Facultad de Matemáticas

Licenciatura en Ingeniería de Software

Aseguramiento de la Calidad de Software

**Equipo 7**

Plan de Aseguramiento de la calidad de software

InspiraTech - Versión 2.0 (2025)

**Docente:**

Edwin Jesús León Bojórquez

Becky Zhu Wu

Erick Gilberto Gómez Manzanero

Jesús Oswaldo Chan Uicab

Juan Omar Trivellari Ramirez

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Fecha | Estado | Responsables | Descripción del Cambio |
| 1.0 | 26/03/2025 | Creación | InspiraTech | Creación del Documento |
| 1.1 | 16/04/2025 | Revisión | InspiraTech | Extensión de resultados |
| 2.0 | 27/05/2025 | Revisión final | InspiraTech | Agregar anexoss y revisión final |

[Propósito 3](#_Toc1188349531)

[Documentos de referencia 3](#_Toc1718360875)

[Gestión 4](#_Toc906940508)

[Organización 4](#_Toc175211208)

[4](#_Toc876111508)

[Tareas 4](#_Toc2047613874)

[1. Planificación de la SQA 4](#_Toc1445938135)

[2. Revisión de Requisitos 5](#_Toc1950970590)

[3. Control de Diseño y Arquitectura 5](#_Toc799533465)

[4. Revisiones de Código y Control de Configuración 5](#_Toc438095065)

[5. Verificación y Validación 5](#_Toc1934681676)

[6. Auditorías y Revisiones de Progreso 5](#_Toc1155687908)

[7. Gestión de Incidentes y No Conformidades 6](#_Toc1232691141)

[8. Publicación y Mantenimiento del SQAP 6](#_Toc157120868)

[Responsabilidades 6](#_Toc456434544)

[Documentación 9](#_Toc680508600)

[Propósito 9](#_Toc1726159740)

[Documento de especificación de Requisitos 9](#_Toc2036269031)

[Descripción del diseño de software 9](#_Toc2101636982)

[Plan de verificación y validación 10](#_Toc504114747)

[Informe de Resultados de la Verificación 10](#_Toc986576909)

[Informe de Resultados de Validación 10](#_Toc1727818540)

[Documentación de Usuario 11](#_Toc2109441015)

[Plan de Gestión de Configuración de Software (SCMP) 11](#_Toc637919498)

[Plan de Desarrollo de Software (SDP) 11](#_Toc375658012)

[Manual de Mantenimiento de Software (SMM) 12](#_Toc1111307791)

[Especificaciones de la Interfaz Externa 12](#_Toc50034481)

[Especificaciones de la Interfaz Interna 13](#_Toc1858962151)

[Plan de Métricas de Software 13](#_Toc1403727700)

[Plan de Seguridad del Software 13](#_Toc293903064)

[Estándares, practicas, convenciones y métricas. 14](#_Toc400795132)

[Revisiones y auditorías 14](#_Toc1695822550)

[Propósito 14](#_Toc1681332870)

[Revisión de requisitos de software (SRR) 14](#_Toc1733905763)

[Resultados de la SRR: 15](#_Toc1303503145)

[Revisión preliminar de diseño (PDR) 15](#_Toc1281317168)

[Resultados de la PDR: 16](#_Toc1368322054)

[Revisión crítica del diseño (CDR) 16](#_Toc429416013)

[Resultados de la CDR: 16](#_Toc1123014271)

[Revisión del plan de verificación y validación de software (SVVP) 17](#_Toc1330241201)

[Resultados de la SVVPR: 17](#_Toc1087229423)

[Auditorías funcionales 17](#_Toc1549134154)

[Auditorías físicas 18](#_Toc746352161)

[Auditorías en proceso 18](#_Toc1955998111)

[Revisiones gerenciales 18](#_Toc985968161)

[Resultados de las revisiones gerenciales: 19](#_Toc1096767574)

[Revisión del Plan de Gestión de la Configuración del Software (SCMPR) 19](#_Toc850628026)

[Resultados de la revisión del plan de gestión de la configuración del software: 19](#_Toc1255214442)

[Revisión posterior a la implementación 19](#_Toc1452752801)

[Resultados de la revisión posterior a la implementación 20](#_Toc2143486978)

[Revisión de Documentación para el Usuario (UDR) 20](#_Toc916422491)

[Resultados de la revisión de documentación para el usuario: 20](#_Toc50089045)

[Auditoria de aseguramiento de calidad 20](#_Toc1291750911)

[Pruebas 21](#_Toc1308530543)

[Planificación de pruebas 21](#_Toc999537126)

[Preparación de entorno 21](#_Toc1241137868)

[Secuencia de ejecución 21](#_Toc806319709)

[Tipos de pruebas 21](#_Toc735215604)

[Pruebas a nivel unidad 21](#_Toc310696994)

[Pruebas de integración 22](#_Toc766000969)

[Pruebas a nivel sistema 22](#_Toc1868881613)

[Pruebas de estrés 22](#_Toc1800569015)

[Pruebas de seguridad 23](#_Toc342771535)

[Pruebas de aceptación 23](#_Toc1026191844)

[Pruebas de caja blanca y caja negra 24](#_Toc1894495358)

[Control de Código 27](#_Toc1052298520)

[Control de medios 28](#_Toc1455897717)

[Recopilación, mantenimiento y retención de documentos 29](#_Toc1438495414)

[Colección de documentos 30](#_Toc384245357)

[Mantenimiento de documentos 31](#_Toc608742926)

[Retención de documentos 31](#_Toc1579831530)

[Capacitación 32](#_Toc2039738558)

[Referencias 33](#_Toc253817088)

# Propósito

El propósito de este Plan de Aseguramiento de la Calidad del Software (SQAP) es establecer las pautas, procedimientos y estándares que garanticen la calidad del software desarrollado durante las etapas de diseño, implementación y pruebas para “DevWebCamp”, una plataforma web que permitirá a los usuarios registrarse, explorar y adquirir acceso a conferencias sobre diversos temas tecnológicos mediante un sistema de compra de pases. Es un proyecto moderadamente crítico, ya que un mal funcionamiento podría afectar la experiencia de usuario, la gestión de pagos o el acceso a contenido.

Este plan es de utilidad para el equipo de desarrollo, “testers”, administradores y stakeholders involucrados en DevWebCamp, incluyendo a los desarrolladores equipo de aseguramiento de calidad, “product managers”, clientes o patrocinadores.

El desarrollo del proyecto se llevará a cabo utilizando Laravel como framework de backend, lo que permitirá una estructura escalable y segura para la gestión de datos y funcionalidades. En el frontend, se empleará Tailwind CSS para garantizar una interfaz moderna y adaptable, optimizando la experiencia del usuario en distintos dispositivos.

Este SQAP se basa en IEEE Std 730-2002 y se elabora para **c**umplir con los estándares del curso y garantizar que el software cumpla con los requisitos funcionales, usabilidad y seguridad básica esperados en un proyecto de este tipo. Además, servirá como guía para:

* Proporcionar métricas y criterios objetivos para evaluar la calidad del software en estas etapas específicas
* Llevar a cabo un proceso de construcción de calidad por medio de comunicar
* Cumplir con los objetivos planteados en el proyecto de manera satisfactoria.
* Mantener un código limpio y mantenible para futuras actualizaciones.
* Evitar errores críticos en el proceso de registro y pagos.
* Asegurar una buena experiencia de usuario en la navegación y acceso a conferencias.

Además, la revisión del producto de software se llevará a cabo por medio de auditorías e inspecciones formales después de cada sprint basándose en el IEEE Std 1028-2008. Standard for Software Reviews and Audits.

# Documentos de referencia

* IEEE standard 730-2014 software quality assurance processes (2014).
* Modelo de Procesos para la Industria de Software: MoProSoft. 10.13140/2.1.2229.5043 (2005).
* International standard ISO/IEC 12207 IEEE Std 12207-2008 (2008).
* IEEE Std 1028-2008. Standard for Software Reviews and Audits.

# Gestión

## Organización

Dentro de un mismo equipo de desarrollo o departamento encargado del desarrollo o mantenimiento de software hay roles y líderes que se encargan de asegurar una buena gestión del trabajo que se llevará a cabo junto con sus debidas funciones delimitadas anteriormente.

Dentro de la organización del equipo se encuentra la jerarquía de responsabilidades de la siguiente estructura.

## 

## Tareas

#### 1. Planificación de la SQA

* **Entrada:** Requisitos del software, objetivos de calidad.
* **Salida:** Plan de SQA aprobado.
* **Actividades:**
  + Definir objetivos de calidad.
  + Identificar estándares aplicables.
  + Determinar los recursos y herramientas necesarias.
  + Asignar roles y responsabilidades.
  + Establecer procedimientos para auditorías y revisiones.

#### 2. Revisión de Requisitos

* **Entrada:** Especificación de requisitos del software (SRS).
* **Salida:** Informe de revisión de requisitos.
* **Actividades:**
  + Verificar la claridad y completitud de los requisitos.
  + Evaluar la factibilidad técnica.
  + Asegurar la trazabilidad con los objetivos del proyecto.
  + Identificar posibles inconsistencias o ambigüedades.

#### 3. Control de Diseño y Arquitectura

* **Entrada:** Documentación de diseño del software.
* **Salida:** Informe de revisión de diseño.
* **Actividades:**
  + Evaluar la adecuación de la arquitectura.
  + Verificar el cumplimiento de los requisitos.
  + Revisar la modularidad, reutilización y mantenibilidad.
  + Identificar riesgos y proponer mitigaciones.

#### 4. Revisiones de Código y Control de Configuración

* **Entrada:** Código fuente, lineamientos de codificación.
* **Salida:** Informe de revisión de código.
* **Actividades:**
  + Revisar la calidad del código.
  + Verificar conformidad con estándares de codificación.
  + Identificar defectos y mejoras.
  + Asegurar la gestión adecuada de versiones y configuración.

