

REUTILIZACIÓN DE LA INFORMACIÓN DEL SECTOR PÚBLICO

BUENAS PRÁCTICAS EN EL DISEÑO DE APIS Y LINKED DATA



ÍNDICE

OBJETIVOS DIDÁCTICOS	3
REQUISITOS PREVIOS	3
CONTENIDOS	4
1. API	4
1.1 Conceptos generales	5
1.2 ¿Qué problemas resuelve una API?	8
1.3 Servicios web y APIs REST	9
1.4 Web APIs Hoy	12
2. CONSIDERACIONES PARA EL DISEÑO DE APIS WEB	15
2.1 Modelo de datos: recursos y relaciones	16
2.2 Parámetros de búsqueda para el filtrado de resultados.....	22
2.3 Formato y paginación de resultados.....	23
2.4 Protocolo y seguridad: credenciales de acceso y cifrado.....	24
3. LINKED DATA	25
3.1 De los datos abiertos a los datos enlazados.....	25
3.2 Metadatos: de formatos estructurados a formatos semánticos	26
3.3 Conceptos básicos de la web semántica	27
3.4 ¿Qué problemas resuelve la web semántica?.....	29
3.5 Herramientas para el uso de <i>Linked data</i>	30
3.6 SPARQL: lenguaje de consulta de fuentes <i>linked data</i>	31
3.7 Datos enlazados hoy: ejemplos de proyectos institucionales relevantes.....	33
RESUMEN	37

OBJETIVOS DIDÁCTICOS

Comenzamos presentando los **Objetivos didácticos** de esta Unidad:

- ✓ Comprender qué es un API Web y su importancia como estándar de facto como mecanismo de interoperabilidad.
- ✓ Conocer las estrategias y proyectos institucionales más relevantes que apuestan por el uso de API Web como mecanismo para ofrecer datos abiertos gubernamentales.
- ✓ Entender los principios generales de las APIs web basadas en protocolo HTML.
- ✓ Ser capaz de realizar un diseño a alto nivel del esquema de direcciones (URLs) y operaciones (métodos) de un API REST para acceso a datos gubernamentales teniendo en cuenta las recomendaciones técnicas ya existentes.
- ✓ Comprender qué son los formatos semánticos y la importancia de los datos enlazados como solución técnica al movimiento por los datos abiertos.
- ✓ Entender la evolución y relación entre los conceptos de datos abiertos (open data) y datos enlazados (linked data)
- ✓ Conocer los proyectos institucionales más relevantes que apuestan por el uso de datos enlazados y web semántica para ofrecer datos abiertos gubernamentales.
- ✓ Entender los principios generales y tecnológicos de la web semántica, y su diferencia con formatos.
- ✓ Entender cómo funcionan las consultas semánticas mediante el lenguaje SPARQL.

REQUISITOS PREVIOS

Para poder asimilar los conceptos que vamos a desarrollar en la unidad, deberías contar con:

- Conocimientos técnicos de la web y del concepto de cliente / servidor.
- Preferiblemente, conocimientos del funcionamiento de Internet y de las distintas capas que componen una comunicación en Internet.
- Conocimientos generales sobre el funcionamiento de una base de datos relacional.

- Conocimientos generales de XML.
- Sería deseable algún conocimiento de programación contra recursos situados en servidores remotos.

CONTENIDOS

Resumimos los aspectos a tratar en los apartados de la Unidad:

1. API

Se describen los **conceptos generales** relacionados con las APIs y los **principios** que se deben tener en cuenta para asegurar un buen diseño de API.

2. CONSIDERACIONES PARA EL DISEÑO DE APIS WEB

Se enumeran los aspectos más importantes en el **diseño de APIs web**: modelo de datos, parámetros de búsqueda, formato, protocolos y seguridad.

3. LINKED DATA

Datos enlazados o **Linked data**, su relación con Open data y la relevancia que adquirirá en la gestión de la información, tanto pública como privada.

1. API

Las APIs son **elementos fundamentales** para la utilización masiva de datos y clave para la **conectividad** de diversas iniciativas. Las APIs, en general, proporcionan datos, si bien también pueden proporcionar los metadatos de los datos. En este primer tema vamos a conocer:

- **Conceptos generales:** Principales definiciones y conceptos sobre APIs.
- **Soluciones y requerimientos:** Vemos los problemas que resuelven las APIs y qué es necesario para que podamos aprovechar todas sus ventajas.
- **Servicios web y protocolos:** Conocemos los servicios web como mecanismos de interacción entre aplicaciones y sus protocolos.
- **Economía y ejemplos:** Vemos el estado actual de las APIs y las implicaciones que su uso tiene en la economía digital.

1.1 Conceptos generales

DEFINICIONES

A continuación vamos a conocer la definición de API, y de otros términos y conceptos relacionados con la misma:

- **API (*Application Programming Interface*):** La interfaz de programación de aplicaciones, abreviada del término inglés *API (Application Programming Interface)*, es la suma de subrutinas, funciones, procedimientos o métodos que un sistema de información ofrece para ser utilizado por otro sistema.
- **Vocabularios:** Definen la forma de estructurar los contenidos y su consenso en el ámbito de la web semántica y facilita la interacción entre los distintos sistemas. Cuando están normalizados sirven para definir elementos que son comunes del mundo real.

Ejemplo

Definir los campos necesarios para describir una persona, empresa, evento, producto, libro, película, etc.

- **Recurso:** Desde el punto de vista semántico, un recurso es cualquier objeto de información que podamos apuntar que no sea un literal asignado a una propiedad de un recurso.

Son identificados mediante un **identificador uniforme de recurso (URI)**¹ con la estructura que se describe en la pantalla siguiente.

+ MÁS INFORMACIÓN

El sitio de vocabularios más importante es schema.org, que contiene inventariados 642 tipos de objetos. Schema.org es un sitio web soportado por los principales buscadores mundiales. Incluye para cada esquema un conjunto abierto de especificaciones de datos para su uso en la web de los datos. Además, permite mecanismos de extensión de estos objetos base para su uso en aplicaciones específicas. A los distintos formatos se puede acceder de forma libre en su rama en github [Fuente: https://github.com/schemaorg/schemaorg](https://github.com/schemaorg/schemaorg).

¹ Del término inglés Uniform Resource Identifier (la U no corresponde a Universal como aparece en algunos textos, si bien tendría también sentido ya que el identificador es unívoco en el espacio de red en el que se encuentre). Definido por el IETF como RFC 3986.

Otro concepto relacionado es el de IRI (Internationalized resource identifier) que permite la introducción en las URI de caracteres internacionales para que la accesibilidad en cualquier juego de caracteres sea posible. Definido por el IETF como RFC 3987.

ESTRUCTURA URI

Su estructura está compuesta de:

- Estándar URI.
- URI en norma técnica de interoperabilidad.

ESTÁNDAR URI

{Esquema}+{Autoridad}+{Ruta}+{Consulta}+{Fragmento}

- **{Esquema}**
“HTTP;” El protocolo de conexión
- **{Autoridad}**
“USUARIO:CONTRASEÑA@AOURDOMAIN.COM:111” Identificación (si fuera necesaria), nombre del host y puerto de enlace (si fuera necesario).
- **{Ruta}**
“RECURSOS/DIRECCIONADO” Una ruta en host destino.
- **{Consulta}**
“CAMPO=DESCRIPTION” Por ejemplo pedimos la descripción de una de las propiedades del recurso.
- **{Fragmento}**
“2” nos ayudaría a ubicarnos dentro de la respuesta sobre el contenido del recurso.

Ejemplo

`http://usuario:contraseña@our.domain.com:1111/recursos/direccionado?campo=description#2`

URI EN NORMA TÉCNICA DE INTEROPERABILIDAD

`http://{base}/{carácter}/{sector}/{dominio}/{concepto}].[ext]`

- **{Base}**
“DATOS.GOB.ES” Identificativo único de la entidad que publica la información.
- **{carácter}**
“RECURSO” Componente obligatorio (y normalizado) definiendo la naturaleza de la información publicada.
- **{Sector}**
“SECTOR-PUBLICO” Es un componente opcional (y también normalizado) indicando el concepto específico para clasificar la información.
- **{Dominio}**
“AUTONOMIA” Componente opcional que identifica con más precisión el concepto.
- **{concepto}**
“ANDALUCÍA” Componente opcional para identificar de forma específica el recurso frente a otros similares.
- **.[ext]**
“JSON” Componente opcional que identifica el formato en el que se representa el documento apuntado.

Ejemplo

`http://datos.gob.es/recurso/sector-publico/territorio/Autonomia/Andalucia.json`

Se puede observar que ambas codificaciones son muy parecidas, pudiendo existir recursos que cumplan ambas especificaciones. Los elementos carácter, sector, dominio y concepto pueden provenir de vocabularios normalizados o parcialmente normalizados.

