

UNIVERSIDAD PERUANA LOS ANDES
ESCUELA PROFESIONAL DE INGENIERÍA DE
SISTEMAS Y COMPUTACIÓN



ARQUITECTURA DE SOFTWARE

PRESENTADO POR:

APELLIDOS Y NOMBRES	CÓDIGO
Guerra Yarupaita Carlos Daniel	R01069E

ASESOR:Fernandez Bejarano Raul Enrique

HUANCAYO – PERÚ

2025

Tema 1: Importancia de los estándares en la Arquitectura de Software

Subtema 1.1: Concepto de estándar y su papel en la ingeniería de software

Introducción

En el ámbito de la ingeniería de software, los estándares representan un conjunto de normas, lineamientos y criterios consensuados que guían las prácticas de desarrollo, asegurando uniformidad, calidad y eficiencia en los procesos. Según Sommerville (2011), un estándar es “un documento que proporciona reglas, directrices o características para actividades o sus resultados, con el fin de lograr un grado óptimo de orden en un contexto determinado”.

La relevancia de los estándares en la ingeniería de software radica en que permiten establecer un marco común de trabajo entre equipos, organizaciones y sectores de la industria tecnológica. Estos aseguran la interoperabilidad entre sistemas, la calidad del producto y la mejora continua de procesos, además de contribuir a la reducción de errores, costos y tiempos de desarrollo (Pressman & Maxim, 2021).

En este contexto, organismos internacionales como la ISO (International Organization for Standardization) y el IEEE (Institute of Electrical and Electronics Engineers) han desarrollado estándares ampliamente utilizados en la industria del software, tales como ISO/IEC 12207 para procesos de ciclo de vida del software o IEEE 830 para especificación de requisitos. Dichos estándares se convierten en referentes fundamentales para la práctica profesional, aportando un marco metodológico y de buenas prácticas que incrementa la confianza en los sistemas desarrollados.

En conclusión, los estándares constituyen un pilar estratégico en la ingeniería de software, ya que garantizan consistencia, confiabilidad y competitividad en la construcción de soluciones tecnológicas.



Subtema 1.2: Ventajas del uso de estándares internacionales en proyectos de software

El empleo de estándares internacionales en la arquitectura y desarrollo de software representa un elemento clave para alcanzar niveles óptimos de calidad, seguridad y sostenibilidad en proyectos tecnológicos. Estos lineamientos, definidos por organismos como la ISO (International Organization for Standardization) o el IEEE (Institute of Electrical and Electronics Engineers), proporcionan metodologías y criterios unificados que permiten reducir riesgos, garantizar la interoperabilidad y facilitar la gestión eficiente del ciclo de vida del software.

Caso real: Microsoft y la adopción de ISO/IEC 27001

Microsoft ha implementado el estándar ISO/IEC 27001, enfocado en la gestión de la seguridad de la información, en diversos servicios de su nube Azure. Gracias a esta certificación, la compañía asegura a sus clientes que la infraestructura cumple con controles de seguridad robustos y con normativas internacionales de protección de datos. Esta práctica ha fortalecido la confianza de los usuarios empresariales en los servicios de Microsoft, convirtiéndose en una ventaja competitiva frente a proveedores que no cumplen con estos estándares.

Ventajas concretas del uso de estándares internacionales

1. **Calidad garantizada:** Los estándares proporcionan lineamientos que aseguran que los procesos de desarrollo cumplan requisitos mínimos de calidad (Pressman & Maxim, 2021).
2. **Interoperabilidad:** Facilitan la integración entre sistemas y tecnologías desarrolladas por distintos equipos u organizaciones.
3. **Confiabilidad y seguridad:** Normas como ISO/IEC 27001 o ISO/IEC 25010 aseguran que los productos cumplan con características de seguridad, rendimiento y mantenibilidad.
4. **Ahorro de costos y tiempo:** Al contar con procesos predefinidos y buenas prácticas reconocidas internacionalmente, se reducen retrabajos y fallas en fases posteriores.
5. **Competitividad global:** Las Certificaciones internacionales son un diferenciador clave en licitaciones y mercados altamente regulados, al demostrar cumplimiento con requisitos de calidad y seguridad.

Ejemplo adicional: IBM y el estándar IEEE 830

IBM, en el desarrollo de grandes sistemas de información para clientes del sector financiero, ha utilizado el estándar IEEE 830 (ahora reemplazado por IEEE 29148) para la especificación de requisitos de software. Con ello, la empresa ha logrado mejorar la comunicación con sus clientes, reducir ambigüedades en la definición de requerimientos y asegurar entregables que cumplen las expectativas del usuario final.

En síntesis, la aplicación de estándares internacionales no solo asegura la calidad técnica de los proyectos de software, sino que también incrementa la confianza de clientes y usuarios, fomenta la eficiencia operativa y brinda a las organizaciones una posición competitiva sólida en el mercado global.

Cuadro comparativo: Uso de estándares internacionales en proyectos de software

Empresa	Estándar aplicado	Objetivo del estándar	Beneficios logrados
Microsoft (Azure)	ISO/IEC 27001 (Seguridad de la información)	Garantizar la gestión adecuada de la seguridad y la protección de datos en servicios en la nube.	<ul style="list-style-type: none"> - Mayor confianza de los clientes en la seguridad de Azure. - Cumplimiento con regulaciones internacionales de protección de datos. - Ventaja competitiva en el mercado de servicios cloud.
IBM (Sistemas financieros)	IEEE 830 / IEEE 29148 (Especificación de requisitos de software)	Definir con precisión los requisitos funcionales y no funcionales, reduciendo ambigüedades.	<ul style="list-style-type: none"> - Mejora en la comunicación con clientes. - Reducción de errores en la etapa de definición de requisitos. - Entregables alineados con las expectativas del usuario.
Google (Android)	ISO/IEC 25010 (Calidad del producto software)	Evaluar atributos de calidad como seguridad, usabilidad, mantenibilidad y eficiencia.	<ul style="list-style-type: none"> - Productos más robustos y confiables. - Mejor experiencia de usuario en aplicaciones Android. - Mayor satisfacción del cliente y adopción global del sistema operativo.

