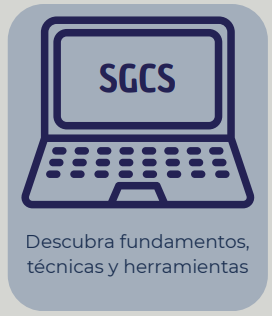
**INICIO**

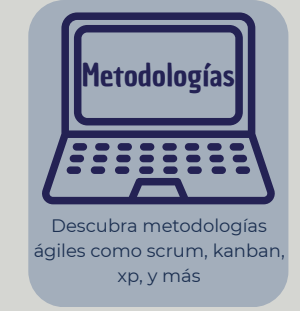
¡Bienvenido a nuestra página dedicada al mundo de la ingeniería de software! Acá vas a encontrar lo necesario para comprender como se construyen, gestionan y mejoran los sistemas de software modernos. Exploraremos los sistemas de gestión de la calidad del software (SGCS), los procesos de desarrollo y las metodologías agiles. Además, tenes acceso a un espacio de recursos y aprendizaje, con glosarios, preguntas frecuentes, enlaces de interés y ejercicios prácticos, junto con una sección dedicada a nuestro equipo y motivación. Todo esto es un sitio pensado para que aprendas de manera clara, ordenada y práctica.



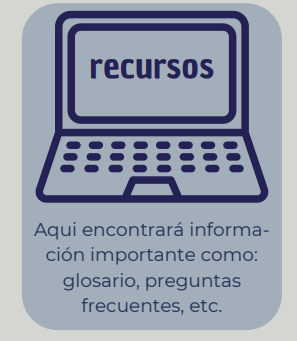
**SGCS**  
Descubre los fundamentos de la calidad del software, sus métricas y herramientas clave para asegurar productos confiables y eficientes.



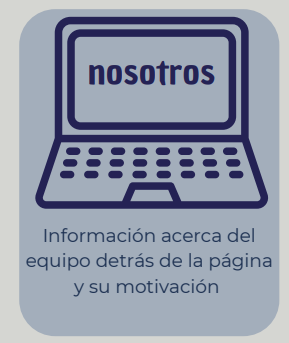
**Procesos**  
Conoce las etapas del ciclo de vida del software y los modelos tradicionales que guían la planificación y construcción de proyectos.



**Metodologías**  
Aprende cómo trabajan los equipos con metodologías ágiles como Scrum, Kanban y XP, y compáralas con los enfoques clásicos.

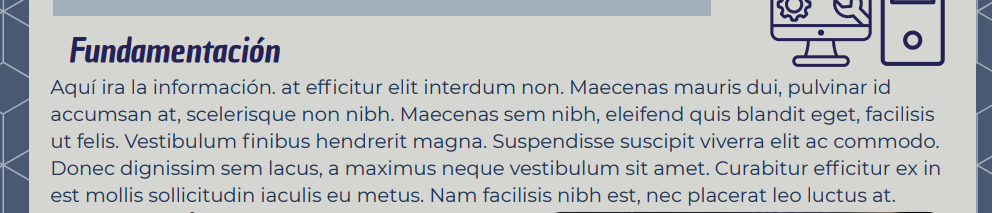


**Recursos**  
Accede a glosarios, preguntas frecuentes, enlaces de interés y casos prácticos para reforzar tu aprendizaje de forma dinámica.

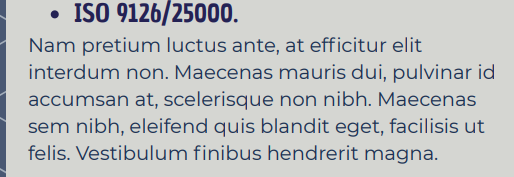


**Nosotros**  
Conoce al equipo detrás del sitio, nuestra pasión por la ingeniería de software y el objetivo que nos inspira a compartir este proyecto.

