UNIVERSIDAD PERUANA LOS ANDES

FACULTAD: **INGENIERÍA**
ESCUELA PROFESIONAL: **SISTEMAS Y COMPUTACIÓN**

PROFESOR: **Fernandez Bejarano Raul Enrique**

INTREGRANTES:

Guerra Yarupaita Carlos Daniel
Limaylla Carhuallanqui Sebastian
Retamozo Ataucusi Nehemias
Sanchez Romero Aldair

CICLO: VIII

HUANCAYO 2025

Tema 01: Conceptos Fundamentales de la Evaluación Arquitectónica

Ensayo Técnico Breve

Fundamentos y Estándares Internacionales en la Evaluación Arquitectónica de Software

1. **Introducción**
   En el ámbito de la ingeniería de software, la **evaluación arquitectónica** representa una etapa clave que permite analizar la solidez, coherencia y calidad de una solución antes de su implementación. Este proceso busca garantizar que la arquitectura propuesta responda eficazmente a los requerimientos funcionales y no funcionales del sistema, reduciendo riesgos técnicos y mejorando la mantenibilidad del producto final. En el caso del *Sistema de Gestión Académica Universitaria (SIGA-UPLA)*, dicha evaluación es esencial para asegurar que la infraestructura tecnológica sea capaz de soportar la gestión de matrículas, asignaturas, calificaciones y reportes de manera eficiente y sostenible. Asimismo, la aplicación de estándares internacionales como **ISO/IEC 25010** y **IEEE 1471** proporciona un marco de referencia que permite medir la calidad del software con criterios objetivos, promoviendo la mejora continua y la transparencia en las decisiones de diseño arquitectónico.

2. **Desarrollo**

   La evaluación arquitectónica es un proceso analítico que permite determinar en qué medida una arquitectura de software cumple con los atributos de calidad definidos por los usuarios y las necesidades del sistema. En este sentido, se convierte en una herramienta esencial para identificar fortalezas y debilidades del diseño antes de su implementación, reduciendo así los riesgos de fallos, sobrecostos o deficiencias en el rendimiento.

   Uno de los marcos de referencia más importantes para esta evaluación es la **norma ISO/IEC 25010**, la cual establece un modelo integral de calidad del producto de software, organizado en características como **eficiencia del desempeño, confiabilidad, seguridad, usabilidad, mantenibilidad y portabilidad**. Cada uno de estos atributos aporta criterios medibles que orientan las decisiones de diseño hacia la mejora continua y la satisfacción de las expectativas del usuario.

   De manera complementaria, la **norma IEEE 1471** (actualmente **ISO/IEC/IEEE 42010**) define los lineamientos para la **documentación y descripción de arquitecturas de software**, fomentando la claridad en la comunicación entre los actores del proyecto. Este estándar establece la importancia de representar los diferentes puntos de vista y perspectivas arquitectónicas —como las de desarrollo, implementación o despliegue— que facilitan la comprensión global del sistema y su relación con los objetivos del negocio.

   En el contexto del *SIGA-UPLA*, la aplicación de estos estándares resulta fundamental para garantizar una arquitectura tecnológica coherente y adaptable. El sistema académico requiere una infraestructura que soporte procesos críticos como la gestión de matrículas, calificaciones, asignaturas y reportes, por lo que su diseño debe priorizar atributos como **seguridad**, **escalabilidad** y

mantenibilidad. A través de una evaluación arquitectónica rigurosa, es posible anticipar cuellos de botella, mejorar la eficiencia del desempeño y asegurar la continuidad operativa del sistema, incluso ante el crecimiento del número de usuarios o nuevas demandas institucionales.

En síntesis, la evaluación arquitectónica no solo permite medir la calidad técnica del software, sino también sustentar la toma de decisiones estratégicas durante el desarrollo, contribuyendo a la creación de soluciones robustas, eficientes y sostenibles a largo plazo.