#### 5. Verificación y Validación

* **Entrada:** Casos de prueba, criterios de aceptación.
* **Salida:** Reportes de pruebas y validación.
* **Actividades:**
  + Definir estrategias de prueba.
  + Implementar pruebas unitarias, de integración y sistema.
  + Registrar y analizar resultados.
  + Verificar el cumplimiento de los requisitos.

#### 6. Auditorías y Revisiones de Progreso

* **Entrada:** Reportes de avance, entregables del proyecto.
* **Salida:** Informe de auditoría y cumplimiento.
* **Actividades:**
  + Planificar auditorías periódicas.
  + Evaluar el cumplimiento de estándares y procesos.
  + Documentar hallazgos y acciones correctivas.

#### 7. Gestión de Incidentes y No Conformidades

* **Entrada:** Registro de errores, reporte de usuarios.
* **Salida:** Informe de resolución de incidentes.
* **Actividades:**
  + Establecer un sistema de gestión de errores.
  + Registrar y clasificar defectos.
  + Definir tiempos de respuesta y resolución.
  + Realizar análisis de causa raíz.

#### 8. Publicación y Mantenimiento del SQAP

* **Entrada:** Versión preliminar del SQAP.
* **Salida:** SQAP actualizado y distribuido.
* **Actividades:**
  + Definir mecanismos de distribución.
  + Asegurar la disponibilidad y acceso a la documentación.
  + Implementar un sistema de revisión y mejora continua.

## Responsabilidades

Roles en el sistema SQA

Líder de SQA. Prepara programas de actividades y planes de desarrollo de sistemas SQA para el departamento. Denomina miembros del equipo y miembros del comité de SQA. Representa a la organización ante organismos externos en temas de calidad del software.

Gerente de calidad. Se encarga principalmente del seguimiento del cumplimiento de los procedimientos e instrucciones de trabajo de SQA por parte del departamento. También se encarga de realizar las revisiones de los documentos establecidos en el plan junto con el jefe de SQA.

Especialista en desarrollo SQA. Decide qué estándares de SQA se adoptarán, así como en el desarrollo de los procedimientos de la organización. Se encarga de dar seguimiento de desarrollos y cambios en SQA y estándares de ingeniería de software; introducción de procedimientos adicionales y cambios relevantes para la organización. Reporta fallas severas de calidad del software al gerente de calidad.

Especialista en capacitación SQA. Se encarga de identificar las necesidades de capacitación en SQA de toda la organización y proponer un programa de instrucción o capacitación adecuado y notificarle al líder de SQA.

Roles

Responsable de Administración del Proyecto: Específico será el encargado de la planificación del aseguramiento de la calidad del software. Definirá los objetivos de calidad, identificará los estándares aplicables y asignará los recursos necesarios para garantizar el cumplimiento de los requisitos del software. Asimismo, establecerá los procedimientos para auditorías y revisiones, asegurando que cada rol cumpla con sus responsabilidades dentro del proceso de calidad.

Jefe del departamento: Es el responsable de liderar y gestionar el equipo para garantizar el desarrollo oportuno, eficiente y de alta calidad de software que satisfaga las necesidades y expectativas de la organización y sus clientes.

Analista de Requisitos: se enfocará en las actividades de aseguramiento del producto, identificando los productos de software y la documentación requerida. Será responsable de evaluar la asignación de requisitos y verificar la conformidad de los productos desarrollados con los requisitos establecidos. También revisará la documentación relacionada para garantizar su alineación con los criterios definidos.

Diseñador: tiene un papel clave en la evaluación de la calidad del diseño del software, asegurando que las soluciones propuestas cumplan con los estándares y requisitos establecidos. Colaborará con el responsable de Pruebas para validar la factibilidad de las pruebas y con el Programador para asegurar la correcta implementación de los elementos diseñados.

Diseñador UI: se encargará de evaluar la calidad de la interfaz de usuario, asegurando que las pautas de usabilidad y accesibilidad se cumplan. Además, colaborará con el Analista de Requisitos y el responsable de Manuales para garantizar la coherencia entre la interfaz diseñada y la documentación proporcionada a los usuarios.

Programador: será responsable de implementar las funcionalidades siguiendo los estándares de calidad definidos en el proyecto. También participará en la medición de productos, asegurando que las implementaciones sean consistentes con los requisitos del software y contribuyendo a la identificación de posibles brechas en el cumplimiento de los estándares establecidos.

Responsable de Pruebas: se encargará de la evaluación de procesos del ciclo de vida y la medición de productos. Realizará auditorías periódicas para verificar la conformidad de las actividades de desarrollo con los planes definidos y garantizará que las pruebas ejecutadas reflejen fielmente la calidad del software. También analizará los resultados de las mediciones para identificar áreas de mejora y garantizar que las acciones correctivas sean efectivas.

Responsable de Manuales: supervisará la documentación del software y verificará que refleje con precisión la funcionalidad del sistema. Participará en la evaluación de habilidades y conocimientos del personal, asegurando que los materiales de capacitación sean adecuados y reflejen las necesidades del proyecto. Además, colaborará con el Diseñador UI y el Analista de Requisitos para garantizar que la documentación sea clara, accesible y alineada con la interfaz del software.

Comités

Comité de Verificación y Validación del Software. Es un esfuerzo colaborativo que implica la participación del líder de SQA, el gerente de calidad, el jefe de proyecto y el especialista en métricas de calidad. Cada uno de estos roles contribuye con su experiencia y conocimientos para garantizar que se diseñe un plan integral que aborde todas las áreas críticas del software y cumpla con los estándares de calidad del proyecto.

Comité de Gestión de la Configuración del Software. Es un esfuerzo colaborativo que implica la participación del líder de SQA, el gerente de calidad, el jefe de proyecto y el especialista en herramientas de gestión de la configuración; con la finalidad de definir, supervisar y garantizar la implementación efectiva de políticas, procesos y prácticas relacionadas con la gestión de configuración del software en un proyecto o en toda la organización.

Comité de Desarrollo de Software. Es un esfuerzo colaborativo que implica la participación del líder de SQA, el especialista en desarrollo SQA, el jefe de proyecto y el personal de desarrollo de software. Son los encargados de la supervisión y coordinación de las actividades de desarrollo de software en una organización, asegurando que se cumplan los objetivos estratégicos y operativos relacionados con el desarrollo de software y que se mantengan los estándares de calidad y las mejores prácticas en toda la organización.

|  |  |
| --- | --- |
| **Responsables** | **Tarea** |
| Responsable de Administración del Proyecto, Líder de SQA, Gerente de calidad | Planificación de la SQA |
| Analista de Requisitos, Responsable de Administración del Proyecto, Gerente de calidad | Revisión de Requisitos |
| Diseñador, Programador, Especialista en desarrollo SQA, Comité de Desarrollo de Software | Control de Diseño y Arquitectura |
| Programador, Especialista en desarrollo SQA, Comité de Gestión de la Configuración del Software, Responsable de Pruebas | Revisiones de Código y Control de Configuración |
| Comité de Verificación y Validación del Software, Responsable de Pruebas, Analista de Requisitos | Verificación y Validación |
| Responsable de Administración del Proyecto, Gerente de calidad, Líder de SQA, Comité de Verificación y Validación del Software | Auditorías y Revisiones de Progreso |
| Gerente de calidad, Responsable de Pruebas, Especialista en desarrollo SQA | Gestión de Incidentes y No Conformidades |
| Líder de SQA, Especialista en desarrollo SQA, Responsable de Manuales, Responsable de Administración del Proyecto | Publicación y Mantenimiento del SQAP |

# Documentación

## Propósito

Establecer un medio por el cual sea posible tener un registro y se comunique de manera efectiva aspectos relevantes del desarrollo, uso y funcionamiento del software creado. Se establece una referencia a arquitectos de software y demás miembros del equipo de trabajo que sean participes durante el ciclo de vida del proyecto. Se facilita por este medio la comprensión del software creado para usuarios finales, administradores y cualquier parte interesada del mismo.

## Documento de especificación de Requisitos

El Documento de Especificación de Requisitos de Software constituye una guía detallada que define de manera precisa las funcionalidades y limitaciones del sistema en desarrollo. Su propósito es actuar como un acuerdo formal entre el cliente y el equipo de desarrollo, especificando con claridad las expectativas del sistema a construir.

Para su elaboración, se adopta el estándar IEEE reconocido, garantizando así la coherencia y eficacia en todas las fases del SRS.  
**IEEE Std. 29148-2011** *Systems and software engineering - Life cycle processes - Requirements engineering* establece las mejores prácticas para gestionar los requisitos a lo largo del ciclo de vida del sistema, definiendo terminología clave y detallando los procesos necesarios para recopilar, analizar, validar y mantener dichos requisitos.

Con el objetivo de asegurar la calidad del documento, se recomienda realizar una revisión técnica especializada que permita validar su viabilidad técnica, mejorar el resultado final y optimizar la eficiencia del desarrollo.

## Descripción del diseño de software

El documento SDD tiene como finalidad principal asegurar que la construcción del sistema se realice de forma estructurada y eficiente, sirviendo además como referencia futura para tareas de mantenimiento o evolución del software.

Este documento se basa en un estándar reconocido por la NASA que proporciona lineamientos claros para una documentación coherente del diseño.  
**NASA-GB-8719.13-2004 NASA Software Safety Guidebook**, específicamente el capítulo 7 sobre diseño de software, ofrece directrices para adaptar el proceso de diseño a las necesidades del proyecto. Se enfoca en desarrollar software seguro mediante estrategias como la tolerancia a fallas, reutilización de componentes, restricciones de lenguaje, estándares de codificación y técnicas de programación defensiva.

Para validar su calidad, debe llevarse a cabo una inspección que permita detectar posibles inconsistencias, asegurando que el documento se alinee con normas, especificaciones, procedimientos y estándares aplicables.

## Plan de verificación y validación

Detalla los métodos y acciones que se seguirán para confirmar que el software cumple con los requisitos y se comporta adecuadamente bajo diferentes escenarios.

Se elabora con base en el estándar IEEE, que proporciona una estructura coherente para esta clase de documentos.  
**IEEE Std. 1012-2016** *Standard for System, Software, and Hardware Verification and Validation* describe en detalle cómo establecer un plan de verificación y validación, incluyendo objetivos, alcance, recursos, actividades y criterios de aceptación.

