PIEMONTE
FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020
fondo sociale europeo
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

RESTful API

Pier Paolo Pittavino

in collaborazione con:
ISTITUTI TECNICI SUPERIORI
Ministero dell'Istruzione
UNIONE EUROPEA
REGIONE PIEMONTE
per una crescita intelligente, sostenibile ed inclusiva
www.regione.piemonte.it/europa2020
INIZIATIVA CO-FINANZIATA CON FSE

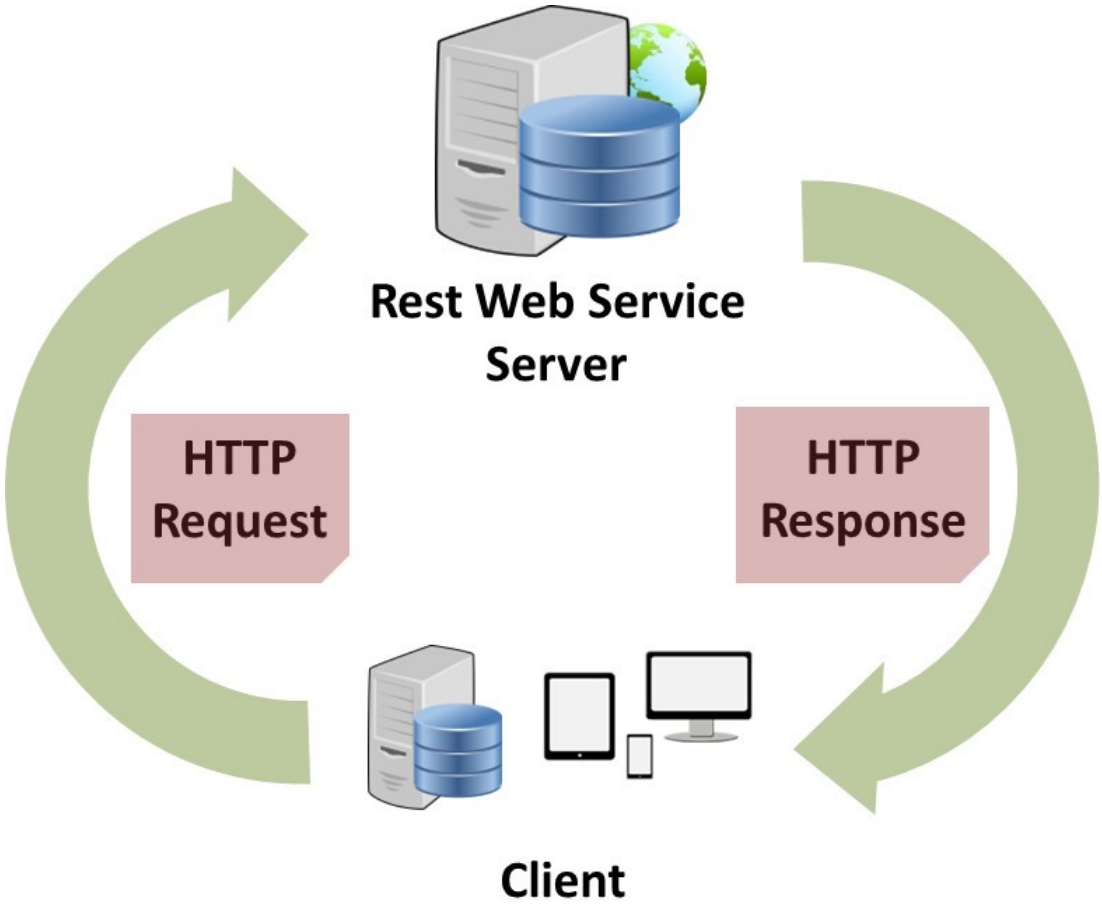
Definizione 0.1: RESTful API

Interfaccia di programmazione delle applicazioni (API o API web) conforme ai vincoli dello stile architetturale REST, che consente l'interazione con servizi web RESTful. Il termine REST, coniato dall'informatico Roy Fielding, è l'acronimo di REpresentational State Transfer.