¿QUÉ PODEMOS HACER CON UNA API?

Una API nos permite interactuar con un sistema de información sin necesidad de un conocimiento de la estructura interna o de la tecnología utilizada.

¿QUÉ NECESITAMOS SABER PARA UTILIZAR UNA API?

Necesitamos conocer los mecanismos de uso de la API, para ello se pone disponible una documentación describiendo los mecanismos de acceso y parámetros para interactuar con la API.

Ejemplo

La API de Datos abiertos de Zaragoza [Fuente: https://www.zaragoza.es/ciudad/risp/ayuda-api.htm#PrámetrosAPI](https://www.zaragoza.es/ciudad/risp/ayuda-api.htm#PrámetrosAPI) es un ejemplo de una buena implementación de API, y puedes ver la descripción de su uso.

1.2 ¿Qué problemas resuelve una API?

- **Evolución de los sistemas:** Las tecnologías de construcción de software (lenguajes de programación, herramientas, etc.) evolucionan a gran velocidad; también las necesidades de los usuarios. Por ambas causas nuestros **sistemas de información están en constante evolución**.
- **Utilidad global:** Cuantos más sistemas de información existen, mayor es la posibilidad de **incrementar su utilidad mediante su conexión**. La API permite la conexión de multitud de sistemas sin tener que detener su evolución interna.
- **Desarrollo separado para cliente y servidor:** La definición de una API permite desarrollar por separado la parte del cliente (por ejemplo: el interfaz web o el interfaz para distintos dispositivos móviles) de una aplicación frente a la parte servidora de la aplicación.
- **Interacción tecnologías dispares:** La evolución de las tecnologías provoca la convivencia de sistemas con tecnologías muy dispares. La disponibilidad de una API permite que tales sistemas puedan interactuar y proporcionar servicios a los usuarios.
- **Interacción asíncrona:** Las APIs permiten la interacción asíncrona entre sistemas, normalmente gracias a **protocolos sin estado**².

² Los protocolos sin estado, en cada transacción, contienen toda la información necesaria para procesar cada petición y no dependen de transacciones anteriores o de estados anteriores del sistema.

REQUERIMIENTOS

Aprovechar las ventajas del uso de una API requiere a cambio:

PUBLICACIÓN DE SU ESPECIFICACIÓN

Es necesario que su especificación sea publicada, o que sea posible ser descubierta mediante protocolos al uso (por ejemplo: **WADL**³ [Fuente: https://www.w3.org/Submission/wadl/](https://www.w3.org/Submission/wadl/)).

HTTP es el protocolo más utilizado para las API sobre **TCP/IP**⁴.

DIMENSIONAR EL SERVIDOR

También requiere del dimensionamiento adecuado para atender dichas peticiones ya que la disponibilidad de una API pública podría permitir un buen número de conexiones simultáneas.

1.3 Servicios web y APIs REST

SERVICIOS WEB

Definición

Un Servicio Web es un tipo de mecanismo de interacción entre aplicaciones que normalmente opera a través de HTTP (el protocolo de la web), si bien se podrían utilizar otros protocolos (p.e. SMTP).

La definición de web service del **W3C**⁵ [Fuente: https://www.w3.org/TR/ws-gloss/](https://www.w3.org/TR/ws-gloss/) menciona que los Servicios Web típicamente:

- ✓ Utilizan como protocolo HTTP.
- ✓ Intercambian mensajes en formato **SOAP** (SOAP se codifica en el estándar XML).

+ MÁS INFORMACIÓN

SOAP, Simple Object Access Protocol, es un protocolo estándar que define cómo servicios en diferentes procesos pueden comunicarse por medio de intercambio de datos XML encapsulados. [En este enlace Fuente: https://www.w3.org/TR/2007/REC-soap12-part0-20070427/](https://www.w3.org/TR/2007/REC-soap12-part0-20070427/) puedes acceder a la última versión disponible.

³ WADL es el equivalente RESTful de WSDL.

⁴ Protocolo de Control de Transmisión / Internet Protocol, protocolos de comunicación de datos en Internet.

⁵ WORLD WID WEB Consotium, organización que vela por la definición y mantenimiento de estándares en Internet.

REST

Definición

Es un conjunto de principios de arquitectura para la construcción de interfaces entre sistemas. El término REST proviene del acrónimo inglés Representational State Transfer.

Características

- ✓ El uso de **HTTP** Fuente: https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol como protocolo de comunicación para obtener datos, o indicar la ejecución de operaciones sobre los datos (mediante un conjunto limitado de opciones como se describe a continuación), pero sin añadir niveles adicionales de abstracción (como **SOAP** Fuente: <https://es.wikipedia.org/wiki/SOA>).
- ✓ La transferencia de los datos se realiza, sobre todo utilizando formatos como **XML** Fuente: https://es.wikipedia.org/wiki/Extensible_Markup_Language, **JSON** Fuente: <https://es.wikipedia.org/wiki/JSON>, pero también podrían usarse CSV, u otros.
- ✓ Los sistemas que siguen los principios REST se llaman *RESTful*.

APIS REST: PRINCIPIOS CLAVE

El éxito de los protocolos REST se basa en cuatro principios clave:

PROTOCOLO

Un protocolo cliente/servidor sin estado

El protocolo utilizado http es un **protocolo sin estado**⁶, es decir, no recuerda pasadas interacciones. En caso necesario, sin embargo, la utilización de cookies y otros mecanismos nos permiten mantener la sesión.

Algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST. La ventaja del uso de protocolos sin estado es la no necesidad de almacenar el estado, lo que permite una gestión más liviana y, por tanto, aumenta la capacidad de procesar peticiones y su escalabilidad.


SINTAXIS

Una sintaxis uniforme en la red para identificar los recursos

Un sistema REST está orientado a la descripción de los recursos más que de acciones. Cada recurso es direccionable unívocamente a través de su URI, si bien

⁶ Los protocolos sin estado permiten que cada interacción sea independiente y que no haya que gestionar estados de solicitudes anteriores. Así cada solicitud y respuesta son independientes del resto y por tanto permiten la simplificación de sus mecanismos de gestión.

distintas URIs pueden apuntar al mismo recurso. Es decir, nuestro recurso puede tener varios 'alias' para enlazar.

Para más información sobre cómo codificar de forma eficiente URI puedes acceder al apartado 6.1.2 de la [Guía de aplicación de la Norma Técnica de Interoperabilidad: ... contenidos/descargas/GuiaNTI.pdf](#)  de reutilización de recursos de información del sector público.

OPERACIONES

Un conjunto de operaciones bien definidas que proporcionan un interfaz uniforme

Las operaciones más importantes de http son **POST, GET, PUT y DELETE**⁷ [Fuente: https://tools.ietf.org/html/rfc7235](#). Estas operaciones que se realizan sobre recursos (unívocamente identificados por URIs), son similares a las básicas de una base de datos relacional (creación, lectura, actualización y borrado).

HIPERMEDIOS

El uso de hipermedios (hiperenlaces)

Tanto para la información de la aplicación como para las transiciones de la aplicación. Su representación de este estado en un sistema REST se realiza típicamente en HTML o XML. Así es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

OPEN API

Open API es una especificación formal para desarrollar API REST en cualquier lenguaje de programación permitiendo describir, producir, consumir y visualizar servicios. Además:

- ✓ Es libre bajo licencia libre Apache 2.0.
- ✓ Open API está disponible en [github](#) [Fuente: https://github.com/OAI/OpenAPI-Specification](https://github.com/OAI/OpenAPI-Specification).

¿CUÁL ES SU PRINCIPAL VENTAJA?

A medida que se actualiza el código del servicio, lo hace la documentación de los métodos, parámetros y modelos permitiendo que API y su documentación estén siempre sincronizadas.

⁷ Especificación de seguridad de la autenticación http 1.1

¿QUÉ COMPONENTES TIENE SU ESPECIFICACIÓN?

- Listado de recursos (resource listing): devuelve un listado con todas las APIs a documentar/probar.
- *API Declaration*: devuelve la información relativa a una API, como sus operaciones o el modelo de datos.

+ MÁS INFORMACIÓN

Anteriormente era conocida como *Swagger RESTful API Documentation*. *Swagger* Fuente: <http://swagger.io/> es un ecosistema de herramientas, especificaciones y métodos alrededor de las API.

1.4 Web APIs Hoy

1.4.1 CULTURA MASHUP

Los *mashups* son **aplicaciones capaces de combinar contenidos de distintas fuentes**, normalmente otras aplicaciones. En la práctica se llama así a aquellos entornos que facilitan esa labor, y donde el usuario puede adaptar la mezcla de fuentes de información a sus necesidades mediante interfaces que no requieren ninguna capacitación del usuario, o bien ésta es limitada.

