ANÁLISIS Y DESARROLLO DE SOFTWARE

ENTREGABLES DEL PROYECTO BELLA Y VALE

Ficha: 2627092

Creación de informe con los resultados del comportamiento del software. GA11-220501098-AA2- EV01.

Proyecto de Software

Aprendices:

EQUIPO DE TRABAJO

Carlos Téllez, Jonnathan Reyes, Miguel Guevara, Santiago Muentes y Yuly Toro

Profesor:

NICOLAS SALDARRIAGA

Servicio Nacional de Aprendizaje
SENA

Junio de 2024

Jamundí 12 de junio de 2024

Respetado profesor

Nicolas Saldarriaga
Instructor Vocero
Ficha: 2627092

Cordial saludo,

Asunto remisión: GA11-220501098-AA2- EV01.

De manera atenta me permito presentar el entregable de la evidencia Creación de informe con los resultados del comportamiento del software.

GA11-220501098-AA2- EV01.

Atentamente,

Jonnathan Reyes Mosquera
CC.1.061.532.705
Cel. 3146140320
Email: jonnathanreys@gmail.com

ACTIVIDAD

Crear el documento de informe de resultados del comportamiento del software.

Elementos a tener en cuenta:

● Leer y analizar el componente formativo “Aplicación de pruebas de software”.

● Realizar una bitácora con los procesos documentales.

● Seleccionar las buenas prácticas de calidad de acuerdo con el referente de los marcos de trabajo.

● Seguir las normas básicas de presentación de un documento escrito, es decir el documento debe tener como mínimo una portada, introducción y conclusiones.

● Registrar en el documento un resumen con los recursos utilizados para la evaluación:

- Equipo evaluador.

- Métricas utilizadas.

- Ponderación.

- Fidelidades de medición.

- Criterios de aprobación.

- Recursos de infraestructura.

- Tipos de pruebas y pruebas realizadas.

DESARROLLO DE LA ACTIVIDAD

1. Introducción

Para todo sistema de información, después de la etapa de desarrollo se continúa con la implementación del sistema, y allí se requiere de un ejercicio importante de ingeniería de software. Se debe de realizar antes de hacer la entrega final al cliente, y es comprobar que el sistema cumple con los requerimientos del usuario y que su funcionamiento es correcto, es decir sin errores o defectos. Para esto se deben implementar las **pruebas de software.**

Podemos concretar que cuando se desarrollan aplicaciones web, uno de los desafíos es verificar que la calidad del producto sea óptima y sin errores, de este modo las pruebas de software son parte fundamental del proceso de calidad y adopción de buenas prácticas.

1.1. Definiciones de calidad del software

La Calidad del Software según el estándar IEEE 6.10-1990, es *“el grado con el que un sistema, componente o proceso cumple con los requisitos especificados y las*

necesidades o expectativas del cliente o usuario". Otros como Bolaños, Sierra y Alarcón en 2008 la definen como un *"proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes los producen y a quienes lo utilizan".* Ahora Según Cataldi, 2000, la calidad está asociada a tres usos importantes del usuario:

- Características de operación.
- Capacidad para soportar cambios (ser modificado).
- Adaptabilidad a nuevos cambios.

1.2. Dimensión de la calidad.

David Garvin, plantea ocho dimensiones la calidad que pueden ser aplicadas al software, que se describen en la siguiente tabla:

Dimensión	Descripción
Calidad del desempeño.	Presenta el contenido, las funciones y las características especificadas en el modelo de requerimientos.
Calidad de las características.	Genera sorpresa y agrado en la primera impresión del usuario.
Confiabilidad.	Está disponible cuando se necesita, sin errores y sin fallas.
Conformidad.	Es coherente con los estándares locales e internacionales
Durabilidad.	Permite con facilidad el mantenimiento (cambio) y la depuración (corrección).
Servicio.	El mantenimiento y la depuración se pueden hacer en un tiempo aceptablemente breve.
Estética.	Posee cierta elegancia, flujo único y presencia aceptable por los usuarios en general.
Percepción.	Recibe en general buenos comentarios por parte de los usuarios.

