

**UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA**

**Carrera de ciencias de la Computación**

**Informe Proyecto Bimestral 2**

**Fecha:** 15/07/2025

**Nombres:** Israel Calle, Andrés Yaguachi

**Docente:** Wayner Granda

**Análisis**

1.Contexto del problema.

La carrera de ingeniería en telecomunicaciones de la UTPL ha iniciado un proyecto de comunicación móvil para la comunidad universitaria, aprovechando su infraestructura tecnológica. Sin embargo, el desarrollo se encuentra detenido por falta de un sistema informático para gestionar:

* Clientes
* Planes Móviles
* Facturación Mensual

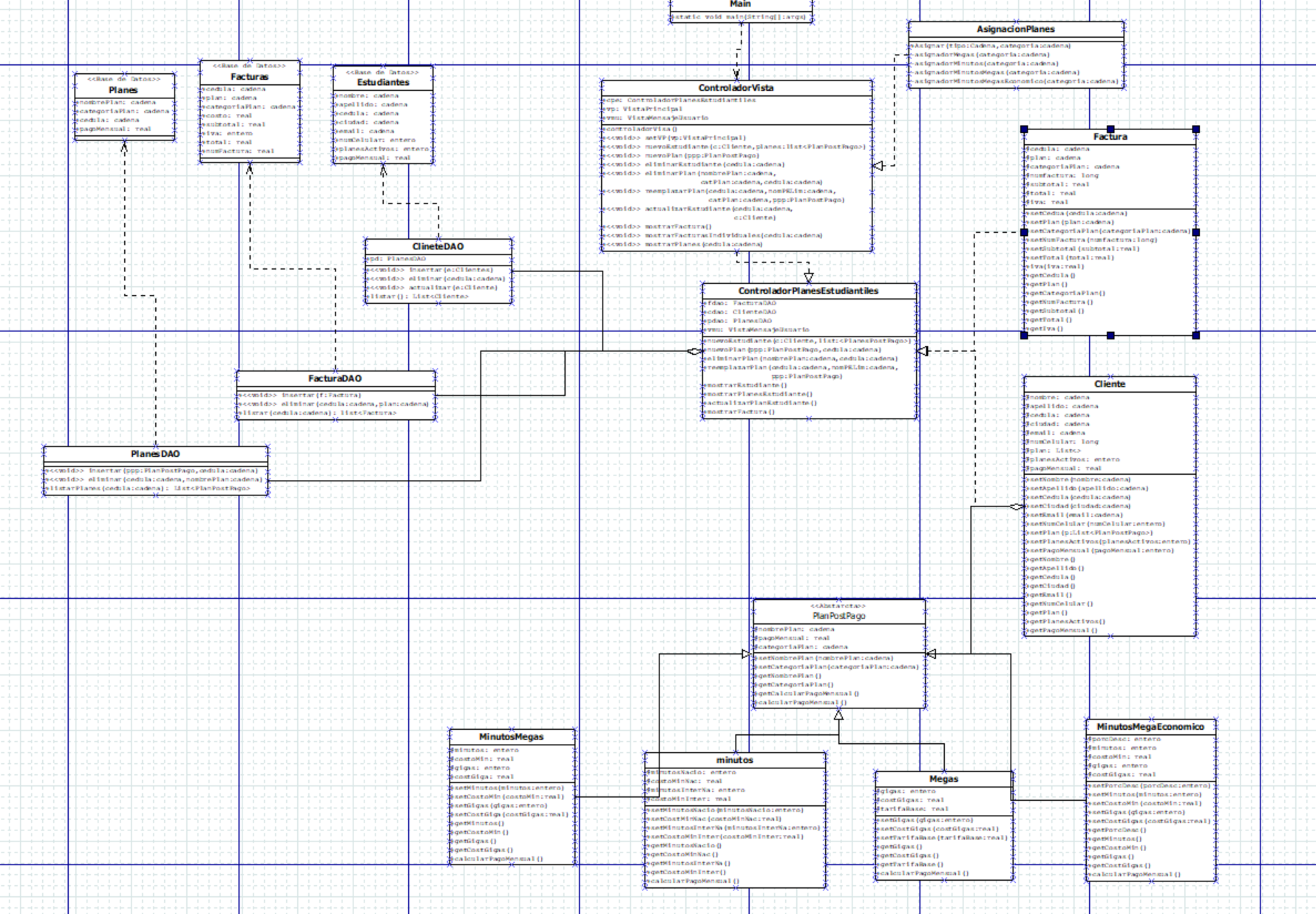
2.Objetivo Del Sistema.

Desarrollar un sistema computacional basado en POO, utilizando herencia, polimorfismo y base de datos SQLite bajo arquitectura MVC que permita resolver la problemática del proyecto planteado, el sistema debe realizar las operaciones CRUD sobre los clientes y planes móviles, así como la generación de facturas.

