**UNIVERSIDAD NACIONAL AGRARIA DE LA SELVA**

**FACULTAD DE INGENIERÍA EN INFORMÁTICA Y SISTEMAS**

**ESCUELA ACADÉMICA PROFESIONAL DE INGENIERÍA EN INFORMÁTICA Y SISTEMAS**

Diagrama

Descripción generada automáticamenteLogotipo

Descripción generada automáticamente

**PATRONES DE COMPORTAMIENTO**

**CONSECIONARIA “SPEED X MOTORS” (Chain of Responsibility)**

AUTORES : PAZ FLORES GREGORIO BENJAMIN

JUNCO BERROSPI, JONATHAN MANUEL

ASIGNATURA : DISEÑO DESOFTWARE

DOCENTE : SOLIS BONIFACIO, HUBEL

**TINGO MARÍA - 2023**

**PERU**

INDICE DE CONTENIDO

[**I.** **INTRODUCCION** 3](#_Toc136217182)

[**II.** **CONSECIONARIA “SPEED X MOTORS”** 4](#_Toc136217183)

[**2.1.** **Planteamiento del problema** 4](#_Toc136217184)

[**2.2.** **Requisitos Específicos** 4](#_Toc136217185)

[**2.3.** **DESCRIPCION DEL SISTEMA** 5](#_Toc136217186)

[**2.3.1.** **INICIO** 5](#_Toc136217187)

[**2.3.2.** **ADMINISTRADOR** 6](#_Toc136217188)

[**2.3.2.1.** **Loguin del administrador** 6](#_Toc136217189)

[**2.3.2.2.** **Menú del administrador** 6](#_Toc136217190)

[**2.3.2.3.** **Almacén** 7](#_Toc136217191)

[**2.3.2.4.** **Registro de clientes** 9](#_Toc136217192)

[**2.3.3.** **CLIENTE** 10](#_Toc136217193)

[**2.3.3.1.** **Menú del cliente** 10](#_Toc136217194)

[**2.3.3.2.** **Lista de vehículos mostrado al cliente** 10](#_Toc136217195)

[**2.3.3.3.** **Registro de compra del vehículo** 11](#_Toc136217196)

[**2.4.** **CONCLUCIONES** 13](#_Toc136217197)

1. **INTRODUCCION**

En el diseño de software, los patrones de comportamiento son soluciones reutilizables para problemas comunes que se encuentran al desarrollar aplicaciones. Estos patrones describen la comunicación y la interacción entre objetos y clases, y proporcionan pautas para resolver situaciones específicas de manera efectiva.

Los patrones de comportamiento se centran en la forma en que los objetos y las clases se comunican y colaboran entre sí para lograr un objetivo común. Ayudan a definir cómo los objetos interactúan y cómo se pueden modificar estas interacciones sin afectar a otras partes del sistema.

Existen varios patrones de comportamiento ampliamente reconocidos, algunos de los cuales incluyen:

1. **Chain of Responsibility:** Permite pasar solicitudes a lo largo de una cadena de manejadores. Al recibir una solicitud, cada manejador decide si la procesa o si la pasa al siguiente manejador.
2. **Conmand:** convierte una solicitud en un objeto independiente, que contiene toda la información de la solicitud. Esta transformación te permite parametrizarlos métodos con diferentes solicitudes, retrasar o poner en cola de ejecución de una solicitud y soportar operaciones que no se puede realizar.
3. **Iterator:** Permite recorrer elementos de una colección sin exponer su representación subyacente (lista, pila árbol, etc.).
4. **Mediator:** Permite reducir las dependencias caóticas entre objetos. El patrón restringe las comunicaciones directas entre los objetos, forzándolos a colaborar únicamente a través de un objeto mediador.
5. **Memento:** Permite guardar y restaurar el estado previo de un objeto sin revelar los detalles de implementación.
6. **Observer:** Permite definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier efecto que está observando.
7. **State:** Permite a un objeto alterar su comportamiento cuando su estado interno cambia. Parece como si el objeto cambiara su clase.
8. **Strategy:** Permite definir una familia de algoritmos, colocar cada uno de ellos en una clase separada y hacer sus objetos intercambiables.
9. **Template Method:** Define el esqueleto en la superclase, pero permite que las subclases sobrescriban pasos del algoritmo sin cambiar su estructura.
10. **Visitor:** permite separar algoritmos de los objetos sobre los que operan:

A continuación, presentaremos un caso practico en la cual aplicaremos algunos de los patrones de comportamiento.

