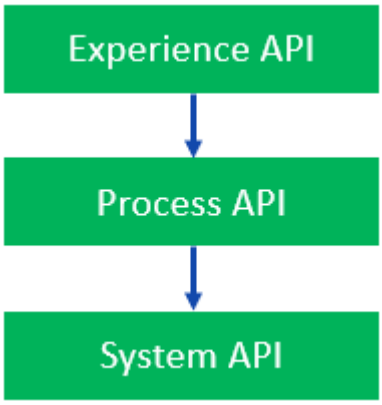
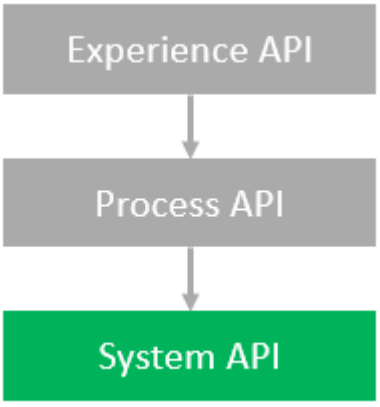
API Led Connectivity es un estilo arquitectónico que conecta datos con aplicaciones a través de API reutilizables y con un propósito.

En este blog, discutiremos qué variedades de arquitectura de conectividad LED API se pueden implementar. **API Led Connectivity** habla de **tres capas** como **Experience API**, **Process API** y **System API**. Cada capa tiene sus propios roles, responsabilidades y funcionalidades.

API Led Connectivity es un estilo arquitectónico que conecta datos con aplicaciones a través de API reutilizables y con un propósito.



System API



La **System API** le permite recuperar los datos sin procesar del sistema de registros como JDBC, SAP, Salesforce, AS400, etc. La recuperación de datos de los sistemas backend se puede exponer a la API ascendente de una manera segura y confiable.

- La System API responsable de conectar los sistemas backend y obtener los datos sin procesar necesarios.

- Transformar la obtención de datos desde los sistemas backend.
- Limpieza de los datos sin procesar de los sistemas backend.
- Exponer los datos a la API ascendente (**Process API** o **Experience API**) de forma segura y confiable.
- Manejo de errores, mapeo de los errores de los sistemas backend.

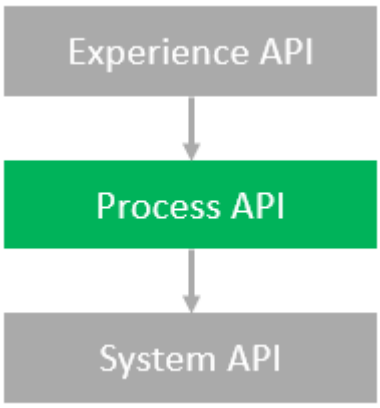
Es necesario abordar las siguientes **preguntas para diseñar las System API** de manera adecuada.

- ¿Cuál será la funcionalidad principal de la API del sistema?
- ¿Necesitamos API del sistema?
- ¿Cuántos sistemas están involucrados?
- ¿Necesitamos crear una API del sistema para cada sistema backend?
- ¿Qué tipo de medidas de seguridad se deben tener en cuenta para las API del sistema?
- ¿Necesitamos un contenedor para la API del sistema existente que no esté escrita en MuleSoft?

La siguiente lista de cosas que se deben tener en cuenta al diseñar la **System API**.

- La **System API** generalmente no se expone públicamente y debe implementarse dentro de una red privada o un puerto privado (por ejemplo, en CloudHub, la API del sistema se puede implementar en un puerto privado dentro de Anypoint Virtual Private Cloud).
- Proporcione siempre una URL interna a la API ascendente para comunicarse con las **System API**.
- Debe haber una **System API** para un sistema backend. En caso de que siga el diseño basado en dominios, puede haber múltiples API de sistema para un sistema (por ejemplo, tiene un servicio web monolítico que tiene operaciones relacionadas con múltiples dominios; en tal caso, puede agrupar operaciones relacionadas con dominios y crear múltiples **System API**, según número de dominios).
- Es posible que algunos sistemas backend no estén preparados para captar mucho tráfico. En tales casos, proteja la **System API** con políticas de control de picos o limitación de velocidad.

Process API



Process API es responsable de orquestar múltiples API posteriores (es decir, recopilar los datos de múltiples API posteriores). La funcionalidad principal de **Process API** es implementar lógica empresarial, agregación de datos, enrutamiento, etc.

- Exponer los datos a la API (**Experience**) ascendente de forma segura y confiable.
- Responsable del manejo de la lógica empresarial, enrutamiento, agregación de datos, enrutamiento, etc.
- Manejo de errores. Mapeo de los errores de las **System API**.
- Responsable de la orquestación de la llamada a la **System API** (es decir, uso de componentes dispersos, elección de enrutador, etc.).
- La línea de negocio es responsable de la **Process API**. También se la conoce como capa de innovación y agilidad.