A cosa servono?

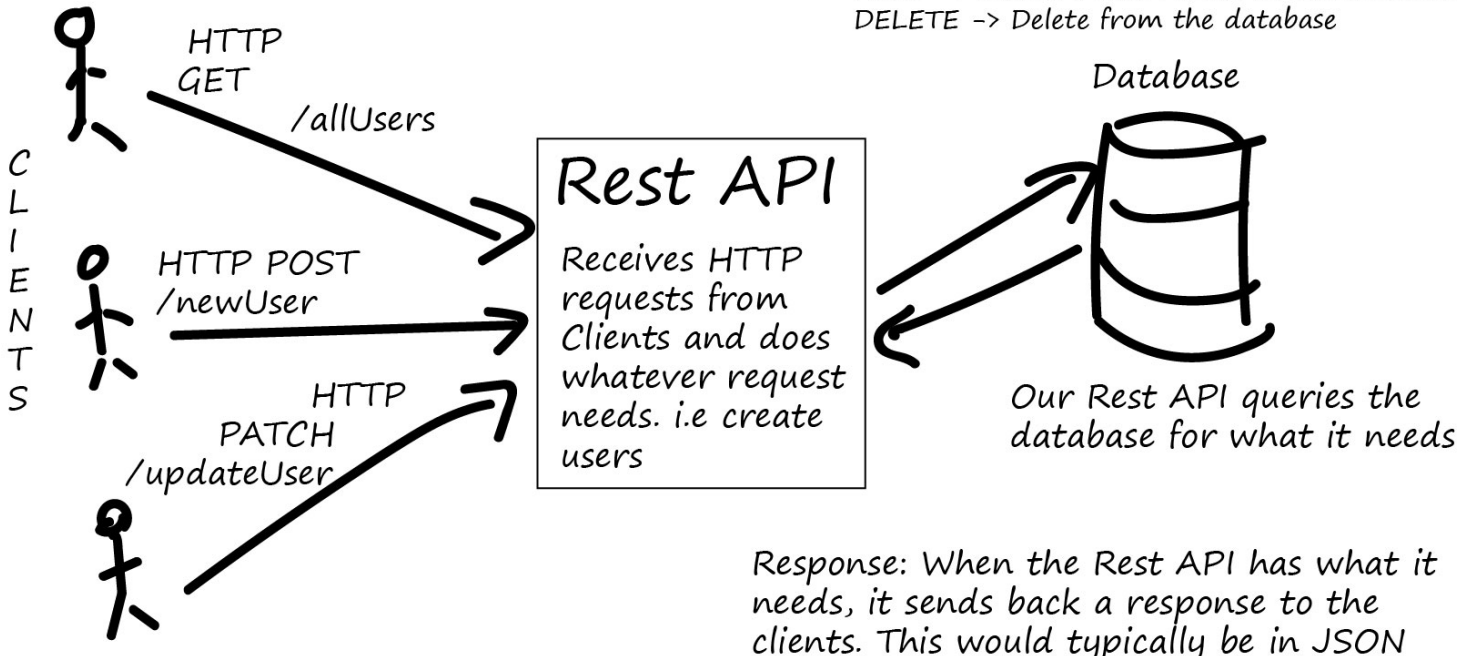
L'API funge quindi da elemento di intermediazione tra gli utenti o i clienti e le risorse o servizi web che questi intendono ottenere. È anche un mezzo con il quale un'organizzazione può condividere risorse e informazioni assicurando al contempo sicurezza, controllo e autenticazione, poiché stabilisce i criteri di accesso.

L'utilizzo dell'API inoltre non impone all'utente di conoscere le specifiche con cui le risorse vengono recuperate o la loro provenienza


Rest Web Service
Server
HTTP
Request
HTTP
Response
Client

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



Our Rest API queries the database for what it needs

Our Clients, send HTTP Requests and wait for responses

Response: When the Rest API has what it needs, it sends back a response to the clients. This would typically be in JSON or XML format.