1. **CONSECIONARIA “SPEED X MOTORS”**
   1. **Planteamiento del problema**

Al llevar un registro de los vehículos disponibles en el almacén de la concesionaria, se crea una complicación el manejo y revisión de los documentos y existencias.

Por ellos el cliente desea un software que permita tener un registro de sus vehículos disponibles en su inventario

* 1. **Requisitos Específicos**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PRODUCT BACKLOG | | | | |
| ID | Como un(a) | Quiero poder | De modo que | Prioridad |
| PBA1 | **Administrador** | Acceder a la lista de vehículos disponibles | Pueda visualizar una tabla de la lista de vehículos disponibles. | Alta |
| PBA2 | **Administrador** | Puedo modificar los datos de un vehículo determinado | Pueda seleccionar un vehículo y poder modificar sus datos. | Alta |
| PBA3 | **Administrador** | Pueda eliminar un vehículo del inventario | Pueda seleccionar un vehículo y poder eliminarlo. | Alta |
| PBA4 | **Administrador** | Pueda agregar los datos de un nuevo vehículo | Pueda agregar los datos de un nuevo vehículo llegado a la concesionaria. | Alta |
| PBA5 | **Administrador** | Pueda acceder a la lista de clientes. | pueda visualizar en una tabla la lista de clientes que realizar una compra en la concesionaria. | Alta |
| PBA6 | **Administrador** | Modificar, guardar, eliminar datos de un cliente | Pueda seleccionar un cliente y tener las opciones de modificar, guardar o eliminar datos de dicho cliente. | Alta |
| PBC1 | **Cliente** | Acceder a la lista de vehículos de la concesionaria | Pueda visualizar una tabla de la lista de vehículos disponibles | Alta |
| PBC2 | **Cliente** | Pueda seleccionar un vehículo y decidir si hacer la compra | Pueda seleccionar un vehículo de su agrado. | Alta |
| PBC3 | **Cliente** | Realizar la compra, y actualizar automáticamente el inventario. | Pueda realizar la compra, y se registre en la base de datos de la concesionaria, y actualice su inventario de vehículos. | Alta |
| PBS1 | **SISTEMA** | Actualizar los datos al cambio. | Ante cualquier cambio, la actualización en la base de datos sea inmediata | Alta |

* 1. **DESCRIPCION DEL SISTEMA**

**Lenguaje de programación: C#**

**IDE de desarrollo: Visual Studio Comunity 2022**

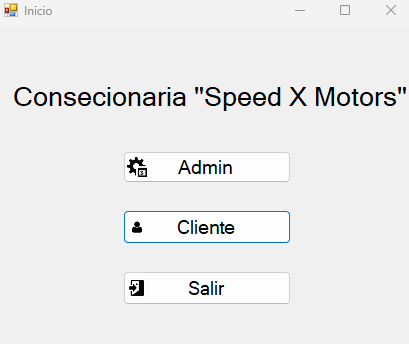
**Año: 2023**

**Patrón de comportamiento: Chain of Responsibility**

* + 1. **INICIO**

Muestra un menú interactivo para la elección de dos tipos de usuarios (Administrados-Cliente), según quiera usar el software.

Aquí vemos que manda una solicitud y de acuerdo a ello realiza una acción, aplicado el **Chain of Responsibility,** este patrón lo veremos en varias partes del sistema.



* + 1. **ADMINISTRADOR**
       1. **Loguin del administrador**

validamos el usuario y password del administrador (user-Fiis2023 respectivamente).

* En caso de no ingresar algún dato o ingresar datos erróneos tenemos las alertas.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* + - 1. **Menú del administrador**

Tenemos un menú de opciones (solicitudes) para el administrador.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Las dos solicitudes para el almacén se dirigen a una misma tabla, así que según sea la elección, se mandara una llave, la cual elegirá la acción que realizara el módulo almacén.

