**DISEÑO DE SOFTWARE**

**TAREA #3**

**Informe sobre la Implementación y Evaluación de un Sistema de Gestión utilizando Patrones de Diseño**

**Integrantes del Grupo:**

- Ariel Arias

- Erick Murillo

- Cesar Mera

- Adrián Toledo

Tabla de contenido

[**Introducción** 3](#_Toc188041386)

[**Objetivos Específicos** 3](#_Toc188041387)

[**Resultado de Aprendizaje** 3](#_Toc188041388)

[**UML corregido y retroalimentación del profesor** 4](#_Toc188041389)

[**Modificación del patrón Abstract Factory para apuestas simples** 4](#_Toc188041390)

[**Incorporación del patrón de diseño Singleton al sistema de apuestas** 4](#_Toc188041391)

[**Flexibilidad del diseño original para incorporar cambios** 4](#_Toc188041392)

[**Beneficios y limitaciones de los patrones de diseño aplicados** 5](#_Toc188041393)

# **Introducción**

Este informe presenta la implementación de un sistema de gestión diseñado mediante el uso de patrones de diseño, siguiendo las especificaciones previas en UML, y desarrollando el código correspondiente en Java. El objetivo principal de este proyecto es asegurar la funcionalidad básica del sistema sin necesidad de una interfaz gráfica, y evaluar cómo los patrones de diseño utilizados facilitan la incorporación de nuevos requerimientos y la implementación de cambios en el sistema. Durante el proceso, se utilizó una metodología orientada a objetos para estructurar las clases, interfaces y métodos según los diagramas UML entregados, asegurando la escalabilidad y mantenibilidad del sistema. Además, se evaluaron los beneficios y limitaciones de los patrones de diseño aplicados, tales como Factory Method, Observer, Strategy y siglenton para entender su impacto en la flexibilidad del sistema.

## **Objetivos Específicos**

1. Completar la implementación del proyecto utilizando los diagramas UML y patrones de diseño previamente establecidos.
2. Evaluar la flexibilidad y facilidad proporcionada por los patrones de diseño al incorporar nuevos requerimientos y realizar cambios en el sistema.

# **Resultado de Aprendizaje**

1. Capacidad para implementar sistemas siguiendo los patrones de diseño preestablecidos, respetando la estructura y los diagramas UML entregados.
2. Habilidad para evaluar la robustez y extensibilidad del diseño implementado al realizar cambios o agregar nuevos requerimientos en el sistema.

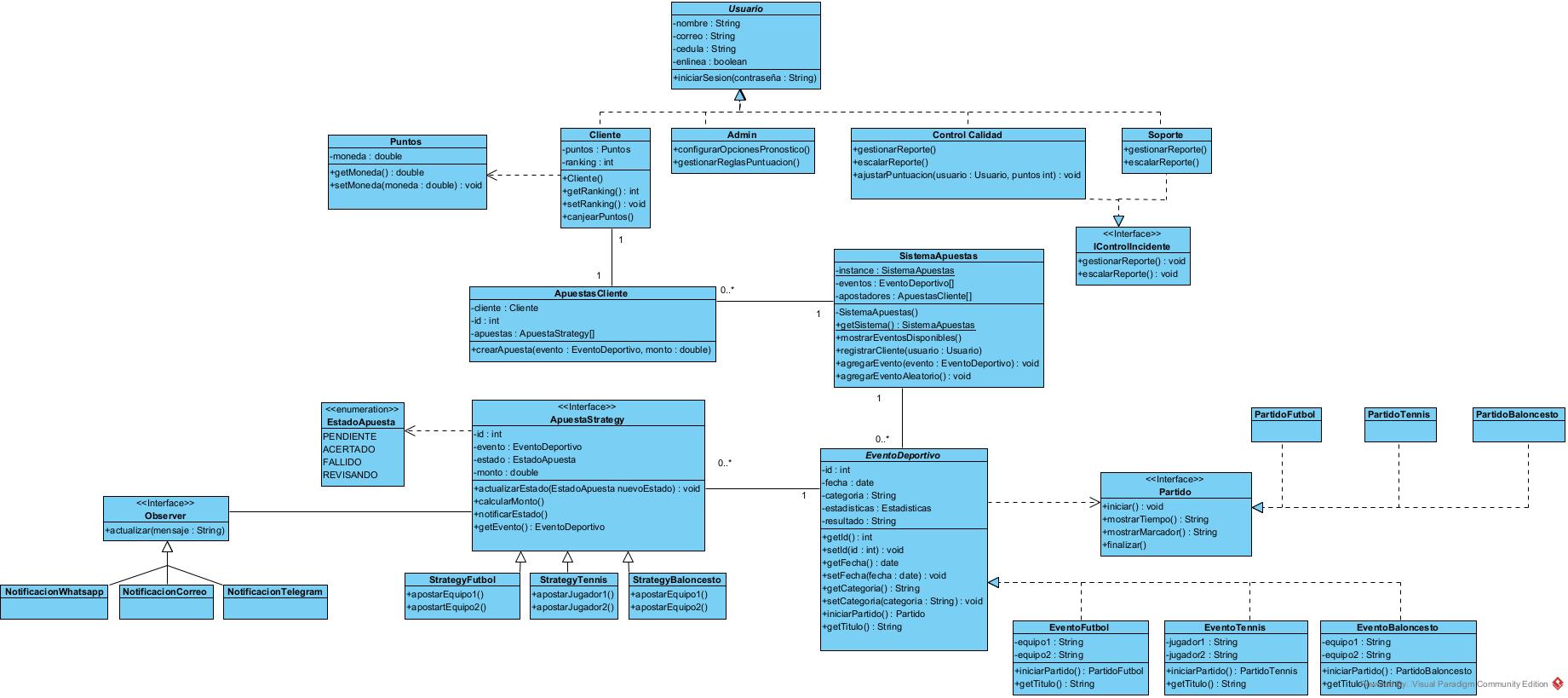
# **UML corregido y retroalimentación del profesor**