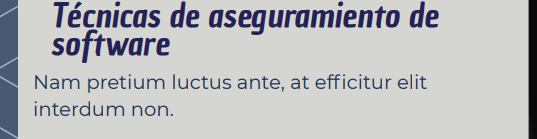
**Gestión de la Calidad del Software**

****

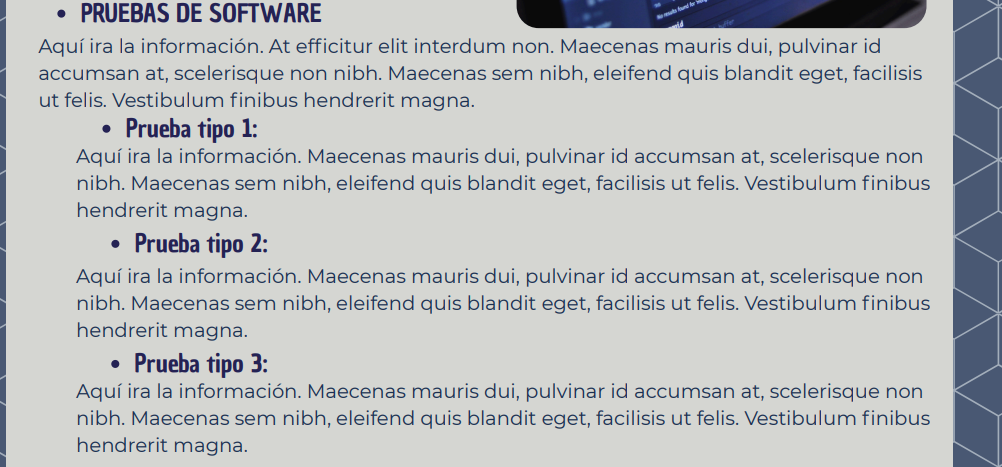
La gestión de la calidad del software busca asegurar que un producto cumpla con las necesidades de los usuarios, que sea confiable y fácil de mantener. Su objetivo es garantizar que el software funciones de manera eficiente en distintos entornos y cumpla con los estándares reconocidos. Para ello se usan métricas de calidad que permuten evaluar aspectos como la funcionalidad, la seguridad, la eficiencia, la portabilidad y la usabilidad. Gracias a estas prácticas, se minimizan errores y se le ofrece al cliente un producto de confianza y buena calidad.



La norma ISO 9126 fue de las primeras en definir características esenciales de la calidad del software, dividiéndola en atributos como la funcionabilidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad. Más adelante la remplazaron por la ISO/IEC 25000, conocida como SQauRE (Software Product Quality Requirements and Evaluation), que amplia y actualiza estas métricas. Estas normas sirven como guía internacional para medir y evaluar la calidad de un software, permitiendo que los proyectos tengan un marco de referencia común y confiable.



Son el conjunto de prácticas y actividades aplicadas a lo largo del ciclo de vida para garantizar la calidad del producto y del proceso: previenen y detectan defectos, verifican que se cumplan los requisitos y validan que el software funcione en condiciones reales.



Pruebas de software

Son el corazón del aseguramiento de calidad. Nos permite detectar errores, validad requerimientos y garantizar que el sistema funcione antes de ser entregado al usuario. Se realizan en diferentes etapas y niveles, adaptándose a la complejidad del proyecto.

**Prueba unitaria:**

Evalúa módulos o componentes individuales del software. El objetivo es asegurar que cada parte aislada funcione de acuerdo con lo esperado. Son pruebas rápidas y suelen automatizarse para mayor eficacia.

**Prueba de integración:**

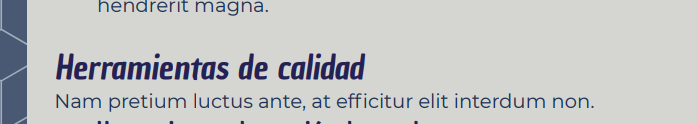
Verifica cómo interactúan los distintos módulos entre sí. Su meta es encontrar errores en la comunicación y compatibilidad de componentes. Permite descubrir problemas que no se ven en pruebas unitarias.

