ADA 2.1 Atributos de calidad en Diseño de Software

Entrando en contexto con el propósito general del proyecto. El producto de software consistirá en el desarrollo de una aplicación de escritorio para la gestión integral de un gimnasio; para propósitos de la asignatura, se creará el *POC* del software, excluyendo las funcionalidades completas pero estableciendo las bases arquitectónicas y de diseño.

Atributos de Calidad según el Modelo de McCall:

Fiabilidad: La capacidad del software para desempeñarse de manera confiable bajo diversas condiciones y entornos. Se refiere a la consistencia en la ejecución de las funciones y la prevención de fallos que puedan afectar la experiencia del usuario [1].

Una actividad centrada en la fiabilidad es la elaboración del Plan de Pruebas de Calidad. Este plan incluye escenarios de prueba que abarcan aspectos como la funcionalidad principal, la estabilidad, la tolerancia a fallos y la capacidad de recuperación. Un ejemplo real es el SO Windows de Microsoft, que implementa estrategias para minimizar fallos. Microsoft realiza pruebas exhaustivas, como las de unidad, integración y aceptación del usuario, para identificar y corregir posibles problemas antes del lanzamiento.

Mantenibilidad: La mantenibilidad combina la capacidad del programa para ser ampliable (extensibilidad), adaptable y servicial, y además que pueda probarse, ser compatible y configurable y que cuente con la facilidad para instalarse en el sistema y para que se detecten los problemas [1].

Para la mejora de la mantenibilidad del sistema, se destaca la creación de un diagrama de estructura del sistema (SSD). Durante esta actividad, los diseñadores visualizan los diversos componentes del sistema, como módulos de software, interfaces de usuario y bases de datos, junto con sus relaciones y dependencias. Un ejemplo práctico es el uso de SSDs en el framework de desarrollo web Ruby on Rails, una herramienta eficaz, demuestra su utilidad al identificar y modificar componentes individuales.

Usabilidad: Grado en que un sistema, producto o servicio puede ser utilizado por usuarios específicos para lograr objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado [2].

Una herramienta basada en la usabilidad es la creación y evaluación de prototipos de interfaz de usuario. Durante esta etapa, los diseñadores visualizan la interfaz del software mediante representaciones visuales interactivas, que van desde bocetos simples hasta prototipos completamente funcionales. Una herramienta con la cual se puede implementar esta actividad es InVision, una plataforma de diseño y colaboración que facilita la creación de prototipos interactivos de alta fidelidad. Los diseñadores pueden importar diseños desde

distintas herramientas, y luego vincular diferentes pantallas para probar la navegación y funcionalidad de la aplicación o sitio web.

Eficiencia: La capacidad del software para utilizar los recursos disponibles de manera óptima, como el tiempo de respuesta, el consumo de memoria y el rendimiento general del sistema. Se busca minimizar el impacto en el hardware y maximizar la velocidad de ejecución. [1]

Durante la fase de diseño de software, la optimización de algoritmos es esencial para mejorar la eficiencia de los sistemas. Esto implica analizar algoritmos existentes, identificar áreas de mejora, investigar técnicas de optimización, implementar cambios, realizar pruebas de rendimiento y ajustes finales. Un ejemplo real de esta actividad es el desarrollo de Google Chrome, donde la optimización del algoritmo de renderizado fue crucial para garantizar un rendimiento rápido y eficiente al cargar y renderizar páginas web, proporcionando así una experiencia de navegación fluida y receptiva a los usuarios.

Referencias

[1] Pressman, R. S. (2010). Software Engineering: A Practitioner's Approach (7ma). McGraw Hill

[2] ISO. (2018). ISO 9241-11:2018 - Ergonomics of human-system interaction. Part 11: Usability: Definitions and concepts