

prototype

Mov-UTPL

Siguiente

Emilio Cueva

POO
ING. Wayner Bustamante

Abraham Ayala

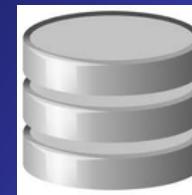
Descripcion General

El Sistema Mov-UTPL es una aplicación de gestión de telefonía móvil para estudiantes universitarios que permite:

- Gestionar clientes universitarios
- Administrar diferentes tipos de planes móviles
- Generar facturas automáticamente
- Mantener un máximo de 2 planes por cliente
- Aplicar principios de POO (herencia y polimorfismo)

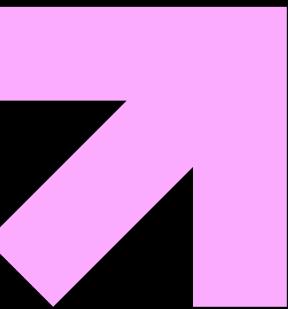


NETBEANS 26



DATABASE SQLITE

REQUERIMIENTOS FUNCIONALES IDENTIFICADOS



- CRUD de clientes con validaciones
- CRUD de planes móviles (4 tipos diferentes)
- Generación automática de facturas
- Restricción de máximo 2 planes por cliente
- Cálculo polimórfico de costos según tipo de plan
- Persistencia en base de datos SQLite
- Reportes del sistema

