Tarea 12: Patrones de Diseño, Estandares en servicios, Plataformas Tecnologicas y Seguridad en Computo en la Nube.

Patrones de Diseño:

¿Qué es un Patrón de Diseño?
“Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software.”

En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Debemos tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

Existen varios patrones de diseño popularmente conocidos, los cuales se clasifican como se muestra a continuación:

- **Patrones Creacionales:** Inicialización y configuración de objetos.
- **Patrones Estructurales:** Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- **Patrones de Comportamiento:** Más que describir objetos o clases, describen la comunicación entre ellos.

Ejemplos:

Patrones Creacionales

Fábrica Abstracta (Abstract Factory)
El problema a solucionar por este patrón es el de crear diferentes familias de objetos, como por ejemplo la creación de interfaces gráficas de distintos tipos (ventana, menú, botón, etc.).
Método de Fabricación (Factory Method)
Parte del principio de que las subclases determinan la clase a implementar.

```
public class ConcreteCreator extends Creator
  {
  protected Product FactoryMethod()
     {
         return new ConcreteProduct();
     }
}
public interface Product{}
public class ConcreteProduct implements Product{}
     public class Client
     {
         public static void main(String args[])
         {
              Creator UnCreator;
              UnCreator = new ConcreteCreator();
              UnCreator.AnOperations();
```

```
        }
    }
```

Prototipado (Prototype)
Se basa en la clonación de ejemplares copiándolos de un prototipo.
Singleton
Restringe la instanciación de una clase o valor de un tipo a un solo objeto.

```
    public sealed class Singleton
    {
        private static volatile Singleton instance;
        private static object syncRoot = new Object();
        private Singleton()
        {
            System.Windows.Forms.MessageBox.Show("Nuevo Singleton");
        }
        public static Singleton GetInstance
        {
            get
            {
                if (instance == null)
                {
                    lock(syncRoot)
                    {
                        if (instance == null)
                            instance = new Singleton();
                    }
                }
                return instance;
            }
        }
     }
```

MVC (Model View Controler)
Este patrón plantea la separación del problema en tres capas: la capa model, que representa la realidad; la capa controler , que conoce los métodos y atributos del modelo, recibe y realiza lo que el usuario quiere hacer; y la capa vista, que muestra un aspecto del modelo y es utilizada por la capa anterior para interaccionar con el usuario.

Patrones Estructurales

- **Adaptador (Adapter):** Convierte una interfaz en otra.
- **Puente (Bridge):** Desacopla una abstracción de su implementación permitiendo modificarlas independientemente.
- **Objeto Compuesto (Composite):** Utilizado para construir objetos complejos a partir de otros más simples, utilizando para ello la composición recursiva y una estructura de árbol.
- **Envoltorio (Decorator):** Permite añadir dinámicamente funcionalidad a una clase existente, evitando heredar sucesivas clases para incorporar la nueva funcionalidad.
- **Fachada (Facade):** Permite simplificar la interfaz para un subsistema.
- **Peso Ligero (Flyweight):** Elimina la redundancia o la reduce cuando tenemos gran cantidad de objetos con información idéntica.
- **Apoderado (Proxy):** Un objeto se aproxima a otro.

Patrones de Comportamiento

- **Cadena de responsabilidad (Chain of responsibility):** La base es permitir que más de un objeto tenga la posibilidad de atender una petición.
- **Orden (Command):** Encapsula una petición como un objeto dando la posibilidad de "deshacer" la petición.
- **Intérprete (Interpreter):** Intérprete de lenguaje para una gramática simple y sencilla.
- **Iterador (Iterator):** Define una interfaz que declara los métodos necesarios para acceder secuencialmente a una colección de objetos sin exponer su estructura interna.
- **Mediador (Mediator):** Coordina las relaciones entre sus asociados. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones.
- **Recuerdo (Memento):** Almacena el estado de un objeto y lo restaura posteriormente.
- **Observador (Observer):** Notificaciones de cambios de estado de un objeto.

```
Public Class Articulo
  Delegate Sub DelegadoCambiaPrecio(ByVal unPrecio As Object)
  Public Event CambiaPrecio As DelegadoCambiaPrecio
  Dim _cambiaPrecio As Object
  Public WriteOnly Property Precio()
    Set(ByVal value As Object)
      _cambiaPrecio = value
      RaiseEvent CambiaPrecio(_cambiaPrecio)
    End Set
  End Property
```

```
End Class
Public Class ArticuloObservador
  Public Sub Notify(ByVal unObjecto As Object)
    Console.WriteLine("El nuevo precio es:" & unObjecto)
  End Sub

End Class
```

- **Estado (Server):** Se utiliza cuando el comportamiento de un objeto cambia dependiendo del estado del mismo.

- **Estrategia (Strategy):** Utilizado para manejar la selección de un algoritmo.

- **Método plantilla (Template Method):** Algoritmo con varios pasos suministrados por una clase derivada.

- **Visitante (Visitor):** Operaciones aplicadas a elementos de una estructura de objetos heterogénea.

Estandares en Servicios:
Cuando los servicios de nube están funcionando sin problemas y los acuerdos de nivel de servicio (SLAs) están en su lugar, las empresas y agencias tal vez quieran transferir los datos a un proveedor distinto (por ejemplo, IBM® SmartCloud), pero descubrir que no pueden por varias razones. Una que se me viene a la mente ocurre cuando las llamadas de API que fueron usadas para almacenar los datos en una nube requieren que los datos estén en un formato que no sea compatible o interoperable con el formato de datos requerido por llamadas de API que un proveedor distinto usa para almacenar los datos en la nube.

