# HTTP

# Lo usiamo tutti i giorni.

Approfondiamo!

*Salvatore La Spata (salvatore.la.spata@gmail.com)*

# Agenda

1. Cosa è l'HTTP
2. URL
3. Metodi
4. Status Code
5. Request e Response?
6. REST vs oData e perchè no GraphQL
7. Dalla teoria alla SEGW
8. Live Demo

Iniziamo...
**Cosa è** e **come funziona l'HTTP**?!

# Quale **HTTP** sei?

Anche se il protocollo **HTTP** ha più di 30 anni, ad oggi ne esistono solo tre versioni attuali.

**HTTP/1.1**

L'ultima versione del protocollo iniziale. Di gran lunga la più famosa e la più utilizzata.

**HTTP/2**

Sviluppato sulla base di SPDY per eliminare alcuni problemi di HTTP.

**HTTP/3**

Sviluppato sulla base di QUIC per risolvere i problemi TCP.

# Quale **HTTP** supporta **SAP**

**HTTP/1.1**

**HTTP/2**

Iniziamo...
**Cosa è** e **come funziona l'HTTP**?!

# Cosa è l'HTTP?

"Acronimo di `HyperText Transfer Protocol` è un protocollo a livello applicativo usato come principale sistema per la trasmissione d'informazioni sul web ovvero in un'architettura tipica client-server.

Le specifiche del protocollo sono gestite dal World Wide Web Consortium (W3C)"

Wikipedia (Hypertext_Transfer_Protocol)

# Attori dell'**HTTP**

## Client

E' il componente che inizia una richiesta, il web browser nel caso del web, ma potrebbe anche essere una app che comunica con un server API.

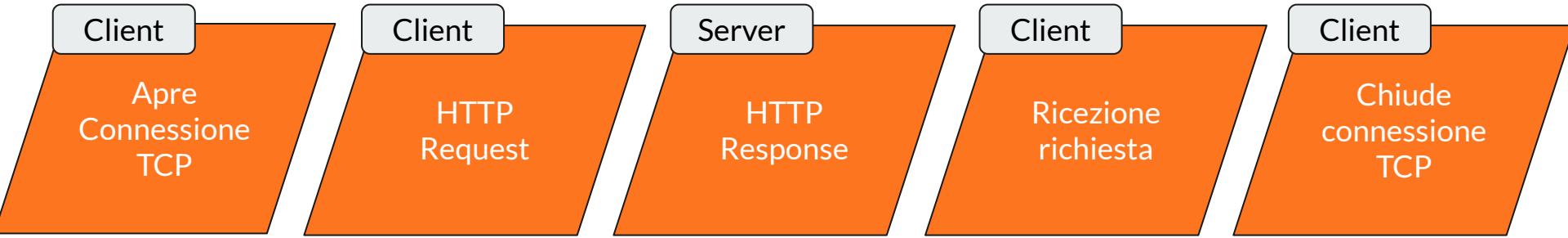
## Server

Un server non è necessariamente una singola macchina; riceve le richieste e risponde con il contenuto richiesto, nel fare ciò potrebbero entrare in gioco componenti quali load balancer, cache, database, etc…

## Proxy

Nel mondo reale, tra il client e il server ci possono essere diversi computer che si "passano" la richiesta e la risposta per farla arrivare alla macchina giusta;

# Workflow


Client
Apre Connessione TCP
Client
HTTP Request
Server
HTTP Response
Client
Ricezione richiesta
Client
Chiude connessione TCP

Ma...
**L'HTTP** è diverso dall'**HTTPS**?!

L'unica differenza è la **s** finale 😎. **L'HTTPS è un'estensione** del protocollo di trasmissione HTTP ma che ne definisce criteri di sicurezza (Secure Socket Layer).