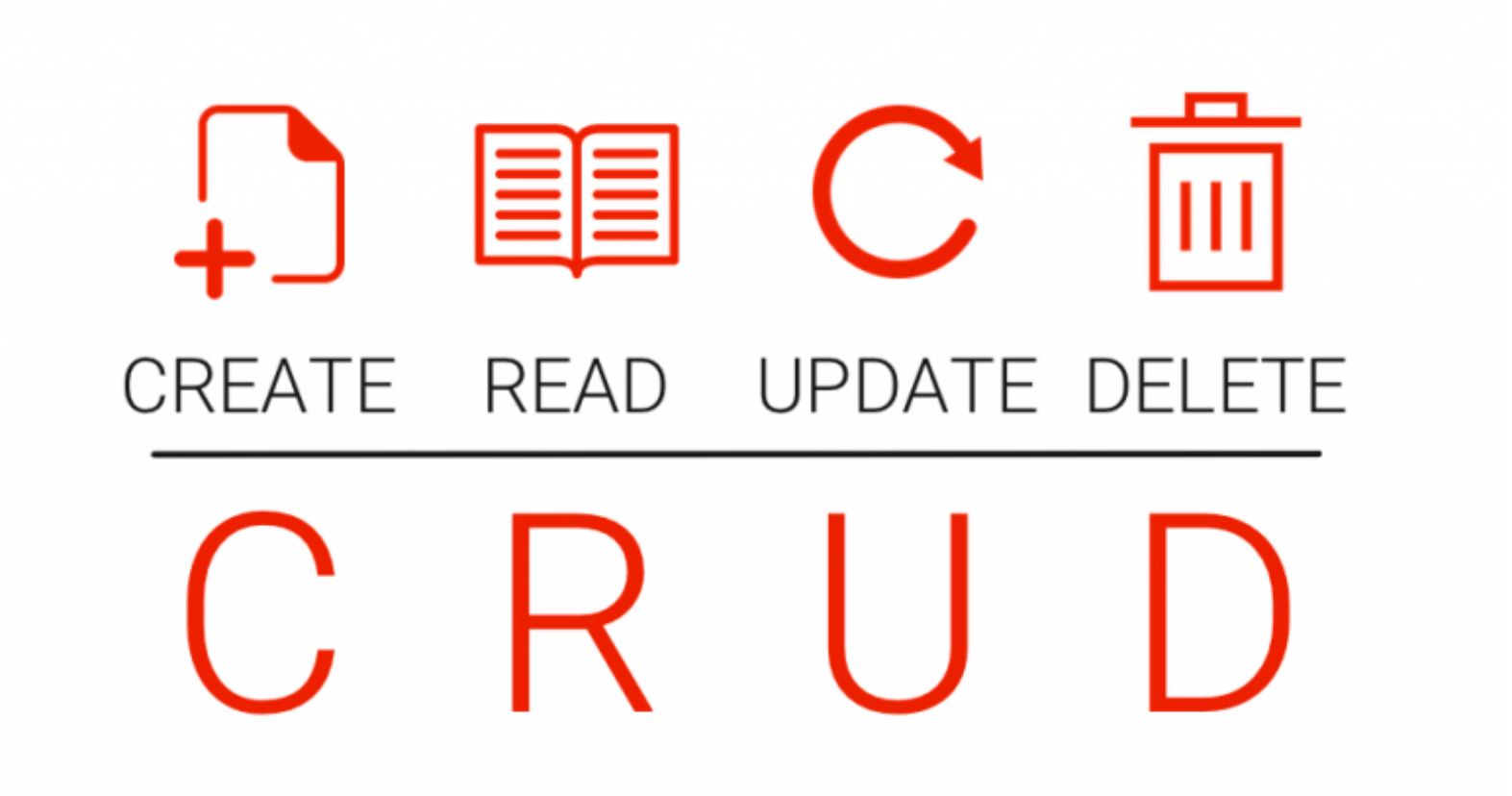
Alcuni aspetti da chiarire

- **REST** è un insieme di vincoli architetturali, non un protocollo né uno standard
- L'informazione, o rappresentazione, viene consegnata in uno dei diversi formati tramite HTTP (*JSON*,*XML*,*etc...*)
- intestazioni e parametri sono importanti nei metodi HTTP di una richiesta HTTP API RESTful