3.Requisitos fundamentales.

|  |  |
| --- | --- |
| Requisitos | Descripción |
| 1 | Registrar, modificar, eliminar y consultar clientes. |
| 2 | Asignar hasta dos planes por cliente. |
| 3 | Registrar distintos tipos de planes móviles. |
| 4 | Calcular el valor mensual según el tipo de plan. |
| 5 | Generar facturas automáticas por cliente y plan. |
| 6 | Almacenar toda la información en SQLite. |

Análisis Diagrama modelado en Dia



**1. Paquete Modelo**

**Clases de Dominio**

* **Cliente**
  + Atributos: cédula, nombre, apellido, dirección, carrera, etc.
  + Métodos: constructores, setters y getters.
* **Factura**
  + Atributos: código, cédula, fecha, subtotal, iva, total, tipo de plan.
  + Métodos: constructores, setters, getters.
* **Planes**
  + Atributos: código, tipo, costo base.
  + Métodos: constructores, setters, getters.
* **Minutos**, **Megas**
  + Atributos: tiempo o datos, costo, consumo.
  + Métodos: constructores, setters, getters.
* **MinutosMegas**, **MinutosMegaEconomico**
  + Parecen ser subtipos de planes empaquetados, donde se incluyen megas y minutos.
  + **MinutosMegaEconomico** tiene adicionalmente un descuento.
* **PlanPostPago**
  + Clase abstracta o genérica de la que heredan otras como MinutosMegas, MinutosMegaEconomico, Minutos, Megas.

**2. DAO**

* **ClienteDAO**, **FacturaDAO**, **PlanesDAO**
  + Son responsables de la comunicación con la base de datos.
  + Métodos típicos:
    - guardar, listar, buscar, actualizar.
  + Interactúan con las tablas Estudiantes, Facturas, Planes en la base de datos.

**3. Controladores**

* **ControladorVista**
  + Maneja la interacción de usuario.
  + Métodos como verMenu(), ingresarCliente(), mostrarFactura(), etc.
* **ControladorPlanesEstudiantes**
  + Se encarga de la lógica de asignación de planes y estudiantes.
  + Funciones para registrar planes, guardar facturas, etc.
* **AsignadorPlanes**
  + Clase especializada para asignar dinámicamente diferentes tipos de planes a los clientes.

**4. Vista / Main**

* **Main**
  + Punto de entrada del sistema.
  + Inicia la interfaz (probablemente consola o GUI).

**5. Tablas de Base de Datos (Representadas como clases)**

* **Estudiantes**, **Facturas**, **Planes**
  + Tienen los mismos atributos que sus clases correspondientes del modelo.
  + Representan la persistencia de los datos en la base de datos.

**Relaciones**

* **Generalización (Herencia):**
  + Minutos, Megas, MinutosMegas, MinutosMegaEconomico heredan de PlanPostPago.
* **Asociaciones:**
  + Cliente está asociado con Factura.
  + Factura con PlanPostPago.
* **Dependencias:**
  + Controladores dependen de DAOs.
  + Controladores usan clases del modelo.

**Comportamiento del Sistema (flujo básico)**

1. El usuario inicia el sistema desde Main.
2. ControladorVista se encarga de recibir datos desde la vista.
3. Se registra un cliente a través del DAO ClienteDAO.
4. Se asigna un plan desde AsignadorPlanes.
5. Se genera una Factura, que se guarda en FacturaDAO.
6. Toda la lógica de combinación entre clientes y planes está en ControladorPlanesEstudiantes y ControlVista.

**Notas importantes**

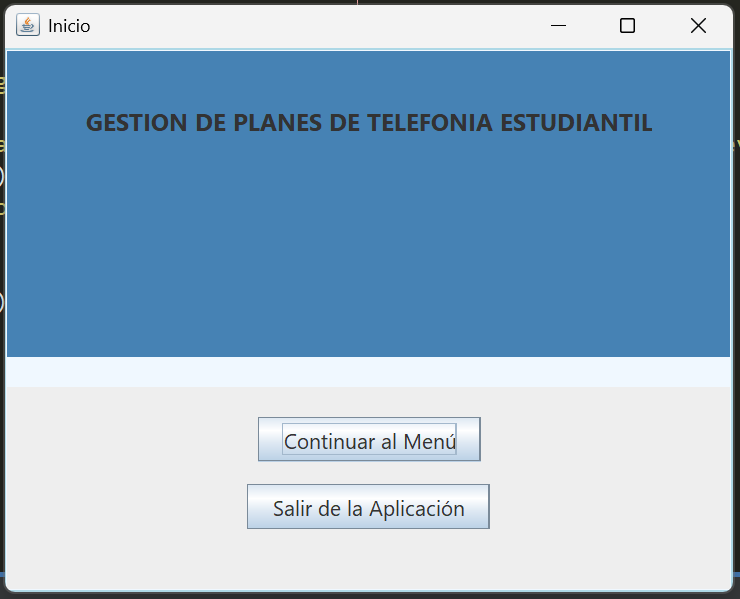
* El sistema separa claramente el modelo, la vista y el controlador.
* Usa herencia para representar distintos tipos de planes.
* Aplica el principio de **abierto/cerrado**, ya que puedes agregar más tipos de planes heredando de PlanPostPago.
* Factura actúa como un nodo de unión entre cliente y plan.

