Estándares para el Diseño de Apis REST - Banca Digital

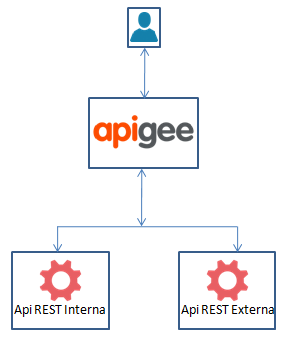
El termino REST (***Representational State Transfer o Transferencia de Estado Representacional***) se originó en el año 2000 por Roy Fielding, quien lo describe como un conjunto de restricciones con las que podemos crear un estilo de arquitectura de software, permitiendo la creación de aplicaciones web respetando HTTP con un estándar lógico y eficiente para el desarrollo de servicios.

Características

* **Sin estado**: cada una de las peticiones HTTP debe contener toda la información necesaria para poder ejecutarse.
* **Cacheable**: debe admitir un sistema de almacenamiento en cache.
* **Interfaz uniforme**: indica que cada uno de los recursos debe contar con una única dirección.
* **Sistema de capas**: arquitectura jerárquica entre los componentes.
* **Métodos estándar http:** sus operaciones principales son POST, GET, PUT Y DELETE

Dentro de Banca Digital se han establecido dos tipos de Apis:

**Externas:** Son aquellas que serán utilizadas por terceros que no tienen ninguna relación con Grupo Salinas; este tipo de Apis no deberá devolver información sensible como número de cliente, icu, cliente único, por mencionar algunos.

**Internas:** Son aquellas que serán utilizadas por áreas que pertenecen a Grupo Salinas, por ejemplo, Italika, Tiendas Neto, etc.

*¿Cómo debo construir el nombre de mi recurso?*

* Utilizar sustantivos en forma plural.
* No utilizar verbos.
* Utilizar como inicio para el nombre del recurso el nombre de tu Api en plural
* Utilizar la version del api dentro de tu recurso: {nombre del api}/**v1**, el control de versiones permite un mejor control para la actualización de los recursos.

Ejemplos:

|  |  |
| --- | --- |
| **Recurso** | **POST** |
| /clientes/v1/ | Crea un nuevo cliente |
| /usuarios/v1/compras | Agrega una nueva compra al usuario |

Características POST

* Hace referencia a la acción Crear (Create) del CRUD.
* Se consideran inseguras debido a que alteran el estado del servidor.
* Se consideran no idempotentes, lo que significan que las solicitudes repetidas darán como resultado la creación de varios recursos.
* Contienen un cuerpo (body request)

|  |  |
| --- | --- |
| **Recurso** | **GET** |
| /clientes/v1/ | Recupera la información del cliente |
| /catalogos/v1/ocupaciones | Recuperar la información del catalogo ocupaciones |

Características GET

* Hace referencia a la acción Leer (Read) del CRUD.
* Son seguras debido a que no alteran el estado del servidor.
* Son idempotentes, lo que significa que deben ser repetibles sin causar efectos secundarios.
* No tiene cuerpo (body request)

|  |  |
| --- | --- |
| **Recurso** | **UPDATE** |
| /clientes/v1 | Actualiza a todos los clientes registrados |
| /catalogos/v1/ocupaciones/125 | Actualiza la información de la ocupación 125 |

Características UPDATE

* Hace referencia a la acción Actualizar (Update) del CRUD.
* Son inseguras debido a que alteran el estado del servidor.
* Son idempotentes, lo que significa que al repetir la misma acción se debe obtener el mismo resultado.
* Contiene un cuerpo (body request)

|  |  |
| --- | --- |
| **Recurso** | **DELETE** |
| /clientes/v1/12/compra | Elimina todas las compras del cliente 12 |
| /catalogos/v1/ocupaciones | Elimina la información del catalogo de ocupaciones |

Características DELETE

* Hace referencia a la acción Eliminar (Delete) del CRUD.
* Son inseguras debido a que alteran el estado del servidor.
* Son idempotentes, lo que significa que al repetir la acción no se eliminara un recurso que ya ha sido eliminado.
* No tiene cuerpo (body request)

*¿Qué códigos de estado HTTP debo utilizar?*

**2XX.- Peticiones correctas**

**200 - OK**

Es la respuesta estándar para las peticiones procesadas de forma exitosa, usado generalmente en peticiones GET.

**201 - Created**

Indica que un recurso ha sido creado o actualizado, usado generalmente en peticiones POST y UPDATE.

**202 - Aceptada**

Indica que la petición ha sido atendida, sin embargo su procesamiento continua en segundo plano, usado generalmente en peticiones POST, UPDATE y DELETE.

**4XX.- Errores del Cliente**

**400 - Bad Request**

Indica que la solicitud no pudo ser procesada debido a que uno o varios campos de entrada no cumplen con el formato solicitado.

**404 - Not Found**

Indica que no se ha encontrado información del recurso solicitado.

**5XX.- Errores del servidor**

**500 - Internal Server Error**

Indica que se ha producido un error dentro del servidor el cual no ha sido controlado.

*¿Cómo debo definir la estructura para los errores dentro de mi Api?*

Como en cualquier otro sistema, el manejo de excepciones es muy importante y juega un aspecto clave para su correcto funcionamiento. Una excepción podría deberse a una razón técnica como un error de red con el sistema de back-end o algo impulsado por reglas de negocio.

Desde la perspectiva del desarrollador que consume las APIs, todo lo que se encuentra al otro lado de esa interfaz es una caja negra, por lo tanto, los errores se convierten en una herramienta clave que proporciona contexto y visibilidad sobre cómo usar una API.

Los mensajes de error deben ser consistentes en todas las APIs de la plataforma. Estos mensajes deben contener información para el usuario final, códigos de respuesta estructurados y, opcionalmente, una lista que detalle los errores ocurridos, sin embargo información o datos sobre el sistema de Back-End no deberán ser informados al consumidor.

Recomendaciones:

* Utilizar siempre el código de estado HTTP más apropiado.
* Proporcionar información suficiente del error, la cual permita describir lo ocurrido.
* Evite utilizar el mismo mensaje de error para todos los tipos de error que pueden ocurrir.
* Documenta los errores críticos.
* No incluir información que pueda dar pie a un posible ataque.
* Establecer los posibles escenarios de error para los recursos y crear sus mensajes de error.

Estructura para los códigos de error:

**{**

**"codigo":"{Código http}.{Nombre del área}-{Nombre del Api}.{Código de error}",**

**"mensaje":"{Mensaje de error}",**

**"info":"{URL Developer Portal}#{Código http}.{Nombre del área}-{Nombre del Api}.{Código de error}",**

**"folio": "{Folio de la operación}",**

**"detalles":[**

**{**

**"{Lista con la información del o los errores encontrados, es opcional}"**

**}**

**]**

**}**