ILAF

Open Geospatial Consortium, Inc.

JIIDE 2018 MENORCA

Sentando las bases para una nueva generación de Servicios OGC basados en OpenAPI. El enfoque usado en WFS 3.0

JIIDE 2018, Manorca

Joan Masó

17 Octubre 2018

OGC®
Making location count.

Ya hablamos de esto un poquito el año pasado:

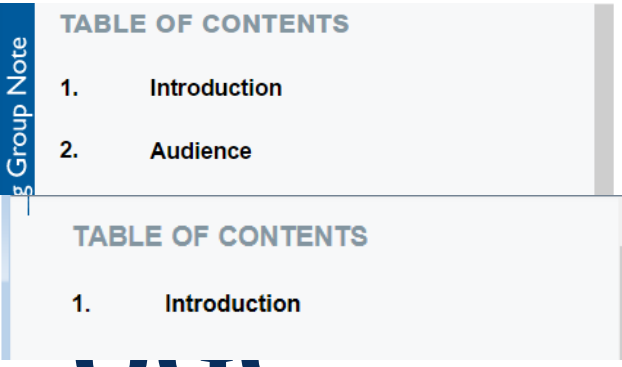
Mirando fuera o los efectos de las buenas prácticas para los datos espaciales en la web

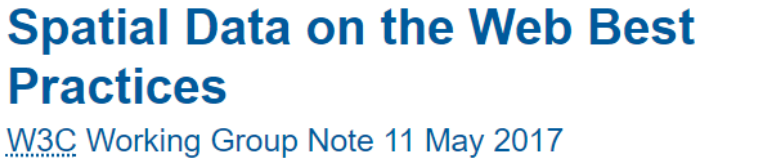
Fuertemente inspirado en una presentació de Clemens Portele para el St. John's TC. 27 Junio 2017

OGC®
Making location count.

Spatial Data on the Web Best Practice

- https://www.w3.org/TR/sdw-bp/
- OGC-W3C
- Tema: ***uso de tecnologías Web*** *en tanto que pueda ser aplicada a* ***localización***
- Audiencia: ***profesionales***, *incluyendo* ***desarrolladores web*** *y* ***gestores de datos espaciales***
- *Compilado en base a* ***evidencias de aplicaciones en el mundo real***
- Extiende W3C DWBP para cubrir aspectos relativos a la información espacial.





From Linda's Opening Plenary slides

¿Qué pasa con tanta API? (1/2)

- En el pasado las aplicaciones web no podían conectar y definimos la idea de “interoperabilidad”.
- Aprendimos que debíamos usar **interfaces interoperables**
- Pero esto no es tan fácil:
  - Debemos crear un servidor y un cliente por separado.
  - En la práctica, si deseo incrustar un mapa en mi cliente, necesito programarme un cliente que use WMS/WMTS…
  - … lo que requiere bastante código.

OGC®

¿Qué pasa con tanta API? (2/2)

- Descubrimos que con el API de Google maps o de OpenLayers podíamos incrustar un mapa casi sin generar código.
- Esto ha conducido a una proliferación de las APIs y, a lo que es peor, a un cambio de actitud respecto a la interoperabilidad que se resume en:
- "**Yo** he definido **mi** API y soy interoperable. ¡**Tú** tienes que usar **mi** API!".

- Y la pregunta es:
  - **¿Y si no me da la gana?**
- Es necesario pensar que papel deben tener las API el OGC

OGC®

¡¡Llevamos intentando hacer R**T desde hace 10 años!!.

OGC®
Making location count.

SOA and ROA concepts

https://portal.opengeospatial.org/files/?artifact_id=31575&version=1 Closing plenary slide 112

- The recognition of two Architecture Orientations and the separation between SOA and ROA was done in the **Valencia** TC in back in **2008**

Services Motion

- The Architecture DWG Services Subcommittee recommends that the OGC Technical Committee promote development of platform-independent abstract specifications and platform-dependent implementation specifications for all OGC service standards that support both procedure-oriented and resource-oriented service styles or patterns
  - Clarification: (RPC and REST within an SOA)
  - Motion: Josh Lieberman
  - Second: John Herring
  - Discussion: If adopted, this can be dropped by the TC for service standards for which this is not appropriate.
- There was no objection to unanimous consent

OGC®

Helping the World to Communicate Geographically

OGC®

Empezamos OpenAPI

OGC®
Making location count.