#crittografia #sicurezza

# **URL** - Uniform Resource Locator

https://**<DOMAIN>**/sap/opu/odata/sap/**<ODATA_PROJECT>**/**<ENTITY_SET>**?**<QUERY_OPTIONS>**#**<HASH>**

| DOMAIN | ODATA_PROJECT | ENTITY_SET | QUERY_OPTIONS | HASH |
|---|---|---|---|---|
| Host + Port<br>IP + Port | Nome del Progetto oData | Nome del Servizio oData | Query applicati a una risorsa | Identificatore frammento dell'URL |

# Method





Ecco la lista completa: **# CONNECT; # DELETE; # GET; # HEAD; # OPTIONS; # POST; # PUT; # TRACE;**
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

# Status Code


(100 – 199)
Informational


(200 – 299)
Successful


(300 – 399)
Redirection


(400 – 499)
Client Error


(500 – 599)
Server Error

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

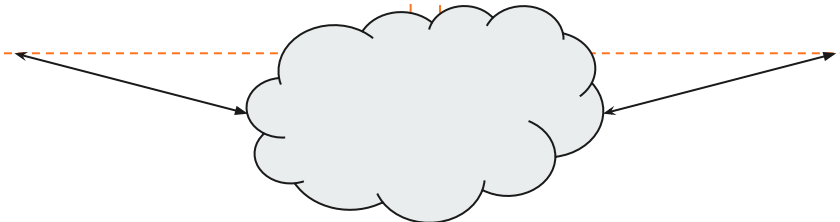
# Request e Response

Il messaggio di **richiesta** è composto da 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).

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).



# Gli header

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

`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

`Custom`:

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. La codifica di tali tipi (detti Media type) è registrata presso lo IANA (Internet Assigned Number Authority); essi sono detti tipi MIME (Multimedia Internet Mail Extensions), la cui codifica è descritta nel documento RFC 1521.

`Custom`:

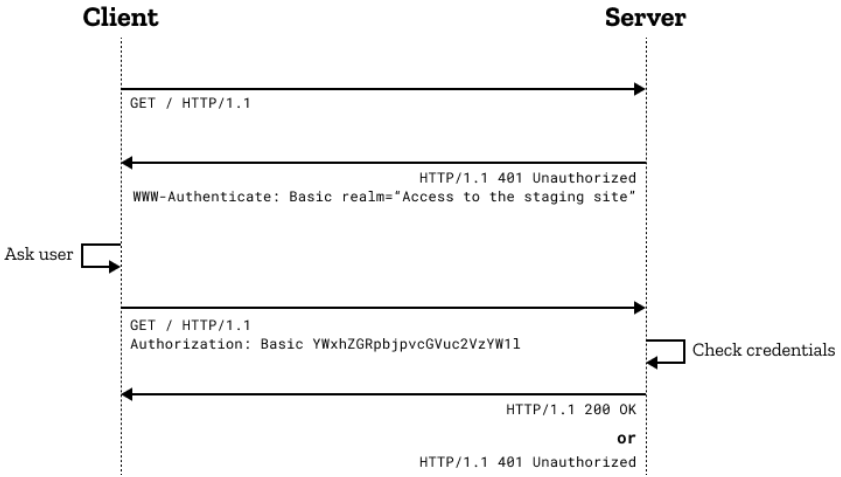
# Authorization

`Basic`

[RFC 7617] Credenziali codificate in base64.
Header:
**Authorization:** Basic <user>:<password> codificato in base64.

`Bearer`

[RFC 6750] Token bearer per accedere a risorse protette da OAuth 2.0 o da API Key.



# XML vs JSON

Definiti dalle proprietà Content-Type e Accept nell'header della richiesta e nel Content-Type della risposta oppure ancora tramite la query option $format. Sono rappresentazioni di dati utilizzate nello scambio di dati tra applicazioni, JSON è l'opzione più recente, più flessibile e più popolare.

## XML

XML è un linguaggio di markup che fornisce regole per definire qualsiasi dato.

Utilizza i tag per distinguere tra attributi di dati e dati effettivi.