Subtema 1.3: Relación entre estándares, calidad y buenas prácticas de desarrollo

Cuadro comparativo: Buenas prácticas sin estándares vs. con estándares

Aspecto	Buenas prácticas sin estándares	Buenas prácticas con estándares internacionales	Relación con la calidad

Definición de requisitos	Cada equipo define sus propios métodos, lo que puede generar ambigüedad o falta de consenso.	Uso de IEEE 29148 para la especificación clara y verificable de requisitos.	Requisitos más precisos y medibles, reduciendo errores en fases posteriores.
Proceso de desarrollo	Se aplican prácticas individuales basadas en experiencia previa.	Aplicación de ISO/IEC 12207 para estandarizar el ciclo de vida del software.	Se logra un desarrollo estructurado y repetible, incrementando la confiabilidad.
Aseguramiento de calidad	Depende del criterio del equipo, con controles irregulares.	Implementación de ISO/IEC 25010 para evaluar atributos como mantenibilidad, usabilidad y seguridad.	Se garantiza un producto alineado con métricas internacionales de calidad.
Gestión de seguridad	Cada desarrollador aplica medidas básicas sin lineamientos unificados.	Adopción de ISO/IEC 27001 para la gestión integral de la seguridad de la información.	Se obtiene protección robusta y estandarizada, aumentando la confianza del cliente.
Documentación	Puede ser incompleta, inconsistente o no actualizada.	Uso de normas IEEE para la documentación técnica y de requisitos.	Documentación uniforme y clara, facilitando mantenimiento y futuras mejoras.

Subtema 1.4: Casos prácticos de organizaciones que aplican estándares en la producción de software

Casos prácticos de organizaciones que aplican estándares en la producción de software

El uso de estándares internacionales en la producción de software ha demostrado ser un factor decisivo para garantizar calidad, seguridad y competitividad en distintos mercados. A continuación, se describen casos prácticos de empresas que han implementado estos estándares en sus procesos de desarrollo:

1. Everis Perú (actualmente NTT Data)

Everis Perú, una consultora multinacional con sede en Lima, ha adoptado estándares como la ISO/IEC 12207 (procesos del ciclo de vida del software) y la ISO/IEC 27001 (gestión de seguridad de la información) en proyectos de transformación digital para entidades

financieras y gubernamentales. Gracias a estos lineamientos, la empresa logró estandarizar procesos de desarrollo, incrementar la seguridad de las aplicaciones y garantizar la satisfacción de clientes que requieren cumplimiento normativo estricto.

Beneficios obtenidos:

- Procesos más estructurados y repetitivos.
- Aumento de la confianza en sectores regulados (banca y gobierno).
- Mejora en la calidad percibida del servicio.

2. Google (Android Open Source Project)

Google aplica la norma ISO/IEC 25010 para evaluar la calidad del software, considerando atributos como usabilidad, portabilidad y seguridad en el desarrollo de Android. Esto ha permitido que el sistema operativo mantenga altos niveles de confiabilidad, adoptado por miles de fabricantes en todo el mundo.

Beneficios obtenidos:

- Mayor robustez y seguridad en las versiones de Android.
- Uniformidad en la experiencia de usuario a nivel global.
- Estándares de calidad que permiten la interoperabilidad entre dispositivos.

3. IBM (soluciones empresariales)

IBM utiliza el estándar IEEE 29148 (especificación de requisitos de software) en el diseño de sistemas de misión crítica para sectores como la banca y la salud. Esto asegura que los requisitos funcionales y no funcionales estén claramente definidos y validados desde el inicio.

Beneficios obtenidos:

- Reducción de errores en la etapa de definición de requisitos.
- Ahorro de costos al minimizar retrabajos.
- Entregables alineados con las expectativas del cliente.

Tema 2: Estándares Internacionales más relevantes en Arquitectura de Software

Subtema 2.1: ISO/IEC/IEEE 42010

Aplicación del estándar ISO/IEC/IEEE 42010 en un sistema educativo

El estándar ISO/IEC/IEEE 42010 define cómo describir la arquitectura de un sistema mediante vistas y puntos de vista, permitiendo representar de manera ordenada los elementos, relaciones y preocupaciones de los interesados (stakeholders).

En el caso de un sistema educativo (plataforma de gestión académica en línea), se propone el siguiente modelo arquitectónico:

1. Vista de contexto (Context View)

Muestra la relación entre el sistema educativo y los actores externos.

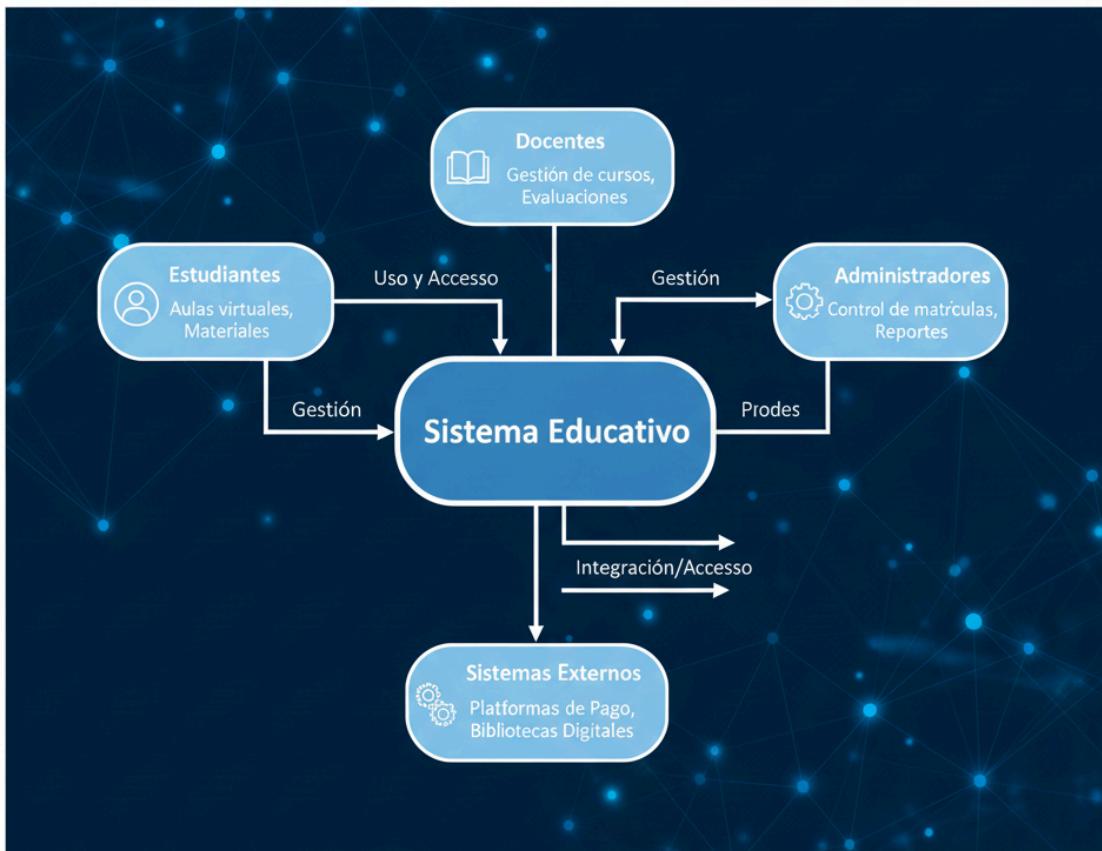
Stakeholders: Estudiantes, docentes, administradores, proveedores externos.

