Universidad Técnica Particular de Loja

Trabajo final del 2do.Bimestre

Tema

Gestión de telefonía móvil estudiantil: Mov-UTPL.

Docente

Wayner Bustamante Granda

Materia

Programación Orientada a Objetos

Integrantes

Fernando Muñoz

Rafaella Palacios Hidalgo

Fecha

24-07-2024

1. **Objetivo General**

Aplicación los pilares de la POO: herencia y polimorfismo, generando resultados desde/hacia DB.

2. **Objetivos Especificos**

- Facilitar las operaciones CRUD de los clientes y/o planes de la telefónica.
- Generación de las facturas por cliente/plan, en base a los lineamientos anteriores.
- El motor de DB debe ser SQLite, para todo el sistema (Clientes, planes, facturas, etc.)
- Documentar de forma oportuna su análisis y diseño en UML, aplicando de forma oportuna y eficiente la herencia y el polimorfismo.
- Se debe aplicar la arquitectura MVC, considerando sus principios de diseño.

3. **Problemática**

La carrera de Telecomunicación-UTPL a impulsado un proyecto de comunicación móvil para la comunidad Universitaria, por cuanto dispone de toda la infraestructura tecnológica, pero se encuentra detenida por que requiere de un Sistema de Gestión de sus clientes y facturación, con ciertos planes celulares según la siguiente información:

4. **Análisis**

- **Datos básicos del cliente**
  Nombre
  Apellido
  Cedula
  Ciudad
  Marca
  Modelo
  Numero
  Correo
  Plan
  PagoMensual

- **Tipos de planes móviles**

    1. **PlanPostPagoMinutosMegasEconomico**
       Este plan almacena la siguiente información: minutos, costo minutos, megas expresados en gigas, costo por cada gigas, porcentaje de descuento.

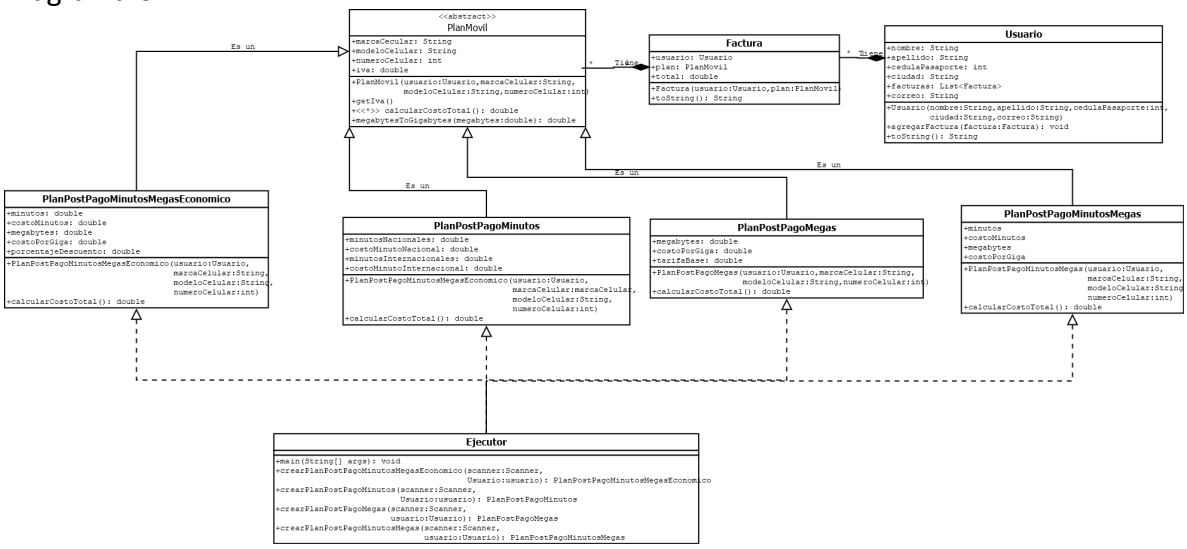
    2. **PlanPostPagoMinutos**
       Este plan almacena la siguiente información: minutos nacionales, costo minuto nacional, minutos internacionales, costo minuto internacional.

3. **PlanPostPagoMegas**
   Informacion del plan: megas expresados en gigas, costo por cada giga, tarifa base.

4. **PlanPostPagoMinutosMegas**
   Inofrmacion del plan: minutos, costo minutos, megas expresados en gigas, costo por cada giga.

