NOMBRE: \_\_\_Jesús David Cifuentes Rodríguez\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Caso de negocio**:

La empresa de producción de prendas de vestir “**New experience**” está desarrollando una solución para la ampliación de sus canales de ventas, los clientes podrán comprar los productos mediante una plataforma virtual (Tienda virtual), en el que seleccionarán las diferentes prendas de temporada, agregarlas a su compra, navegar las diferentes secciones, registrarse, detallar sus datos para él envió de sus productos a domicilio (es posible que sea diferente para cada compra), tener una vista previa de los productos seleccionados y la posibilidad de pagar en línea. Es necesario registrarse para probar algunas funcionalidades, como por ejemplo completar los flujos de la compra.

Acuden a usted para realizar las pruebas ya que el responsable de este no está disponible. Asimismo, le entregan los diferentes criterios que se tienen que cumplir:

Ruta de la aplicación: <http://automationpractice.com/index.php>

1. **Criterios tácticos:**
2. 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.
3. Considere la creación de casos de prueba. (Considerar pasos si es necesario)
4. De estos casos, clasifique que casos son automatizables y no automatizables.
5. Indique si existen y cuáles son las precondiciones para ejecutar los casos de prueba.
6. Presente estadísticas de prueba que permitan describir la conclusión de un indicador que defina.
7. Sugiera mejoras al módulo probado.

***SOLUCION CREADA EN EL EXCEL COMPARTIDO CON EL CORREO ELECTRONICO.***

1. **Criterios técnicos:**
2. Realice la automatización (programación) de los casos de prueba que seleccionó como automatizables. Se requiere implementar el uso de las siguientes Herramientas:
   * + Lenguaje: Java
     + Tipo de proyecto: Deseable Gradle
     + Framework de ejecución: Junit
     + Framework de pruebas: Serenity/Selenium
     + Driver o navegador: Chrome
     + Arquetipo: Page Object Model o Screenplay
     + Escribir HU en: Cucumber
     + Ejecución: Junit o Jenkins Local
3. Realice la ejecución de lo automatizado, con sus respectivas evidencias.
4. Colocar el código automatizado en **github** y enviarlo. (Repositorio público)
5. **Agilismo**
6. De las sugerencias de mejora propuestas, defina historias de usuario que plantearía para un nuevo desarrollo, con sus respectivos criterios de aceptación.
7. 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.

***SOLUCION CREADA EN EL EXCEL COMPARTIDO CON EL CORREO ELECTRONICO.***

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

¡Este es un reto! Por ende, puede consultar toda la información que sea necesaria u otra opción es preguntar las dudas que tenga a personas conocidas.

**Solución**

1. **4 Conceptos Programación / CI / CD**
2. **Principios de SOLID**: Es un acrónimo para los cinco principios básicos de la programación el cual el propósito es eliminar malos diseños, evitar la refactorización y construir un código más eficiente.

* **S** – single Responsability Principle: Principio en cual define que un componente debe tener una responsabilidad única, simplificando el código.
* **O**- Open: Establece que los componentes del software tienen que estar abiertos para extender a partir de ellos.
* **I**-Interface segretation principle: Este principio establece que los clientes no deben ser forzados a depender a un interfaz que no utilizan
* **D**-Dependency inversión principle: Establece que las secciones o módulos de alto nivel no deben depender de las abstracciones.

1. **Patrón Singleton:** El propósito de patrón singleton es evitar que sea creado más de un objetivo por clase , y eso lo podemos lograr creando objetivos en solo una clase proporcionando un solo punto de acceso universal a ella , también podemos concluir que no se encarga de la creación de objetivos en sí , sino que llega a centrarse en la restricción en la creación de un objetivo
2. **Patrón FIRST:** Básicamente trata de aumentar la calidad de nuestros tests unitarios, donde FIRST es el acrónimo de las cinco características que deben tener los tests unitarios.

* **Fast:** Como su nombre lo dice esto posibilita ejecutar los tests muy frecuente y con ello detectar bugs de forma muy rápida y práctica.
* **Independent:** Generaliza que todas las pruebas que tengamos que hacer deben ser independientes de las otras, ya que en el momento que un test falla por el orden en el que se ha ejecutado, tenemos claro que ese test está mal desarrollado.
* **REPEATABLE:** Como su nombre lo dice puede ser repetible ya que el resultado de las pruebas debe ser el mismo independientemente del servidor en el que se ejecute.
* **SELF-VALIDATING:** Se puede ejecutar las pruebas simplemente al pulsar un botón y que se ejecuten de forma automática tras otro proceso.
* **TIMELY:** Desarrollar las pruebas posteriormente al desarrollo del producto , al ser posible las pruebas deben desarrollarse antes de empezar a desarrollar el código de la aplicación TDD (TEST DRIVEN DEVELOPMENT) pruebas para un método mientras evalúan ya que pueden ser falsas ayuda a tener una calidad de código. BDD (BEHAVIOR DRIVEN DEVELOPMENT) desarrollo guiado por comportamiento más que todo se busca el comportamiento de la aplicación o el propósito.

1. **Patron AAA : El** patrón sugiere dividir una prueba unitaria en tres secciones las cuales cada una de las secciones tiene un propósito

* **Arrange: (organizar/inicializa)** inicializa los objetivos y estable datos que vamos a utilizar en el test.
* **Act:(Actuar)** Realiza la llamada de método a probar con parámetros preparados con esa finalidad .
* **Assert: (Confirmar/comprobar)** Prueba que el método de pruebas ejecutado se comporta óptimamente.

1. **Pull Request** : Es una petición que el dueño de un fork de un repositorio hace el propietario de repositorio original , básicamente permiten llevar de forma ordenada las tareas en la etapa del desarrollo, sino también crear propuestas o cambios que puedan ser incluidos al proyecto.
2. **Release Train :**Es un método que nos sirve para planificar la entrega de un software según algún tipo de calendario determinado.
3. **Quality Gates:** Es una herramienta que permite monitorizar la calidad de los datos de una forma centralizada esto puede ayudar que las empresas mejoren de manera constante y eficiente la calidad de la información.
4. **Diferencias servicios SOAP / REST:**

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| Es un protocolo diseñado originalmente para permitir la comunicación entre las aplicaciones que se diseñan con diferentes lenguajes y aplicativos. | Se ajusta según a las necesidades de los servicios web |
| Estandariza la forma de proteger y transferir los mensajes usando identificadores únicos llamados tokens | Una arquitectura cliente-servidor compuesta por clientes, servidores y recursos. |