**Evaluación 2 Arquitectura de Software**

**Complementando las exposiciones de Patrones de Comportamiento**

**Valor: 35 puntos**

**Condiciones**

* Cada reto consiste en tomar un aplicativo genérico que ya existe.
* Ustedes deberán para cada módulo de ese aplicativo, identificar unas clases genéricas dependiendo de lo que haga cada módulo. Por ejemplo: un **Módulo de Productos** que Administra el catálogo y stock, deberá tener mínimo clases como Producto y RegistroInventario.
* Estas clases deben ir en un diagrama compacto UML, inicialmente con las relaciones que sean necesarias.
* Luego, deberán incorporar las clases e interfaces que necesitan los patrones sugeridos. Si considera que cabe otro patrón, son libres de implementarlos.
* Solo las clases para diseñar los patrones deben ir con atributos y métodos propios del patrón(atributos que implementen las relaciones, métodos especiales del patrón, constructores con inyección, etc.). Las interfaces por obvias razones, solo con atributos. Recuerden tener en cuenta cuando implementan una interface y cuándo inyectan una dependencia.
* No debe quedar ninguna clase sin relacionarse en el diagrama.
* Recomendación: pinte las clases de cada patrón con colores distintos.
* Forma de presentación: Vídeo, donde muestren el diagrama y lo expliquen con detalle, argumentando muy bien los patrones. Tiempo máximo: 10 minutos. Esta vez, para quienes no lo hicieron, DEBEN SALIR EN EL VÍDEO LOS INTEGRANTES, EXPLICANDO EL CÓDIGO.

**Rúbrica**

* **Diagrama de clases con los patrones y las clases relacionadas con los módulos 15 puntos**
* **Explicación y justificación del diseño de acuerdo con la teoría de cada patrón, 20 puntos.**

**MIGRACIÓN DE UNA APLICACIÓN MONOLÍTICA A MICROSERVICIOS**

**📌 CONTEXTO EMPRESARIAL**

**Empresa:** **FastCommerce** - Plataforma de e-commerce B2B.

FastCommerce ha crecido rápidamente y ahora atiende a miles de clientes. Sin embargo, su sistema monolítico dificulta la implementación de nuevas funciones y genera fallos frecuentes en tiempos de alta demanda.

**📌 AS IS (ESTADO ACTUAL DEL SISTEMA)**

**Módulos actuales:**

1. **Módulo de Usuarios:** Maneja autenticación y permisos.
2. **Módulo de Productos:** Administra el catálogo y stock.
3. **Módulo de Pedidos:** Procesa órdenes y genera facturas.
4. **Módulo de Pagos:** Procesa transacciones y aplica impuestos.
5. **Módulo de Notificaciones:** Envía alertas de estado de pedidos.

**⚠️ PROBLEMAS**

* Alto acoplamiento entre módulos.
* No es posible escalar módulos individualmente.
* Fallas en un módulo pueden afectar todo el sistema.

**🚀 DESAFÍOS**

* Aplicar **Facade**, **Adapter** y **Repository** para desacoplar componentes.
* Usar **Singleton** para gestionar conexiones de base de datos.
* Aplicar **Chain of Responsibility** para manejar solicitudes de servicios en la API Gateway y permitir flexibilidad en la gestión de tráfico.

**🔍 ENFOQUE SOLID**

* **SRP (Responsabilidad Única):** Separar funcionalidades en servicios independientes.
* **DIP (Inversión de Dependencias):** Usar interfaces para desacoplar módulos.

**🎯 OBJETIVO DEL REDISEÑO**

Este reto busca que antes de diseñar el componente de microservicios, aplicar patrones y generar un nuevo diseño a alto nivel que permita ver cómo va a ser la nueva arquitectura a la que se va a migrar separando módulos como usuarios, productos, pedidos y pagos.

1. **APLICACIÓN DE PAGOS CON REGLAS POR PAÍS**

📌 **CONTEXTO EMPRESARIAL:** *PayGlobal - Plataforma de pagos internacionales*

📌 **AS IS (ESTADO ACTUAL DEL SISTEMA)**

1. **Módulo de Autenticación**
2. **Módulo de Procesamiento de Pago**
3. **Módulo de Cálculo de Impuestos**
4. **Módulo de Conversión de Moneda**
5. **Módulo de Facturación**

⚠️ **PROBLEMAS**

* Código monolítico con lógica de negocio en condicionales (if-else).
* Dificultad para agregar nuevos países sin afectar otras reglas.

🚀 **DESAFÍOS**

* Usar **Strategy** para calcular impuestos y tarifas por país.
* Implementar **Factory Method** para crear procesadores de pago dinámicos.
* Aplicar **Decorator** para agregar impuestos sin modificar la lógica central.
* Utilizar **Command** para encapsular la lógica de procesamiento de pagos y permitir su ejecución en distintos contextos, como reembolsos o transacciones diferidas.