## **Modificación del patrón Abstract Factory para apuestas simples**

El patrón de diseño Abstract Factory puede ser modificado para optimizarlo y enfocarlo exclusivamente en la creación de apuestas simples, eliminando la complejidad innecesaria y mejorando la eficiencia.

## **Incorporación del patrón de diseño Singleton al sistema de apuestas**

El patrón de diseño Singleton es una solución ideal para garantizar que ciertas clases clave del sistema de apuestas tengan una única instancia global, controlando su acceso y mejorando la eficiencia. Su incorporación al sistema puede implementarse de la siguiente manera:



# **Flexibilidad del diseño original para incorporar cambios**

El diseño del sistema muestra una estructura bien organizada con la aplicación de varios patrones de diseño, lo que facilita la extensibilidad y la incorporación de nuevas funcionalidades. Entre los aspectos clave:

**Uso de interfaces y enumeraciones:** Permite incorporar nuevos comportamientos o estados sin modificar las clases existentes, por ejemplo, la interfaz ApuestaStrategy puede extenderse para soportar más tipos de apuestas.

**Aplicación de la estrategia y el patrón observador:** Estos patrones permiten una separación clara de responsabilidades, facilitando la adición de nuevos modos de notificación o reglas de apuestas.

**Clases bien definidas:** Las responsabilidades están distribuidas en diferentes clases, reduciendo el acoplamiento y mejorando la modularidad.

**Limitaciones detectadas:** El diseño podría volverse rígido si se necesitan modificaciones profundas en el comportamiento de las estrategias (ApuestaStrategy) o en la funcionalidad de los observadores (Observer). Las relaciones entre clases, aunque claras, podrían complicarse al escalar el sistema con nuevos deportes o eventos.

# **Beneficios y limitaciones de los patrones de diseño aplicados**

**Beneficios:**

*Patrón estrategia:* Ofrece flexibilidad al permitir cambiar las reglas de apuesta (por ejemplo, StrategyFutbol, StrategyTennis) sin modificar el resto del sistema.

*Patrón observador:* Permite una notificación eficiente y descentralizada a través de múltiples canales (WhatsApp, correo, Telegram) sin depender de la clase principal.

*Singleton (SistemaApuestas):* Garantiza una instancia única para gestionar el sistema, manteniendo el control centralizado.

**Limitaciones:**

*Complejidad adicional:* La inclusión de múltiples patrones puede aumentar la curva de aprendizaje para nuevos desarrolladores.

*Dependencia fuerte en interfaces:* Si bien ayuda con la extensibilidad, una implementación incorrecta podría aumentar el acoplamiento, dificultando cambios masivos.

*Escalabilidad limitada del patrón observado*r*:* A medida que crecen los usuarios o canales de notificación, el rendimiento puede degradarse si no se gestiona adecuadamente.