- **Mashup enablers**: Son pequeñas aplicaciones que convierten los resultados originales de las fuentes de datos en formatos compatibles para su mezcla en la plataforma de *mashup*. Con los *mashup* enablers se pueden crear **nuevos servicios** que aprovechan los datos y servicios originales.
- **Representación gráfica**: Normalmente en los mashups existe una representación gráfica de los resultados, aunque los resultados también podrían ser una lista (por ejemplo, un rss combinado de varias fuentes) en lugar de una visualización.
 - Algunos mashups avanzados permiten la interacción con las representaciones gráficas alcanzadas, en forma de zooms, filtros, capas, etc.
 - Los mashups comerciales son capaces de integrar distintas aplicaciones (ERP, CRM, etc), bases de datos relacionales y no relacionales ficheros de texto (-csv.,.txt, .xml), etc. **Otros ejemplos** Fuente: <http://www.programmableweb.com/category/all/mashups>.

Ejemplo

Un *mashup* básico puede verse [en este enlace](http://ogd.ifs.tuwien.ac.at/mashup/) Fuente: <http://ogd.ifs.tuwien.ac.at/mashup/> donde se combinan numerosas fuentes de datos abiertos y permite conectarlas con solo arrastrar y soltar desde los elementos de enlace (anclas) y representarlos de forma gráfica con cuatro visualizaciones: mapa, gráfico de tarta, de barras y de columnas.

1.4.2 ECONOMÍA DEL API

La masiva irrupción en el mercado de las **apps** y la necesidad de cierta personalización de los servicios, tanto por la variedad de dispositivo de visualización como por el tipo de usuarios, han incentivado la **generalización de las APIs en empresas** y en menor medida en servicios públicos.



Las compañías de cierta dimensión han descubierto que pueden captar clientes y/o usuarios adicionales si abren sus servicios (en forma de API) a desarrolladores independientes. Estos desarrolladores, utilizando los API disponibles, **crean servicios innovadores** mezclándolos con otros, aportando características propias, bien gratuitas o de pago.

Ejemplo

- ✓ Aunque hay numerosos ejemplos, uno bien documentado puede encontrarse en la **compañía creadora del acortador de URL bit.ly** Fuente: http://dev.bitly.com/get_started.html (bitly Developer).
- ✓ En España, un buen ejemplo lo tenemos en las **API de BBVA** Fuente: https://www.bbvaapimarket.com/web/api_market/.

1.4.3 APIS EN REPOSITORIOS DE DATOS

Los repositorios de datos abiertos en su afán de dar difusión a sus contenidos abrazaron pronto las APIs en sus **plataformas**. Una de las más populares es **CKAN** Fuente: <http://docs.ckan.org/en/latest/api/> (*Comprehensive Knowledge Archive Network*).

¿QUÉ ES CKAN?

Es un software de código abierto para el almacenamiento y distribución de conjuntos de datos (*datasets*) de cualquier tipo.

El código fuente está disponible en [github](https://github.com/ckan/ckan) Fuente: <https://github.com/ckan/ckan> bajo la licencia GNU Affero General Public License (AGPL) v3.0. Es usado de forma casi mayoritaria en Europa y mantienen una [web con implementaciones](http://ckan.org/instances/) Fuente: <http://ckan.org/instances/> en diversos países.

ORIGEN

Creada por la Open Knowledge Foundation (ahora [Open Knowledge](https://okfn.org/) Fuente: <https://okfn.org/>), cuenta con una API abierta para la recuperación de los datos (aunque también es posible recuperar los metadatos).

Algunas de las ventajas para los repositorios de información (de datos abiertos) del uso de API son:

- ✓ Mejora los flujos de información internos puesto que obliga a normalizarlos para poder responder a las interacciones normalizadas del API.
- ✓ Facilitan la cooperación entre entidades ya que están definidos los mecanismos de recuperación de información y de interacción.
- ✓ Facilita el enriquecimiento semántico de los datos.
- ✓ Estandariza el acceso a los datos.
- ✓ Facilita la federación de contenidos entre repositorios posibilitando así el aumento de la difusión de contenidos.

+ MÁS INFORMACIÓN

La interacción de las APIs permite la generación de nuevas posibilidades más allá de las inicialmente proporcionadas por cada una de las entidades publicadoras.

1.4.4 APIS GUBERNAMENTALES

A continuación, presentamos algunos ejemplos de APIs gubernamentales.

- **API ZARAGOZA:** Esta API [Fuente: https://www.zaragoza.es/ciudad/risp/api.htm](https://www.zaragoza.es/ciudad/risp/api.htm) proporciona interacción con tres servicios de la ciudad:
 1. API mapas colaborativos: Permite la búsqueda y el acceso al detalle en los mapas, crear y modificar mapas de usuario, utilizar los puntos de interés de las bases de datos de la ciudad y el borrado tanto de mapas de usuarios como de sus puntos de interés en mapas.
 2. API quejas y sugerencias: Permite el listado de servicios y el alta de una queja, su listado y el acceso a su detalle. Responde con datos en JSON y XML.
 3. API cita previa: Permite el acceso al listado de agendas disponibles, a su detalle y al estado de las citas de cada agenda. También permite el alta, consulta y cancelación de citas.

Existe, además, un **entorno de pruebas** [Fuente: https://apex-test.zaragoza.es/](https://apex-test.zaragoza.es/) para los desarrolladores de las API.

- **API DATA.GOV:** Está basada en la plataforma CKAN y además agrega datos de multitud de APIs provenientes de las distintas agencias del gobierno federal de Estados Unidos, contiene cerca de 200.000 juegos de datos, casi 10.000 de ellos **accesibles vía API** [Fuente: http://catalog.data.gov/dataset?q=-aapi+api+OR++res_format%3Aapi#topic=developers_navigation](http://catalog.data.gov/dataset?q=-aapi+api+OR++res_format%3Aapi#topic=developers_navigation). Puedes acceder a **esta API aquí** [Fuente: http://catalog.data.gov/dataset](http://catalog.data.gov/dataset).
- **OTROS:** Un listado con más de 14.000 APIs de todo tipo, tanto públicas como privadas puede encontrarse en <http://www.programmableweb.com/>.

2. CONSIDERACIONES PARA EL DISEÑO DE APIS WEB

El diseño de APIs web ayudará a optimizar nuestros esfuerzos para fomentar la reutilización de nuestras informaciones y el enriquecimiento de otras. Para ello vamos a ver:

- **Modelo de datos:** Conoceremos el protocolo HTTP y buenas prácticas para el diseño de APIs.
- **Parámetros de búsqueda:** Cómo filtrar los resultados para optimizar las comunicaciones de nuestro servidor.
- **Formato y paginación:** Qué hacer para limitar la cantidad de resultados al usuario.

- **Protocolo y seguridad:** Recomendaciones de acceso y cifrado que hagan segura la transmisión de datos.

2.1 Modelo de datos: recursos y relaciones

De forma previa, y para facilitar la comprensión de los contenidos se incluyen a continuación algunos de los verbos del protocolo http, que responden a una variedad de acciones y respuestas estandarizadas y flexibles con las que poder diseñar nuestro sistema.

FUNCIONES

En sus funciones principales encontramos las acciones básicas con datos y las gestiones propias de la conexión. Veamos las principales:

HEAD

Solicita la cabecera de la respuesta. Permite obtener los metadatos de entidades sin solicitar la entidad propiamente. La respuesta es idéntica a la de una petición GET.

GET

Solicita el recurso especificado. La petición puede ser simple o compuesta por parámetros.

POST

Envía un recurso. Esta petición suele usarse tanto para crear como para actualizar contenidos.

PUT

Sube o carga un recurso especificado.

DELETE

Solicita la eliminación del recurso especificado.

API

OPTIONS

Solicitamos las opciones que el servidor soporta para un URL específico.

+ MÁS INFORMACIÓN

En España la [Guía de aplicación de la normativa técnica de interoperabilidad: ...contenidos/descargas/GuiaNTI.pdf](#) (en su apartado 6.1.2) se referencia la forma de crear URIs que puedan ser localizables mediante protocolo http.

CLASIFICACIÓN

Los métodos http se clasifican según dos criterios:

SEGURIDAD

Métodos http seguros: Un método es seguro si no produce efectos sobre el servidor. Normalmente solo los métodos de lectura tienen esta propiedad (**GET, HEAD y OPTIONS son seguros**).

IDEMPOTENCIA

Métodos http idempotentes: Un método es idempotente si la ejecución repetida del mismo (con los mismos parámetros) produce el mismo efecto que cuando se ejecuta una única vez (**todos son idempotentes menos POST**).

