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

Design patterns:

What is a Design Pattern?
"Design patterns are the skeleton of solutions to common problems in software development."

In other words, they provide an already proven and documented solution to software development problems that are subject to similar contexts. We must keep in mind the following elements of a pattern: its name, the problem (when applying a pattern), the solution (abstract description of the problem) and the consequences (costs and benefits).

There are several popularly known design patterns, which are classified as shown below:

- **Creational patterns:** Initialization and configuration of objects.

- **Structural patterns:** Separate the implementation interface. They deal with how classes and objects are grouped, to form larger structures.

- **Behavior Patterns:** More than describing objects or classes, they describe the communication between them.

Examples:
Creational patterns
Abstract Factory (Abstract Factory)
The problem to be solved by this pattern is to create different families of objects, such as the creation of graphic interfaces of different types (window, menu, button, etc.).
Manufacturing Method (Factory Method)
It starts from the principle that the subclasses determine the class to be implemented.

```
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();
          }
     }
```

Prototyping (Prototype)
It is based on the cloning of copies by copying them from a prototype.
Singleton
Restricts the instantiation of a class or value of a type to a single object.

```
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)
This pattern poses the separation of the problem into three layers: the model layer, which represents reality; the controler layer, which knows the methods and attributes of the model, receives and performs what the user wants to do; and the view layer, which shows an aspect of the model and is used by the previous layer to interact with the user.

Structural patterns

• **Adapter:** Converts one interface into another.

• **Bridge (Bridge):** Decouples an abstraction of its implementation allowing to modify them independently.

• **Composite Object:** Used to construct complex objects from simpler ones, using recursive composition and a tree structure.

• **Wrapper (Decorator):** Allows dynamically add functionality to an existing class, avoiding inheriting successive classes to incorporate the new functionality.

• **Facade:** Allows you to simplify the interface for a subsystem.

• **Flyweight:** Eliminates redundancy or reduces it when we have a large number of objects with identical information.

• **Proxy (Proxy):** One object approaches another.

Behavior patterns

• **Chain of responsibility:** The basis is to allow more than one object to be able to meet a request.

• **Command:** Encapsulates a request as an object giving the possibility of "undoing" the request.

• **Interpreter (Interpreter):** Language interpreter for a simple and simple grammar.

• **Iterator:** Defines an interface that declares the necessary methods to access a collection of objects sequentially without exposing its internal structure.

• **Mediator (Mediator):** Coordinates the relationships between its members. It allows the interaction of several objects, without generating strong connections in those relationships.

• **Memory (Memento):** Stores the state of an object and restores it later.

• **Observer:** Notifications of changes in the state of an object.

```
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
```

- **State (Server):** Used when the behavior of an object changes depending on the state of the object.

- **Strategy (Strategy):** Used to manage the selection of an algorithm.

- **Template Method:** Algorithm with several steps supplied by a derived class.

- **Visitor:** Operations applied to elements of a heterogeneous object structure.

Standards in Services:

When cloud services are running smoothly and service level agreements (SLAs) are in place, companies and agencies may want to transfer data to a different provider (for example, IBM® SmartCloud), but discover that They can not for several reasons. One that comes to mind occurs when the API calls that were used to store the data in a cloud require that the data be in a format that is not compatible or interoperable with the data format required by API calls that a Different provider uses to store the data in the cloud.

The company is then faced with a data transfer failure caused by a failure to compare the data storage formats used by different providers before it has selected a provider to host a cloud service. (One potential way to lighten the damage is to negotiate with the provider to allow more flexibility in transferring the data to a different provider, this includes changing the code to the API calls from the provider's cloud service.)

Cloud clients deserve more than just interoperable APIs; they need cloud service standards to ensure interoperability for all cloud delivery models:

- **Infrastructure as a Service:** Virtual machines that work for the IaaS hosted by a supplier to be compatible with the virtual machines that work for the IaaS hosted by another provider.
- **Platform as a Service:** Platforms that work in a IaaS to be compatible with any PaaS that works in another IaaS.
- **Software as a Service:** Applications developed in a PaaS to work in a compatible PaaS.

To help you get started in making these determinations, this article provides a list of expectations for interoperability standards that a provider or consumer of cloud services should expect. Afterwards, it goes deeper into the organizations that are shaping standards in the various aspects of cloud services, so that it is possible to visit the appropriate ones for their needs and use their resources as interoperability tools. You may even want to participate in the communities that are around them and contribute to the evolution of the standards.

Expectations of the cloud service customer

A customer of the cloud (consumer or provider of applications, platform or infrastructure services) should be able to expect reasonable interoperability in the following areas:

- **Interoperability of the delivery model:** especially from IaaS to IaaS and from PaaS to PaaS.
- **Inferfaces and cloud-based interactions:** for example, interaction between cloud and non-cloud systems.
- **Architectures oriented to services and other web services:** support for interoperability between cloud systems and SOA reference architectures, infrastructures and integration models.
- **Business IT management systems:** standards for keeping different IT products acting as if they were one happy family.
- **Storage:** systems that manage archiving and access to data; this can be a critical function, since some of this data can be a resource that enables the function of a cloud application.
- **Security:** interoperability with protocols and utilities that manage security problems in the cloud such as message queues, identity and authentication, and infrastructure topology and application orchestration configurations.
- **Transition:** the tools that an organization uses for transition applications (or even for the entire IT environment) to the cloud must also be based on standards.
- **A user-oriented arbiter:** if a very large organization that is theoretically controlled by its users (federal governments, for example) were to establish cloud interoperability standards, it

would alleviate the "pain" that some cloud products manufacturers feel By engineering (re-engineering?) your products for interoperability, knowing that a large portion of the market is bound to accept a standard.

Interoperability is more easily established by the creation, adoption and refinement of standards.

Technological platforms

The Technology Platforms are public-private teamwork structures led by the industry, in which all the Spanish Science-Technology-Innovation System agents interested in a technological field work jointly and in coordination to identify and prioritize the technological, research needs and innovation in the medium or long term.

Its main objective is to achieve the scientific and technological advances that ensure the competitiveness, sustainability and growth of our business fabric, aligning the strategies of the different agents and concentrating the R + D + i efforts.

Cloud Computing Security:

Security in cloud computing is a fast-growing service that offers many of the functions that traditional IT security has. This includes the protection of critical information against theft, data leakage and deletion.

One of the advantages of cloud services is that they can operate at scale and continue to enjoy protection. It is similar to the way it manages security today, with the difference that there are other ways to provide security solutions that face other worrying aspects. Cloud security does not change the security management approach around prevention, detection and resolution. However, it allows you to perform these activities more agilely.

Your data is protected in data centers. As some countries require data to be stored within the country, it may be useful to choose a partner that has several data centers around the world.

Data storage usually requires certain compliance requirements, especially for storing credit card numbers or health information. Many cloud providers offer third-party audit reports to ensure that they have internal processes and their effectiveness in managing security in the facilities where their data is stored.