API REST

Conceptos API

Introducción a las API

Definición: API significa Interfaz de Programación de Aplicaciones (Application Programming Interface).

Función: Permite la comunicación entre diferentes sistemas de software.

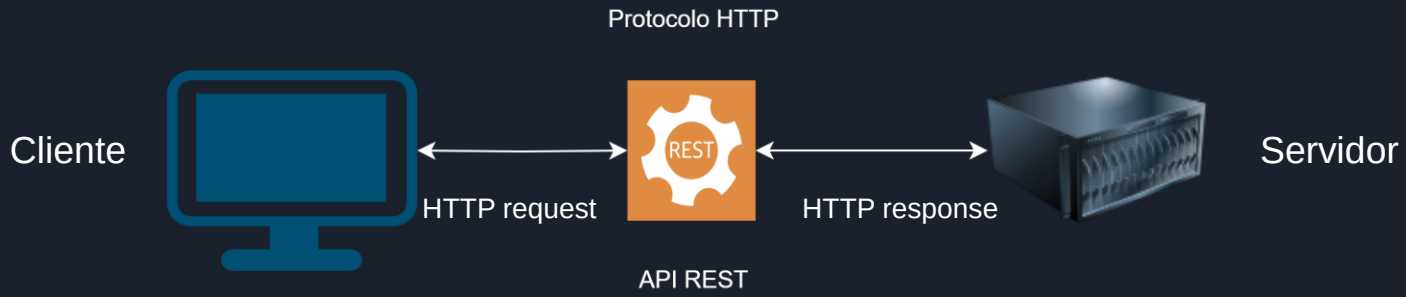
Ejemplo: Cómo una app de java obtiene datos desde de un servicio de base de datos.

Introducción a las API REST

- **REST** (Representational State Transfer) es un acrónimo para las siglas REpresentational State Transfer fue definido por Roy Fielding en su tesis doctoral en el año 2000.
- **REST**: Es una lógica de restricciones y recomendaciones bajo la cual se construye la API ***(estilo de arquitectura)***
- **Motivación**: Crear un estándar para sistemas distribuidos escalables y flexibles.

