1. **Criterios tácticos:**

1. **Determine que debe probar (Para esto puede explorar la sección como guste y parta de la condición de que existen errores) para asegurar que se cumple con el mínimo de calidad.**

\*Pruebas de documentación

\*Pruebas de funcionalidad del sitio

\*Pruebas de usabilidad

\*Pruebas de interfaz de usuario

\*Pruebas de compatibilidad (configuración)

\*Pruebas de rendimiento

\*Pruebas de seguridad

\*Pruebas relacionadas con el cambio

\*Pruebas amigables para móviles

\*Prueba Final

**Pruebas de documentación**

Estudia la documentación recibida y examina los diseños finales del sitio

**realiza un plan de prueba en el sitio web para realizar más pruebas).**

• Requerimientos

• Plan de prueba

• Casos de prueba

• Matriz de trazabilidad

**Pruebas de funcionalidad del sitio web**

Lo que hace el sistema

• Enlaces salientes

• Corrección de enlaces internos

• No hay enlaces que conducen a la misma página.

• Los enlaces que se utilizan para enviar correos electrónicos a los administradores del sitio

• Si hay páginas que no están referenciadas

• No hay enlaces rotos

\*Pruebas de formularios para todas las páginas.

Utiliza formularios para la comunicación interactiva con sus usuarios.

• La validez de los datos de entrada.

• Valores permitidos para el campo de datos

• Valores de entrada no válidos para el campo de datos

• Opciones para formularios en los que es posible la eliminación o cualquier otra modificación de los datos.

**Prueba de cookies**

• Prueba un sitio con cookies deshabilitadas

• Probar un sitio con cookies habilitadas

• Verifique que la cookie esté encriptada antes de escribirla en la máquina del usuario

• Compruebe los aspectos de seguridad al eliminar las cookies.

• Si las cookies tienen una duración de acción, se comprueba si están activas en el período de tiempo especificado.

Validación HTML / CSS

• Errores de sintaxis HTML

• Verifique que el sitio esté disponible para las máquinas de búsqueda.

• Verifique que su página web tenga un mapa del sitio preciso en formato XML y HTML

Pruebas funcionales se utilizan:

Selenio automatizada

Usuariosnap pruebas manuales

**Pruebas de usabilidad**

la capacidad del usuario para aprender a operar, preparar entradas para e interpretar salidas de su sitio.

**Pruebas de navegación**

• Todas las páginas de su sitio son comprensibles y fáciles de usar

• Botones, formas y campos son convenientes para el uso

• Hay un acceso al menú principal desde todas las páginas.

Pruebas de contenido lista de verificación

• No hay errores gramaticales y ortográficos.

• Las imágenes se colocan correctamente con los tamaños adecuados.

• Verifique la optimización de la paleta de colores del sitio y los tamaños de fuente

• El contenido debe ser informativo, comprensible, estructurado y lógicamente vinculado

• Las instrucciones son claras y contienen información correcta.

evaluar la usabilidad del portal web

• Es su sitio web comprensible y conveniente?

• ¿Es conveniente la navegación?

• ¿Qué impresión le da al usuario?

• ¿Hay cosas superfluas o innecesarias?

Para las pruebas de usabilidad se utiliza : Zoom de usuario

**Pruebas de interfaz de usuario**

Proporciona para verificar que la interfaz gráfica de usuario de su sitio web cumpla con las especificaciones.

• cumplimiento de las normas de interfaces gráficas.

• Evaluación de elementos de diseño: diseño, colores, fuentes, tamaños de fuente, etiquetas, cuadros de texto, formato de texto, títulos, botones, listas, iconos, enlaces

• Pruebas con diferentes resoluciones de pantalla.

• Pruebas de versiones localizadas: precisión de traducción (multilenguaje, multidivisa), verificación de la longitud de los nombres de los elementos de la interfaz, etc.

• Prueba de la interfaz gráfica de usuario en dispositivos de destino: teléfonos inteligentes y tabletas.

pruebas de interfaz de usuario: FitNesse

**Pruebas de compatibilidad (configuración)**

Para probar su sitio web con cada una de las configuraciones de software y hardware compatibles:

• Configuración del sistema operativo

• Configuración del navegador

• Configuración de la base de datos

Pruebas multiplataforma permite evaluar el trabajo de su sitio en diferentes sistemas operativos

Métodos de prueba de sitios de navegador

Para verificar el trabajo correcto del sitio en diferentes configuraciones de navegador: Mozilla Firefox, Google Chrome, Internet Explorer y Opera, etc.