OpenAPI: Filosofia

- The OpenAPI Initiative (OAI) was created by a consortium of forward-looking industry experts who recognize the immense value of **standardizing** on how **REST APIs** are described.
- An open governance structure under the **Linux Foundation**,
- Antes se llamaba **Swagger** (SmartBear Software) pero ha sido "donado"

- **APIs** form the connecting glue between modern applications. Nearly every application uses APIs to connect with corporate data sources, third party data services or other applications.
- Creating **an open description format for API services** that is vendor neutral, portable and open is critical to accelerating the vision of a truly connected world.

OGC®

OpenAPI vs. Capabilities

- OpenAPI es como un documento de Capabilities
  - Que describe el API y el Service
  - Tiene un contenido similar
    - Operations
    - Content
    - Security
- OpenAPI es más que un documento de Capabilities
  - es un documento de diseño
  - Puede ser procesado y genera código!!.

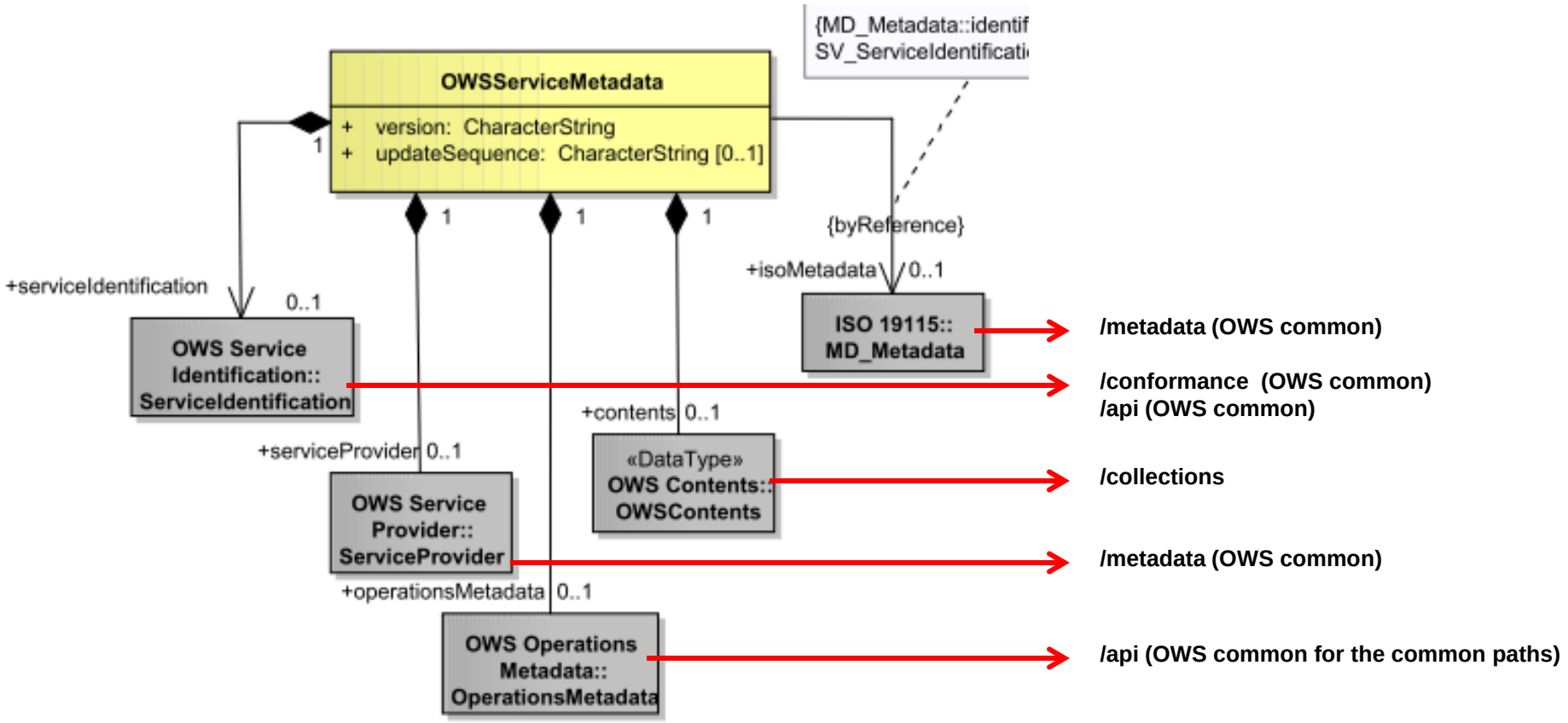
OGC®

Estructura

Un documento OpenAPI (OAS) contiene las siguientes secciones:

- Information –metadatos
- Paths – aquí és donde trabajas con **los recursos**.
  - Especifica URL templates
  - SOLAMENTE HTTP Operations: GET, PUT, POST, DELETE
- Queries – Puedes incluir filtros despues del ‘?’.
- Components –
  - Parametros que aplican a todas las operaciones de este path
- Servers – puede haber más de uno
- Security – incluida en tanto que puede usar HTTPS, OAuth2, OpenID Connect, API Key

A donde va a parar el GetCapabilities?



OGC®

~~ows:OperationsMetadata~~ → /api

- https://raw.githubusercontent.com/opengeospatial/WFS_FES/master/openapi.yaml
- https://editor.swagger.io/

```
              $ref: '#/components/schemas/root'
          text/html:
            schema:
              type: string
'/conformance':
  get:
    summary: information about standards that this API conforms to
    description: >-
      list all requirements classes specified in a standard (e.g.,
        WFS 3.0
      Part 1: Core) that the server conforms to
    operationId: getRequirementsClasses
    tags:
      - Capabilities
    responses:
      '200':
        description: the URIs of all requirements classes supported
          by the server
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/req-classes'
      default:
        description: An error occured.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/exception'
'/collections':
  get:
    summary: describe the feature collections in the dataset
    operationId: describeCollections
    tags:
      - Capabilities
```

Capabilities Essential characteristics of this API including information about the data.

GET	/	landing page of this API
GET	/conformance	information about standards that this API conforms to