1.3. Factores de la calidad.

A partir de las definiciones del estándar ISO 9126, presenta los siguientes factores o atributos claves asociados a la calidad del software que son:

a) *En cuanto a **revisión** del producto*: Facilidad de recibir mantenimiento y Flexibilidad
b) *En cuanto a **transición** del producto*: Funcionalidad, Portabilidad y Reusabilidad
c) *En cuanto a **operación** del producto*: Confiabilidad, Usabilidad, Eficiencia, Corrección e Integridad

Las actividades más importantes en la **gestión de los procesos** de acuerdo con la norma **ISO 9000 y 9001:2015** son:

- *Funcionalidad*: satisfacción a las necesidades de adaptabilidad, exactitud, interoperabilidad, cumplimiento y seguridad.
- *Confiabilidad*: cantidad de tiempo que el software se encuentra disponible para su uso, con madurez, tolerancia a fallas y recuperación.
- *Usabilidad*: facilidad para usar, siendo entendible, aprendible y operable.
- *Eficiencia*: uso óptimo de los recursos del sistema.
- *Identificar clientes y sus necesidades*: en esta actividad se determinar el objetivo fundamental de la organización y su razón de ser.
- *Facilidad de recibir Mantenimiento*: facilidad para realizar reparaciones, es decir, es analizable, cambiable, estable y susceptible a pruebas.
- *Portabilidad*: facilidad para ser llevado a otro ambiente, es decir, es adaptable, instalable y sustituible.
- *Corrección*: cumple con las especificaciones y necesidades del cliente.
- *Integridad:* control de acceso al software o datos de usuarios no autorizados.
- *Flexibilidad*: capacidad para permitir modificaciones cuando el software ya está en operación.
- *Reusabilidad*: grado en el que puede ser usado por otras aplicaciones.

En particular, para evaluar una **interfaz**, se debe tener en cuenta los siguientes factores de calidad:

- *Intuitiva*: la interfaz sigue patrones de uso esperados, facilitando la comprensión, localización de operaciones y la entrada de datos.
- *Eficiencia*: grado en el que es posible localizar o iniciar las operaciones y la información.
- *Robustez*: capacidad para tratar entradas erróneas de datos o interacción inapropiada del usuario.
- *Riqueza*: interfaz con abundantes características, que permite la personalización según las necesidades del usuario, y la identificación de una secuencia de operaciones comunes por medio de una acción o comando.

1.4. Verificación y validación (V&V).

En el control de calidad de software se distinguen dos procesos de evaluación propios del proceso de desarrollo de software: **la verificación y la validación**. IEEE estándar 729-1983 da las siguientes definiciones:

Verificación: “Proceso para determinar si los productos de una determinada fase del desarrollo de software cumplen o no los requisitos establecidos durante la fase anterior”.

Validación: “Proceso de evaluación del software al final del proceso de desarrollo para asegurar el cumplimiento de las necesidades del cliente”.

Nos podemos preguntar: ¿Se ha construido el sistema correctamente?

La validación y verificación se puede realizar usando básicamente dos tipos de técnicas: Dinámicas y Estáticas.

2. Informe del comportamiento del software

2.1. Bitácora del Proyecto de Software Bella y Vale

Fecha	Tarea-descripción	Tiempo invertido	Resultados
31/05/2024	Planificación de la Evaluación	2	En esta tarea se definió el alcance, los objetivos y los criterios de evaluación del proceso. Se identificaron las métricas y los indicadores que se utilizarán para medir el rendimiento y la calidad del software.
31/05/2024	Preparación de Recursos	2	Se asignaron los recursos necesarios para llevar a cabo la evaluación, incluyendo herramientas de evaluación, equipos de prueba, datos de prueba con la intervención de todos los miembros del equipo.
01/06/2024	Recolección de Datos	2	Se recopilaron datos relevantes sobre el software Bella y Vale como su funcionalidad, rendimiento, usabilidad, fiabilidad y seguridad.
01/06/2024	Pruebas Funcionales	2	Se realizaron algunas pruebas para verificar que el software cumple con los requisitos funcionales establecidos, probando sus características y funciones en diferentes escenarios.
01/06/2024	Pruebas de Rendimiento	2	Se evaluó el rendimiento del software en términos de velocidad, eficiencia y capacidad de respuesta bajo diferentes cargas y condiciones de uso.
02/06/2024	Pruebas de Usabilidad	2	Se evaluó la facilidad de uso y la experiencia del usuario del software, identificando posibles problemas de usabilidad y proponiendo mejoras.
02/06/2024	Pruebas de Seguridad	2	Se realizaron pruebas para identificar posibles vulnerabilidades de seguridad en el software y evaluar su capacidad para proteger los

