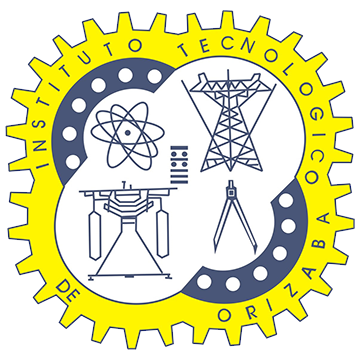
**INSTITUTO TECNOLÓGICO DE ORIZABA**



**Desarrollo de un sistema aplicando patrones de diseño**

**29/11/2023**

**Estudiante: IEVND. Abdiel Labrado Flores**

**Maestría en Sistemas Computacionales**

**Modelado Orientado a Objetos**

**Catedrático:**

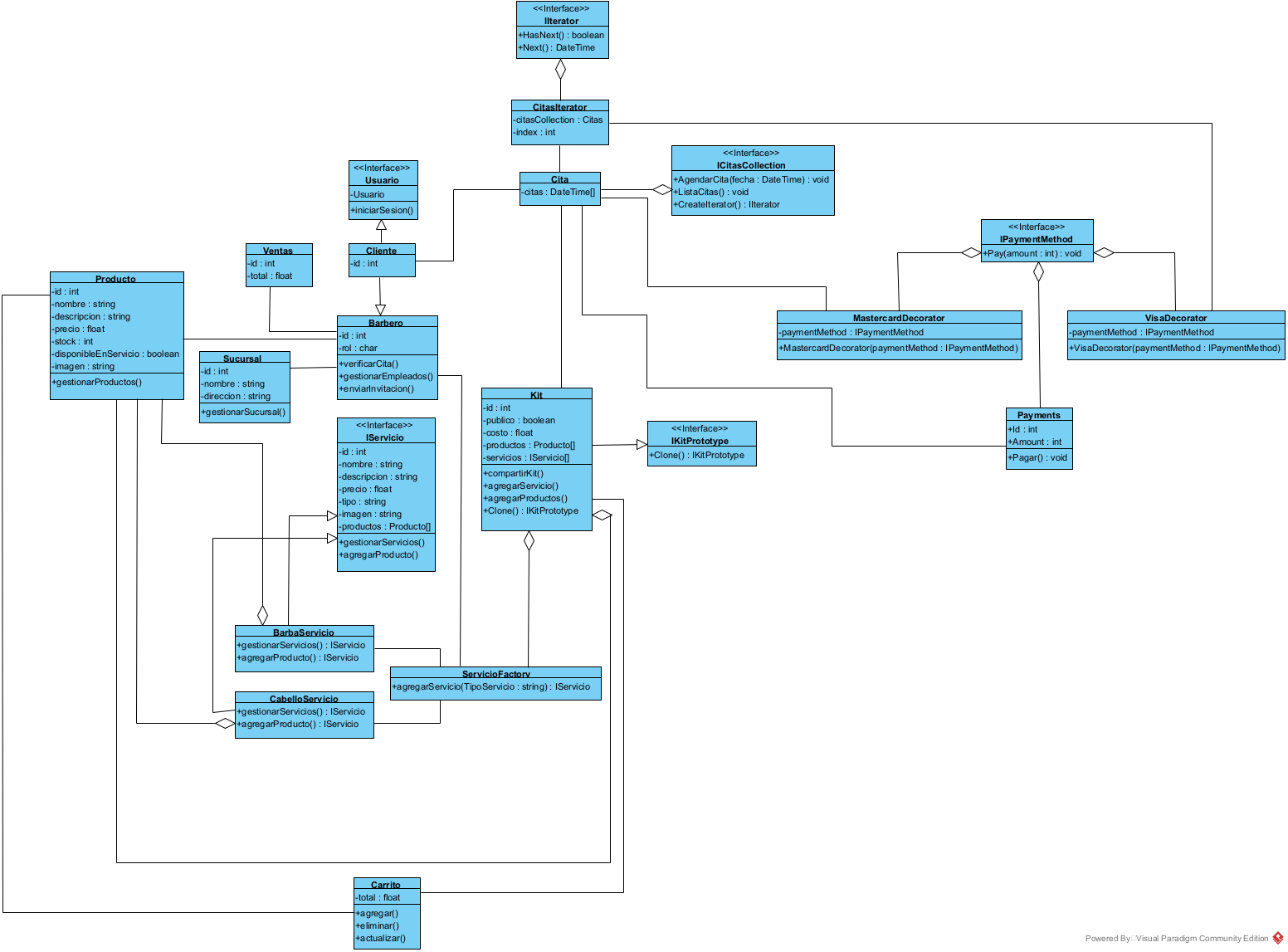
**Dr. Jose Luis Sánchez Cervantes**

**INTRODUCCIÓN**

Un patrón de diseño es una estructura establecida para lograr las buenas prácticas de la Programación Orientada a Objetos. De esta forma se tienen distintas categorías según el objetivo de cada uno de los patrones existentes. En el presente documento se describen los utilizados para el desarrollo del sistema “BarberDate”.

**DIAGRAMA DE CLASES**

Implementar un patrón de diseño implica representarlo en un diagrama de clases para entender cómo los objetos se van a comunicar o relacionar entre sí. De esta forma se tiene la certeza que cualquier persona que tome el proyecto entenderá lo codificado y lo que falta por codificar.



El anterior diagrama de clases para el sistema “BarberDate” incluye cuatro patrones de diseño que se describen en el apartado a continuación.

**PATRONES DE DISEÑO**

**FACTORY METHOD**

Es un patrón de diseño creacional que proporciona una interfaz para crear objetos en una superclase, mientras permite a las subclases alterar el tipo de objetos que se crearán. La decisión de utilizar este patrón de diseño es principalmente porque un Kit puede armarse con varios servicios, y un servicio puede adoptar distintas características según cuál sea (corte de cabello, afeitado de barba, aplicación de mascarilla facial, por mencionar algunos). Por tanto, se necesita de una estructura que facilite la reutilización del código para aprovechar una interfaz servicio para cada uno de sus subtipos.

**PROTOTYPE**

Es un patrón de diseño creacional que permite copiar objetos existentes sin que el código dependa de sus clases. Prácticamente describe toda la funcionalidad para crear Kits nuevos sobre las características de otros ya existentes.

**DECORATOR**

Es un patrón de diseño estructural que permite añadir funcionalidades a objetos colocando estos objetos dentro de objetos encapsuladores especiales que contienen estas funcionalidades. El uso de este patrón de diseño se justifica en el método de pago, principalmente al especificar los procesos con que trabaja Vissa contra los de Mastercard, por lo que se utiliza el mismo método para distintos procesos.

**ITERATOR**

Es un patrón de diseño de comportamiento que permite recorrer elementos de una colección sin exponer su representación subyacente (lista, pila, árbol, etc.). Las citas que genera un usuario, así como aquellas que tiene el barbero que cumplir para con sus clientes, necesitan ser recuperadas para realizar acciones concretas con ellas. Por esto es necesario tener noción de aquellos recorridos que se pueden realizar para llegar a ellas y optimizar los tiempos de respuesta, así como la reducción del código para lograrlo.

**REFERENCIAS**

Refactoring.Guru. (2023). Refactoring.Guru. <https://refactoring.guru/es>