**Código Relevante**

|  |  |
| --- | --- |
| **Metodo de inserción de Estudiante** | |
| public void nuevoEstudiante(Cliente c, List<PlanPostPago> planes) {  Cliente est;  double pMen = 0;  if (!cdao.Buscar(c.getCedula())) {  est = c;  est.setPlanesActivos(planes.size());  est.setPlan(planes);  for (int i = 0; i < planes.size(); i++) {  pMen += planes.get(i).getpagoMensual();  }  est.setPagoMensual(pMen);  for (PlanPostPago plan : planes) {  Factura f = new Factura();  pdao.insertar(plan, est.getCedula());  f.setCedula(est.getCedula());  f.setPlan(plan.getNombrePlan());  f.setCategoriaPlan(plan.getCategoriaPlan());  f.setSubtotal(plan.getpagoMensual());  f.calcularIva();  f.calculartotal();  f.setNumFactura(fdao.numeroFactura());  fdao.insertar(f);  }  cdao.insertar(est);  vmu.informacion("Estudiante Agregado Correctamente");  } else {  vmu.advertencias("Ya existe un Estudiante registrado con esa cedula");  }  } |

|  |  |
| --- | --- |
| **Metodo de Nuevo Plan** | |
| public void nuevoPlan(PlanPostPago ppp, String cedula) {  Cliente est;  double mensual = 0;  Factura f = new Factura();  List<PlanPostPago> planuevo = new ArrayList<>();  est = cdao.estudiante(cedula);  if (est == null) {  vmu.error("Esta persona no se encuentra registrada....");  } else if (est.getPlanesActivos() == 2) {  vmu.error("Esta persona ya tiene la cantidad maxima de planes, elimine uno para agregar uno nuevo");  } else if (repetido(cedula, ppp.getNombrePlan(), ppp.getCategoriaPlan())) {  vmu.error("Esta persona ya tiene ese metodo, escoga uno distinto");  } else {  est.setPlan(pdao.listarPlanes(cedula));  est.setPlanesActivos(est.getPlanesActivos() + 1);  planuevo = est.getPlan();  planuevo.add(ppp);  est.setPlan(planuevo);  for (PlanPostPago plan : est.getPlan()) {  mensual += plan.getpagoMensual();  }  est.setPagoMensual(mensual);  pdao.insertar(ppp, cedula);  f.setCedula(est.getCedula());  f.setPlan(ppp.getNombrePlan());  f.setCategoriaPlan(ppp.getCategoriaPlan());  f.setSubtotal(ppp.getpagoMensual());  f.calcularIva();  f.calculartotal();  f.setNumFactura(fdao.numeroFactura());  fdao.insertar(f);  cdao.actualizar(est);  vmu.informacion("Plan Asignado con exito");  }  } |

**Evidencias De Funcionamiento:**

****

Interfaz de usuario gráfica, Aplicación

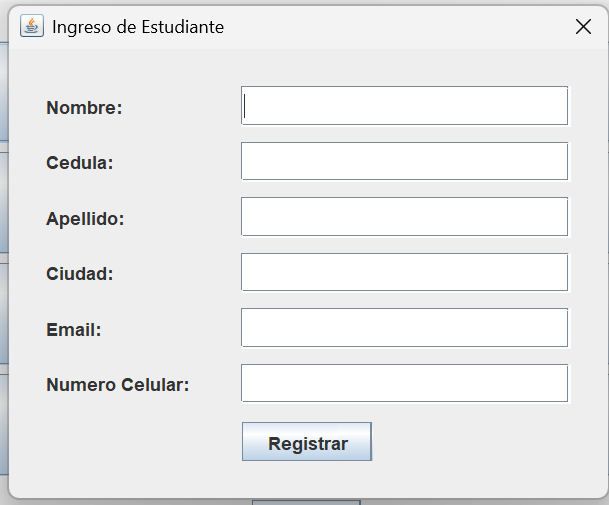
El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.



Link de GitHub: <https://github.com/ProOrientadaObjetos-P-E-AA2025/aab2-25-proyecto-bimestral-2do-bim-grupal-Andres-Yaguachi.git>