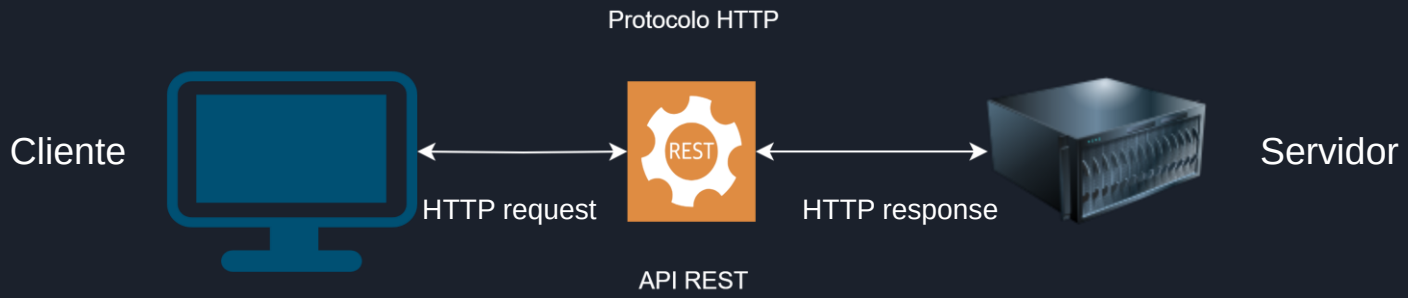
API REST

Funcionamiento



API REST - Características

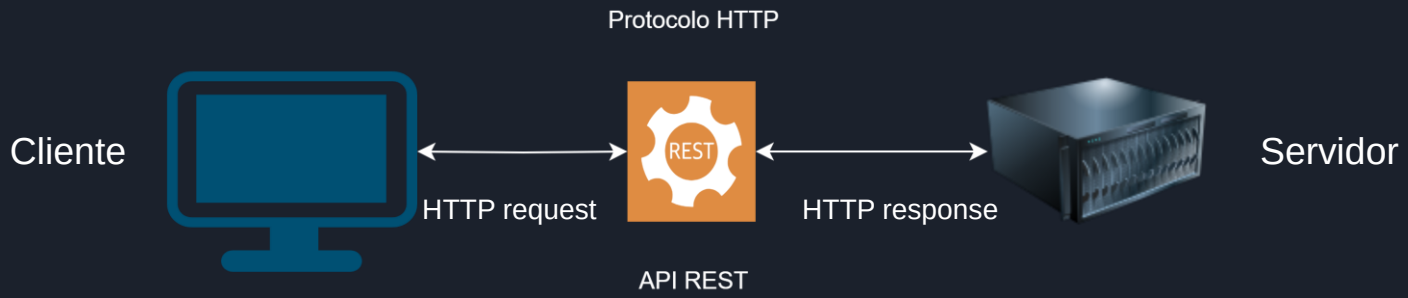
Funcionamiento



1. Client-Server
Establece que el cliente y el servidor deben estar separados. El cliente no debe saber qué hay en el servidor ni viceversa.

API REST - Características

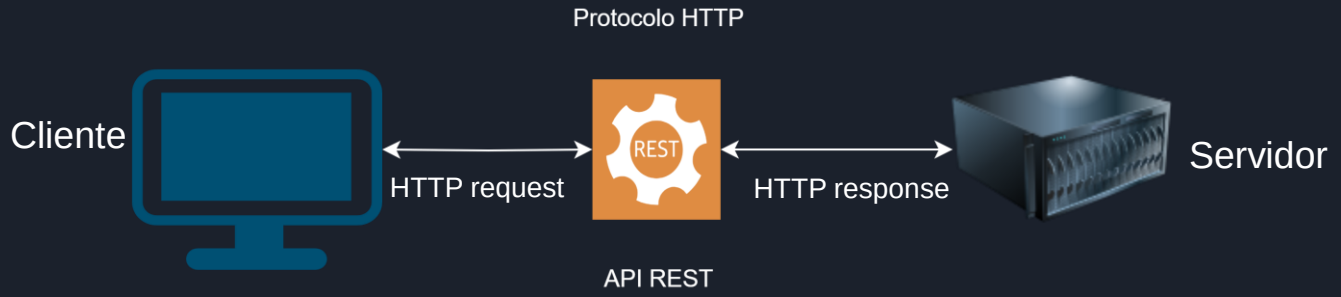
Funcionamiento



2. Stateless (sin estado)
El estado necesario para manejar cada request debe estar contenido en el mismo request. No se debe guardar session state variables del lado del servidor.

API REST - Características

Funcionamiento



3. Cacheable

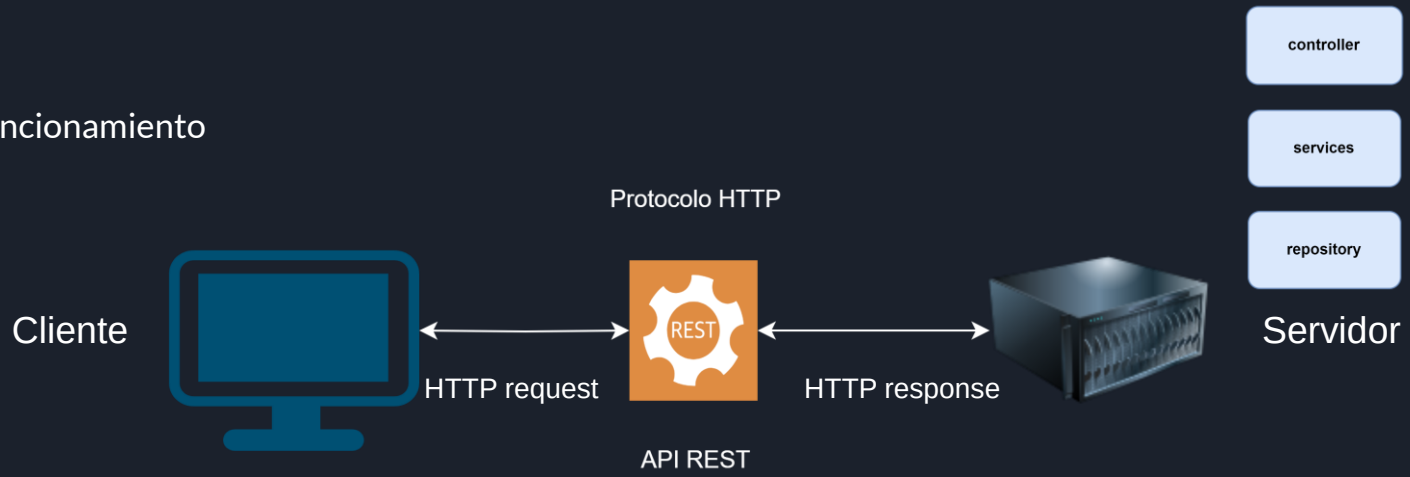
Cada mensaje de respuesta debe especificar explícitamente si puede ser cacheado o no. De este modo podemos eliminar algunas interacciones cliente-servidor y prevenir que el cliente use data desactualizada.