Para asegurar su efectividad, es necesaria una auditoría en proceso que verifique la pertinencia de los métodos definidos y que garantice la recolección completa de datos de evaluación.

## Informe de Resultados de la Verificación

Este informe presenta de forma detallada los resultados obtenidos a partir de las tareas de verificación aplicadas a un sistema, software, hardware o proceso. La verificación se centra en comprobar que el producto cumple con los requisitos establecidos durante su diseño y desarrollo.

Su redacción se guía por el estándar IEEE correspondiente.  
**IEEE Std. 1012-2016** ofrece una metodología estructurada para llevar a cabo la verificación de productos tecnológicos a lo largo de su ciclo de vida. Incluye desde la planificación hasta la documentación, resaltando la importancia de la trazabilidad, transparencia e integridad de los resultados.

Se requiere una inspección formal que confirme el cumplimiento de los requisitos y asegure la detección temprana de fallos, contribuyendo a la calidad del producto final.

### **Informe de Resultados de Validación**

El Informe de Validación resume los resultados obtenidos tras evaluar si el sistema cumple con los requisitos especificados y si está listo para ser aceptado. La validación se realiza generalmente al final del ciclo de desarrollo.

Este documento se basa en el mismo estándar IEEE que el utilizado para la verificación.  
 **IEEE Std. 1012-2016** proporciona una base sólida para presentar los hallazgos de validación, estableciendo actividades específicas, criterios y tareas relacionadas con la elaboración de resultados de pruebas.

Para asegurar la integridad del documento, se recomienda una inspección que valide la conformidad con las especificaciones y facilite la identificación de errores desde etapas tempranas.

### **Documentación de Usuario**

La documentación de usuario contiene información clave sobre el uso correcto del software, facilitando al usuario final una experiencia efectiva y funcional con el producto.

Para su desarrollo, se sigue un estándar IEEE dedicado a este tipo de documentación.  
 **IEEE Std. 1063-2001** *Standard for Software User Documentation* define los componentes mínimos necesarios para redactar manuales de usuario de alta calidad, incluyendo un marco estructurado con descripciones claras.

La calidad se asegura mediante una revisión técnica especializada, orientada a evaluar la utilidad y comprensión del documento desde la perspectiva del usuario final.

### **Plan de Gestión de Configuración de Software (SCMP)**

Este plan establece una estructura clara para controlar de forma sistemática la configuración del software, minimizando riesgos como la pérdida de información o cambios no autorizados.

Se apoya en un estándar IEEE que ofrece lineamientos para la gestión del ciclo de vida de los elementos de configuración.  
 **IEEE Std. 1042-1987** *Guide to Software Configuration Management* proporciona un conjunto de prácticas para organizar, controlar y seguir el rastro de los artefactos del software.

Se recomienda una revisión de gestión para validar la eficacia e integridad del proceso de configuración a lo largo del proyecto.

### **Plan de Desarrollo de Software (SDP)**

Este plan detalla los pasos y estrategias a seguir en el ciclo de vida del desarrollo del software, asegurando que todos los miembros del equipo comprendan sus roles, responsabilidades y los procesos involucrados.

Se emplea el estándar conjunto de ISO/IEC para su elaboración.  
 **ISO/IEC 12207-2008** *Software life cycle processes* ofrece un enfoque integral para gestionar desde la definición de requisitos hasta la eliminación del sistema, permitiendo adaptaciones según el tamaño o criticidad del proyecto.

Para garantizar su calidad, se recomienda realizar una auditoría en proceso que permita detectar y corregir desviaciones antes de impactar negativamente el desarrollo.

### **Manual de Mantenimiento de Software (SMM)**

Este manual proporciona una guía sobre cómo gestionar el software después de su entrega inicial, incluyendo actividades de mantenimiento correctivo, adaptativo y evolutivo.

Se basa en el estándar IEEE orientado a la gestión del mantenimiento.  
 **IEEE Std. 1219-1998** *Standard for Software Maintenance* define un marco sólido para planificar, documentar y evaluar las acciones de mantenimiento.

Para validar su utilidad, debe someterse a una revisión técnica con el objetivo de asegurar la claridad del contenido y su aplicabilidad por parte del personal técnico o usuario final.

### **Especificaciones de la Interfaz Externa**

Este documento describe detalladamente cómo el sistema interactúa con componentes o sistemas externos, definiendo protocolos de comunicación, datos compartidos y formatos esperados.

Se guía por una fuente reconocida de la NASA, además de una plantilla práctica para su estructuración.  
 **Guía de Ingeniería de Sistemas de la NASA** (capítulo 6.3 “Gestión de Interfaz”) proporciona el marco conceptual, mientras que la **plantilla Smartsheet (IC-IE-Technical-Specification-Template\_ES)** ofrece una estructura organizacional adecuada para la documentación técnica.

Una inspección garantiza el cumplimiento de los requisitos definidos, detectando errores y asegurando que se alcancen los atributos de calidad establecidos.

### **Especificaciones de la Interfaz Interna**

Este documento detalla la interacción entre los diferentes módulos internos del sistema, asegurando una correcta integración funcional entre sus componentes.

Se apoya también en la guía de la NASA y la plantilla Smartsheet.  
 **Guía de Ingeniería de Sistemas de la NASA** (capítulo 6.3) sirve como referencia para identificar requisitos de interacción interna.  
 La **plantilla Smartsheet (IC-II-Technical-Specification-Template\_ES)** organiza eficazmente los elementos clave para documentar estas interacciones.

Su calidad se valida mediante inspecciones enfocadas en verificar la conformidad con regulaciones y especificaciones, además de registrar los errores detectados.

### **Plan de Métricas de Software**

Este documento define los indicadores clave que permitirán medir la calidad, rendimiento y eficiencia del producto o proceso de desarrollo.

Se utiliza el estándar IEEE correspondiente.  
 **IEEE Std. 1061-1998** *Standard for a Software Quality Metrics Methodology* proporciona una metodología para establecer, aplicar y evaluar métricas de calidad del software en cualquier etapa del ciclo de vida.

La calidad se garantiza mediante una revisión técnica que permita evaluar la idoneidad del producto con base en las métricas definidas.

### **Plan de Seguridad del Software**

El plan de seguridad establece las políticas y controles necesarios para proteger el software contra amenazas, asegurando la confidencialidad, integridad y disponibilidad del sistema.

Se elabora siguiendo el estándar de la NASA.  
 **NASA-GB-8719.13-2004**, capítulo 3, define los procesos clave para integrar la seguridad en todas las fases del desarrollo, incluyendo la gestión de riesgos y la aplicación de controles específicos.

Una revisión técnica permitirá validar la consistencia del plan, identificando posibles desviaciones frente a los lineamientos establecidos.

# Estándares, practicas, convenciones y métricas.

## Propósito

# Garantizar la calidad, consistencia, eficiencia y productividad del proceso de desarrollo en las fases de diseño, implementación y pruebas.

## Fase de requerimientos

# IEEE Std. 29148-2011 Systems and software engineering - Life cycle processes - Requirements engineering . Este estándar se aplica para ayudar a los ingenieros de software a expresar claramente los requisitos del sistema, reduciendo la ambigüedad y la confusión durante el proceso. Es fundamental seguir este estándar porque se aplica a los procesos utilizados en la obtención, el análisis, la asignación, la documentación y la gestión de los requisitos, asegurando la calidad, coherencia y trazabilidad del documento. Las métricas que se evalúan de acuerdo con los procesos del estándar son la claridad y comprensión de los requisitos, la completitud de los requisitos determinando si el documento captura todos los requisitos necesarios para el proyecto, la consistencia para verificar que no haya conflictos o contradicciones entre los diferentes requisitos especificados en el documento y la clasificación de los requisitos según su importancia y criticidad para el éxito del proyecto.

## Control de Medios

# ISO/IEC 27002:2022 Information security, cybersecurity and privacy protection - Information security controls. Este estándar se aplica para garantizar la integridad y confidencialidad de los medios, al seguir las pautas proporcionadas por esta norma, se implementan políticas claras, se capacita al personal en seguridad de la información, se establecen medidas físicas de seguridad y se utilizan tecnologías adecuadas para proteger los medios de almacenamiento. Es de suma importancia seguir este estándar, ya que proporciona directrices y mejores prácticas para establecer, implementar, mantener y mejorar la seguridad de la información en una organización.

## Fase de diseño

# NASA-GB-8719.13-2004 NASA Software Safety Guidebook. Este estándar se aplica para hacer diversos análisis de diseño, que incluyen actualizaciones de análisis previos, análisis de seguridad, análisis de independencia, inspecciones formales, métodos formales de verificación, análisis lógico de diseño, análisis de datos de diseño, análisis de trazabilidad, análisis de elementos de software, y análisis de requisitos de máquinas de estado. Es fundamental seguir este estándar porque garantizan la seguridad, fiabilidad, mantenibilidad y cumplimiento de los requisitos del software. Las métricas mencionadas en el estándar sirven para evaluar un diseño en busca de cualidades que mejoren la capacidad del software para mantenerse durante un largo período de tiempo. Muchas de las métricas proporcionan una forma de evaluar partes del diseño y darle un valor numérico.

# IEEE Std. 1042-1987 Guide to Software Configuration Management. Este estándar se aplica para establecer prácticas para la gestión de la liberación del software, incluyendo la planificación, preparación y entrega de versiones del software, así como definir roles y responsabilidades dentro del plan, asegurando que las actividades de gestión de configuración sean llevadas a cabo por personal competente. Es fundamental seguir este estándar porque contribuye a la calidad del software al reducir la posibilidad de errores y problemas relacionados con la configuración incorrecta o inconsistente.