Pruebas de base de datos se realiza para garantizar el trabajo correcto de su sitio en diferentes configuraciones de base de datos: Oracle, DB2, MySql, MSSQL Server, Sybase.

• Verifique que las fuentes, los gráficos de página, la alineación de página se pueda imprimir correctamente

• Asegúrese de que todas las páginas de su sitio se ajusten al tamaño del papel y al tamaño definido en la opción de impresión.

CrossBrowserTesting por Smart Bear

Para las pruebas de compatibilidad de su sitio web

**Pruebas de rendimiento**

Tiene como objetivo determinar cómo funciona un sistema en términos de capacidad de respuesta y estabilidad bajo una carga determinada.

\*Comprobar el comportamiento del sitio en o más allá de los límites de su carga de trabajo anticipada (Pruebas de estrés)

\*Probando el comportamiento del sitio al aumentar la carga de trabajo (Prueba de carga)

\*Probar la capacidad de trabajar dentro o justo por encima del período aceptable (Pruebas de estabilidad)

\*Prueba del rendimiento del sitio web al aumentar el volumen de datos en la base de datos (Prueba de volumen)

\*Prueba del rendimiento del sitio web cuando múltiples usuarios inician sesión en él. (Prueba de concurrencia)

\*Probar el comportamiento de su sitio cuando la carga de trabajo adicional se da continuamente (Pruebas de resistencia)

\*Prueba de velocidad de carga de página

Apache JMeter para pruebas de rendimiento

**Pruebas de seguridad**

Se realiza para verificar que el sistema de información protege los datos y mantiene la funcionalidad según lo previsto.

Puede simular el ataque de origen malicioso para evaluar el nivel de seguridad de su sitio (Penetración pruebas).

Otro tipo de pruebas de seguridad, Pruebas de vulnerabilidad, permite evaluar el monto total de los riesgos involucrados.

• Asegúrese de que el acceso no autorizado a páginas seguras no sea posible

• Verifique que las sesiones se eliminen automáticamente después de una inactividad prolongada del usuario

• Probar las funciones de seguridad SSL

• Todos los intentos de romper, informar errores, etc. deben registrarse y almacenarse en un archivo separado para su posterior análisis.

• Verifica el trabajo de captcha usando scripts automáticos.

• Asegúrese de que los archivos restringidos no se puedan descargar sin el acceso adecuado

• Asegúrese de que no haya capacidad de inicio de sesión al ingresar la contraseña o el nombre de usuario incorrectos

**Pruebas relacionadas con el cambio**

Pruebas relacionadas con el cambio tiene dos propósitos principales:

• Asegurarse de que todos los errores detectados se han solucionado con éxito (Re-prueba o prueba de confirmación). En pocas palabras, debe ejecutar los casos de prueba que originalmente detectaron los errores nuevamente y esta vez pasan sin ningún problema.

• Asegurando que no han aparecido nuevos defectos después de los cambios. (Pruebas de regresión). Además de los casos de prueba de errores detectados, también contiene casos de prueba que verifican todas las funcionalidades de su sitio.

para teléfonos móviles

• Verificar la compatibilidad con teléfonos inteligentes y tabletas.

• Asegúrate de que la navegación del sitio sea lo más sencilla posible.

• Optimiza el tiempo de carga de tu sitio.

• Asegúrese de que los botones sean lo suficientemente grandes para las personas con dedos grandes

• Optimizar el tamaño de todas las imágenes

• No uses Flash y ventanas emergentes

• Usa viñetas y oraciones cortas.

• Asegúrese de que su número de teléfono esté a un clic de distancia de ser marcado

• Verifique que el sitio web pueda acceder a su ubicación a través de GPS

**Prueba beta**

La etapa final de la prueba preliminar. Como regla general, es realizado por usuarios finales.