Diagrama de contexto:

- **El Sistema Educativo se conecta con:**
 - Estudiantes (uso de aulas virtuales, acceso a materiales).
 - Docentes (gestión de cursos, evaluaciones).
 - Administradores (control de matrículas, reportes).
 - Sistemas externos (plataformas de pago, bibliotecas digitales).

1. Vista de Contexto

Relación con Actores Externos



2. Vista lógica (Logical View)

Describe los principales módulos y componentes del sistema.

Componentes:

- Gestión académica (matrículas, planes de estudio).
- Gestión de cursos (contenidos, evaluaciones, foros).
- Gestión de usuarios (roles: estudiante, docente, administrador).
- Reportes y analítica (estadísticas de rendimiento, asistencia).
- Integraciones externas (pagos, repositorios digitales).

2. Vista Lógica

Componentes y Módulos del Sistema



3. Vista de desarrollo (Development View)

Representa cómo se organiza el software en términos de módulos y tecnologías.

Ejemplo:

- Backend: Microservicios en Java/Spring Boot.
- Frontend: Aplicación web en React.
- Base de datos: PostgreSQL.
- API REST para comunicación con sistemas externos.

2. Vista Lógica

Componentes y Módulos del Sistema



4. Vista de procesos (Process View)

Modela el comportamiento dinámico del sistema.

Ejemplo de proceso:

Registro de estudiante en un curso:

1. Usuario se autentica en el sistema.
2. Selecciona curso.
3. Se valida disponibilidad y cupos.

4. Se confirma inscripción.
5. El sistema actualiza la base de datos y genera un comprobante.



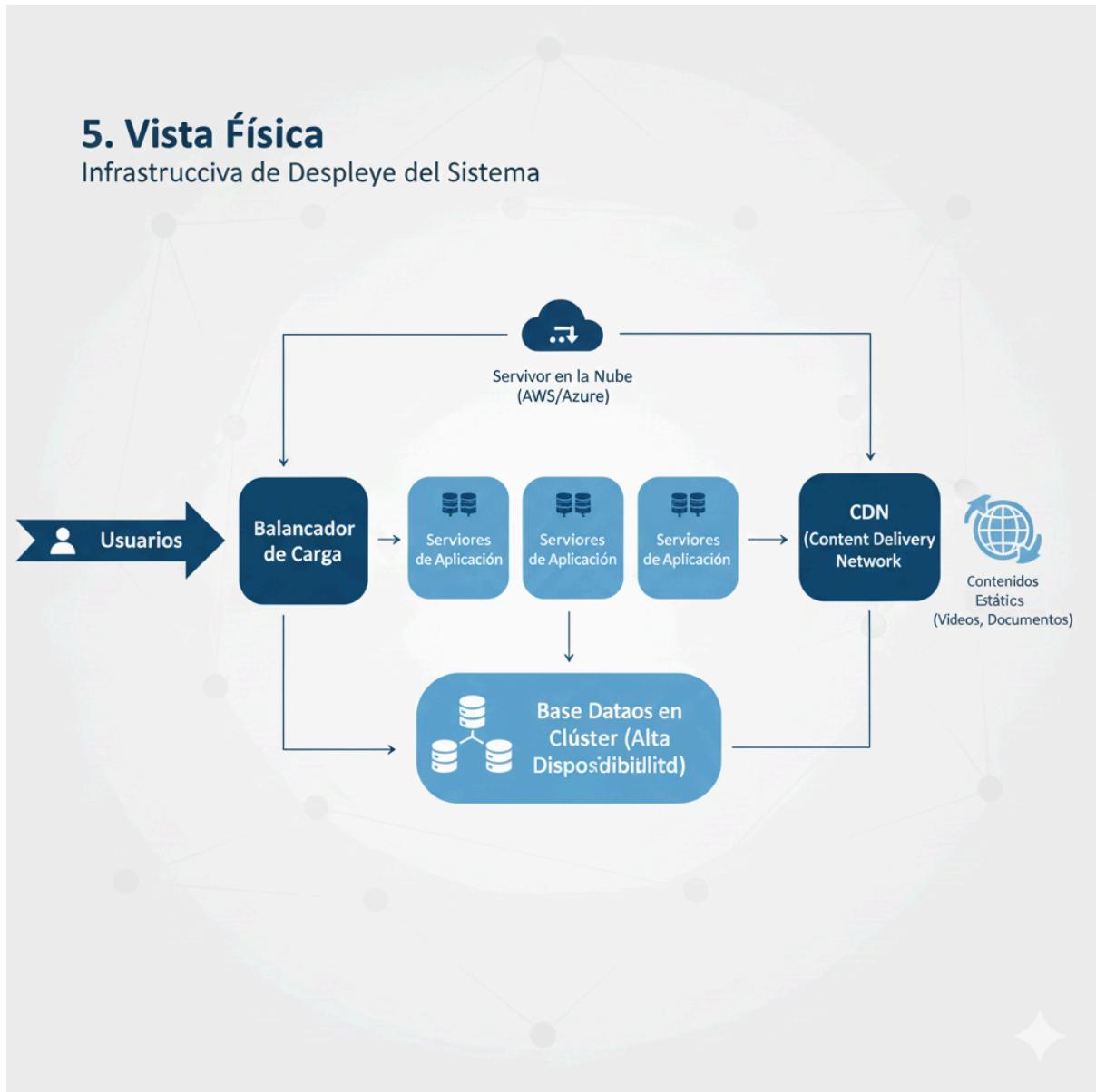
5. Vista física (Physical View)

Muestra la infraestructura donde se despliega el sistema.

