Soporte y Seguridad de Apps

OBSERVACIÓN 4

OBJETIVO:

Aplicar herramientas básicas utilizadas en el proceso de aseguramiento de la calidad de software, desde la planificación y ejecución de pruebas hasta la documentación y gestión de defectos.

Sitio web para pruebas: Geek Store - Your buggy e-commerce

INSTRUCCIONES:

1. Configuración del entorno de testing, usando las siguientes herramientas:

- **1.1** TestRail: para la gestión de casos de prueba.
- **1.2** Test Case Studio: extensión para generar casos automáticamente desde el navegador.
- **1.3** JIRA: para la gestión del flujo de trabajo SCRUM.
- **1.4** Chrome DevTools / Lighthouse: para métricas de rendimiento y accesibilidad.
- **1.5** BrowserStack: para compatibilidad en distintos navegadores y dispositivos.

2. Crear un tablero SCRUM en JIRA y definir las columnas del flujo de trabajo:

Backlog → To Do → In Progress → Testing → Done

Agregar entre 3 y 5 User Stories del sitio web. Por ejemplo:
"Como usuario, quiero agregar productos al carrito para realizar compras fácilmente."
"Como visitante, quiero registrarme y hacer login para guardar mis datos."
"Como usuario, quiero buscar productos por nombre o categoría."

3. Configurar repositorio en GitHub, llamado Observacion-4, de manera que

- **3.1** Centralizar la evidencia y documentación del proceso de pruebas (capturas, reportes, archivos .csv o .pdf de resultados).
- **3.2** Subir los casos de prueba y reportes de bugs.
- **3.3** Versionar los archivos (guardar cambios con historial y comentarios).
- **3.4** Permitir colaboración en equipo y seguimiento de quién aportó qué.

4. Creación de casos de prueba *(TestRail)* con al menos 5 casos de prueba, cubriendo las siguientes áreas:

ID	Módulo / Funcionalidad	Descripción	Tipo de prueba
TC01	Registro y Login	Validar que los campos de registro/login funcionen correctamente.	Funcional
TC02	Búsqueda de productos	Verificar que la búsqueda retorne resultados esperados.	Exploratoria
TC03	Validaciones	Probar campos requeridos, formatos y mensajes de error.	Validación

5. Generar casos de prueba automáticamente mientras se navega por el sitio, usando la extensión de Test Case Studio. Debe crear un Test Run para ejecutar los casos diseñados. Marcar los resultados según su estado *(Passed, Failed, Blocked, Retest)*.

6. Documentar los errores detectados con la siguiente estructura:

Campo	Descripción
Título	Nombre corto del bug
Descripción	Qué ocurre y por qué es un error
Pasos para reproducir	Paso a paso que lleva al bug
Resultado esperado	Qué debería pasar
Resultado obtenido	Qué pasó realmente
Severidad	Baja / Media / Alta / Crítica
Evidencia	Captura, video o enlace al reporte

7. Usar Chrome DevTools – Lighthouse para generar un reporte de métricas que evalúe el rendimiento, accesibilidad y buenas prácticas usando:

7.1 Network: medir tiempos de carga y tamaño de recursos.

7.2 Elements: identificar errores visuales o de CSS.

7.3 Recorder: grabar un flujo completo del usuario como evidencia.

8. Usar BrowserStack para comprobar que el sitio web funcione correctamente en distintos navegadores y dispositivos.

8.1 Navegadores: Chrome, Edge, Firefox.

8.2 Dispositivos: Laptop, tablet, smartphone.

9. Cada grupo deberá entregar un reporte consolidado en PDF o documento, que incluya:

9.1. Portada del proyecto.

9.2. Descripción breve del sitio probado.

9.3. Capturas del tablero SCRUM y GitHub.

9.4. Casos de prueba (de TestRail y Test Case Studio).

9.5. Evidencias de ejecución y bugs encontrados.

9.6. Reporte Lighthouse con interpretación de métricas.

9.7. Resultados del cross-browser testing.

9.8. Conclusiones y recomendaciones.

GLOSARIO:

Categoría	Definición / Descripción	Ejemplo
User Story	Requisito expresado desde la perspectiva del usuario.	“Como cliente, quiero agregar productos al carrito para comprarlos más tarde.”
Prueba funcional	Verifica que el sistema cumpla los requerimientos definidos.	El botón “Agregar al carrito” agrega correctamente el producto.
Prueba exploratoria	El tester explora libremente el sistema para descubrir errores no documentados.	Probar enlaces o botones sin caso previo.
Prueba de validación	Confirma que los resultados coincidan con lo esperado.	Validar que el total del carrito sea correcto.
Passed	Caso exitoso, sin errores.	El login funciona correctamente.
Failed	Caso con error o comportamiento incorrecto.	El botón “Pagar” no responde.
Blocked	Caso no ejecutado por dependencia o bug previo.	No se puede continuar por fallo en registro.
Retest	Caso reejecutado tras corrección.	Reprobar formulario luego de fix.
Performance	Velocidad y eficiencia de carga del sitio.	First Contentful Paint: 1.8 s
Accessibility	Facilidad de uso para personas con discapacidades.	Contraste correcto, etiquetas alt.
Best Practices	Cumplimiento de estándares modernos y seguridad.	Sin errores JS, HTTPS habilitado.
SEO	Optimización técnica para motores de búsqueda.	Uso correcto de metadatos y encabezados.
Tiempo de carga (Load Time)	Duración total desde que se solicita la página hasta que se muestra.	2.4 segundos.
Tamaño de recursos (Resource Size)	Peso total de imágenes, scripts y estilos descargados.	3.2 MB.
Errores visuales CSS	Desalineaciones, superposiciones, contrastes incorrectos.	Texto fuera de cuadro en versión móvil.
Interaction to Next Paint	Tiempo que tarda el sitio en reaccionar a la siguiente	180 ms (óptimo < 200 ms).

(INP)	acción del usuario.	

ENTREGABLE:

1. Documentación