2.1.1 OPERACIONES

Nuestra API web debe implementar al menos **CRUD (creación, lectura, actualización y borrado de registros)** siempre bajo las **credenciales y permisos adecuados**.

LECTURA

Consideraciones en lectura

- ✓ **Estructuración de los recursos. Creación de recursos hijos:** Cuando los recursos disponibles consten de colecciones muy numerosas o su número de campos asociados sea muy alto, se puede considerar la segregación de algunos grupos de campos en entidades hijas de la primera, para facilitar el acceso a los datos más frecuentes y así limitar la demanda de tráfico y capacidad de cómputo por campos que rara vez se utilizan.
- ✓ **Estructuración de los recursos. Contenidos extensos:** En el caso de que el problema sea que los contenidos de un campo sean muy extensos (p.e. el texto completo de una normativa) lo más adecuado puede ser dividirlo en capítulos. Esta técnica es conocida como **slicing**⁸.
- ✓ **Respuesta a petición de recurso:** Nuestra API debe elegir la respuesta ante una petición de acceso directo a un recurso:
 - Responder con la lista (únicamente con las URI de los elementos de ese tipo de recurso).
 - Responder con un listado completo de los datos básicos de la colección de elementos del recurso.

⁸ Tanto la creación de recursos hijos como el slicing pueden ser combinados para casos de contenidos con gran extensión, si bien esto introduce complejidad en la sintaxis del API y además genera URLs poco amigables para los buscadores, lo cual reducirá nuestro impacto.

ACTUALIZACIÓN Y BORRADO

Consideración en actualización y borrado

De forma genérica una petición del tipo PUT <http://our.domain.com/api/vi/recurso> (sin parámetros, podría ser entendida como una actualización a vacío) debería afectar a toda la colección de elementos de ese recurso, al poder estar vacía la cadena. Esto podría, en la práctica, ser equivalente a un borrado de toda la colección. Nuestro diseño debería incluir restricciones adicionales ante este tipo de métodos. De la misma manera aplicaría este principio a la opción DELETE.

CREACIÓN DE RECURSOS

Consideraciones en creación de recursos

Se podría utilizar POST para la creación de un elemento nuevo de una colección, si bien al ser un método no idempotente, en caso de problemas en la comunicación generaría instancias repetidas del elemento. Es más recomendable el uso de PUT en estos casos.

2.1.2 BUENAS PRÁCTICAS

A continuación y en base a una serie de criterios, vemos las buenas prácticas para el diseño de APIs:

1. Nomenclatura de las operaciones: uso de nombres y no de verbos.

Es una buena práctica para el diseño de API **no incluir verbos⁹ (pag 29)** en los literales de la URI sino nombres que identifican a los recursos, y que sean las operaciones utilizadas las que definan la interacción con los datos sobre los recursos (GET, POST, PUT y DELETE) y así podríamos utilizar la siguiente tabla.

RECURSO	GET	POST	PUT	DELETE
Acción genérica	Leer	Crear	Actualizar	Borrar
Ejemplo / Recurso	Proporciona listado de recursos	Crea un nuevo recurso	Actualiza un recurso	Borrar el recurso
Recurso / NNN	Lista el recurso NNN	No permitido	Actualiza el recurso NNN	Borrar el recurso NNN

! NNN representa un número identificando un elemento de la colección de un tipo de recurso

⁹ GET <http://our.domain.com/api/v1/recurso32/ActualizarNombreLiteral-Nombre>

2. No usar GET para cambios de estado de un recurso

Las modificaciones de estado de un recurso deben ir asociadas a POST / PUT o DELETE Y no deberían ser asociadas a GET /recurso/modificacion-de-estado o GET /recurso/NNN/modificacion-de-estado

Por ejemplo, es una mala práctica algo como esto:

GET `http://our.domain.com/api/v1/recurso32/ActualizarNombre Literal-Nombre`

Mejor:

PUT `http://our.domain.com/api/v1/recurso32?campo=nombre Literal-Nombre`

3. Usar singular o plural pero solo uno de ellos

Ser consistente en la nomenclatura de acceso a los recursos.

4. Codificar las relaciones como sub-recursos

Si dos recursos tienen relación entre ellos se pueden utilizar, salvo el caso N a N. **GET/recurso/NNN/subrecurso** para acceder a todos los subrecursos del recurso NNN y de forma similar al primer nivel **GET/recurso/NNN/subrecurso/MMM**

P. ej., **GET `http://our.domain.com/api/v1/recurso/32/subrecurso/3`**

nos daría todos los campos del subrecurso 3 del recurso 32) Para acceder al subrecurso MMM del recurso NNN. NNN y MMM son los identificadores numéricos del recurso y del subrecurso respectivamente.

5. Usa cabeceras http para especificar el formato

Incluye en el http-header el formato del contenido. La lista de formatos puede encontrarse en el [Directorio oficial de tipos mime](http://www.iana.org/assignments/media-types/media-types.xhtml) Fuente: <http://www.iana.org/assignments/media-types/media-types.xhtml>.

6. Proporciona capacidades de filtrado, ordenado y acceso a información de cada campo

En el apartado 2.3 haremos una descripción más detallada.

7. Incluye la versión de la API dentro de la URL

Por ejemplo **`http://our.domain.com/api/v1/recurso`** para acceder a la primera versión del API y **`http://our.domain.com/api/v2/recurso`** para acceder a la versión 2 del API. De esta manera no se generarán conflictos entre versiones.

8. Permitir el método http overriding

Algunos *proxies* no permiten conexiones que no sean POST o GET, usando el método X-HTTP-Method-Override podremos extender nuestra capacidad de interacción a las operaciones PUT y DELETE.

9. Permitir que el usuario elija el formato de salida

No deben limitarse las capacidades de desarrollo de los usuarios y será necesario ofrecer opciones, como mínimo que incluyan formatos de salida JSON y XML. P. e.: **http://our.domain.com/api/v2/recurso/output=XML**

10. Permitir operaciones de prueba

Incluye facilidades para que los desarrolladores realicen pruebas que no afecten a los datos reales, bien mediante entornos de demo o bien mediante opciones que limiten el alcance de las interacciones.

DELETE http://our.domain.com/api/v2/recurso?test=true

En este caso la operación no borraría el dato pero respondería si el dato existe para ser borrado y si se tienen las credenciales necesarias para realizarlo.

2.1.3 CÓDIGOS HTTP PARA DEFINIR ERRORES

Los errores http de nuestra API son una fuente importante de información para los desarrolladores que la utilicen. Vamos a ver los más relevantes/frecuentes:

2xx: *Peticiones correctas*

Conjunto de respuestas para indicar la recepción, comprensión y aceptación de peticiones.

- ✓ **200 OK.** Respuesta correcta. Indica que el servidor ha recibido, entendido y aceptado la petición.
- ✓ **201 Created.** La petición ha sido aceptada y ha resultado en la creación de un nuevo recurso.
- ✓ **202 Accepted.** La petición ha sido aceptada para procesarse pero aún no se ha completado.
- ✓ **204.** El recurso ha sido borrado con éxito.

3xx: *Redirecciones*

Esta categoría permite informar de que la petición ha de concretarse en algún sentido o se atiende desde otra ubicación.

- ✓ **300 Multiple choices.** La petición corresponde a cualquier elemento de un

conjunto y debe especificarse uno en concreto. La respuesta ha de incluir las posibilidades.

- ✓ **304 Not Modified.** Especifica que el documento solicitado no ha variado con respecto al solicitado.

4xx: Errores del cliente

Esta categoría permite informar de peticiones incorrectas por parte del cliente.

- ✓ **401 Unauthorized.** El recurso solicitado requiere autorización.
- ✓ **403 Forbidden.** Prohibido. La petición se ha entendido pero no hay autorización para atenderla.
- ✓ **404 Not Found.** El servidor no encuentra lo solicitado en la petición.

5xx: Errores internos del servidor

Esta categoría permite indicar cuando el servidor no responde satisfactoriamente las peticiones y es consciente de ello. Irán acompañados de una descripción del error y si éste es de tipo temporal o permanente.

- ✓ **500 Internal Server Error.** El servidor encontró algo inesperado que le impidió resolver la petición.

¡! ATENCIÓN: CONSEJO

No debemos dejar estas páginas en su versión por defecto sino que en la medida de lo posible debemos proporcionar información suficiente para que nuestros usuarios sepan qué está ocurriendo.

Formato de respuesta para una petición de error

Una buena práctica de codificación (JSON) de una respuesta ante errores podría ser:

```
{
  "error": [{
    "MensajeParaUsuario": " Lo sentimos ese recurso no existe",
    "MensajeInterno": " No encontrado ese recurso en la base de datos",
    "code": 22,
    "mas información": "http://our.domain.com/blog/api/v1/error/22"
  }]
}
```

El primer mensaje será para que el usuario pueda entender la causa del error, el segundo para depurar nuestro interfaz y a mejorar las guías de uso, y el enlace un recurso informativo para el usuario.