**Prueba de sistema:**

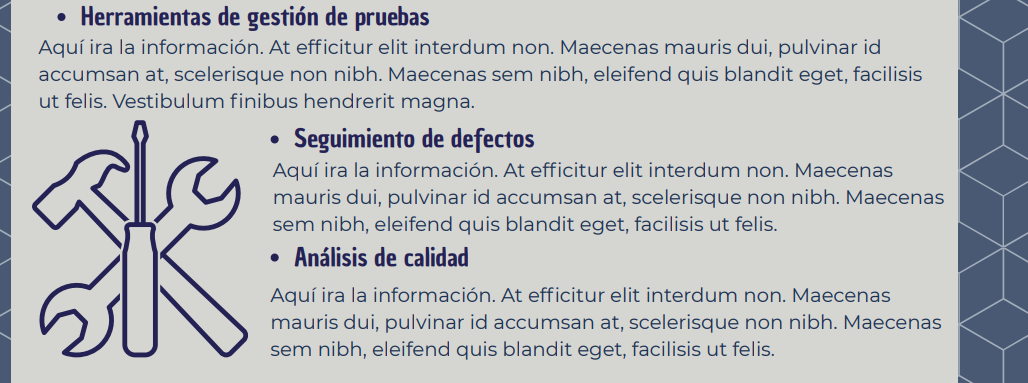
Examina el software completo como un todo. Acá se valida que el sistema cumpla con los requerimientos funcionales y no funcionales establecidos en la especificación.

**Prueba de aceptación:**

Se realizan junto al cliente o usuario final, con el fin de comprobar que el software satisface sus necesidades reales. Es la última etapa antes de la entrega del producto.



Son aplicaciones que automatizan y organizan las actividades de aseguramiento.



**Herramientas de gestión de pruebas**

Permiten organizar casos de pruebas, registrar resultados y llevar u control de que funcionalidades fueron evaluadas. Ayudan a planificar y ejecutar pruebas de forma sistemática, reduciendo olvidos o repeticiones innecesarias.

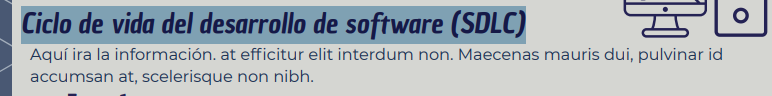
**Seguimiento de defectos:**

Registran errores detectados durante las pruebas, asignan prioridades y los distribuyen entre los desarrolladores para su corrección. Fundamentales para mantener trazabilidad y garantizar que no queden problemas sin resolver.

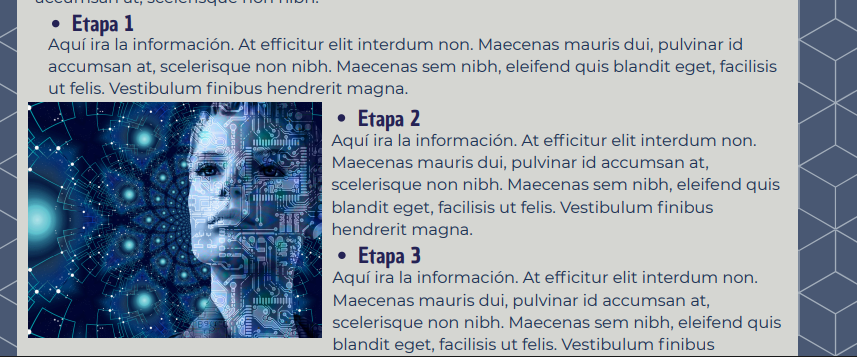
**Análisis de calidad:**

Examinan el código fuente para detectar malas prácticas, vulnerables o posibles fallos antes de ejecutar el software. Ejemplos son SonarQube para análisis estático y Selenium para automatizar pruebas funcionales. Con su uso, se mejora la seguridad y se aumenta la confianza en el producto final.

**Procesos de desarrollo de software**



El ciclo de vida del software (SDLC) es un proceso estructurado que define las etapas necesarias para crear un sistema de calidad. Nos permite organizar desde la planificación inicial hasta el mantenimiento final, asegurando que el producto cumpla con los requisitos del cliente y los estándares de calidad establecidos. Cada etapa tiene un objetivo específico y contribuye a un desarrollo ordenado y eficiente.