2. **Considere la creación de casos de prueba. (Considerar pasos si es necesario)**



3**. De estos casos, clasifique que casos son automatizables y no automatizables**

|  |  |  |
| --- | --- | --- |
| **CASOS** | **DESCRIPCION DE LA PLRUEVA** | **TIPO DE PRUEBA** |
| 1 | INGRESO A LA PAGINAhttp://automationpractice.com/index.php | MANUAL O AUTOMATIZADA |
| 2 | VERIFICACION DE CONTACTOS | MANUAL O AUTOMATIZADA |
| 3 | VERIFICACION DE CAMPOS DE REGISTRARSE: | MANUAL O AUTOMATIZADA |
| 4 | VERIFICACION DE CAMPO BUSCAR | MANUAL O AUTOMATIZADA |
| 5 | VERIFICACION DE BOTON MUJER | MANUAL O AUTOMATIZADA |
| 6 | VERIFICACION DE BOTON VESTIDOS | MANUAL O AUTOMATIZADA |
| 7 | VERIFICACION DE BOTON CAMISETAS | MANUAL O AUTOMATIZADA |
| 8 | VERIFICACION DE BOTON POPULAR | MANUAL O AUTOMATIZADA |
| 9 | VERIFICACION DE BOTON MAS VESTIDOS | MANUAL O AUTOMATIZADA |
| 10 | VERIFICACION DE COMPRA | MANUAL O AUTOMATIZADA |
| 11 | VALIDACION DE REGISTRO EN SERVIDOR | AUTOMATIZADA |
| 12 | VALIDACION DE RENDIMIENTO | AUTOMATIZADA |
| 13 | VALIDACION DE SEGURIDAD | AUTOMATIZADA |

Las pruebas automatizadas son las pruebas de rendimiento, validación de registro en el servidor y de seguridad.

4. **Indique si existen y cuáles son las precondiciones para ejecutar los casos de prueba**.

Las precondiciones son condiciones que determina el resultado de la prueba

En el caso 1 verificación de campos de registros CREA UNA CUENTA:

\*En el campo nombre hay precondición y este caso seria solo ingreso de letras

\*En el campo apellido hay precondición y este caso sería solo ingreso de letras

\*En el campo correo hay precondición y este caso sería ingreso de formato de correo seria Hombre de correo @ servidor de correo y país de donde se creó el correo.

\*En el campo contraseña hay precondición y este caso sería un número mínimo de caracteres para que sea segura, además la clave digitada no se debe ver cuando se ingrese.

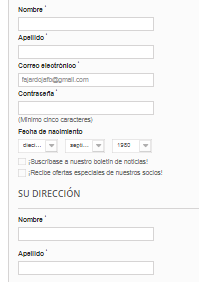
\*En el campo de contactos donde están la información de la empresa y números de contactos son precondiciones ya que es importante que esta información sea legitima.

5. Presente estadísticas de prueba que permitan describir la conclusión de un indicador que defina.

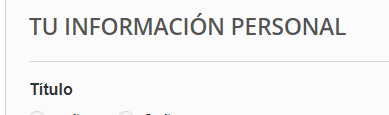
Se realizan varias pruebas y se encuentra que el producto funciona se evidencia presencia de defectos de escritura de algunos campos de formulario, se encontró botón búsqueda no realiza su búsqueda.

**6. Sugiera mejoras al módulo probado**.

La mejora seria no repartir campos que el usuario ya ingreso como son el campo hombre y el campo apellido esta dos veces no es necesario



