

API Design

I «must have» per governare un'Architettura Headless



I must have per passare ad un'Architettura Headless

27 Mar 2019 - Nessun commento - by Denis Signoretto

Nel precedente articolo "[Architettura IT: il futuro è Headless](#)" abbiamo introdotto alcuni concetti chiave delle **Architetture Headless**, evidenziando come la loro funzione sia finalizzata a **supportare strategie digitali evolute**. Nello specifico, un'infrastruttura IT headless supporta l'integrazione di interfacce utente moderne in modo indipendente dai sistemi di backend e dai relativi vincoli tecnologici, disponendo una gestione centralizzata di contenuti e servizi e garantendo **esperienze digitali coerenti e soddisfacenti agli utenti**.

Per passare ad un approccio headless **non basta solo la tecnologia, ma è indispensabile un approccio strategico**. Gli elementi abilitanti sono certamente le API ma per evolvere un'infrastruttura IT verso un paradigma headless, non è sufficiente esporre API REST: serve definire una roadmap che supporti questa evoluzione, affrontando anche [le sfide che il mondo dell'API Management richiede](#).

6 elementi strategici per un approccio headless

1. La sicurezza dei dati *über alles*

Come punto di partenza fondamentale per passare ad un approccio strategico headless, è importante che l'infrastruttura identificata garantisca e offra servizi di:

- **Autenticazione delle API**, adottando standard che assicurino un'integrazione semplice con livelli di sicurezza

I must have per passare ad un'Architettura



6. Analisi del dominio e design delle API

In ultima, ma non per questo di secondaria importanza, è fondamentale che le API siano strutturate secondo un modello che rappresenti le entità del dominio, le proprie modalità di business, le depuri e le renda indipendenti dalle complessità dei sistemi legacy creando un adeguato **strato di interfaccia** che favorisca la **riusabilità** e **isoli i client dalla complessità e dalle interdipendenze dei sistemi sottostanti**.

Gli elementi abilitanti sono certamente le API ma per evolvere un'infrastruttura IT verso un paradigma headless, non è sufficiente esporre API REST: serve definire una roadmap che supporti questa evoluzione, affrontando anche le sfide che il mondo dell'API Management richiede.

6 elementi strategici per un approccio headless

1. La sicurezza dei dati liber allies

Come punto di partenza fondamentale per passare ad un approccio strategico headless, è importante che l'infrastruttura identificata garantisca e offra servizi di:

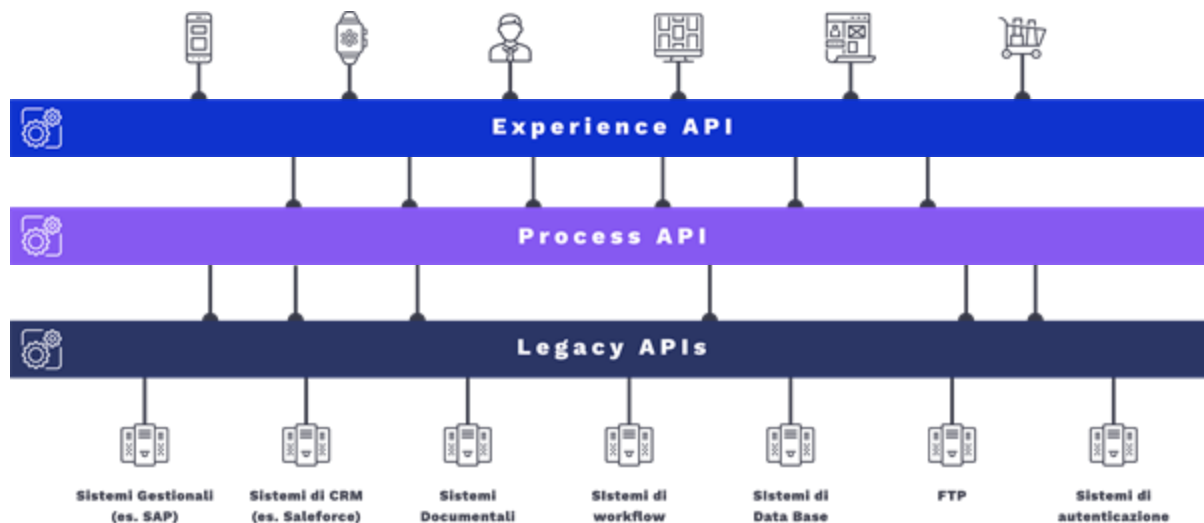
- Autenticazione delle API, adottando standard che assicurino un'integrazione semplice con livelli di sicurezza



A step back, first ...

