

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

Subtema 1.1: Definición de arquitectura de software

Actividad ABP: Investigar distintas definiciones en fuentes normativas (ISO/IEC/IEEE 42010, Pressman, Bass, etc.) y compararlas.

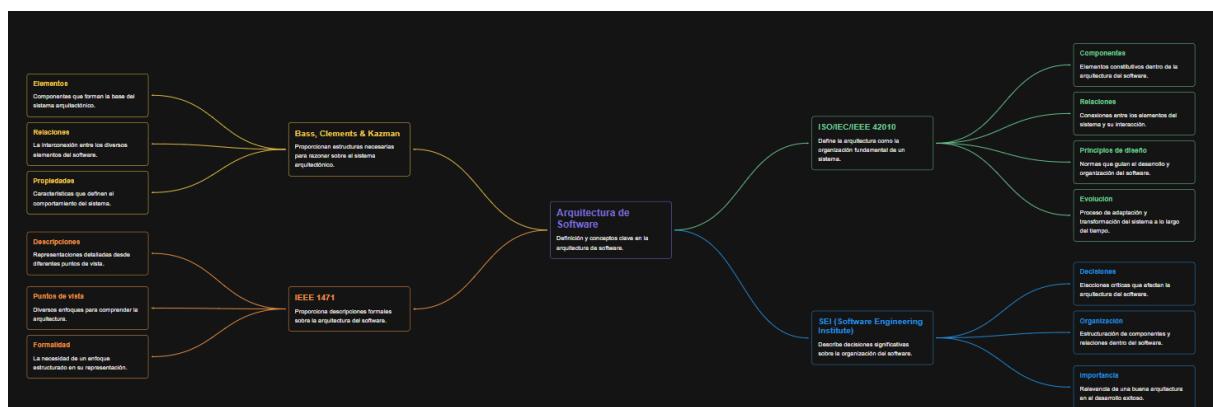
Evidencia:

Mapa conceptual (texto):

- **ISO/IEC/IEEE 42010:** La arquitectura es la **organización fundamental de un sistema** en sus componentes, sus relaciones y su entorno.
- **Bass, Clements & Kazman:** Arquitectura = **estructura del sistema** que incluye elementos, relaciones y principios de diseño.
- **Pressman:** Arquitectura = **patrón de organización** de un software que guía diseño, evolución y evaluación.

Comparación:

- Todas resaltan **estructura y relaciones**.
- ISO resalta **normativa y entorno**.
- Bass enfatiza **principios de diseño**.
- Pressman vincula con **evolución del sistema**.



Subtema 1.2: Diferencias entre diseño y arquitectura de software

Actividad ABP: Analizar una aplicación web educativa.

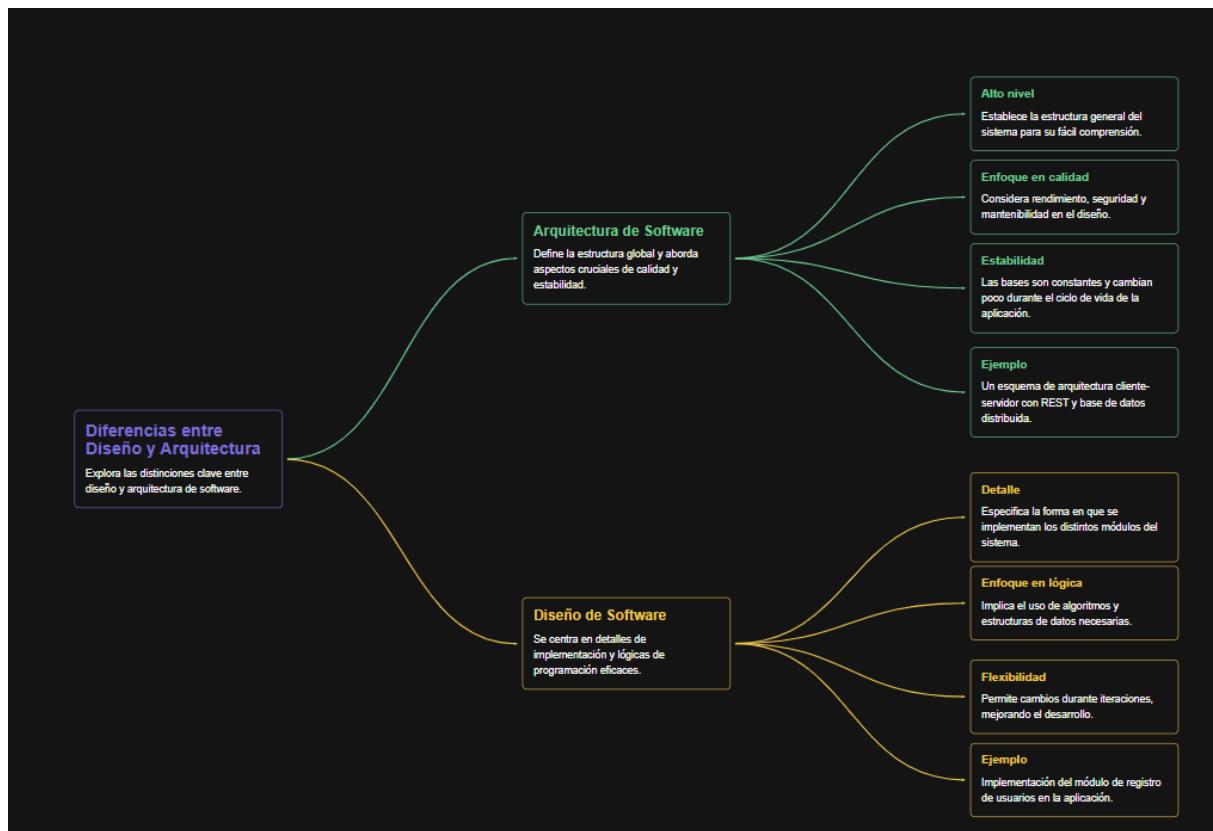
Evidencia:

Documento técnico (resumen):

- **Arquitectura:**
 - División en capas (presentación, negocio, datos).
 - Definición de tecnologías (Angular, API REST, SQL Server).
 - Patrones globales (MVC, microservicios).
- **Diseño:**
 - Interfaces gráficas (botones, menús, dashboards).
 - Diagramas de clases de módulos específicos.
 - Estilos CSS y organización del frontend.

Mapa conceptual (texto):

- **Arquitectura:** visión macro, decisiones estratégicas, impacto en calidad.
- **Diseño:** visión micro, decisiones tácticas, detalles de implementación.
- Relación: el diseño **implementa** lo definido en la arquitectura.



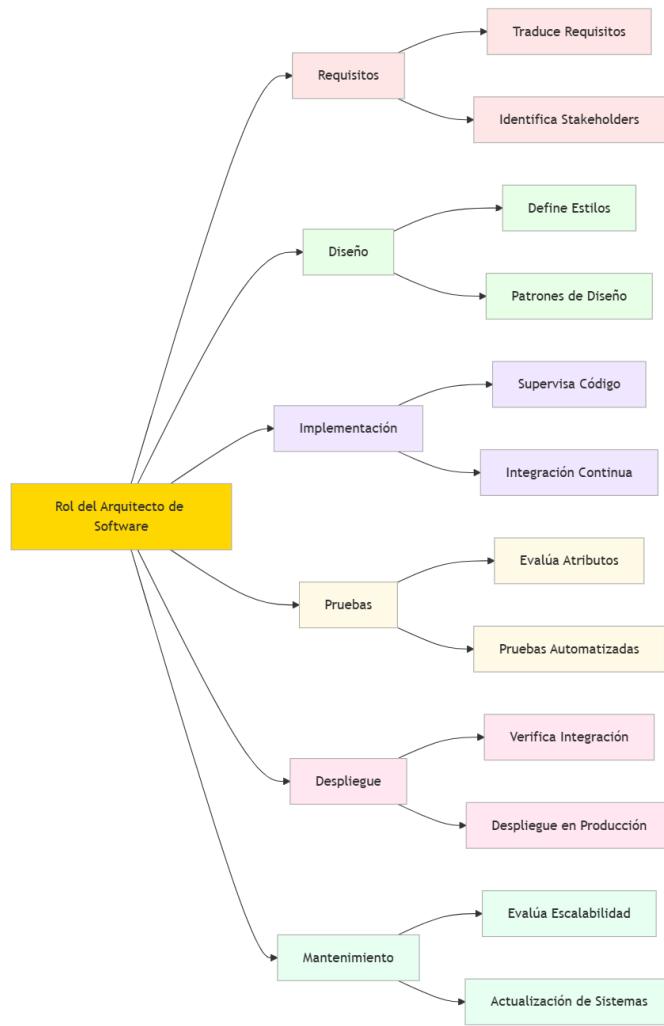
Subtema 1.3: Rol del arquitecto de software en el ciclo de vida

Actividad ABP: Simular el rol de arquitecto en un proyecto.