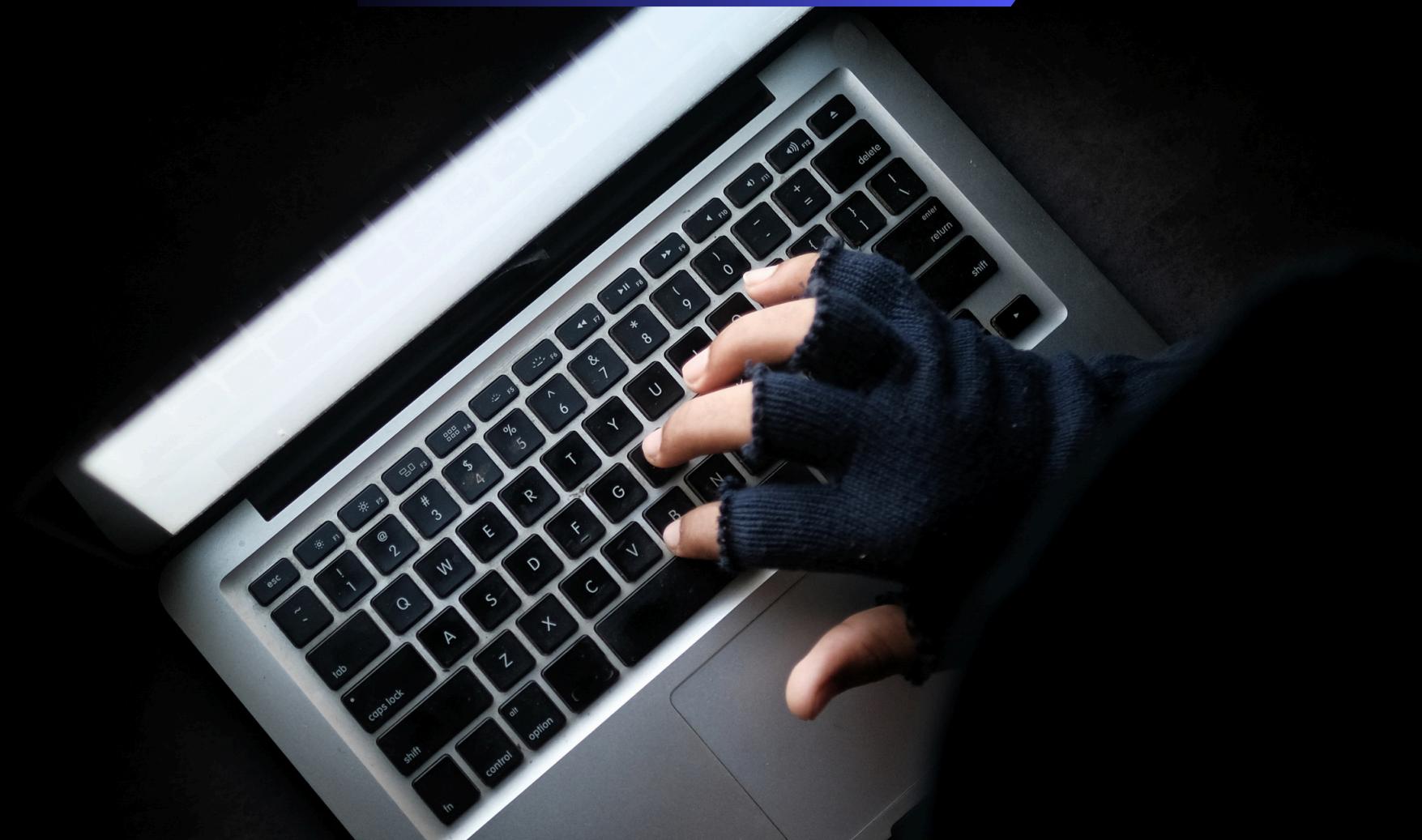