# Guía ampliada para ingeniería de sistemas de la NASA. Esta guía se aplica para ayudar a controlar el desarrollo de productos cuando los esfuerzos se dividen entre las partes involucradas (el gobierno, los contratistas, los equipos técnicos geográficamente diversos, etc.), así como para definir y mantener el cumplimiento entre los productos que deben interoperar. Es fundamental seguir esta guía porque proporciona los principales documentos útiles para el desarrollo de especificaciones de interfaz externa, lo que contribuye a garantizar la interoperabilidad y el éxito del sistema en su entorno operativo. Las métricas mencionadas en el estándar deben evaluarse para determinar su posible impacto en otros elementos de interfaz y comunicarse con los desarrolladores afectados.

# IEEE Std. 1061-1998 Standard for a Software Quality Metrics Methodology. Este estándar se aplica para identificar, definir y utilizar métricas de calidad del software de manera efectiva. Es fundamental seguir este estándar porque proporciona una metodología para desarrollar un conjunto de métricas de calidad del software que sean relevantes, confiables y significativas para un proyecto o una organización específica.

# NASA-GB-8719.13-2004 NASA Software Safety Guidebook. Este estándar se aplica para planificar el esfuerzo de seguridad del software y cómo adaptarlo de acuerdo con el nivel de riesgo del sistema. Es fundamental seguir este estándar porque asigna un índice de riesgo de acuerdo con el nivel de riesgo identificado, para poder estimar la cantidad de esfuerzo que se debe realizar para garantizar la seguridad del software. Las métricas utilizadas que menciona el estándar sirven para determinar el índice de riesgo de software. El nivel de riesgo del software determinará el alcance del esfuerzo de seguridad del software. Este ejercicio deberá perfeccionarse a medida que la arquitectura de diseño y la implementación revelen cómo se modulariza o no la funcionalidad.

## Fase de implementación

### Documentación de usuario.

# IEEE Std. 1063-2001 Standard for Software User Documentation. Es un estándar internacional que establece un marco de referencia con respecto a los componentes necesarios para la realización de cualquier manual o documentación de usuario de la mejor calidad. Este estándar es de suma importancia debido a que establece las secciones a documentar con respecto a los elementos clave del sistema como comandos, tutoriales o hasta los diferentes errores que podrían surgir durante el uso del producto para aumentar la usabilidad en cuanto al producto y con esto la satisfacción del usuario final.

### Plan de desarrollo de software.

# ISO-IEC 12207-2008. Systems and software engineering — Software life cycle processes. Es un estándar que se aplica para proporcionar un modelo de ciclo de vida del software y los procesos relacionados. Este estándar es de suma importancia en el modelo debido a que proporciona un marco de referencia con respecto a las fases de ciclo de vida de software, además de las diferentes actividades que se deben realizar en cada etapa y los resultados esperados. Por otro lado, se utiliza debido a que es un estándar que cuenta con una gran flexibilidad en cuanto a factores como el tamaño de la organización o las tecnologías a utilizar.

## Fase de pruebas

### Plan de Verificación y Validación.

# IEEE Std. 1012-2016, Standard for System, Software, and Hardware Verification and Validation. Este estándar se aplica para proporcionar pautas que permitan determinar si los productos de desarrollo de una actividad determinada se ajustan a los requisitos de esa actividad y si el producto satisface el uso previsto y las necesidades del usuario. Esta determinación puede incluir el análisis, evaluación, revisión, inspección, valoración y prueba de productos y procesos.

# ISO/IEC 29119-2 Software and systems engineering — Software testing — Part 2: Test processes. Este estándar se aplica para especificar los procesos de prueba que se pueden utilizar para gobernar, gestionar e implementar pruebas de software. Es fundamental seguir este estándar porque contribuye a la mejora de la calidad del producto y la eficacia del proceso de pruebas en proyectos de desarrollo de software.

# ISO/IEC 29119-3 Software and systems engineering — Software testing — Part 3: Test documentation. Este estándar se aplica para asegurar la disponibilidad de una documentación exhaustiva y precisa de todas las actividades realizadas durante las pruebas. Es fundamental seguir este estándar para garantizar una trazabilidad adecuada y transparencia en todas las fases del proceso de prueba.

# Revisiones y auditorías

## Propósito

El propósito de esta sección es definir las revisiones técnicas y de gestión, así como las auditorías que se llevarán a cabo durante el ciclo de vida del software， con el objetivo de evaluar el progreso, la calidad técnica del trabajo realizado y su conformidad con los requisitos y estándares establecidos.

Las revisiones y auditorías se realizarán de manera planificada al término de cada sprint, siguiendo procedimientos formales basados en el estándar IEEE Std 1028-2008. Estas actividades permitirán identificar deficiencias técnicas, asegurar la integridad del diseño, mejorar los métodos de aseguramiento de calidad, y garantizar que se cumplan los plazos y presupuestos definidos para el proyecto.

Esta sección también describe cómo se ejecutarán las revisiones y auditorías, quiénes serán los participantes responsables (desarrolladores, testers, product managers y administradores), los formatos de reporte a utilizar, los destinatarios de dichos reportes, y las acciones de seguimiento necesarias para corregir las no conformidades detectadas, incluyendo los responsables de implementar y verificar dichas acciones.

## Revisión de requisitos de software (SRR)

La Revisión de Requisitos de Software (SRR) tiene como objetivo es evaluar la adecuación, factibilidad técnica y completitud de los requisitos definidos en el documento de Especificación de Requisitos de Software (SRS).  
Esta revisión garantiza que los requisitos sean comprensibles, verificables, consistentes, modificables, rastreables y utilizables durante las fases de operación y mantenimiento, de acuerdo con las características descritas en el estándar IEEE Std 830-1993.

Durante la SRR se verifica:

* Que todos los requisitos funcionales, de rendimiento, de interfaz, de seguridad y humanos estén correctamente especificados.
* La consistencia y factibilidad técnica de los requisitos planteados.
* La trazabilidad completa entre los requisitos del sistema y los requisitos de software, generalmente mediante un Matriz de Trazabilidad de Verificación de Requisitos (RVTM).
* Que existan criterios claros de aceptación y verificación para cada requisito.
* La SRR es organizada por el área de aseguramiento de calidad de software (SQA), y participan representantes de ingeniería de sistemas, desarrollo de software, pruebas, clientes y usuarios.  
   Al concluir, se genera un informe de SRR que documenta todas las deficiencias detectadas, además de un plan y calendario para corregirlas. El SRS revisado pasa a control de configuración como base para las fases siguientes

### Resultados de la SRR:

* Se deberá generar un informe de la SRR que identifique las deficiencias detectadas, proponga acciones correctivas específicas, y establezca un plan y cronograma de ejecución.
* Basado en los resultados, se tomará una decisión sobre la viabilidad de continuar con las siguientes fases del proyecto, considerando estimaciones de costos, estudios de factibilidad y evaluaciones de riesgos.
* Una vez corregidas las deficiencias, el SRS será puesto bajo control de configuración para establecer la línea base que se usará durante el diseño del software.
* A partir del SRS aprobado, el área de SQA desarrollará una Matriz de Trazabilidad de Verificación de Requisitos (RVTM) que se utilizará para validar la correcta satisfacción de los requisitos en el diseño, el código, las pruebas y las funcionalidades finales del sistema.

## Revisión preliminar de diseño (PDR)

La Revisión de Diseño Preliminar (PDR) constituye un paso crucial en la fase de diseño, llevándose a cabo para evaluar la suficiencia técnica del diseño preliminar del software, incluyendo su arquitectura general, antes de iniciar el diseño detallado. Esta revisión permite valorar el progreso, la consistencia y la viabilidad técnica del enfoque de diseño seleccionado, asegurando su compatibilidad con los requisitos funcionales y de desempeño establecidos en el Documento de Especificación de Requisitos de Software (SRS).

Además, la PDR verifica la compatibilidad e integración de las interfaces entre el software, el hardware y los usuarios finales, asegurando que el diseño preliminar cumpla adecuadamente los requisitos definidos en el SRS. Durante esta revisión, también se actualiza la Matriz de Trazabilidad de Verificación de Requisitos (RVTM) para mapear el diseño preliminar a los requisitos y verificarlo mediante el área organizacional de SQA.

En la PDR se analiza:

* La adecuación técnica del Preliminary Software Design Description (SDD).
* La asignación de funciones a módulos de software, flujos funcionales, bases de datos y necesidades de almacenamiento.
* La correcta identificación de todas las interfaces del sistema (con hardware, otro software, usuarios, redes de comunicación, etc.).
* El cumplimiento de factores humanos y usabilidad en el diseño preliminar.
* La capacidad del diseño para ser probado, evaluando si es posible determinar el comportamiento del sistema a través de pruebas planeadas.
* Aspectos de desempeño como tiempos de ejecución, ausencia de condiciones de carrera y estados de bloqueo (deadlocks).

### Resultados de la PDR:

* Los hallazgos deberán ser documentados en un informe de PDR, donde se identificarán las deficiencias detectadas, así como un plan y un cronograma para su corrección.
* El documento de diseño preliminar de software (Preliminary SDD) actualizado será colocado bajo control de configuración para establecer la línea base que se utilizará en el esfuerzo de diseño detallado.

## Revisión crítica del diseño (CDR)

La CDR es una revisión a fondo del diseño detallado del software. Aquí debe de definirse: las funciones, flujo de datos, interacción con hardware o con otras piezas de software. Esta revisión se hace antes de empezar a programar formalmente (aunque sí se permite que haya algunos prototipos o pruebas de concepto si fueron necesarias).

Durante la CDR se evalúan los siguientes puntos:

* El diseño debe estar correctamente vinculado a los requisitos del SRS.
* Los Algoritmos, estructuras de datos, módulos e interfaces deben estar bien definidos y completos.
* Se revisa si el diseño es factible de implementar con las técnicas y herramientas seleccionadas.
* Se debe verificar la claridad y compatibilidad de las interfaces externas e internas.
* El diseño debe cumplir requisitos de rendimiento, seguridad, confiabilidad y mantenibilidad.
* Se evalúa si los recursos (hardware, memoria, procesamiento) son suficientes para el diseño propuesto.
* Se revisan las estrategias de recuperación de fallos y tolerancia a errores.
* Debe existir un plan claro de pruebas unitarias, de integración y de sistema basado en el diseño.
* Confirmar que todos los elementos del diseño están cubiertos en la matriz de trazabilidad de requisitos.

