

PROYECTO DE SOFTWARE

Cursada 2022

TEMARIO

- Api Rest.

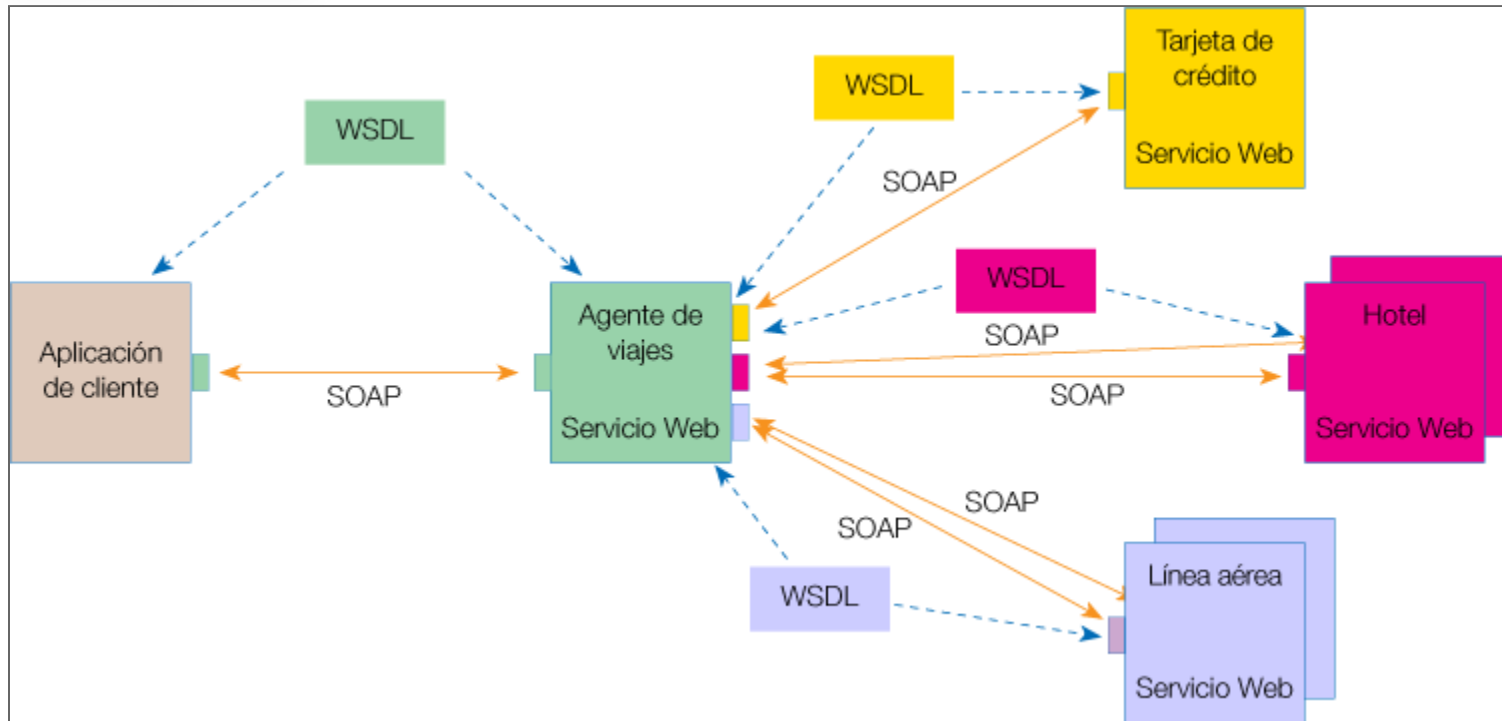
API REST

- ¿Qué es una API REST?

SERVICIOS WEB

- Un **servicio web** es una tecnología que utiliza un conjunto de **protocolos** y **estándares** para intercambiar datos entre aplicaciones.
- Algo importante: lograr **interoperabilidad**.
- Uso de estándares abiertos
 - XML-RPC, JSON-RPC
 - WSDL: Web Services Description Language
 - SOAP: Simple Object Access Protocol
 - ¿Rest?