## JSON

Formato di interscambio di dati aperto e leggibile sia da persone che da macchine.

JSON è indipendente da qualsiasi linguaggio di programmazione ed è un output API comune in un'ampia varietà di applicazioni.

# RESTful vs oData vs Graphql?

## Chi sarà il cavallo vincente

**Spoiler:** l'ha scelto **SAP** per noi 😑😑 - oData

# 😎 oData V2 (Spoiler: **V4** alla prossima puntata)

# Cosa è il “protocollo” oData?

Acronimo di `Open Data Protocol` è uno standard OASIS approvato da ISO/IEC che definisce un insieme di best practice per la creazione e il consumo di `API RESTful`.
Una `API RESTful` è un'interfaccia di programmazione delle applicazioni (API o API web) conforme ai vincoli dello stile architetturale `REST` (`RE`presentational `S`tate `T`ransfer)

Affinché un'API sia considerata RESTful:

- Comunicazione tramite **HTTP**
- Una comunicazione **client-server stateless**
- **Dati memorizzabili nella cache** che ottimizzano le interazioni client-server
- **Un'interfaccia uniforme** per i componenti (rappresentata dal **$metadata**)

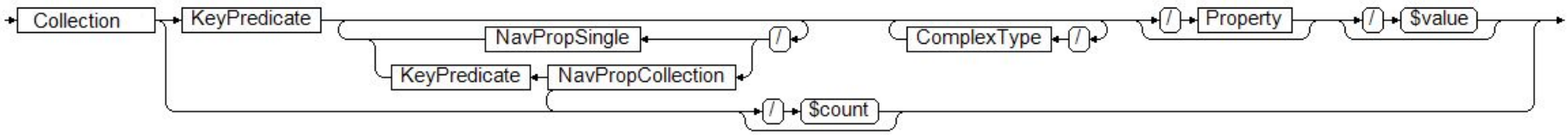
https://www.odata.org/
https://redhat.com/it/topics/api/what-is-a-rest-api