3. Conclusión

La evaluación arquitectónica se consolida como una práctica indispensable dentro del ciclo de vida del software, ya que permite verificar la calidad estructural, la eficiencia y la sostenibilidad de una solución tecnológica antes de su implementación. En el caso del *Sistema de Gestión Académica Universitaria (SIGA-UPLA)*, su aplicación garantiza que la arquitectura propuesta responda a las necesidades funcionales de la institución y mantenga altos niveles de desempeño, seguridad y escalabilidad.

El uso de estándares internacionales como **ISO/IEC 25010** y **IEEE 1471** aporta un marco metodológico sólido que orienta el análisis y la validación de las decisiones arquitectónicas, promoviendo la transparencia y la mejora continua en el desarrollo de sistemas. Gracias a esta evaluación, es posible anticipar fallas, optimizar recursos y asegurar la coherencia entre los objetivos del proyecto y la infraestructura tecnológica que los sustenta.

En conclusión, la evaluación arquitectónica no solo representa un instrumento técnico, sino también una estrategia de gestión de calidad que fortalece la confiabilidad del software, incrementa su valor institucional y garantiza la sostenibilidad de las soluciones digitales a largo plazo.


Conceptos Fundamentales de la Evaluación Arquitectónica
Evaluación Arquitectónica
Proceso para analizar la calidad de una arquitectura de software.
Busca anticipar problemas antes de la implementación.
Evalúa **atributos de calidad** del sistema.
Atributos de Calidad (ISO/IEC 25010)
Eficiencia del desempeño → uso óptimo de recursos.
Mantenibilidad → facilidad de corrección y mejora.
Usabilidad → facilidad de uso por parte del usuario.
Seguridad → protección frente a amenazas y accesos no autorizados.
Escalabilidad → capacidad de adaptación ante mayor demanda.
Estándares Internacionales
ISO/IEC 25010: modelo de calidad del software.
IEEE 1471 / ISO/IEC/IEEE 42010: descripción y documentación de arquitecturas.
Objetivo común: evaluar la calidad estructural y comunicativa del sistema.
Aplicación en el SIGA-UPLA
Garantiza una arquitectura robusta y escalable.
Soporte confiable para matrícula, asignaturas, calificaciones y reportes.
Mejora la eficiencia y confiabilidad institucional.
Conclusión
La evaluación arquitectónica asegura:
Cumplimiento de estándares.
Sostenibilidad técnica.
Alta calidad del software.

Tema 02: Escalabilidad como criterio de evaluación arquitectónica

Ensayo Técnico Breve

1. **Introducción**

En el contexto de los sistemas informáticos modernos, la escalabilidad constituye un criterio esencial dentro de la evaluación arquitectónica, al determinar la capacidad de un software para adaptarse al incremento de usuarios, datos y transacciones sin comprometer su rendimiento. Este atributo se vuelve especialmente relevante en entornos académicos como el Sistema de Gestión Académica Universitaria (SIGA-UPLA), donde el volumen de operaciones puede aumentar significativamente durante los periodos de matrícula o generación de reportes. Analizar la escalabilidad permite seleccionar la arquitectura más adecuada ya sea monolítica, cliente-servidor distribuida o basada en microservicios asegurando la estabilidad, disponibilidad y eficiencia del sistema a largo plazo.

2. **Desarrollo**

La **escalabilidad** se define como la capacidad de un sistema para mantener o mejorar su rendimiento ante un aumento progresivo de la carga de trabajo. Este atributo, contemplado en los estándares **ISO/IEC 25010**, es un componente crítico de la calidad del software, directamente relacionado con la **eficiencia del desempeño** y la **capacidad de adaptación** del sistema a nuevas demandas.

Existen diversas estrategias arquitectónicas que abordan la escalabilidad desde diferentes enfoques:

1. **Arquitectura monolítica:** concentra todas las funciones del sistema en un solo bloque de código. Aunque es más simple de implementar, su mantenimiento y expansión se dificultan conforme crece el número de usuarios.
2. **Arquitectura cliente-servidor distribuida:** separa responsabilidades y mejora la capacidad de respuesta, pero puede generar cuellos de botella si no se gestiona adecuadamente la concurrencia.
3. **Arquitectura de microservicios:** divide el sistema en servicios independientes que se comunican mediante APIs, permitiendo escalar solo los módulos que lo requieren y optimizando el uso de recursos.

