

TABLA. PRÁCTICAS SOSTENIBLES DEL SOFTWARE

Prácticas sostenibles	Descripción	Referencias
Sostenibilidad en el ciclo de vida del software	Se refiere a la práctica de desarrollar, implementar y mantener software de manera que minimice su impacto ambiental, social y económico a lo largo de su vida útil. Esto implica considerar aspectos como la eficiencia energética, el uso responsable de recursos, la gestión adecuada de residuos de cómputo y uso del software.	[1]
Gestión de requisitos	Se refiere a identificar, documentar, validar y gestionar los requisitos funcionales y no funcionales del software y que a lo largo de su ciclo de vida sea sostenible.	[3] [22] [20] [21]
Gestión y documentación ágil	Se refiere a la aplicación de los principios ágiles en la gestión y documentación de los procesos de desarrollo, entrega y operación de software dentro de un enfoque DevOps.	[22] [23] [24]
Diseño modular y escalable	Se refiere a la práctica de diseñar sistemas divididos en módulos independientes y cohesivos, capaces de crecer y adaptarse fácilmente a medida que las necesidades del negocio cambian.	[22] [23] [24]
Aseguramiento de Calidad de software sostenible	Se refiere a la integración de principios y prácticas de calidad en todo el ciclo de vida del desarrollo de software, con el objetivo de garantizar la entrega de software de alta calidad de manera continua y sostenible.	[22] [23] [24]
Optimización de recursos	Se refiere a la práctica de utilizar eficientemente los recursos disponibles en el proceso y operación del software. Esperando minimizar el desperdicio, reducir el tiempo de espera, y maximizar el valor entregado al cliente utilizando la menor cantidad de recursos posibles, como tiempo, dinero, hardware, energía	[22] [23] [24]
Entrega continua	Se refiere al proceso automatizado y repetible de integrar, probar y desplegar cambios de forma continua, oportuna y segura.	[2] [3] [18] [22] [23] [24]
Integración continua	Consiste en fusionar de manera frecuente y automatizada cambios realizados a nivel del código fuente de un proyecto en un repositorio compartido. Esta práctica tiene implicaciones positivas para la sostenibilidad al reducir desperdicio.	[2] [18] [22] [23] [24]
Control de versiones	Se refiere a la gestión y rastreo de cambios en el código fuente y otros artefactos de software a lo largo del tiempo utilizando herramientas específicas, para mantener un historial, revertir a versiones anteriores si es necesario, y/o coordinar el trabajo en paralelo.	[22] [23] [24]
Gestión de la configuración	Se refiere al control de los elementos de configuración del software, como pueden ser el código fuente, la infraestructura, las dependencias y o todo aquello necesario para el desarrollo, la implementación y la operación del sistema.	[22] [23] [24]
Desarrollo basado en troncos	Consiste en trabajar directamente sobre la rama principal (o tronco) del repositorio de código, en lugar de crear ramas separadas para el desarrollo de nuevas funcionalidades o correcciones de errores. Esta práctica promueve la integración continua y la entrega continua al facilitar una colaboración más fluida y una mayor visibilidad del progreso del trabajo en curso.	[22] [23] [24]
Pruebas continuas y automatizadas	Se refiere a la ejecución automática de pruebas de software en cada etapa del proceso de desarrollo y entrega, desde la integración hasta la implementación y más allá.	[22] [23] [24]
Desarrollo dirigido por hipótesis	Se centra en la experimentación y la validación de ideas a través de hipótesis claras y medibles. Se definen experimentos y luego se ajustan en función de los resultados obtenidos.	[20]
Gestión de datos de pruebas	Se refiere a la administración de los datos utilizados en las pruebas de software, esto implica asegurar que estos sean confiables, consistentes, relevantes y estén disponibles cuando sean necesarios.	[22] [23] [24]

Despliegue continuo	Se refiere al proceso de implementación de cambios en el software de manera que estos cambios puedan ser entregados a ambientes de producción de forma rápida, frecuente y confiable.	[22] [23] [24]
Sostenibilidad técnica	Se refiere a la capacidad de un sistema de software para mantener su integridad, calidad y eficacia a lo largo del tiempo, incluso a medida que se realizan cambios y se añaden nuevas funcionalidades en aspectos de infraestructura y arquitectura.	[1] [9]
Gestión técnica o de arquitectura	Se refiere a las prácticas y procesos utilizados para gestionar la calidad técnica y la arquitectura del software a lo largo de su ciclo de vida. Esto incluye decisiones relacionadas con el diseño, la implementación, la refactorización, la automatización, la integración continua, la entrega continua y otras actividades propias de DevOps.	[22] [23] [24]
Arquitectura débilmente acoplada	Se refiere al diseño e implementación de manera que tienen pocas dependencias entre sí, limitando el impacto en la integración, la evolución y la mantenibilidad del sistema	[22] [23] [24]
Infraestructura como código	Se refiere a la práctica de gestionar y provisionar infraestructura de TI utilizando archivos de código, esto implica escribir scripts o definiciones de configuración utilizando lenguajes de programación.	[3] [22] [23] [24]
Infraestructura de nube	Se refiere a la utilización de recursos informáticos, como servidores, almacenamiento y redes, que son proporcionados y gestionados a través de servicios de nube pública, privada o híbrida.	[22] [23] [24]
Seguridad como Código	Se refiere a la configuración de seguridad, las políticas de acceso, la detección de vulnerabilidades y otras medidas de seguridad como código, lo que significa que se gestionan y aplican de manera automatizada y reproducible, utilizando herramientas.	[22] [23] [24]