Forma

El contenido generado por IA puede ser incorrecto.

DARWIN SANTIAGO CASTILLO PAREDES

Abril / Agosto 2025

FECHA:

**Universidad Técnica Particular de Loja**

**Facultada de Ingenierías y Arquitectura**

ASIGNATURA:

DOCENTE:

AUTOR:

Programación Orientada a Objetos

WAYNER BUSTAMANTE

Abril / Agosto 2025

AÑO:

ASIGNATURA:

DARWIN SANTIAGO CASTILLO PAREDES

PROGRAMACIÓN ORIENTADA A OBJETOS

Wayner Bustamante

DOCENTE:

AUTORES:

Tabla de contenido

[Análisis 3](#_Toc204355690)

[Arquitectura 4](#_Toc204355691)

[Validaciones en código 4](#_Toc204355692)

[UML 4](#_Toc204355693)

[Codigo 5](#_Toc204355694)

[URL GIT 5](#_Toc204355695)

[Resultados 6](#_Toc204355696)

## Análisis

Tabla de procesamiento, de cómo se planteó:

| **Módulo** | **Objetivo principal** | **Funciones clave** |
| --- | --- | --- |
| Clientes | Gestión del registro y datos de clientes | Crear, leer, actualizar, eliminar (CRUD) |
| Planes | Asignación y gestión de planes móviles | CRUD + Validación de máximo 2 planes por cliente |
| Facturación | Generación de facturas por cliente | Calcular valor a pagar según plan, emitir y consultar facturas |
| Persistencia | Almacenamiento confiable de información | Base de datos SQLite integrada con Java |
| Arquitectura | Diseño mantenible y escalable | Patrón Modelo-Vista-Controlador (MVC) |

Requerimiento a cumplir:

1. **Registro de clientes** con nombre, identificación, etc.
2. A**signación de planes**:

* PlanPostPagoMinutos
* PlanPostPagoMegas
* PlanPostPagoMinutosMegas
* PlanPostPagoMinutosMegasEconomic

1. **Restricción de hasta 2 planes por cliente**, gestionado desde el DAO.
2. **Cálculo automático de montos a pagar** según tipo de plan y consumo.
3. **Emisión de facturas**: debe registrar cliente, plan, fecha, monto, y aplicar descuentos según tipo de plan.
4. I**nterfaz amigable para CRUD** y generación de facturas