2.1.4 ALGUNAS PRÁCTICAS A EVITAR

La utilización de extensiones para distintos formatos de los datos.

- ✓ Hay que tener en cuenta que un API puede devolver un mismo juego de datos en distintos formatos (json, XML, csv, xls, etc). Aunque es posible codificar API con una nomenclatura como las siguientes, en general URL del tipo:

http://our.comain.com/api/v1/recurso23.json

http://our.comain.com/api/v1/recurso23.xml

- ✓ Más importante que la extensión es incluir en la cabecera el **mime-type** Fuente: <http://www.iana.org/assignments/media-types/media-types.xhtml>, como se especifica en el ejemplo siguiente:

```
GET/api/v1/recurso/23 HTTP/1.1
Host: our.comain.com
Accept: application/json
```

De esta manera la herramienta tendrá perfecto conocimiento del tipo de aplicación a utilizar para procesar la información.

2.2 Parámetros de búsqueda para el filtrado de resultados

Vemos las principales opciones que pueden ayudarnos a seleccionar la información:

BÚSQUEDA

La posibilidad de filtrado de los resultados reducirá la demanda de comunicaciones de nuestro servidor (a costa de aumentar ligeramente las necesidades de proceso). Unas opciones mínimas podrían ser las siguientes:

GET http://our.domain.com/api/v1/recurso?parametro=valor

incluye las posibilidades de devolver la información filtrada por el valor de la propiedad del recurso parámetro (similar al campo de la base de datos), o con alguna lógica de filtrado mayor ofrecida por el uso de algunos operadores lógicos:

GET http://our.domain.com/api/v1/recurso?parametro>valor

ORDENACIÓN

También es recomendable que las peticiones puedan ser devueltas conforme a cierto orden de la forma:

GET http://our.domain.com/api/v1/recurso?sort=parametro1,-parametro2

donde parametro1 y parametro2 son propiedades del recurso (campos de la base de datos) y ordenaría primero de forma ascendente por la propiedad 1 (parámetro1) y de forma descendente por la propiedad2 (parámetro2).

SELECCIÓN DE CAMPOS

Finalmente, de cara al filtrado de resultados, puede ser interesante implementar opciones que permitan obtener sólo parte de las propiedades del recurso (campos de la base de datos)

GET

http://our.domain.com/api/v1/recurso?fields=parametro1,parametro3,parametro5

donde parámetro1, parámetro3 y parámetro5 son propiedades del recurso, como si fueran campos de una base de datos.

2.3 Formato y paginación de resultados

Nuestra API debe proporcionar facilidades para que el usuario limite la cantidad de resultados. Una buena práctica es incluir un límite por defecto (por ejemplo 50) salvo que el usuario establezca otro límite, sea éste mayor o menor.

Buenas prácticas de codificación de API.

- ✓ Para descartar los 10 primeros resultados y mostrar los 50 primeros (porque aplicaría el valor por defecto) si los hubiere:

GET http://our.domain.com/api/v1/recurso?offset=10

- ✓ Para conseguir un listado de todos los resultados a partir del 10 podría incluirse:

GET http://our.domain.com/api/v1/recurso?offset=10&limit=-1

- ✓ Para enlazar a la siguiente página de resultados o a la previa, deberíamos proporcionar la información en el header de la comunicación. Por ejemplo:

```
<https://our.domain.com/api/v1/recurso?offset=15&limit=5>; rel="next",  
<https://our.domain.com/api/v1/recurso?offset=50&limit=3>; rel="last",  
<https://our.domain.com/api/v1/recurso?offset=0&limit=5>; rel="first",  
<https://our.domain.com/api/v1/recurso?offset=5&limit=5>; rel="prev"
```

¡! IMPORTANTE

Una alternativa que podría simplificar la programación a los utilizadores de nuestra API sería generar las distintas páginas como recursos independientes, que de esta manera podrían ser descubiertos directamente.

2.4 Protocolo y seguridad: credenciales de acceso y cifrado

La seguridad es siempre una necesidad, y por ello las principales recomendaciones son:

- ✓ La transmisión de datos será https por defecto.
- ✓ Exigir credenciales de seguridad únicamente para aquellas operaciones que produzcan modificaciones como: creación de recursos, borrado o modificación de un registro.

La autenticación en http es un protocolo desafío/respuesta en el que deben proporcionarse las credenciales adecuadas para acceder a los recursos restringidos.

Existen dos mecanismos de autenticación:

- **Basic:** Este mecanismo lo único que hace es codificar en base 64 la cadena compuesta por “usuario:contraseña”, por lo que su transmisión debe realizarse mediante protocolo https pues, en caso contrario, cualquiera que detectara la transmisión sólo tendría que descodificar y acceder a las credenciales.
- **Digest:** Este mecanismo lo que transmite es un hash compuesto por: el usuario, la password y un elemento de un solo uso.



+ MÁS INFORMACIÓN

Peticiones de recursos protegidos bajo protocolo OAuth

La autenticación por protocolo **OAuth** Fuente: <http://oauth.net/> genera a partir de un token de acceso una *consumer key* y *consumer token* para el acceso. No forma parte de http.

3. LINKED DATA

3.1 De los datos abiertos a los datos enlazados

La publicación de datos con licencia abierta de forma que sean reutilizables por otros organismos o personas es denominado **Open data** Fuente: <http://opendatahandbook.org/guide/es/what-is-open-data/>. Es una tendencia muy activa entre administraciones públicas, y progresivamente en el sector privado y no gubernamental.

Por otra parte, los **Datos enlazados** o *Linked data* va a ser, posiblemente, uno de los conceptos más importantes relacionados con la gestión de la información, tanto pública como privada, en los próximos años. Vamos a ver la relación entre ambos términos (puedes ampliar la información sobre estos términos en la unidad *Conceptos básicos*):



La publicación de datos con licencia abierta de forma que sean reutilizables por otros organismos o personas es denominado Open data.

El enriquecimiento de los datos abiertos con **metadatos** permite una explotación más avanzada de los mismos.

Cuando estos metadatos cumplen los requisitos de la web semántica se denominan datos enlazados o **Linked data**.

+ MÁS INFORMACIÓN

Enriqueciendo el *open data* para llegar a *Linked data*

El enriquecimiento de los datos hasta convertirlos en *Linked data* supone un incremento de coste y requiere de conocimientos adicionales. Pero las ventajas que proporciona, hace que iniciativas de *open data* vayan paulatinamente enriqueciendo semánticamente, inicialmente sus catálogos, y más adelante sus datos.

PRINCIPIO DE LOS DATOS ENLAZADOS

Tim Berners-Lee estableció **4 principios** para *Linked Data*:

- ✓ **Usar URIs como nombres para las cosas:** el uso de URI evita ambigüedades y ofrece una forma normalizada e inequívoca para identificar cualquier recurso.
- ✓ **Usar URIs HTTP:** al usar URI compatibles con http los usuarios podrían buscar y localizar esos nombres en la web.
- ✓ **Utilizar estándares semánticos:** cuando alguien busque empleando un URI, proporcionar información útil, utilizando estándares (RDF Fuente: <https://www.w3.org/RDF/>, SPARQL).
- ✓ **Incluir enlaces a otros URIs:** de igual manera que en las páginas web se incluyen hiperenlaces para conectar con otras páginas, el uso de enlaces a otras URI facilita el uso de los datos enlazados

3.2 Metadatos: de formatos estructurados a formatos semánticos

La especificación de HTML permite incluir metadatos en las páginas web principalmente a través de las etiquetas *meta* o también puede hacerse a través de la etiqueta *link* (metadatos remotos).

Aunque el estándar HTML no precisa de etiquetas *meta* (su utilización es un tema de consenso), su uso está normalizado debido a que, en la web semántica, los metadatos tienen una función precisa de localización, identificación y descripción de recursos.

¿QUÉ ES EL CISE?

El Centro de Transferencia de Tecnología del Ministerio de Hacienda y Administraciones Públicas de España contiene el **Centro de Interoperabilidad Semántica (CISE)** Fuente: <http://cise.redsara.es/SGAS/VisorAction.action> que mantiene modelos de datos normalizados que se publican en el gestor de activos semánticos.