Evidencia:

Mapa conceptual (texto):

- **Requisitos:** Identificar requerimientos funcionales y no funcionales.
- **Diseño:** Definir estilos arquitectónicos, patrones, modelos de componentes.
- **Pruebas:** Validar que la arquitectura soporte escalabilidad, seguridad y rendimiento.
- **Mantenimiento:** Asegurar evolución, documentación y adaptabilidad.



Subtema 1.4: Impacto de la arquitectura en calidad y eficiencia

Actividad ABP: Evaluar un sistema real (ejemplo: biblioteca digital).

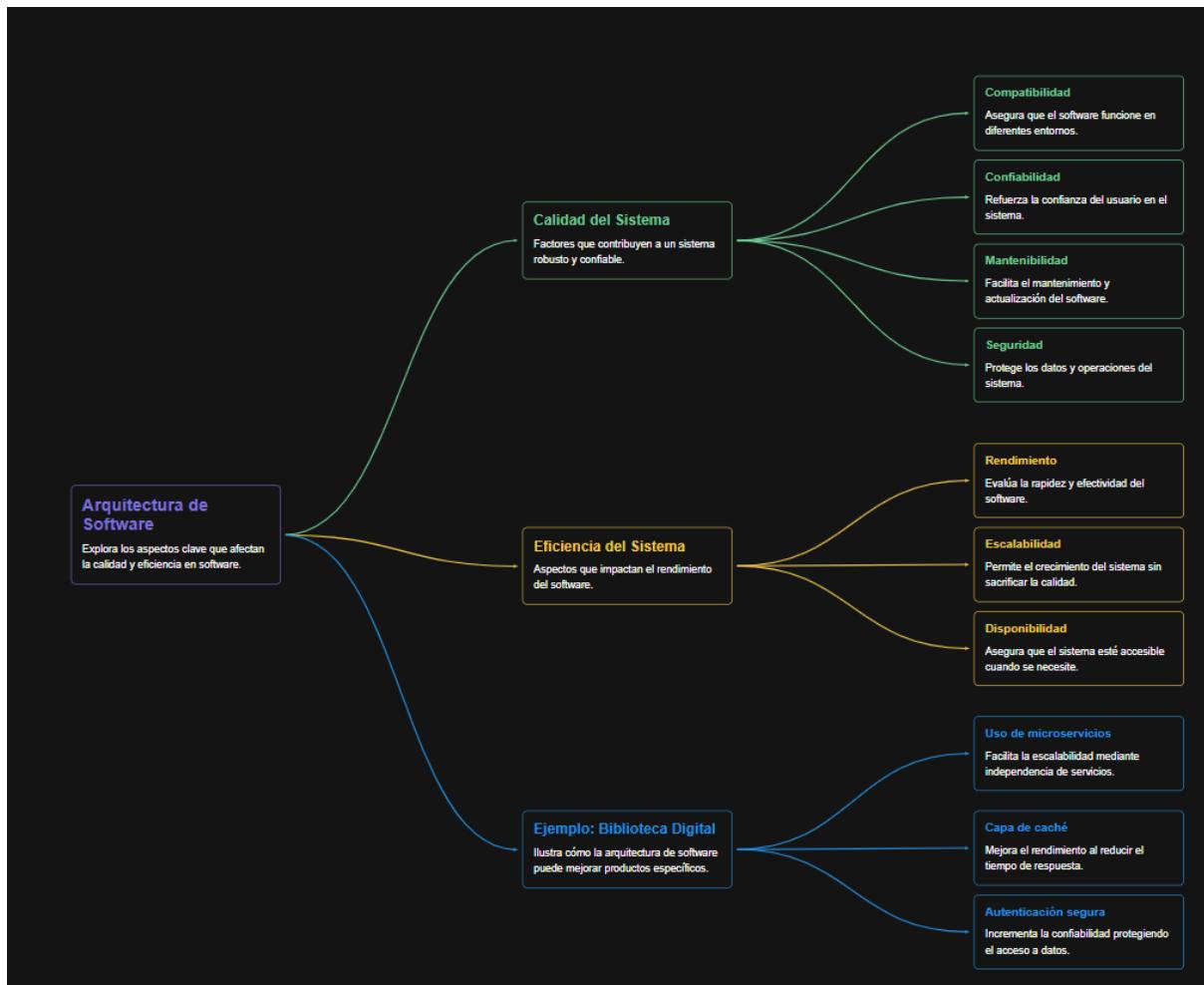
Evidencia:

Análisis escrito (resumen):

- **Calidad:** Arquitectura en capas permite fácil mantenibilidad y escalabilidad.
- **Eficiencia:** Uso de caché + balanceo de carga mejora tiempos de respuesta.
- **Impacto:** Una arquitectura bien definida reduce fallos, facilita mejoras y soporta crecimiento. EN LA PARTE DEL LOGO QUE VAYA UNA FOTO

Mapa conceptual (texto):

- Nodo central: “Impacto de la Arquitectura”
 - **Calidad:** mantenibilidad, usabilidad, seguridad.
 - **Eficiencia:** rendimiento, consumo de recursos.
 - **Resultado:** usuarios satisfechos + menor costo de mantenimiento.



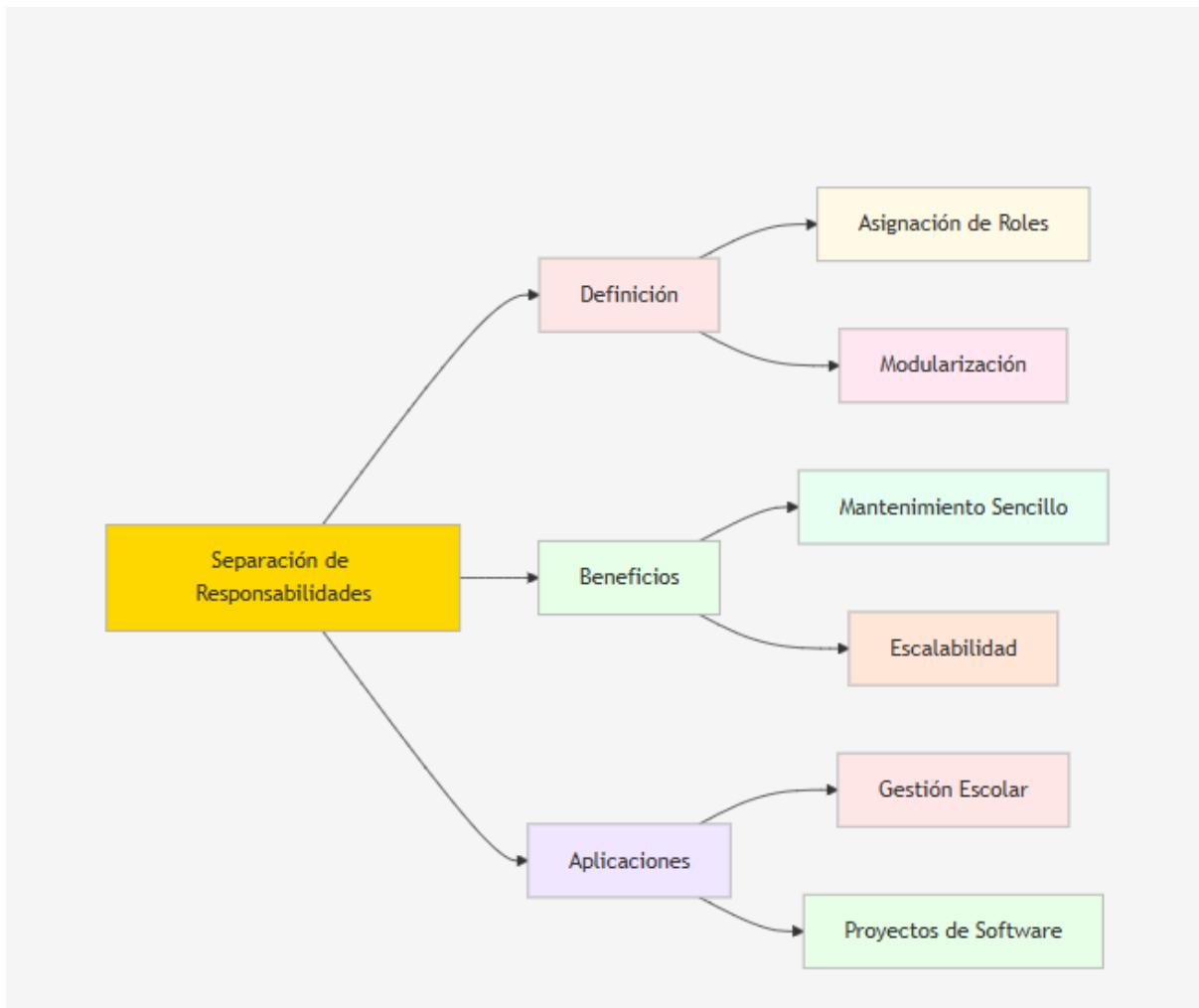
Subtema 2.1: Principios de separación de responsabilidades

Actividad ABP: Desarrollar un esquema arquitectónico para una aplicación de gestión escolar aplicando este principio.