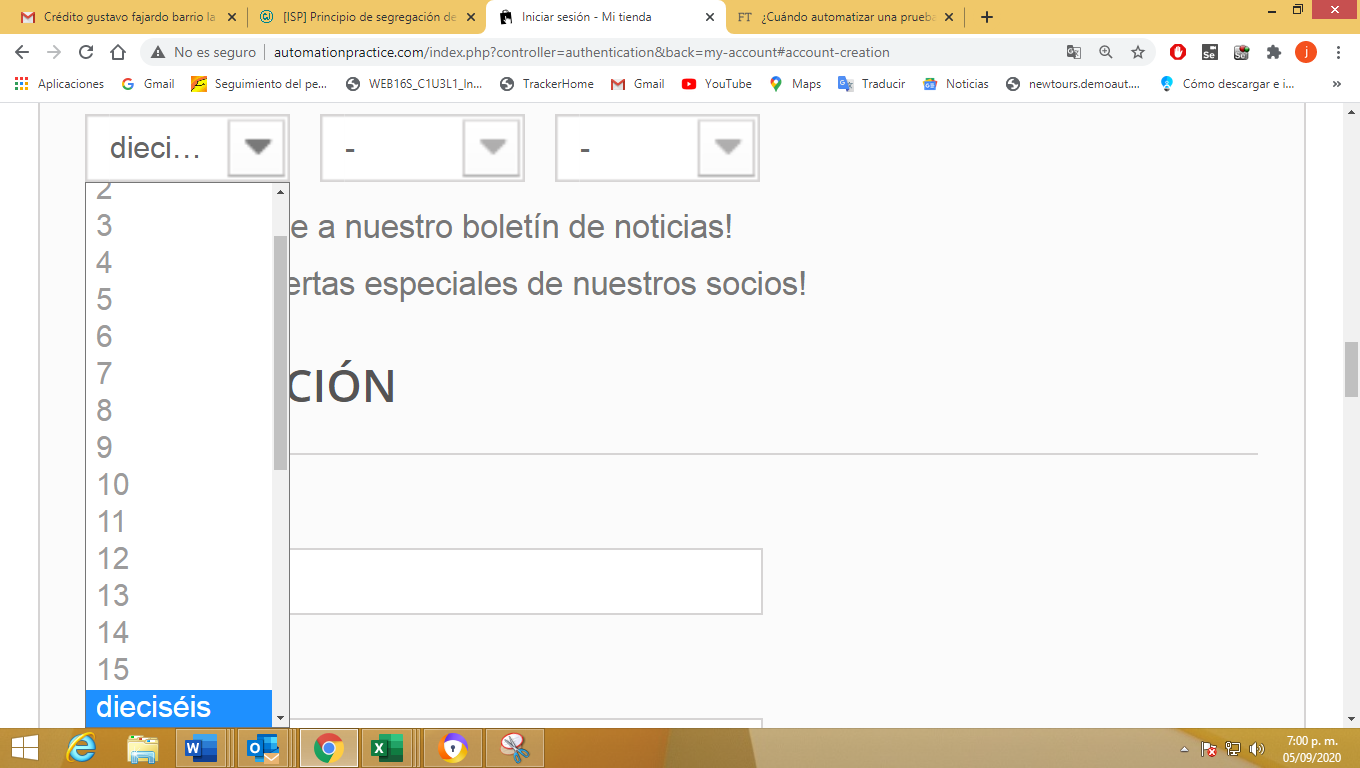
La segunda mejora quitar la palabra titulo no es necesario



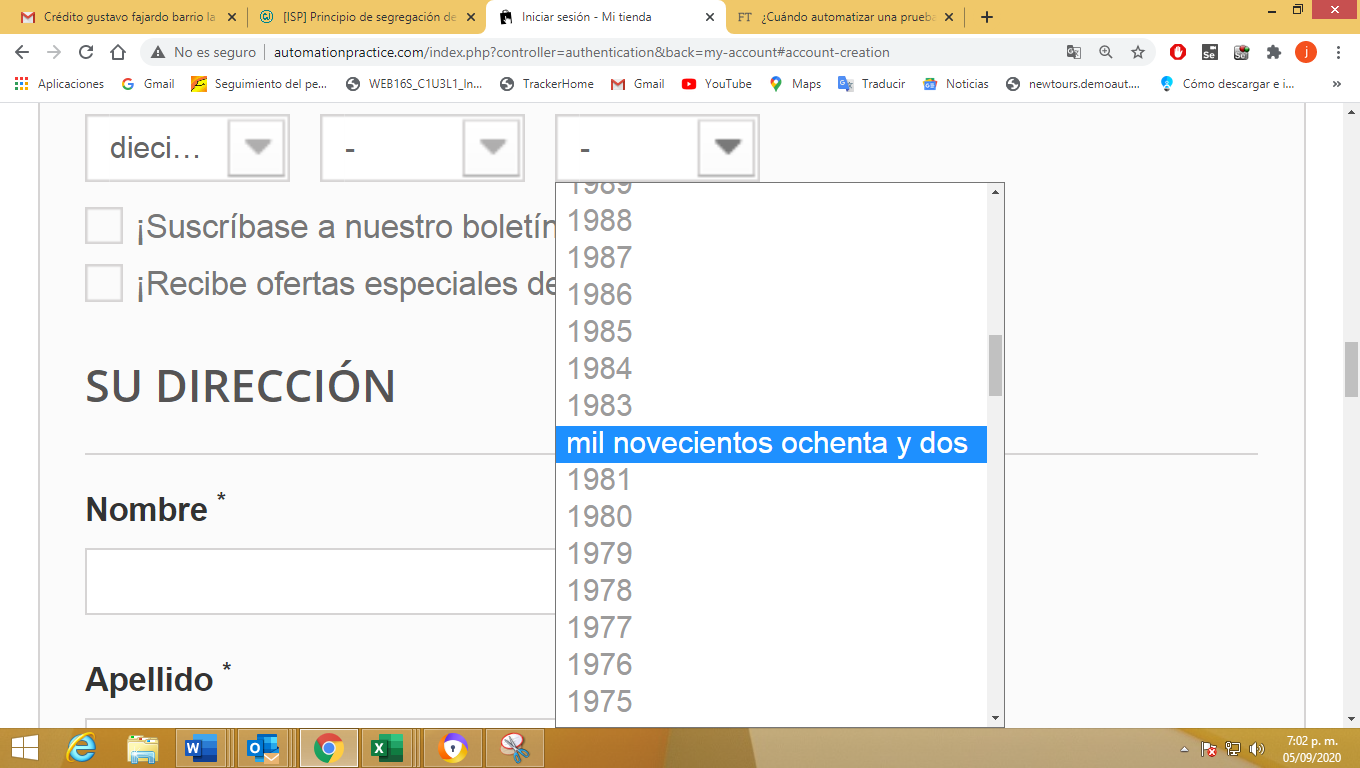
Se debe quitar la selección de señor no hay ropa para señor ni para señora.