GET	/collections	describe the feature collections in the dataset
GET	/collections/{collectionId}	describe the {collectionId} feature collection

Features Access to data (features).

GET	/collections/{collectionId}/items	retrieve features of feature collection {collectionId}
GET	/collections/{collectionId}/items/{featureId}	retrieve a feature; use content negotiation to request

~~ows:Content~~ → /collections

```yaml
'/collections':
  get:
    summary: describe the feature collections in the dataset
    operationId: describeCollections
    tags:
      - Capabilities
    responses:
      '200':
        description: Metdata about the feature collections shared by
          this API.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/content'
          text/html:
            schema:
              type: string
      default:
        description: An error occured.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/exception'
          text/html:
            schema:
              type: string
'/collections/{collectionId}':
  get:
    summary: 'describe the {collectionId} feature collection'
    operationId: describeCollection
    tags:
      - Capabilities
    parameters:
```

200

Metdata about the feature collections shared by this API.

application/json

Controls Accept header.

Example Value | Model


```
  "collections": [
    {
      "name": "buildings",
      "title": "Buildings",
      "description": "Buildings in the city of
Bonn.",
      "links": [
        {
          "href":
"http://data.example.org/collections/buildings/items
",
          "rel": "item",
          "type": "application/geo+json",
          "title": "Buildings"
        },
        {
          "href":
"http://example.org/concepts/building.html",
          "rel": "describedBy",
          "type": "text/html",
          "title": "Feature catalogue for buildings"
        }
      ],
```


OGC®

~~Profiles~~ → /conformance

```
            schema:
              type: string
'/conformance':
  get:
    summary: information about standards that this API conforms to
    description: >-
      list all requirements classes specified in a standard (e.g.,
        WFS 3.0
      Part 1: Core) that the server conforms to
    operationId: getRequirementsClasses
    tags:
      - Capabilities
    responses:
      '200':
        description: the URIs of all requirements classes supported
          by the server
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/req-classes'
      default:
        description: An error occured.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/exception'
'/collections':
  get:
    summary: describe the feature collections in the dataset
    operationId: describeCollections
    tags:
      - Capabilities
    responses:
      '200':
```

200

the URIs of all requirements classes supported by the server

No links

application/json

Controls Accept header.