			datos y prevenir intrusiones no autorizadas.
02/06/2024	Análisis de Resultados	2	Se analizaron los datos recopilados durante las pruebas para identificar áreas de mejora, puntos fuertes y debilidades del software, y tomar decisiones informadas sobre su calidad y rendimiento.
03/06/2024	Generación de Informes	2	Se elaboraron informes detallados que documentan los resultados de la evaluación, incluyendo hallazgos, conclusiones, recomendaciones y acciones correctivas sugeridas.
03/06/2024	Comunicación de Resultados	2	Se comparten los resultados de la evaluación con las partes interesadas relevantes, como el equipo de desarrollo, algunos usuarios finales, y al profesor de ADSO-SENA, para informar sobre el estado y la calidad del software.
03/06/2024	Implementación de Mejoras	2	Se implementaron las mejoras y correcciones necesarias basadas en los hallazgos y recomendaciones de la evaluación, con el objetivo de mejorar la calidad y el rendimiento del software.
04/06/2024	Seguimiento y Retroalimentación	2	Debemos realizar con el tiempo un seguimiento continuo del software después de la evaluación para monitorear su desempeño y recopilar retroalimentación adicional de los usuarios, con el fin de realizar ajustes y mejoras adicionales según sea necesario.
Comentario: Estas actividades y tareas forman parte del proceso de evaluación del producto de software y contribuyen a garantizar que el software cumpla con los estándares de calidad y rendimiento requeridos para satisfacer las necesidades y expectativas de los usuarios finales.			

2.2. Buenas prácticas de calidad

1. Utilizar herramientas versátiles como Desarrollador Scrum

Cuando se trata de software de colaboración de calidad, las hojas de Excel no son suficientes. En su lugar, use herramientas versátiles que estén diseñadas

específicamente para Scrum. Por ejemplo, utilizando JIRA para la gestión de proyectos, Toggl para el seguimiento del tiempo o Slack para una mejor colaboración en equipo.

2. Organizar stand-ups siendo un Desarrollador Scrum

Una vez que tenga las fechas de inicio y finalización del sprint, y un compromiso del equipo de sprint para la entrega, comience su sprint. Cuando trabaje en Scrum, siempre complete sus tareas en sprint para asegurarse de que se pueda enviar. Durante este período, programe una llamada diaria con su equipo para abordar cualquier problema que pueda surgir y mantenerse al día con el cronograma de la tarea.

3. Revisar reuniones - Su participación como Desarrollador es importante

Si bien las reuniones se llevan a cabo de manera regular cuando su proyecto aún está en progreso, se lleva a cabo una reunión de revisión después de que se completa el proyecto. La reunión se centrará en las historias de usuarios que cumplieron con la satisfacción del cliente, mientras que las historias de usuarios incompletas permanecerán como trabajos pendientes de productos.

4. Experimentar con procesos

Scrum es uno de los marcos más flexibles disponibles en la actualidad y está diseñado teniendo en cuenta la flexibilidad del usuario. Por lo tanto, una práctica que es mejor para una empresa, puede no ser tan buena para otra.

Pídale a su equipo que experimente con el proceso y pruebe cosas nuevas. Si ha estado usando sprints de dos semanas hasta ahora, pídale a su equipo que realice un sprint de una o tres semanas y observe los resultados.

5. Colaboración entre pares

En resumen, un Scrum Master es un profesional encargado de aplicar metodologías ágiles en los equipos de trabajo Scrum convirtiéndose en un maestro y mentor para todos. En proyectos que necesiten la intervención de muchas personas y procesos complejos, se convierte en una pieza indispensable para la organización del equipo y agilizar el trabajo de cada integrante.

2.3. Resumen con los recursos utilizados para la evaluación

a) Equipo evaluador.

El equipo evaluador esta conformado por Carlos Téllez, Jonnathan Reyes, Santiago Muentes, Miguel Guevara y Yuly Toro

b) Métricas utilizadas.