Evidencia:

Mapa Conceptual (texto descriptivo):

- **Separación de responsabilidades** → Divide el sistema en capas/módulos.
- **Capa de presentación (UI)**: Interacción con el usuario (docentes, alumnos, administradores).
- **Capa de lógica de negocio**: Procesa reglas escolares (matrículas, notas, asistencia).
- **Capa de datos**: Gestión de base de datos de estudiantes, cursos, docentes.
- **Beneficios**:
 - Mantenibilidad.
 - Reutilización.
 - Escalabilidad.
 - Reducción de dependencias.



Subtema 2.2: Cohesión y acoplamiento en la arquitectura

Actividad ABP: Analizar un sistema académico y detectar módulos con alto acoplamiento, proponiendo mejoras.

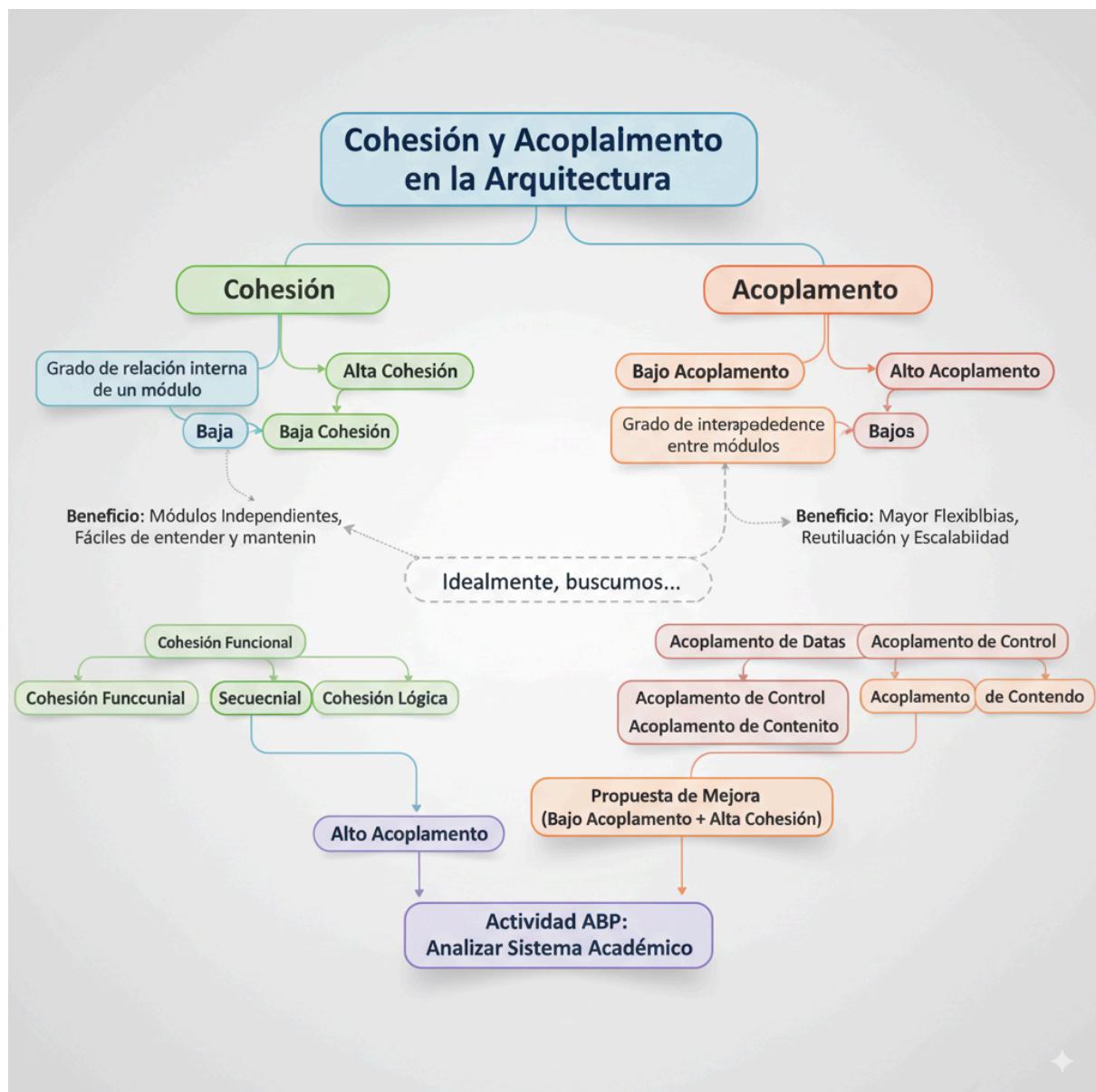
Evidencia:

Documento crítico (breve):

- Problema detectado: El módulo de notas depende directamente del módulo de asistencia → alto acoplamiento.
- Propuesta: Introducir una interfaz de servicios compartidos (API interna) para desacoplar funcionalidades.

Mapa Conceptual:

- **Cohesión:** Cada módulo tiene una función clara.
- **Acoplamiento:** Nivel de dependencia entre módulos.
- **Problema:** Alto acoplamiento = baja flexibilidad.
- **Solución:** Interfaces → APIs → microservicios → menor dependencia.



Subtema 2.3: Abstracción, modularidad y reutilización

Actividad ABP: Diseñar módulos reutilizables para un sistema de e-commerce.

Evidencia:

Mapa Conceptual con ejemplos:

- **Abstracción:** Ocultar detalles → Ejemplo: “Gestor de pago” (tarjeta, PayPal, transferencia).
- **Modularidad:** Dividir en módulos independientes → Ejemplo: carrito de compras, catálogo, usuarios.
- **Reutilización:** Usar los módulos en otros sistemas → Ejemplo: “Módulo de autenticación” aplicable a diferentes apps.

Abstracción, Modularidad y Reutilización



Subtema 2.4: Escalabilidad, mantenibilidad y confiabilidad

Actividad ABP: Evaluar cómo un sistema de streaming podría escalar frente al crecimiento de usuarios.

Evidencia:

Informe técnico (resumen):