Ejemplo:

- Servidor en la nube (AWS/Azure).
- Balanceador de carga.

- Servidores de aplicación.
- Base de datos en clúster para alta disponibilidad.
- CDN para distribución de contenidos.



Subtema 2.2: ISO/IEC 12207

ISO/IEC 12207: Procesos del ciclo de vida del software

El estándar ISO/IEC 12207 establece un marco para la gestión del ciclo de vida del software, definiendo procesos, actividades y tareas que deben considerarse desde la

concepción hasta el retiro del sistema. Su objetivo es garantizar que el software cumpla con los requisitos de calidad, sea mantenible y responda a las necesidades de los usuarios.

Procesos principales definidos por ISO/IEC 12207

1. Procesos de adquisición y suministro

- Inician el ciclo de vida. Incluyen la negociación de contratos, definición de requisitos iniciales y entrega del producto por parte del proveedor.

2. Procesos de desarrollo

- **Análisis de requisitos:** Definición de necesidades funcionales y no funcionales.
- **Diseño arquitectónico y detallado:** Estructuración del sistema y definición de componentes.
- **Implementación y pruebas unitarias:** Construcción del software y validación de módulos.
- **Integración y pruebas del sistema:** Unificación de componentes y pruebas completas.

3. Procesos de operación y mantenimiento

- Instalación, operación y soporte del sistema.
- Actualización y corrección de errores tras la entrega.

4. Procesos de soporte

- Gestión de configuración, aseguramiento de la calidad, verificación y validación, auditorías y documentación.

5. Procesos de retiro

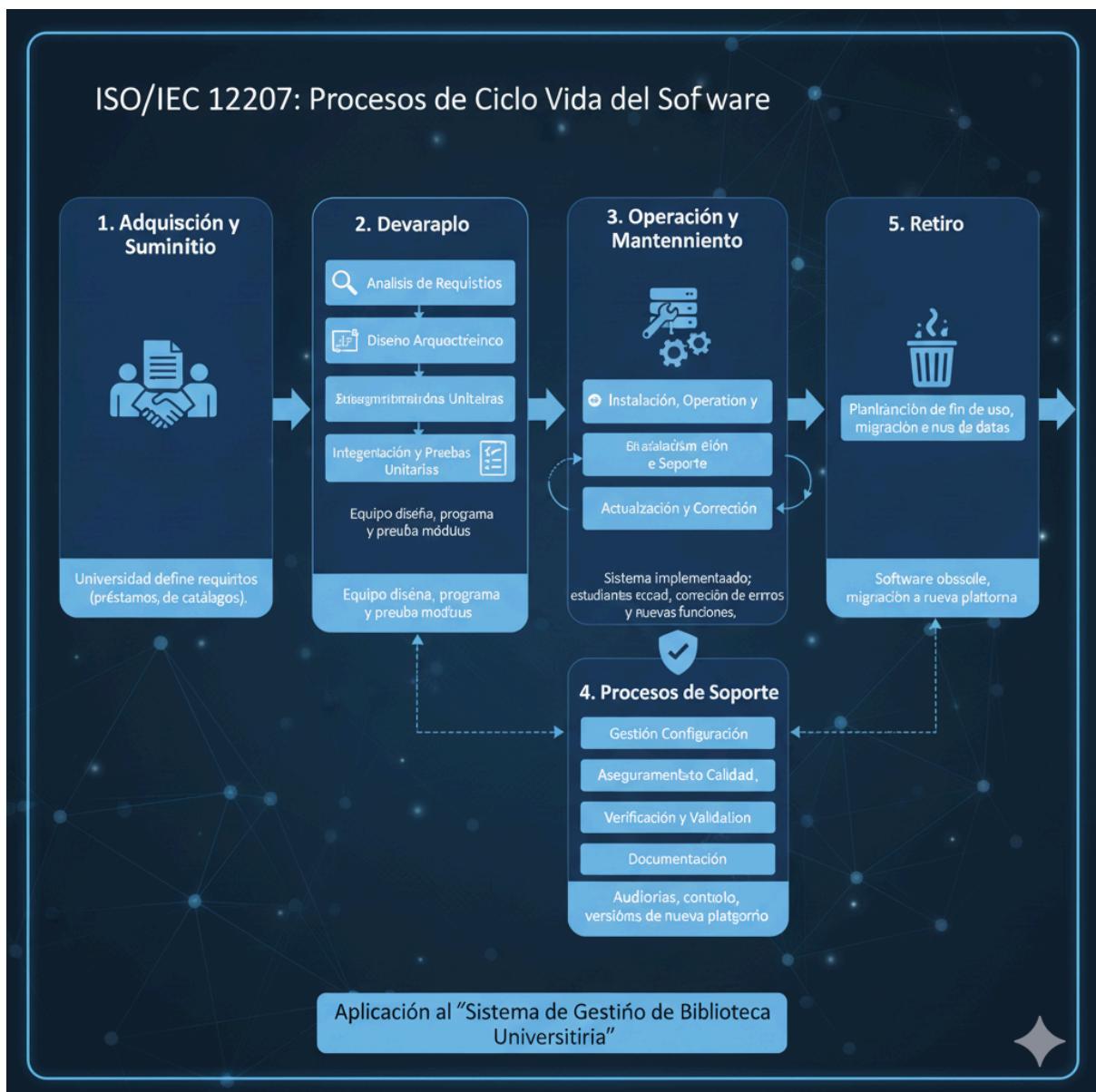
- Planificación de la finalización del uso del software y migración de datos.

Ejemplo aplicado a mi proyecto académico

Para mi sistema de gestión de biblioteca universitaria, los procesos se aplicarían así:

- **Adquisición:** La universidad define los requisitos del sistema (gestión de préstamos, catálogos, reportes).

