**Evaluación de Arquitectura**

**Evaluación de Arquitectura de la Solución**

Para realizar una evaluación exhaustiva de la arquitectura de nuestra solución, hemos seguido el método ATAM (Architecture Tradeoff Analysis Method), tomando en cuenta los principales concerns de nuestro cliente que impactan directamente en los atributos de calidad y las decisiones arquitectónicas.

**Concerns Importantes del Cliente**

1. **Escalabilidad**
   * **Descripción del Concern:** La capacidad del sistema para manejar un crecimiento significativo en el número de usuarios y la cantidad de datos procesados.
   * **Impacto en la Arquitectura:** Las decisiones sobre la estructura modular del sistema, el uso de microservicios, y la implementación de bases de datos distribuidas son cruciales para asegurar que el sistema pueda escalar horizontalmente sin pérdida de rendimiento.
   * **Análisis:** Nuestro diseño actual utiliza microservicios, lo que permite escalar componentes individuales del sistema según la demanda. Si encontramos que algún servicio específico es un cuello de botella, podemos descomponerlo en servicios más pequeños y especializados.
   * **Trade-offs:** Aumentar la modularidad y la independencia de los servicios puede incrementar la complejidad en la gestión de interdependencias y en la comunicación entre servicios. Es vital implementar mecanismos eficientes de orquestación y monitoreo.
2. **Seguridad**
   * **Descripción del Concern:** Protección de datos sensibles y prevención de accesos no autorizados.
   * **Impacto en la Arquitectura:** Las decisiones sobre la autenticación, autorización, encriptación de datos y auditoría de accesos son fundamentales para cumplir con los requisitos de seguridad.
   * **Análisis:** Hemos integrado OAuth 2.0 para la gestión de autenticaciones y autorizaciones, y usamos encriptación AES para proteger los datos almacenados. Además, se han implementado registros de auditoría detallados para el seguimiento de accesos.
   * **Trade-offs:** La implementación de estas medidas puede afectar el rendimiento debido a la sobrecarga computacional. Es necesario equilibrar las capas de seguridad con la necesidad de mantener un rendimiento aceptable.
3. **Mantenibilidad**
   * **Descripción del Concern:** Facilidad de actualización, corrección de errores y adición de nuevas funcionalidades.
   * **Impacto en la Arquitectura:** La modularidad del código, la documentación clara, y la separación de preocupaciones son esenciales para asegurar que el sistema sea mantenible a largo plazo.
   * **Análisis:** Nuestro sistema sigue principios de diseño orientado a objetos y patrones de diseño como MVC (Model-View-Controller), lo que facilita la mantenibilidad y extensibilidad del código.
   * **Trade-offs:** Asegurar una alta mantenibilidad puede llevar a un mayor esfuerzo inicial en la documentación y en la estructuración del código, pero a largo plazo, estos esfuerzos se traducen en una mayor eficiencia en el desarrollo y mantenimiento.

**Cambios Necesarios**

Para abordar completamente estos concerns, hemos identificado algunas áreas de mejora:

* **Para la Escalabilidad:** Implementar un sistema de monitoreo en tiempo real y ajustes automáticos en la asignación de recursos.
* **Para la Seguridad:** Mejorar las políticas de manejo de claves y actualizar regularmente las librerías de seguridad.
* **Para la Mantenibilidad:** Adoptar herramientas de automatización de pruebas y CI/CD (Integración Continua/Despliegue Continuo).

**Potenciales Riesgos**

* **Escalabilidad:** Riesgo de sobrecarga en los servicios de orquestación si no se gestionan correctamente las dependencias.
* **Seguridad:** Posible vulnerabilidad debido a la actualización tardía de componentes de terceros.
* **Mantenibilidad:** Riego de deuda técnica si no se realiza un refactoring periódico y no se actualiza la documentación regularmente.

**Iteración de HUs**

**Evaluación y Modificaciones Basadas en las Pruebas**

A partir de las pruebas ejecutadas en el Hito 3 y la evaluación arquitectónica, hemos identificado áreas clave para la mejora de nuestras Historias de Usuario (HUs).

**Principales Observaciones de las Pruebas**

* **HU1: Registro de Usuarios**
  + *Observación:* La validación de datos no estaba cubriendo todos los casos de uso.
  + *Modificación:* Hemos ampliado las validaciones y añadido pruebas unitarias adicionales para asegurar la integridad de los datos de registro.
* **HU2: Autenticación de Usuarios**
  + *Observación:* Algunos usuarios experimentaron lentitud en el proceso de autenticación.
  + *Modificación:* Optimizamos las consultas a la base de datos y mejoramos el manejo de sesiones para reducir la latencia.
* **HU3: Gestión de Perfiles de Usuario**
  + *Observación:* La interfaz de usuario no era completamente intuitiva.
  + *Modificación:* Realizamos mejoras en la usabilidad de la interfaz y añadimos instrucciones claras para la navegación.

**Rationale de las Decisiones**

* Las modificaciones realizadas buscan mejorar la experiencia del usuario final y asegurar que el sistema sea robusto y escalable.
* La optimización de las consultas y el manejo de sesiones en la HU2 responde a la necesidad de ofrecer un sistema eficiente y ágil, fundamental para la satisfacción del usuario.
* Las mejoras en la interfaz de la HU3 aseguran que los usuarios puedan interactuar con el sistema de manera intuitiva, lo cual es esencial para la adopción y uso continuo de la plataforma.

**Incorporación a la Plataforma**

* Hemos implementado las modificaciones y realizado despliegues incrementales para validar los cambios en un entorno de prueba antes de la implementación final.
* Cada HU modificada ha sido versionada y documentada en el repositorio, asegurando la trazabilidad y la referencia futura para el equipo de desarrollo y mantenimiento.

Con estas iteraciones, hemos alineado nuestro desarrollo con los feedbacks recibidos y asegurado que nuestra solución no solo cumple con los requisitos actuales, sino que está preparada para futuras expansiones y mejoras.