Universidad Nacional de Quilmes

CONSTRUCCIÓN DE INTERFACES DE USUARIO

2do Cuatrimestre de 2018

Resúmen

- JSON
- Protocolo HTTP
- API REST
- Alicaciones Stateles vs Stateful

JSON
(Javascript Object Notation)

- Estándar basado en texto plano
- No es parte de Javascript
- Independiente de cualquier lenguaje de programación
- Formato de intercambio de datos
- Simple de entender
- Alternativa al XML
- Uno de los usos es para transferencia de información.

JSON
Como se forma?

{ } Para delimitar el inicio y fin de objetos

“ ” Para atributos

, Separación de atributos y valores no numéricos

[ ] Arreglos

OJO: con los espacios y las comas

JSON Ejemplos

```
{
  "nombre": "SOLEPCC",
  "valor1": 1351824120,
  "sensors": [
    {
      "name": "Sensor Humedad",
      "description": "Mide Humedad",
      "sensor-type": "Humedad"
    },
    {
      "name": "Sensor temperatura",
      "description": "Mide Temperatura",
      "sensor-type": "Temperatura"
    }
  ]
}
```

```
{
    "libro": [
        {
            "id": "01",
            "lenguaje": "Java",
            "edición": "tercera",
            "autor": "Herbert Schildt"
        },
        {
            "id": "07",
            "lenguaje": "C++",
            "edición": "seguna",
            "autor": "E.Balagurusamy"
        }
    ]
}
```

Protocolo HTTP
Hypertext Transfer Protocol

- Protocolo de Capa de aplicación (nivel 7 del Modelo OSI)
- Comunicación para la transferencia de información en la WWW.
- Sin estado, es decir, no almacena información sobre conexiones anteriores.
- Orientado al esquema Petición-Respuesta entre un cliente y un servidor

Protocolo HTTP

Protocolo HTTP
Algunos métodos de Petición

- GET: Solo para recuperar información, parámetros deben ser enviados por URL.
- POST: Envía datos para ser procesados, los datos deben ser enviados en el cuerpo (body) de la petición.
- PUT: Sube, carga o realizar un “upload” de un recurso específico (file)
- DELETE: Elimina un recurso específico

Protocolo HTTP
Códigos de Respuesta

1xx: Mensaje informativo.