Las métricas de calidad del software son criterios y medidas utilizadas para evaluar qué tan bien se desarrolla una solución de software por un equipo. Son una brújula que nos ayuda a entender la eficiencia, confiabilidad y rendimiento del software, asegurando que cumpla con los estándares y requisitos establecidos y estas son:

1-Rotación total de códigos: Es la cantidad de líneas de código agregadas, modificadas o eliminadas de una base de código durante un período de tiempo.

2-Tasa de fallas: es la frecuencia con la que falla un producto de software.

3-Disponibilidad del sistema: evalúa la cantidad de tiempo que un sistema está en funcionamiento. Compara el tiempo de actividad del sistema con su tiempo de inactividad durante un período de tiempo. Y considera que un sistema que está disponible es tan rápido como se espera y no falla a menos que ocurran condiciones imprevistas.

4-Densidad de defectos: es la cantidad de defectos en un producto de software en comparación con su tamaño. Esto significa que es un número relativo.

5-Tiempo medio de detección (MTTD): Es el tiempo promedio que le toma a tu equipo detectar errores en un producto de software.

6-Tiempo medio entre fallas (MTBF): es el tiempo promedio entre dos fallas del sistema. Se trata de errores que se encuentran tras el lanzamiento del producto y que se deben, por ejemplo, a un defecto no detectado.

7-Tiempo medio de resolución (MTTR): es el tiempo promedio que le toma a tu equipo resolver un error en un producto de software después de que alguien lo descubre.

8-Satisfacción del cliente (CSAT): es un número que representa la forma en que los clientes experimentan tu producto de software. Y se llega a ello recopilando y analizando los datos de las encuestas de satisfacción de tus clientes sobre el funcionamiento del software.

9-Tiempo medio para remediar una vulnerabilidad: es el tiempo promedio que le toma a tu equipo reparar las vulnerabilidades de ciberseguridad en tu producto de software.

10-Cobertura de código: evalúa la cantidad de código fuente de tu producto para el cual existe una prueba unitaria.

c) Ponderación.

La ponderación es una técnica estadística que se puede utilizar para corregir cualquier desequilibrio en los perfiles de muestra después de la recolección de datos.

Una ponderación estadística es una cantidad que se da para aumentar o disminuir la importancia de un elemento, ya que muchas veces los datos recopilados de las encuestas no son exactamente de una muestra representativa de la población. Ej:

Si entrevistamos a una muestra de 10 personas dentro de esta población, de las cuales 7 son hombres y 3 son mujeres, entonces sabríamos que nuestra muestra sobre-representa a los hombres.

Ponderar los datos resultantes puede ayudarnos a corregir este desequilibrio. Las proporciones objetivo para hombres y mujeres son del 50%. Por lo tanto, la proporción de hombres tendría que disminuir del 75 % al 50 %, mientras que la proporción de mujeres necesita aumentar del 25 % al 50 %. Quedarían 5 y 5.

d) Fidelidades de medición.

La fidelidad de medición se refiere a la consistencia o confiabilidad de las medidas obtenidas a través de un instrumento de medición o en este caso, a través de un software. En el contexto de la evaluación de software, las fidelidades de medición incluyen varios aspectos: 1. Consistencia de resultados 2. Fidelidad de contenido 3. Fidelidad de interoperabilidad 4. Fidelidad de usabilidad 5. Fidelidad de modularidad 6. Fidelidad de retrocompatibilidad 7. Fidelidad de Confiabilidad. 8. Fidelidad de Validez 9. Fidelidad de Exactitud. 10. Fidelidad de Repetibilidad

e) Criterios de aprobación.

Los criterios de aprobación del software pueden variar dependiendo del contexto y los requisitos específicos del proyecto, pero aquí tienes algunos criterios comunes que suelen considerarse al evaluar y aprobar un software:

Cumplimiento de requisitos funcionales: El software debe cumplir con todas las funcionalidades especificadas en los requisitos funcionales definidos durante la etapa de planificación del proyecto.

Cumplimiento de requisitos no funcionales: Además de las funcionalidades específicas, el software debe cumplir con los requisitos no funcionales, como rendimiento, seguridad, usabilidad, escalabilidad, etc.

Estabilidad y confiabilidad: El software debe ser estable y confiable en su funcionamiento. Debe ser capaz de ejecutarse sin fallos significativos durante períodos prolongados y bajo diversas condiciones.