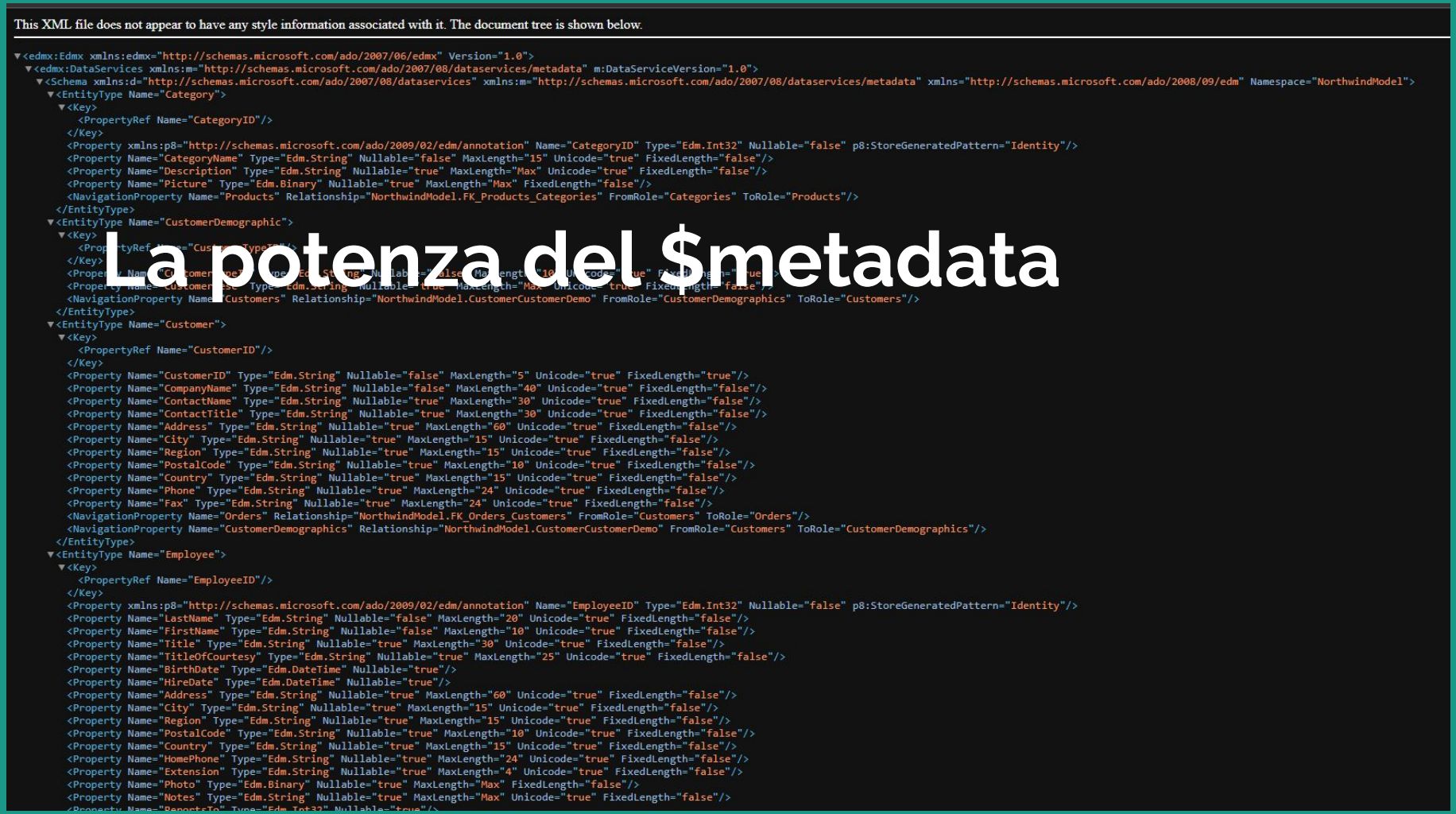
# Query Options

- $filter
- $orderby
- $top & $skip
- $count
- $expand
- $select
- $search

```
https://services.odata.org/OData/OData.svc
                  service root URL

https://services.odata.org/OData/OData.svc/Category(1)/Products?$top=2&$orderby-name
                  service root URL          resource path       query options
```






# La potenza del $metadata

# $Metadata

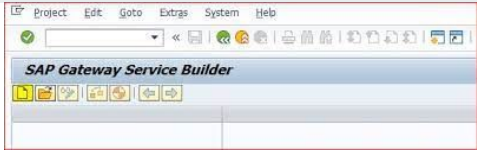
Rappresenta la firma del progetto oData.

Nel primo schema sono censiti i tipi
Sono descritti le Entity Type con le relative chiavi e property (definendone anche i tipi), le Association che rappresentano le relazioni tra le entity

Nel secondo schema sono censite le entità da richiamare (implementazioni)

```
<Schema>
	<EntityType>
		<Key>
			<PropertyRef  />
			...
		</Key>
		<Property />
		...
	</EntityType>
	...
	<Association>...</Association >
	...
</Schema>
<Schema>
	<EntityContainer>
		<EntitySet />
		...
		<AssociationSet />
		...
	</EntityContainer>
</Schema>
```

# Tutto bello ma la **SEGW**




# SAP Gateway Service Builder

SEGW

La SEGW permette di creare e definire servizi oData tramite la SAP GUI. Mette a disposizione dei wizard per generare i servizi/associazioni/implementazioni. Ovviamente è possibile ridefinire il comportamento standard

Struttura del progetto oData

| | |
|---|---|
| Progetto | Generalmente inizia per Z* |
| Data Model | Tipi e Associazioni |
| Entity Types | Censimento delle proprietà (constraints) |
| Associations | Aggregazioni |
| Entity Sets | Definizione interfaccia di comunicazione |
| Service Implementation | Definizione dell'implementazione |
| Runtime Artifacts | Oggetti tecnici ABAP (generati e non) |