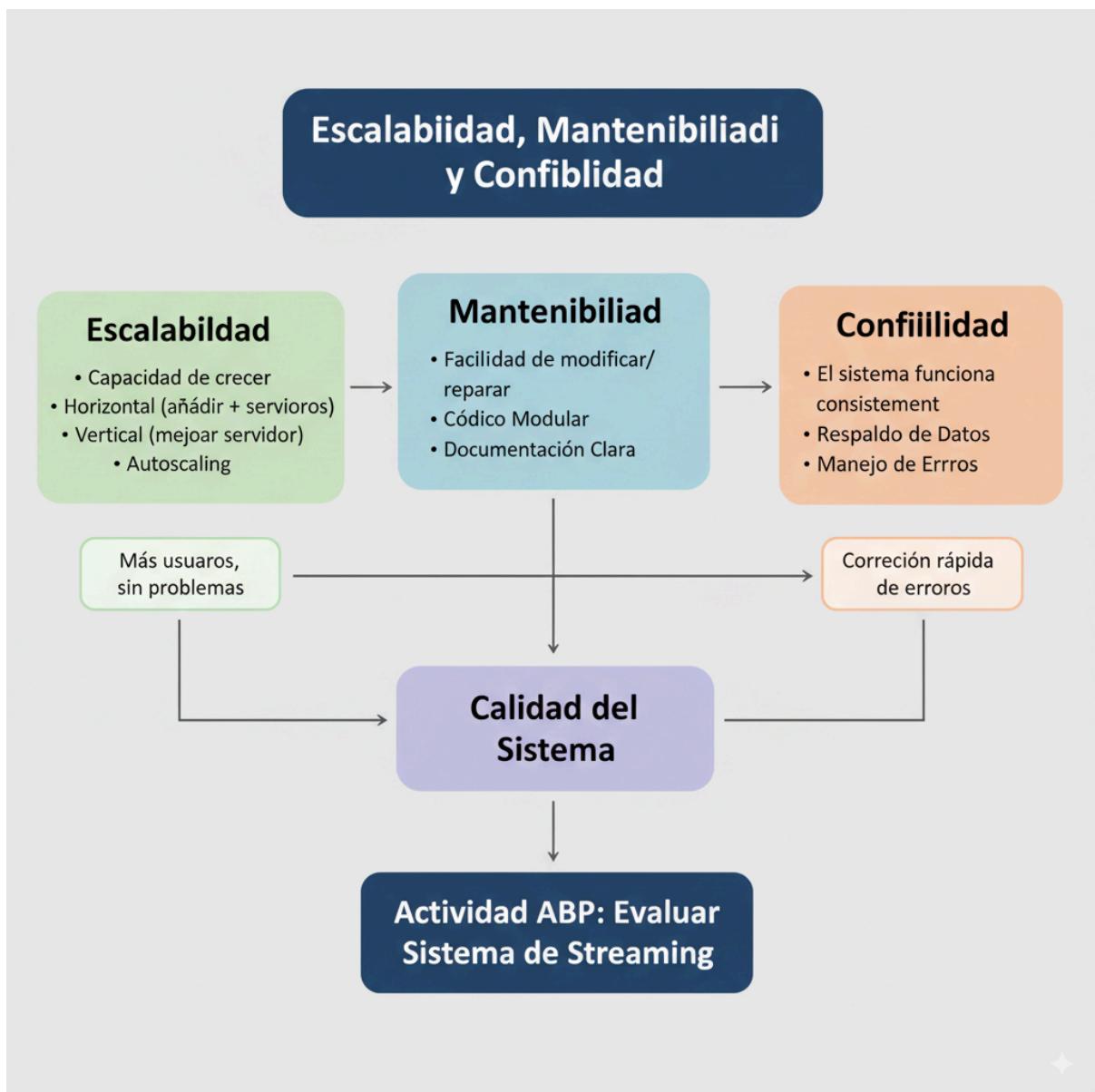
- **Escalabilidad:** Uso de balanceadores de carga, CDN y microservicios.
- **Mantenibilidad:** Código limpio, pruebas automáticas, documentación.

- **Confiabilidad:** Tolerancia a fallos, redundancia de servidores, monitoreo.

Mapa conceptual:

- **Calidad del sistema:**

- Escalabilidad → más usuarios sin pérdida de rendimiento.
- Mantenibilidad → facilidad de actualización.
- Confiabilidad → disponibilidad y tolerancia a fallos.



Tema 3: Estándares Internacionales en Arquitectura de Software

Subtema 3.1 ISO/IEC/IEEE 42010:2011

Objetivo de aprendizaje: Comprender los elementos clave que exige la norma para describir una arquitectura y poder aplicarlos en una descripción arquitectónica formal.

Actividad ABP (encargó): Elaborar un análisis de los elementos clave que exige la norma para describir una arquitectura.

Indicaciones para el trabajo:

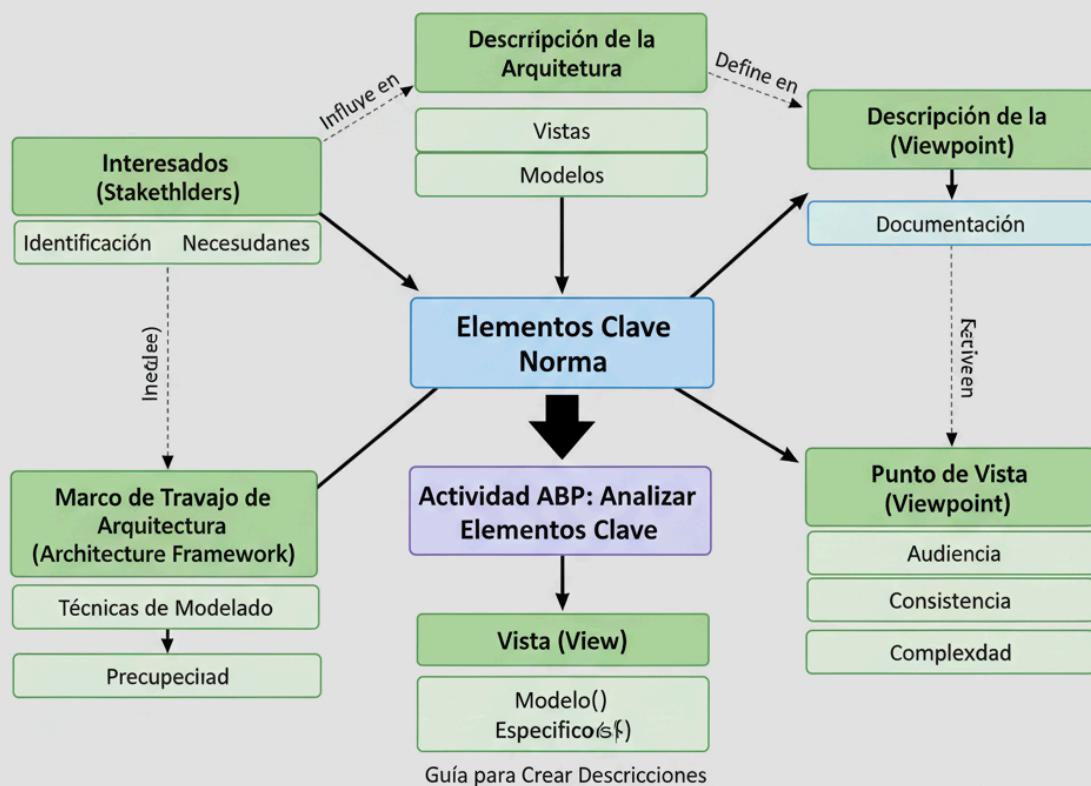
- Investigar la norma ISO/IEC/IEEE 42010:2011 (arquitectura de sistemas y software — descripción de la arquitectura).
- Identificar y describir los siguientes elementos obligatorios según la norma:
 - Stakeholders (interesados)
 - Concerns (preocupaciones)
 - Architecture description (descripción de arquitectura)
 - Views y Viewpoints (vistas y puntos de vista)
 - Correspondence rules (reglas de correspondencia)
 - Rationale (justificación)
 - Model kinds (tipos de modelos)
- Para cada elemento, explique su propósito, formato recomendado y un ejemplo aplicado a un sistema concreto (por ejemplo, sistema de ventas).

Evidencia requerida: Mapa conceptual normativo que muestre relaciones entre stakeholders, concerns, viewpoints, y artefactos de la descripción arquitectónica.

Plantilla sugerida para el análisis (tabla breve):

- Columna A: Elemento (Stakeholders, Concerns, Viewpoint, View, Model Kind, Correspondence, Rationale)
- Columna B: Definición según 42010
- Columna C: Ejemplo en un sistema (1–2 oraciones)
- Columna D: Artefactos/diagramas recomendados

ISO/IEC/IMEE 42010:2011 Mapa Conceptual Normativo



Evidencia: Mapa Conceptual Normativo

Subtema 3.2 ISO/IEC 25010 (Calidad del software)

Objetivo de aprendizaje: Conocer las características de calidad (producto de software) y poder compararlas y aplicarlas al análisis de un sistema.

Actividad ABP (encargó): Crear un cuadro comparativo de las características de calidad y un mapa conceptual explicativo.

Características principales (ISO/IEC 25010):

- Functional suitability
- Performance efficiency
- Compatibility
- Usability
- Reliability
- Security

- Maintainability
- Portability

Instrucciones para el cuadro comparativo:

- Filas: Características de calidad (cada una de las 8/ subcaracterísticas si se quiere mayor detalle)
- Columnas: Definición, Métricas / Indicadores posibles, Técnicas de evaluación (pruebas, revisiones, herramientas), Ejemplo práctico (sistema de ventas / agenda digital)



Subtema 3.3 Marcos de referencia internacionales (TOGAF, Zachman)

Objetivo de aprendizaje: Entender los enfoques de TOGAF y Zachman y saber proponer su aplicación en un caso real.

Actividad ABP (encargó): Analizar un caso real (empresa pública o privada) y proponer cómo aplicaría TOGAF o Zachman.