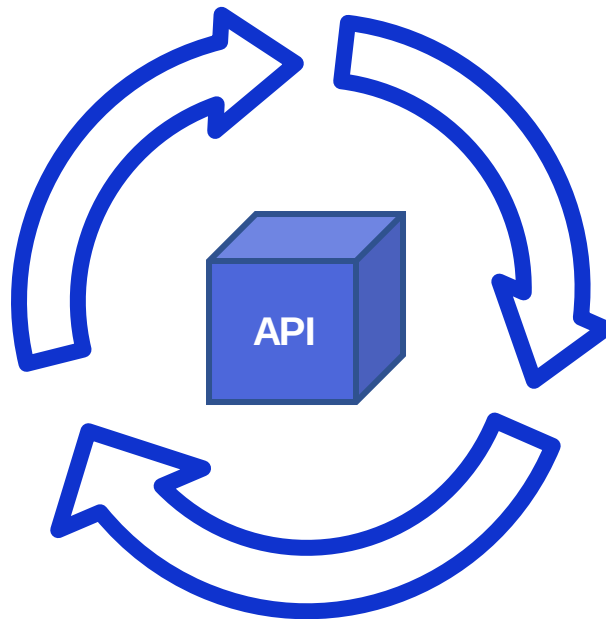
API vs Web Services

No siloes & Multiple Consumers



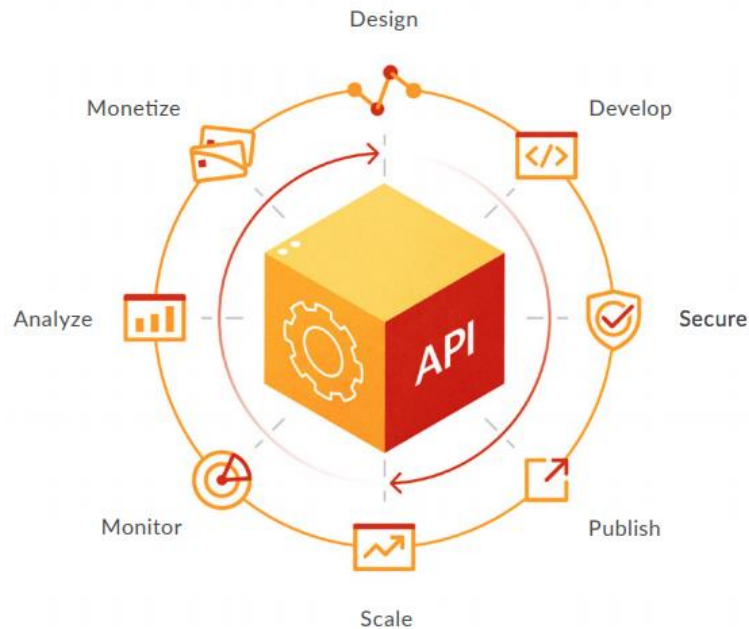
API vs Web Services

Reuse



API as a Product

Life-Cycle



API as a Product

The background of the slide is a dark blue, monochromatic image of a vinyl record. A tonearm and stylus are visible on the right side, resting on the record's surface. The record's grooves are clearly visible, and the overall image has a soft, blurred quality.

SIDE A

Who are the API customers?

Developers

What are their needs?



\$> Accessible and discoverable API

\$> Documentation

\$> Standards

\$> Adoption of known best practices and conventions

\$> Sandbox

\$> Examples

\$> Accessible and discoverable API

\$> Documentation

\$> Standards

\$> Adoption of known best practices and conventions

\$> Sandbox

\$> Working Examples !!!

SIDE B

Is there an «API Product Manager»?

API Design is an important pillar

How to put API Design into Practice ?!?