Se debería colocar solo números y no mesclar la escritura de números como aparece en la imagen



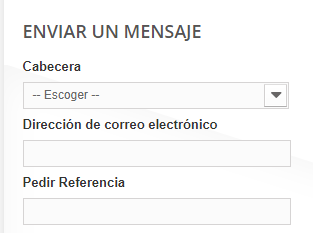
Se evidencia el campo año tiene mezcla de números y también tiene la escritura de un año debería ser solo números



Se evidencia que las etiquetas no tienen el nombre que debería

Cabecera no es necesario que un usuario contacte el desarrollador del sitio web

El titulo pedir referencia no es el nombre correcto



1. **Criterios técnicos:**

**Se realiza las siguientes pruebas**

|  |  |
| --- | --- |
| **caso** | **DESCRIPCION DE LA PLRUEVA** |
| 1 | INGRESO A LA PAGINAhttp://automationpractice.com/index.php |
| 2 | VERIFICACION DE LINK CONTACTENOS |
| 3 | VERIFICACION DE CAMPOS DE REGISTRARCE: |
| 4 | VERIFICACION DE CAMPO BUSCAR |

1. **Agilismo**

1.De las sugerencias de mejora propuestas, defina historias de usuario que plantearía para un nuevo desarrollo, con sus respectivos criterios de aceptación.

\* La mejora seria no repartir campos de el usuario en el formulario de registro de la pagina ya ingreso como son el campo hombre y el campo apellido esta dos veces no es necesario

\* La mejora seria retirar la palabra dirección ubicada usuario en el formulario de registro ya que no está dando ninguna instrucción y confunde.

\*En el formulario de registro aparece la palabra titulo que no brinda ninguna información y no ocupa espacio.

\*El botón buscar no genera ninguna búsqueda.

2. Teniendo en cuenta que estas historias de usuario se seleccionaron en un sprint, liste que tareas de prueba y de desarrollo son necesarias realizar estas historias.

**Tareas de desarrollo**

el formulario de registro de la pagina

\*hombre y apellido repetido

\*retirar texto su SU DIRECION

\*retirar parabra titulo

Pagina principal botón búsqueda funcione.

Lista de pruebas.

\*se debe hacer pruebas de regresión par verificar si se solucionaron los defectos

\*verificar todos los campos de ingreso de datos sean acorde a la información requerida y retirar los campos no necesarios o títulos.

\*los botones generen la acción que se requiere.