Calidad del código: Se evalúa la calidad del código fuente del software en términos de legibilidad, mantenibilidad, modularidad y buenas prácticas de programación.

Interoperabilidad: El software debe poder integrarse y operar correctamente con otros sistemas y tecnologías con las que interactúa.

Documentación: Debe haber documentación adecuada que describa el funcionamiento del software, cómo instalarlo, cómo configurarlo y cómo mantenerlo. Esto incluye manuales de usuario, guías de instalación, documentación técnica, etc.

Pruebas y validación: El software debe haber sido sometido a pruebas exhaustivas para validar su funcionamiento y garantizar que cumpla con los requisitos establecidos. Esto incluye pruebas unitarias, de integración, de sistema, de aceptación del usuario, entre otras.

Seguridad: El software debe cumplir con los estándares de seguridad establecidos para proteger los datos y la privacidad de los usuarios, así como para prevenir vulnerabilidades y ataques cibernéticos.

Cumplimiento normativo: Dependiendo del tipo de software y del sector en el que se utilice, puede ser necesario cumplir con ciertos estándares y regulaciones específicos.

Aceptación del usuario final: Finalmente, el software debe ser aceptado por los usuarios finales, quienes deben estar satisfechos con su funcionamiento y usabilidad.

f) Recursos de infraestructura.

Al evaluar el software, es importante tener acceso a una variedad de recursos de infraestructura que te permitan probar y evaluar adecuadamente su rendimiento, funcionalidad y compatibilidad. Aquí hay algunos recursos de infraestructura que puedes utilizar durante la evaluación de un software:

Hardware adecuado: Se debe contar con hardware que cumpla con los requisitos mínimos del sistema especificados por el software. Esto puede incluir servidores, estaciones de trabajo, dispositivos de red, dispositivos de almacenamiento, entre otros.

Entornos de prueba: Configura entornos de prueba que reflejen fielmente los entornos de producción donde se implementará el software. Esto puede incluir entornos de desarrollo, pruebas y producción, así como entornos virtuales o en la nube si es relevante.

Sistemas operativos: Verifica la compatibilidad del software con diferentes sistemas operativos, como Windows, Linux, macOS, Android, iOS, etc. Puedes utilizar máquinas virtuales o dispositivos físicos para ejecutar diferentes sistemas operativos y probar la interoperabilidad del software.

Redes: Evalúa el rendimiento del software en diferentes condiciones de red, incluyendo diferentes velocidades de conexión, latencia, pérdida de paquetes y ancho de banda. Puedes utilizar herramientas de emulación de red para simular condiciones de red específicas y observar cómo responde el software.

Bases de datos: Si el software utiliza una base de datos, asegúrate de tener acceso a las bases de datos necesarias para probar su funcionamiento. Esto puede incluir bases de datos SQL, NoSQL u otras tecnologías de almacenamiento de datos.

Herramientas de monitoreo y análisis: Utiliza herramientas de monitoreo y análisis para medir el rendimiento del software en términos de uso de recursos (CPU, memoria, almacenamiento), tiempos de respuesta, errores, entre otros. Esto te ayudará a identificar cuellos de botella y áreas de mejora.

Herramientas de automatización de pruebas: Implementa herramientas de automatización de pruebas para realizar pruebas de regresión, pruebas de carga, pruebas de estrés, pruebas de seguridad, etc. Estas herramientas te permitirán ejecutar pruebas repetitivas de manera eficiente y obtener resultados consistentes.

Herramientas de virtualización y contenedores: Utiliza tecnologías de virtualización y contenedores, como Docker, Kubernetes, VMware, etc., para crear entornos de desarrollo y pruebas reproducibles y aislados. Esto te permitirá probar el software en diferentes configuraciones de manera segura y eficiente.

Al utilizar estos recursos de infraestructura durante la evaluación de un software, podrás obtener una visión más completa de su rendimiento, funcionalidad y compatibilidad en diferentes entornos y condiciones.

g) Tipos de pruebas y pruebas realizadas.

Cuando se evalúa un software, es crucial realizar una variedad de pruebas para garantizar su calidad y funcionalidad. Algunos tipos de pruebas comunes que se pueden realizar durante la evaluación de software son:

Pruebas unitarias: Se centran en probar unidades individuales de código, como funciones o métodos, para asegurarse de que funcionen correctamente de manera aislada. Se utilizan marcos de prueba como JUnit, NUnit o PHPUnit para automatizar estas pruebas.

