Estandar de Diseño de Servicios REST

1. Introducción

Este documento, tiene como objetivo definir los estandares y lineamiento para diseñar servicios REST en la SUNAT.

2. Recursos

2.1. Definición

Los recursos, para nuestro caso especifico, representan alguna Entidad de nuestro Modelo Conceptual (ver el capitulo 5.1 DE LA ENTIDAD del Manual de Estandares y Modelamiento de Datos)

Ejemplos de Recursos:

- Contribuyente
- AccionFiscalizacion
- Fraccionamiento

2.2. Atributos de los Recursos

Los Recursos tienen atributos, los cuales contienen la información del Recurso, en forma similar a los atributos de las Entidades del Modelo Conceptual (ver el capitulo 5.1 DE LOS ATRIBUTOS del Manual de Estandares y Modelamiento de Datos)

LINEAMIENTO
Cada atributo debe tener un nombre unico, expresado como un sustantivo que describer las caracteristicas del atributo. El mismo significado no debe aplicarse a diferentes nombres, y de la misma forma un mismo nombre no puede tener dos significados.

Los atributos de los Recursos deben seguir el Standard lowerCamelCase y estar formado solo por caracteres Alfanumericos.

Los atributos tienen como parte del nombre un prefijos que representa el tipo de atributo, segun lo definido en el punto 5.2.5 del Manual de Estandares y Modelamiento de Datos.

Ejemplos:

```
numId: numero identificador
numPeriodo: numero de periodo
fecEmision: fecha de emision
mtoTotalSaldo: monto total del saldo
```

2.3. Formato de intercambio de Información en los Servicios REST

Es el formato utilizado tanto para enviar datos como recibir infomación de los Recursos desde el Servidor

LINEAMIENTO
Tanto los Datos enviados y recibidos desde y hacia Servidor, de los Servicios REST, deben estar en Formato JSON (ver definición)

Ejemplo de un recurso en formato JSON:

Recurso

```
{
 "numId":710010071784,
 "numPeriodo":200801,
 "fecEmision":"2015-06-02T21:03:11.78-05:00",
 "mtoSaldoTotal":100.01
}
```

NOTA
El tipo de dato JSON no necesariamente debe coincidir con el tipo de dato con el que se almacena la informacion en la BD (Por ejemplo en el Formato JSON no existe el tipo de dato Fecha). Por ello es necesario que el Servicio Rest este bien documentado respecto a los datos de respuesta.

Ejemplo:

Recurso

```
{
 "numId":"710010071784",
 "numPeriodo":"200801",
 "fecEmision":"2015-06-02T21:03:11.78-05:00",
 "mtoSaldoTotal":100.01
}
```

2.4. Formato de URL para los servicios REST

2.4.1. Formato de URL base

El URL Base para los Servicios REST en la SUNAT es:

http://api.sunat.peru/{version del API}/{dominio}/{proceso}/{e|t}

2.4.2. Formato de URL para Servicios de Entidad

El URL para los Servicios REST de Entidad es:

http://api.sunat.peru/{version del API}/{dominio}/{proceso}/e/{Nombre del Recurso Entidad}

> **NOTA**
> **Si el dominio es muy amplio, y hay la posibilidad de que existan 2 uri iguales , se agregara el macroproceso en la uri**
>
> **http://api.sunat.peru/{version del API}/{dominio/macroceso]}/{proceso}/e/{Nombre del Recurso Entidad}**

2.4.3. Formato de URL para Servicios de Tarea

El URL para los Servicios REST de Tarea es:

http:// api.sunat.peru/{version del API}/{dominio}/{proceso}/t/{Nombre del Sub Proceso}/{nombre de la tarea}

> **NOTA**
> **Si el dominio es muy amplio, y hay la posibilidad de que existan 2 uri iguales , se agregara el macroproceso en la uri**
>
> **http:// api.sunat.peru/{version del API}/{dominio/macroproceso]}/{proceso}/t/{Nombre del Sub Proceso}/{nombre de la tarea}**

Ejemplos :

http:// api.sunat.peru/v1/auditoria/tributaria/asignacion/e/fiscalizables

http:// api.sunat.peru/v1/controladuanero/ingreso/t/revisionaduanas/finalizarcontrolvehicular

2.5. Nombre de los Recursos de tipo Entidad en la URI

A un recurso se accede mediante una unica URI definida de la siguiente forma

<URL base>/{nombre del recurso entidad}[/{id}]

Los nombres de los recursos de tipo entidad en la URI deben ser definidos como sustantivo, en plural y en minusculas sin usar camelcase.