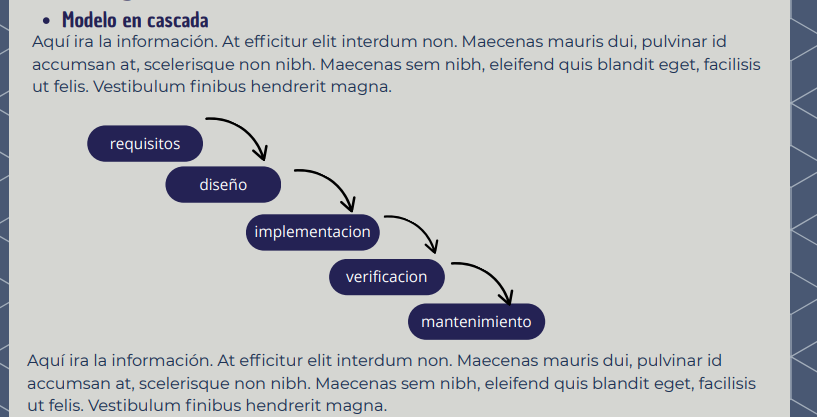
**Etapa 1: Análisis de requisitos**  
En esta fase se identifican y documentan las necesidades del cliente, así como las funcionalidades que debe cumplir el sistema. Es crucial porque de acá surgen las bases del proyecto y se definen los objetivos a alcanzar.

**Etapa 2: Diseño**  
Consiste en transformar los requisitos en una arquitectura técnica que guíe el desarrollo. Se definen diagramas, interfaces, estructuras de datos y modelos que servirán como plano para la programación.

**Etapa 3: Implementación**  
En esta etapa los programadores convierten el diseño en código funcional. Se desarrollan los módulos del software, integrando las funciones planificadas y asegurando su correcto funcionamiento inicial.

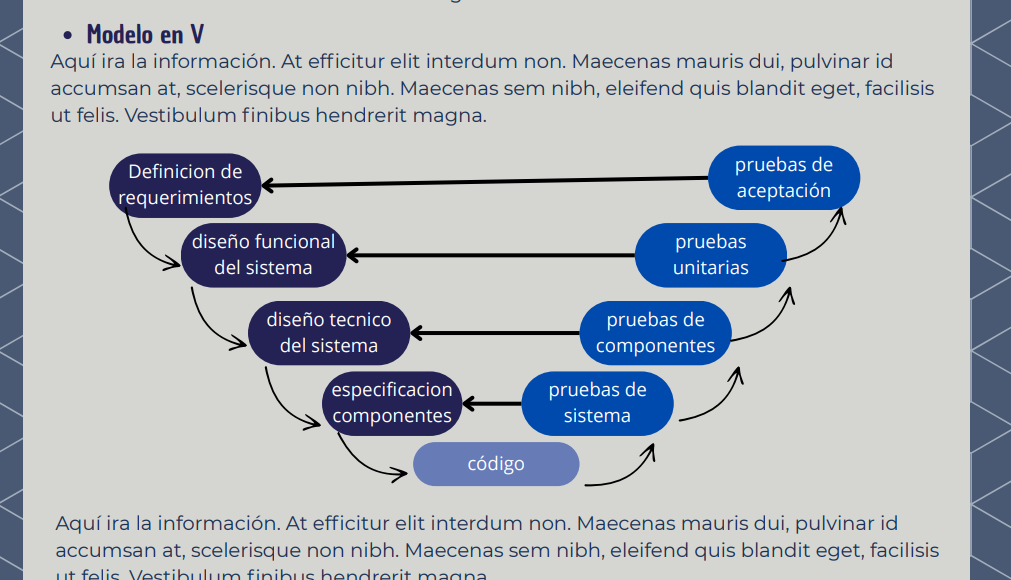
**Etapa 4: Pruebas**  
Se realizan distintos tipos de pruebas para validar que el software cumple con los requisitos planteados y que no existan errores críticos. El objetivo es garantizar calidad, rendimiento y confiabilidad del sistema.