Example Value | Model

```
{
  "conformsTo": [
    "http://www.opengis.net/spec/wfs-1/3.0/req/core",
    "http://www.opengis.net/spec/wfs-
1/3.0/req/oas30",
    "http://www.opengis.net/spec/wfs-1/3.0/req/html",
    "http://www.opengis.net/spec/wfs-
1/3.0/req/geojson"
  ]
}
```

default

An error occured.

No links

application/json

Example Value | Model

```
{
  "code": "string",
  "description": "string"
```

OGC®

Hay que aprender otro encoding: YAML

```yaml
openapi: 3.0.0
info:
  title: OGC Web Feature Service
  version: 3.0.0-alpha
  contact:
    name: Acme Corporation
    url: 'http://example.org/'
  license:
    name: CC-BY 4.0 license
servers:
  - url: 'https://example.org/ad'
    description: >-
      The production server.
paths:
  /:
    get:
      summary: the API description - this document
      operationId: getApiDescription
      tags:
        - Capabilities
      parameters:
        - $ref: '#/components/parameters/f1'
```

```yaml
responses:
        '200':
          content:
            application/openapi+json;version=3.0:
              schema:
                type: object
            text/html:
              schema:
                type: string
  '/{featureCollection}':
    get:
      summary: retrieve features of a feature collection
      operationId: getFeatures
      tags:
        - Features
      parameters:
```

Es como un JSON pero sin comillas ni corchetes.

OGC®

Limitaciones de OpenAPI

- Pub-Sub
  - https://www.asyncapi.com/ - extends OpenAPI to include Pub-Sub
  - Pub-Sub is not REST – not likely to become part of OpenAPI
- WebSockets – not a part of the REST universe
- Non-HTTP Protocols (Video Streaming over RTP)
- Unusual Encodings (Protobuf, GeoPackage)
- Mix of anonymous and authenticated users
- Others (TBD)

WFS3.0: Un Proceso abierto

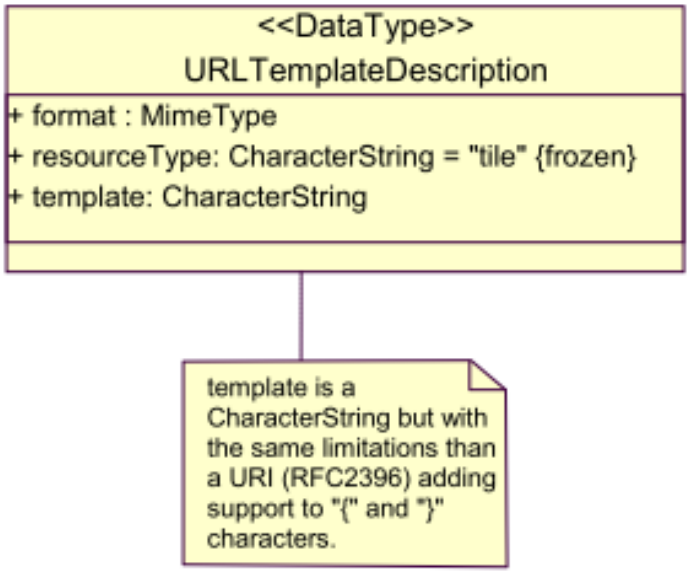
- Es posible participar en cualquier momento el GitHub
  - https://github.com/opengeospatial/WFS_FES
- OGC organizó un Hackathon
  - 6 and 7 March 2018 in Fort Collins, CO, USA
  - https://github.com/opengeospatial/wfs3hackathon
- OGC Seeks Public Comment on WFS 3.0 Candidate Standard
  - http://www.opengeospatial.org/pressroom/pressreleases/2830
  - 3 July 2018 UTC

OGC®

Otros estandares lo estan probando.