Case History Journey

in domain Modeling & Rest API Design

API Design approach

Domain Design

- Functional & Non-functional Requirements
- Summary of terms
- Use Case, Scenarios and Acceptance Criteria
- Flow and Sequence Diagrams

API Design

- Consumer-Oriented Design
 - “Outside-in” approach
 - API First Design
- Agile Design
- Mock and Simulation

Domain & API

Specification Languages

Domain Modeling

UML Use Cases

Scenarios and acceptance criteria

As a logged-out user

I want to be able to sign in to a website

So that I can find access my personal profile

Scenario: System user signs in with valid credentials

Given I'm a logged-out system user

and I'm on the Sign-In page

When I fill in the "Username" and "Password" fields with my authentication credentials

and I click the Sign-In button

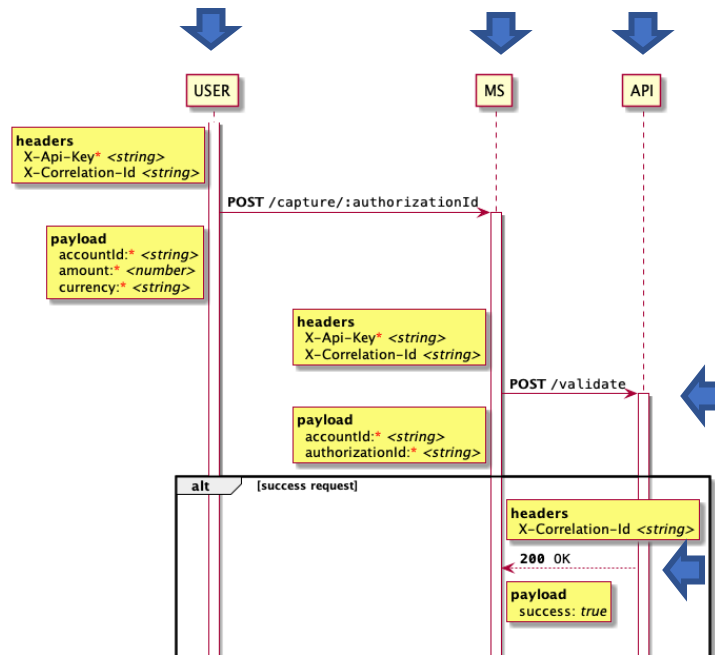
Then the system signs me in"