Ejemplo:

http://api.sunat.peru/v1/auditoria/tributaria/asignacion/e/accionesfiscalizacion

2.5.1. ID del RECURSO

El parametro **/{id}** en la URI representa el Identificador Unico de un Recurso, solo debe existir uno y solo un Recurso con esa URI

Ejemplo:

http://api.sunat.peru/v1/contribuyente/adminfo/e/contribuyentes/1024002254

De forma excepcional, en ciertos casos es posible que un recurso tenga un ID compuesto por 2 o mas Datos, en este caso estos valores se colocaran separados por guiones simples, de mas esta aclarar que tambien en este caso, solo debe existir uno y solo un Recurso con esa URI

<URL base>/{nombre del recurso entidad}[/{id1}-{id2}....[-{idn}]]

http://api.sunat.peru/v1/controladuanero/ingreso/declaracionesaduaneras/2015-10-00001-118

2.6. Nombre de los Recursos de tipo Tarea en la URI

A un recurso se accede mediante una unica URI definida de la siguiente forma

<URL base>/{nombre de la actividad}/{nombre de tarea}

Los nombres de las actividades y de las tareas son definido con un verbo porque representa una accion. Se escribe en singular y en minusculas sin usar camelcase.

Ejemplos

http:// api.sunat.peru/v1/controladuanero/ingreso/t/revisionaduanas/finalizarcontrolvehicular

> **NOTA**
> A partir de este punto en el documento, se omitira intencionalmente la URL base en las URI de ejemplo, solo para efectos de simplificar el documento.

2.7. Mapeo de Recursos con el métodos HTTP

De acuerdo con el modelo de madurez de Richardson y los principios de los servicios REST es necesario mapear los metodos HTTP con nuestro recursos segun las operaciones que desemos realizar.

Ejemplo para el Recurso "***Contribuyente***"

Operacion del Servicio	Metodo HTTP	URI	Descripción
listarContribuyentes	GET	`/contribuyentes`	Listar todos los contribuyentes
insertarContribuyente	POST	`/contribuyentes`	Registrar un nuevo Contribuyente
obtenerContribuyentePorID	GET	`/contribuyentes/10245577849`	Obtener el contribuyente con id 10245577849
actualizarContribuyentePorID	PUT	`/contribuyentes/10245577849`	Actualizar contribuyente con id 10245577849
eliminarContribuyentePorID	DELETE	`/contribuyentes/10245577849`	eliminar el contribuyente con id 10245577849

3. Sub-Recursos

En algunos casos los Recursos estan asociados con otros Recursos (En el modelo conceptual podria verse como relaciones de tipo Agregacion, Composicion, Maestro Detalle, Todo-Parte). En estos casos se pueden utilizar lo que se conoce como Sub-Recursos. Los Sub-Recurso siempre esta asociados a un Recurso mediante el ID

El formato de URI para los Sub-Recursos es el siguiente.

<URL base>/{recurso}/{id}/subrecurso

> **NOTA**
> Para el Sub-Recurso y sus atributos se aplica las mismas reglas que para los Recursos.

Ejemplos de Sub-Recursos

Operacion del Servicio	Metodo HTTP	URI	Descripción
listarDomiciliosFiscalesContribuyente	GET	/contribuyentes/10245577849/domiciliosfiscales	Obtener todos los DomicilioFiscal del contribuyente con id 10245577849

agregarDomicilioFiscalContribuyente	POST	/contribuyentes/10245577849/domiciliosfiscales	Agregar un DomicilioFiscal al contribuyente con id 10245577849
actualizarDomicilioFiscalContribuyente	PUT	/contribuyentes/10245577849/domiciliosfiscales/11	Actualizar el DomicilioFiscal con id 11 del contribuyente con id 10245577849
eliminarDomicilioFiscalContribuyente	DELETE	/contribuyentes/10245577849/domiciliosfiscales/12	Elliminar el DomicilioFiscal con id 12 del contribuyente con id 10245577849

NOTA
Si los recursos relacionados son de diferentes dominios, no se pueden representar como sub-recursos (ver el lineamiento mas abajo), en ese caso ambos Recursos deben tener sus propias URIS en su propio dominio, con sus respectivas operaciones y la relacion entre ellos se establece mediante filtros (ver Filtros)