En el caso del *SIGA-UPLA*, la arquitectura de microservicios representa una alternativa ideal, pues facilita la administración independiente de módulos como matrícula, calificaciones, asignaturas o reportes, sin afectar el funcionamiento del sistema completo. Este enfoque permite ejecutar simulaciones de carga y pruebas de rendimiento que demuestran su capacidad para mantener tiempos de respuesta estables ante la concurrencia de múltiples usuarios.

Además, los estándares **IEEE 1471** y **ISO/IEC/IEEE 42010** proporcionan lineamientos para documentar las decisiones arquitectónicas relacionadas con la escalabilidad, estableciendo criterios objetivos para evaluar la capacidad del sistema frente a escenarios de crecimiento. Dichas prácticas favorecen la toma de decisiones

técnicas informadas y alineadas con los objetivos institucionales de sostenibilidad y eficiencia.

3. **Conclusión**

La escalabilidad es un atributo fundamental en la evaluación arquitectónica, pues garantiza que un sistema sea capaz de evolucionar sin degradar su rendimiento ni comprometer la experiencia del usuario. En el *SIGA-UPLA*, este criterio adquiere especial relevancia al permitir la expansión funcional y el incremento del número de usuarios sin afectar la estabilidad del servicio.

La adopción de arquitecturas modernas como los **microservicios**, junto con la aplicación de estándares internacionales, ofrece una base sólida para desarrollar sistemas académicos resilientes, modulares y preparados para el crecimiento institucional. En consecuencia, evaluar la escalabilidad no solo mejora la calidad técnica del software, sino que asegura su vigencia, eficiencia y adaptabilidad a los desafíos futuros de la gestión universitaria.

Tema 03: Mantenibilidad como criterio de evaluación arquitectónica

1. Introducción

El Sistema de Gestión Académica Universitaria (SIGA-UPLA) es una plataforma institucional que administra información relacionada con matrículas, control de asistencia, calificaciones, gestión docente, horarios, y reportes académicos.

Con el tiempo, el SIGA-UPLA se ha convertido en un sistema difícil de mantener, debido a su crecimiento descontrolado y la falta de una arquitectura modular. El código presenta alto acoplamiento y baja cohesión, lo que genera errores frecuentes al intentar agregar o modificar funcionalidades.

2. Problemas detectados en el sistema legado

Categoría	Descripción del problema	Impacto en el mantenimiento
Alto acoplamiento	Los módulos de matrícula, notas, y asistencia dependen entre sí; comparten código y llamadas directas a la base de datos.	Cualquier cambio en un módulo afecta a los demás. Dificulta el mantenimiento y genera errores colaterales.
Baja cohesión	Las clases y funciones realizan múltiples tareas sin un propósito único.	Dificulta la comprensión y el reuso del código.

Diseño monolítico	Toda la lógica (interfaz, lógica de negocio y datos) está en un solo bloque.	Baja escalabilidad y lentitud en despliegues.
Ausencia de modularización	No existen capas o módulos independientes.	Dificulta pruebas unitarias y reutilización.
Falta de documentación y control de versiones	No se registran cambios ni se documentan las funciones.	Aumenta la dependencia del personal técnico actual.

3. Análisis técnico

El SIGA-UPLA fue desarrollado con una estructura monolítica, donde la interfaz web, la lógica de negocio y las consultas SQL están integradas directamente.

Este diseño viola principios SOLID y genera dependencias cíclicas, lo que hace costoso el mantenimiento y la incorporación de nuevas funciones.

Por ejemplo, al modificar el módulo de asistencia, también se deben ajustar los módulos de notas y matrícula.