Conoce más sobre el CISE:

- **Finalidad:** La finalidad del CISE es fomentar la interoperabilidad semántica entre los organismos públicos.
- **Ramas principales:** A marzo de 2016 mantienen cinco ramas principales: Conferencia de Rectores de las Universidades Españolas, Proyecto Indalo (precedente del CISE), Instituto Nacional de Estadística, Ministerio de Educación, cultura y deportes y Ministerio de Hacienda y Administraciones Públicas.
- **Activos semánticos:** Contiene unos 600 activos semánticos recogidos que cubren áreas financieras, de recursos humanos, de registro de documentos, de

comunicaciones electrónicas, etc.

- **Ejemplo:** Un ejemplo de estos activos semánticos es la **norma SICRES** [Fuente: http://administracionelectronica.gob.es/ctt/sicres](http://administracionelectronica.gob.es/ctt/sicres) para el intercambio de información entre registros y que cuenta con **guía de aplicación:** ...contenidos/descargas/GuiaSICRES.pdf.

3.3 Conceptos básicos de la web semántica

TRIPLETAS

Formalmente una tripleta (o triple) es una sentencia RDF en la que se describe la relación de un recurso con otro a través de un sujeto, un predicado (o propiedad), y un objeto. Generalizando:

sujeto¹⁰ → predicado¹¹ → objeto¹²

Una tripleta puede ser de los siguientes tipos:

- **Una relación entre dos recursos en la Web.**

Ejemplo: Coche → es un → Vehículo

Tanto coche como vehículo identifican recursos y el predicado establece una relación entre ellos, en este caso de pertenencia de uno al tipo del otro. Una posible redacción de esta tripleta en formato RDF sería

```
<rdf:Descriptionrdf:about "coche">
    <predicaterdf:resource="Vehiculo"/>
</rdf:Description>
```

- **La asignación de un valor a un recurso.**

Ejemplo: Coche → es de color → Verde

En este caso, el coche es un recurso y el predicado establece una relación de atribución al recurso, especificando su color (una de las propiedades del primer objeto). Y su posible redacción

¹⁰ El sujeto siempre será un recurso.

¹¹ Un predicado establece una relación o una propiedad (atributo) del sujeto.

¹² Un objeto puede ser otro recurso relacionado o bien un valor de ese atributo referido al sujeto (un literal).

```
<rdf:Descriptionrdf:about="Coche">  
    <predicate>verde</predicate>  
</rdf:Description>
```

GRAFO

Un grafo semántico no es más que una relación entre tripletas.

Ejemplo: Un grafo creado basado en las tripletas anteriores sería conocer que hay vehículos de color verde.

Algunos Vehículos → son de color → Verde

Este hecho permite la creación de conocimiento que previamente no ha sido definido formalmente.

ONTOLOGÍAS

Una **Ontología**¹³ es una especificación consensuada que describe un dominio de información. En el caso concreto de la web semántica, esto se realiza por medio de tripletas y resulta en grafos que se definen por medio de lenguajes de descripción de ontologías como **OWL**¹⁴ Fuente: <https://www.w3.org/2001/sw/wiki/OWL> y **RDF Schema**¹⁵ Fuente: <https://www.w3.org/TR/rdf-schema/>.

SKOS

Simple Knowledge Organization System

SKOS¹⁶ Fuente: <https://www.w3.org/TR/skos-reference/> es una iniciativa del W3C, que modeliza la estructura y el contenido de ámbitos de conocimiento codificada en RDF pero con **menores necesidades**¹⁷ de rigor y lógica que una ontología. Actúa como puente entre la falta de conceptualización de la web habitual y el formalismo de las ontologías codificadas en OWL.

¹³ Definición semántica forma de ontología: una 'ontología' es una definición formal de tipos, propiedades, y relaciones entre entidades que realmente o fundamentalmente existen para un dominio de discusión en particular.

¹⁴ Ontology Web Language, especificación del W3C.

¹⁵ Especificación de RDF Schema.

¹⁶ Referencia oficial de SKOS en la web de W3C <http://skos.um.es/TR/skos-primer/>

¹⁷ Permite el establecimiento de distintos tipos de relaciones e identifica los conceptos mediante URIs, permitiendo que estas puedan incluir etiquetas con traducciones en varios idiomas y distintos tipos de anotaciones. Permite también crear colecciones ordenadas así como agrupaciones de conceptos. También permite la organización jerárquica de conceptos.

3.4 ¿Qué problemas resuelve la web semántica?

En la web, los datos normalmente están entremezclados en archivos HTML, que para el uso humano limitado puede ser suficiente. El problema con la mayoría de los datos en este formato es que se hace difícil, sino imposible, su **uso a gran escala**¹⁸.

La **web semántica**, o mejor, “web de los datos”, busca añadir la información de los conceptos, de forma que, tanto humanos como programas automatizados puedan beneficiarse de la publicación en formato web, y donde la interconexión de las páginas enriquezca los contenidos.

3.4.1 VENTAJAS DE LA WEB SEMÁNTICA

- **Búsquedas más eficaces:** Actualmente, los buscadores están focalizados en la comparación de cadenas de texto y sólo con tecnología adicional son capaces de incorporar conceptos cercanos basados en el texto. La web semántica permitiría la búsqueda por conceptos próximos de forma nativa y certera.

Por ejemplo: Al buscar casa actualmente, con estas tecnologías adicionales, también se puede buscar el concepto hogar, pero también edificio, restaurante, etc. que pueden ser conceptos alejados del inicial de nuestro interés.

- **Ayuda a la creación y evaluación de conocimiento:** El enlace semántico entre los distintos recursos permite la identificación de relaciones desconocidas que, gracias a la codificación semántica, son reveladas. Así mismo, permitiría conocer con mayor precisión la popularidad de conceptos y su impacto en cada ámbito.
- **Eliminación de la publicidad tal y como la conocemos:** Dado que los conceptos estarían enlazados, la dispersión entre páginas que nos ofrecen publicidad como parte de sus contenidos podría ser evitada. Por tanto la publicidad se vería abocada a una transformación radical, al menos obligaría a reformularla.

3.4.2 BARRERAS DE LA WEB SEMÁNTICA

La web semántica desde su nacimiento hace más de quince años ha superado algunas barreras para su difusión y aún ha de superar otras, entre las que podemos citar:

- **Complejidad técnica:** Actualmente, la adición de información semántica en la elaboración de una web requiere de herramientas algo más complejas que las habituales por lo que, para su éxito deberían existir incentivos. La indexación

¹⁸ Para recuperar esta multitud de información disponible presentada en numerosos sitios (en HTML) y volver a convertirla en estructurada existe una tecnología denominada scraping. Puedes consultar información sobre la misma en la unidad Herramientas de tratamiento de datos.

preferente por los grandes buscadores de los sitios con contenidos enriquecidos con información semántica comienza a incentivar de forma masiva su adopción.

- **Heterogeneidad de tecnologías:** Hasta llegar al momento actual, ha habido un largo camino en la elaboración y divulgación de los diversos estándares que componen la web semántica, desde el propio protocolo http a RDF, SKOS, etc. La necesidad de cubrir todos los puntos y la falta de consolidación de alguno de estos estándares ha frenado en el pasado su avance.
- **Capacidad de cómputo y de transmisión:** La inclusión de tecnologías semánticas obliga a una mayor capacidad de procesamiento de datos. La progresiva mejora de los sistemas y de las capacidades de transmisión de datos prácticamente han eliminado esta barrera.
- **Categorización de conceptos:** La disponibilidad de ontologías en todos los ámbitos crece de forma regular y, para usuarios normales, ya no supone una barrera, sin embargo en algunos ámbitos todavía no existen ontologías consensuadas.

3.5 Herramientas para el uso de *Linked data*

VIRTUOSO UNIVERSAL SERVER

Es un *middleware* y base de datos que combina la funcionalidad de un sistema de base de datos relacional y de orientación a objetos base de datos virtual, RDF, XML, texto libre, servidor web de aplicaciones y funcionalidad de servidor de ficheros en un único sistema. De ahí el término de universal, cuando sería más apropiado decir integrado.

Existen dos versiones, una de código abierto bajo licencia GPL 2, (OpenLink Virtuoso) que no incluye algunas características de la versión en software privativo. El fabricante es OpenLink Software. La versión libre se puede descargar [en este enlace de la web de Open link](http://virtuoso.openlinksw.com/download/) Fuente: <http://virtuoso.openlinksw.com/download/>.

ELDA

Es una implementación de una API utilizando datos *linked data*. Es open source con [código en github](https://github.com/epimorphics/Elda) Fuente: <https://github.com/epimorphics/Elda> bajo licencia apache 2.0 y soportado por la compañía Epimorphics. Permite recuperar datos de forma nativa en 5 formatos con diversa carga semántica (rdf/xml, json, turtle, xml, y html).

GRAPHDB

Es un repositorio de almacenamiento de datos semánticos. Incluye también motor de inferencia y servidor de consultas SPARQL. Existe una versión gratuita para su descarga.