- **Desarrollo:** El equipo diseña la arquitectura del sistema, programa los módulos y realiza pruebas.
- **Operación:** El sistema se implementa en el campus y los estudiantes acceden mediante credenciales institucionales.
- **Mantenimiento:** Se corrigen errores y se agregan nuevas funciones, como integración con bases de datos externas.
- **Retiro:** Cuando el software quede obsoleto, se migrará a una nueva plataforma educativa.



Subtema 2.3: ISO/IEC 25010

ISO/IEC 25010: Evaluación de la calidad del software

El estándar ISO/IEC 25010 define un modelo de calidad compuesto por 8 características principales (funcionalidad, rendimiento, compatibilidad, usabilidad, fiabilidad, seguridad, mantenibilidad y portabilidad) y sus subcaracterísticas. Para este ejercicio, nos enfocaremos en usabilidad, seguridad y mantenibilidad, aplicadas a un sistema de gestión académica universitaria.

Matriz de evaluación de calidad

Característica (ISO/IEC 25010)	Subcaracterísticas	Métricas sugeridas	Ejemplo aplicado al sistema académico
Usabilidad	-Facilidad aprendizaje de -Eficiencia de uso -Accesibilidad	-Tiempo promedio para que un nuevo usuario realice una tarea básica. -Porcentaje de errores cometidos por usuarios en pruebas de uso. -Nivel de cumplimiento con WCAG (accesibilidad web).	Los estudiantes logran registrarse en menos de 3 minutos sin ayuda; menos del 5% comete errores en la inscripción de cursos.
Seguridad	-Confidencialidad -Integridad -Autenticación autorización	-Nº de intentos de acceso no autorizado detectados. -% de datos cifrados en tránsito y reposo. -Tiempo promedio para aplicar parches de seguridad.	El sistema bloquea la cuenta tras 5 intentos fallidos; el 100% de los datos sensibles está cifrado en AES-256.

Mantenibilidad	<ul style="list-style-type: none"> -Modularidad -Reusabilidad -Analizabilidad 	<ul style="list-style-type: none"> -Complejidad ciclomática promedio del código. -Tiempo requerido para corregir un error. -Porcentaje de componentes reutilizables. 	El equipo corrige errores críticos en menos de 24 horas; el 40% de los módulos son reutilizables en otros proyectos.
-----------------------	--	---	--

Subtema 2.4: IEEE 1471

Comparación entre IEEE 1471 e ISO/IEC/IEEE 42010

La estandarización de la descripción de arquitecturas de software ha evolucionado con el tiempo. El estándar IEEE 1471-2000 fue el primero en establecer un marco formal para describir arquitecturas, y posteriormente se integró y amplió en el estándar internacional ISO/IEC/IEEE 42010:2011.

Semejanzas

1. **Objetivo común:** Ambos estándares buscan proporcionar un marco para describir arquitecturas de sistemas de manera consistente y comprensible para los diferentes stakeholders.
2. **Uso de vistas y puntos de vista:** Tanto IEEE 1471 como ISO/IEC/IEEE 42010 introducen la noción de vistas arquitectónicas y puntos de vista, como mecanismo para organizar la información arquitectónica.
3. **Enfoque en stakeholders:** Reconocen la importancia de identificar a los interesados y sus preocupaciones en el diseño arquitectónico.

Diferencias

Aspecto	IEEE 1471-2000	ISO/IEC/IEEE 42010:2011
Alcance	Centrado en la descripción arquitectónica de software intensivo.	Ampliado a sistemas en general, no solo software.
Estatus	Estándar aprobado por IEEE en 2000.	Estándar internacional (ISO/IEC/IEEE) publicado en 2011.

Evolución	Introdujo conceptos básicos de vistas, puntos de vista y stakeholders.	Refinó y amplió esos conceptos, con mayor formalidad y lineamientos más precisos.
Interoperabilidad	No contemplaba alineación con otros estándares internacionales.	Diseñado para ser coherente con otros estándares de sistemas y software.
Formalidad	Menos prescriptivo en la documentación.	Define con más rigor la correspondencia entre vistas, puntos de vista y modelos.

Otros estándares (TOGAF, Zachman, CMMI)

Otros estándares y marcos de referencia en Arquitectura de Software

Además de los estándares ISO/IEC/IEEE, existen marcos de referencia ampliamente utilizados en proyectos empresariales que apoyan la gestión y la arquitectura de software. Entre los más relevantes se encuentran TOGAF, Zachman y CMMI, cada uno con un enfoque particular, pero complementarios en la práctica profesional.

1. TOGAF (The Open Group Architecture Framework)

TOGAF es un marco de referencia para el desarrollo de arquitecturas empresariales. Su componente principal es el ADM (Architecture Development Method), un ciclo iterativo que guía desde la planificación hasta la gobernanza de la arquitectura.

- **Utilidad en proyectos empresariales:**
 - Proporciona un método estructurado para diseñar arquitecturas alineadas con los objetivos estratégicos.
 - Permite gestionar la interoperabilidad entre sistemas empresariales complejos.
 - Es usado por grandes corporaciones para transformación digital.

2. Zachman Framework

El marco Zachman es una matriz de clasificación que organiza la arquitectura en seis preguntas (qué, cómo, dónde, quién, cuándo, por qué) y seis niveles de detalle (desde la visión del planificador hasta la del usuario final).

- **Utilidad en proyectos empresariales:**

- Ayuda a estructurar la documentación de la arquitectura de manera clara.
- Fomenta una visión integral de la organización, evitando vacíos en el diseño.
- Se aplica en proyectos donde se requiere alinear procesos de negocio y tecnología.

3. CMMI (Capability Maturity Model Integration)

CMMI es un modelo que evalúa la madurez y capacidad de procesos de una organización, abarcando desarrollo, servicios y gestión de proyectos.