Instrucciones:

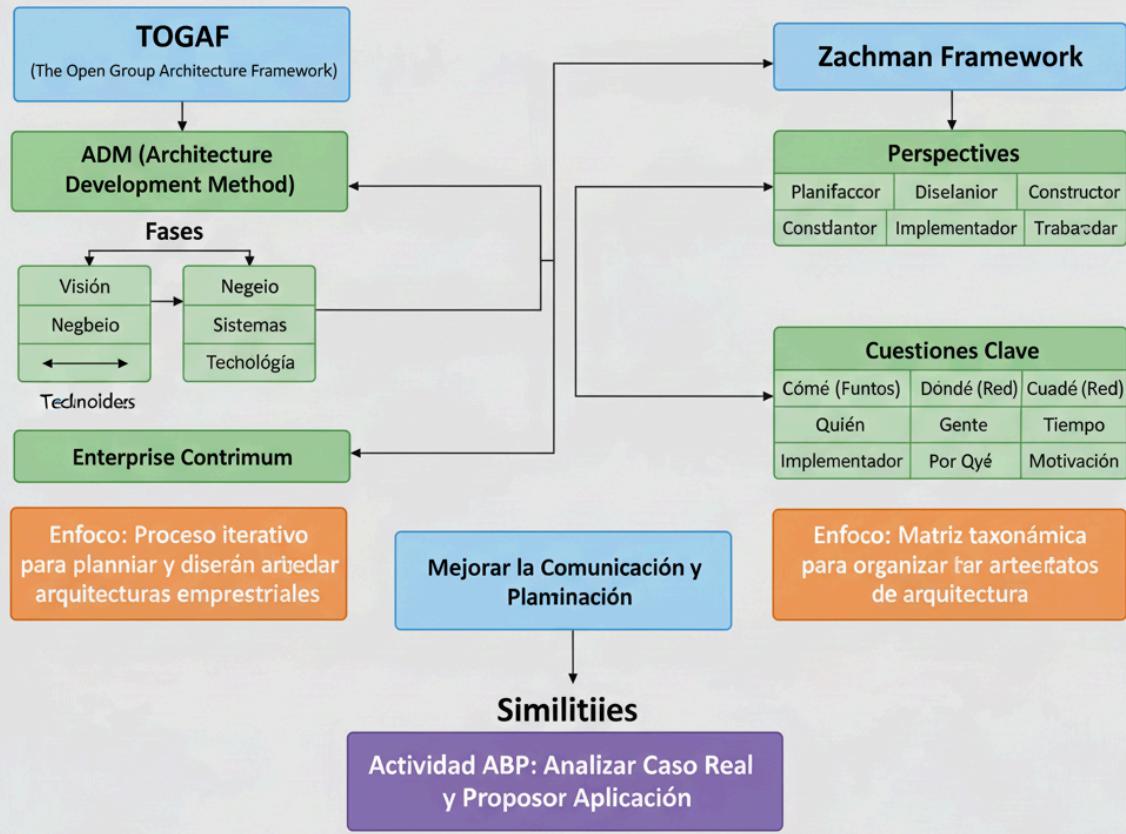
- Seleccionar un caso real (ej.: municipalidad local, empresa de transporte, clínica privada, fintech pequeña).
- Para **TOGAF**: elaborar un plan de aplicación breve que incluye: Preliminary, Architecture Vision, Business Architecture, Information Systems Architecture, Technology Architecture, Opportunities & Solutions, Migration Planning, Implementation Governance, Architecture Change Management. Identificar artefactos clave a producir en cada fase.
- Para **Zachman**: mapear elementos del caso en la matriz Zachman (What, How, Where, Who, When, Why) por cada perspectiva (Planner, Owner, Designer, Builder, Subcontractor, Functioning Enterprise).
- Comparar ventajas y límites de aplicar TOGAF vs Zachman en el caso escogido.

Evidencia requerida: Documento técnico que contenga el análisis y un mapa conceptual mostrando correspondencias entre el marco aplicado y partes del negocio.

Plantilla de entrega:

- Introducción al caso
- Relevamiento de stakeholders y concerns
- Selección del marco (TOGAF o Zachman) y justificación
- Plan resumido de trabajo/artefactos por fase
- Riesgos y supuestos
- Conclusión y recomendaciones

Marcos de Referencia Internacionales (TOGAF, Zachman)



Evidencia: Documento Técnico + Mapa Conceptual

Subtema 3.4 Importancia de la estandarización en proyectos de software

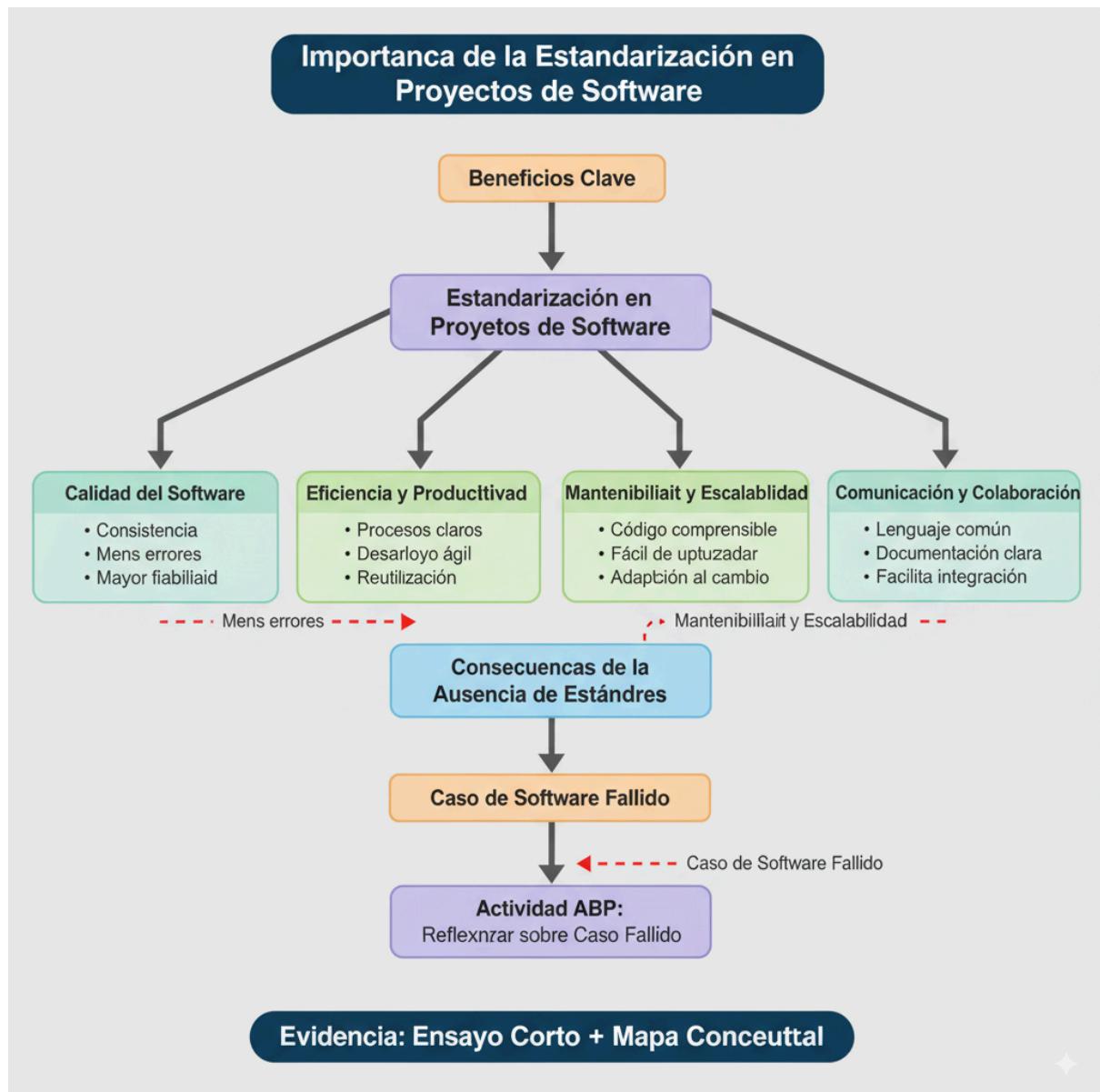
Objetivo de aprendizaje: Reflexionar sobre el rol de estándares en la calidad, mantenimiento y escalabilidad de proyectos de software.

Actividad ABP (encargo): Reflexionar en equipos sobre un caso de software fallido por ausencia de estándares.

Instrucciones:

- Buscar un caso (real o mediático) de fallo por falta de estandarización (p. ej. problemas de compatibilidad, problemas de seguridad, falta de gobernanza arquitectónica).

- Analizar causas raíz relacionadas con ausencia de estándares (documentación pobre, ausencia de procesos formales, arquitectura no definida).
- Proponer medidas estándar (normas, plantillas, revisiones de arquitectura, políticas de control de cambios) que hubieran mitigado el problema.