**Etapa 5: Mantenimiento**  
El software, una vez entregado, requiere correcciones, actualizaciones y mejoras para adaptarse a cambios en los requerimientos o en el entorno tecnológico. Esta fase asegura la continuidad y vigencia del sistema.



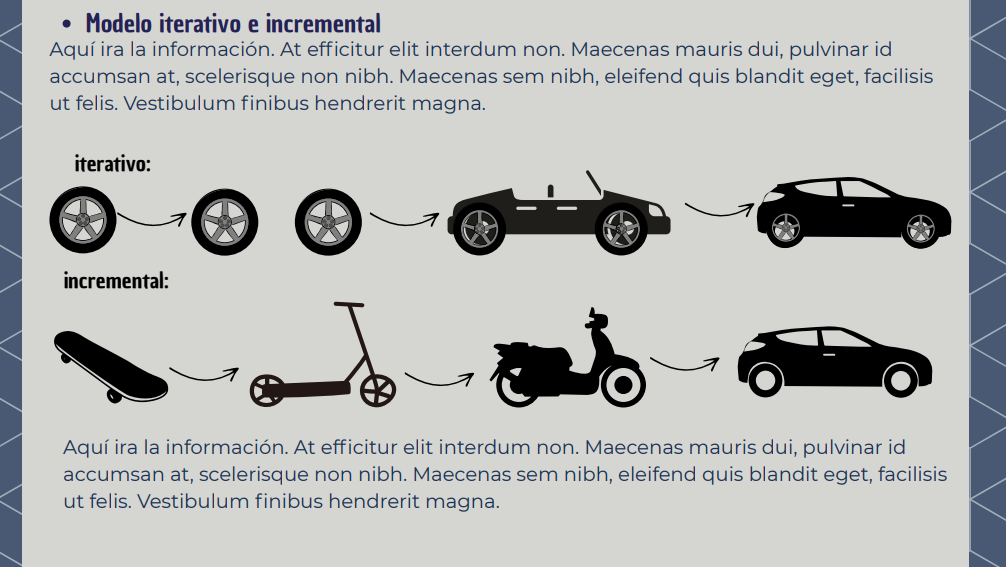
El modelo en cascada es uno de los enfoques más clásicos del desarrollo de software. Se basa en avanzar de manera secuencial a través de fases bien definidas: requisitos, diseño, implementación, verificación y mantenimiento. Cada etapa debe completarse antes de pasar a la siguiente, lo que facilita la planificación y el control del proyecto.

Su principal fortaleza es la claridad y el orden, pero la rigidez es su mayor desventaja: si surge un error en fases tempranas o cambian los requisitos del cliente, resulta costoso y complicado volver atrás. Por eso, se recomienda en proyectos con requisitos muy bien definidos y estables.



El modelo en V surge como una mejora al modelo en cascada, manteniendo su estructura secuencial, pero vinculando cada fase de desarrollo con una fase de prueba equivalente. Así, por ejemplo, el diseño de requisitos se relaciona con las pruebas de aceptación, el diseño técnico con las pruebas de integración y el desarrollo de componentes con las pruebas unitarias.

De esta forma se asegura que cada etapa tenga un control de calidad inmediato, reduciendo el riesgo de errores críticos al final del proyecto. Sin embargo, al igual que el modelo en cascada, es poco flexible ante cambios inesperados, por lo que se adapta mejor a proyectos donde los requisitos están claros desde el inicio.



Este modelo propone desarrollar el software en ciclos repetidos (iteraciones) y entregar versiones funcionales parciales (incrementos) en cada etapa. En lugar de esperar al final para ver el producto completo, los usuarios reciben versiones intermedias que pueden probar y validar, lo que permite detectar errores y ajustar los requisitos de manera temprana.

Su mayor ventaja es la flexibilidad y la retroalimentación constante entre desarrolladores y clientes, lo que mejora la calidad y satisfacción del producto final. No obstante, requiere una gestión más compleja y mayor compromiso del cliente, ya que su participación es clave durante todo el proceso.