* + - 1. **Almacén**

Lista de existencias Motocicletas-Automóvil

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza media

Usando la opción Agregar

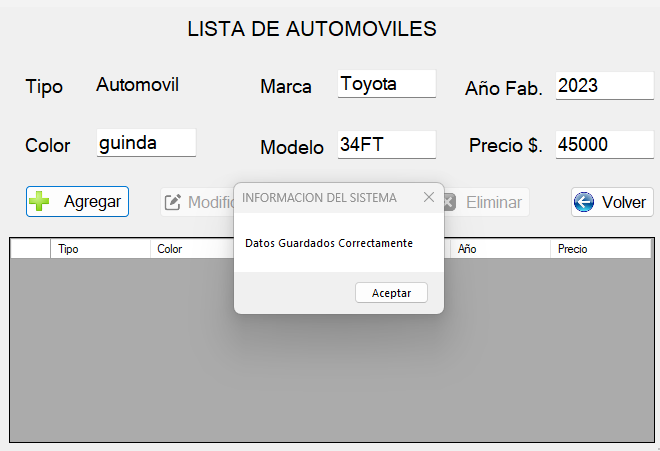


Tabla Actualizada

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Se creo un txt que funciona como BD, para guardar y cargar los datos

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Podemos eliminar algún vehículo

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Datos Antes de eliminar

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Datos Eliminados en el sistema

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

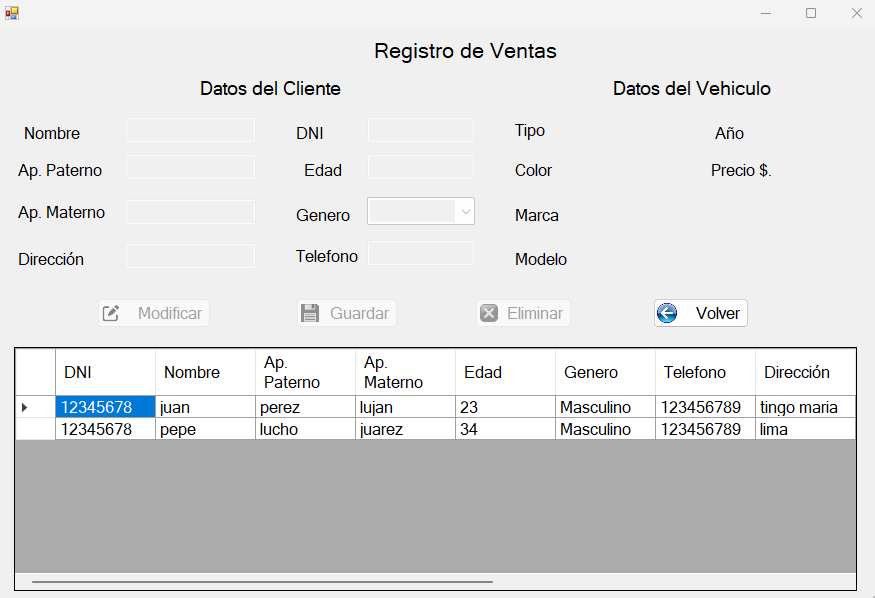
Datos actualizados en la BD

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

* + - 1. **Registro de clientes**

Al seleccionar un cliente se activa el botón modificar

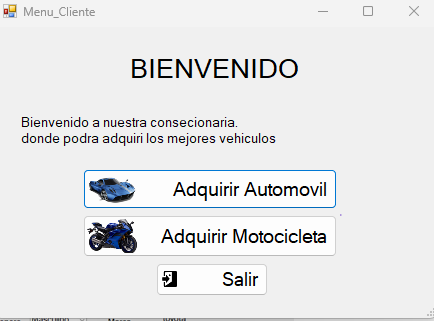
****

Y podemos realizar cualquiera de las acciones

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

* + 1. **CLIENTE**
       1. **Menú del cliente**

****

Las dos solicitudes para el almacén se dirigen a una misma tabla, así que según sea la elección, se mandara una llave, la cual elegirá la acción que realizara el módulo almacén.

* + - 1. **Lista de vehículos mostrado al cliente**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media**

Al seleccionar un vehículo se activa el botón comprar para hacer la solicitud

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* + - 1. **Registro de compra del vehículo**

Rellenamos los datos para el registro de la compra

Interfaz de usuario gráfica

Descripción generada automáticamente

Le damos a realizar pago

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Se actualizo la DB

Texto

Descripción generada automáticamente con confianza media

El vehículo fue retirado del inventario

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

Y también de la BD

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

* 1. **CONCLUCIONES**

Aplicando el patrón de comportamiento **Chain of Responsibility,** para el manejo de solicitudes en el desarrollo del software, hemos logrado satisfacer la necesidad del cliente cumpliendo todos los requisitos que registramos e el product backlog.