REGIONAL PUBLIC GOODS

Integration and Trade Sector

IDB

Transformación Digital en Salud para Mitigar los Efectos de COVID-19 en América Latina y el Caribe

Presentación de JSON - REST

RG-T3769

REGIONAL PUBLIC GOODS

Integration and Trade Sector

Json - Rest

Autor
amedina@cens.cl

IDB

RG-T3769



JSON

¿Qué es JSON?

JSON significa JavaScript Object Notation

- Descubierto por Douglas Crockford
- Es una reimplementación de Javascript Object
- Puede ser usado en múltiples lenguajes de programación
- Es un lenguaje de intercambio de datos liviano y fácil de leer
- Se describe en Texto Plano

```
{
    “key1”: “value1”,
    “key2”: [“uno”, “dos”],
    “key3”: true,
    “key4”: {
        “key5”: “value2”
    },
    “key6”: 2021
}
```

Reglas de Sintaxis de JSON

- Los datos se representan en pares de clave/valor
- Los datos se separan mediante comas
- Es posible representar otros objetos mediante llaves { }
- Es posible representar listas de elementos mediantes [ ]

```
{
    “key1”: “value1”,
    “key2”: [“uno”, “dos”],
    “key3”: true,
    “key4”: {
        “key5”: “value2”
    },
    “key6”: 10
}
```

Tipos de Datos en JSON

- string
- int
- otro objeto json
- arreglos
- booleano
- null

```
{
    “nombre”: “Alejandro”,
    “hijas”: [“Victoria”, “Beatriz”],
    “activo”: true,
    “contacto”: {
        “telefono”: “+564567”
    },
    “edad”: 34
}
```

JSON y XML

- Json y XML son fáciles de leer por humanos
- Json y XML son jerárquicos
- Json y XML pueden ser parseados por múltiples lenguajes de programación
- Json y XML pueden ser transmitidos mediante requests

```
{
    “key1”: “value1”,
}
```

```
<key1>value1</key1>
```



REST

Qué es REST?

REST REpresentational State Transfer (Transferencia de Estado Representacional)

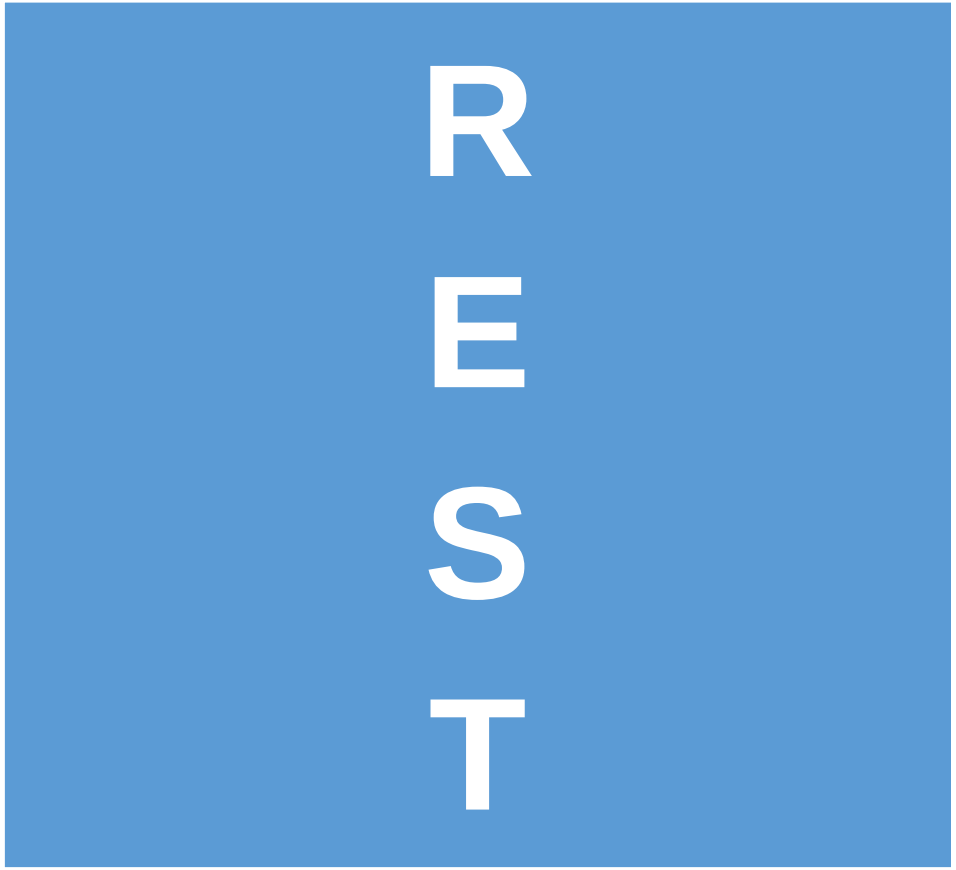
- Es un conjunto de condiciones, métodos y estilo arquitectónico que lleva a un escalable, confiable y fácil uso de interfaces.
- El lanzamiento de REST cambió por completo el desarrollo de software.
- Aprovecha de mejor manera el uso de HTTP

