# PROYECTO FINAL

## Diseño y Desarrollo de aplicaciones

### Descripción breve

Proyecto final de la materia de Diseño y desarrollo de aplicaciones de la universidad ort
2025

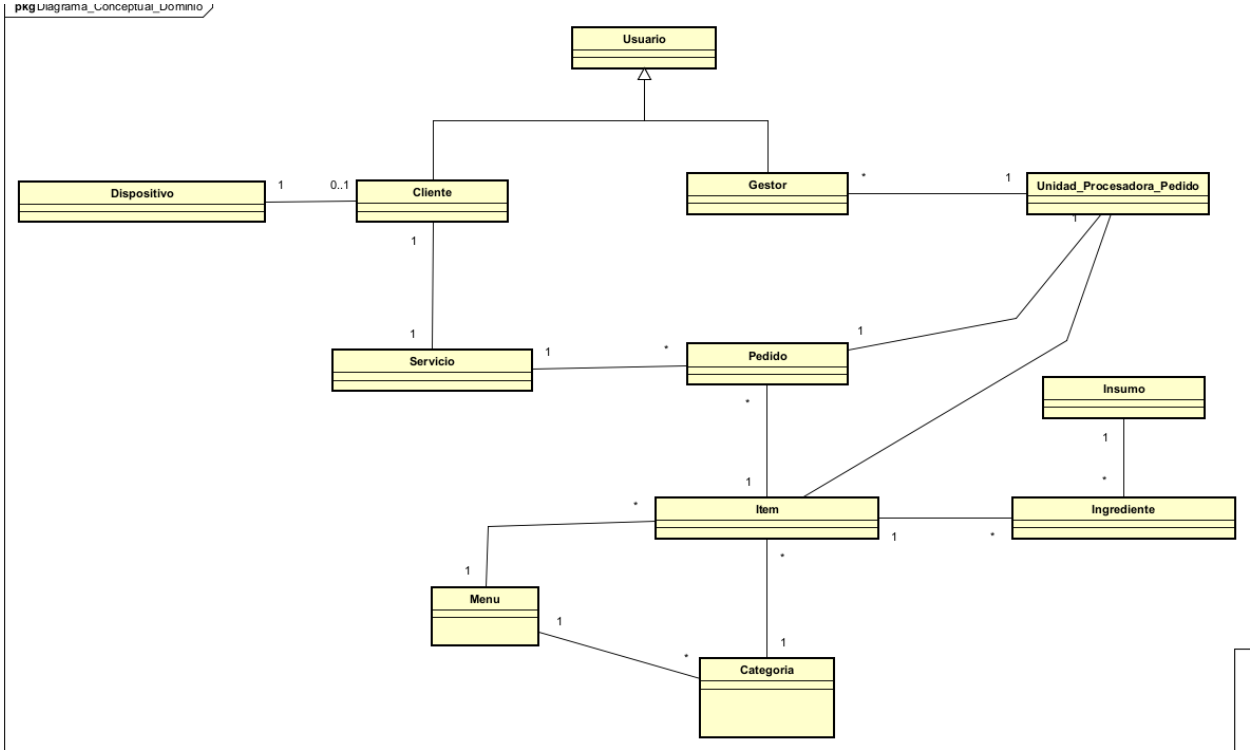
Luca Podesta y Santiago Caceres

# Contenido

# Descripcion

Esta documentación contara con los diferentes diagramas de la solución del proyecto y algunas justificaciones y puntos a tener en cuenta que nos gustaría expandir y hacer énfasis.

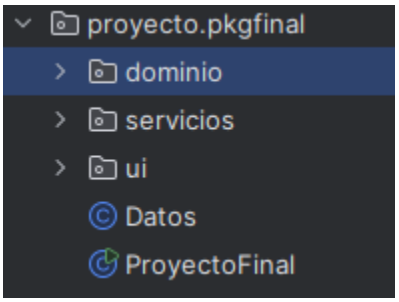
# Diagrama Conceptual de Dominio




# Diagrama de Diseño

# Division Lógica

EL proyecto se dividio en tres paquetes:




Dominio

Toda la estructura del programa.

Servicios

Servicios asociados , como Observador y fachada

Ui

Patron MVC aplicado

# Experto

Las clase se encargan cada una de realizar su tarea en especifico, así sea solucionando el requerimiento o llamando a su responsable.