La empresa entonces se enfrenta con una falla de transferencia de datos causada por una falla al comparar los formatos de almacenamiento de datos empleados por distintos proveedores antes de que haya seleccionado un proveedor para alojar un servicio de nube. (Una forma potencial de aligerar los daños es negociar con el proveedor que permita una mayor flexibilidad al transferir los datos hacia un proveedor distinto. Esto incluye cambiar el código a las llamadas de API del servicio de nube del proveedor.)

Los clientes de la nube se merecen más que sólo APIs interoperables; necesitan estándares de servicio de nube para asegurar la interoperabilidad para todos los modelos de entrega de la nube:

- Infraestructura como un Servicio: Máquinas virtuales que trabajan para la IaaS alojadas por un proveedor para que sean compatibles con las máquinas virtuales que trabajan para la IaaS alojadas por otro proveedor.
- Plataforma como un Servicio: Plataformas que trabajan en una IaaS para que sean compatibles con cualquier PaaS que trabaje en otro IaaS.
- Software como un Servicio: Aplicaciones desarrolladas en una PaaS para trabajar en una PaaS compatible.

Para ayudarle a iniciarse en hacer estas determinaciones, este artículo proporciona una lista de expectativas de estándares de interoperabilidad que un proveedor o consumidor de servicios de nube debe esperar. Después profundiza más en las organizaciones que están dando forma a estándares en los diversos aspectos de los servicios de nube, de forma que sea posible visitar las apropiadas para sus necesidades y usar sus recursos como herramientas de interoperabilidad. Tal vez incluso quiera participar de las comunidades que se encuentran alrededor de ellas y contribuir con la evolución de los estándares.

Expectativas del cliente de servicios de nube
Un cliente de la nube (consumidor o proveedor de aplicaciones, plataforma o servicios de infraestructura) debe poder esperar interoperabilidad razonable en las siguientes áreas:

- Interoperabilidad del modelo de entrega: especialmente de IaaS a IaaS y de PaaS a PaaS.
- Inferfaces e interacciones basadas en la nube: por ejemplo, interacción entre sistemas de nube y que no son de nube.
- Arquitecturas orientadas a servicios y otros servicios web: soporte para interoperabilidad entre sistemas de nube y arquitecturas de referencia de SOA, infraestructuras y modelos de integración.
- Sistemas de gestión de TI empresarial: estándares para mantener productos distintos de TI actuando como si fueran una sola familia feliz.
- Almacenamiento: sistemas que gestionen el archivado y el acceso para los datos; esto puede ser una función crítica, ya que algunos de estos datos pueden ser un recurso que habilite la función de una aplicación de nube.

- Seguridad: interoperabilidad con protocolos y utilidades que gestionen problemas de seguridad en la nube tales como colas de mensajes, identidad y autenticación y topología de infraestructura y configuraciones de orquestación de aplicación.
- Transición: las herramientas que una organización utiliza para aplicaciones de transición (o incluso para todo el entorno de TI) hacia la nube deben también estar basadas en estándares.
- Un árbitro orientado al usuario: si una organización muy grande que en teoría es controlada por sus usuarios (gobiernos federales, por ejemplo) fuera a establecer estándares de interoperabilidad de la nube, aliviaría el "dolor" que algunos fabricantes de productos de nube sienten al hacer la ingeniería (¿re-ingeniería?) de sus productos para interoperabilidad el saber que una gran porción del mercado está obligada a aceptar un estándar.

La interoperabilidad es más fácilmente establecida por la creación, adopción y refinamiento de estándares.

Plataformas tecnológicas

Las Plataformas Tecnológicas son estructuras público-privadas de trabajo en equipo lideradas por la industria, en las que todos los agentes sistema español de Ciencia-Tecnología-Innovación interesados en un campo tecnológico trabajan conjunta y coordinadamente para identificar y priorizar las necesidades tecnológicas, de investigación y de innovación a medio o largo plazo.

Su principal objetivo es conseguir los avances científicos y tecnológicos que aseguren la competitividad, la sostenibilidad y el crecimiento de nuestro tejido empresarial, alineando las estrategias de los diferentes agentes y concentrando los esfuerzos de I+D+i.

Seguridad de Computo en la Nube:

La seguridad en la informática en la nube es un servicio de rápido crecimiento que ofrece muchas de las funciones que tiene la seguridad de TI tradicional. Esto incluye la protección de información crítica frente al robo, la filtración de datos y la eliminación.

Una de las ventajas de los servicios en la nube es que puede operar a escala y seguir disfrutando de protección. Es similar al modo en que administra la seguridad en la actualidad, con la diferencia de que existen otras formas de proporcionar soluciones de seguridad que afrontan otros aspectos preocupantes. La seguridad en la nube no cambia el enfoque de administración de seguridad en torno a la prevención, la detección y la resolución. Sin embargo, le permite realizar estas actividades de manera más ágil.

Sus datos están protegidos en centros de datos. Como algunos países exigen que los datos se almacenen dentro del país, le puede resultar útil elegir un socio que disponga de varios centros de datos en todo el mundo.

El almacenamiento de datos suele exigir ciertos requisitos de conformidad, sobre todo para guardar números de tarjetas de crédito o información sanitaria. Muchos proveedores de la nube ofrecen informes de auditorías de terceros para garantizar que disponen de procesos internos y su eficacia a la hora de administrar la seguridad en las instalaciones en las que se almacenan sus datos.