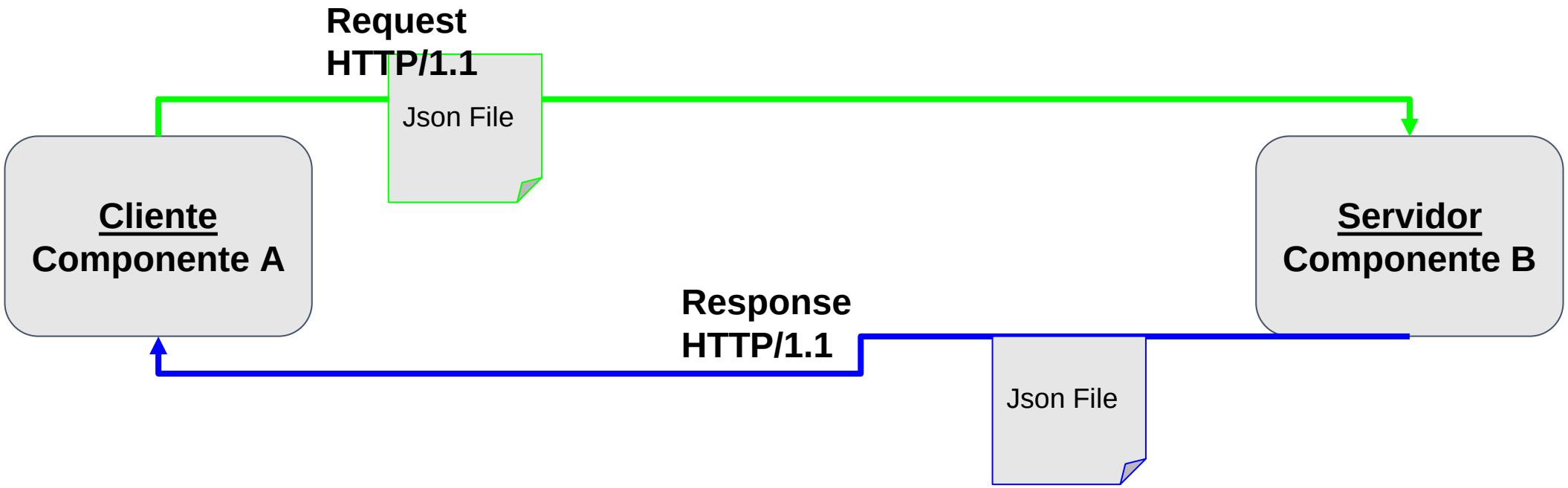


https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

REST: Principios Fundamentales

- Cliente-Servidor: El cliente es un componente independiente del servidor.
- Sin Estado: Cada petición contiene toda la información para ejecutar una acción
- Cacheable: Posibilidad de utilizar caché para peticiones recurrentes
- Interfaz Uniforme: Simplifica y desacopla la arquitectura
- Sistema de Capas: Los componentes no pueden ver más allá de su capa




