OpenAPI

Marco A.P.F

Mail: marcoapferegrino@gmail.com
Twitter: @ApfMark
Linkedin: Marco A. Pérez Feregrino

¿De que trata el curso?

- Conocer la especificación de OpenAPI
- Ser capaces de Diseñar un API RESTful
- Ventajas de diseñar un API antes de programarla

Lo que **NO** veremos

- Programación
- Test, implementación, nube

¿Qué es un API?

- Application Programming Interface
- Conjunto de funcionalidades para conectar sistemas
- Producto para extender funcionalidad

Tipos de APIs

Alcance: Internas, Partners, Públicas

Síncronas y asíncronas

REST, RPC, Event driven

¿Qué es HTTP?

"Hypertext Transfer Protocol"

*"HTTP es un protocolo que permite **obtener recursos**. Es la base de cualquier intercambio de datos en la Web enfocado a cliente-servidor."*

*"HTTP es un protocolo de nivel de aplicación **sin estado** ..."*

Métodos HTTP

POST, PUT, DELETE, PATCH, HEAD, CONNECT, OPTIONS, TRACE

RFC 7231

HTTP Method	Idempotente	Seguro
POST	NO	NO
GET	SI	SI
HEAD	SI	SI
PUT	SI	NO
PATCH	NO	NO
DELETE	SI	NO

HTTP 1.1

- Transfiere texto plano
- Se tiene que romper la conexión TCP cada vez que se requiere hacer una nueva petición :(
- múltiples paquetes de datos no pueden pasar entre sí cuando viajan al mismo destino
- Puede haber muchas conexiones pero afecta el rendimiento.

Demo link

HTTP2

- Transfiere con una capa de binario
- Mantiene verbos, métodos y headers de su versión anterior
- Mayor velocidad al empaquetar en menor tamaño los paquetes de datos
- Con una conexión puede haber múltiples streams de data
- Múltiples respuestas del servidor (*server push*)

¿Qué es REST?

“Representational State Transfer ”

“Patrón de arquitectura para desarrollo web y sistemas distribuidos”

Beneficios

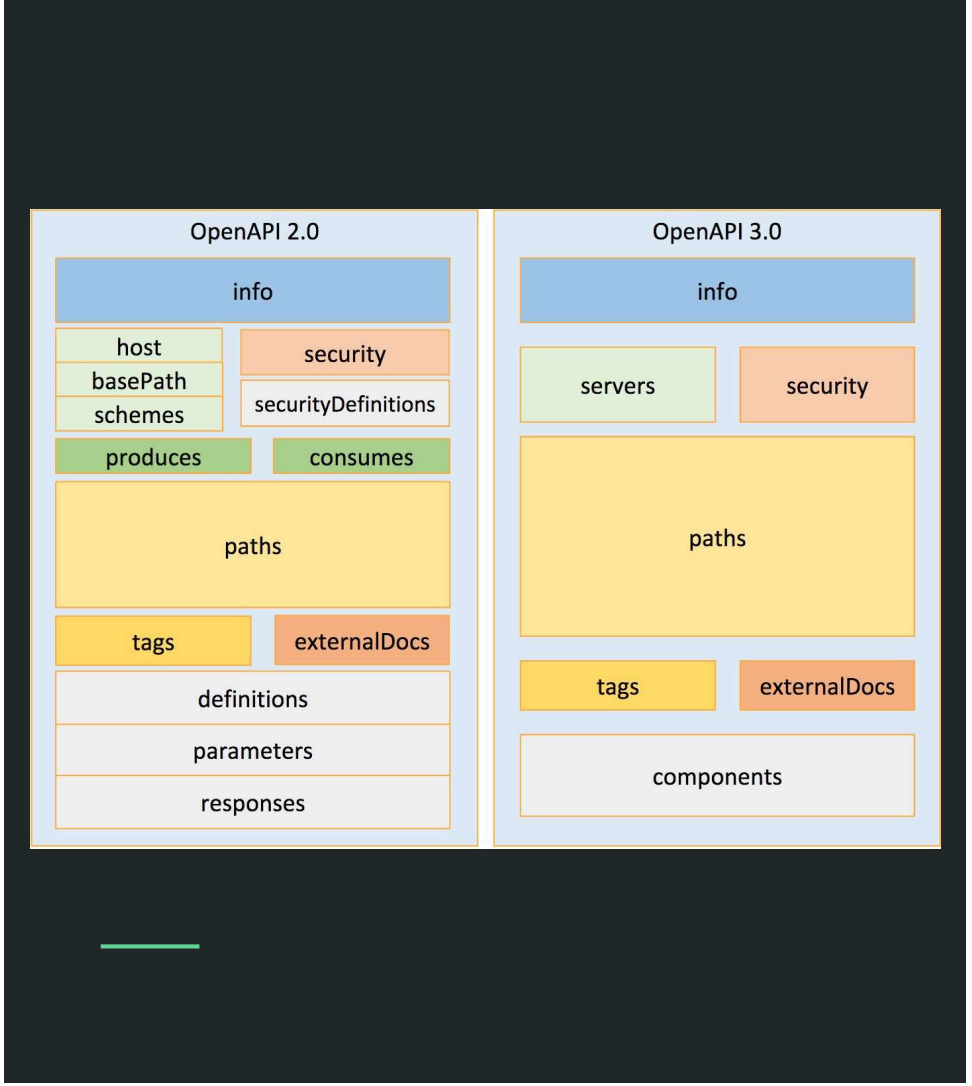
- **Performance:** “Serve and forget” No tiene contextos
- **Sacalability:** Como no tiene contexto y es cliente servidor puede tener tantos nodos se requieran
- **Simplicity:** Sólo necesitas 1 endpoint no toda la documentación técnica y de negocio para comprenderla
- **Reliability:** La implementación no la conoce el consumidor pues consume una interfaz uniforme