🔍 **ENFOQUE SOLID**

* **OCP (Abierto/Cerrado):** Agregar nuevas reglas sin modificar código existente.

🎯 **OBJETIVO DEL REDISEÑO**  
Crear un sistema flexible que permita **agregar nuevos países sin modificar el código central**.

1. **PLATAFORMA DE E-LEARNING CON MÚLTIPLES MODOS DE ENTREGA**

📌 **CONTEXTO EMPRESARIAL:** *EduTech360 - Plataforma de educación en línea*

📌 **AS IS (ESTADO ACTUAL DEL SISTEMA)**

1. **Módulo de Usuarios**
2. **Módulo de Contenidos**
3. **Módulo de Evaluaciones**
4. **Módulo de Notificaciones**

⚠️ **PROBLEMAS**

* Código difícil de modificar al agregar nuevos formatos.

🚀 **DESAFÍOS**

* Usar **Factory Method** para gestionar la creación de cursos.
* Aplicar **Observer** para notificar a los estudiantes sobre nuevas lecciones o cambios en el curso.
* Implementar **State** para manejar los diferentes estados de una clase (en vivo, grabada, programada), permitiendo cambios dinámicos en su comportamiento.

🔍 **ENFOQUE SOLID**

* **DIP (Inversión de Dependencias):** Modularizar formatos de contenido.

🎯 **OBJETIVO DEL REDISEÑO**  
Permitir la fácil incorporación de nuevos formatos con **Factory Method**, **State** y **Observer**.

1. **SISTEMA DE RESERVAS DE HOTEL CON DESCUENTOS Y PROMOCIONES**

📌 **CONTEXTO EMPRESARIAL:** *StayNow - Plataforma global de reservas de hoteles*

📌 **AS IS (ESTADO ACTUAL DEL SISTEMA)**

1. **Módulo de Clientes**
2. **Módulo de Habitaciones**
3. **Módulo de Reservas**
4. **Módulo de Pagos**
5. **Módulo de Descuentos**

⚠️ **PROBLEMAS**

* Reglas de descuentos difíciles de modificar.

🚀 **DESAFÍOS**

* Usar **Decorator** para agregar descuentos dinámicamente.
* Aplicar **Strategy** para gestionar reglas de precios de manera flexible.
* Implementar **Template Method** para definir la estructura de cálculo de precios y permitir variaciones específicas según temporada o cliente.

🔍 **ENFOQUE SOLID**

* **OCP (Abierto/Cerrado):** Agregar nuevas promociones sin cambiar código existente.

🎯 **OBJETIVO DEL REDISEÑO**  
Facilitar la combinación de múltiples promociones con **Decorator**, **Strategy y Template**.

1. **CHAT EN TIEMPO REAL CON MÚLTIPLES PROTOCOLOS**

📌 **CONTEXTO EMPRESARIAL:** *ConnectNow - Plataforma de mensajería empresarial*

📌 **AS IS (ESTADO ACTUAL DEL SISTEMA)**

1. **Módulo de Usuarios**
2. **Módulo de Mensajes**
3. **Módulo de Notificaciones**
4. **Módulo de Conectividad**

⚠️ **PROBLEMAS**

* Código poco mantenible y difícil de escalar.

🚀 **DESAFÍOS**

* Implementar **Bridge** para desacoplar protocolos de mensajería.
* Usar **Factory Method** para gestionar diferentes clientes de chat.
* Aplicar **Observer** para permitir que los usuarios se suscriban a eventos de mensajes y recibir notificaciones en tiempo real.

🔍 **ENFOQUE SOLID**

* **LSP (Sustitución de Liskov):** Asegurar compatibilidad entre protocolos.

🎯 **OBJETIVO DEL REDISEÑO**  
Modularizar los protocolos de comunicación con **Bridge**, **Observer** y **Factory Method**.

1. **SISTEMA DE REPORTES CON MÚLTIPLES FORMATOS DE EXPORTACIÓN**

**📌 CONTEXTO EMPRESARIAL**

**Empresa:** **DataVizPro** - Plataforma de análisis de datos para empresas.

DataVizPro genera reportes detallados en diferentes formatos, pero su arquitectura actual hace que agregar nuevos tipos de exportación sea complejo y propenso a errores.

**📌 AS IS (ESTADO ACTUAL DEL SISTEMA)**

**Módulos actuales:**

1. **Módulo de Datos:** Recopila y procesa información.
2. **Módulo de Exportación:** Genera archivos en PDF, Excel y JSON.

**⚠️ PROBLEMAS**

* Código duplicado en cada formato de exportación.
* Dificultad para agregar nuevos formatos sin modificar múltiples partes del código.

**🚀 DESAFÍOS**

* Implementar **Factory Method** para generar exportaciones de manera flexible.
* Aplicar **Bridge** para desacoplar la generación de reportes de los formatos de exportación.
* Usar **Memento** para permitir la restauración de versiones anteriores de reportes antes de su exportación final.