4. Propuesta de mejora arquitectónica

Estrategia de mejora	**Descripción técnica**	**Beneficio para la mantenibilidad**
Refactorización del código	Reestructurar funciones largas y clases con múltiples responsabilidades, aplicando el principio de responsabilidad única (SRP).	Facilita comprensión y mantenimiento.
Arquitectura en capas	Separar en capas: Presentación, Negocio y Datos.	Permite mantener y probar cada capa de forma independiente.
Aplicación del patrón MVC	Dividir el sistema en Modelo (datos), Vista (interfaz), y Controlador (lógica).	Reduce acoplamiento y mejora la organización.
Arquitectura Hexagonal (Ports & Adapters)	Desacoplar la lógica del negocio de la infraestructura (bases de datos, APIs, interfaces).	Mejora la flexibilidad y pruebas automatizadas.
Modularización funcional	Crear módulos independientes: Gestión de Matrícula, Notas, Asistencia, Docentes, Usuarios.	Posibilita mantenimiento individual sin afectar al resto.

Uso de control de versiones (Git)	Implementar repositorio con ramas por módulo.	Permite revertir errores y seguir el historial de cambios.
Documentación técnica (Swagger, README, diagramas UML)	Documentar endpoints, estructuras y diagramas de clases.	Facilita el trabajo colaborativo y la capacitación.

5. Resultados esperados

Con la aplicación de estas estrategias:

- Se reducirá el tiempo de mantenimiento y el riesgo de fallos colaterales.
- Se facilitará la incorporación de nuevos módulos.
- El sistema podrá migrarse a una arquitectura modular o basada en microservicios a mediano plazo.

6. Conclusión

La mantenibilidad del SIGA-UPLA puede mejorar significativamente al adoptar una arquitectura en capas o hexagonal, acompañada de buenas prácticas de refactorización y modularización. Esto permitirá que el sistema evolucione de forma sostenible y ágil.

Tema 04: Seguridad como criterio de evaluación arquitectónica

1. Introducción

El SIGA-UPLA almacena información sensible: datos personales de estudiantes y docentes, calificaciones, historial académico y credenciales de acceso.

Por ello, es fundamental aplicar principios de seguridad en su arquitectura, considerando los riesgos identificados en el OWASP Top 10, que recopila las vulnerabilidades más comunes en sistemas web.

2. Vulnerabilidades detectadas (basadas en OWASP Top 10)

Vulnerabilidad OWASP	Descripción	Ejemplo en SIGA-UPLA	Riesgo

A01: Inyección SQL	Entrada del usuario sin validación usada directamente en consultas SQL.	Formulario de login sin uso de parámetros preparados.	Acceso no autorizado y robo de datos.
A03: Exposición de datos sensibles	Datos sin cifrado durante transmisión.	Envío de contraseñas por HTTP sin TLS.	Fuga de información.
A05: Configuración de seguridad incorrecta	Servidor con puertos abiertos o credenciales por defecto.	Uso de “admin123”.	Exposición a ataques remotos.
A07: Fallos de identificación y autenticación	Contraseñas débiles y sin expiración.	Contraseñas como “12345” o “up123”.	Suplantación de identidad.
A08: Fallos de validación de entrada (XSS)	Permite ejecución de código malicioso en formularios.	Campo “observaciones” acepta scripts.	Robo de sesiones o redirección maliciosa.

3. Principios de seguridad aplicables

Principio	**Descripción**	**Aplicación en SIGA-UPLA**
Confidencialidad	Solo los usuarios autorizados deben acceder a los datos.	Autenticación con JWT y cifrado AES de contraseñas.
Integridad	Los datos no deben ser modificados sin autorización.	Control de acceso por roles (estudiante, docente, admin).
Disponibilidad	El sistema debe estar disponible ante fallos o ataques.	Servidores replicados y respaldo automático diario.

4. Estrategias de seguridad recomendadas

Estrategia	**Descripción técnica**	**Beneficio principal**
Zero Trust Architecture (ZTA)	No confiar en ningún usuario o red por defecto. Cada acceso se valida.	Evita accesos no autorizados y movimientos laterales.

Defensa en profundidad	Múltiples capas de protección: firewall, cifrado, autenticación y auditoría.	Minimiza el impacto de ataques exitosos.
Autenticación fuerte (OAuth2, JWT)	Genera tokens cifrados para sesiones seguras.	Evita secuestro de sesiones.
Cifrado de datos (TLS, AES-256)	Protección de datos en tránsito y reposo.	Evita robo de información sensible.
Auditoría y monitoreo (Logs seguros)	Registro de accesos, errores y eventos críticos.	Permite detección temprana de incidentes.
Validación de entrada/salida	Filtrado y sanitización de datos en formularios.	Previene inyecciones y XSS.