# SAP Gateway Service Builder

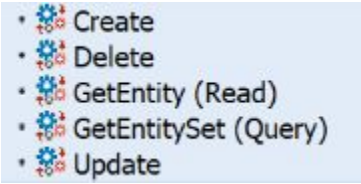
## Data Model

In questa sezione vengono censite le entity e le rispettive proprietà.
Sono presenti:

- **Entity Type**
  - **Properties ***
  - **Navigation Properties ****
- **Associations ****
- **Entity Sets**

* Le property sono caratterizzati da un tipo Edm (docs) con le relative constraints, campo ABAP, tipo ABAP, l'identificazione chiave, etc..

** Le navigation property rappresentano le aggregazioni ($expand). Vengono definite tramite le Associations




# SAP Gateway Service Builder

## Service Implementation

In questa sezione vengono censite implementazioni e quindi la logica di business richiamata dall'entity in relazione al metodo:

- **Create** — Utilizzata per le creazioni
  **POST** /<ODATA_PROJECT>/**<ENTITY_SET>**
- **Delete** — Utilizzata per le cancellazioni
  **DELETE** /<ODATA_PROJECT>/**<ENTITY_SET>(<KEYS>)**
- **GetEntity** (Read) — Restituisce una sola entity rispetto alle chiavi trasmesse
  **GET** /<ODATA_PROJECT>/<ENTITY_SET>(<KEYS>)
- **GetEntitySet** (Query) — Restituisce una collezione (array) di entity rispetto ai filtri definiti
  **GET** /<ODATA_PROJECT>/**<ENTITY_SET>?<QUERY_OPTIONS>**
- **Update** — Utilizzata per gli aggiornamenti delle singole entity
  **PUT** /<ODATA_PROJECT>/**<ENTITY_SET>(<KEYS>)**

# I metodi in dell'**HTTP** in **SAP oData**

| HTTP Method | SAP Service Implementation |
|---|---|
| **GET** | **Query** - Richiede un array di entità |
| **GET** | **Read** - Richiede una singola entità |
| **POST** | **Create** - Crea un'entità |
| **PUT** | **Update** - Aggiorna un'entità |
| **DELETE** | **Delete** - Cancella l'entità |
| **GET** - Aggregata (con la query option $expand o richiedendo la specifica risorsa) | GET_EXPANDED_ENTITY - struttura con i dati aggregati<br>GET_EXPANDED_ENTITYSET - tabella con i dati aggregati |
| **POST** - Aggregata | CREATE_DEEP_ENTITY |

# SAP Gateway Service Builder

## Runtime Artifact

Le classi generate ed estensibili:Informazioni sul progetto (_MDL), definizioni (_SRV), tipi (_MPC*) e logica di business (_DPC*)

Project_Name_**ANNO_MDL**
Annotazioni

Project_Name_**MDL**
Nome Tecnico Modello

Project_Name_**SRV**
Nome Tecnico Servizio

**Z_CL_**Project_Name_**RDS_DPC**
**[GENERATED]** Data Provider Class

**Z_CL_**Project_Name_**RDS_DPC_EXT**
Classe di implementazione

**Z_CL_**Project_Name_**RDS_MPC**
**[GENERATED]** Model Provider Class

**Z_CL_**Project_Name_**RDS_MPC_EXT**
Classe di implementazione

# SAP Gateway Service Builder

## Runtime Artifact
MPC & MPC_EXT

In questa classe sono censiti tutti i tipi fruiti dal servizio oData. E' comunque possibile inserirni di nuovi nella classe MPC_EXT e, in alcuni casi, è richiesto di modificare un specifico tipo a runtime utilizzando il metodo DEFINE

# SAP Gateway Service Builder

## 1. Runtime Artifact
DPC & DPC_EXT

Ogni entity è caratterizzata da tutte le operazioni CRUD.