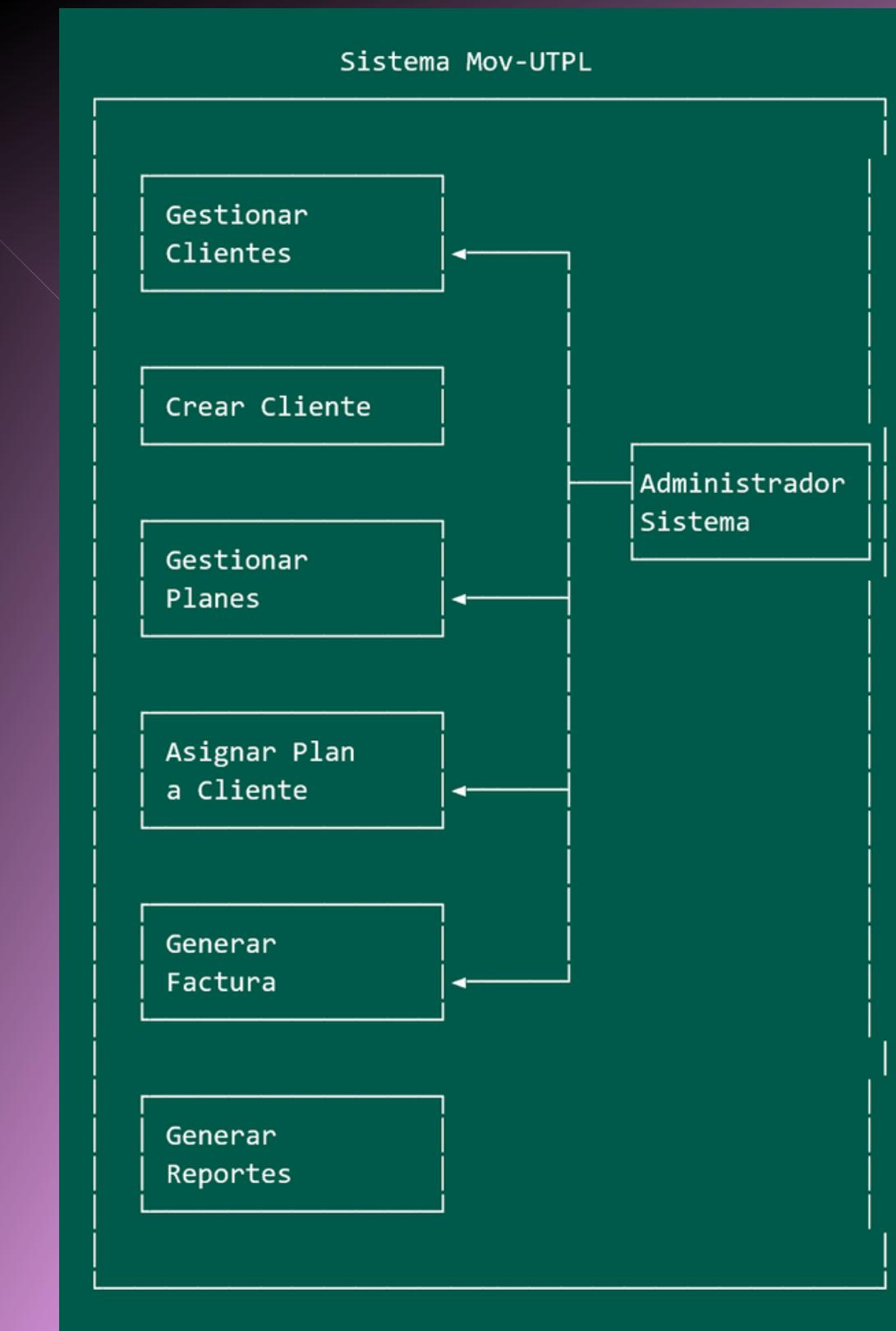