2xx: Exito

200 OK

201 Created

202 Accepted

204 No Content

3xx: Redirección 300 Multiple Choice

301 Moved Permanently

302 Found

304 Not Modified ?

4xx: Error del cliente

400 Bad Request

401 Unauthorized

403 Forbidden

404 Not Found

5xx: Error del servidor

500 Internal Server Error

501 Not Implemented

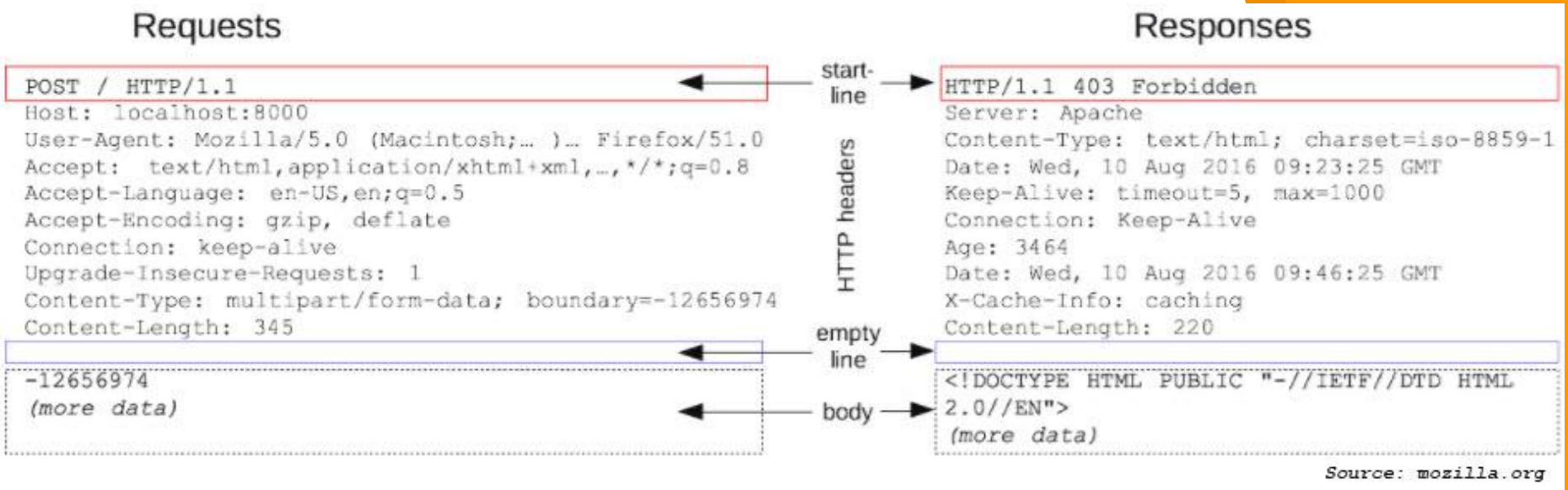
502 Bad Gateway

503 Service Unavailable

Protocolo HTTP
Códigos de Respuesta

- 200: OK (Petición correcta)
- 400: Bad Request (Petición incorrecta) el servidor no pudo interpretar la solicitud, por una sintaxis errónea.
- 404: Not Found (Recurso no encontrado)
- 500: internal Server Error (error no controlado) El servidor no puede controlar una excepción.