Pruebas de integración: Verifican que los diferentes componentes del software funcionen correctamente juntos como un sistema completo. Se prueban las interfaces entre los módulos y se identifican posibles problemas de comunicación y compatibilidad.

Pruebas de sistema: Se realizan para evaluar el comportamiento del sistema en su conjunto, verificando que todas las funciones y características del software funcionen según lo esperado y cumplan con los requisitos establecidos.

Pruebas de aceptación del usuario (UAT): Se llevan a cabo por usuarios finales o representantes del cliente para validar que el software cumpla con sus necesidades y expectativas. Estas pruebas suelen incluir escenarios de uso real y casos de prueba específicos del usuario.

Pruebas de regresión: Se realizan para asegurarse de que los cambios o actualizaciones realizados en el software no hayan introducido nuevos errores y que las funciones existentes sigan funcionando como se espera. Se vuelven a ejecutar pruebas previamente realizadas para detectar posibles efectos secundarios no deseados.

Pruebas de carga: Evalúan el rendimiento del software bajo condiciones de carga pesada, simulando múltiples usuarios concurrentes o grandes volúmenes de datos. Estas pruebas ayudan a identificar cuellos de botella y evaluar la escalabilidad del sistema.

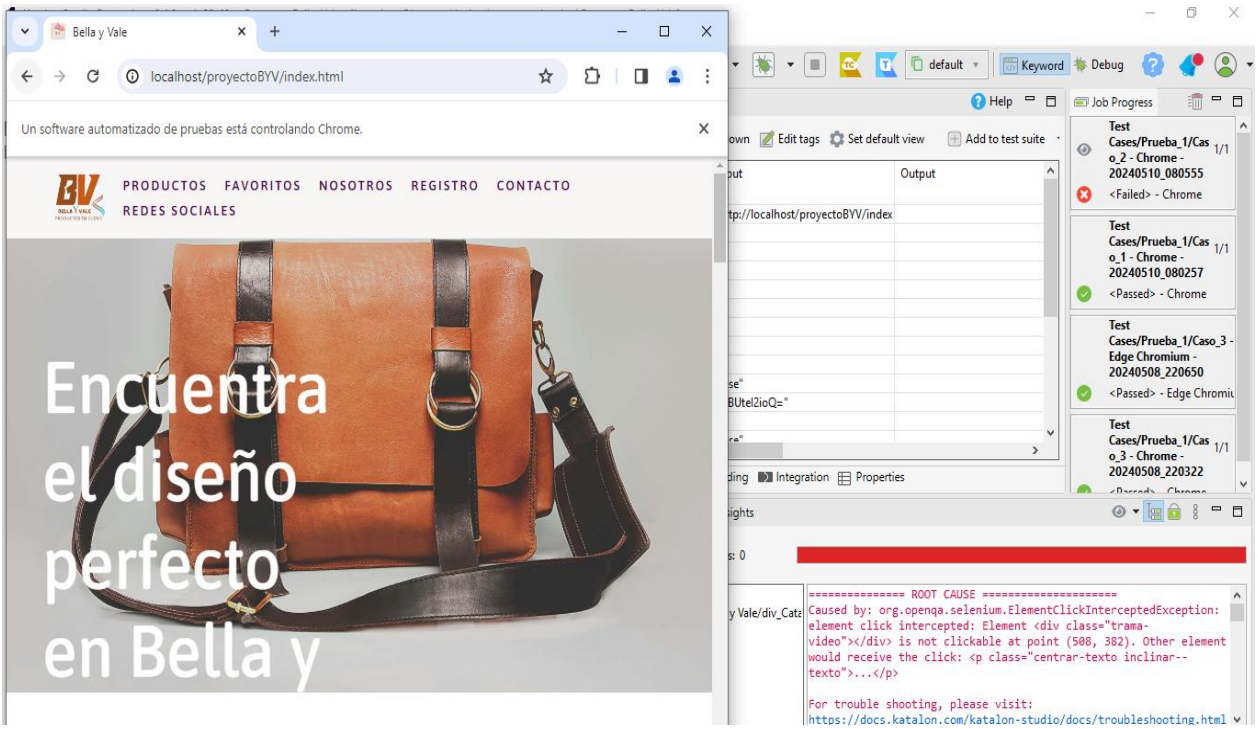
Pruebas de estrés: Buscan determinar los límites de capacidad y estabilidad del software sometiéndolo a condiciones extremas o inusuales, como picos de tráfico repentinos o recursos limitados.