Constraints

- **Client-Server:** “request y response”**,** Mínimo de información posible para realizar la funcionalidad
- **Stateless:** Amnesia entre una petición y otra; sessions, context, memoria etc
- **Cacheable:** Tanto como sea posible
- **Layered:** El consumidor no sabe que hay detrás, microservicios, proxy, cache, justo eso permite que sea escale el bajo acoplamiento.
- **Uniform Interface:** Nombrado, convenciones, guías de estilo

¿Qué es OpenAPI specification?

Defines a **standard**, **language-agnostic** interface to RESTful APIs which allows both **humans and computers** to **discover** and **understand** the **capabilities** of the service without access to source code, documentation, or through network traffic inspection



Beneficios

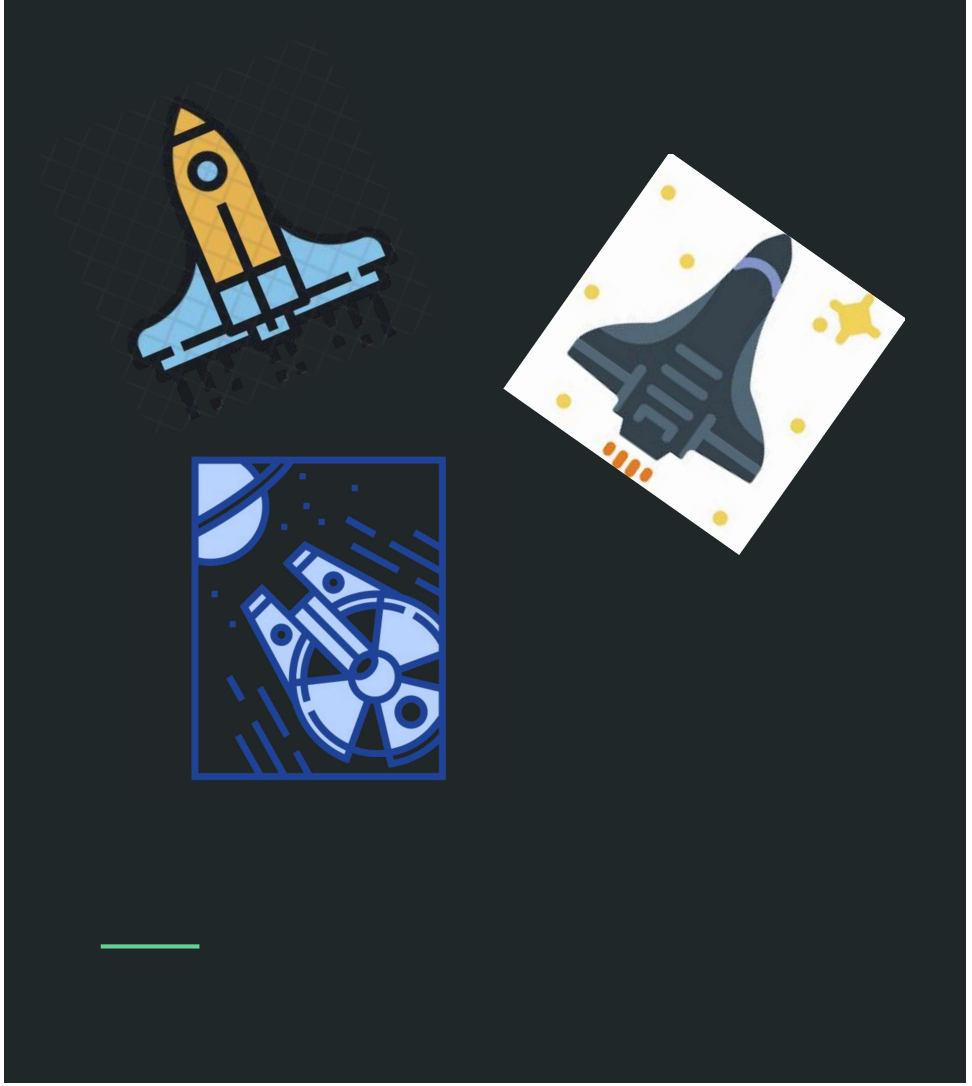
- Generar documentación y mostrar el API
- Generación de código: servidores y clientes
- Pruebas
- Una sola fuente de verdad
- Homogeneidad