Il comportamento standard può essere esteso ridefinendo il metodo specifico alla richiesta.

In particolare:

| | | |
|---|---|---|
| - *CREATE_ENTITY | **POST** | /<ODATA_PROJECT>/**<ENTITY_SET>** |
| - *UPDATE_ENTITY | **PUT** | /<ODATA_PROJECT>/**<ENTITY_SET>(<KEYS>)** |
| - *DELETE_ENTITY | **DELETE** | /<ODATA_PROJECT>/**<ENTITY_SET>(<KEYS>)** |
| - *GET_ENTITY | **GET** | /<ODATA_PROJECT>/**<ENTITY_SET>?<QUERY_OPTIONS>** |
| - *GET_ENTITYSET | **GET** | /<ODATA_PROJECT>/**<ENTITY_SET>(<KEYS>)** |

La keys è composta come **/<ENTITY_SET>(<chiave1>='<valore1>',<chiave2>='<valore2>')**
Se la chiave è 1 sola è possibile fare **/<ENTITY_SET>('<valore_chiave>')**
**N.B.** I tipi sono rilevanti nelle chiavi (es: **datetime'2014-03-11T14:49:52'**)

# SAP Gateway Service Builder

## 2. Runtime Artifact
DPC & DPC_EXT

**DEEP**
- /IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_EXPANDED_ENTITYSET
- /IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_EXPANDED_ENTITY

**FILE**
- /IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_STREAM
- /IWBEP/IF_MGW_APPL_SRV_RUNTIME~CREATE_STREAM

**BATCH**
- /IWBEP/IF_MGW_APPL_SRV_RUNTIME~CHANGESET_BEGIN
- /IWBEP/IF_MGW_APPL_SRV_RUNTIME~CHANGESET_PROCESS
- /IWBEP/IF_MGW_APPL_SRV_RUNTIME~CHANGESET_END

**Come gestiamo il ritorno?**
**Successo** tutto bello
Qualcuno le chiama eccezioni…

# TCODE & Utils

**SEGW**
**LPD_CUST**
**PFCG**
/n/UI2/**SEMOBJ**
/n/IWFND/**ERROR_LOG**
/n/IWNFD/**CACHE_CLEANUP**
/n/IWFND/**MAINT_SERVICE**
/n/IWFND/**GW_CLIENT**

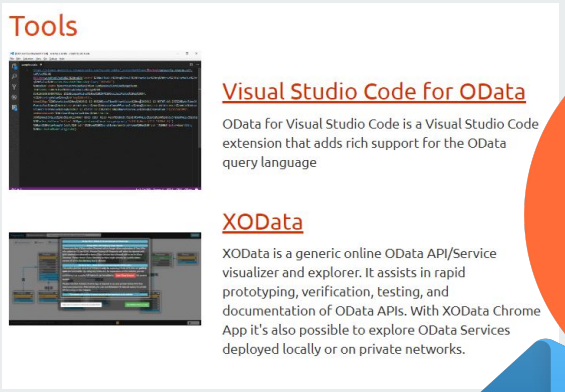
/UI2/FLPD_CUST
/n/IWFND/TRACES

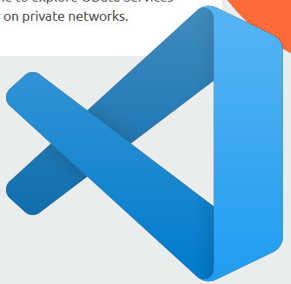
**SE38:** /UI5/APP_INDEX_CALCULATE
**SE38:** /UI2/INVALIDATE_CLIENT_CACHES
**SICF:** /sap/opu/odata/sap/*

https://community.sap.com/t5/technology-blogs-by-members/list-of-useful-sap-fiori-tcodes/ba-p/13193712

# TCODE & Utils

# tips & tricks



**HTTP Client** (Strumento essenziale)

- Visual Studio Code HTTP Client
- Postman

**oData Tools**

- Visual Studio Code for OData
- XOData