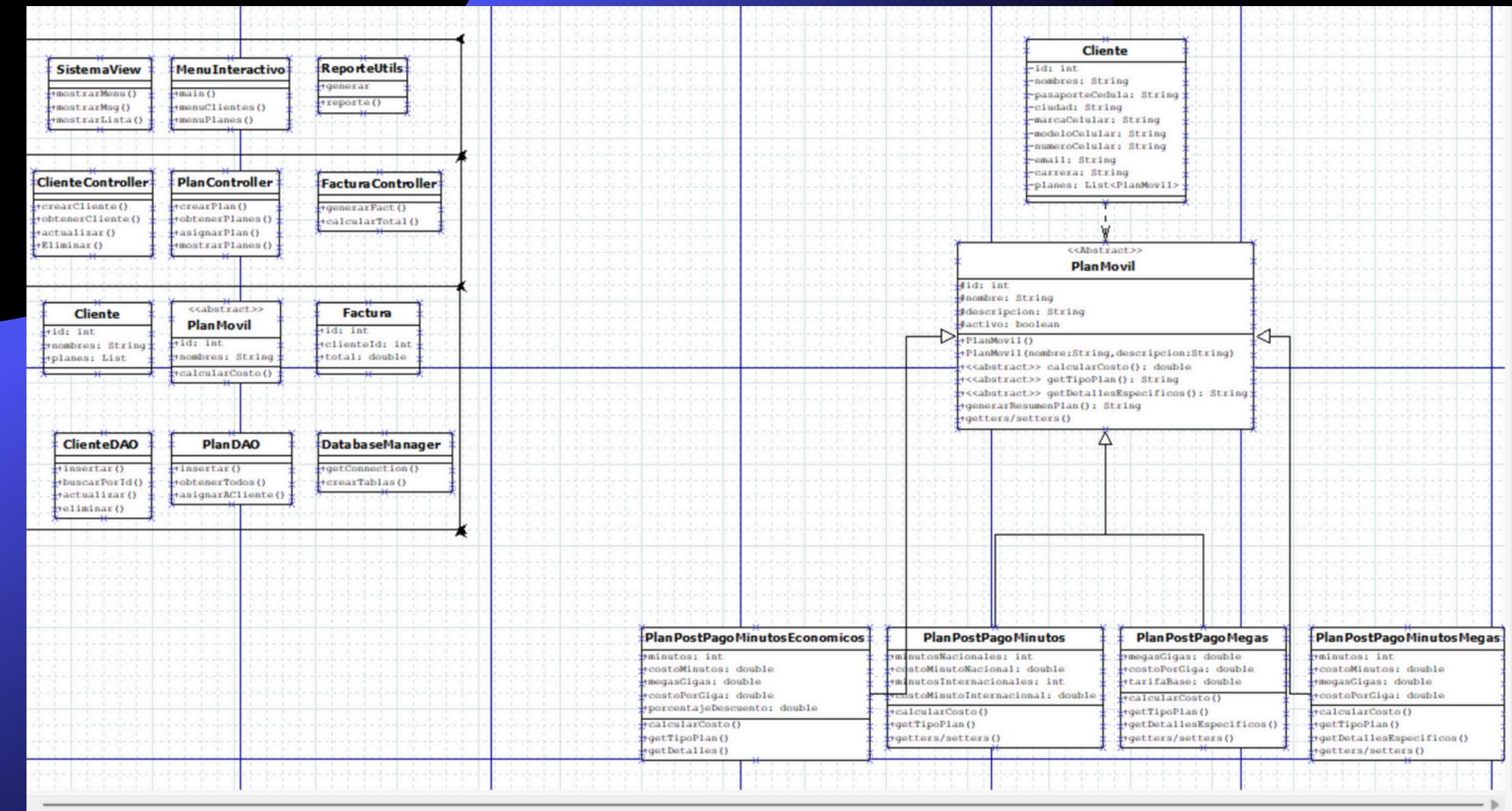