Request
HTTP/1.1
Json File
Cliente
Componente A
Servidor
Componente B
Response
HTTP/1.1
Json File

REST: Request/Petición

- Método
- URI
- Versión del Protocolo
- Cabeceras
- Cuerpo del Mensaje

```
POST /autos/ HTTP/1.1
Host: autos.venta.cl
Content-Type:
application/json
Accept: application/json

{
  "marca":"peugeot",
  "color":"azul"
}
```

REST: Método/Verbo

- Representa la acción que se desea realizar.
- Mediante los métodos le indicamos al servidor la forma en que debe de tratar una determinada petición.
- Una misma URL puede ser tratada de forma diferente por el servidor.

POST

GET

PUT

DELETE

REST: URI/Path

- Identifica a una entidad/recurso en un servidor específico al cual se desea acceder o se desea ejecutar una acción.
- Puede contener parámetros de búsqueda
- Puede referirse a un elemento específico o a todos los elementos.

/
/autos
/autos/25
/autos?color=azul

REST: Cabeceras/Headers

- Representa la información que tiene relación con el mensaje que se desea transmitir o de una respuesta.
- Permite establecer ciertas características:
  - **Cabeceras generales:** Afectan a todos los mensajes.
  - **Cabeceras de petición:** Se utilizan en peticiones
  - **Cabeceras de respuesta:** Se utilizan en respuestas
  - **Cabeceras de Entidad:** Se aplican al cuerpo del mensaje

Content-Type
Accept
Authorization
Server
Host
Date
Content-Length
User-Agent

REST: Cuerpo del Mensaje/Body

- Representa la información del elemento.
- Puede ser información parcial en ciertas operaciones.
- No todas las operaciones tienen un cuerpo del mensaje.

JSON ⟶
```
{
  “key1”: ”value1”,
  “key2”: “value2”
}
```

XML ⟶
```
<?xml version="1.0" encoding="UTF-8"?>
<ficha>
  <nombre> Gabriel </nombre>
</ficha>
```

REST: Request

```
POST /autos/ HTTP/1.1
Host: autos.venta.cl
Content-Type: application/json
Accept: application/json

{
  "marca":"peugeot",
  "color":"azul"
}
```

REST: Response

- Versión del Protocolo
- Código de estado
- Cabeceras
- Cuerpo del Mensaje

```
HTTP/1.1 201 CREATED
Server: nginx/1.14.0 (Ubuntu)
Date: Mon, 10 May 2021
17:41:21 GMT
Content-Type: application/json
Location: /autos/1

{
  "id": "1",
  "marca": "peugeot",
  "color": "azul"
}
```