### Resultados de la CDR:

* Se generará un informe de la CDR que identifique deficiencias encontradas en el diseño detallado, proponiendo acciones correctivas específicas y estableciendo un plan y cronograma para su implementación.
* Basado en los resultados, se tomará la decisión de autorizar el inicio formal de la codificación y pruebas.
* El diseño detallado corregido será puesto bajo control de configuración, estableciendo la línea base (baseline) para el desarrollo de software.

## Revisión del plan de verificación y validación de software (SVVP)

La SVVPR es una revisión formal para evaluar si el Plan de Verificación y Validación de Software (SVVP) describe métodos adecuados y completos para garantizar que el software cumpla todos los requisitos. Dado que el SVVP puede desarrollarse en etapas, pueden realizarse varias revisiones.

Durante la SVVPR, se verifica:

* Que los métodos de verificación y validación sean trazables a los requisitos funcionales y de desempeño del SRS.
* Que los informes documentan correctamente resultados de revisiones, auditorías y pruebas.
* Que la configuración del software a probar esté claramente definida, incluyendo herramientas de prueba.
* Que los planes de prueba cubran todos los requisitos y que los procedimientos de prueba sean claros y completos.
* Que exista un calendario de pruebas detallado (qué pruebas, cuándo y por quién).
* Que se validen los requisitos en la Matriz de Trazabilidad de Requisitos (RVTM) actualizada.

Si el SVVP fue desarrollado de manera incremental, también se revisa que las deficiencias de versiones anteriores hayan sido corregidas. El área de SQA lidera esta revisión, con participación de ingeniería de sistemas, desarrollo, pruebas, clientes y usuarios.

Finalmente, los resultados se documentan en un informe de SVVPR, identificando deficiencias y un plan de acciones correctivas. El SVVP actualizado se pone bajo control de configuración como base para todas las actividades de verificación y validación.

### Resultados de la SVVPR:

* Se actualizan el reporte de revisión SVVPR, documentando todas las deficiencias detectadas en el plan de verificación y validación.
* Se establece qué correcciones se deben de hacer y su tiempo requerido para completarse.
* Se actualiza la matriz de trazabilidad de requisitos para mostrar las pruebas necesarias en relación con el diseño, el código y los requisitos del sistema.

## Auditorías funcionales

Estas comparan el software desarrollado (incluyendo ejecutables y documentación disponible) con los requisitos establecidos en el documento de Especificación de requisitos base.  
Su objetivo es asegurar que el código implementa todos y solo los requisitos funcionales definidos en el SRS. Estas auditorías se realizan con base en el estándar **IEEE Std. 1028-2008 Standard for Software Reviews and Audits,** el cual da una base y pautas a seguir para la realización de diferentes tipos de auditorías.

## Auditorías físicas

Se centra en la revisión y evaluación de los aspectos tangibles del software, como la infraestructura física en la que se ejecuta el software y los recursos de hardware asociados tales como la infraestructura, ambiente, seguridad, entre otros elementos. Estas auditorías garantizarán su integridad y exactitud basándose en el estándar I**EEE Std. 1028-2008 Standard for Software Reviews and Audits.**

## Auditorías en proceso

Estas auditorías se centran en revisar y evaluar los procesos y actividades que se llevan a cabo durante el ciclo de vida del desarrollo del software con el afán de identificar y corregir posibles problemas o desviaciones antes de que afecten la calidad del producto final. Estas auditorías se basarán en el estándar **IEEE Std. 1028-2008 Standard for Software Reviews and Audits** para garantizar la calidad, eficiencia e integridad de las actividades realizadas durante la auditoría.

## Revisiones gerenciales

Las revisiones gerenciales son evaluaciones periódicas cuyo objetivo es supervisar la correcta ejecución del Plan de Aseguramiento de la Calidad del Software (SQAP). Permiten a la dirección independiente analizar el avance del proyecto, verificar si los planes y cronogramas se están cumpliendo, y evaluar si los métodos de gestión aplicados son efectivos para alcanzar los objetivos establecidos. Estas revisiones también sirven para apoyar decisiones sobre correcciones necesarias, reasignaciones de recursos o ajustes en el alcance del proyecto. Están guiadas por el estándar **IEEE Std. 1028-2008 Standard for Software Reviews and Audits** que proporciona lineamientos para asegurar que estas evaluaciones sean sistemáticas, completas y efectivas.

Se analiza que lo planeado y lo ejecutado sean iguales, también, se detectan desviaciones, se evalúan la adecuación y la efectividad de los procedimientos de gestión establecidos, así como identificar áreas que requieran mejoras

### Resultados de las revisiones gerenciales:

Se genera un informe que resume los hallazgos, señala cualquier excepción encontrada respecto a lo que establece el SQAP y recomienda cambios o mejoras necesarias para fortalecer la ejecución del proyecto.

## Revisión del Plan de Gestión de la Configuración del Software (SCMPR)

La Revisión del Plan de Gestión de Configuración del Software (SCMPR) consiste en evaluar si el Plan de Gestión de Configuración de Software (SCMP) es adecuado para garantizar el control y la trazabilidad de los productos de software y su documentación. Esta revisión está basada en el estándar **IEEE Std. 828-2012 Standard for Configuration Management in Systems and Software Engineering,** el cual define los procesos básicos de la gestión de configuración, como la identificación de la configuración, el control de cambios, la auditoría de la configuración y la gestión de versiones y entregas.

Se revisan los siguientes aspectos: la correcta planificación de las actividades de gestión de configuración, la claridad en la asignación de responsabilidades, la identificación de todos los elementos de configuración, el control de los cambios, la disponibilidad de reportes de estado, la aplicación de auditorías, y la relación de la gestión de configuración con otras disciplinas como ingeniería de sistemas o desarrollo de software.

### Resultados de la revisión del plan de gestión de la configuración del software:

Como producto de esta revisión se genera un informe que valida que los métodos y procedimientos del SCMP sean completos y aplicables. También se documentan las excepciones encontradas, las mejoras recomendadas y se da seguimiento a la implementación de los cambios para asegurar una correcta gestión de configuración en el proyecto.

## Revisión posterior a la implementación

La revisión posterior a la implementación tiene como finalidad inspeccionar los productos de software ya terminados para detectar anomalías y asegurar que se cumplan los requisitos de calidad definidos en las etapas iniciales del proyecto. Esta revisión está basada en el estándar **IEEE Std. 1028-2008 Standard for Software Reviews and Audits**, que establece un procedimiento estructurado para realizar inspecciones de software y documentar sus resultados.

En este proceso se verifican los productos finales del software, comprobando la presencia de errores, defectos o incumplimientos respecto a los estándares de calidad previstos. Se revisan las entradas, los procedimientos de inspección, los roles y responsabilidades de los participantes, así como las conclusiones y salidas que debe generar la revisión.

### Resultados de la revisión posterior a la implementación

Se espera identificar los defectos menores que quedan del software y su clasificación, así como la validación de que los estándares fueron seguidos de manera correcta, asimismo se espera que existan propuesta de acciones correctivas o mejoras para las próximas versiones.

## Revisión de Documentación para el Usuario (UDR)

La Revisión de Documentación para el Usuario (UDR) tiene como finalidad evaluar la adecuación técnica del enfoque y diseño de la documentación para el usuario, basada en versiones preliminares (en formato digital o impreso). Esta revisión garantiza que la documentación sea clara, precisa y esté alineada con el funcionamiento real del software. La UDR se basa en el estándar IEEE Std. 1028-2008 Standard for Software Reviews and Audits que establece las prácticas para revisiones y auditorías de software.

Durante la UDR se revisan:

* Los métodos utilizados para validar que el producto de software coincide con la documentación de usuario.
* Los planes, procedimientos y casos de prueba que aseguran que los manuales de usuario y operador se prueben junto con el software.
* El cumplimiento de la documentación de usuario con los estándares de documentación aprobados en el proyecto.

### Resultados de la revisión de documentación para el usuario:

Como resultado se debe de obtener un informe de UDR donde se documentan:

* Deficiencias encontradas
* Plan y programa para corregir las deficiencias encontradas

## Auditoria de aseguramiento de calidad

La Auditoría de Aseguramiento de Calidad (QA Audit) tiene como propósito evaluar la adecuación del Plan de Aseguramiento de Calidad del Software (SQAP) y los procedimientos asociados. Esta auditoría puede ser solicitada por la gerencia, independientemente de si el área de SQA es autónoma o forma parte del grupo de desarrollo. Se utiliza principalmente para garantizar la independencia, objetividad y efectividad del aseguramiento de calidad.

En estas auditorías se revisan:

* El contenido y la adecuación del SQAP.
* La correcta aplicación de los procedimientos de aseguramiento de calidad definidos.
* La necesidad de realizar acciones correctivas en caso de encontrar deficiencias.

# Pruebas

El propósito de la sección de pruebas es definir los criterios, actividades, responsabilidades y procedimientos relacionados con la verificación y validación del software, con el fin de asegurar que los productos desarrollados cumplan con los requisitos especificados y los estándares de calidad establecidos. Este apartado busca garantizar que todas las funciones del software sean probadas de manera sistemática, que se detecten errores en etapas tempranas y que se documente formalmente la conformidad del software con los objetivos del proyecto.

## Planificación de pruebas

### Preparación de entorno

Se definirán y configurarán los entornos necesarios para la ejecución de las pruebas, asegurando que repliquen adecuadamente las condiciones de operación esperadas en producción. Esto incluye la instalación de software, bases de datos, herramientas de prueba y cualquier otro recurso necesario.

### Secuencia de ejecución

Se establecerá el orden en que se ejecutarán las pruebas, considerando dependencias entre módulos, componentes o sistemas, y priorizando pruebas críticas para el negocio o para la estabilidad del sistema.

Para estructurar adecuadamente estas actividades de planificación y ejecución, se seguirá el marco definido en el estándar **ISO/IEC 29119-2 Software and systems engineering - Software testing - Part 2: Test processes**.