## Arquitectura

Modelo-Vista-Controlador

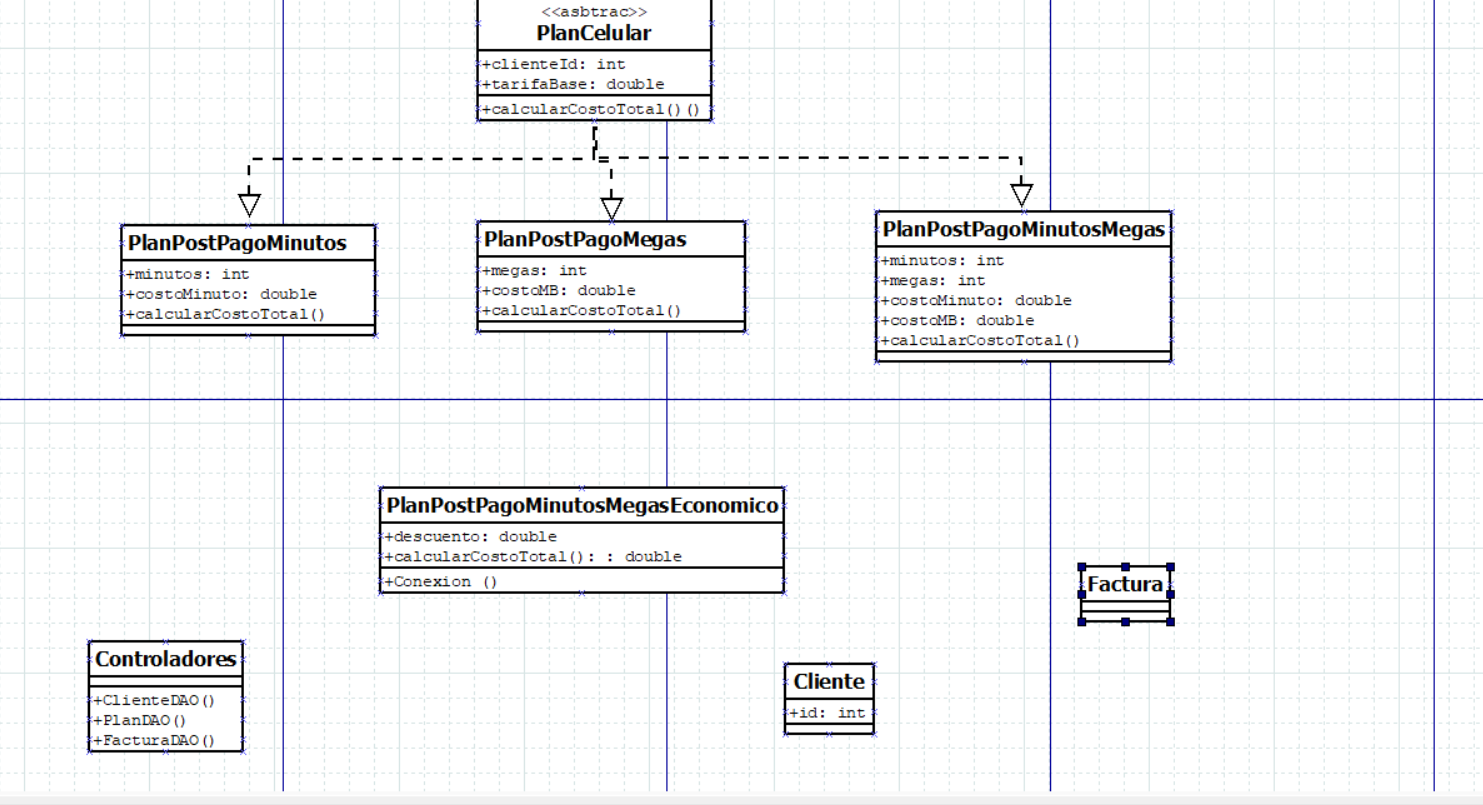
* Separar responsabilidades (datos, lógica, interfaz).
* Facilitar el mantenimiento y escalabilidad.
* Integrar patrones orientados a objetos como herencia (para los tipos de planes) y polimorfismo (para el cálculo de factura).

Además, se elige SQLite como motor de base de datos por ser liviano, embebido y compatible con Java mediante JDBC, facilitando su portabilidad en entornos de escritorio.

## Validaciones en código

* **Validación de límite de planes**: Se implementa en PlanDAO, mediante un método que consulta la cantidad de planes asociados a un cliente antes de insertar uno nuevo.
* **Polimorfismo** aplicado en método calcularCostoTotal() que es sobreescrito por cada tipo de plan.

## UML



* PlanCelular usa herencia para permitir el uso de polimorfismo. Así, FacturaDAO o cualquier parte del sistema puede tratar a todos los planes de forma uniforme, llamando calcularCostoTotal() sin importar el tipo de plan.
* Cliente está asociado con hasta dos objetos PlanCelular. Esta relación se valida en PlanDAO mediante un método como clienteTieneMenosDeDosPlanes().
* Factura usa la información de un plan específico para generar el costo total al cliente.

## Codigo

public boolean clienteTieneMenosDeDosPlanes(int clienteId) {

String sql = "SELECT COUNT(\*) FROM planes WHERE cliente\_id = ?";

try (Connection conn = Conexion.conectar(); PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setInt(1, clienteId);

ResultSet rs = pstmt.executeQuery();

if (rs.next()) {

return rs.getInt(1) < 2;

}

} catch (SQLException e) {

System.out.println("Error: " + e.getMessage());

}

return false;

}

public boolean insertarPlan(PlanCelular plan) {

if (!clienteTieneMenosDeDosPlanes(plan.getClienteId())) {

System.out.println(" El cliente ya tiene 2 planes asignados.");

return false;

}

}

## URL GIT

<https://github.com/ProOrientadaObjetos-P-D-AA2025/aab2-25-proyecto-bimestral-2do-bim-grupal-SantiagoC00/tree/main/2_SOLUCION>

## Resultados