Protocolo HTTP - Estructura



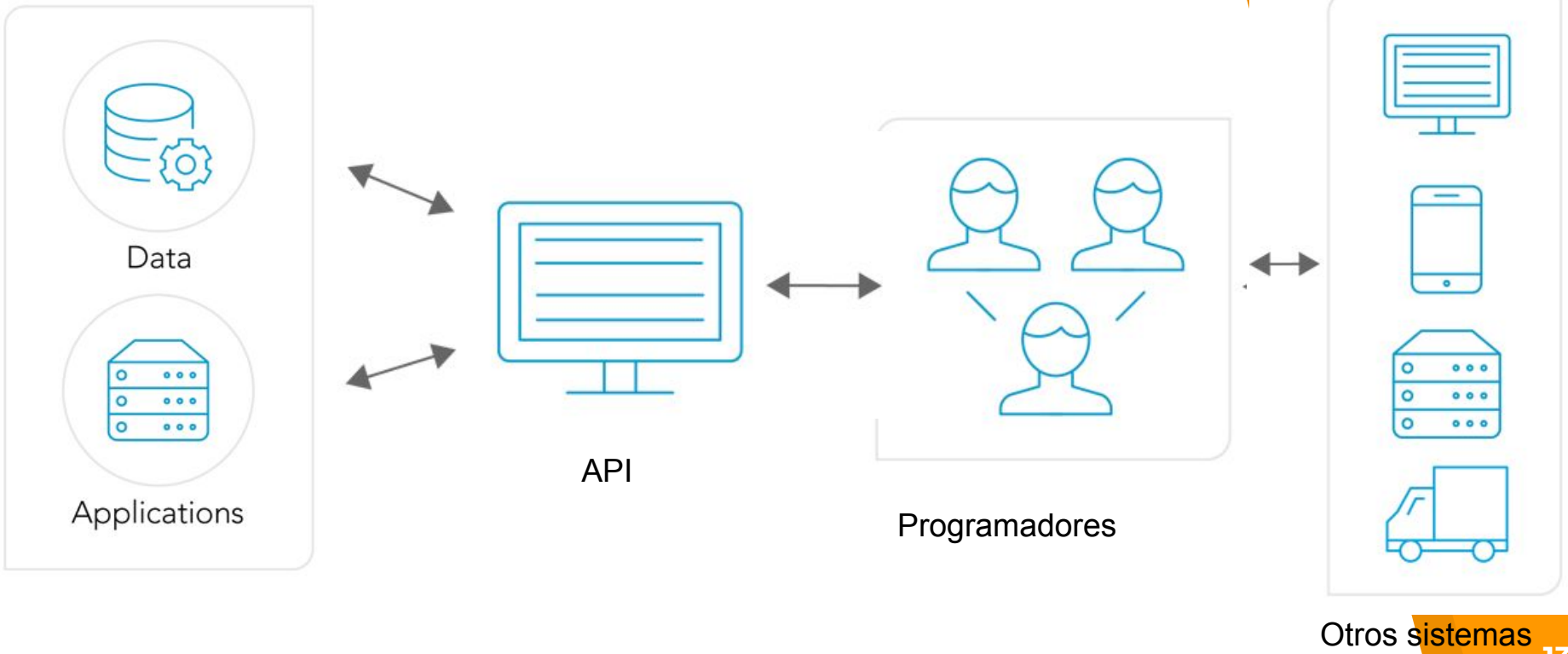
Source: mozilla.org

API
Application Programming Interface

Interfaz de programación de aplicaciones

Conjunto de subrutinas, funcionalidades y procedimientos expuestos para ser utilizados por otro software.

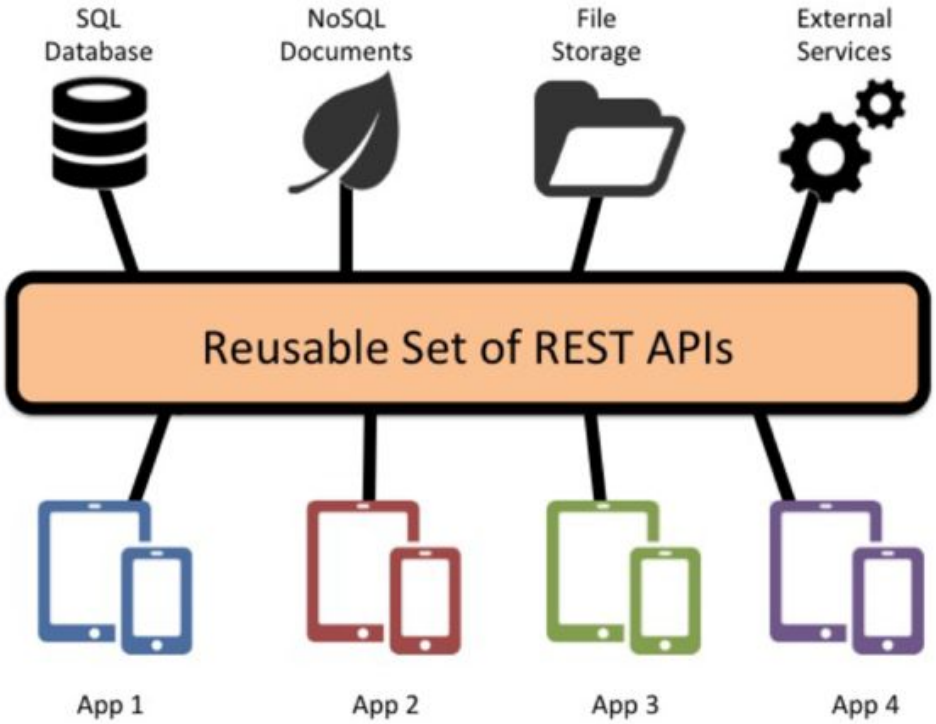
API
Application Programming Interface



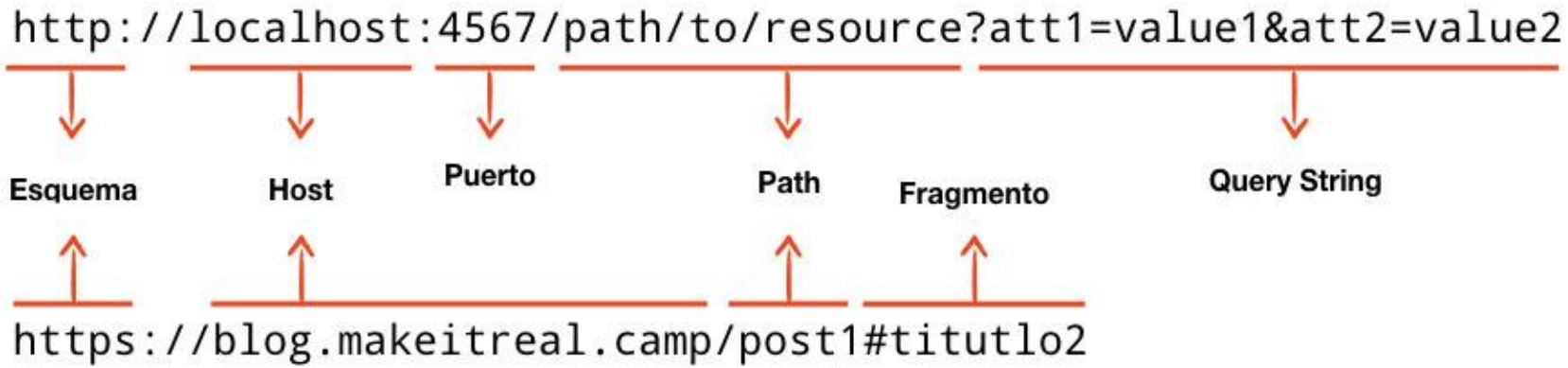
API REST
Representational State Transfer