PUBBY

Es una herramienta para generar interfaces linked data a endpoints SPARQL. Aunque su desarrollo puede estar un poco parado (más de dos años) su código **está disponible en github** [Fuente: https://github.com/cyqri/pubby](https://github.com/cyqri/pubby). Está implementada como una aplicación web java que debe ser ejecutada sobre un servidor de servlets como Tomat o Jetty.

LA PLATAFORMA D2RQ

Es un sistema de acceso a bases de datos relacionales como grafos RDF de solo lectura. Ofrece acceso basado en RDF al contenido de las bases de datos relacionales sin tener que replicar en RDF la base de datos original. E incluso permite la creación de volcados personalizados en RDF accesibles media la API de Apache Jena. Es de código abierto con licencia Apache 2.0. El código fuente de D2QR **está disponible en GitHub** [Fuente: https://github.com/d2rq/d2rq](https://github.com/d2rq/d2rq).

VIZIQUER

Es un software para la construcción gráfica de consultas SPARQL a través del esquema de datos que se puede cargar desde el archivo de la ontología o mediante las obtenidas en la exploración de un SPARQL endpoint. ViziQuer es compatible con la formulación de consulta sencilla y agregados.

PUBLISH MY DATA

Plataforma (de pago) [Fuente: http://www.swirrl.com/](http://www.swirrl.com/) para la publicación en modo servicio de datos *Linked data*.

+ MÁS INFORMACIÓN

W3C mantiene una lista de herramientas Linked data [Fuente: https://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/SemWebClients](https://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/SemWebClients).

3.6 SPARQL: lenguaje de consulta de fuentes *linked data*

SPARQL es un lenguaje de consulta sobre datos estructurados en formato RDF. Es decir, un lenguaje de consulta semántica de bases de datos, capaz de **recuperar y manipular los datos y relaciones** que hayan sido almacenados en ese formato.

3.6.1 IMPLEMENTACIONES Y HERRAMIENTAS

IMPLEMENTACIONES

Existen implementaciones para varios lenguajes de programación y herramientas que permiten conectar y construir una consulta SPARQL para un punto de consulta

(endpoint) SPARQL de forma semi-automática (por ejemplo ViziQuer). Además, existen herramientas que traducen consultas SPARQL a otros lenguajes de interrogación por ejemplo a SQL y Xquery.

Para implementar nuestro sistema es importante que admitamos un lenguaje de consulta semánticos, como SPARQL. Nuestro repositorio de información puede utilizar bases de datos orientadas a grafos (**Neo4j** Fuente: <http://neo4j.com/>, GraphDB, **Infogrid** Fuente: <http://infogrid.org/trac/>, etc) o por medio de un tradicional SGBD relacional dotado de una adaptación para utilización de información semántica (orientación a grafos).

HERRAMIENTAS PARA TRANSFORMACIÓN

Así mismo, existen varios sistemas o herramientas capaces de transformar los datos disponibles en formatos estructurados tales como bases de datos relacionales o .csv a formatos semánticos. Estas herramientas suelen permitir tanto transformaciones automatizadas como la definición de reglas para su transformación.

3.6.2 CONSULTAS

El elemento principal de SPARQL es, obviamente, la **consulta**, que consta de dos secciones claramente diferenciadas:

- **Encabezado:** El encabezado de la consulta, donde estableceremos qué tipo de información queremos recuperar, se identifica con la palabra clave **SELECT**. Irá seguida de las variables y tipos que queramos proyectar así como, de operadores que establecen el formato en el que obtendremos la información.
- **Patrón:** El patrón de consulta es donde estableceremos las restricciones o características que ha de cumplir la información que queremos recuperar. El patrón se estructura mediante tripletas que establecen las condiciones o características además de operadores que establecen las relaciones entre esas tripletas. Se identifica por seguir a la palabra clave **WHERE**.

Las posibilidades de SPARQL permiten realizar consultas sobre varias fuentes, de manera que podamos **relacionar datos e informaciones** y realizar inferencias sobre y con estos resultados. Esto permite la generación de nuevo conocimiento lo que genera un enorme potencial, limitado solo por la imaginación y la capacidad de cómputo.

Puedes ampliar información sobre SPARQL visitando [esta página](https://www.w3.org/TR/rdf-sparql-query/) Fuente: <https://www.w3.org/TR/rdf-sparql-query/>.

3.7 Datos enlazados hoy: ejemplos de proyectos institucionales relevantes

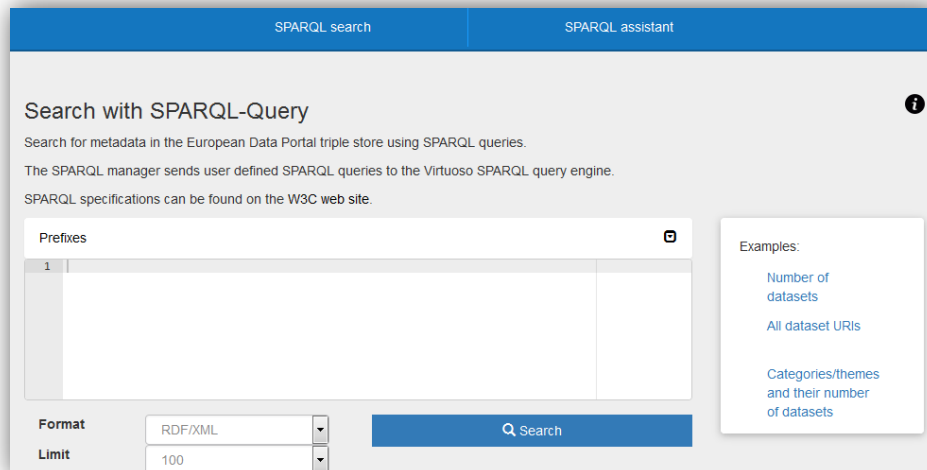
PORTAL PANEUROPEO DE DATOS Y EUROSTAT

Portal paneuropeo de datos

WEB Fuente: <http://www.europeandataportal.eu/sparql-manager/en/>

DCAT Fuente: <https://www.w3.org/TR/vocab-dcat/>

Con casi un cuarto de millón de juegos de datos es el portal más grande de la UE. Es un agregador de fuentes públicas de 28 países y con contenidos en más de 20 idiomas. Dispone de un punto de consulta SPARQL y ha promovido una normalización semántica de contenidos denominada DCAT-AP basada en la especificación para catálogos de datos DCAT.

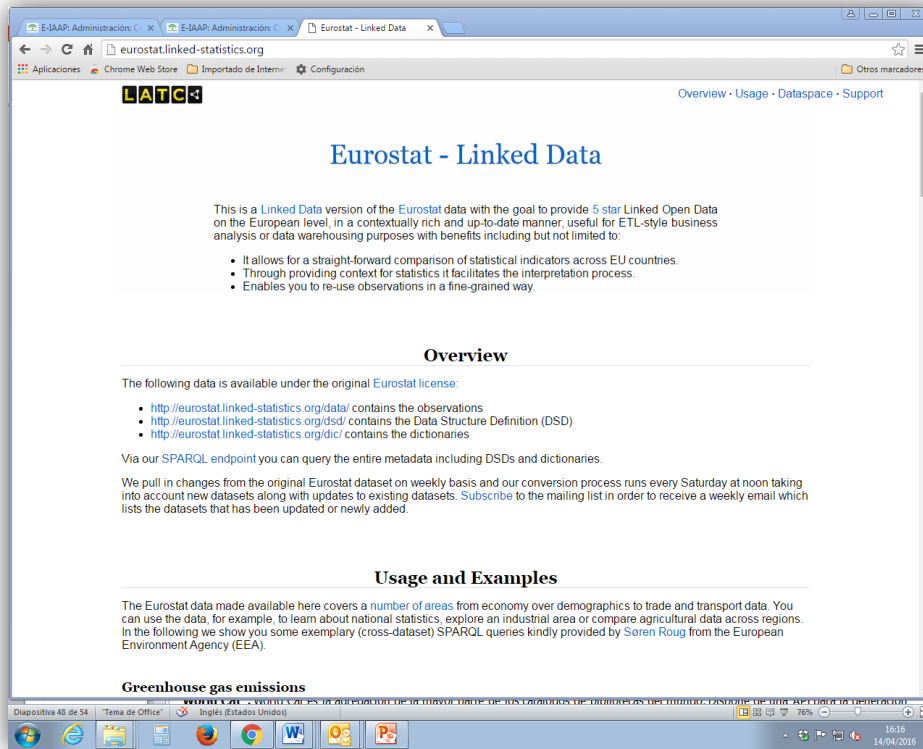


Eurostat-Linked data

WEB Fuente: <http://eurostat.linked-statistics.org/>

SPARQL Fuente: <http://eurostat.linked-statistics.org/sparql>

Es la versión semánticamente enlazada de Eurostat, el portal de datos estadísticos de la Unión Europea. Cuenta con un punto de consulta SPARQL.