1. **Conceptos Programación / CI / CD**

Describa con sus propias palabras cada uno de los siguientes conceptos:

1. Principios SOLID
2. Patrón Singleton
3. Patrón FIRST
4. Patrón AAA
5. Pull Request
6. Release Train
7. Quality Gates
8. Diferencias servicios SOAP / REST

1. **Principios SOLID**

Se conoce como uno de los fundamentos de la [arquitectura y desarrollo de software](https://profile.es/que-hacemos/desarrollo-web/) basándose en los principios de la programación orientada a objetos para obtener código menos propenso a errores.  Convirtiéndose buenas prácticas de desarrollo.

Está relacionado con patrones de diseño, que nos permitirán mantener una alta cohesión y por tanto, un bajo acoplamiento de software.

El acoplamiento se refiere al grado de relación de dependencia mutua y equitativa que tienen dos unidades de software clases, subtipos, métodos, módulos, funciones, bibliotecas, etc.

Si dos unidades de software son completamente independientes la una de la otra, decimos que están desacopladas.

La cohesión de software es elementos diferentes de un sistema permanecen unidos para alcanzar un mejor resultado.

* S – Single Responsibility Principle (SRP)(Principio de responsabilidad única)
* O – Open/Closed Principle (OCP)( Principio abierto / cerrado)
* L – Liskov Substitution Principle (LSP)( [Principio de Sustitución de Liskov](https://profile.es/blog/principios-solid-desarrollo-software-calidad/#3_principio_de_sustitucion_de_liskov))
* I – Interface Segregation Principle (ISP)( [Principio de Segregación de la Interfaz](https://profile.es/blog/principios-solid-desarrollo-software-calidad/#4_principio_de_segregacion_de_la_interfaz))
* D – Dependency Inversion Principle (DIP)( [Principio de Inversión de Dependencias](https://profile.es/blog/principios-solid-desarrollo-software-calidad/#5_principio_de_inversion_de_dependencias))

**S – Single Responsibility Principle (SRP)(Principio de responsabilidad única)**

Una clase debe tener una y solo una única causa por la cual puede ser modificada y cada clase debe ser responsable de realizar una actividad del sistema.

**•O – Open/Closed Principle (OCP)( Principio abierto / cerrado)**

Consiste en tener la posibilidad de poder ampliar la funcionalidad de una clase sin necesidad de tener que modificar su comportamiento interno.

Todas las aplicaciones cambian durante su ciclo de vida, y siempre habrá nuevas versiones que añadirán alguna nueva funcionalidad al diseño y debemos implementar únicamente lo que realmente se requiera.

Son cerradas para la modificación, y por tanto el código fuente de dichas clases debería permanecer inalterado.

Son abiertas para la extensión; es decir, que la lógica o el comportamiento de esas clases puede ser extendida en nuevas clases.

• **L – Liskov Substitution Principle (LSP)( Principio de Sustitución de Liskov**

Cada clase que hereda de otra puede usarse como su padre sin necesidad de conocer las diferencias entre ellas, además promueve la reutilización de código y su mantenimiento.

es una colección de pautas para crear jerarquías de herencia en las cuales un cliente podrá utilizar de manera confiable cualquier clase o subclase.

El uso del polimorfismo se debe usar desde diseño del proyecto. deberíamos poder usar cualquiera de sus subclases sin interferir en la funcionalidad del programa.

* **I – Interface Segregation Principle (ISP)(** [**Principio de Segregación de la Interfaz**](https://profile.es/blog/principios-solid-desarrollo-software-calidad/#4_principio_de_segregacion_de_la_interfaz))

Establece que los clientes de un programa dado sólo deberían conocer de éste aquellos métodos que realmente usan, y no aquellos que no necesitan usar.

Como la priorización de la creación de múltiples interfaces específicas y Como resultado de su aplicación, se facilita mantener una arquitectura desacoplada, resultando más sencillo realizar refactorizaciones y modificaciones.

Las interfaces proporcionan una capa de abstracción que nos ayudan a desacoplar módulos se encuentras varios clientes que emplean las operaciones de la clase.

* **D – Dependency Inversion Principle (DIP)(** [**Principio de Inversión de Dependencias**](https://profile.es/blog/principios-solid-desarrollo-software-calidad/#5_principio_de_inversion_de_dependencias)**)**

es una forma específica de desacoplar módulos de software. Al seguir este principio, las relaciones de dependencia convencionales establecidas desde los módulos de alto nivel de establecimiento de políticas a los módulos de dependencia de bajo nivel se invierten, lo que hace que los módulos de alto nivel sean independientes de los detalles de implementación del módulo de bajo nivel

Las clases de alto nivel no deberían depender de las clases de bajo nivel. Ambas deberían depender de las abstracciones.