Cualquier interfaz entre sistemas que utilicen el protocolo HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON.

API REST


SQL Database
NoSQL Documents
File Storage
External Services
Reusable Set of REST APIs
App 1
App 2
App 3
App 4

API REST
Estructura de URL



API REST
Buenas prácticas de diseño

1.	*GET /tickets*	Devuelve una lista de tickets
2.	*GET /tickets/12*	Devuelve el ticket #12
3.	*POST /tickets*	Crea un nuevo ticket
4.	*PUT /tickets/12*	Actualiza el ticket #12
5.	*DELETE /tickets/12*	Elimina el ticket #12

API REST
Buenas prácticas de diseño

1. Separación en recursos lógicos
2. Descripción de recursos como sustantivos y no verbos
    - Ejemplos: pelicula, usuario, video
3. No revelar datos de la implementación.
4. Definición Acciones, mediante protocolo HTTP

API REST - Ejemplos XTRest

1. Ver ejemplo
2. Descargar Postman o cualquier cliente Rest.
3. Consumir API
4. Exponer API

API REST
Ejemplos de mercado

- https://developer.twitter.com/en/docs/api-reference-index.html
- https://www.instagram.com/developer/
- http://petstore.swagger.io/
- http://developer.ted.com/API_Docs
- https://www.football-data.org/

API REST - Ejemplos XTRest

Problema recursión infinita en asociaciones bi-direccional

Pelicula - > listAutores

Autor - > Película

“com.fasterxml.jackson.databind.JsonMappingException: Infinite recursion (StackOverflowError) ”

@JsonIgnore: Ignorar el campo

@JsonBackReference : Cortar la serialización

Stateless vs Stateful

Stateless - Sin estado

- El protocolo HTTP es sin estado.
- El servidor no almacena peticiones anteriores
- Tras responder una petición, se cierra la conexión
- El servidor toma cualquier petición como una nueva.
- El servidor no retiene información de sesión
- No hay necesidad de liberar recursos tomados

Aplicaciones
Stateless vs Stateful

Stateful - CON estado

- El servidor mantiene información de sesión, con el objetivo de representar flujos de trabajos y estados.
- NO se persiste el estado, no se mantienen al reiniciar los servidores. Información volátil
- Diferentes peticiones pueden mantener datos compartidos. Se mantiene información del cliente a lo largo de varias invocaciones.
- Es necesario más capacidad de hardware

Aplicaciones
Stateless vs Stateful

Stateful - Con estado

- Uso de cookies, proporcionan la forma de almacenar información en el cliente que le permita al servidor reconstruir la sesión. Incluyen datos generados por el servidor
- Tas una petición se mantiene activa una conexión
- Necesidad de reconstruir la historia de peticiones anteriores
- Necesidad de liberar recursos ante el no uso prolongado.
- Contexto dinámico.

	Con Estado	Sin Estado
Mantiene información de sesión	SI	NO
Diferentes peticiones pueden compartir información	SI	NO
Mantiene información volátil	SI	NO
Contexto y tamaño predecible	NO	SI
Necesidad de liberación de recursos (garbage collector)	SI	NO
Necesidad de reconstrucción ante caídas	SI	NO