OGC®
Making location count.

WMTS: No URL templates in the capabilities: We had to invent something



- <layer>
  - <ResourceURL format="image/png" resourceType="tile“ template="http://www.maps.bob/etopo2/default/{TileMatrixSet}/{TileMatrix}/{TileRow}/{TileCol}.png">
  - <ResourceURL format="application/gml+xml; version=3.1" resourceType="FeatureInfo" template="http://www.maps.bob/etopo2/default/TileMatrixSet}/{TileMatrix}/{TileRow}/{TileCol}/{J}/{I}.xml">
- </layer>

OGC®

Lack of a mechanism to define the URI variables. We had to invent something

Table 32 — URL template variables and possible values for tile

URL template variable	Meaning	Possible values	Multiplicity
"style"	Style identifier	identifier in Table 7 ./style/ows:Identifier	One (mandatory) [a]
./Dimension/ ows:Identifier	Dimension value	identifier in Table 9 ./Dimension[ows:Identifier={./Dimension/ ows:Identifier}]/Value	One for each dimension available (mandatory if there are dimensions defined)
"TileMatrix Set"	tile matrix set identifier	identifier in Table 6 ./ TileMatrixSetLink/TileMatrixSet	One (mandatory) [a]
"TileMatrix"	tile matrix identifier	identifier in Table 14 /Capabilities/Contents/TileMatrixSet[ows:Identifier={TileMatrixSet}]/TileMatrix / ows:Identifier	One (mandatory) [a]
"TileRow"	row index of tile matrix	If TileMatrixSetLimits is present, see Table 10, (./TileMatrixSetLimits), SHALL be any integer value between MinTileRow and MaxTileRow in Table 12 (both included) (./tileMatrixSetLimits/tileMatrixLimits[./ TileMatrix={TileMatrix}]/MinTileRow and ./tileMatrixSetLimits/tileMatrixLimits[./TileMatrix={TileMatrix}]/MaxTileRow). else SHALL be any integer value between 0 and MatrixHeight – 1, see Table 14, (both included) (0 and /Capabilities/Contents/TileMatrixSet[ows:Identifier={TileMatrixSet}]/TileMatrix[ows:	One (mandatory) [a]

- Solved by OpenAPI

OGC®

New draft WMTS 3.0 OpenAPI

- https://portal.opengeospatial.org/wiki/WMS1x4swg/WmtsWebApi
- https://editor.swagger.io/

Swagger Editor File Edit Generate Server Generate Client

```
166                 type: string
167   '/collections/{collectionId}/tiles':
168     get:
169       summary: 'retrieve a set of tiles of layer {collectionId}'
170       description: >-
171         Every tile in a dataset belongs to a layer. A dataset may
172         consist of multiple layers. A layer is often a renderization of
173         collection of features of a similar type, based on a common schema.\
174
175         Use content negotiation to request a ZIP or a multipart file.
176       operationId: getTiles
177       tags:
178         - Tiles
179       parameters:
180       - $ref: '#/components/parameters/collectionId'
181       - $ref: '#/components/parameters/bbox'
182       - $ref: '#/components/parameters/time'
183       responses:
184         '200':
185           description: >-
186             Information about the feature collection plus the tiles
187             matching the selection parameters.
188           content:
189             application/x-zip-compressed:
190               schema:
191                 type: string
192                 format: binary
193             multipart/related:
194               schema:
195                 type: string
196                 format: base64
197         default:
198           description: An error occured.
199           content:
200             application/json:
201               schema:
```

GET /collections/{collectionId}/tiles/{tileMatrixSet}/{tileMatrix}/{tileRow}/{tileCol}

retrieve a tile; use content negotiation to request PNG or a JPEG

Parameters | Try it out

Name	Description
collectionId * required string *(path)*	Identifier (name) of a specific collection
tileMatrixSet * required string *(path)*	Local identifier of a specific TileMatrixSet
tileMatrix * required string *(path)*	Local identifier of a specific TileMatrix. The name of the tile matrix are limited depending on the collectionId.
tileRow * required integer *(path)*	Row index of tile matrix. Value between 0 and MatrixHeight-1 of this tile matrix defined in the ServiceMetadata document.
tileCol * required	

OGC®

Buscando nuestros principios

OGC®
Making location count.