**2.Patrón Singleton**

**Es instancia única es un patrón de diseño que permite restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.** **consiste en garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella.**

**se puede acceder a la instancia desde cualquier parte del código.**

**3.Patrón FIRST**

**FIRST es el acrónimo de las cinco características que deben tener nuestros tests unitarios para ser considerados tests con calidad**

**Fast (rápido)**

**Independent (independiente)**

**Repeatable (repetible)**

**Self-validating (auto evaluable)**

**Timely (oportuno)**

**4.Patrón AAA**

Es utilizado en pruebas unitaria :

Arrange (Organizar/Inicializa) => inicializa los objetos y establece los valores de los datos que vamos a utilizar en el Test que lo contiene

.

Act (Actuar) => realiza la llamada al método a probar con los parámetros preparados para tal fin.

Assert (Confirmar/Comprobar) => comprueba que el método de pruebas ejecutado se comporta tal y como teníamos previsto que lo hiciera.

El primer beneficio y más evidente que cualquier otro es que la organización de nuestros Tests hace que cualquier desarrollador pueda entenderlos rápidamente.

Éstos quedarán más claros y quedarán explicados por sí solos.

5.Pull Request

Una Pull Request es la acción de validar un código que se va a mergear de una rama a otra. En este proceso de validación pueden entrar los factores que queramos: Builds (validaciones automáticas), asignación de código a tareas, validaciones manuales por parte del equipo.

6.Release Train

es un modo de planificar la entrega de software según un calendario predeterminado y regular, como si de un «horario de trenes» se tratara. Este calendario es público para todos los equipos que deben contribuir a la entrega y representa un compromiso por su parte, pues el calendario debe ser cumplido. Si usamos la metáfora: «el tren se llevará a los pasajeros que estén listos para viajar», es decir, se integrará sólo el software que esté listo para ser integrado.

7. Quality Gates

Es la mejor manera de hacer cumplir una política de calidad en su organización con un conjunto de condiciones booleanas basadas en umbrales de medida contra los cuales se miden los proyectos.

Tres métricas le permiten hacer cumplir una calificación determinada de confiabilidad, seguridad y capacidad de mantenimiento, no solo en general, sino también en el código nuevo. Estas métricas se recomiendan y forman parte de la puerta de calidad predeterminada. Le recomendamos encarecidamente que ajuste sus propias puertas de calidad para usarlas y que los comentarios sean más claros para sus desarrolladores que miran su puerta de calidad en la página de su proyecto.

8.Diferencias servicios SOAP / REST

Diferencias

**SOAP**

WSDL define el esquema entre el cliente y el servicio y es estático por su naturaleza.

SOAP construye un protocolo basado en XML sobre HTTP o, a veces, TCP / IP.

SOAP es un sucesor de XML-RPC y es muy similar, pero describe una forma estándar de comunicación.

Varios lenguajes de programación tienen soporte nativo para SOAP, normalmente necesita de una URL de servicio web y puede llamar a sus funciones de web service sin la necesidad de un código específico.

**REST**

REST permite muchos formatos de datos diferentes, mientras que SOAP solo permite XML. Si bien esto puede parecer que agrega complejidad a REST porque necesita manejar múltiples formatos, en mi experiencia, ha sido bastante beneficioso. JSON por lo general es un mejor ajuste para los datos y analiza mucho más rápido. REST permite un mejor soporte para los clientes del navegador debido a su soporte para JSON.

REST tiene mejor rendimiento y escalabilidad. Las lecturas REST se pueden almacenar en caché; Las lecturas basadas en SOAP no se pueden almacenar en caché.

No se requiere de ninguna herramienta costosa para interactuar con el servicio web. Incluso en lenguajes de programación antiguos se puede hacer la comunicación a través de un HTTP.

Eficiente (SOAP usa XML para todos los mensajes, REST puede usar formatos de mensajes más pequeños como son texto plano, HTML, XML, JSON etc.).

Rápido (no requiere procesamiento extenso).