# NOMBRE: Arnold Mauricio Sinisterra Lozano

**NUMERO DE DOCUMENTO: 1993445613**

# PRUEBA DE CONOCIMIENTO

* 1. ***Patrones de Software: para cada ítem se requiere una definición breve y un ejemplo de una situación donde se deba usar.***
     1. **Patrón Singleton:**

**El objetivo es obtener una sola instancia de una clase en concreto, un ejemplo de estos y colocando como ejemplo código que les he enviado serían las claves de acceso que sueltan la base de datos en mi código encontraran un objeto llamado firebaseConfig, en este caso se puede usar el patrón siglenton.**

* + 1. **Patrón Observador:**

**Se una este tipo de patrones cuando quieres detectar cambios dentro de tu aplicación, ejemplo, si tienes una app de compras y un usuario realiza una, con el uso de este patrón le notificaremos la al vendedor sobre la compra realizada**

1. **Explique Patrón MVC (diagrama de clases):**

**MVC**

**Modelo: se encarga de manejo de los datos con los que opera la aplicación.**

**Vista: es la interfaz grafica de la app, diría que es la parte más importante ya que de esto depende el interés del usuario por la misma**

**Controlado: responde a eventos y solicitudes del usuario**

***Explique se significa Bajo Acoplamiento y Alta Cohesión:***

**Cuando hablamos de bajo acoplamiento en desarrollo de software nos referimos a el nivel de dependencia de nuestros componentes al decir que el acoplamiento es bajo, destacamos que nuestros componentes pueden modificarse sin afectarse entre sí.**

**La alta cohesión en desarrollo es cuando realizamos módulos independientes, que realicen tareas especificas**

* 1. ***Describa las ideas y problemas que originaron SOA, lo que implica la práctica, los beneficios que pueden lograrse y las consecuencias negativas que pueden observarse por desarrollos que utilizan esta práctica.***

Arquitectura orientada a servicios

Prácticas y veneficios: al usar este tipo de practicas integramos todos los servicios en una sola interface ahorrándonos trabajo, tiempo y ayudándonos a realizar un código más leíble y reutilizable.

Consecuencias negativas: diría que al tener este tipo de servicio para manejo de datos y funciones ya que esta trabaja de manera global sobre todas las aplicaciones de la empresa podría dar como consecuencia brechas de información por lo cual no lo utilizaría en organizaciones de gran tamaño, también la comunicación entre aplicaciones de la organización requeriría de tiempo y código.

* 1. ***Describa las ideas y problemas que originaron la Inyección de dependencia y la inversión de control - DI/IOC, lo que implica la práctica, los beneficios que pueden lograrse y las consecuencias negativas que pueden observarse por desarrollos que utilizan esta práctica.***

#### **Delegando responsabilidades.**

Prácticas y veneficios: se refiere a todo diseño se software que requiera un control de flujo de ejecución asiendo de forma automática y transparente sobre este un ejemplo de esto los framework en específico Angular quien con su CLI facilita la creación de código y reutilización del mismo.

Consecuencias negativas: Pueden generar código que no se utilice si no sabes controlar el frame o herramienta que uses con estas características.

* 1. ***Describa las ideas y problemas que originaron la Programación Orientada a Aspectos - AOP, lo que implica la práctica, los beneficios que pueden lograrse y las consecuencias negativas que pueden observarse por desarrollos que utilizan esta práctica.***

Programación Orientada a Aspectos.

**Delegando responsabilidades:** es un patrón de diseño Modular lo cual significa que dividimos nuestra aplicación en varios componentes lo cual nos da la posibilidad de poder reutilizar el código dentro del mismo programa o en otros proyectos.

Consecuencias negativas: puede hacer difícil el leer el código si se trata de proyectos de gran tamaño, aunque hay frames donde se prevé esto.

# PRUEBA DE DESARROLLO.

Desarrollar un TIC-TAC-TOE con las siguientes caracteristicas:

* Se debe utilizar una base de datos real-time como firebase u otra que de este tipo para los servicios.
* La pantalla inicial debe presentar dos opciones: Nueva partida o unirse a partida
* Si se da en nueva partida debe generar un id de partida (valido para un solo usuario que lo utilice), si la opcion es Unirse a partida debe presentar un cuadro de texto para ingredsar el id
* Debe terner un espacio editable para poner el nombre del jugador y por defecto llenarlo con “Jugador 1” al creador y “Jugador 2” al invitado (Debe ser visible y no ediable el nombre del contrincante)
* Cada jugador puede jugar en su respectivo turno, la partida inicia por el jugador 1 la primera vez y cada vez que se reinicie debe iniciar el otro jugador (Se deben alternar por nuevas partidas)
* No se puede sobreescribir sobre un campo ya jugado previamente
* No debe haber confirmacion de movimiento (si se equivoco poniendo la figura, queda como está)
* El juego termina cuando todas las casillas esten llenas o existan 3 figuras iguales en linea recta
* Debe existir un boton de reiniciar la partida que solo aparece cuando el juego ha finalizado.
* Se debe poder jugar en dos navegadores.
* El código fuente debe entregarse en un repositorio de Bitbucket el cual debe ser compartido a este correo.
* En el readme debe especificar que aspectos tenicos se deben tener en cuenta para el correcto despleigue de la aplicación para realizar las pruebas
* El código fuente debe estar debidamente documento.
* Tener en cuenta buenas practicas de codifcacion y patrones de diseño