Ejercicio Práctico Por fin !!

Inventario Espacial

En una estación espacial necesita el **inventario** de las **naves espaciales**.
Se ha encomendado a un grupo de diseñadores de APIs la tarea de diseñar de forma escalable y reutilizable un API, considerando posibles escenarios como la creación, actualización y el borrado de una nave así como dar de alta los viajes de las naves

Aunque no está en el alcance se deberá considerar la bitácora de vuelos de las naves.
Se necesitará saber información de los pilotos así como de sus viajes.



Componentes

Estructuras comunes, reusables y organizadas.

```yaml
components:
  # Reusable schemas (data models)
  schemas:
    ...
  # Reusable path, query, header and cookie parameters
  parameters:
    ...
  # Security scheme definitions (see Authentication)
  securitySchemes:
    ...
  # Reusable request bodies
  requestBodies:
    ...
  # Reusable responses, such as 401 Unauthorized or 400 Bad Request
  responses:
    ...
  # Reusable response headers
  headers:
    ...
  # Reusable examples
  examples:
    ...
```

Guidelines

Puntos de nuestra API a considerar !

1. **Formatos**
2. **JSON**
3. **Namings**
4. **URIS**
5. Versionado
6. Documentación

Guidelines Formato

Monedas en **ISO-4217** : MXN, EUR USD
Países en **ISO-3166-1_alpha-2**, PE, MX, AR
Fechas y tiempos en **RFC3339**: 1937-01-01T12:00:27.87+00:20

Guidelines JSON

- JSON: **RFC7159**
  - Respuestas siempre en un wrapper de **data**
    - `{`
    - `    "data": {`
    - `        ...`
    - `    }`
    - `}`
  - Valores null no son aceptados en el JSON es mejor no usarlos

Guidelines Naming

- Nombres de recursos deben ser **sustantivos en plural no verbos**.
  A menos que se use un patrón controller, ahí puede usar un verbo pero no es un recurso lo que se está modelando.
- Nombres de atributos deben seguir por ejemplo : **lowerCamelCase**, **lower_case**
- Nombres de valores ENUM en **UPPER_CASE**
- Arrays deben estar en plural

Guidelines URIs

- Nombre en lower case. Ejemplo **/this-lower-case**
- No tiene que tener / al final de la URI
- Cada recurso tiene su propio identificador, no puede tener varios identificadores. Ejemplo no válido: **/customer/1234/abcdef**
- Deben tener un orden jerárquico

Modelo de Madurez de Richardson