**1. Adaptabilidad**

**- Definición:** La capacidad del software para ser adaptado eficientemente a diferentes entornos de operación sin requerir modificaciones adicionales.

**- Ejemplo:** Imagina un sistema de gestión de contenidos (CMS) que puede ser fácilmente configurado para diferentes tipos de sitios web con variados requisitos de contenido y diseño. Esto es crucial para que el CMS pueda servir a una amplia gama de usuarios y necesidades específicas sin necesidad de rehacer todo el desarrollo desde cero cada vez.

**2. Fiabilidad**

**- Definición:** La capacidad del software para mantener el desempeño esperado bajo condiciones específicas y durante un período de tiempo determinado.

**- Ejemplo:** Piensa en un sistema bancario que gestiona transacciones críticas de manera consistente sin fallos significativos. Esto es fundamental para mantener la confianza de los clientes y asegurar que las operaciones financieras se realicen de manera segura y confiable día tras día.

**3. Rendimiento**

**- Definición:** La medida de la capacidad del software para responder eficientemente a las demandas y realizar sus funciones dentro de límites establecidos.

**- Ejemplo**: Considera un servidor de aplicaciones que puede manejar concurrentemente un alto número de solicitudes web sin experimentar degradación significativa en el tiempo de respuesta. Esto asegura que las aplicaciones y servicios alojados en el servidor puedan atender a múltiples usuarios simultáneamente sin afectar su velocidad de respuesta.

**4. Funcionalidad**

**- Definición:** El conjunto de funciones y características que ofrece el software para satisfacer las necesidades especificadas.

**- Ejemplo:** Un software de contabilidad que permite gestionar libros mayores, emitir facturas, y generar informes financieros. Esto permite a las empresas llevar un registro preciso de sus finanzas y cumplir con sus obligaciones contables de manera eficiente.

**5. Innovación**

**- Definición:** La capacidad del software para incorporar nuevas ideas, métodos o tecnologías que mejoren significativamente su valor y eficacia.

**- Ejemplo:** Imagina una aplicación de redes sociales que introduce un nuevo algoritmo de recomendación de contenido basado en inteligencia artificial. Esto no solo enriquece la experiencia del usuario al mostrar contenido relevante, sino que también mantiene la plataforma competitiva y atractiva en un mercado cambiante.

**6. Escalabilidad**

**- Definición**: La capacidad del software para adaptarse y manejar eficientemente un creciente volumen de trabajo o usuarios.

**- Ejemplo**: Un servicio de almacenamiento en la nube que puede aumentar la capacidad de almacenamiento y el rendimiento de manera dinámica conforme más usuarios acceden simultáneamente. Esto asegura que el servicio pueda expandirse según las necesidades de los usuarios sin comprometer su velocidad o confiabilidad.

**7. Estrategias**

**- Definición**: Los planes o métodos utilizados para gestionar el desarrollo, mantenimiento y mejora continua del software.

**- Ejemplo:** La implementación de metodologías ágiles como Scrum o Kanban para gestionar el desarrollo de software de manera eficiente y colaborativa. Estas estrategias ayudan a los equipos a gestionar proyectos de manera efectiva, fomentando la comunicación y la flexibilidad en la adaptación a cambios durante el ciclo de vida del desarrollo.

**8. Estándares**

**- Definición:** Conjunto de reglas y directrices que definen cómo debe ser desarrollado y evaluado el software para asegurar su calidad y compatibilidad.

**- Ejemplo:** Un proyecto de desarrollo de software que sigue estrictamente las normas ISO/IEC 9126 para garantizar la calidad del producto final. Cumplir con estándares establecidos asegura que el software sea confiable, seguro y fácilmente mantenible, siguiendo prácticas reconocidas a nivel internacional.

**9. Seguridad**

**- Definición:** El grado en el cual el software protege la información y los recursos contra accesos no autorizados y asegura la integridad de los datos.

**- Ejemplo:** Un sistema de gestión de identidades que utiliza autenticación multifactorial y encriptación fuerte para proteger los datos sensibles de los usuarios. Garantizar la seguridad es esencial para proteger la privacidad y la confidencialidad de la información manejada por el software.

**10. Eficacia**

**- Definición:** La medida de la precisión y efectividad con la cual el software cumple sus objetivos y requisitos.

**- Ejemplo:** Un software de reconocimiento facial que logra identificar correctamente a los usuarios con una tasa de precisión superior al 95%. Esto asegura que el software cumpla con su propósito principal de manera confiable y precisa.

**11. Confiabilidad**

**- Definición:** La capacidad del software para realizar funciones requeridas de manera consistente bajo condiciones específicas durante un período de tiempo.

**- Ejemplo:** Un sistema de reservas de vuelos en línea que permite a los usuarios reservar y pagar por boletos sin errores críticos o fallos del sistema. La confiabilidad es fundamental para mantener la operatividad continua y la satisfacción del usuario.

**12. Eficiencia**

**- Definición:** La capacidad del software para lograr sus objetivos utilizando la menor cantidad de recursos posibles.

**- Ejemplo**: Un algoritmo de compresión de archivos que reduce el tamaño de los archivos sin pérdida significativa de calidad o información. Ser eficiente permite al software realizar tareas complejas de manera rápida y económica en términos de recursos.

**13. Precisión**

**- Definición**: El grado de exactitud y corrección con el cual el software realiza sus operaciones y produce resultados esperados.

**- Ejemplo:** Un software de análisis estadístico que calcula correctamente los valores medios y desviaciones estándar de un conjunto de datos. La precisión es crucial para garantizar la fiabilidad de las conclusiones derivadas del análisis realizado por el software.

**14. Funcional**

**- Definición**: Relativo a la capacidad del software para realizar las funciones específicas para las cuales fue diseñado.