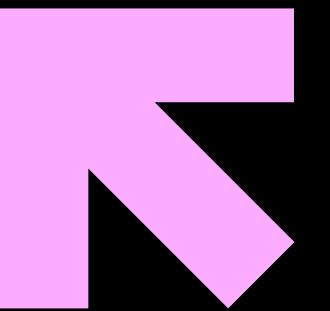
DIAGRAMA DE CASOS DE USO



MODELADO UML-DIA



CONCEPTOS DE -POO- APLICADOS



1. HERENCIA

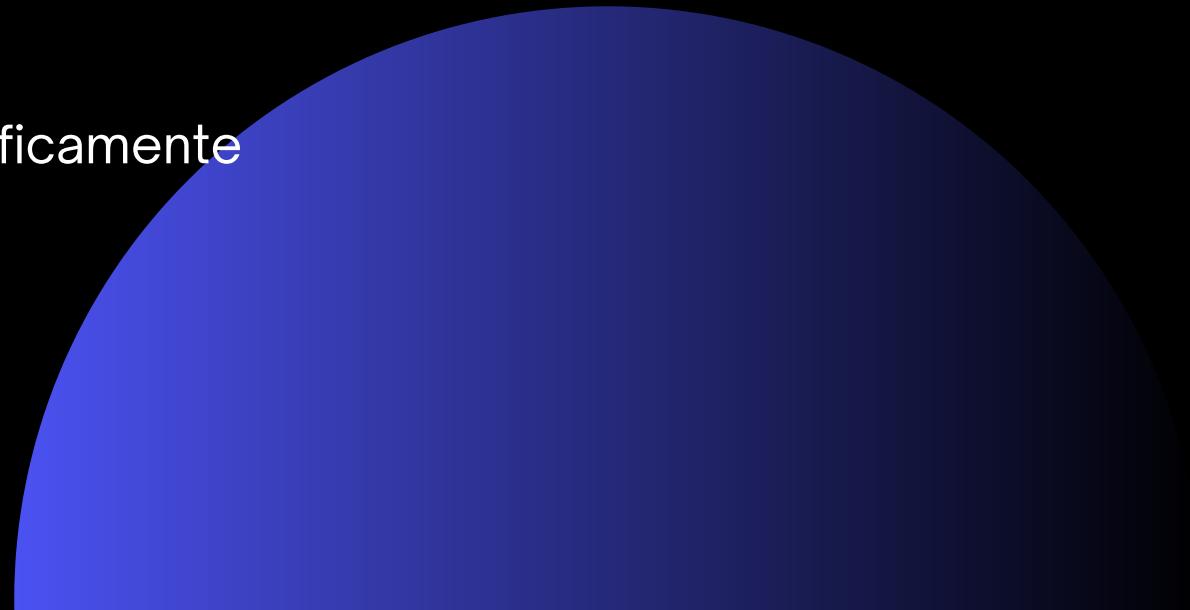
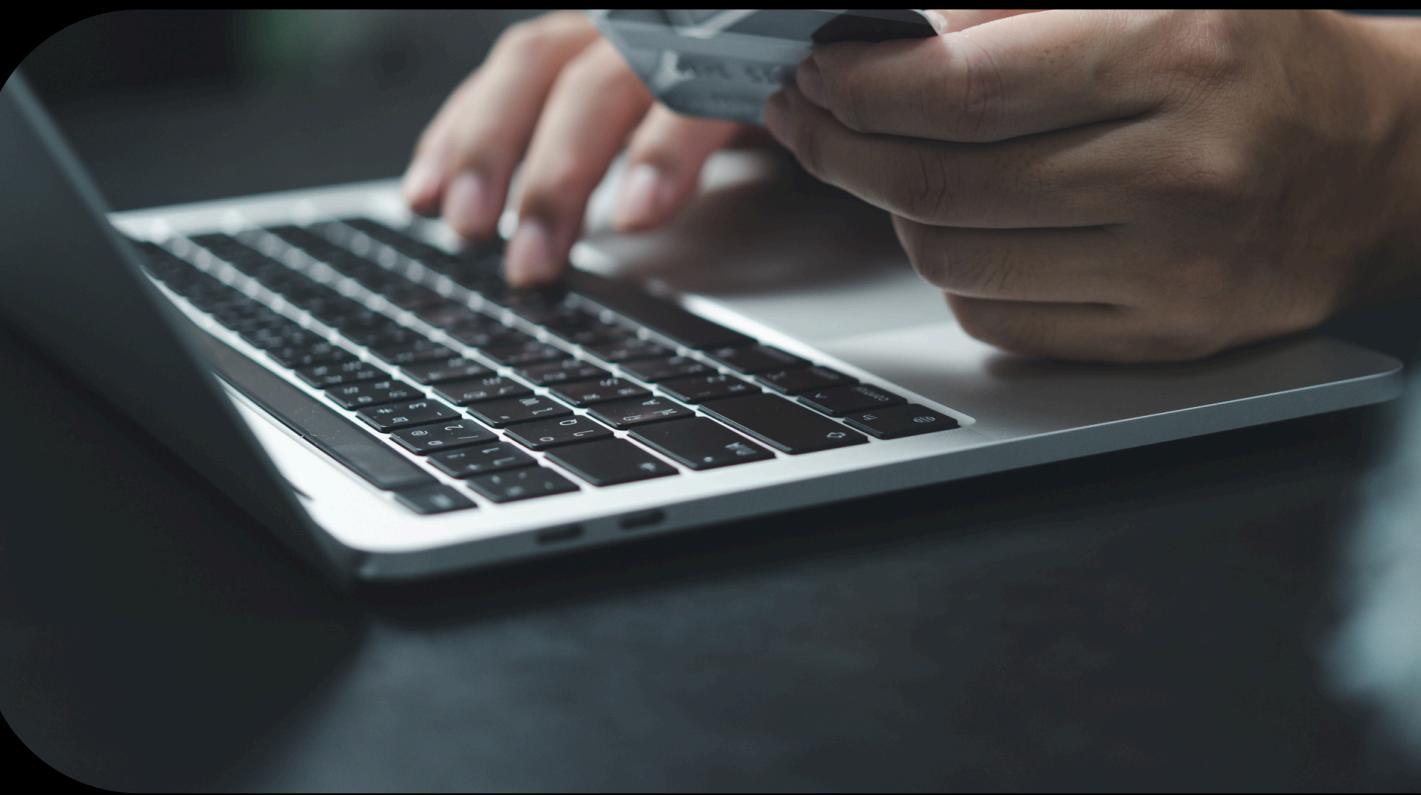
- Clase abstracta “PlanMovil” como clase padre
- 4 clases hijas: PlanPostPagoMinutosEconomico, PlanPostPagoMinutos, PlanPostPagoMegas, PlanPostPagoMinutosMegas
- Cada clase hija hereda atributos y métodos de la clase padre

2. POLIMORFISMO

- Métodos abstractos: calcularCosto(), getTipoPlan(), getDetallesEspecificos()
- Cada clase hija implementa estos métodos de forma diferente
- El mismo código puede trabajar con diferentes tipos de planes sin saber cuál es específicamente