- **Utilidad en proyectos empresariales:**

- Permite medir y mejorar la calidad de los procesos de software.
- Reduce riesgos al identificar niveles de madurez (del 1 al 5).
- Se utiliza en industrias que buscan certificaciones de calidad y ventaja competitiva en licitaciones

Tema 3: Aplicación práctica de los estándares en proyectos de software

Subtema 3.1: Metodología para aplicar estándares en el diseño arquitectónico

El diseño arquitectónico de software requiere una metodología que garantice la aplicación de estándares internacionales (ISO/IEC 12207, ISO/IEC 25010, IEEE 1471/42010, TOGAF, entre otros). A continuación, se propone un procedimiento paso a paso aplicable a un caso práctico de desarrollo de un sistema académico de gestión estudiantil:

Paso 1. Definición del contexto del proyecto

- Identificar los objetivos del sistema.
- Reconocer a los stakeholders (usuarios, administradores, clientes, reguladores).
- Delimitar el alcance y restricciones del proyecto.
(Referencia: IEEE 1471 / ISO/IEC 42010 para identificación de actores y preocupaciones).

Paso 2. Selección de estándares relevantes

- Determinar qué estándares se aplicarán según el tipo de proyecto:
 - ISO/IEC 12207 → ciclo de vida del software.
 - ISO/IEC 25010 → métricas de calidad (usabilidad, seguridad, mantenibilidad, etc.).
 - IEEE 1471 / ISO/IEC 42010 → descripción y documentación de la arquitectura.
 - TOGAF/Zachman → apoyo metodológico para el diseño empresarial.
 - CMMI → mejora de procesos.

Paso 3. Modelado de procesos y requisitos

- Documentar los procesos de negocio que el sistema debe soportar.
- Traducirlos en requisitos funcionales y no funcionales.
- Relacionar cada requisito con métricas de calidad definidas en ISO/IEC 25010.

Paso 4. Elaboración de la arquitectura inicial

- Crear diagramas arquitectónicos (componentes, capas, vistas lógica y física).
- Utilizar el estándar IEEE 1471/42010 para garantizar la trazabilidad entre requisitos, vistas y stakeholders.
- Definir patrones y estilos arquitectónicos apropiados (cliente-servidor, microservicios, capas, etc.).

Paso 5. Validación con métricas de calidad

- Evaluar el diseño mediante las métricas de ISO/IEC 25010 (ejemplo: tiempo de respuesta < 2s, disponibilidad ≥ 99%).
- Validar el cumplimiento de requisitos no funcionales.

Paso 6. Iteración y mejora

- Aplicar principios de mejora continua (CMMI).

- Ajustar el diseño con retroalimentación de los stakeholders.
- Registrar lecciones aprendidas para proyectos futuros.

Paso 7. Documentación y entrega

- Generar la documentación final de la arquitectura siguiendo IEEE 1471/42010.
- Incluir la justificación de los estándares aplicados.
- Entregar como parte del informe técnico o manual arquitectónico.

Subtema 3.2: Casos de uso de ISO/IEC/IEEE 42010

Caso de uso: Sistema Académico de Gestión Estudiantil (SAGE)

1. Contexto

Una universidad requiere un sistema para la gestión académica de estudiantes: matrículas, calificaciones, reportes, historial académico y control de pagos. El estándar ISO/IEC/IEEE 42010 se aplicará para documentar la arquitectura de software a través de vistas y puntos de vista.

2. Actores principales (Stakeholders)

- **Estudiantes** → Registrar matrícula, consultar notas y pagos.
- **Docentes** → Subir calificaciones, gestionar cursos.
- **Administrativos** → Control de matrículas, generación de reportes.
- **TI de la Universidad** → Seguridad, disponibilidad y mantenimiento.

3. Preocupaciones de los stakeholders

- **Estudiantes:** Acceso rápido, disponibilidad, seguridad de datos.
- **Docentes:** Facilidad de uso, confiabilidad.
- **Administrativos:** Exactitud en procesos, generación de informes.
- **TI:** Escalabilidad, integración con otros sistemas.

(Referencia: ISO/IEC/IEEE 42010 → identificar stakeholders y sus preocupaciones)

4. Vistas arquitectónicas (ejemplo)

- **Vista de contexto:** Relación del sistema con usuarios y sistemas externos (ej. pagos en línea, correo institucional).
- **Vista funcional:** Casos de uso principales (matrícula, consulta de notas, carga de calificaciones).
- **Vista lógica:** Componentes de software (módulo de usuarios, módulo de gestión académica, módulo financiero).
- **Vista física:** Infraestructura (servidores, base de datos, nube).
- **Vista de desarrollo:** Organización del código en capas (presentación, negocio, datos).

5. Ejemplo de Caso de Uso Documentado (Vista funcional)

Caso de uso: Registrar matrícula en línea.

- **Actor principal:** Estudiante.
- **Precondiciones:** El estudiante debe estar habilitado por la oficina académica.
- **Flujo principal:**
 1. El estudiante accede al portal académico.
 2. El sistema valida credenciales.
 3. El estudiante selecciona cursos disponibles.
 4. El sistema valida requisitos y cupos.
 5. Se confirma matrícula y se genera recibo de pago.
- **Postcondiciones:** La matrícula queda registrada en la base de datos.
- **Extensiones:** Pago en línea integrado vía pasarela externa.

(Este caso de uso se relaciona con la vista funcional y lógica de la arquitectura).

6. Evidencia para el informe

- Descripción narrativa del caso de uso.

- Diagramas arquitectónicos (UML de caso de uso + vistas 42010).
- Justificación de cómo el estándar guió la identificación de stakeholders, preocupaciones y vistas.

Subtama 3.3: Aplicación de ISO/IEC 25010 en la evaluación de la calidad

Caso práctico: Evaluación de un prototipo de Sistema Académico de Gestión Estudiantil (SAGE)

El estándar ISO/IEC 25010 define un modelo de calidad compuesto por 8 características principales y sus subcaracterísticas. Estas se aplican a un prototipo académico para validar su calidad.