SERVICIOS WEB



- Ejemplo típico con WSDL y SOAP

¿QUÉ ES REST?

- **REpresentational State Transfer**: arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP.
- REST nos permite crear servicios y aplicaciones que pueden ser usadas por **cualquier dispositivo o cliente que entienda HTTP**, por lo que es más simple y convencional que otras alternativas que se han usado en los últimos diez años como SOAP y XML-RPC.
- REST fue definido por Roy Fielding (coautor de la especificación HTTP) en el año 2000.

REST

- Los sistemas que siguen los principios REST se los denomina también RESTful.
- Se basa en **HTTP** para intercambiar información.
- **SIN estado**.
- Se piensa en los **recursos** como una entidad que puede ser accedido públicamente.
- Cada objeto tiene su propia URL y puede ser fácilmente cacheado, copiado y guardado como marcador.

RECORDEMOS QUE

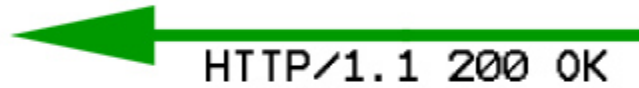
- Una petición HTTP consta de:
 - Una **URL** y un **método** de acceso (GET, POST, PUT,...).
 - **Cabeceras**. Meta-información de la petición.
 - **Cuerpo del mensaje** (opcional).

HTTP MODEL

Client



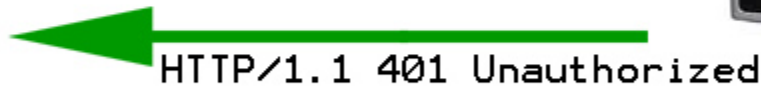
GET / HTTP/1.1



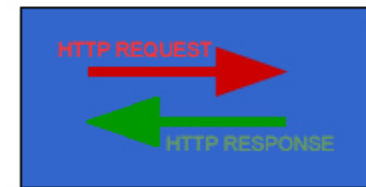
Server



POST /login HTTP/1.1



Client



MÉTODOS HTTP

- **GET**: usado para solicitar un recurso al servidor.
- **PUT**: usado para modificar un recurso existente en el servidor.
- **POST**: usado para crear un nuevo recurso en el servidor.
- **DELETE**: usado para eliminar un recurso en el servidor.

¿POST = GET + PUT + DELETE?

- Por lo general, las peticiones de tipo **PUT** y **DELETE** son realizadas a través de peticiones **POST**.
- La petición **POST** se utiliza tanto para crear, borrar o actualizar un recurso.
- Pero hay una diferencia: **POST NO es idempotente**.