3. ENCAPSULAMIENTO

- Atributos privados con getters y setters
- Métodos públicos para interactuar con los objetos



ARQUITECTURA MVC



Arquitectura MVC

- Modelo (Model)
 - Cliente.java - Entidad cliente con validación de máximo 2 planes
 - PlanMovil.java - Clase abstracta base para planes
 - Factura.java - Entidad para facturación
 - Clases DAO para acceso a datos
- Vista (View)
 - SistemaView.java - Interfaz básica
 - ReporteUtils-mostrarReportes
 - TestSistema-Pruebas
 - Métodos para mostrar información y menús
- Controlador (Controller)
 - ClienteController.java - Lógica de negocio para clientes
 - PlanController.java - Lógica de negocio para planes
 - FacturaController.java - Lógica de facturación
- Base de Datos SQLite
 - Tablas Creadas:
 - clientes - Datos básicos + email y carrera (atributos adicionales)
 - planes - Diferentes tipos de planes con campos específicos
 - cliente_planes - Relación muchos a muchos
 - facturas - Facturación del sistema

CODIFICACION



java with ant

Clase Abstracta “PlanMovil”

```
public abstract class PlanMovil {

    protected int id;
    protected String nombre;
    protected String descripcion;
    protected boolean activo;

    // Constructor vacío
    public PlanMovil() {
        this.activo = true;
    }

    // Constructor con parámetros
    public PlanMovil(String nombre, String descripcion) {
        this();
        this.nombre = nombre;
        this.descripcion = descripcion;
    }

    // Métodos ABSTRACTOS - Las clases hijas DEBEN implementarlos (POLIMORFISMO)
    public abstract double calcularCosto();

    public abstract String getTipoPlan();

    public abstract String getDetallesEspecificos();

    // Método concreto que pueden usar todas las clases hijas
    public String generarResumenPlan() {
        String resumen = "==== RESUMEN DEL PLAN ====\n";
        resumen += "ID: " + id + "\n";
        resumen += "Nombre: " + nombre + "\n";
        resumen += "Tipo: " + getTipoPlan() + "\n";
        resumen += "Descripción: " + descripcion + "\n";
        resumen += "Detalles Específicos: " + getDetallesEspecificos() + "\n";
        resumen += "Costo mensual: $" + String.format("%.2f", calcularCosto()) + "\n";
        resumen += "Estado: " + (activo ? "Activo" : "Inactivo") + "\n";
        return resumen;
    }
}
```

Clase del Cliente

```
public class Cliente {

    private int id;
    private String nombres;
    private String pasaporteCedula;
    private String ciudad;
    private String marcaCelular;
    private String modeloCelular;
    private String numeroCelular;
    private String email; // Atributo adicional 1
    private String carrera; // Atributo adicional 2
    private List<PlanMovil> planes;

    // Constructor vacío
    public Cliente() {
        this.planes = new ArrayList<>();
    }

    // Constructor con parámetros
    public Cliente(String nombres, String pasaporteCedula, String ciudad,
                  String marcaCelular, String modeloCelular, String numeroCelular,
                  String email, String carrera) {
        this();
        this.nombres = nombres;
        this.pasaporteCedula = pasaporteCedula;
        this.ciudad = ciudad;
        this.marcaCelular = marcaCelular;
        this.modeloCelular = modeloCelular;
        this.numeroCelular = numeroCelular;
        this.email = email;
        this.carrera = carrera;
    }

    // Método para agregar un plan (máximo 2)
    public boolean agregarPlan(PlanMovil plan) {
        if (planes.size() < 2) {
            planes.add(plan);
        } else {
            System.out.println("Error: Un cliente no puede tener más de 2 planes");
            return false;
        }
    }

    // Método para remover un plan
    public boolean removerPlan(int planId) {
        for (int i = 0; i < planes.size(); i++) {
            if (planes.get(i).getId() == planId) {
                planes.remove(i);
                return true;
            }
        }
        return false;
    }

    // Método que usa POLIMORFISMO para calcular el pago total
    public double calcularPagoMensual() {
        double total = 0;
        for (PlanMovil plan : planes) {
            total += plan.calcularCosto(); // Aquí se aplica polimorfismo
        }
        return total;
    }

    // Método para mostrar información del cliente
    public String mostrarInformacion() {
        String info = "==== INFORMACIÓN DEL CLIENTE ====\n";
        info += "ID: " + id + "\n";
        info += "Nombres: " + nombres + "\n";
        info += "Cédula/Pasaporte: " + pasaporteCedula + "\n";
        info += "Ciudad: " + ciudad + "\n";
        info += "Celular: " + marcaCelular + " " + modeloCelular + "\n";
        info += "Número: " + numeroCelular + "\n";
        info += "Email: " + email + "\n";
        info += "Carrera: " + carrera + "\n";
        info += "Planes contratados: " + planes.size() + "/2\n";
        info += "Pago mensual total: $" + String.format("%.2f", calcularPagoMensual()) + "\n";
        return info;
    }
}
```

