliente "+ " con cedula:" +clienteFijo.getCedula()+" el "+f+" "+h,cliente-
Fijo);
            HistorialClientesFijos historial = new HistorialClientesFijos();
            historial.setFechaHora(fecha);
            historial.setDescripcion("Sale cliente " + " con cedula:" + cliente-
Fijo.getCedula() + " el " + f + " " + h);
            historial.setClienteF(clienteFijo);
            //int id, Calendar FechaHora, String descripcion, ClienteFijo clien-
teF

            if (controladorHistorial.crear(historial)) {
```

```

        JOptionPane.showMessageDialog(this, "El Cliente Salio del Par-
queadero ");
        if (controladorClienteFijo.actualizar(clienteFijo)) {
            System.out.println("SE ACTUALIZO ESTADO DEL CLIENTE CUANDO
INGRESO AL PARQUEADERO");
        }
    }
    cargarClientesFijosTbl();
}

private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    txtCedula.setText("");
    cargarClientesFijosTbl();
}

private void formInternalFrameActivated(javax.swing.event.InternalFrameEvent
evt) {
    cargarClientesFijosTbl();
}

private void tblPuestosMouseClicked(java.awt.event.MouseEvent evt) {
}

```

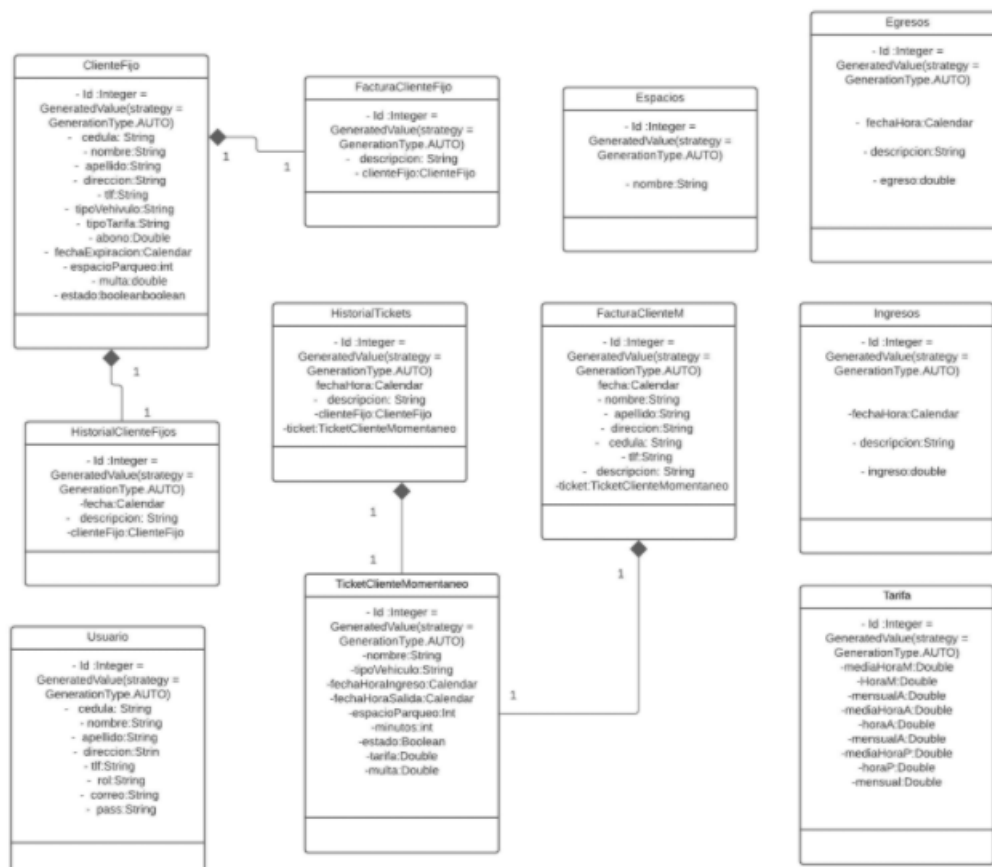
**Cientes**  
CEDULA:

**Informacion**  
Fecha de Ingreso   
Hora de Ingreso:

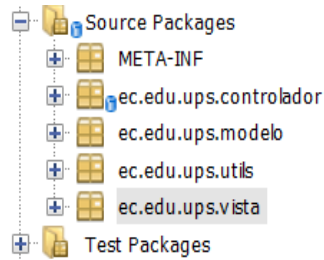
INGRESAR RETIRAR LIMPIAR

**Historial Clientes Fijos**

ID	Cedula	Nombre	Apellido	Tipo de Vehiculo	Descripcion



## 5- Arquitectura del sistema.



## 6- Requerimientos de HW y SW.

Requerimientos de hardware		Requerimientos de software
Si en el caso el programa se implementa en su funcionamiento. Se necesitaria un monitor, un mouse y un teclado, ademas de una impresora en la que se imprimirian las caracteristicas del ticket de parqueo junto con los datos del usuario o cliente que le dio uso.	Y	<ul style="list-style-type: none"><li>- JRE (Java runtime environment)</li><li>- JDK (Java desktop kit)</li><li>- JMF (Java media framework)</li><li>- NetBeans version 11.3.</li><li>- Windows, Linux.</li></ul>
+info		+info

### RESULTADO(S) OBTENIDO(S):

- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.

### CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
- Los estudiantes implementan soluciones graficas en sistemas.
- Los estudiantes están en la capacidad de implementar la persistencia en base de datos.

### RECOMENDACIONES:

- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.
- **Consultar con el docente las dudas que puedan surgir al momento de realizar la práctica.**

**Nombre de estudiante:**

PAUL ALEXANDER GUAPUCAL CARDENAS

**Firma de estudiante:**