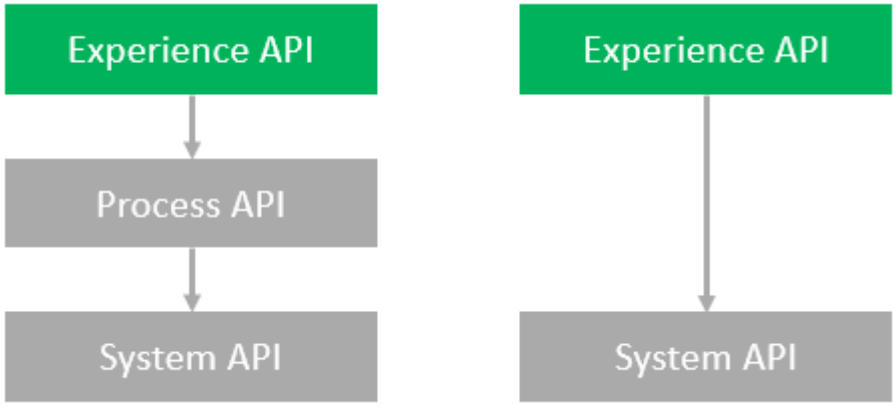
Es necesario abordar las siguientes preguntas para diseñar las API de proceso adecuadas.

- ¿Cuál será la funcionalidad principal de **Process API**?
- ¿Necesitamos **API Process**?
- ¿Existe ya la lógica empresarial en el servicio backend (por ejemplo, un procedimiento almacenado en una base de datos que tiene lógica empresarial)?
- ¿Necesitamos implementar lógica de negocios en la capa de procesos? (Siempre es una buena práctica tener la mayor parte de la lógica empresarial compleja en el sistema backend, si es posible. En caso de que los sistemas backend no puedan manejar la lógica empresarial, MuleSoft/IBM ACE/ Tools ESB proporcionan todos los componentes necesarios para implementar cualquier tipo de lógica empresarial) .
- ¿Quiénes serán los consumidores de **Process API**? (por ejemplo, la mayoría de las veces la API de proceso será consumida por múltiples **Experience API** de usuario para promover la reutilización).
- ¿Qué medidas de seguridad se deben tener en cuenta en una **Process API**?

La siguiente lista de cosas que se deben tener en cuenta al diseñar una **Process API**

- La **Process API** generalmente no está expuesta públicamente y debe implementarse dentro de una red privada o un puerto privado (por ejemplo, en CloudHub, la **Process API** se puede implementar en un puerto privado dentro de Anypoint Virtual Private Cloud). En caso de que desee exponer **Process API** a los consumidores, puede crear **API Proxy** encima de **Process API** y pedirles a los consumidores que se conecten a través de **API Proxy**. De esta manera, podemos dejar de exponer directamente la implementación de **Process API**.
- Proporcione siempre una URL interna a la API interna ascendente para comunicarse con las **Process API**.
- La **Process API** debe protegerse mediante la aplicación de políticas de seguridad como las políticas de cumplimiento de identificación de cliente.

Experience API



Experience API es la API orientada al usuario y puede ser reutilizada por diferentes canales como aplicaciones móviles, aplicaciones web o cualquier otro canal.

Generalmente, las **Experience API** son específicas del canal (por ejemplo, la aplicación móvil puede consumir API móvil, la aplicación web puede consumir API web, etc.).

- Manejo de errores. Mapeo de los errores de las API del proceso o del sistema.
- Consume **Process API** o **System API**.
- Realizar la transformación de datos según la necesidad de los consumidores.
- Exponer la API específica del canal.

Es necesario abordar las siguientes preguntas para diseñar las **Experience API** de manera adecuada

- ¿Cuál será la funcionalidad principal de **Experience API**?
- ¿Tiene **Experience API** varios consumidores?

- ¿Necesitamos **Experience API**?
- ¿Qué medidas de seguridad se deben tener en cuenta en **Experience API**?
- ¿Cómo proteger las **Experience API** de ataques DoS o DDoS?
- ¿Los diferentes consumidores tienen diferentes requisitos de seguridad?

La siguiente lista de cosas que se deben tener en cuenta al diseñar la **Experience API**

- **Experience API** no debe ser responsable de la orquestación, el enrutamiento ni de ninguna lógica empresarial.
- La **Experiencia API** debe estar habilitada con autenticación y autorización sólidas como OAuth 2.0
- La **Experiencia API** debe tener seguridad adicional para evitar ataques DoS, DDoS o cualquier API.

Julio Pari (IT Architect BIAN)

Especialista BIAN Semantic API | Gobierno de Integración | IBM Integration CP4I | IBM API Connect 10 | IBM ACE | IBM DataPower | OpenShift | Azure | AWS. Cualquier consulta envíame un mensaje a: info@arquitecturabank.com o sino a través de Linkedin: https://www.linkedin.com/in/juliopari/