**🔍 ENFOQUE SOLID**

* **OCP (Abierto/Cerrado):** Permitir agregar nuevos formatos sin modificar código existente.

**🎯 OBJETIVO DEL REDISEÑO**

Facilitar la generación de reportes en distintos formatos utilizando **Factory Method**, **Bridge y Memento**.

1. **PLATAFORMA DE E-COMMERCE CON DIFERENTES ESTRATEGIAS DE ENVÍO**

**📌 CONTEXTO EMPRESARIAL**

**Empresa:** **ShipEasy** - Plataforma de gestión de envíos para tiendas en línea.

ShipEasy ofrece diferentes opciones de envío, pero su lógica de cálculo de costos y tiempos está mezclada, lo que dificulta agregar nuevos transportistas o métodos de entrega.

**📌 AS IS (ESTADO ACTUAL DEL SISTEMA)**

**Módulos actuales:**

1. **Módulo de Logística:** Calcula costos y tiempos de envío.
2. **Módulo de Transportistas:** Gestiona transportistas como DHL y UPS.

**⚠️ PROBLEMAS**

* Todos los tipos de envío están en el mismo módulo (if-else).
* No permite agregar fácilmente nuevos transportistas o estrategias de envío.

**🚀 DESAFÍOS**

* Aplicar **Strategy** para manejar diferentes opciones de envío.
* Usar **Singleton** para administrar configuraciones de logística.
* Implementar **Observer** para que los clientes reciban actualizaciones en tiempo real sobre el estado de su pedido.

**🔍 ENFOQUE SOLID**

* **LSP (Sustitución de Liskov):** Asegurar que cada estrategia de envío cumpla con la misma interfaz.

**🎯 OBJETIVO DEL REDISEÑO**

Facilitar la incorporación de nuevas opciones de envío sin modificar el código existente, utilizando **Strategy**, **Singleton Observer**.

1. **MIGRACIÓN DE UN SISTEMA LEGADO A UNA NUEVA ARQUITECTURA**

**📌 CONTEXTO EMPRESARIAL**

**Empresa:** **BankSecure** - Plataforma de banca en línea utilizada por millones de clientes.

El sistema de BankSecure fue desarrollado hace más de 15 años y es difícil de actualizar. La empresa necesita modernizarlo sin afectar su operación diaria.

**📌 AS IS (ESTADO ACTUAL DEL SISTEMA)**

**Módulos actuales:**

1. **Módulo de Clientes:** Gestiona cuentas y datos personales.
2. **Módulo de Transacciones:** Procesa depósitos y retiros.
3. **Módulo de Reportes:** Genera estados de cuenta.

**⚠️ PROBLEMAS**

* Tecnología obsoleta que no es compatible con APIs modernas.
* Dificultad para integrar nuevos servicios como pagos móviles o autenticación biométrica.

**🚀 DESAFÍOS**

* Usar **Adapter** para conectar la API con el sistema heredado, de forma genérica.
* Aplicar **Facade** para simplificar el acceso a funcionalidades antiguas.
* Implementar **Mediator** para gestionar la comunicación entre módulos antiguos y nuevos, evitando dependencias directas.

**🔍 ENFOQUE SOLID**

* **DIP (Inversión de Dependencias):** Permitir la transición sin afectar el código base.

**🎯 OBJETIVO DEL REDISEÑO**

Migrar el sistema a una arquitectura basada en APIs, utilizando **Adapter**, **Facade y Mediator** para minimizar el impacto en el código existente.

1. **PLATAFORMA DE IOT CON DISPOSITIVOS DE MÚLTIPLES FABRICANTES**

**📌 CONTEXTO EMPRESARIAL**

**Empresa:** **SmartHomeHub** - Plataforma de control inteligente para el hogar.

SmartHomeHub conecta dispositivos de diferentes marcas, como luces, termostatos y cámaras de seguridad, pero la integración es complicada porque cada fabricante usa su propio protocolo.

**📌 AS IS (ESTADO ACTUAL DEL SISTEMA)**

**Módulos actuales:**

1. **Módulo de Dispositivos:** Registra sensores y actuadores.
2. **Módulo de Comunicación:** Conecta con protocolos propietarios.

**⚠️ PROBLEMAS**

* Cada fabricante requiere código específico (if-else).
* No es flexible para nuevos dispositivos.

**🚀 DESAFÍOS**

* Aplicar **Adapter** para conectar distintos protocolos de dispositivos.
* Usar **Bridge** para desacoplar los dispositivos del sistema central.
* Implementar **Observer** para permitir que dispositivos IoT envíen actualizaciones en tiempo real a la plataforma central.

**🔍 ENFOQUE SOLID**

* **ISP (Segregación de Interfaces):** Asegurar que cada dispositivo implemente solo las funcionalidades que necesita.

**🎯 OBJETIVO DEL REDISEÑO**

Crear una arquitectura flexible para la integración de nuevos dispositivos utilizando **Adapter**, **Bridge y Observer**.