LINEAMIENTO
Las URI compuestas por más de 1 recurso (Sub-Recursos) no deben ser formados por el cliente, si no, solo deben ser accedidas siguiendo hyperlinks (ver Hyperlinks)

Ejemplo: Si se desea acceder al Sub-Recurso DomicilioFiscal, mediante el Recurso Contribuyente, se deben efectuar 2 Request como minimo

1er Request: GET http://api.sunat.peru/v1/contribuyentes/10405678945

Response

```
{
    ...
 ...
 "_links":[ { ....., ....,
 {"domiciliosfiscales":"http://api.sunat.peru/v1/contribuyentes/1040
5678945/domiciliosfiscales"}]
}
```

2do Request: GET http://api.sunat.peru/v1/contribuyentes/10405678945/domiciliosfiscales

LINEAMIENTO
En las URI compuestas por más de 1 recurso (Sub-Recursos), no se deben referenciar a Recursos de Otros Dominios

Ejemplo: Si Contribuyente y AccionFiscalizacion pertenecen a diferentes Dominios (Contribuyentes y Auditoria respectivamente)

Entonces no se puede formar la siguiente URI y tampoco puede ser devuelta mediante un Hiperlink

GET /v1/auditoria/tributario/contribuyentes/10405678945/accionesfiscalizacion

LINEAMIENTO
En las URI no se pueden poner 2 recursos juntos, siempre deben ir separados por un ID

Ejemplo: Si Contribuyente y DomicilioFiscal pertenecen al mismo Dominio

No se puede formar la siguiente URI y tampoco puede ser devuelta mediante un Hiperlink

GET /v1/auditoria/tributario/contribuyentes/domiciliosfiscales

LINEAMIENTO
Los servicios REST de tipo TAREA no tienen Sub-Recursos

4. Limitar atributos

Ya que un Recurso puede tener multiples atributos, es posible realizar un filtrado de estos para que el resultado devuelva solo los atributos que deseamos

para ello usaremos la variable f (fields) = para seleccionarlos.

Sintaxis

```
/{Recurso}[/{id}]?f=[campo 1],[campo 2],...[campo N]
```

Ejemplo:

GET http://api.sunat.peru/v1/valores/0710010071784?f=numPeriodo,fecEmision,mtoSaldo

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
{
 "numPeriodo":"200801" ,
 "fecEmision":"2015-06-02T21:03:11.78-05:00",
 "mtoSaldo":100.00
}
```

5. Filtrar Recursos

Los Filtros nos permiten limitar los Recursos devueltos, en base al valor de sus atributos.

Sintaxis

```
/{Recurso}?a1=valor1[&a2=valor2][&....]
```

Ejemplo:

GET http://api.sunat.peru/v1/contribuyente/comprobantes?numRuc=201456748541&serie=267871

```
HTTP/1.1 200 OK
Content-Type: application/json
[{
 "nroComprob":"12"
 "numRuc" : "201456748541",
 "tipDoc" : "1" ,
 "serie" : "267871"
 "fecAct" : "2015-06-02T21:03:11.78-05:00" ,
  },
{
 "nroComprob":"13"
 "numRuc" : "201456748541",
 "tipDoc" : "1" ,
 "serie" : "267871"
 "fecAct" : "2015-06-02T21:03:11.78-05:00" ,
}]
```

Nota
Si el siguiente query

GET /v1/contribuyente/comprobantes?numRuc=201456748541&serie=267871

No encuentra ningun recurso que coincida con los filtros

El Servidor retorna:

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
[]
```

Donde [ ] es una lista vacia

6. Ordenamiento

Es posible ordenar los Recursos devueltos, por uno o mas atributos, para ello utilizar la siguiente sintaxis

Sintaxis

```
/v1/{dominio}/{Recurso}?s=-atrib1,atrib2
```

Ejemplo:

GET http://api.sunat.peru/v1/contribuyente/comprobantes?s=-numRuc,serie

Ordena los comprobantes de forma descendente por el numRuc y la serie

7. Paginación

Esta funcionalidad nos permite devolver los resultados en paginas de acuerdo a los parametros siguientes:

page: el numero de página

per_page: el numero de registros que nos devuelve cada página

Sintaxis

```
/v1/{dominio}/{Recurso}?page=valor1&per_page=valor2
```

Ejemplo:

GET http://api.sunat.peru/v1/contribuyente/comprobantes?page=1&per_page=10

8. Hyperlinks

Sintaxis

```
"links":[
         {
          "rel": "self",
          "href": "<URI absoluto al recurso>"
         },
         {
          "rel":"<otra rel>",
          "href":"<URI absoluto del recurso relacionado>"
         }


         ]

// Cuando se haga referencia al mismo Recurso usar "self"
```

Ejemplo:

GET http://api.sunat.peru/v1/auditoria/tributaria/e/accionesfiscalizacion/20140901451

```
{
   "numeroAccionFisca": "20140901451",
   "fechaGeneracion": "2015-06-02T21:03:11.78-05:00",
   "duracion" : "30",
   "links":[

      {
         "rel": "self",
         "href":
"http://api.sunat.peru/v1/auditoria/tributaria/e/accionesfiscalizacion/201
40901451"
      },
      {
         "rel": "funcionarios",
         "href":
"http://api.sunat.peru/v1/auditoria/tributaria/accionesfiscalizacion/20140
901451/funcionarios"
      }
   ],
}
```


9. Recursos Embebidos

Cuando se necesite evitar varias consultas para obtener un recurso y ademas sus referencias a recursos relacionados. podemos usar el concepto de recursos embebidos.

e (embed) : Parametro para definir los recursos Relacionados que se desean obtener

```
http://api.sunat.peru/v1/{dominio}/[Recurso]?e=[recurso 1],[recurso
2],...[recurso 3]

Request :
http://api.sunat.peru/v1/{dominio}/accionesfiscalizacion/20140901451?e=car
ta,funcionario.division

Response:
{"numeroAccionFisca": "20140901451",
      "fechaGeneracion": "2015-06-02T21:03:11.78-05:00",
      "duracion" : "30",
      "carta":{
       "numeroCarta" : "C0001",
       "fechaCreacion" : "2015-06-02T21:03:11.78-05:00"
       "fechaAprobacion" : "2015-06-02T21:03:11.78-05:00"
       "codigoPlantilla" : "P11002014"
      }
      "funcionario": {
       "division" : "Division de Fiscalizacion"
      }

     }
```

LINEAMIENTO
Los Recursos embebidos no pueden ser recursos de otros Dominios

Ejemplo: Si Contribuyente y AccionFiscalizacion pertenecen a diferentes Dominios (Contribuyentes y Auditoria respectivamente)

Entonces **NO** se puede formar la siguiente URI

GET /v1/auditoria/tributario/contribuyentes/10405678945?e=accionesfiscalizacion

10. Content Media Type

Para indicar el formato del recurso que se consulta , se debe usar en el Header el internet media type que corresponda. Algunos ejemplos:

Media Type		
application/json	application/gzip	application/zip
application/pdf	text/plain	text/csv
application/xml	image/jpeg	audio/mpeg
	image/gif	video/mp4
	image/png	video/mpeg

Ejemplo

Request

```
GET /v1/contribuyente/contribuyentes
Accept: application/json
```

En el ejemplo previo el cliente espera la Respuesta en Formato json.

Nota
Si el Servidor no soporta el media type solicitado, este devuelve lo siguiente

Response

```
HTTP/1.1 406 Not Acceptable
Content-Type: application/json
{ json de Respuesta}
```

El Detalle del json de respuesta de acuerdo al Error lo puede ver aqui

LINEAMIENTO
El formato de la respuesta se debe espeficar en el Header

Ejemplo:

GET /v1/contribuyente/contribuyentes
Accept: application/json

LINEAMIENTO
En la URI no se debe especificar el formato de la respuesta

Ejemplos de URIs no validas

GET /v1/contribuyente/contribuyentes.json

GET /v1/contribuyente/contribuyentes.htm

GET /v1/contribuyente/contribuyentes.xml

11. Estructura del Response y Códigos de Respuesta (HTTP status)

Para el parametro de respuesta "HTTP status" se deben usar los siguientes codigos segun sea el caso:

200 OK - Indica una respuesta satisfactoria

4XX Errores del Cliente

5XX Errores del Servidor

11.1. Response Satisfactorio

Es la respuesta que se devuelve al cliente en el caso que el request termino de forma satisfactoria. En este caso se debe utilizar el codigo de respuesta HTTP 200, con la siguiente estructura:

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
{ json de respuesta }
```

En este caso el Json de respuesta dependera de la especificación del Servicio

11.2. Response con Errores HTTP

LINEAMIENTO
Para el caso de los Errores HTTP de tipo 4XX y 5XX se devolverá el Codigo y Mensaje de error HTTP correspondiente mas una descripcion del error, en el Body del Mensaje, de acuerdo a la tabla adjunta.

Codigo HTTP	Mensaje	Descripción
400	Bad Request	El Request no puede ser entendido por el Servidor debido a errores de Sintaxis, El cliente no debe repetir el Request sin modificaciones
401	Unauthorized	Fallo en la autenticación del Cliente
403	Forbidden	El Cliente no tiene autorización para acceder al Recurso
404	Not Found	El Recurso Solicitado no puede ser encontrado
405	Not Allowed	El Metodo HTTP utilizado en el Request no es soportado por el Recurso
406	Not Acceptable	El Recurso no puede responder al Cliente en el Media Type solicitado en el Request
415	Unsupported Media Type	La Entidad en el Body del Request esta en un Media Type que no es soportado por el Recurso
422	Unprocessable Entity	Se presentaron errores de validacion que impidieron completar el Request
500	Internal Server Error	Se presento una condicion inesperada que impidio completar el Request
503	Service Unavailable	El Servidor no esta disponible temporalmente o esta muy ocupado para responder al Request

Ejemplo de Error HTTP 406:

Response

```
HTTP/1.1 406 Not Acceptable
Content-Type: application/json
{"cod":"406","msg":"Not Acceptable - El Recurso no puede responder al
Cliente en el Media Type solicitado en el Request"}
```

Lineamiento
Para los siguientes casos

GET /v1/contribuyente/conttribuyentes (Nombre de Recurso o URI mal escrita)

GET /v1/contribuyente/contribuyentes/10245577 (Recurso con ID 10245577 no existe)

El Servidor retorna:

Response

```
HTTP/1.1 404 Not Found
Content-Type: application/json
{ json de Respuesta }
```

El Detalle del json de respuesta de acuerdo al Error lo puede ver aqui

LINEAMIENTO
Si el cliente invoca un Metodo HTTP que el Servidor no soporta (o no permite), sobre una URI valida, este retorna:

Response

```
HTTP/1.1 405 Method Not Allowed
Content-Type: application/json
{json de Respuesta}
```

El Detalle del json de respuesta de acuerdo al Error lo puede ver aqui

LINEAMIENTO
Para el caso de los Errores HTTP 422 y 500, que son controlados en el servidor, adicionalmente al Codigo y Mensaje de error HTTP correspondiente, en el body del mensaje se devolvera una estructura de acuerdo a lo definido en la Siguiente Sección Mensajes de Error

12. Mensajes de Error

12.1. Estructura de Mensaje de Error para Condiciones Inesperadas en el Servidor (Exceptions)

Este tipo de Mensajes son devueltos cuando ocurre Errores inesperados en el Servidor (Exceptions ej: Out of Memory, Null Pointer, etc), en este caso se debe utilizar el codigo de respuesta HTTP 500, con la siguiente estructura:

Response

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json
{
 "cod":"500",
 "msg":"Internal Server Error - Se presento una condicion inesperada que
impidio completar el Request",
 "exc":"java.lang.NullPointerException at ..."
}
```

El campo “cod” : Es el codigo de error generico 500 para este tipo de Errores.

El campo “msg”: contiene el mensaje de error asociado al codigo de error 500 (Ver Errores Http)

El campo “exc” : contiene el stack trace del Error

12.2. Estructura de Mensaje de Error para Validaciones

Este tipo de Mensajes son devueltos cuando el Backend quiere informar al cliente que ha ocurrido algun tipo de Error de Validación, en este caso se debe utilizar el codigo de respuesta HTTP 422, con la siguiente estructura:

Response

```
HTTP/1.1 422 Unprocessable Entity
Content-Type: application/json
{
 "cod":"422",
 "msg":"Unprocessable Entity - Se presentaron errores de validacion que
impidieron completar el Request",
 "errors":[{
        "cod":"XXXXX",
        "msg":"El ruc 1234 no es valido",
           },
           {
          "cod":"YYYYY",
          "msg":"La sumatoria de los montos 3456.00 no coinciden con el
total 3457.00",
           }]
}
```

El campo “cod” : Es el codigo de error generico 422 para este tipo de Errores.

El campo “msg”: Contiene el mensaje de error asociado al codigo de error 422 (Ver Errores Http)

La estructura “errors” : Es una lista de errrores, con los siguientes atributos:

"cod" : Codigo del error de validacion, debe tener la forma NNNNN

"msg" : Descripcion simplifcada del error (max 100)