Esto se hace por medios de los encabezados HTTP Cache-Control, Expires, y ETag : Ejemplo

```
HTTP/1.1 200 OK
Cache-HTTP/1.1 200 OK
Cache-Control: public, max-age=3600
...
```

API REST - Características

Funcionamiento



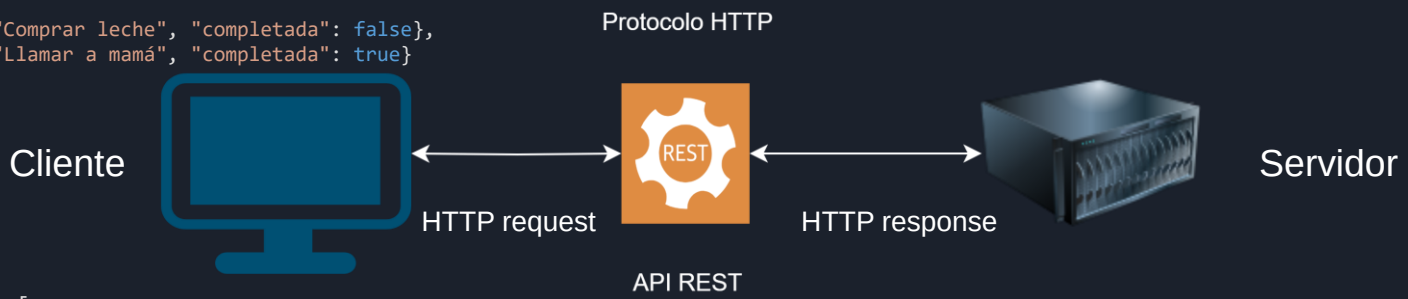
4. Layered system

El cliente no debería poder identificar a qué capa del sistema está conectado. El servidor debe restringir el conocimiento a una sola capa (outer-face contract).

API REST - Características

```
[
    {"id": 1, "tarea": "Comprar leche", "completada": false},
    {"id": 2, "tarea": "Llamar a mamá", "completada": true}
]
```

Funcionamiento



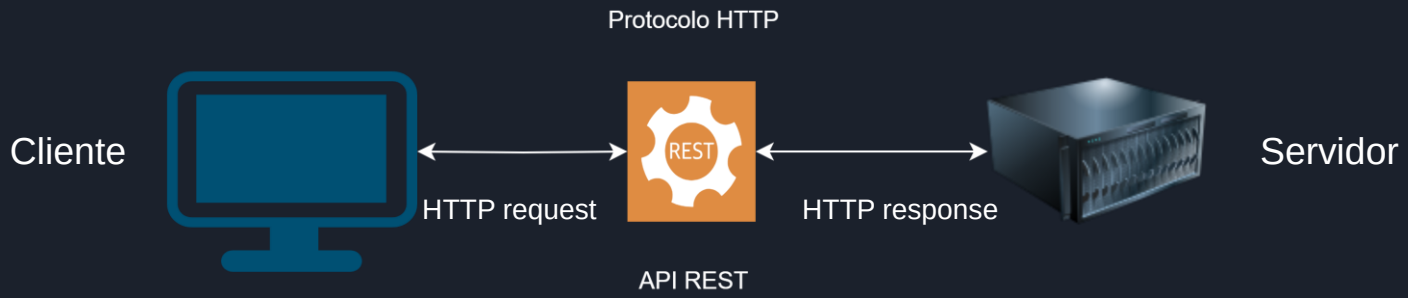

```
{
    "tasks": [
        {"id": 1, "tarea": "Comprar leche", "completada": false},
        {"id": 2, "tarea": "Llamar a mamá", "completada": true}
    ],
    "script": "function mostrarAlertasDeTareas(tareas) { tareas.forEach(t => { if(!t.completada) { alert('Tarea pendiente: ' + t.tarea); } }); } mostrarAlertasDeTareas(tasks);"
}
```

5. Code on demand (Código bajo demanda, opcional)

Los servidores pueden proporcionar código ejecutable (JavaScript) al cliente, permitiendo que este código se ejecute en el contexto del cliente. Este principio es opcional y se utiliza para extender la funcionalidad del cliente.