## Tipos de pruebas

## Pruebas a nivel unidad

Las pruebas unitarias consisten en evaluar individualmente cada módulo de un conjunto de módulos de un programa de software. Su propósito es revisar que cada módulo cumpla correctamente con su funcionalidad prevista de manera aislada, antes de integrarlo con otros módulos o componentes del sistema.  
Para la ejecución de las pruebas unitarias se utiliza el estándar ISO/IEC 29119-2 Software and systems engineering - Software testing - Part 2: Test processes, el cual proporciona un marco estructurado para llevar a cabo pruebas de software de forma organizada y controlada. Su aplicación contribuye tanto a mejorar la calidad del software como a incrementar la eficiencia de los procesos de prueba en los proyectos de desarrollo.  
 Además, para garantizar que las actividades de las pruebas unitarias queden documentadas de manera completa y precisa, se adopta el estándar ISO/IEC 29119-3, que se enfoca en la documentación del proceso de pruebas de software. El cumplimiento de este estándar asegura una adecuada trazabilidad y transparencia en todas las fases de las pruebas, permitiendo una mejor colaboración entre todos los participantes del proyecto.

## Pruebas de integración

Estas pruebas consisten en combinar y probar diferentes componentes de software, hardware o una mezcla de ambos, con el fin de evaluar cómo interactúan entre sí. Este término se refiere tanto a la integración de componentes individuales como a la integración de sistemas completos.

El estándar ISO/IEC 29119-2 Software and systems engineering - Software testing - Part 2: Test processes será utilizado para las pruebas de integración. Este estándar proporciona un marco estructurado para realizar pruebas de software de manera ordenada y controlada, lo que contribuye a mejorar la calidad del software y la eficiencia del proceso de pruebas en proyectos de desarrollo.

Además, para asegurar que las actividades realizadas durante las pruebas de integración se documenten correctamente, se hará uso del estándar ISO/IEC 29119-3 Software and systems engineering - Software testing - Part 3: Test documentation. Este estándar se enfoca en la correcta documentación de todo el proceso de pruebas, asegurando la trazabilidad y la transparencia en cada una de sus etapas. Siguiendo estas directrices, se logrará mantener un registro preciso y accesible de todas las pruebas realizadas, lo que facilitará la colaboración entre los miembros del equipo involucrados en el proyecto.

## Pruebas a nivel sistema

Las pruebas a nivel sistema se realizan para validar el comportamiento completo e integrado de un sistema de software, asegurando que cumpla con los requisitos especificados. Estas pruebas consideran al sistema como un todo, verificando tanto la funcionalidad como otros atributos de calidad, como la seguridad, el desempeño y la usabilidad.

Para llevar a cabo estas pruebas, se aplicará el estándar ISO/IEC 29119-2 Software and systems engineering - Software testing - Part 2: Test processes. Este estándar proporciona una estructura organizada para realizar pruebas de software de manera sistemática y controlada, contribuyendo significativamente a la calidad del producto y a la eficiencia del proceso de pruebas dentro del proyecto.

Asimismo, para asegurar una documentación clara, completa y precisa de todas las actividades realizadas durante las pruebas a nivel sistema, se seguirá el estándar ISO/IEC 29119-3 Software and systems engineering - Software testing - Part 3: Test documentation. Este estándar garantiza la adecuada trazabilidad y transparencia en todas las fases del proceso de prueba, permitiendo mantener un registro detallado que facilite la comunicación y colaboración entre los distintos participantes del proyecto.

## Pruebas de estrés

Estas pruebas son de desempeño que se llevan a cabo para evaluar el comportamiento de un componente de software cuando se somete a condiciones de carga superiores a las que se anticiparon en los requisitos de capacidad, o cuando los recursos disponibles son menores a los requeridos para el funcionamiento óptimo.

Para realizar las pruebas de estrés, se empleará el estándar ISO/IEC 29119-2 Software and systems engineering - Software testing - Part 2: Test processes. Este estándar ofrece un marco organizado para ejecutar pruebas de software de manera eficiente y controlada, lo que favorece tanto la calidad del software como la mejora de la eficacia de las actividades de pruebas en proyectos de desarrollo.

Con el fin de garantizar que las actividades durante las pruebas de estrés estén adecuadamente documentadas, se utilizará el estándar ISO/IEC 29119-3 Software and systems engineering - Software testing - Part 3: Test documentation. Este estándar tiene como objetivo asegurar que todas las etapas del proceso de pruebas sean transparentes y trazables. Siguiendo estas pautas, se logrará mantener una documentación precisa y comprensible de las pruebas realizadas, facilitando la colaboración entre todos los miembros del equipo involucrado en el proyecto.

## Pruebas de seguridad

Estas pruebas tienen como objetivo identificar vulnerabilidades, debilidades y riesgos potenciales en el software, garantizando que el sistema proteja adecuadamente los datos y recursos frente a accesos no autorizados o maliciosos. Estas pruebas evalúan aspectos como la confidencialidad, integridad, autenticación, autorización y disponibilidad del sistema.

Para la realización de estas pruebas, se utilizará el estándar ISO/IEC 29119-2 Software and systems engineering - Software testing - Part 2: Test processes, que ofrece un marco estructurado para llevar a cabo las pruebas de software de manera sistemática y eficiente, promoviendo la mejora continua de la calidad y seguridad del producto.

Además, para asegurar que la documentación de las actividades de prueba sea completa, precisa y trazable, se seguirá el estándar ISO/IEC 29119-3 Software and systems engineering - Software testing - Part 3: Test documentation. Este enfoque garantiza que todas las pruebas de seguridad queden debidamente registradas y que la información esté disponible para su análisis y revisión, favoreciendo la colaboración de todos los actores involucrados en el proyecto.

## Pruebas de aceptación

En este tipo de pruebas se enfocan en verificar que el software cumple con los requisitos y expectativas establecidos por el cliente o usuario final. Estas pruebas son fundamentales para determinar si el producto está listo para su liberación y despliegue, validando tanto la funcionalidad como el cumplimiento de los criterios de calidad acordados.

Para llevar a cabo las pruebas de aceptación, se empleará el estándar **ISO/IEC 29119-2 Software and systems engineering - Software testing - Part 2: Test processes**, el cual proporciona un enfoque organizado y sistemático para la ejecución de pruebas de software, ayudando a asegurar que el proceso de validación sea riguroso y eficiente.

Asimismo, para garantizar una documentación adecuada y detallada de todas las actividades realizadas durante las pruebas de aceptación, se aplicará el estándar **ISO/IEC 29119-3 Software and systems engineering - Software testing - Part 3: Test documentation**. Este estándar facilita una trazabilidad clara y una comunicación efectiva de los resultados de las pruebas, asegurando que toda la información relevante esté disponible para los interesados en el proyecto.

### Pruebas de caja blanca y caja negra

Las pruebas de caja negra son aquellas en las que los casos de prueba se desarrollan exclusivamente a partir de la especificación del elemento que se va a probar, sin necesidad de analizar el código o el diseño interno del software. En cambio, las pruebas de caja blanca se realizan teniendo en cuenta la estructura interna del software, con el objetivo de verificar la cobertura total del código. Estas pruebas suelen requerir el uso de herramientas de software especializadas para el soporte de las pruebas.

El estándar **ISO/IEC 29119-2 Software and systems engineering - Software testing - Part 2: Test processes** es utilizado tanto para las pruebas de caja negra como para las de caja blanca. Este estándar proporciona un marco estructurado para realizar las pruebas de software de forma sistemática y controlada, contribuyendo a mejorar tanto la calidad del software como la eficacia de los procesos de prueba durante el desarrollo del software.

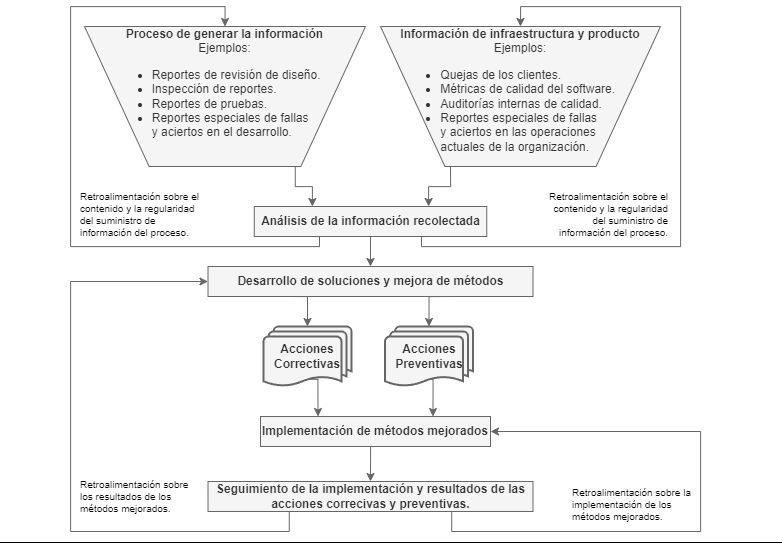
Adicionalmente, para garantizar que las actividades realizadas durante las pruebas de caja negra y caja blanca estén debidamente documentadas, se emplea el estándar **ISO/IEC 29119-3 Software and systems engineering - Software testing - Part 3: Test documentation**. Este estándar está enfocado en la correcta documentación del proceso de pruebas, asegurando que se mantenga una trazabilidad adecuada y una completa transparencia en cada fase del proceso. Al seguir este estándar, se consigue mantener un registro claro y accesible de todas las pruebas realizadas, lo que facilita la colaboración entre todos los miembros involucrados en el proyecto.

**Reporte de problemas y acciones correctivas**

Para la elaboración de este informe nos basamos en el enfoque CAPA (Acciones Correctivas y Preventivas), el cual se emplea como una guía estructurada para detectar, analizar y resolver las causas principales de problemas o no conformidades. Además, este método permite implementar medidas que ayuden a evitar que dichas situaciones se repitan en el futuro.