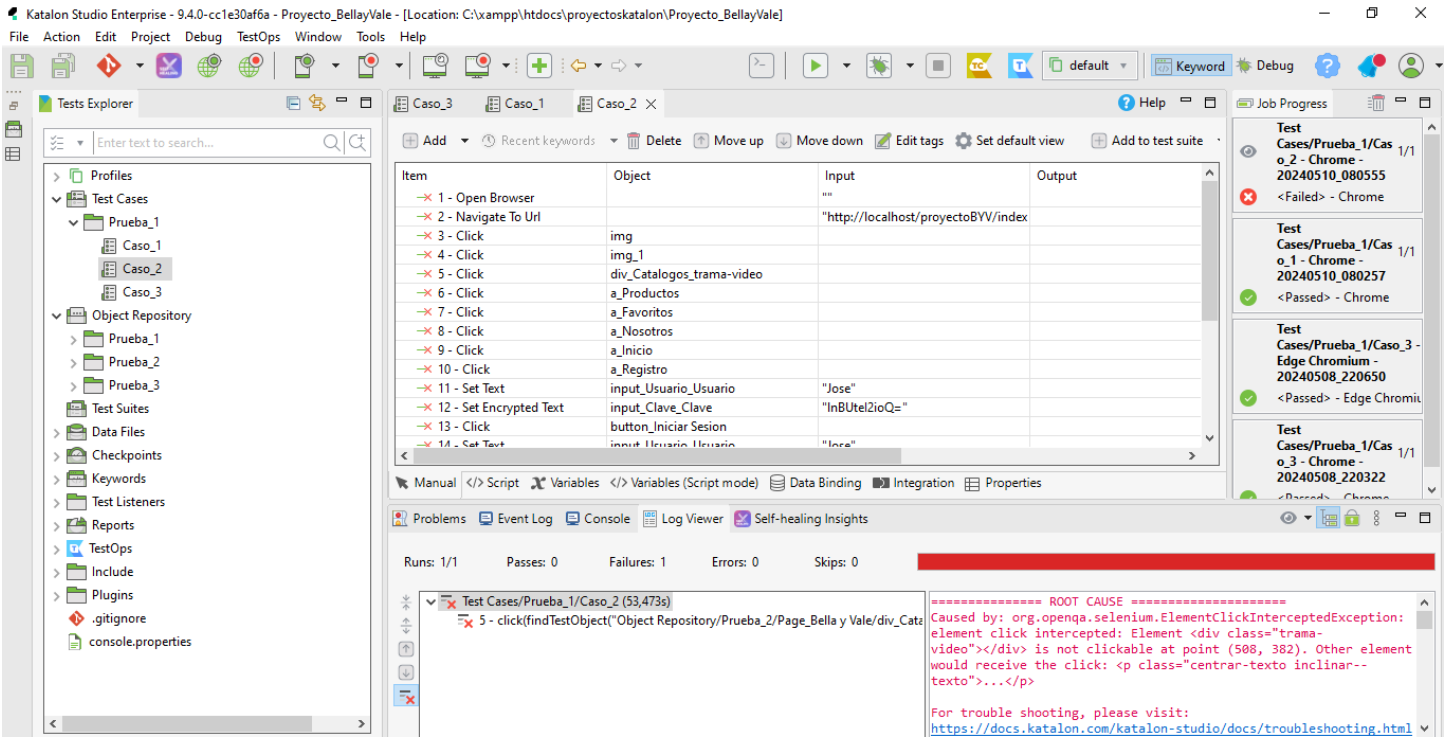
Pruebas de seguridad: Se enfocan en identificar vulnerabilidades de seguridad y posibles puntos de entrada para ataques maliciosos. Estas pruebas pueden incluir pruebas de penetración, análisis estático de código y escaneos de vulnerabilidades.

Pruebas de usabilidad: Evalúan la facilidad de uso y la experiencia del usuario del software, incluyendo aspectos como la navegación, la presentación de la información y la capacidad de realizar tareas de manera intuitiva.