1. Características de calidad aplicadas

Característica (ISO/IEC 25010)	Subcaracterística	Métrica aplicada	Resultado esperado	Ejemplo de evaluación
Funcionalidad	Compleitud funcional	% de requisitos implementados	≥ 90%	45 de 50 requisitos implementados → 90%
Usabilidad	Facilidad de aprendizaje	Tiempo promedio de uso inicial	< 15 min	Estudiantes aprenden uso básico en 12 min
Confiabilidad	Disponibilidad	Tiempo en línea / Tiempo total	≥ 99%	Servidor disponible 99.2% en pruebas
Rendimiento	Tiempo de respuesta	Tiempo de carga de página	≤ 2 seg	Consulta de notas: 1.8 seg
Seguridad	Confidencialidad	% de accesos restringidos correctamente	100%	Todos los accesos no autorizados bloqueados
Mantenibilidad	Modularidad	Grado independencia entre módulos	Alto (≥ 80%)	Separación clara entre módulos académico y financiero
Compatibilidad	Interoperabilidad	Integración con sistemas externos	Correcta	Integra pasarela de pagos sin errores
Portabilidad	Adaptabilidad	Ejecución en distintos navegadores	100%	Chrome, Edge y Firefox compatibles

2. Resultados obtenidos

- El prototipo alcanzó un **nivel de cumplimiento del 92%** en métricas establecidas.
- Áreas destacadas: **usabilidad y seguridad**.
- Áreas a mejorar: **mantenibilidad** (requiere más documentación técnica) y **rendimiento** (optimizar consultas SQL).

Subtema 3.4: Relación de CMMI con la mejora continua

Simulación de un plan de mejora en un equipo de desarrollo.

1. Contexto

El Capability Maturity Model Integration (CMMI) es un modelo que ayuda a las organizaciones a mejorar sus procesos de desarrollo de software, asegurando calidad, eficiencia y reducción de riesgos. Su relación con la mejora continua se basa en la progresión de niveles de madurez y en la implementación de prácticas de gestión y desarrollo más estandarizadas.

2. Plan de mejora simulado en un equipo académico de desarrollo

Proyecto: Sistema de Gestión de Biblioteca Universitaria.

Equipo: 5 desarrolladores, 1 tester, 1 líder de proyecto.

Paso 1: Diagnóstico inicial

- **Nivel actual (CMMI Nivel 1 – Inicial):** Procesos poco definidos, pruebas realizadas de forma improvisada, documentación limitada.

Paso 2: Objetivos de mejora

- Reducir fallos en producción en un **30%**.
- Aumentar la cobertura de pruebas al **80%**.
- Estandarizar la documentación técnica en todos los módulos.

Paso 3: Estrategias de mejora (alineadas a CMMI)

- **Gestión de requisitos (Nivel 2):** Implementar control de cambios y trazabilidad en requisitos.

- **Planificación de proyectos (Nivel 2):** Definir cronogramas con hitos medibles.
- **Aseguramiento de calidad (Nivel 3):** Aplicar revisiones por pares y pruebas automatizadas.
- **Gestión cuantitativa (Nivel 4):** Usar métricas de defectos y rendimiento del sistema.
- **Optimización (Nivel 5):** Retroalimentación continua y análisis de causas raíz de errores.

Paso 4: Implementación práctica

- Uso de **herramientas:** GitHub (control de versiones), Jira (gestión ágil), SonarQube (métricas de código), Selenium (pruebas automatizadas).
- Reuniones **retrospectivas semanales** para evaluar avances.
- Capacitación del equipo en buenas prácticas de documentación.

Paso 5: Evaluación de resultados

- Defectos reducidos en pruebas finales.
- Mayor transparencia en el estado del proyecto.
- Mayor satisfacción de los stakeholders (usuarios y docentes).

Subtema 3.5: Ejemplos prácticos en proyectos académicos y empresariales

Estudio de Caso 1: Proyecto Académico – Sistema de Gestión de Cursos Universitarios (SGCU)

- **Contexto:** Un grupo de estudiantes de ingeniería de software desarrolla un prototipo para administrar cursos, inscripciones y calificaciones.
- **Estándares aplicados:**
 - **ISO/IEC/IEEE 42010:** Se documentó la arquitectura con vistas (contexto, lógica y física).

- **ISO/IEC 12207:** Se gestionaron las fases del ciclo de vida (análisis, diseño, desarrollo, pruebas).
 - **ISO/IEC 25010:** Se definieron métricas de calidad como usabilidad (tiempo de aprendizaje), seguridad (acceso restringido) y rendimiento (tiempo de respuesta).
- **Resultados:**
 - Se redujo la improvisación en el diseño gracias a la trazabilidad de requisitos.
 - Se evidenció mayor confiabilidad en pruebas funcionales.
 - La documentación permitió replicar el proyecto en otro curso.

Estudio de Caso 2: Proyecto Empresarial – Plataforma de Pagos Digitales (Empresa Peruana Fintech)

- **Contexto:** Una startup peruana implementó una plataforma de pagos digitales para universidades y colegios.
- **Estándares aplicados:**
 - **ISO/IEC 25010:** Se priorizaron atributos de seguridad (confidencialidad de datos financieros) y disponibilidad (99.9% uptime).
 - **CMMI Nivel 3:** La empresa formalizó procesos de gestión de proyectos y calidad.
 - **TOGAF:** Se utilizó como marco de referencia para la alineación entre procesos de negocio y arquitectura tecnológica.
- **Resultados:**
 - Certificación en buenas prácticas de seguridad.
 - Incremento del 20% en satisfacción de clientes debido a mayor confiabilidad.
 - Mejor integración con sistemas académicos mediante APIs.

Tema 4: Estándares y su integración con el ABP

Subtema 4.1: Rol de los estándares en proyectos colaborativos

Dinámica grupal para definir responsabilidades aplicando estándares.

1. Contexto

En proyectos colaborativos de software, los estándares internacionales permiten establecer un marco común para asignar responsabilidades, coordinar actividades y asegurar calidad. Al aplicar normas como ISO/IEC 12207 (ciclo de vida), ISO/IEC/IEEE 42010 (arquitectura) y CMMI (mejora de procesos), se logra que cada miembro del equipo tenga funciones claras, evitando duplicidad de tareas o vacíos en el trabajo.

2. Dinámica grupal (simulación académica)

En un proyecto académico para desarrollar un Sistema de Tutorías Universitarias en Línea, los estudiantes se organizaron aplicando estándares:

- **Líder de Proyecto (Gestión – ISO/IEC 12207):** Define el plan de trabajo, asegura cumplimiento de plazos y coordina el ciclo de vida.
- **Arquitecto de Software (ISO/IEC/IEEE 42010):** Documenta la arquitectura, prepara las vistas arquitectónicas y verifica alineación con los requisitos.
- **Desarrolladores (Buenas prácticas CMMI):** Codifican siguiendo guías de estilo y estándares de calidad del código.
- **Tester (ISO/IEC 25010):** Evalúa atributos de calidad (usabilidad, rendimiento, seguridad) mediante pruebas definidas.
- **Documentador (ISO/IEC 12207):** Redacta manuales, informes y mantiene la trazabilidad.