El modelo CAPA se desarrolla en un ciclo iterativo compuesto por cinco etapas fundamentales que permiten abordar los problemas desde su raíz. Estas son:

* Detección del problema: En esta etapa se identifica una situación problemática relacionada con el dominio en cuestión. Para ello, se revisan distintos documentos y se toman en cuenta comentarios previos tanto de procesos como de productos durante el ciclo de vida del software.
* Análisis: Se lleva a cabo una revisión de la información recopilada con el objetivo de comprender a fondo las causas del problema, así como los factores que pueden haber influido en su aparición. Que hacen jsjsjs
* Implementación de acciones correctivas: Una vez identificado el problema, se actúa inmediatamente para corregirlo desde su origen. Esto puede incluir la revisión de procedimientos internos, la capacitación del equipo o la reparación de productos que hayan sido afectados, con la finalidad de evitar que el inconveniente se repita.
* Acciones preventivas: Después de tomar las medidas correctivas, se pasa a establecer mecanismos que ayuden a prevenir la reincidencia del problema. Esto abarca desde análisis de riesgos y mejoras en los procesos, hasta mantenimiento preventivo y monitoreo continuo.
* Verificación y seguimiento: Esta fase se enfoca en evaluar si las acciones tomadas realmente surtieron efecto. Para ello, se da seguimiento al problema, se verifica su resolución y, en caso necesario, se ajustan las estrategias adoptadas.



Para aplicar el modelo al informe a desarrollar, se empleó como base la plantilla llamada *“Plantilla Análisis Problemática CAPA - 5 ¿Por qué?”*, obtenida del sitio web Smartsheet. Esta herramienta resulta útil especialmente en la fase de identificación y análisis de las causas raíz, ya que permite desglosar posibles orígenes del problema y comenzar a planificar las acciones correctivas adecuadas.

Asimismo, de utilidad será la plantilla *“Acciones CAPA”* de la página Safety Culture. Pasando a la siguiente fase del proceso: la definición de acciones concretas tanto correctivas como preventivas. Esta plantilla ofrece una estructura clara para organizar la información correspondiente a cada etapa del desarrollo de dichas acciones.

**Herramientas, técnicas y metodologías**

Durante la elaboración de nuestro Plan de Aseguramiento de la Calidad del Software (SQAP), consideramos un conjunto de herramientas, técnicas y metodologías que nos han permitido asegurar la calidad a lo largo de todo el ciclo de desarrollo: desde el diseño, pasando por la codificación, hasta las etapas de prueba. A continuación, describimos las principales herramientas y metodologías que decidimos utilizar, junto con el propósito que cumplen y la forma en que las implementamos dentro del equipo.

Herramientas seleccionadas

* Jira  
   *Propósito:* Facilita la organización y seguimiento de tareas y proyectos en entornos colaborativos.  
   *Uso:* La empleamos para asignar tareas, gestionar incidencias, dar seguimiento al progreso del desarrollo y mantener una comunicación clara entre todos los miembros del equipo.
* SonarQube  
   *Propósito:* Permite analizar automáticamente el código para detectar errores, vulnerabilidades y problemas de calidad.  
   *Uso:* Lo utilizamos para llevar a cabo revisiones estáticas del código fuente, asegurando que se cumplan buenas prácticas de programación y se minimicen riesgos técnicos.
* Git  
   *Propósito:* Sistema de control de versiones distribuido, ideal para registrar y gestionar cambios en el código de manera eficiente.  
   *Uso:* Nos apoyamos en Git para mantener un historial detallado de nuestras modificaciones, facilitar el trabajo simultáneo y recuperar versiones anteriores si es necesario.
* GitHub  
   *Propósito:* Plataforma que permite almacenar el código en la nube y colaborar de forma remota.  
   *Uso:* Nos ayuda a centralizar los repositorios del proyecto, manejar pull requests y realizar revisiones de código entre pares.
* VSCode  
   *Propósito:* Editor de código ligero y flexible, compatible con múltiples lenguajes.  
   *Uso:* Fue nuestra herramienta principal para el desarrollo, donde codificamos, depuramos y personalizamos nuestro entorno según las necesidades del proyecto.
* GitKraken  
   *Propósito:* Interfaz gráfica para Git que simplifica la gestión visual de ramas y versiones.  
   *Uso:* Nos permitió tener un mayor control visual sobre la estructura de ramas, facilitar merges y resolver conflictos de forma más intuitiva.
* WAMP  
   *Propósito:* Entorno local que incluye Apache, MySQL y PHP, ideal para desarrollo web en Windows.  
   *Uso:* Lo empleamos para montar y probar las aplicaciones localmente antes de su publicación, facilitando la detección de errores y el trabajo independiente de cada integrante.
* Composer  
   *Propósito:* Administrador de dependencias para proyectos PHP.  
   *Uso:* Nos permitió gestionar bibliotecas externas de forma eficiente, asegurando compatibilidad entre versiones y simplificando la instalación de paquetes.
* MySQL Workbench  
   *Propósito:* Herramienta visual para modelar y administrar bases de datos MySQL.  
   *Uso:* La utilizamos para diseñar esquemas de base de datos, ejecutar consultas SQL y gestionar la información almacenada durante el desarrollo de nuestras funcionalidades.

Metodología aplicada

* SCRUM  
   *Propósito:* Marco de trabajo ágil que permite organizar el desarrollo en ciclos cortos e iterativos.  
   *Uso:* Optamos por aplicar SCRUM para estructurar el trabajo en sprints, facilitando así la entrega continua de avances funcionales y fomentando una comunicación fluida dentro del equipo. Esta metodología nos permitió adaptarnos a los cambios, recibir retroalimentación constante y mejorar el producto de forma progresiva.

# Control de Código

Para el control de código, se toma como guía el estándar internacional ISO/IEC/IEEE 12207:2008 (puntos 7.2.2 - 7.2.2.3.1), específicamente el proceso de Gestión de la Configuración del Software (Software Configuration Management Process), con el objetivo de establecer y mantener la integridad de los elementos de software a lo largo del ciclo de vida del proyecto.

Este proceso establece una serie de resultados clave que deben lograrse para asegurar una gestión eficaz de la configuración del software:

* Desarrollo de una estrategia de gestión de la configuración del software.
* Identificación, definición y establecimiento de líneas base de los elementos generados por el proceso o proyecto.
* Control de las modificaciones y liberaciones de los elementos.
* Disponibilidad de las modificaciones y liberaciones para las partes interesadas.
* Registro y reporte del estado de los elementos y sus modificaciones.
* Aseguramiento de la completitud y consistencia de los elementos.
* Control del almacenamiento, manejo y entrega de los elementos.

Para proteger y garantizar la validez del código, se llevaron a cabo las siguientes actividades conforme al proceso de Gestión de la Configuración del Software de la norma ISO/IEC/IEEE 12207:2008

**Uso de Sistemas de Control de Versiones:**

Se implementó un sistema de control de versiones utilizando herramientas como Git y plataformas como GitHub para almacenar y gestionar las diferentes versiones y configuraciones de los componentes de software. Esto permite identificar y etiquetar entidades de referencia, así como capturar y rastrear el estado de los cambios en dichas entidades.

**Procesamiento de Cambios:**

Se establecieron procedimientos para la solicitud, evaluación, aprobación y registro de cambios en los elementos de configuración. Esto incluye la definición de la información necesaria y el enrutamiento de las solicitudes de cambio, asegurando que todas las modificaciones sean controladas adecuadamente.

**Comité de Gestión de la Configuración del Software:**

Se conformó un comité responsable de revisar, aprobar o rechazar cambios, garantizando la seguridad y protección contra alteraciones no autorizadas después de que el código se haya establecido como parte del sistema. Este comité actúa como una autoridad central para la toma de decisiones relacionadas con la configuración del software.

**Registro y Reporte del Estado de la Configuración:**

Se mantuvieron registros actualizados del estado de cada elemento de configuración, incluyendo información sobre versiones, cambios realizados y estado actual. Estos registros permiten una trazabilidad completa y facilitan la auditoría y revisión del estado del software en cualquier momento.

**Evaluación de la Configuración:**

Se llevaron a cabo evaluaciones periódicas para asegurar la completitud funcional y física de los elementos de software, verificando que cumplan con los requisitos establecidos y que su diseño y código reflejen una descripción técnica actualizada.

**Gestión de Liberaciones y Entregas:**

Se controlaron formalmente las liberaciones y entregas de productos de software y documentación, manteniendo copias maestras del código y la documentación durante la vida útil del producto. Se aseguraron políticas adecuadas para el manejo, almacenamiento, empaquetado y entrega de elementos críticos en términos de seguridad y funcionalidad.

Implementar estas actividades conforme al estándar ISO/IEC/IEEE 12207:2008 garantiza un control de código eficaz, contribuyendo a la calidad y éxito del proyecto de software.

# Control de medios

El **control de medios** comprende la implementación de métodos y mecanismos destinados a identificar adecuadamente los medios empleados para cada producto informático, así como la documentación requerida para su almacenamiento seguro, incluyendo los procedimientos de copia de seguridad y restauración de datos. Además, se establecen medidas de protección contra accesos no autorizados o daños, garantizando de esta forma la **integridad**, **disponibilidad** y **confidencialidad** de la información almacenada.

La definición de este proceso se basa en el estándar **ISO/IEC 27002:2022**, el cual proporciona una guía detallada sobre la implementación de controles de seguridad de la información. Particularmente, las siguientes secciones son relevantes para el control de medios:

* **Cláusula 8.12 ("Data masking")** y **8.11 ("Data masking")**: establecen prácticas de protección de datos, incluyendo la manipulación y resguardo de la información almacenada.
* **Cláusula 8.9 ("Information transfer")** y **8.10 ("Access control")**: abordan medidas de control de acceso y transferencia segura de información para evitar accesos no autorizados .
* **Cláusula 7.10 ("Storage media")**: trata explícitamente sobre el manejo seguro de medios de almacenamiento, destacando la necesidad de protección física, control de acceso y procedimientos de eliminación segura de los medios .