5. Modelo de seguridad propuesto

Flujo general de autenticación segura:

Usuario → Frontend (Validación de entrada) → API Gateway → Servicio de Autenticación (JWT) → Autorización basada en roles → Acceso al módulo correspondiente

Capas de defensa:

1. Capa de firewall y proxy inverso.
2. Capa de autenticación (OAuth2).
3. Capa de validación de datos.
4. Capa de auditoría (logs cifrados).

6. Conclusión

La adopción de estrategias de Zero Trust y defensa en profundidad, junto con la mitigación de vulnerabilidades OWASP, permitirá que el SIGA-UPLA garantice la confidencialidad, integridad y disponibilidad de la información académica.

El rediseño arquitectónico debe integrar la seguridad desde la base del sistema, no como un añadido final.

Tema 05: Estrategias y Métodos de Evaluación Arquitectónica

Objetivo: Aplicar un método formal de evaluación arquitectónica para identificar fortalezas, debilidades, trade-offs y oportunidades de mejora en un sistema empresarial (ERP o CRM), en este caso el SIGA-UPLA.

1. Selección del método

Los métodos más comunes son:

- **ATAM (Architecture Tradeoff Analysis Method):** Evalúa cómo la arquitectura soporta requerimientos de calidad y permite identificar trade-offs.
- **CBAM (Cost Benefit Analysis Method):** Evalúa decisiones arquitectónicas considerando costos y beneficios.
- **Evaluación basada en escenarios:** Centrada en escenarios específicos de uso o calidad.

Para SIGA-UPLA, ATAM es adecuado porque permite analizar cómo la arquitectura actual soporta requerimientos de calidad como escalabilidad, seguridad y mantenibilidad.

2. Proceso de evaluación

1. **Recolección de información:**
    - Describir el sistema: SIGA-UPLA como plataforma web para gestión académica.
    - Identificar stakeholders: estudiantes, docentes, administrativos, TI.

      Listar requerimientos de calidad: escalabilidad, disponibilidad, seguridad, mantenibilidad.
2. **Identificación de escenarios de calidad:**
    - Escenario de escalabilidad: aumento del 50% de estudiantes en el semestre sin degradar el rendimiento.
    - Escenario de seguridad: protección de datos personales y académicos según normativas.
    - Escenario de mantenibilidad: facilidad de integrar nuevos módulos (como módulo de pagos o notificaciones).
3. **Análisis de trade-offs:**
    - Ejemplo: usar microservicios mejora escalabilidad pero aumenta complejidad de mantenimiento.

- Centralizar base de datos mejora consistencia pero puede limitar escalabilidad.

4. **Recomendaciones de mejora:**

    - Introducir balanceadores de carga y caché para mejorar escalabilidad.
    - Implementar autenticación multifactor y cifrado de datos para seguridad.
    - Documentar módulos y APIs para mejorar la mantenibilidad.

3. Documentación

- Informe técnico que incluya:
    - Descripción del sistema y stakeholders.
    - Escenarios de calidad evaluados.
    - Diagramas de arquitectura (UML: diagramas de componentes, despliegue).
    - Tabla de trade-offs identificados.

Tema 06: Integración y Proyecto de Evaluación Arquitectónica

Como cierre de la unidad, los equipos desarrollarán un proyecto integrador, seleccionando un caso real o ficticio (plataforma de salud, sistema bancario, portal educativo). Deberán:

1. Definir requerimientos de calidad.

2. Aplicar criterios de escalabilidad, mantenibilidad y seguridad.

3. Elaborar un informe de evaluación arquitectónica con diagramas UML y recomendaciones.

4. Presentar y defender su propuesta ante la clase, recibiendo retroalimentación.

Objetivo: Integrar lo aprendido desarrollando un proyecto de evaluación arquitectónica completo sobre SIGA-UPLA.