OGC Web API Guidelines

- https://github.com/opengeospatial/OGC-Web-API-Guidelines

Design Principles

Why OGC API Design Principles

The following is a set of common design principals for developing any Web API.

- The design of a Web API should follow a common pattern to ensure easy adoption.
- There are common design principles in main-stream IT that should be adopted to ease the adoption of OGC Web APIs.
- But still, there are some aspects that would need to be agreed upon to ensure seamless APIs for different thematic topics in OGC.
- In particular avoid that APIs are fundamentally different to access and manage different kinds of geospatial resources such as Features, Maps, Tiles, Coverages, Observations, Processes, etc.

Please follow the issues created for a discussion to observe the idea of the proposed OGC Web API design principles. Please use these Issues to discuss changes, corrections, and enhancements to the principles.

Important note: You may not and must not entirely agree with the presented principles. Important is to share your agreement or disagreement by contributing to an issue about the principle! Unfortunately issue numbers DOES NOT correspond to principle numbers. Relay on the issue title instead

Principle #1 – Don't Reinvent

Follow the discussion on GitHub Issue 3

For aspects and functional capabilities that are already solved in main-stream IT and meet geospatial community requirement, simply adopt these API elements.

OGC Web API Guidelines

- Principle #1 – Don’t Reinvent
- Principle #2 – Keep It Simple and Intuitive
- Principle #3a - Use Well-Known Resource Types
- Principle #3b – Keep the Base URL Simple
- Principle #4 – Use CRUD
- Principle #5 – Don’t mix Singular and Plural
- Principle #6 – Put Complexity behind the ‘?’
- Principle #7 – Error Handling
- Principle #8 – HTTP Status Codes
- Principle #9 – Use of HTTP Header
- Principle #10 - Pagination
- Principle #11 – Partial Responses
- Principle #12 – Not Accessing Resources
- Principle #13 – Metadata
- Principle #14 – Security
- Principle #15 – API Description
- Principle #16 - Content-Type Negotiation
- Principle #17a - Use IANA well-known identifiers
- Principle #17b - Use OGC services reserved URIs
- Principle #18 - Good APIs are testable at Design Phase already
- Principle #19 - Make use of geospatial relations

OGC®

Principle #4 – Use CRUD

- Allow **`CRUD Create, Read, Update, Delete`**
  - And **`Execute`**
- Allow HTTP methods that operate on resources
  - GET, POST, PUT, DELETE

Resource	POST	GET	PUT	DELETE
`/…/highways`	create a **new** highway	list all highways	bulk **update** of highways	delete all highways
`/…/highways/A8`	Error!	show A8	If exists: Update A8 else: Create A8	delete highway A8

- Also support HTTP communications methods
  - HEAD to return HTTP Headers with no payload
  - OPTIONS to support CORS

Principle #8 – HTTP Status Codes

- More then 70 HTTP status codes exist (summary in RFC 7231)
- You should reduce to the most important once
- Option Set #1 – Basic set
  - 200 – OK
  - 400 – Bad Request
  - 500 – Internal Server Error
- Option Set #2 – Additional
  - 201 – Created
  - 304 – Not Modified
  - 401 – Unauthorized
  - 403 – Forbidden
  - 404 – Not Found
  - 405 – Method Not Allowed
  - 409 – Conflict
  - 410 – Gone
  - 412 – Precondition Failed
  - 503 – Service Unavailable

Resumen

- OGC opta por procesos más abiertos
- OpenAPI parece haber venido para desencallar el R**T
- Existe el riesgo de que cada estándar vaya por su cuenta
- Necesitamos
  - que cada estándar experimente
  - Y participe en el diseño de una arquitectura común

OGC®

El año que viene cumplimos 10 años de ILAF

Primera reunión documentada en Murcia en 2009

OGC®
Making location count.

ILAF

Open Geospatial Consortium, Inc.

JIIDE 2018 MENORCA

Gracias

JIIDE 2018, Manorca

Joan.Maso@uab.cat

OGC®

Making location count.