**REST API**

**Richiesta:**

queryCriteria:  
page  
pageSizekey (token per autorizzati)

**Risposta:**

"data": [{ json}],  
"current\_page":1,  
"results\_per\_page":1,  
"number\_of\_pages":1185,

**IDEE DESIGN REST API:**

<https://techbeacon.com/guide-restful-api-design-35-must-reads>

**Idee e spiegazioni:**

**Qualita api:**

1. Affordance:  
   Intuitiva, richiede rari accessi alla documentazione  
   Come:  
   nomi, unico modo per ottenere un risultato.
2. Corretta granularita: (bilanciare due esigenze opposte)  
   Molte chiamate ad api VS troppe informazioni in payloads
3. Standards:
   1. Livello 1, 2, 3, 4 (Hateoas)

**Documentazione**

1. Inserire esempi di chiamate con curl (pronti per copy paste)

**Best practice:**

**SUBDOMAINS:**

In produzione prevedere accesso tramite subdomains:

* API – [https://api.{fakecompany}.com](https://api.company-name.com/)
* OAuth2 – [https://oauth2.{fakecompany}.com](https://oauth2.company-name.com/)
* Developer portal – [https://developers.{fakecompany}.com](https://developers.company-name.com/)

1. Effectively consume the API
2. Get an OAuth2 token to consume the API
3. Access the API developer portal

**NOMI RISORSE:**

1) Sempre plurale (per accomunare i casi del singolo e del plurale)   
Es.  
/customers/1  
/customers/sort=name,page=7,pageRecords=30

**FUNZIONALITA BASE:**

1. CRUD  
   GET (Retrieve)  
   POST (Create)  
   PUT( Update)  
   DELETE (Delete)
2. Funzioni comuni  
   sort  
   customers/sort=year  
     
   filters:  
   customers/year=>10&location=roma

Search:

Simile al filters.  
Diverso e’ in genere il tipo di risultato.  
  
Simile ai filters= restringe la collections ai record che corrispondono ai filtri.  
Diverso i parametri in ingresso possono permettere sintassi particolari (es   
ed i dati di risposta possono essere diversi dalla risorsa standard.   
(Es. Insieme di link agli id dei risultati)

paging:

Supporto per la paginazione.  
Restituire in risposta:  
1) il numero di records della page,  
2) Il numero di page  
3) il criterio di ordinamento (stessa chiamata deve restituire stesso dataset)

Fields:

Permettere di scegliere quali campi visualizzare della risorsa (utile per mobile)  
 /customers/fileds=name,surname,telephone

**SECURITY:**

Oauth2

With regard to OAuth2 token validation, we recommend implementing Google’s solution, implicit grant flow:

<https://developers.google.com/accounts/docs/OAuth2UserAgent#validatetoken>  
<http://en.wikipedia.org/wiki/Confused_deputy_problem>

We recommend always using HTTPS wwhen communicating with :

OAuth2 providers  
API providers

To validate your OAuth2 implementation, you might want to try the following test:

Develop a client consuming your OAuth2 implementation and make a call to your API  
Then, replace the domain names of your API with Google’s API domain names.

If it works, you’re good to go !

**COERENZA NELLA SCELTA DELLE VARIABILI**   
(CamelCase, SnakeCase, SpinalCase)  
Parametri sia della query string che del body della risposta.

**CONTENT NEGOTIATION:**Supportare almeno json (default)e xml

**HATEOAS:**Collegare ogni risultato ad una coppia di link e url.  
Link definisce l operazione logica.  
Url e l url a cui invocare il servizio per l operazione di prima.  
  
**Vantaggi:**Fare in modo che i clients si colleghino solo al link (operazione logica) e siano quindi disaccoppiati dal url che puo’ essere cambiato sul server senza rompere i clients.  
  
api che si documentano con delle risposte standard. Rendono piu agevole il lavoro degli sviluppatori richiedendo meno accessi alla documentazione.

**CORS:**

**JSONP**

**ECCEZIONI:**1) Usare i codici http per errori a livello protocollo  
2) Mappare le eccezioni di business nel payload della risposta

**SUPPORTO PER CLIENT CHE NON SUPPORTANO PUT, DELETE E PATCH:**Tecnica del X-HTTP-Method-Override