1. Definición de requerimientos de calidad

- **Escalabilidad:** soportar incremento de usuarios sin afectar tiempos de respuesta.
- **Mantenibilidad:** facilidad de actualización y extensión del sistema.
- **Seguridad:** confidencialidad, integridad y disponibilidad de la información.

2. Aplicación de criterios de calidad

- **Escalabilidad:** diseño con servicios desacoplados, balanceadores de carga, base de datos distribuida.
- **Mantenibilidad:** uso de patrones de diseño, documentación, modularización.
- **Seguridad:** control de acceso basado en roles, cifrado de datos, auditorías.

3. Elaboración del informe

El informe debe incluir:

1. **Introducción y objetivos.**
2. **Descripción del sistema SIGA-UPLA.**
3. **Requerimientos de calidad y criterios aplicados.**
4. **Diagramas UML:**
    - Diagrama de componentes.

- Diagrama de despliegue.
- Diagramas de secuencia o actividad relevantes.

5. **Evaluación de trade-offs:** tabla comparativa con ventajas y desventajas de decisiones arquitectónicas.
6. **Recomendaciones finales:** mejoras para escalabilidad, seguridad y mantenibilidad.

4. Presentación y defensa

- Exponer el informe y diagramas.
- Argumentar las decisiones de arquitectura y trade-offs.
- Recibir retroalimentación de los compañeros y docente.

Informe Técnico: Evaluación Arquitectónica del SIGA-UPLA

1. Introducción

El SIGA-UPLA es una plataforma web diseñada para gestionar el ciclo de vida académico de estudiantes, docentes y administrativos. Su propósito es automatizar procesos que actualmente se realizan manualmente o mediante sistemas aislados.

El presente informe aplica el **método** ATAM (Architecture Tradeoff Analysis Method) para evaluar la arquitectura del sistema, identificando fortalezas, debilidades y oportunidades de mejora, enfocándose en **escalabilidad, mantenibilidad y seguridad**.

2. Stakeholders

Stakeholder	**Intereses / Necesidades**
Estudiantes	Acceso rápido a notas, inscripción y consulta de horarios
Docentes	Registro de calificaciones, control de asistencia, comunicación con estudiantes
Administrativos	Gestión de matrícula, reportes estadísticos, seguimiento de pagos

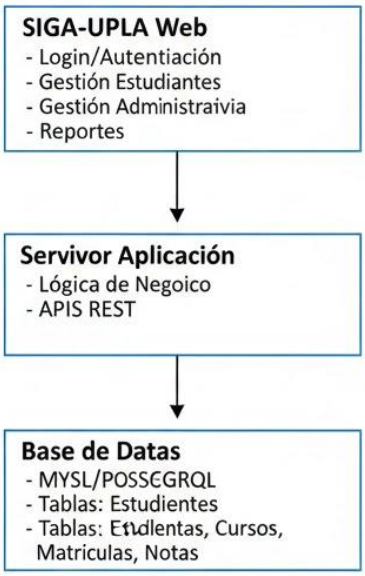
Área TI	Mantenimiento, integración de módulos, seguridad y disponibilidad del sistema

3. Requerimientos de Calidad Evaluados

Requerimiento	**Escenario de Evaluación**
Escalabilidad	Aumento del 50% de usuarios sin degradar tiempos de respuesta
Mantenibilidad	Integración rápida de nuevos módulos (p. ej., módulo de pagos)
Seguridad	Protección de datos académicos y personales, acceso controlado por roles

4. Arquitectura Actual del Sistema

4.1 Diagrama de Componentes UML



4.2 Diagrama de Despleuie UML

5. Escenarios de Calidad y Trade-offs Identificados

Escenario	**Decisión Arquitectónica**	**Ventajas**	**Desventajas / Trade-offs**
Escalabilidad	Servidor centralizado con base de datos única	Fácil de implementar	Limitada escalabilidad; punto único de falla
Mantenibilidad	Modularización por componentes (estudiantes, docentes, administración)	Facilita extensión y pruebas	Requiere coordinación entre módulos

Seguridad	Autenticación basada en roles + cifrado	Protección de datos sensibles	Complejidad en gestión de permisos y claves