RESULTADOS

DB Browser for SQLite - C:\Users\LENOVO\Downloads\ProyectoPOO\ProyectoPOO\movutpl.db

Table: clientes

	id	nombres	pasaporte_cedula	ciudad	marca_celular	modelo_celular	numero_celular	email	carrera
1	1	Juan Pérez	1234567890	Loja	Samsung	Galaxy S21	0987654321	juan.perez@utpl.edu.ec	Telecommunicaciones
2	2	Maria García	0987654321	Quito	iPhone	14 Pro	0912345678	maria.garcia@utpl.edu.ec	Sistemas

Editing row=1, column=0
Type: Text / Numeric; Size: 1 character(s)
Apply

Identity: Select an identity to connect
DBHub.io Local Current Database

Name Last modified Size

SQL Log Plot DB Schema Remote

DB Browser for SQLite - C:\Users\LENOVO\Downloads\ProyectoPOO\ProyectoPOO\movutpl.db

Table: planes

	id	tipo_plan	nombre	descripcion	activo	minutos	costo_minutos	minutos_nacionales	costo_m
1	1	PlanPostPagoMinutosEconomico	Plan Estudiante	Plan especial para estudiantes	1	500	0.05	NULL	
2	2	PlanPostPagoMinutos	Plan Llamadas	Para quienes solo llaman	1	NULL	NULL	1000	
3	3	PlanPostPagoMegas	Plan Internet	Para navegación	1	NULL	NULL	NULL	
4	4	PlanPostPagoMinutosNegas	Plan Completo	Lo mejor de ambos	1	800	0.04	NULL	
5	5	PlanPostPagoMinutosEconomico	Plan Estudiante	Plan especial para estudiantes	1	500	0.05	NULL	
6	6	PlanPostPagoMinutos	Plan Llamadas	Para quienes solo llaman	1	NULL	NULL	1000	
7	7	PlanPostPagoMegas	Plan Internet	Para navegación	1	NULL	NULL	NULL	
8	8	PlanPostPagoMinutosNegas	Plan Completo	Lo mejor de ambos	1	800	0.04	NULL	
9	9	PlanPostPagoMinutosEconomico	Plan Estudiante Básico	Plan económico para estudiantes	1	300	0.05	NULL	
10	10	PlanPostPagoMinutos	Plan Llamadas Ilimitadas	Para quienes hablan mucho	1	NULL	NULL	2000	
11	11	PlanPostPagoMegas	Plan Internet Pro	Navegación sin límites	1	NULL	NULL	NULL	
12	12	PlanPostPagoMinutosNegas	Plan Todo Incluido	La mejor opción completa	1	1000	0.03	NULL	

Editing row=1, column=0
Type: Text / Numeric; Size: 1 character(s)
Apply

Identity: Select an identity to connect
DBHub.io Local Current Database

Name Last modified Size

SQL Log Plot DB Schema Remote

GRACIAS

Emilio Cueva

POO
ING. Wayner Bustamante

Abraham Ayala