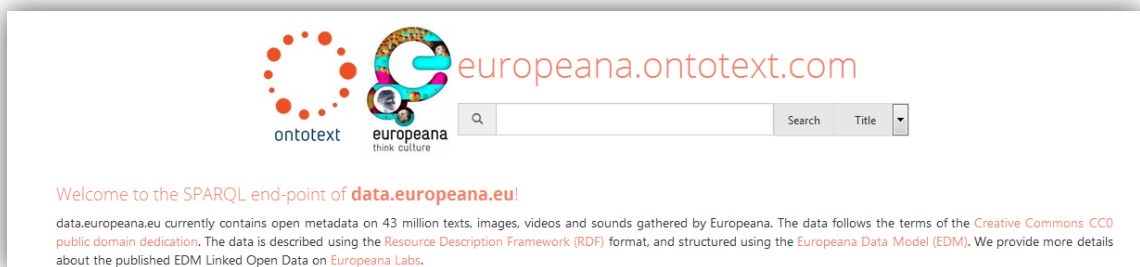


EUROPEANA CONNECT, WORLD CAT Y PORTAL DE DATOS ABIERTOS DE LA UE

Europeana Connect

SPARQL Fuente: <http://europeana.ontotext.com/sparql>

Es la versión linked data de Europeana, el mayor repositorio de objetos culturales de Europa con más de 4 millones. También dispone de un punto de consulta SPARQL.

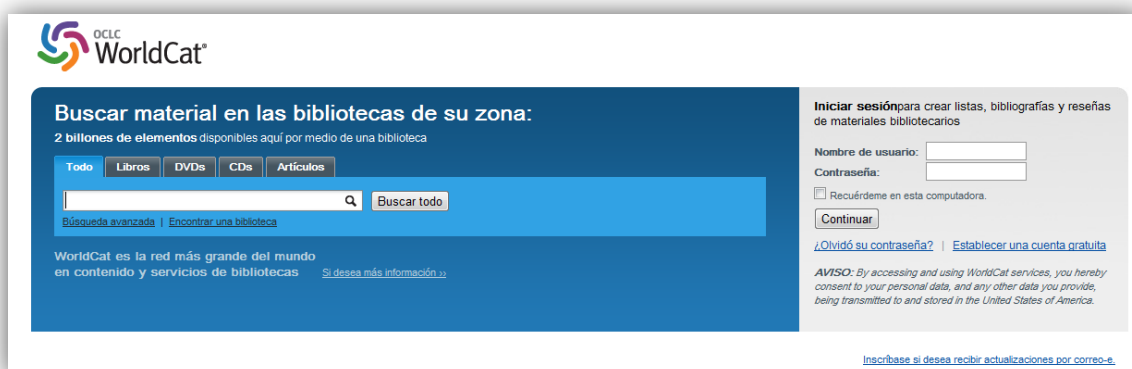


World Cat

WEB Fuente: <https://www.worldcat.org/>

CONSULTAS Fuente: <http://www.oclc.org/developer/develop/web-services/worldcat-search-api.en.html>

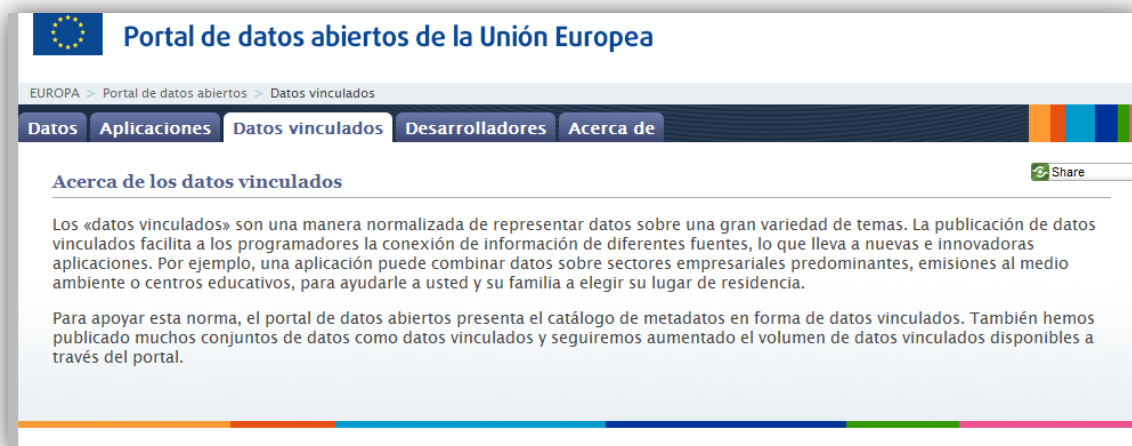
World Cat es la agregación de la mayor parte de los catálogos de bibliotecas del mundo. Dispone de una API para la generación de consultas y no solo incluye libros y artículos sino también CDs (físicos), vídeos, audiobooks y también fotos, y documentos históricos y versiones digitales de algunos de ellos. Todo ello en multitud de lenguajes.



Portal de datos abiertos de la UE

WEB Fuente: <https://open-data.europa.eu/es/linked-data>

El portal de datos abiertos de la Unión Europea publica datos provenientes de la gestión de la administración europea. Incluye casi 8000 juegos de datos en unos 24 lenguajes.



PORTAL DATOS.GOB.ES Y SERVICIO SPARQL DE SMART CITY DE GIJÓN

Portal datos.gob.es

WEB Fuente: <http://datos.gob.es/>

MANUAL Fuente: <http://datos.gob.es/saber-mas?q=node/630515>

CATÁLOGO Fuente: <http://datos.gob.es/sites/default/files/catalogo.rdf>

Catálogo nacional de datos de España. Agrega más de 21.000 conjuntos de datos, y tiene disponible su catálogo en formato RDF/XML. Cuenta con un servicio web que permite explotar su información y cuyo manual de uso está disponible en Datos.gob.es. También permite realizar consultas vía SPARQL.

Servicio *SPARQL* de Smart city de Gijón

WEB Fuente: <https://transparencia.gijon.es/page/12217-servicio-sparql>

Las ciudades también pueden tener recursos *Linked data*, aún teniendo únicamente cerca de 600 *datasets*.



The screenshot shows the website for the Gijón SPARQL service. The header features the Gijón logo and navigation links for 'gijon.es', 'Sede electrónica', 'Mi cuenta', and 'Español'. A search bar is present with the placeholder text 'Texto a buscar'. The main content area is titled 'TRANSPARENCIA' and includes a sidebar with links to 'Gijón Abierto', 'Publicidad activa', 'Observa Gijón', 'Participa Gijón', 'Organigrama', 'Presupuesto 2016', 'Inversiones Municipales', 'Indicadores ITA', 'Otros indicadores', and 'Gobierno'. The main text explains that the Ayuntamiento de Gijón exposes many datasets in RDF format and provides a SPARQL service for querying this data. It also mentions the SPARQL endpoint URL: `http://datos.gijon.es/sparql?query={RDF_QUERY}&output={FORMAT}`.

RESUMEN

Hemos llegado al final de la unidad, y ahora vamos a repasar los **puntos principales** tratados:

- ✓ Las **APIs** son elementos fundamentales para la **utilización masiva de datos y clave para la conectividad** de diversas iniciativas. Resuelven problemas pero para aprovechar todas sus ventajas deben tenerse en cuenta una serie de requerimientos.
- ✓ Los **servicios web** son mecanismos de interacción entre aplicaciones que **operan utilizando protocolos**.
- ✓ El **diseño de API web** fomenta el **acceso y reutilización de datos**, como por ejemplo los datos gubernamentales. Se aprovecha de una normalización en el esquema de direcciones (URLs) y en la utilización del enfoque REST.
- ✓ El enriquecimiento de los datos con metadatos, cuando éstos cumplen los requisitos de la web semántica da lugar a **Linked data o datos enlazados**, cuyos principios fueron establecidos por Tim Berners-Lee.
- ✓ La utilización de los **formatos semánticos** y de los **datos enlazados** permiten expandir el movimiento de datos abiertos.
- ✓ Hay toda una serie de **herramientas para el uso de Linked data** de los que hemos conocido sus principales características, así como un **lenguaje de consulta (SPARQL)** sobre datos estructurados.
- ✓ Hay en marcha una serie **de estrategias y proyectos institucionales relevantes** que apuestan por el uso de API Web como mecanismo para ofrecer datos abiertos gubernamentales, de los que hemos conocido algunos de ellos, como el Portal Paneuropeo de datos, Portal datos.gob.es, World cat, etc.