REST: Códigos de Estado

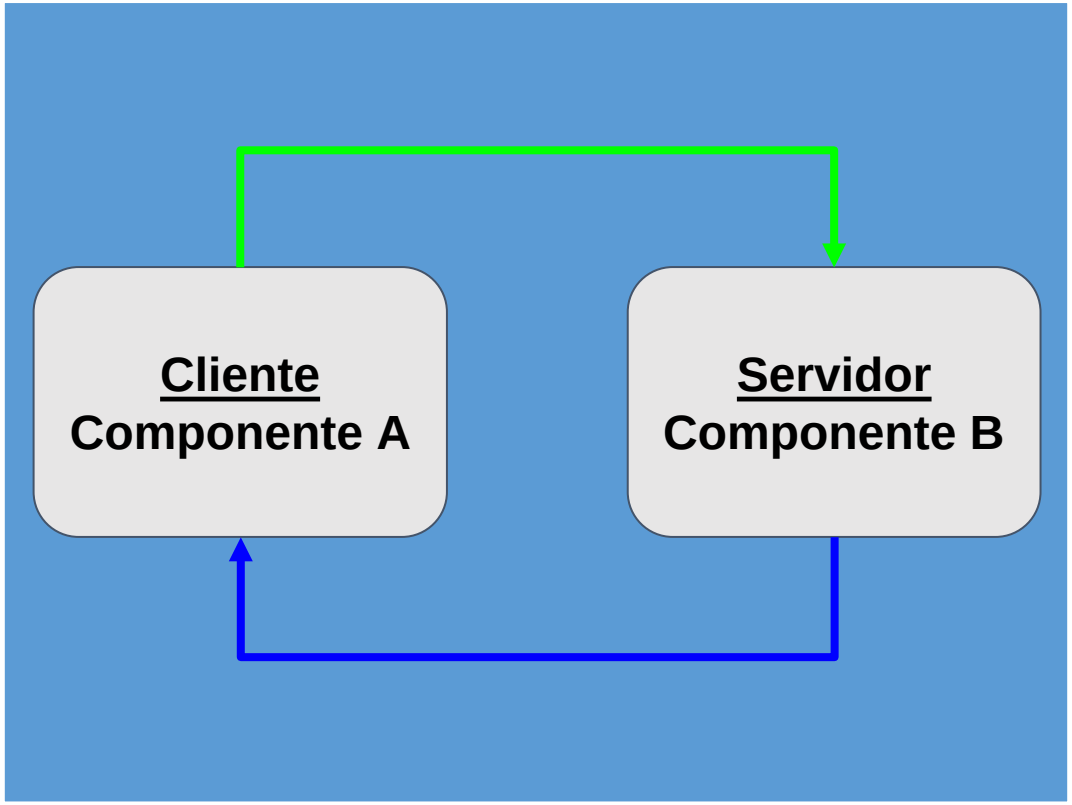
- Respuestas informativas (100–199)
- Respuestas satisfactorias (200–299)
- Redirecciones (300–399)
- Errores de los clientes (400–499)
- Errores de los servidores (500–599)

200 Ok
201 Created
204 No Content
404 Not Found
401 Unauthorized
400 Bad Request
500 Internal Server Error
301 Moved Permanently

REST: Básicas Operaciones

Request HTTP/1.1

- Crear: Método POST, con contenido en el cuerpo del mensaje
- Leer: Método GET, sin contenido en el cuerpo del mensaje
- Actualizar: Método PUT, con contenido a actualizar en el cuerpo del mensaje
- Eliminar: Método DELETE, sin contenido en el cuerpo del mensaje



REST: Crear Recursos / POST

- Permite crear recursos
- Es necesario enviar como contenido los datos del elemento a crear

```
POST /autos/ HTTP/1.1
Host: autos.venta.cl
Content-Type:
application/json
Accept: application/json

{
  "marca": "peugeot",
  "color": "azul"
}
```

REST: Actualizar Recursos/PUT

- Permite actualizar recursos
- Es necesario enviar como contenido los datos del elemento a actualizar
- Es necesario tener el identificador único del recurso a actualizar

```
PUT /autos/1 HTTP/1.1
Host: autos.venta.cl
Content-Type:
application/json
Accept: application/json

{
  "id": "1",
  "marca": "peugeot",
  "color": "azul"
}
```

REST: Eliminar Recursos/DELETE

- Permite eliminar recursos
- No es necesario enviar contenido

```
DELETE /autos/1 HTTP/1.1
Host: autos.venta.cl
Accept: application/json
```

REST: Obtener Recursos/GET

- Permite obtener recursos
- Existen 3 modalidades básicas:
  - Obtener un elemento mediante su identificador.
  - Obtener todos los elementos.
  - Filtrar elementos

```
GET /autos/1 HTTP/1.1
Host: autos.venta.cl
Accept: application/json

GET /autos HTTP/1.1
Host: autos.venta.cl
Accept: application/json

GET /autos?color=azul HTTP/1.1
Host: autos.venta.cl
Accept: application/json
```