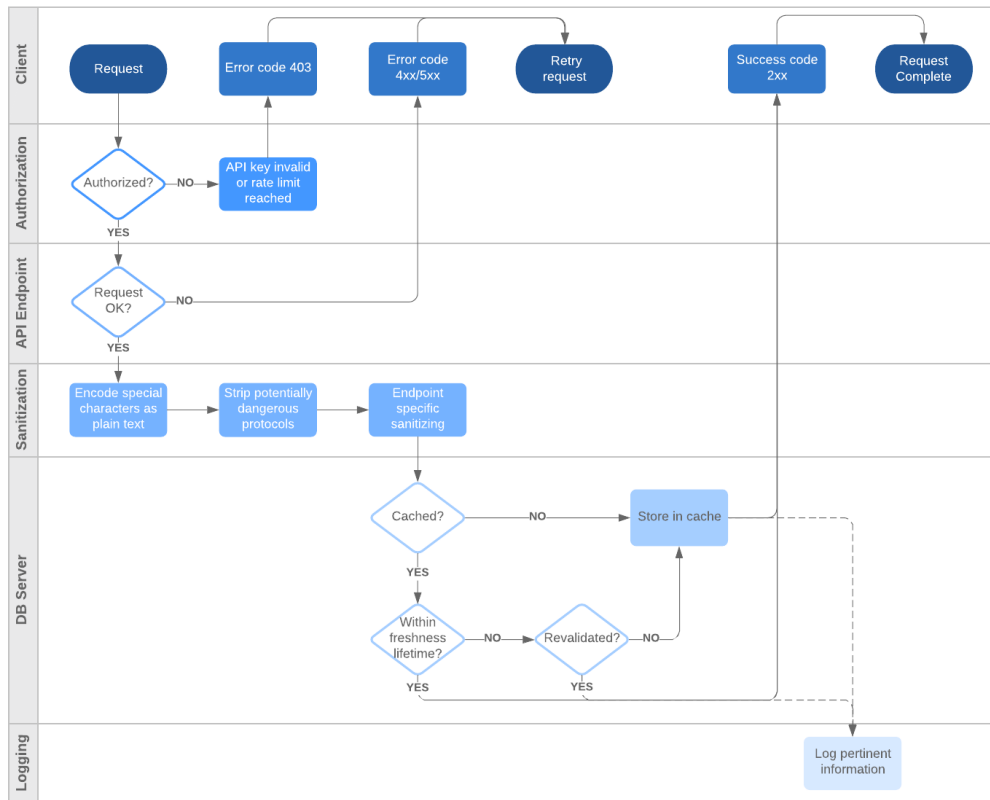
UML Sequence Diagram

Sequence flow:

- Endpoints
- Resources Path & Parameters
- HTTP Verbs
- HTTP status codes
- Extra info: Headers



UML Flow Diagram



Functional and non functional requirements

API Design

«Outside-in» vs «Inside-out» and API First approach

Rest API Design Language

OpenAPI (aka Swagger)

The screenshot displays the Swagger Editor interface, which is used for creating and editing OpenAPI specifications. The interface is divided into two main sections: a code editor on the left and a preview on the right.

Swagger Editor Header:

- Logo:
- File
- Edit
- Generate Server
- Generate Client
- Switch back to previous editor

Code Editor (Left Panel):

```
1 swagger: "2.0"
2 info:
3   description: "This is a sample server Petstore server. You can find
4     out more about Swagger at [http://swagger.io](http://swagger.io)
5     or on [irc.freenode.net, #swagger](http://swagger.io/irc/). For
6     this sample, you can use the api key `special-key` to test the
7     authorization filters."
8   version: "1.0.0"
9   title: "Swagger Petstore"
10  termsOfService: "http://swagger.io/terms/"
11  contact:
12    email: "apiteam@swagger.io"
13  license:
14    name: "Apache 2.0"
15    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
16 host: "petstore.swagger.io"
17 basePath: "/v2"
18 tags:
19 - name: "pet"
20   description: "Everything about your Pets"
21   externalDocs:
22     description: "Find out more"
23     url: "http://swagger.io"
24 - name: "store"
25   description: "Access to Petstore orders"
```

Preview (Right Panel):

Swagger Petstore 1.0.0

[Base URL: petstore.swagger.io/v2]

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on irc.freenode.net, #swagger. For this sample, you can use the api key **special-key** to test the authorization filters.

[Terms of service](#)

[Contact the developer](#)

[Apache 2.0](#)

[Find out more about Swagger](#)

Schemes

HTTP

[Authorize](#)

Rest API Definition

- Entities Schema
- Resources and paths
- HTTP Verbs and HTTP status codes
- Media Types
- Documentation (paths, entities, parameters,...)
- Examples
- ... and



Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

20% complete



For more information about this issue and possible fixes, visit <https://www.windows.com/stopcode>

If you call a support person, give them this info:

Stop code: CRITICAL_PROCESS_DIED

Rest API Definition

HTTP STATUS CODES

4xx Client Error

400 Bad Request
401 Unauthorized
402 Payment Required
403 Forbidden
404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout
409 Conflict
410 Gone

5xx Server Error

500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported
506 Variant Also Negotiates
507 Insufficient Storage
508 Loop Detected
510 Not Extended