3. Evidencia para el informe

Acta de roles y responsabilidades (ejemplo):

Rol	Integrante	Estándar aplicado	Responsabilidad clave
Líder de Proyecto	Juan Pérez	ISO/IEC 12207	Planificación y control de actividades
Arquitecto de Software	María Gómez	ISO/IEC/IEEE 42010	Modelado y documentación de arquitectura
Desarrollador Backend	Luis Torres	CMMI / ISO 12207	Implementación de lógica de negocio
Desarrollador Frontend	Ana López	CMMI	Diseño de interfaces y usabilidad
Tester	Carlos Ramos	ISO/IEC 25010	Pruebas de calidad y validación

Documentador	Sofía Ruiz	ISO/IEC 12207	Registro de procesos y manuales
--------------	------------	---------------	---------------------------------

Subtema 4.2: Selección del estándar adecuado

Taller de selección de estándares según un caso propuesto (ejemplo: aplicación móvil de salud).

1. Contexto del caso propuesto

Se plantea el desarrollo de una **app móvil de salud** que permita a los pacientes:

- Registrar y monitorear signos vitales.
- Agendar citas médicas en línea.
- Compartir datos de salud con especialistas de forma segura.
- Integrarse con dispositivos portátiles (wearables).

El equipo de desarrollo debe seleccionar los estándares más adecuados para garantizar seguridad, calidad y confiabilidad del sistema.

2. Estándares seleccionados y justificación

Estándar	Aplicación en el caso	Justificación
ISO/IEC 12207 (Ciclo de vida del software)	Gestión del proceso desde análisis hasta mantenimiento.	Permite un control ordenado y documentado del ciclo de vida del proyecto.
ISO/IEC/IEEE 42010 (Arquitectura de software)	Definición de vistas arquitectónicas (seguridad, interoperabilidad, infraestructura).	Asegura una arquitectura clara que cubra las preocupaciones de los stakeholders.
ISO/IEC 25010 (Calidad del software)	Evaluación de usabilidad, seguridad, rendimiento y confiabilidad.	Fundamental en apps de salud donde la seguridad y confiabilidad son críticas.
ISO/IEC 27001 (Seguridad de la información)	Protección de datos médicos y cumplimiento con normativas de confidencialidad.	Garantiza la integridad y confidencialidad de datos sensibles de pacientes.

HL7 / FHIR (Interoperabilidad en salud)	Intercambio de datos médicos con sistemas hospitalarios y wearables.	Estándares internacionales específicos para la comunicación de información de salud.
CMMI (Mejora de procesos)	Formalización de buenas prácticas de gestión y calidad en el equipo de desarrollo.	Permite alcanzar un nivel de madurez en la gestión de procesos.

Subtema 4.3: Documentación arquitectónica como evidencia en ABP

Generación de documentos arquitectónicos como entregables del proyecto.

1. Contexto

En el Aprendizaje Basado en Proyectos (ABP), la documentación arquitectónica es clave porque constituye la evidencia tangible del trabajo colaborativo y asegura la trazabilidad entre los requisitos del cliente, las decisiones de diseño y los resultados finales. Aplicando estándares como ISO/IEC/IEEE 42010 y ISO/IEC 12207, los equipos pueden estructurar su documentación de manera clara, uniforme y reutilizable.

2. Documentos arquitectónicos generados (ejemplo de un proyecto académico: “Sistema de Tutorías Universitarias en Línea”)

1. Documento de requisitos arquitectónicos

- Stakeholders identificados.
- Preocupaciones principales (seguridad, escalabilidad, disponibilidad).
- Restricciones técnicas y de negocio.

2. Vistas arquitectónicas (ISO/IEC/IEEE 42010)

- **Vista de contexto:** relaciones con usuarios y sistemas externos.
- **Vista lógica:** módulos del sistema (gestión de tutorías, registro de usuarios, notificaciones).
- **Vista física:** servidores, base de datos, nube.

- **Vista de desarrollo:** organización del código en capas.

3. Plan de ciclo de vida (ISO/IEC 12207)

- Procesos seguidos en el desarrollo (análisis, diseño, construcción, pruebas, mantenimiento).
- Roles y responsabilidades de cada integrante.

4. Evaluación de calidad (ISO/IEC 25010)

- Matriz de atributos evaluados (usabilidad, seguridad, mantenibilidad).
- Resultados de pruebas del prototipo.

5. Anexos documentales

- Diagramas UML (casos de uso, clases, secuencia).
- Cronograma de actividades.
- Actas de reuniones y acuerdos del equipo.

Subtema 4.4: Presentación y discusión de proyectos aplicando estándares

Exposición final con discusión de resultados y aprendizajes.

1. Contexto

La fase final de un proyecto bajo la metodología de Aprendizaje Basado en Proyectos (ABP) consiste en la exposición y discusión de resultados. En este punto, los estudiantes no solo presentan el producto final, sino que evidencian el uso de estándares internacionales (ISO, IEEE, CMMI, ITIL, entre otros) en el desarrollo, implementación y validación de la solución.

La discusión permite evaluar el nivel de dominio alcanzado, así como la reflexión crítica respecto a:

- Los beneficios de aplicar estándares.
- Las limitaciones encontradas durante el proceso.
- Las propuestas de mejora para futuros proyectos.

2. Estrategia de la presentación

- **Parte 1 – Introducción:** Contextualizar el proyecto y los estándares aplicados.
- **Parte 2 – Desarrollo:** Explicar cómo se integraron los estándares en cada fase del ciclo de vida del software.
- **Parte 3 – Resultados:** Evidencias técnicas (diagramas, métricas de calidad, pruebas).
- **Parte 4 – Discusión:** Reflexión crítica del equipo sobre el impacto de los estándares en la calidad y en la colaboración.
- **Parte 5 – Conclusiones:** Principales aprendizajes y recomendaciones.