Tema 4: Estilos y Patrones Arquitectónicos

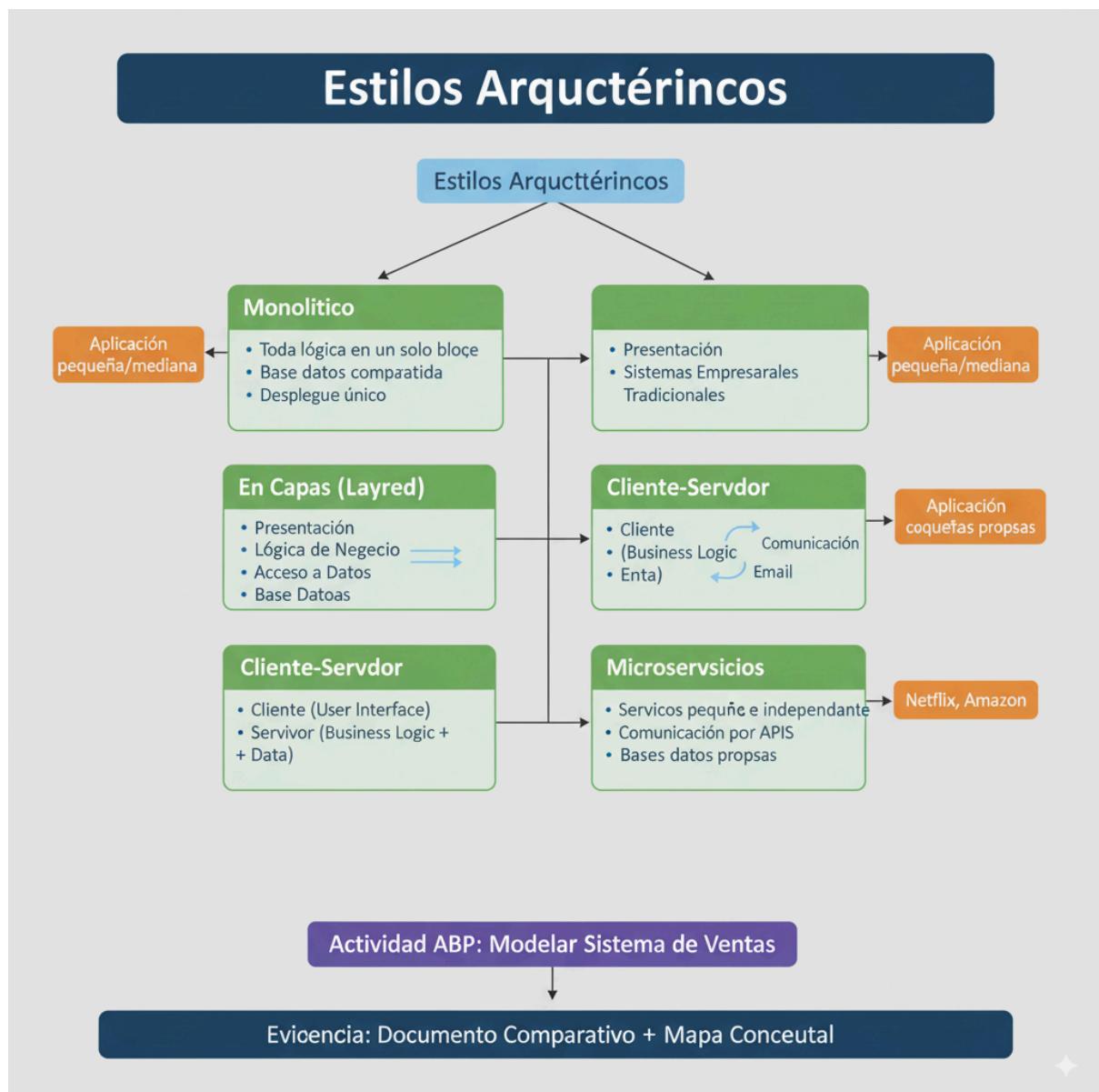
Subtema 4.1 Estilos arquitectónicos (monolítico, en capas, cliente-servidor, microservicios)

Objetivo de aprendizaje: Identificar y comparar estilos arquitectónicos y comprender cuándo aplicar cada uno.

Actividad ABP (encargo): Modelar un mismo sistema (ej. sistema de ventas) en al menos dos estilos arquitectónicos.

Instrucciones:

- Elegir un sistema simple (Sistema de ventas, Sistema de reservas, Agenda digital).
- Modelar en dos estilos distintos (por ejemplo: Monolito en capas y Microservicios; Cliente-Servidor vs En capas).
- Entregar: diagramas (componentes y despliegue), lista de trade-offs (rendimiento, complejidad, despliegue, escalabilidad, coste de operación), y plan de migración (si se pasa de uno a otro).



Subtema 4.2 Patrones arquitectónicos más utilizados (MVC, MVVM, SOA, Event-Driven)

Objetivo de aprendizaje: Conocer patrones comunes y aplicarlos en un prototipo sencillo.

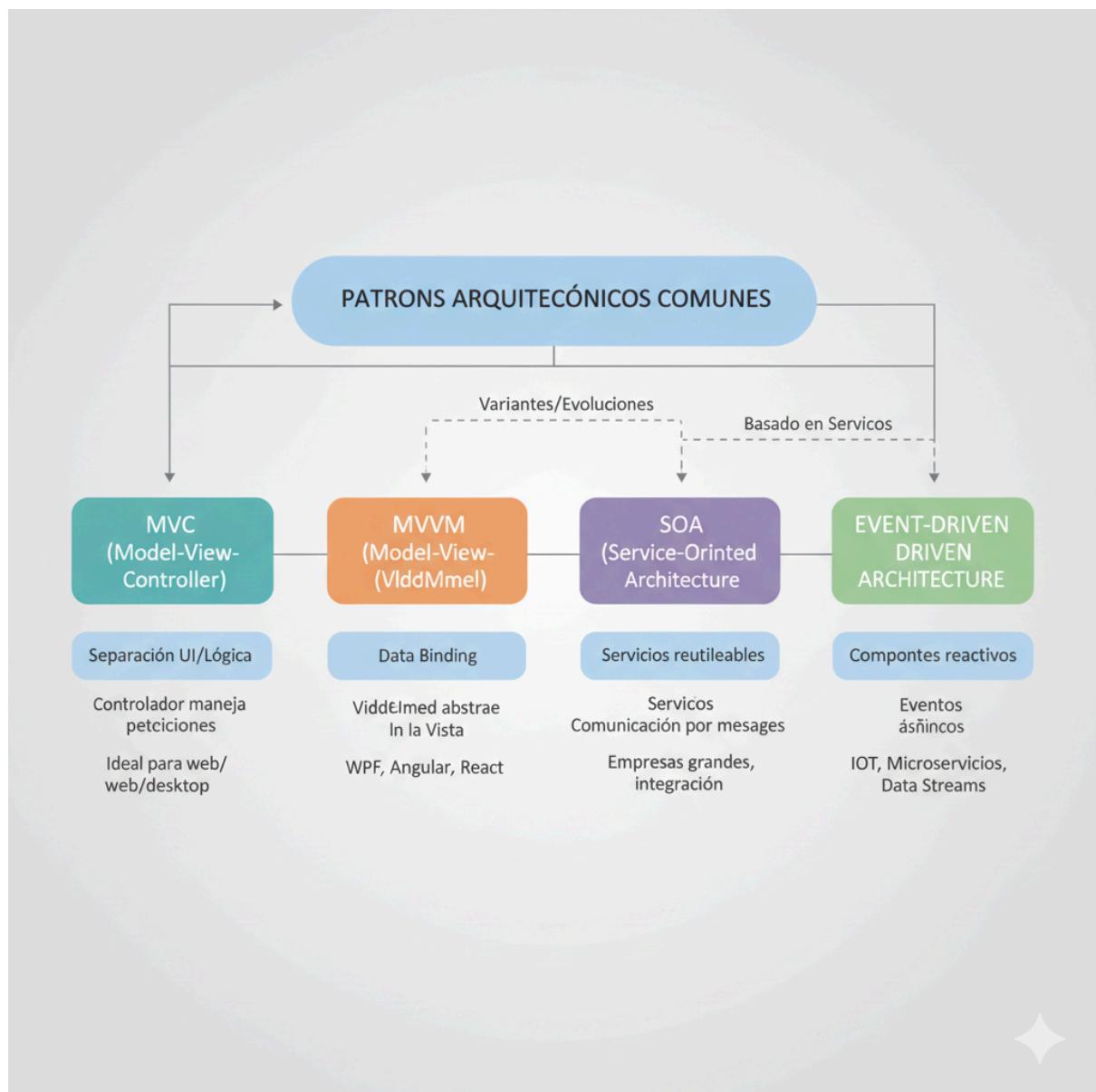
Actividad ABP (encargo): Implementar un prototipo sencillo (ej. agenda digital) aplicando MVC.