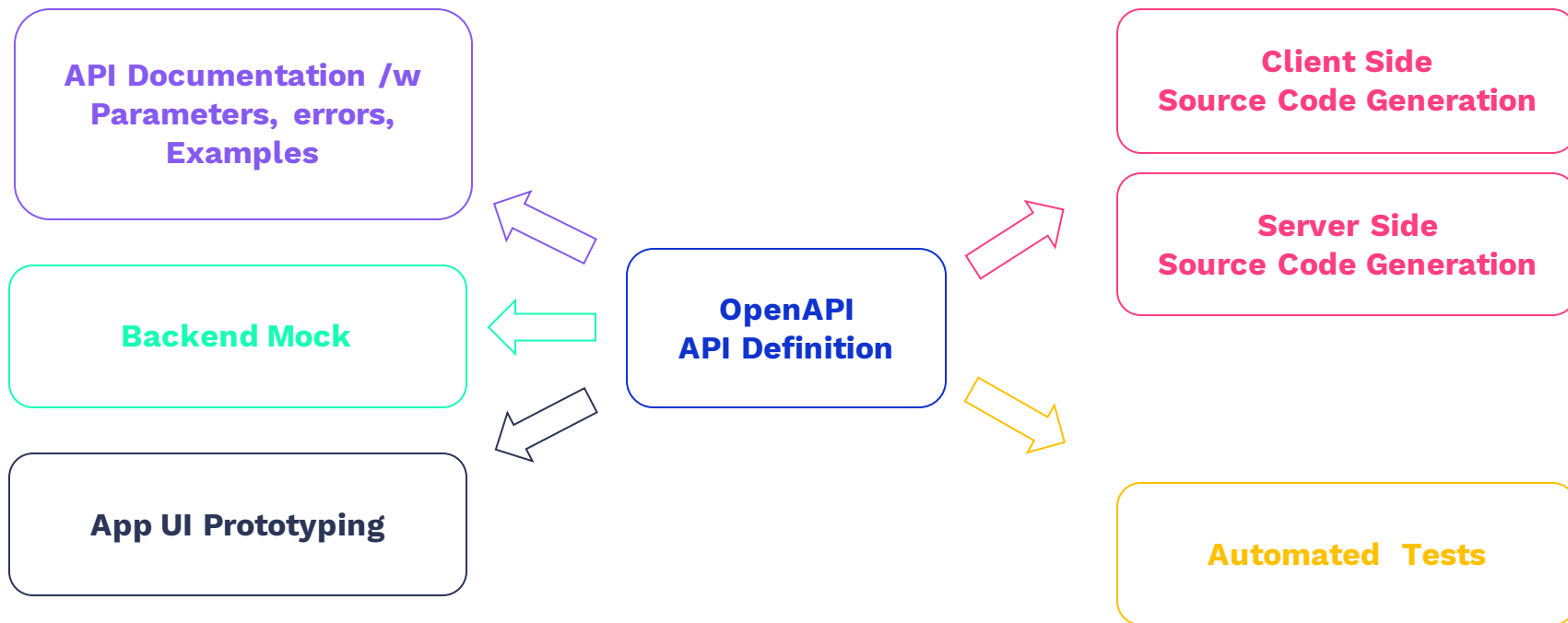
ERRORS generated by intermediate Layers

- Reverse Proxies
- API Gateway
- EBS
- ...

DESIGN for problems

- **RFC 7807 (JSON Problem)**
- Google Errors, Facebook Errors,...
- Error code
- Human Error message
- Error description
- Internal reference ID & documentation

API First advantage



Conclusions

Importance of API Design

Un buon design rende **efficienti**
ed **efficaci** gli **sviluppatori** sia in fase
di prototipazione e integrazione
sia in fase di **problem solving** e **supporto**

Importance of API Design

Cambiare le API una volta pubblicate
impatta sul ciclo di vita delle API

API Architect

- Comprende i **bisogni operativi** degli sviluppatori
- Introduce, fa comprendere e utilizzare gli **strumenti di specifica**
- Assicura tecnicamente **qualità, documentazione, esempi**
- Garantisce **uniformità** e **adesione** agli **standard**
- Garantisce la **sicurezza**
- **Monitoring** ai fini delle **performance** e **disponibilità** delle API

API Product Manager

- Comprende i **bisogni degli utilizzatori in termini di funzionalità**
- Assicura strumenti per garantire **documentazione, facilità di utilizzo e integrazione**
- Promuove l'adozione delle API nel sistema azienda
- **Tiene sotto controllo e massimizza il ROI (Return of Investment)**
- Monitoring e **valuta KPI delle API**

Grazie

Denis Signoretto
IT Architect & Senior Project Manager

@denissignoretto

Email: denis.signoretto@intesys.it