API REST - Características

Funcionamiento



6. Uniform interface

Establece que el API y el consumidor deben compartir una sola interfaz técnica: URI, métodos, media type, etc. Esta restricción se sub-divide en 4:

API REST - Características

- **6. Uniform interface:** Esta restricción se divide en cuatro subprincipios:

- **1. Identificación de recursos**

  Cada recurso se identifica de manera única a través de una URI (Uniform Resource Identifier). Esto significa que cualquier recurso puede ser accesible a través de un URL único.

Ejemplo:
•URI para un recurso de usuario: **https://api.ejemplo.com/usuarios/123**
Aquí, **https://api.ejemplo.com/usuarios/123** identifica de manera única al usuario con ID 123.

API REST - Características

- **6. Uniform interface:** Esta restricción se divide en cuatro subprincipios:

2. Manipulación de recursos a través de representaciones

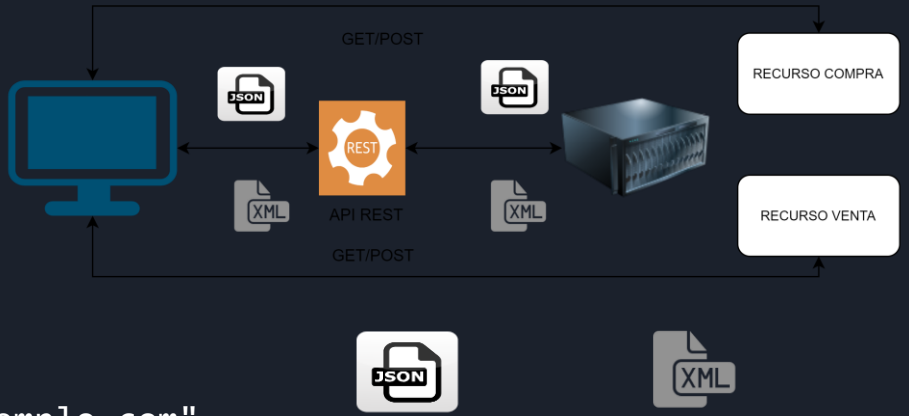
El cliente interactúa con los recursos utilizando representaciones de los mismos, que suelen ser documentos JSON o XML. Estas representaciones se transfieren entre el cliente y el servidor.

Ejemplo:

- **Solicitud (GET):**

```
GET /usuarios/123
Host: api.ejemplo.com
Accept: application/json
```

```
{
    "id": 123,
    "nombre": "Juan Pérez",
    "email": "juan.perez@ejemplo.com"
}
```



API REST - Características

- **6. Uniform interface:** Esta restricción se divide en cuatro subprincipios:

- **3. Mensajes auto-descriptivos**

Cada mensaje en la comunicación entre el cliente y el servidor debe contener suficiente información para describir cómo procesar el mensaje. Esto incluye el uso de encabezados HTTP que describen el tipo de contenido y cómo manejarlo. Ejemplo**:**

Solicitud (POST):

```
POST /usuarios
Host: api.ejemplo.com
Content-Type: application/json

{
    "nombre": "Ana Gómez",
    "email": "ana.gomez@ejemplo.com"
}
```

Respuesta (201 Created):

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /usuarios/124

{
    "id": 124,
    "nombre": "Ana Gómez",
    "email": "ana.gomez@ejemplo.com"
}
```

API REST - Características

- **6. Uniform interface:** Esta restricción se divide en cuatro subprincipios:

4. Hipermedios como el motor del estado de la aplicación (HATEOAS)
El cliente debe ser capaz de descubrir todas las acciones disponibles dinámicamente a través de hipermedios proporcionados por el servidor en las respuestas. Esto significa que las respuestas incluirán enlaces a otras acciones posibles.

Ejemplo:

Solicitud (GET):

```
GET /usuarios/123
Host: api.ejemplo.com
```

Respuesta (200 OK):

```
{
  "id": 123,
  "nombre": "Juan Pérez",
  "email": "juan.perez@ejemplo.com",
  "links": [
    {
      "rel": "self",
      "href": "/usuarios/123"
    },
    {
      "rel": "actualizar",
      "href": "/usuarios/123",
      "method": "PUT"
    },
    {
      "rel": "borrar",
      "href": "/usuarios/123",
      "method": "DELETE"
    }
  ]
}
```

RESTful

- Si cumple con todas las restricciones decimos que es una API RESTful

FIN

GRACIAS!!!!