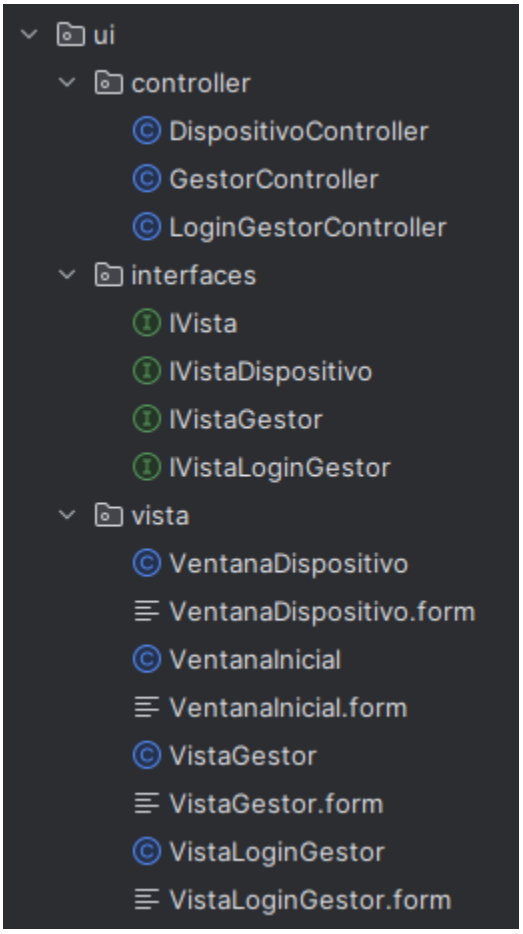
# Fachada y Arquitectura

Patron Singleton Aplicado:

```
//Singleton
private static Fachada instancia;
private Fachada(){
    sAcceso = new SistemaAccesso();
}

public Fachada getInstancia(){
    if(instancia == null) {
        instancia = new Fachada();
    }
    return this.instancia;
}
```

# MVC


ui
controller
DispositivoController
GestorController
LoginGestorController
interfaces
IVista
IVistaDispositivo
IVistaGestor
IVistaLoginGestor
vista
VentanaDispositivo
VentanaDispositivo.form
VentanaInicial
VentanaInicial.form
VistaGestor
VistaGestor.form
VistaLoginGestor
VistaLoginGestor.form

# Observador

```java
public class Observable {
    private ArrayList<Observador> observadores = new ArrayList<>();

    public void agregar(Observador observador) {
        observadores.add(observador);
    }

    public void quitar(Observador observador) {
        observadores.remove(observador);
    }

    public void avisar(Object evento) {
        for (Observador o : observadores) {
            o.actualizar(this, evento);
        }
    }
}
```

```java
public interface Observador {
    void actualizar(Observable origen, Object evento);
}
```

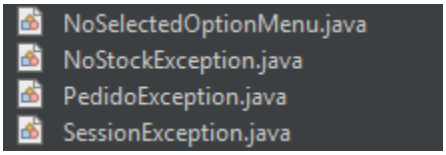
Alguna de sus implementaciones:

```java
@Override
public void actualizar(Observable origen, Object evento) {
    if(evento == Fachada.eventos_pedidos.pedidoAgregado || evento == Fachada.eventos_pedidos.pedidoEliminado ||  evento == Fachada.eventos_pedidos.pedidosConfirmados
        if(dispositivo.esLogueado()){
            if(evento == Fachada.eventos_pedidos.pedidosConfirmados){
                this.verificarStockPedidos();
                actualizarVista(false);
            }else{
                actualizarVista(true);
            }
        }
    }

    if(evento == Fachada.eventos_acceso.login){

        if(dispositivo.esLogueado()){
            vista.mostrarSesion(dispositivo.getClienteLogueado().getNombreCompleto());
            vista.mostrarOk("Logueado con exito! Bienvenido, "+ dispositivo.getClienteLogueado().getNombreCompleto());
        }
    }
}
```

## Manejo de Excepciones




## Polimorfismo

```java
public abstract class TipoCliente {

    boolean tieneBeneficios;

    public TipoCliente(boolean tieneBeneficios){
        this.tieneBeneficios = tieneBeneficios;
    }

    public abstract double calcularDescuento(Servicio servicio);
}
```

```java
public class ClientePreferencial extends TipoCliente{

    public ClientePreferencial() {
        super(true);
    }

    @Override
    public double calcularDescuento(Servicio servicio) {
        double descuento = 0.0;
        for(Pedido pedido : servicio.getPedidos()){
            if (pedido.getItem().esItem("Agua Mineral")){
                servicio.setBeneficioAsignado("Descuento de Agua!");
                descuento += pedido.getItem().getPrecio();
            }
        }
        if (servicio.getMontoTotal() > 2000) {
            double total = (servicio.getMontoTotal() - descuento);
            descuento += total * 0.05;
            servicio.setBeneficioAsignado((servicio.getBeneficioAsignado() != null ? servicio.getBeneficioAsignado() + " y " : " ") + "5% de descuento");
        }
        return descuento;
    }


}
```

## Justificaciones y puntos a tener en cuenta

Hicimos una relación bidireccional entre Servicio y pedido para que pedido pueda conocer su  cliente y el servicio donde trabaja.