Instrucciones del prototipo MVC (mínimo):

- Lenguaje recomendado: JavaScript (Node.js + Express para backend) o Python (Flask) y HTML/CSS/JS sencillo en frontend.
- Requisitos mínimos: CRUD de contactos, validaciones básicas, separación clara entre Model (acceso a datos), View (UI) y Controller (lógica).
- Opcional: pruebas unitarias para el Controller y Model.

Checklist de evaluación del prototipo:

- Correcta separación MVC
- Funcionalidad CRUD completa
- Documentación breve (README) con instrucciones para ejecutar
- Capturas de pantalla y mapa conceptual que muestre relación Modelo-Vista-Controlador



Subtema 4.3 Criterios de selección de un estilo o patrón arquitectónico

Objetivo de aprendizaje: Aprender a seleccionar el estilo/patrón más adecuado según requisitos nofuncionales y contexto del proyecto.

Actividad ABP (encargó): Estudiar tres proyectos ficticios y definir cuál patrón es el más adecuado para cada uno, justificando.

Criterios a considerar (checklist):

- Escalabilidad y volumen esperado
- Tiempo de entrega y coste inicial

- Habilidades del equipo
- Requerimientos de disponibilidad y tolerancia a fallos
- Necesidades de despliegue/infraestructura
- Requerimientos de consistencia y transaccionalidad
- Mantenimiento y gobernanza

Proyectos Ficticios de Ejemplo:

Proyecto 1: Sistema de gestión de inventario para una PyME local.

1.1 Descripción: Una pequeña empresa necesita un sistema para registrar productos, controlar el stock, y gestionar pedidos. El volumen de datos es bajo-medio.

1.2 Requisitos No Funcionales Clave: Bajo costo inicial, rápido tiempo de entrega, facilidad de mantenimiento para un equipo pequeño, alta disponibilidad durante el horario laboral.

1.3 Análisis:

Escalabilidad: No se espera un alto volumen inicialmente.

Tiempo/Costo: Prioridad en la entrega rápida y bajo costo.

Equipo: Desarrolladores con experiencia en monolitos o MVC.

Disponibilidad: Necesaria durante horas de oficina.

Consistencia: Alta, las transacciones de inventario deben ser precisas.

Mantenimiento: Debe ser sencillo.

Decisión de Patrón: MVC o Monolito bien estructurado.

Justificación: Ofrece un buen equilibrio entre rapidez de desarrollo, facilidad de mantenimiento y costos iniciales bajos, adecuado para el volumen y las necesidades de una PyME.

Proyecto 2: Plataforma de streaming de video a gran escala (similar a Netflix en sus inicios).

2.1 Descripción: Una startup busca lanzar una plataforma para miles o millones de usuarios, con contenido diverso y recomendaciones personalizadas.

2.2 Requisitos No Funcionales Clave: Alta escalabilidad, alta disponibilidad (24/7), baja latencia, resiliencia a fallos, despliegue continuo.

2.3 Análisis:

Escalabilidad: Crítica, se espera un alto volumen.

Tiempo/Costo: Mayor inversión inicial pero necesaria para el crecimiento.

Equipo: Se requerirá un equipo más grande y especializado.

Disponibilidad: 24/7, tolerancia a fallos.

Consistencia: Eventual para ciertas partes (recomendaciones), pero fuerte para facturación.

Mantenimiento: Necesidad de independencia de equipos y despliegues.

Decisión de Patrón: Microservicios.

Justificación: Permite escalar componentes de forma independiente, facilita el despliegue continuo y mejora la resiliencia en un entorno de alto tráfico y demanda constante.

Proyecto 3: Sistema de monitoreo de sensores IoT para una fábrica.

3.1 Descripción: Recopilación y procesamiento de datos en tiempo real de miles de sensores en una fábrica para detectar anomalías y disparar alertas.

3.2 Requisitos No Funcionales Clave: Procesamiento de eventos en tiempo real, baja latencia en la detección de anomalías, alta disponibilidad del sistema de monitoreo, capacidad de manejar picos de eventos.

3.3 Análisis:

Escalabilidad: Necesidad de manejar un flujo constante y picos de eventos.

Tiempo/Costo: Inversión en infraestructura de eventos.

Equipo: Experiencia en sistemas distribuidos y procesamiento de eventos.

Disponibilidad: Crítica para la operación de la fábrica.

Consistencia: Importante para la detección de anomalías, pero el flujo de eventos es lo primordial.

Mantenimiento: Requiere herramientas de monitoreo específicas para sistemas de eventos.

Decisión de Patrón: Arquitectura Event-Driven.

Justificación: Ideal para procesar grandes volúmenes de datos en tiempo real, reaccionar a eventos de manera asíncrona y escalar los consumidores de eventos de forma independiente.

CRITERIOS DE SELECCIÓN DE PATRONES ARQUITÉTÓNICOS



Subtema 4.4 Beneficios de aplicar patrones reconocidos en la eficiencia del sistema

Objetivo de aprendizaje: Extraer lecciones aprendidas de casos de éxito y relacionarlas con beneficios medibles.

Actividad ABP (encargó): Analizar un caso de éxito como Netflix y extraer beneficios obtenidos por el uso de microservicios.

Puntos clave a analizar (ejemplo Netflix):

- Independencia de despliegue por servicio
- Escalado horizontal de componentes críticos
- Resiliencia y tolerancia a fallos (circuit breakers, retries)
- Observabilidad (logs, métricas, tracing) y despliegue continuo

- Trade-offs: complejidad operativa, consistencia distribuida



Tema 5: Impacto de la Arquitectura de Software en Sistemas Reales

Subtema 5.1: La arquitectura como guía para el desarrollo y mantenimiento

Actividad ABP

- Simular el desarrollo de un sistema (ejemplo: sistema de gestión de biblioteca, e-commerce o reservas en línea).
- Documentar cómo las **decisiones arquitectónicas** (patrones, capas, estilos) guían tanto el desarrollo como el mantenimiento.

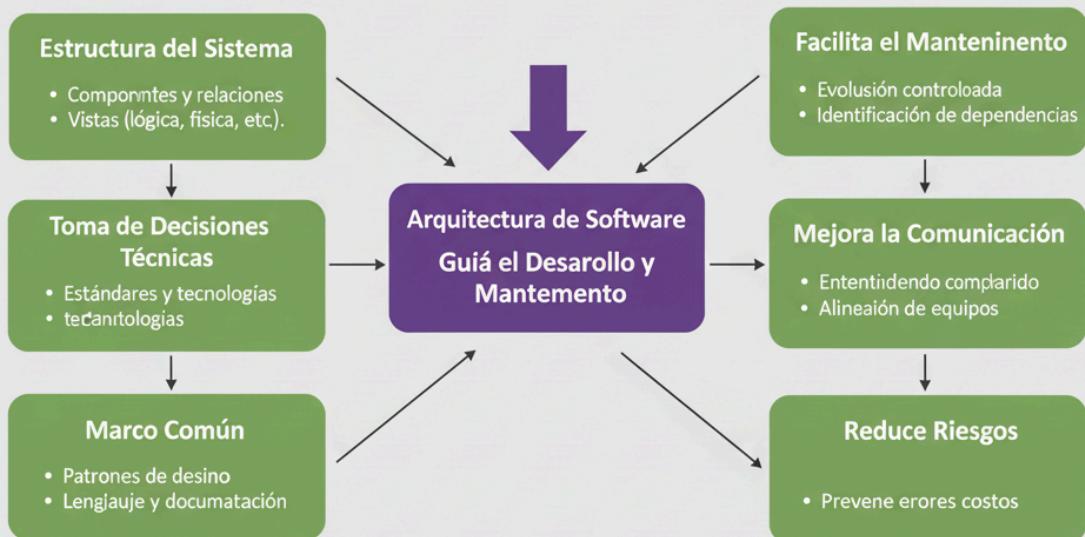
Evidencia

1. Informe técnico:

- Definición del sistema simulado.
- Descripción de las decisiones arquitectónicas tomadas (patrones, estilos, tecnologías).
- Explicación de cómo dichas decisiones impactan en el mantenimiento futuro.
- Justificación técnica de la selección de la arquitectura.

La Arquitectura como Guía para la Desarrollo y Mantenimiento

Subtema 5.1



Actividad ABP: Simular Desarrollo + Documentar

Evidencia: Informe Técnico + Mapa Conceptual



Subtema 5.2: Impacto en la eficiencia, escalabilidad y seguridad del sistema

Actividad ABP

- Estudiar un **sistema bancario** (ejemplo: banca en línea, cajeros automáticos, apps móviles).
- Analizar cómo la arquitectura garantiza:
 - **Eficiencia** (rendimiento, tiempos de respuesta).