Pruebas de compatibilidad: Verifican que el software funcione correctamente en diferentes plataformas, navegadores web, dispositivos y sistemas operativos, asegurando una experiencia consistente para todos los usuarios.

Item	Object	Input	Output
1 - Open Browser		""	
2 - Navigate To Url		"http://localhost/proyectoBYV/index	
3 - Click	img		
4 - Click	img_1		
5 - Click	div_Catalogos_trama-video		
6 - Click	a_Productos		
7 - Click	a_Favoritos		
8 - Click	a_Nosotros		
9 - Click	a_Inicio		
10 - Click	a_Registro		
11 - Set Text	input_Usuario_Usuario	"Jose"	
12 - Set Encrypted Text	input_Clave_Clave	"InBUtel2ioQ="	
13 - Click	button_Iniciar Sesion		
14 - Set Text	input_Usuario_Usuario	"Jose"	
15 - Set Encrypted Text	input_Clave_Clave	"4nvbrPglk7k="	
16 - Click	button_Iniciar Sesion		
17 - Click	i_Nueva coleccin REF Italia importad		
18 - Click	a_Cerrar Sesin		
19 - Click	a_SALIR		
20 - Click	a_Contacto		
21 - Click	div_To navigate, press the arrow keys		
22 - Click	a_Inicio		
23 - Click	a_redes sociales		





3. Conclusiones

Consideramos que la evaluación de software Bella y Vale realizada permitió determinar su calidad, usabilidad y eficacia. Es una etapa importante en el proceso de desarrollo porque determina que nuestro software si cumple las normas y especificaciones requeridas y es adecuado para el fin previsto. Antes de comenzar la evaluación, fue fundamental definir su propósito y alcance, que nos llevó a incluir

la definición de los criterios de evaluación, los usuarios previstos del software y los resultados esperados de la evaluación.

La evaluación del software nos permitió detectar y solucionar posibles problemas en una fase temprana del proceso de desarrollo. Esto ayudó a reducir el número de defectos, fallos y errores en el software. También consideramos que ahorra tiempo y dinero a largo plazo y mejorar la calidad general del software.

Definitivamente la evaluación de nuestro software fue fundamental para la satisfacción del usuario. Logramos determinar que el software satisface las necesidades de los usuarios previstos, evaluándolo desde la perspectiva del usuario. Fue necesario probar la funcionalidad, el rendimiento y la facilidad de uso del software, así como revisar su documentación.

La evaluación del software nos proporcionó información útil para futuras versiones. Lo cual ayuda a garantizar que el software se mantenga actualizado, satisfaga las necesidades cambiantes de los usuarios y siga siendo competitivo en el mercado.

En su comportamiento revisamos el diseño del software, probamos su funcionalidad y rendimiento, como también evaluamos la documentación del software.

Una vez finalizada la evaluación, se analizaron los resultados para identificar los puntos fuertes y débiles y las oportunidades de mejora. El análisis fue lo más imparcial posible en función de los criterios establecidos.

Este informe incluye los criterios de evaluación, un resumen del proceso de evaluación, los resultados de la evaluación y sugerencias para mejorar las cosas.

Concluimos que en el proceso de evaluación del software se deben involucrar las partes interesadas, los usuarios, administradores, el equipo informático y de seguridad. Por lo tanto, deben de trabajar juntos para determinar si el software es viable o útil. Y el tamaño del equipo de evaluación dependerá en gran medida del tamaño de la empresa y del tipo de software que se esté evaluando.

La evaluación del software es un proceso crítico para las empresas que desean invertir en un software fiable, seguro y eficiente. La evaluación del software implica valorar varios aspectos del software, como sus características, funcionalidad, usabilidad, fiabilidad, seguridad, rendimiento, escalabilidad, capacidad de mantenimiento y rentabilidad.

4. Webgrafía

https://ejemplosweb.de/bitacoras-de-un-proyecto-tipos-ejemplos-diferencias/#:
https://gbitcorp.com/blog/posts/modelo-cmmi/
www.g-talent.net/blogs/scrum/5-practicas-de-scrum-que-todo-desarrollador
https://www.questionpro.com/blog/es/evaluacion-de-software/
https://blog.innevo.com/metricas-de-calidad