**Plan de Pruebas para Pruebas Automatizadas**

**1. Alcance y Objetivos**

* **Alcance:** Automatizar las pruebas funcionales de la tienda virtual Demoblaze (<https://www.demoblaze.com/>) para los siguientes casos de uso:
  + Compra de un teléfono por un usuario no registrado.
  + Compra de una laptop por un usuario no registrado.
  + Compra de un teléfono y una laptop por un usuario no registrado.
  + Creación exitosa de una cuenta de usuario no registrado.
  + Intento fallido de crear una cuenta de usuario no registrado con un nombre de usuario existente.
  + Inicio de sesión exitoso de un usuario registrado.
  + Intento fallido de inicio de sesión de un usuario registrado con contraseña inválida.
  + Compra de un monitor por un usuario registrado que inicia sesión exitosamente.
* **Objetivos:**
  + Verificar la funcionalidad del sitio web Demoblaze.
  + Automatizar casos de prueba repetitivos para reducir el tiempo de ejecución de las pruebas.
  + Mejorar la cobertura de las pruebas.
  + Aumentar la confiabilidad del proceso de prueba.
  + Identificar defectos tempranamente en el ciclo de vida del desarrollo de software.

**2. Estrategia de Automatización**

* **Herramientas:**
  + Selenium WebDriver: Para la interacción con el navegador.
  + Java: Lenguaje de programación para escribir los scripts de prueba.
  + Maven: Herramienta de gestión de proyectos para gestionar dependencias y ejecutar pruebas.
  + JUnit: Framework de pruebas para escribir y ejecutar pruebas unitarias.
  + IntelliJ IDEA: Entorno de desarrollo integrado (IDE) para escribir y gestionar el código.
* **Framework:**
  + Se utilizará un framework de pruebas basado en JUnit.
  + Se implementará el patrón Page Object para mantener el código de prueba y facilitar el mantenimiento.
* **Tipos de Pruebas a Automatizar:**
  + Pruebas funcionales: Para verificar que la aplicación funciona según los requisitos.
* **Casos de Prueba para Automatizar:**
  + Todos los casos de uso mencionados en la sección de Alcance.

**3. Entorno de Pruebas**

* **Hardware:** Una computadora con conexión a Internet.
* **Software:**
  + Sistema Operativo: Windows 11.
  + Navegador: Google Chrome.
  + JDK (Java Development Kit) 11 o superior.
  + Apache Maven 3.6.3 o superior.
  + IntelliJ IDEA.
  + ChromeDriver (compatible con la versión de Chrome).
* **Configuración:**
  + El navegador se iniciará en modo normal.
  + Se utilizarán datos de prueba estáticos.

**4. Datos de Prueba**

* **Creación:** Los datos de prueba se crearán manualmente.
* **Gestión:** Los datos de prueba se gestionarán dentro de las clases de prueba.
* **Tipos de Datos:**
  + Nombre de usuario y contraseña para el registro e inicio de sesión.
  + Nombres de productos (teléfono, laptop, monitor) para las pruebas de compra.

**5. Diseño y Desarrollo de Pruebas**

* **Proceso:**
  1. Crear un proyecto Maven en IntelliJ IDEA.
  2. Añadir las dependencias de Selenium WebDriver y JUnit al archivo pom.xml.
  3. Crear clases de Page Object para cada página web con la que interactuarán las pruebas.
  4. Crear clases de prueba JUnit para cada caso de uso.
  5. Escribir scripts de prueba utilizando los métodos proporcionados por Selenium WebDriver y las clases de Page Object.
  6. Ejecutar las pruebas y analizar los resultados.
* **Estándares de Codificación:**
  1. Se seguirán las convenciones de nomenclatura de Java.
  2. El código estará bien comentado.
  3. Se utilizará un manejo de excepciones adecuado.
  4. Se evitará la duplicación de código mediante la creación de métodos reutilizables.

**6. Ejecución de Pruebas y Gestión de Defectos**

* **Ejecución:**
  + Las pruebas se ejecutarán utilizando el comando mvn test de Maven.
  + Las pruebas se ejecutarán después de cada cambio en el código.
* **Gestión de Resultados:**
  + Los resultados de las pruebas se mostrarán en la consola.
  + Los resultados de las pruebas también se pueden ver en un informe generado por JUnit.
* **Gestión de Defectos:**
  + Cualquier fallo de prueba se considerará un defecto.
  + Los defectos se registrarán en un sistema de seguimiento de defectos (no especificado en este plan).
  + Los defectos se asignarán a los desarrolladores para su resolución.
  + Una vez que se corrija un defecto, se volverá a probar para verificar la corrección.

**7. Métricas e Informes**

* **Métricas:**
  + Número de casos de prueba automatizados.
  + Número de casos de prueba ejecutados.
  + Número de casos de prueba aprobados.
  + Número de casos de prueba fallidos.
  + Porcentaje de casos de prueba aprobados.
  + Tiempo de ejecución de las pruebas.
* **Informes:**
  + Se generará un informe de prueba después de cada ejecución de prueba.
  + El informe contendrá un resumen de los resultados de las pruebas, incluidos el número de pruebas ejecutadas, aprobadas y fallidas.
  + El informe también contendrá detalles de cualquier prueba fallida, incluido el mensaje de error y la captura de pantalla (si corresponde).

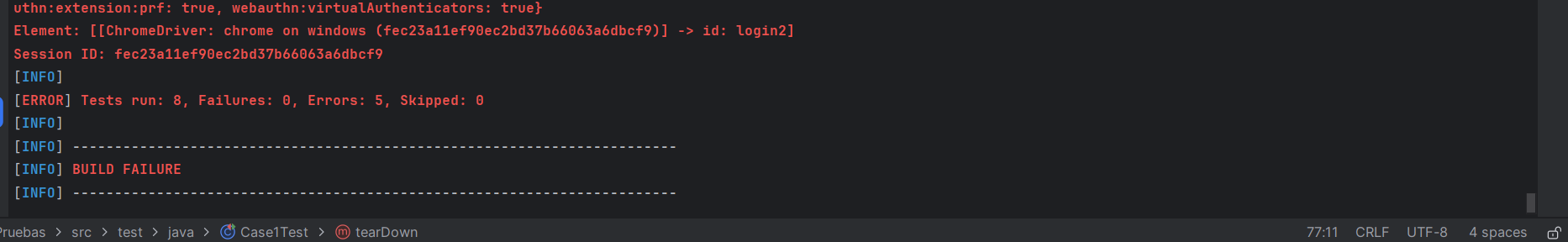
**8. Recursos y Cronograma**

* **Recursos:**
  + 1 Analista de Pruebas de Automatización.
  + 1 Computadora con el software requerido.
* **Cronograma:**
  + Día 1: Configuración del entorno de prueba.
  + Día 2-3: Desarrollo de scripts de prueba.
  + Día 4: Ejecución de pruebas y análisis de resultados.
  + Día 5: Informe de resultados de las pruebas.

**9. Riesgos y Contingencias**

* **Riesgos:**
  + Problemas con la configuración del entorno de prueba.
  + Dificultad para identificar los localizadores de los elementos web.
  + Cambios en la aplicación bajo prueba.
  + Limitaciones de las herramientas.
* **Contingencias:**
  + Proporcionar tiempo adicional para la configuración del entorno de prueba.
  + Utilizar diferentes estrategias de localización de elementos web.
  + Mantener los scripts de prueba para adaptarlos a los cambios de la aplicación.
  + Investigar herramientas alternativas si es necesario.

Anexos:



Texto

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.