- **PlanMovil**
  De esta clase se van a derivar los diferntes planes
  cliente
  marcaCelular
  modeloCelular
  numeroCelular
  iva

5. **Diseño**
   Diagrama UML



6. **Codificacion**
   Codificacion de las operaciones CRUD
   Metodo para ingresar el Cliente

```
public void insertarCliente(cliente cliente) {
    String sql = "INSERT INTO clientes(nombre, apellido, cedula, ciudad, marca, modelo, numero, correo, plan, pagoMensual) VALUES(?

    try (Connection conn = establecerConeccion.connect();
         PreparedStatement pstmt = conn.prepareStatement(string: sql)) {

        pstmt.setString(i: 1, string: cliente.getNombre());
        pstmt.setString(i: 2, string: cliente.getApellido());
        pstmt.setString(i: 3, string: cliente.getCedula());
        pstmt.setString(i: 4, string: cliente.getCiudad());
        pstmt.setString(i: 5, string: cliente.getMarca());
        pstmt.setString(i: 6, string: cliente.getModelo());
        pstmt.setString(i: 7, string: cliente.getNumero());
        pstmt.setString(i: 8, string: cliente.getCorreo());
        pstmt.setString(i: 9, string: cliente.getPlan().toString());
        pstmt.setDouble(i: 10, d: cliente.getPagoMensual());

        pstmt.executeUpdate();
        System.out.println(x: "Cliente añadido exitosamente.");
    } catch (SQLException e) {
        System.out.println(x: e.getMessage());
    }
}
```

Metodo para actualizar al cliente

```
public void actualizarCliente(cliente cliente) {
    String sql = "UPDATE clientes SET nombre = ?, apellido = ?, ciudad = ?, marca = ?, modelo = ?, numero = ?, correo = ?

    try (Connection conn = establecerConeccion.connect();
         PreparedStatement pstmt = conn.prepareStatement(string: sql)) {

        pstmt.setString(i: 1, string: cliente.getNombre());
        pstmt.setString(i: 2, string: cliente.getApellido());
        pstmt.setString(i: 3, string: cliente.getCiudad());
        pstmt.setString(i: 4, string: cliente.getMarca());
        pstmt.setString(i: 5, string: cliente.getModelo());
        pstmt.setString(i: 6, string: cliente.getNumero());
        pstmt.setString(i: 7, string: cliente.getCorreo());
        pstmt.setString(i: 8, string: cliente.getPlan().toString());
        pstmt.setDouble(i: 9, d: cliente.getPagoMensual());
        pstmt.setString(i: 10, string: cliente.getCedula());

        pstmt.executeUpdate();
        System.out.println(x: "Cliente actualizado exitosamente.");
    } catch (SQLException e) {
        System.out.println(x: e.getMessage());
    }
}
```

Metodo para eliminar al cliente

```
public void eliminarCliente(String cedula) {
    String sql = "DELETE FROM clientes WHERE cedula = ?";

    try (Connection conn = establecerConeccion.connect();
         PreparedStatement pstmt = conn.prepareStatement(string: sql)) {

        pstmt.setString(i: 1, string: cedula);
        pstmt.executeUpdate();
        System.out.println(x: "Cliente eliminado exitosamente.");
    } catch (SQLException e) {
        System.out.println(x: e.getMessage());
    }
}
```

Metodo para obtener los clientes

```java
public List<cliente> obtenerTodosClientes() {
    List<cliente> clientes = new ArrayList<>();
    String sql = "SELECT * FROM clientes";

    try (Connection conn = establecerConeccion.connect();
         PreparedStatement pstmt = conn.prepareStatement(string: sql);
         ResultSet rs = pstmt.executeQuery()) {

        while (rs.next()) {
            String nombre = rs.getString(string: "nombre");
            String apellido = rs.getString(string: "apellido");
            String cedula = rs.getString(string: "cedula");
            String ciudad = rs.getString(string: "ciudad");
            String marca = rs.getString(string: "marca");
            String modelo = rs.getString(string: "modelo");
            String numero = rs.getString(string: "numero");
            String correo = rs.getString(string: "correo");
            String planString = rs.getString(string: "plan");
            double pagoMensual = rs.getDouble(string: "pagoMensual");

            PlanMovil plan = deserializarPlan(planString, nombre, apellido, cedula, ciudad, marca, modelo, numero);

            clientes.add(new cliente(nombre, apellido, cedula, ciudad, marca, modelo, numero, correo, plan, pagoMensual));

        }
    } catch (SQLException e) {
        System.out.println(x: e.getMessage());
    }
    return clientes;
}
```

7. Resultados

Table: clientes — Filter in any column

	id	numero	nombre	apellido	cedula	ciudad	marca	modelo	correo	plan	pagoMensual
	...	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	0944366334	Rafa	Palacios	1103647566	Loja	iphone	15	rafa@gmail.com	PlanPostPago{usuario=null, marcaCelular='iphone', ...	403.2
2	...	1	1	1	1	1	1	1	1	PlanPostPago{usuario=null, marcaCelular='1', ...	1.1098828125

```
Factura
Usuario: Pablo Gonzalez
Cédula/Pasaporte: 1147373775
Ciudad: Cuenca
Marca Celular: Iphone
Modelo Celular: 8
Número Celular: 94664664
Total: $172,29
--------------------------
```