**- Ejemplo**: Una aplicación de edición de imágenes que permite ajustar el brillo, contraste y color de las fotografías de manera intuitiva y efectiva. La funcionalidad adecuada permite a los usuarios realizar tareas específicas de manera eficiente y satisfactoria.

**15. Flexibilidad**

**- Definición:** La capacidad del software para ser adaptado o configurado fácilmente para satisfacer diferentes necesidades o requisitos cambiantes.

**- Ejemplo:** Un sistema de gestión de inventarios que permite a los usuarios personalizar campos de datos y procesos de acuerdo a las especificaciones únicas de su negocio. La flexibilidad facilita la personalización del software según las necesidades específicas de cada usuario o situación.

**16. Portabilidad**

**- Definición:** El grado en el cual el software puede ser transferido de un entorno de desarrollo a otro, o ejecutado en diferentes plataformas con mínimas modificaciones.

**- Ejemplo:** Un juego para móviles que puede ser ejecutado tanto en dispositivos Android como iOS sin necesidad de cambios significativos en el código. La portabilidad asegura que el software pueda ser utilizado en diferentes entornos sin perder funcionalidad o eficiencia.

**17. Satisfacción**

**- Definición:** La medida del grado en el cual el software cumple las expectativas y necesidades de los usuarios finales.

**- Ejemplo**: Un sistema de gestión de atención al cliente que recibe altas calificaciones de satisfacción debido a su interfaz amigable y tiempos de respuesta rápidos. La satisfacción del usuario es crucial para la aceptación y éxito continuo del software en el mercado.

**18. Alcance**

**- Definición:** La extensión y profundidad de las funcionalidades y características que ofrece el software.

**- Ejemplo**: Un sistema de gestión educativa que abarca desde la administración de alumnos y profesores hasta la planificación de cursos y la evaluación del rendimiento estudiantil. El alcance amplio asegura que el software pueda cubrir todas las necesidades relevantes de su campo de aplicación.

**19. Consistencia**

    - Definición: El grado en el cual el software mantiene un patrón coherente en su comportamiento, interfaz y resultados a lo largo del tiempo y entre diferentes usuarios.

    - Ejemplo: Un software de diseño gráfico que utiliza atajos de teclado consistentes y menús contextuales que facilitan la navegación y la ejecución de tareas repetitivas. La consistencia mejora la experiencia del usuario al hacer que el software sea predecíble y fácil de usar.

**20. Mejoras**

**- Definición:** Actualizaciones y cambios implementados en el software para añadir nuevas funcionalidades, mejorar el rendimiento, o corregir errores.

**- Ejemplo:** Una aplicación de mensajería instantánea que regularmente introduce mejoras como la adición de emojis nuevos y la optimización del consumo de batería. Las mejoras continuas mantienen el software relevante y competitivo en un mercado dinámico.

**21. Resolución**

**- Definición:** La capacidad del software para manejar problemas y conflictos de manera efectiva y satisfactoria.

**- Ejemplo:** Un sistema de gestión de incidencias que asigna automáticamente tickets a los técnicos más adecuados y realiza un seguimiento detallado del estado de resolución. La resolución eficiente de problemas asegura que los usuarios puedan superar obstáculos sin interrupciones prolongadas en su trabajo.

**22. Prevenir**

**- Definición:** Acciones implementadas en el software para evitar la ocurrencia de problemas o situaciones no deseadas.

**- Ejemplo:** Un sistema de detección de intrusiones que utiliza patrones de comportamiento para identificar y bloquear intentos de acceso no autorizado antes de que ocurran daños. La prevención proactiva protege al software y a los datos contra amenazas potenciales.

**23. Garantizar**

**- Definición:** Medidas adoptadas para asegurar que el software cumpla con estándares de calidad y expectativas definidas.

**- Ejemplo:** Un proceso de pruebas exhaustivas que se realiza antes del lanzamiento de un nuevo software para garantizar que cumple con todos los requisitos funcionales y de seguridad. La garantía de calidad es esencial para lanzar software confiable y libre de defectos al mercado.

**24. Mantenibilidad**

**- Definición:** La facilidad con la cual el software puede ser modificado o mejorado.

**- Ejemplo:** Un sistema de gestión de código fuente que utiliza prácticas de desarrollo ágil y tiene una arquitectura modular que facilita la incorporación de nuevas funcionalidades. La mantenibilidad asegura que el software pueda adaptarse rápidamente a cambios en los requisitos o tecnologías emergentes.

**25. Reusabilidad**

**- Definición**: La capacidad del software para ser utilizado en múltiples aplicaciones o contextos con mínimas modificaciones.

**- Ejemplo:** Una biblioteca de componentes de interfaz de usuario que puede ser incorporada fácilmente en diferentes aplicaciones para mejorar la consistencia visual y la experiencia del usuario. La reusabilidad reduce el tiempo y el costo de desarrollo al permitir el uso repetido de componentes probados.

**26. Renovación**

**- Definición:** El proceso de actualizar el software para mantenerlo relevante y eficaz en un entorno cambiante.

**- Ejemplo:** Un sistema de gestión de contenidos que regularmente incorpora nuevas plantillas y herramientas de edición para adaptarse a las últimas tendencias de diseño web. La renovación asegura que el software pueda competir y satisfacer las expectativas del usuario en un mercado dinámico.

**27. Propiedades**

**- Definición:** Características inherentes o atributos específicos que definen la calidad y funcionalidad del software.

**- Ejemplo:** Un software de modelado 3D que destaca por sus propiedades avanzadas de renderizado y simulación de físicas. Las propiedades distintivas del software lo hacen único y valioso en su campo de aplicación.