REST: Parámetros de Búsqueda

- Pueden ser únicos o múltiples
- Un parámetro puede tener más de un valor
- Es posible agregar modificadores o prefijos en ciertos casos
- La resolución de estos modificadores o prefijos depende de la implementación

```
/autos?color=azul
/autos?color=azul&marca=peugeot
/autos?color=[rojo,azul]
/autos?_page=5&_count=10
```



Demostración REST

Herramienta: https://my-json-server.typicode.com/

REST: Parámetros de Búsqueda

- https://my-json-server.typicode.com/janol77/demo

```
/autos?color=azul
/autos?marcaId=1
/autos?litrosEstanque=40&color=azul
/marca?nombre=jac
```



JSON SCHEMA

JSON Schema: Qué es?

- Es una estructura que permite validar, documentar y controlar la interacción con datos en JSON
- Provee un contrato para que los datos en un archivo json se encuentren correctamente estructurados
- JSON Schema está basado en lo conceptos de XML Schema (XSD)
- No tiene una extensión definida formalmente pero se acostumbra usar *.schema.json

```
{ "type": "number" }
```

42 → OK
1 → OK
-1 → OK
“23” → NOK

JSON Schema: Qué es un Schema?

- Permite definir cómo se debe estructurar los datos, especificando características que deberían tener los datos.
- La especificación define el formato es que estas definiciones se deben realizar.
- Es posible controlar formato, extensión, tipo de dato, expresión regular, etc.

```
{
        "type": "object",
        "properties": {
                "first_name": {
                        "type": "string"
                },
                "last_name": {
                        "type": "string"
                },
                "birthday": {
                        "type": "string",
                        "format": "date"
                },
                "address": {
                        "type": "object",
                        "properties": {

        "street_address": {

        "type": "string"
                                        },
                                        "city": {

        "type": "string"
                                        },
                                        "country":
{

        "type": "string"
                                        }
```

https://json-schema.org/understanding-json-schema/

Guía: Technical Specifications for EU Digital COVID Certificates

Schema: https://github.com/ehn-dcc-development/ehn-dcc-schema

JSON Schema: Validadores Online

- https://www.jsonschemavalidator.net/
- https://json-schema.hyperjump.io/

```
{
  "ver": "1.3.0",
  "nam": {
    "fn": "Araya Rojas",
    "fnt": "ARAYA<ROJAS",
    "gn": "Antonio Javier",
    "gnt": "ANTONIO<JAVIER"
  },
  "dob": "1986-11-08",
  "v": [
    {
      "tg": "840539006",
      "vp": "1119349007",
      "mp": "EU/1/20/1507",
      "ma": "ORG-100031184",
      "dn": 1,
      "sd": 2,
      "dt": "2021-06-11",
      "co": "CL",
      "is": "Ministerio de Salud de Chile",
      "ci":
"URN:UVCI:01:CL:DADFCC47C7334E45A906DB12FD859FB7#1"
    }
  ]
}
```

JSON Schema: Validador de Ejemplo

- Lenguage: Python
- Librería: https://github.com/marksparkza/jschon

```
{
  "ver": "1.3.0",
  "nam": {
    "fn": "Araya Rojas",
    "fnt": "ARAYA<ROJAS",
    "gn": "Antonio Javier",
    "gnt": "ANTONIO<JAVIER"
  },
  "dob": "1986-11-08",
  "t": [
    {
      "tg": "840539006",
      "tt": "LP6464-4",
      "nm": "ELITechGroup, SARS-CoV-2 ELITe MGB® Kit",
      "sc": "2021-06-11T17:30:00Z",
      "tr": "260415000",
      "tc": "Hospital de Rengo",
      "co": "CL",
      "is": "Ministerio de Salud de Chile",
      "ci":
"URN:UVCI:01:CL:DADFCC47C7334E45A906DB12FD859FB7#1"
    }
  ]
}
```