Criteri RESTful

- **Client–server**: richieste gestite tramite HTTP
- **Stateless**: ogni richiesta è distinta e non connessa
- **Uniform interface**: Un'interfaccia uniforme per i componenti, in modo che le informazioni vengano trasferite in una forma standard
- **Cacheable**: Dati memorizzabili nella cache che ottimizzano le interazioni client-server
- **Layered system**:Un sistema su più livelli che si occupa di recuperare le informazioni richieste in gerarchie, invisibile al client
- **Code on demand**:(opzionale) I server possono temporaneamente estendere o personalizzare le funzionalità del client trasferendo del codice eseguibile.

Le operazioni CRUD



CRUD in SQL

SQL	CRUD	Descrizione
INSERT	Create	Crea una nuova risorsa
SELECT	Read	Ottiene una risorsa esistente
UPDATE	Update	Aggiorna una risorsa o ne modifica lo stato
DELETE	Delete	Elimina una risorsa

CRUD con i metodi HTTP

Come mapparli		
HTTP	CRUD	Descrizione
POST	Create	Crea una nuova risorsa
GET	Read	Ottiene una risorsa esistente
PUT	Update	Aggiorna una risorsa o ne modifica lo stato
DELETE	Delete	Elimina una risorsa

Esempio Sbagliato!!

Esempio 0.1: Richiesta non RESTFul

http://www.miosito.it/aggiungiarticoli?name=nerf

NON è conforme ai principi REST!!
il metodo GET serve per accedere alla rappresentazione di una risorsa e non per crearne una nuova

Messaggi di richiesta HTTP

Il messaggio di richiesta è composto di quattro parti:

- riga di richiesta (request line)
- sezione header (informazioni aggiuntive)
- riga vuota (CRLF: i 2 caratteri carriage return e line feed)
- body (corpo del messaggio)

Riga di richiesta

La riga di richiesta è composta da metodo, URI *uniform resource identifier* e versione del protocollo

I metodi sono

- **GET**
- **POST**
- HEAD
- **PUT**
- **DELETE**
- PATCH
- TRACE
- OPTIONS
- CONNECT

Gli header della richiesta

Gli header di richiesta più comuni sono:

- **Host**: nome del server a cui si riferisce l'URL. È obbligatorio nelle richieste conformi HTTP/1.1 perché permette l'uso dei virtual host basati sui nomi
- **User-Agent**: identificazione del tipo di client: tipo browser, produttore, versione...
- **Cookie**: utilizzati dalle applicazioni web per archiviare e recuperare informazioni a lungo termine sul lato client. Spesso usati per memorizzare un token di autenticazione o per tracciare le attività dell'utente

Messaggio di risposta

Il messaggio di risposta è di tipo testuale ed è composto da quattro parti:

- riga di stato (*status-line*)
- sezione header
- riga vuota (CRLF: i 2 caratteri carriage return e line feed)
- body (contenuto della risposta)

Riga di stato 1/2

La riga di stato riporta un codice a tre cifre catalogato nel seguente modo:

- **1xx**: Informational (messaggi informativi)
- **2xx**: Successful (la richiesta è stata soddisfatta)
- **3xx**: Redirection (non c'è risposta immediata, ma la richiesta è sensata e viene detto come ottenere la risposta)
- **4xx**: Client error (la richiesta non può essere soddisfatta perché sbagliata)
- **5xx**: Server error (la richiesta non può essere soddisfatta per un problema interno del server)

Riga di stato 2/2

I codici di risposta più comuni sono:

- **200 OK**: Il server ha fornito correttamente il contenuto nella sezione body
- **301 Moved Permanently**: La risorsa che abbiamo richiesto non è raggiungibile perché è stata spostata in modo permanente
- **302 Found**: La risorsa è raggiungibile con un altro URI indicato nel header Location
- **400 Bad Request**: La risorsa richiesta non è comprensibile al server
- **404 Not Found**: La risorsa richiesta non è stata trovata e non se ne conosce l'ubicazione
- **500 Internal Server Error**: Il server non è in grado di rispondere alla richiesta per un suo problema interno
- **502 Bad Gateway**:Il server web che agisce come reverse proxy non ha ottenuto una risposta valida dal server di upstream
- **505 HTTP Version Not Supported**:La versione di http non è supportata

Gli header della risposta

Gli header della risposta più comuni sono:

- **Server**: indica il tipo e la versione del server. Può essere visto come l'equivalente dell'header di richiesta User-Agent
- **Content-Type**: indica il tipo di contenuto restituito *Media type (RFC 1521)* ad es.:
  - **text/html** Documento HTML
  - **text/plain** Documento di testo non formattato
  - **text/html** Documento XML
  - **text/plain** Immagine di formato JPEG

Esempio di una richiesta

Esempio 0.2: Richiesta HTTP 1.1

```
GET /wiki/Pagina_principale HTTP/1.1
Host: it.wikipedia.org
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.2;
Linux) (KHTML, like Gecko)
Accept: text/html, image/jpeg, image/png, text/*,
image/*, */*
Accept-Charset: iso-8859-1, utf-8;q=0.5, *;q=0.5
Accept-Language: it
Connection: Keep-Alive
```

Gli header della risposta

Gli header della risposta più comuni sono:

- **Server**: indica il tipo e la versione del server. Può essere visto come l'equivalente dell'header di richiesta User-Agent
- **Content-Type**: indica il tipo di contenuto restituito *Media type (RFC 1521)* ad es.:
  - **text/html** Documento HTML
  - **text/plain** Documento di testo non formattato
  - **text/html** Documento XML
  - **text/plain** Immagine di formato JPEG

Risorse autodescrittive

Le risorse di per sè sono concettualmente separate dalle rappresentazioni restituite al client

- I principi REST non pongono nessun vincolo sulle modalità di **rappresentazione** di una risorsa

Accorgimenti

possiamo utilizzare il formato che preferiamo, meglio utilizzare standard, possibilmente multipli

Risorse autodescrittive

Le risorse di per sè sono concettualmente separate dalle rappresentazioni restituite al client

- I principi REST non pongono nessun vincolo sulle modalità di **rappresentazione** di una risorsa
- Il tipo di rappresentazione è indicato nella stessa risposta HTTP tramite un tipo *MIME*

Accorgimenti

il client può richiedere uno specifico formato con l'attributo *Accept* nell'header

Collegamenti tra risorse

Vincolo: le risorse siano tra loro messe in relazione tramite link

Esempio 0.3: Esempio di risposta con link

```
<ordine>
    <numero>12345678</numero>
    <data>04/04/2022</data>
    <cliente rif="http://www.miosito.it/clienti/1234" />
    <articoli>
        <articolo rif="http://www.miosito.it/prodotti/98765" />
        <articolo rif="http://www.miosito.it/prodotti/43210" />
    </articoli>
</ordine>
```

Stateless

- nessun contesto client viene memorizzato sul server tra le richieste

Stateless

- nessun contesto client viene memorizzato sul server tra le richieste
- Ciascuna richiesta dai vari client contiene tutte le informazioni necessarie per richiedere il servizio

Stateless

- nessun contesto client viene memorizzato sul server tra le richieste
- Ciascuna richiesta dai vari client contiene tutte le informazioni necessarie per richiedere il servizio
- ciascuna richiesta non ha alcuna relazione con le richieste precedenti e successive

Stateless

- nessun contesto client viene memorizzato sul server tra le richieste
- Ciascuna richiesta dai vari client contiene tutte le informazioni necessarie per richiedere il servizio
- ciascuna richiesta non ha alcuna relazione con le richieste precedenti e successive
- La responsabilità della gestione dello stato rientra nei compiti del client

ITS
TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

GRAZIE!

POR Piemonte
FSE 2014-2020