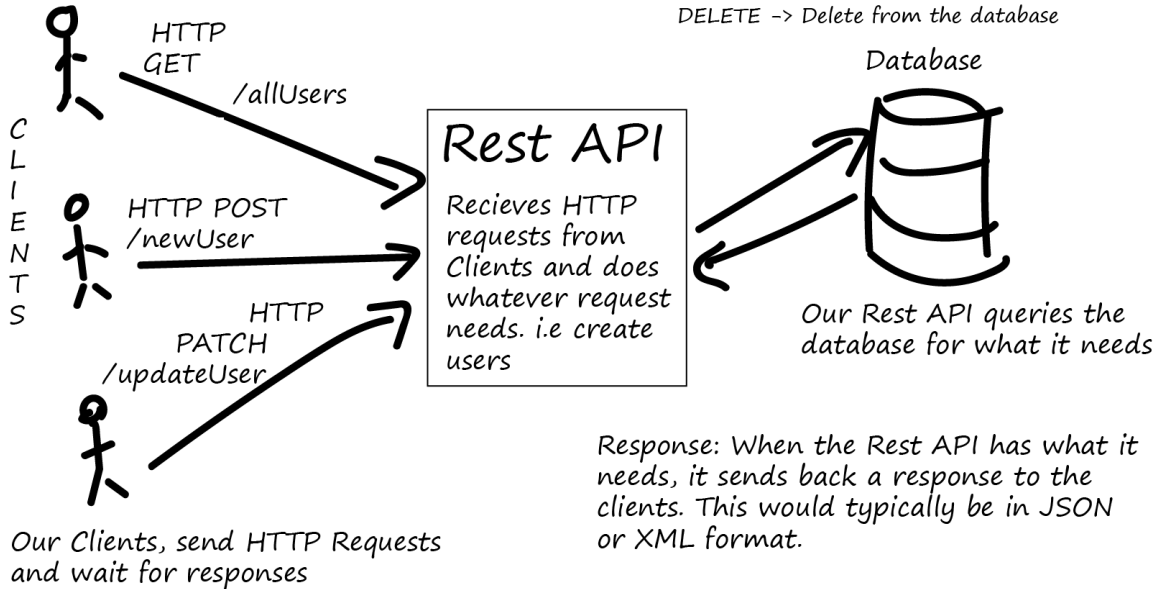
MÉTODOS HTTP

HTTP Method	Idempotent	Safe
OPTIONS	yes	yes
GET	yes	yes
HEAD	yes	yes
PUT	yes	no
POST	no	no
DELETE	yes	no
PATCH	no	no

- Los métodos HTTP **Seguros** son aquellos que no modifican los recursos.
- Los métodos HTTP **Idempotentes** son aquellos que pueden ser llamados múltiples veces y generarán el mismo resultado.
- Más info ver: [rfc7231](#) y [rfc5789](#)

Rest API Basics

Typical HTTP Verbs:
GET -> Read from Database
PUT -> Update/Replace row in Database
PATCH -> Update/Modify row in Database
POST -> Create a new record in the database
DELETE -> Delete from the database



- Ref: <https://medium.com/@audira98/why-should-we-choose-rest-client-server-model-to-develop-web-apps-c3bb2451b13a>

ACEDIENDO A LOS RECURSOS

- La implementación del recurso decide qué información es visible o no desde el exterior, y qué representaciones de dicho recurso se soportan.
- Podríamos pensar en:
 - HTML
 - XML
 - JSON
- Ejemplo: ¿consultamos los feriados de 2022?

ACCEDIENDO A LOS RECURSOS

- Probemos con **curl**:

```
curl -X GET  
https://api.mercadolibre.com/categories/MLA5725
```

- Otros ejemplos:
 - API REST de Mercado Libre:
<https://developers.mercadolibre.com.ar/>
 - Google Translate (es necesario API_KEY):
<https://developers.google.com/apis-explorer/#p/translate/v2/>
 - El clima en OpenWeatherMap (es necesario API_KEY):
[clima en La Plata](#)
 - Algunas API rest públicas: [en desarrolloweb.com](https://en.desarrolloweb.com)

VENTAJAS / DESVENTAJAS

- Separación cliente/servidor.
- Simplicidad.
- Seguridad.
- Uso de estándares.
- Escalabilidad.
- Cambio de esquema: usando REST podemos tener varios servidores donde unos no saben que los otros existen.

+Info: <https://desarrolloweb.com/articulos/ventajas-inconvenientes-apirest-desarrollo.html>

GENERANDO API REST

- A mano.... o,
- Muchos frameworks que facilitan el desarrollo:
 - Django: <https://www.django-rest-framework.org/tutorial/quickstart/>
 - Flask: <https://flask-restful.readthedocs.io/en/latest/quickstart.html>

REFERENCIAS REST

- <http://martinfowler.com/articles/richardsonMaturityModel.html>
- <http://restcookbook.com/Miscellaneous/richardsonmaturitymodel/>
- <http://www.restapitutorial.com/lessons/whatisrest.html>
- <http://asiermarques.com/2013/conceptos-sobre-apis-rest/>
- <http://rest.elkstein.org/>
- <http://restfulwebapis.org/rws.html>
- <https://restfulapi.net/>
- <https://www.paradigmadigital.com/dev/introduccion-django-rest-framework/>
- <https://flask-restful.readthedocs.io/en/latest/>

FIN