- **Escalabilidad** (capacidad de atender más usuarios).
- **Seguridad** (protección de datos y transacciones).

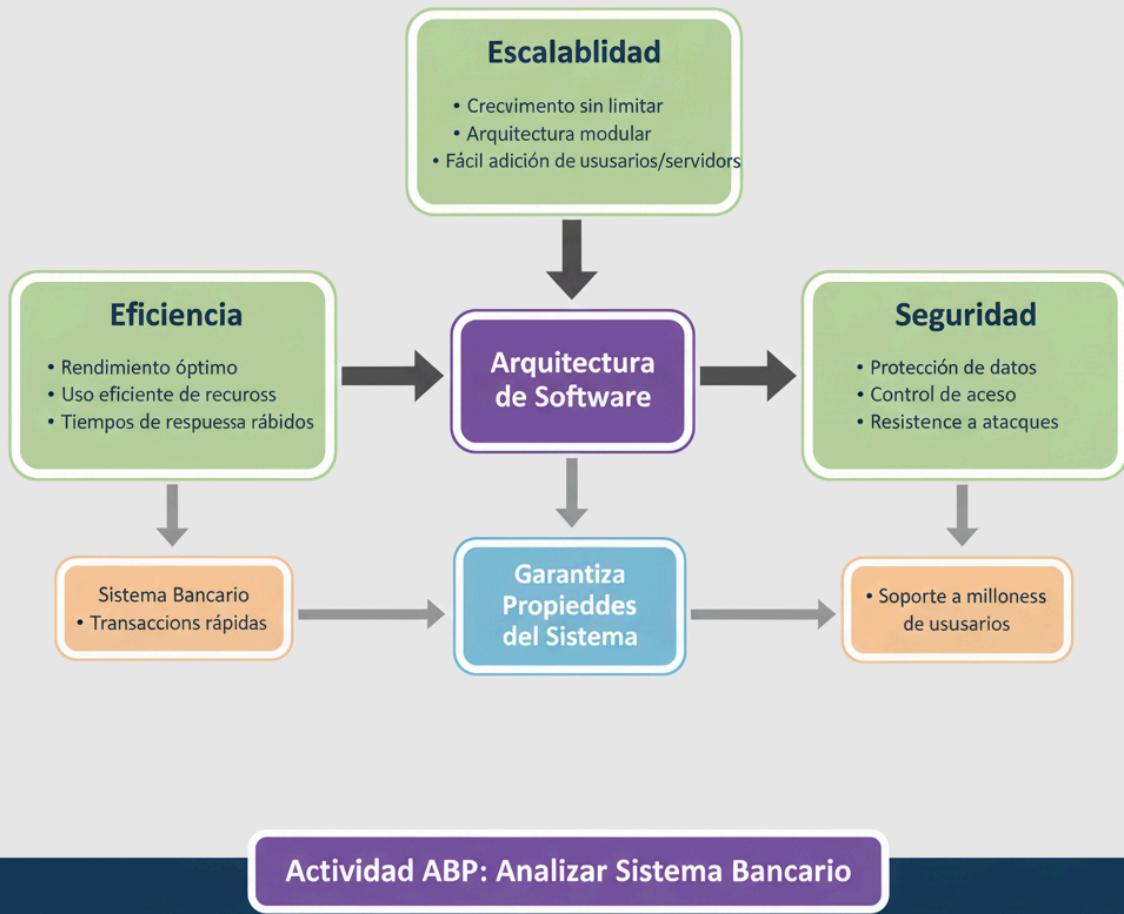
Evidencia

1. Documento analítico:

- Explicación de los retos del sistema bancario.
- Análisis de las soluciones arquitectónicas aplicadas (capas de seguridad, redundancia, balanceo de carga).
- Ejemplos de tecnologías y patrones usados (microservicios, arquitectura orientada a eventos, cifrado).

Impacto en la Eficiencia, Escalabilidad y Seguridad del Sistema

Subtema 5.2



Subtema 5.3: Reducción de riesgos y costos a través de una arquitectura bien definida

Actividad ABP

- Analizar el caso de **Healthcare.gov** (EE. UU., 2013).
- Identificar los **fallos arquitectónicos** que generaron problemas:
 - Caídas del sistema.
 - Sobrecostos millonarios.

- Falta de escalabilidad.
- Relacionar cómo una arquitectura sólida habría evitado dichos riesgos y costos.

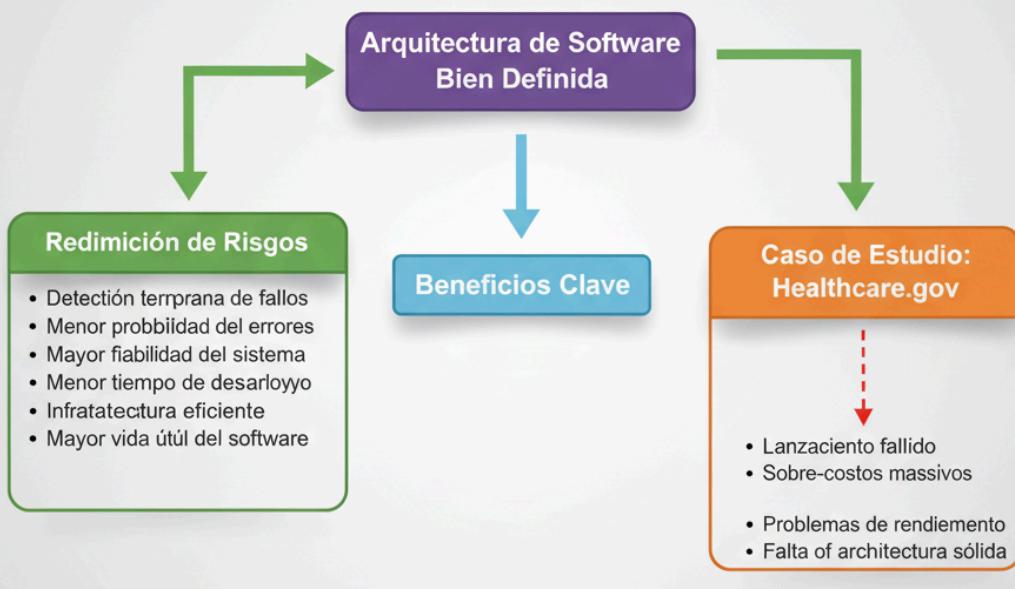
Evidencia

1. **Ensayo crítico:**

- Breve descripción del caso.
- Análisis de los errores arquitectónicos.
- Estimación de costos asociados al fracaso.
- Lecciones aprendidas y propuestas de mejora.

Reducción de Riesgos y Costos a través de una Arquitectura Bien Definida

Subtema 5.3



Actividad ABP: Identificar Fallos + Costos

Evidencia: Ensayo + Mapa Conceptual

Subtema 5.4: Casos prácticos de éxito y fracaso relacionados con la arquitectura de software

Actividad ABP

Estudiar en grupos los casos de éxito de Amazon y Netflix, así como el caso de fracaso del VCF del FBI (Virtual Case File).

Los estudiantes deben:

1. Analizar el rol de la arquitectura de software en los resultados obtenidos.

2. Identificar factores técnicos, organizacionales y estratégicos que influyeron en el éxito o fracaso.
3. Comparar y contrastar las lecciones aprendidas.
4. Elaborar propuestas de buenas prácticas aplicables a futuros proyectos.

Casos de estudio

Caso de Éxito: Amazon

- **Arquitectura:** Migración a microservicios y posteriormente a la nube (AWS).
- **Resultado:** Escalabilidad global, resiliencia, reducción de costos y creación de nuevos modelos de negocio (AWS).
- **Factores clave:**
 - Arquitectura evolutiva y modular.
 - Cultura DevOps y automatización.
 - Estrategia alineada al negocio digital.

Caso de Éxito: Netflix

- **Arquitectura:** Microservicios + computación en la nube (AWS).
- **Resultado:** Alta disponibilidad, streaming a millones de usuarios en tiempo real, personalización basada en datos.
- **Factores clave:**
 - Tolerancia a fallos (Chaos Engineering).
 - Escalabilidad dinámica.
 - Innovación continua.

Caso de Fracaso: VCF del FBI (Virtual Case File)

- **Arquitectura:** Sistema monolítico y rígido, sin considerar adecuadamente requisitos cambiantes.
- **Problemas:** Retrasos, sobrecostos, falta de flexibilidad.

- **Resultado:** Proyecto cancelado tras gastar más de **\$170 millones**.

- **Factores clave:**

- Falta de alineación entre requisitos de negocio y arquitectura.
- Escasa gestión del cambio.
- Mala comunicación entre stakeholders.