Estas recomendaciones buscan asegurar que los datos y programas almacenados se mantengan íntegros, disponibles únicamente para usuarios autorizados, y protegidos contra alteraciones o degradaciones involuntarias.

En el marco de este control, se implementa el uso de **GitHub** como plataforma para el almacenamiento y la gestión de versiones del software. A través de sistemas de permisos configurados, GitHub permite controlar el acceso a los repositorios, asegurando que únicamente usuarios autorizados puedan visualizar o modificar el código fuente. Además, se establece un **procedimiento de respaldo** que detalla:

* La **frecuencia** de las copias de seguridad.
* El **tipo de medios** utilizados.
* La **ubicación física o lógica** de almacenamiento.
* Las condiciones del **entorno de almacenamiento**.
* Los **métodos de restauración** aplicables en caso de pérdida de datos .

**Protección contra acceso no autorizado**  
 Para proteger adecuadamente los medios de los programas informáticos contra el acceso no autorizado, se implementa un esquema de identificación permanente dentro del medio de almacenamiento. Este esquema incluye el cifrado de datos almacenados, añadiendo una capa adicional de seguridad. Los datos en los medios de software son cifrados de manera que solo los usuarios con las claves de desencriptación adecuadas puedan acceder a la información. Este método asegura que, incluso si los medios son comprometidos, los datos permanecen inaccesibles sin las credenciales correctas.

# Recopilación, mantenimiento y retención de documentos

Durante el desarrollo del Plan de Aseguramiento de la Calidad del Software (SQAP), se crean diferentes documentos, por lo que se examina la colección, mantenimiento y retención de documentos para garantizar la integridad y trazabilidad de la documentación.

## Colección de documentos

Para demostrar el cumplimiento del proceso de desarrollo de software y ofrecer información histórica y de referencia para análisis futuros, se recopilan documentos tanto en formato digital como impreso.

Entre los archivos físicos almacenados se incluyen contratos, acuerdos legales, informes departamentales, registros financieros y documentación de recursos humanos. Para gestionar el acceso a estos documentos, se implementa un procedimiento de solicitud que abarca su consulta, modificación y devolución. Este proceso inicia cuando el solicitante envía una petición al Jefe de Proyecto, quien le proporciona un formato de control documental. Una vez completado, el formulario se remite al Jefe de Departamento para su aprobación o rechazo. Además, este último registra en el historial del almacén todas las salidas y devoluciones de los documentos. En cuanto a los documentos digitales, se utilizan las herramientas proporcionadas por *GitHub* para controlar el historial de versiones y acceso a los documentos, asignándoles los roles que les corresponden a cada persona del departamento dependiendo del tipo de documento elaborado en la fase de desarrollo que se describe a continuación:

* Documento de Especificación de Requisitos (SRS)
* Acceso: Especialista en desarrollo SQA y Líder de SQA
* Descripción del Diseño del Software (SDD)
* Acceso: Gerente de calidad y Líder de SQA
* Plan de verificación y validación de software
* Acceso: Especialista en desarrollo SQA y Líder de SQA
* Informe de resultados de la verificación
* Acceso: Especialista en desarrollo SQA, Gerente de calidad y Líder de SQA

Informe de resultados de validación

Acceso: Especialista en desarrollo SQA, Gerente de calidad y Líder de SQA

Documentación de usuario

Acceso: Especialista en capacitación SQA y Líder de SQA

Plan de Gestión de Configuración de Software (SCMP)

Acceso: Especialista en desarrollo SQA y Líder de SQA

Plan de Desarrollo de Software (SDP)

Acceso: Especialista en desarrollo SQA y Líder de SQA

Manual de Mantenimiento de Software (SMM)

Acceso: Gerente de calidad y Líder de SQA

Especificaciones de la Interfaz Externa

Acceso: Especialista en desarrollo SQA y Líder de SQA

Especificaciones de la Interfaz Interna

Acceso: Especialista en desarrollo SQA y Líder de SQA

Plan de Métricas de Software

Acceso: Gerente de calidad y Líder de SQA

Plan de Seguridad del Software

Acceso: Gerente de calidad y Líder de SQA

Plantilla Smartsheet (IC-IE-Technical-Specification-Template\_ES). Para el manejo de los documentos, se hace uso de la plantilla proporcionada por la plataforma Smartsheet la cual permite a las organizaciones planificar, rastrear, automatizar y gestionar una amplia variedad de proyectos y procesos empresariales. Esta plantilla proporciona un marco para la administración de los documentos confidenciales.

## Mantenimiento de documentos

Los documentos impresos se guardan en un almacén el cual solo tiene acceso el Jefe de Departamento, para así controlar el acceso indebido a dicha información y prevenir los robos de información. El almacén cuenta con todos los protocolos de seguridad contra incendios y condiciones adecuadas para prevenir daños por deterioro ambiental, como también un registro con altas, cambios y bajas de los archivos en cuestión.

En cuanto a los documentos digitales, se automatizan todos los procesos mencionados anteriormente, con la herramienta GitHub.

## Retención de documentos

Se conservan todos los documentos durante la vida útil del software, teniendo en cuenta que a la hora de modificación o actualización de estos, son reemplazados en el almacenamiento especificado, con el fin de solo mantener la información actualizada.

A la hora de la destrucción de los documentos, por finalización del proyecto o causas externas, el Jefe de Departamento se encarga de eliminar por completo dichos archivos.

# Capacitación

Para la realización del proyecto actual es indispensable contar un conjunto de habilidades relacionadas al desarrollo de software, las cuales de manera específica son:

Habilidades relacionadas al desarrollo de aplicaciones web:

* Conocimiento en los lenguajes PHP y javascript.
* Conocimiento en manejo de base de datos.
* Conocimiento relacionado al hospedaje de aplicaciones web en servidores.

Habilidades relacionadas al control de la configuración:

* Conocimiento en herramientas de control de versiones.
* Conocimiento en herramientas de control de flujo de trabajo.
* Conocimiento en herramientas de control de documentación.
* Habilidades relacionadas al aseguramiento de la calidad de software:
* Conocimiento en diferentes tipos de revisiones, inspecciones y auditorías.
* Conocimiento en la aplicación de pruebas.
* Conocimiento en métodos de aplicación de métricas.

Para comprobar el cumplimiento de estas capacidades y poder brindar una capacitación acorde se realiza un plan de entrenamiento basado en el estándar ISO 10015:2019 Quality management - Guidelines for competence management and people development, el cual proporciona pautas generales para el desarrollo, implementación y evaluación de programas de formación en organizaciones. Aunque no está dirigido específicamente a la ingeniería de software, ofrece principios y prácticas que son aplicables al desarrollo de planes de entrenamiento en este campo debido a que se basa en verificar las necesidades de formación, así como determinar cómo atenderlas y darles un seguimiento.

Este plan consta de las siguientes actividades:

Identificar las necesidades de formación: El equipo realiza una evaluación de habilidades y conocimientos para identificar las áreas en las que necesitan mejorar, como la adopción de nuevas tecnologías o metodologías de desarrollo.

Planificar la formación: Basándose en las necesidades identificadas, el equipo desarrolla un plan de formación que incluye cursos sobre las nuevas tecnologías y metodologías relevantes para el proyecto en curso.

Implementar el plan de formación: El equipo ofrece sesiones de formación internas, talleres y acceso a recursos de aprendizaje en línea para que los miembros del equipo puedan adquirir las habilidades necesarias.

Evaluar la efectividad de la formación: Después de completar la formación, el equipo realiza una evaluación para medir el impacto en el rendimiento del equipo y la capacidad para aplicar las nuevas habilidades en el trabajo diario.

Mejora continua: Basándose en los resultados de la evaluación, el equipo ajusta su programa de formación, añadiendo nuevas áreas de formación o modificando los enfoques existentes para garantizar que sigan siendo efectivos y relevantes.

# Anexos

[Documentos utilizados](https://drive.google.com/drive/folders/1K9Ypo9eY2xN5UZ9INx0b8INBdksMOuCC)

# Referencias

IEEE. (2016). IEEE Std 1012-2016: IEEE Standard for System and Software Verification and Validation. IEEE.

ISO/IEC. (2013). ISO/IEC/IEEE 29119-2:2013 - Systems and software engineering -- Software testing -- Part 2: Test processes. ISO/IEC.

ISO/IEC. (2013). ISO/IEC/IEEE 29119-3:2013 - Systems and software engineering -- Software testing -- Part 3: Test documentation. ISO/IEC.

IEEE. (1987). IEEE Std 1042-1987: IEEE Guide to Software Configuration Management. IEEE.

IEEE Computer Society. (2012). IEEE Standard for Software Configuration Management Plans (IEEE Std 828-2012). IEEE.

ISO/IEC/IEEE. (2011). Systems and software engineering -- Life cycle processes -- Requirements engineering (ISO/IEC/IEEE 29148:2011). International Organization for Standardization.

National Aeronautics and Space Administration (NASA). (2016). Expanded Guidance for NASA Systems Engineering Volume 1: Systems Engineering Practices.

IEEE Computer Society. (2009). IEEE Standard for a Software Quality Metrics Methodology (IEEE Std 1061-1998 (R2009)). IEEE.

IEEE. (2014). IEEE Std 730-2014: IEEE Standard for Software Quality Assurance Plans. IEEE.

IEEE. (1998). IEEE Std 1219-1998: IEEE Standard for Software Maintenance. IEEE.

National Aeronautics and Space Administration (NASA). (2004). NASA Software Safety Guidebook (NASA-GB-8719.13).

IEEE. (1997). IEEE Standard for Software User Documentation (IEEE Std 1063-1997). IEEE.

International Organization for Standardization. (2008). Systems and software engineering -- Software life cycle processes (ISO/IEC/IEEE 12207:2008). ISO.

International Organization for Standardization. (2019). Quality management -- Guidelines for training (ISO 10015:2019). ISO.

IEEE. (1995). IEEE Std 730.1-1995: IEEE Guide for Software Quality Assurance Planning. IEEE.

ISO/IEC. (2022). ISO/IEC 27002:2022 Information security, cybersecurity and privacy protection — Information security controls.