

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Computación y Electricidad

**Diseño de Software**

**PROYECTO FINAL**

Segundo Avance

**Integrantes:**

* Ronald Neythan León Vásquez
* Carla Sánchez Cevallos
* Miguel Ángel Patiño San-Yeng
* Luis Rodríguez Toaquiza

Tabla de contenido

[Descripción del sistema 3](#_Toc535176868)

[Patrones de diseño utilizados 4](#_Toc535176869)

[Strategy 4](#_Toc535176870)

[Facade 4](#_Toc535176871)

[Factory Method 4](#_Toc535176872)

[Singleton 4](#_Toc535176873)

[Decorator 4](#_Toc535176874)

# Descripción del sistema

El sistema que se diseñará e implementará para este proyecto se trata de una plataforma para escritorio que busca brindar un servicio de compra-venta de productos y servicios, enfocada en estudiantes emprendedores. Para lograr una aplicación robusta y escalable se aplicarán conceptos fundamentales del diseño de software como lo son los patrones de diseño, además se seguirá la arquitectura MVC para la organización del código y se buscará evitar la presencia de malos olores en el código.

# Patrones de diseño utilizados

Los patrones elegidos para la implementación de nuestra aplicación se detallan a continuación:

## Strategy

El patrón de diseño Strategy se implementará para manejar las clases correspondientes al pago, es decir las clases dentro de models.pago (la clase abstracta MetodoPago y sus clases hijas, PagoEfectivo y PagoVirtual ). Se eligió este patrón porque de está manera se garantiza que se pueda en un futuro agregar nuevos métodos de pago y porque está permite que se realice una acción específica, pero de la manera que elija el usuario al momento de ejecución.

## Facade

Se implementa el patrón Facade al momento de mostrar los menús de cada rol, se tiene la ventana de Login, dónde el usuario ingresa su usuario y contraseña para posteriormente mostrar una ventana que muestre las opciones correspondientes al rol con el que se encuentra registrado.

## Factory Method

Se utilizó Factory Method para la creación de usuario, en el paquete models.entities se tiene la clases correspondientes a este patrón. Como a cada usuario se le muestra una ventana distinta según su rol, se utilizó este patrón para que según el rol que se defina se cree el usuario correspondiente (administrador, comprador, vendedor), esto simplifica las tareas en las que se necesita especificar el rol para generar cierta salida, como por ejemplo al momento de generar el menú para el usuario.

## Singleton

El patrón Singleton se utiliza para la construcción de las clases que se encargarán de conectar la aplicación desarrollada en Java con la base de datos en MySQL. Se utiliza este patrón puesto que la conexión solo debe ser instanciada una vez, ya que sólo existe conexión a una sola base de datos, se encuentra en el paquete services::DBConnection.

## Decorator

El patrón de diseño Decorator se implementó para personalizar las ventanas de acuerdo con la opción que se desea mostrar, por ejemplo, se creó una instancia de una ventana de LoginView y con esta de “decoró” la ventana que se tiene como plantilla en la clase StageDecoratorX. Se busca facilitar y agilizar la creación de las interfaces del sistema mediante el uso de este patrón.