Formatos de intercambio

PID_00272098

Blas Torregrosa García

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

Blas Torregrosa García

Ingeniero en Informática y máster universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC) por la Universitat Oberta de Catalunya (UOC). Especializado en ciberseguridad. Profesor colaborador en el máster de Ciencia de Datos de la UOC y profesor asociado en la Universidad de Valladolid (UVA).

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Ferran Prados Carrasco (2020)

Primera edición: febrero 2020

Av. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Índice

Introducción

En este módulo se introducen los formatos más frecuentes en la web de datos, comenzado por XML y JSON, así como los formatos pensados para estar integrados en páginas HTML, como son MicroFormatos, MicroData o RDFa. También el formato de intercambio CSV.

1. XML

XML (*Xtensible Markup Language*) fue propuesto por el W3C como un lenguaje de marcado extensible para la web en 1996. XML deriva de SGML (*Standard Generalized Markup Language*) un metalenguaje que proporciona una sintaxis común para sistemas de marcado textual y del cual derivaron las primeras versiones de HTML. El modelo XML está adaptado para representar información textual que combina texto y elementos de marcado.

El modelo XML consta de una estructura en **árbol** en la que cada nodo del árbol se define como un **elemento** de un tipo de dato. Un elemento es una parte del documento delimitado por una etiqueta de inicio (como <nombre>) y una etiqueta de finalización (como </nombre>). Las etiquetas de inicio y las etiquetas finales rodean el contenido y posiblemente otros elementos de marcado, pudiendo haber elementos vacíos (como <pagina/>).

Toda la estructura del documento está anclada a un elemento raíz (el elemento superior). Mientras que en HTML el elemento raíz es <html>, en XML es posible elegir el elemento raíz del documento.

Figura 1. Ejemplo de datos de un curso representado en XML

```
<?xml version="1.0"?>
<curso nombre="Datos">
  <estudiante id="alicia">
    <nombre>Alicia</nombre>
    <genero>Mujer</genero>
  </estudiante>
  <estudiante id="benito">
    <nombre>Benito</nombre>
    <genero>Varón</genero>
    <fechaNacimiento>1981-11-24</fechaNacimiento>
    </estudiante>
</curso>
```

XML ha desarrollaron tecnologías de consulta y transformación a otros formatos. Entre ellos XPath, que es un lenguaje para seleccionar partes de XML, y que está integrado en otras tecnologías como XSLT o XQuery.

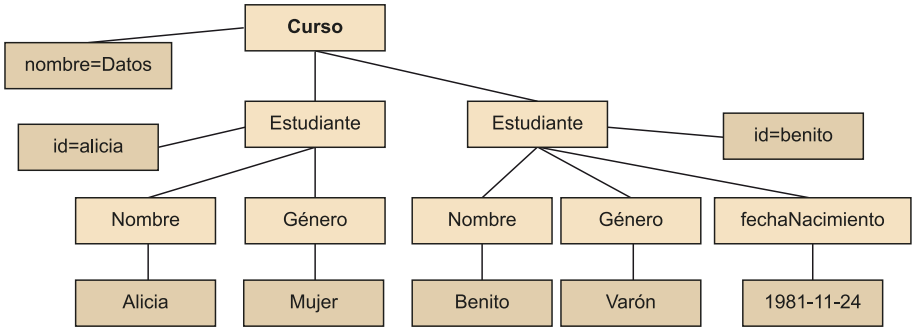
El siguiente fragmento XPath busca los nombres de todos los estudiantes de un curso que son mujeres:

Figura 2. Ejemplo de consulta con XPath

```
// estudiante[genero = "Mujer"]/ nombre
```

XML define el concepto de documentos bien formados y documentos válidos. Los **documentos bien formados** son documentos XML con una sintaxis correcta, mientras que los **documentos válidos** son documentos que, además de estar bien formados, se ajustan a una definición de esquema.

Figura 3. Árbol del documento XML



Para definir un esquema en XML hay varias opciones:

1) *Document Type Definition* (DTD). La especificación XML contiene un mecanismo básico para definir un esquema para documentos XML, heredado de SGML y llamado *DTD*. Una **DTD** genera la estructura de una familia de documentos XML.

DTD utiliza expresiones regulares para definir la estructura de los documentos. Pero DTD tiene un control limitado sobre los tipos de datos. Por ejemplo, no es posible validar que una fecha tenga el formato de fecha.

Figura 4. Ejemplo de DTD

```
<!ELEMENT curso (estudiante*)>
<!ELEMENT estudiante (nombre,genero,fechaNacimiento?)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT genero (#PCDATA)>
<!ELEMENT fechaNacimiento (#PCDATA)>
<!ATTLIST estudiante id ID #REQUIRED>
<!ATTLIST curso name CDATA #IMPLIED>
```

2) XML Schema. Esta especificación se divide en dos partes: la primera define la estructura de los documentos XML y la segunda es un repertorio de tipos de datos de XML Schema.

Un validador de XML Schema añade a cada estructura del documento XML información adicional denominada *PSVI* (*Post-Schema Validation Infoset*). Esta estructura contiene información sobre el proceso de validación.

Figura 5. Ejemplo de XML Schema

```
<xs:schemaxmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:elementname="curso">
    <xs:complexType>
    <xs:sequence>
        <xs:elementname="estudiante" minOccurs='1'
                                     maxOccurs='100' type="Estudiante"/>
    </xs:sequence>
    <xs:attributename="nombre" type="xs:string" />
    </xs:complexType>
  </xs:element>
  <xs:complexTypename="Estudiante">
    <xs:sequence>
      <xs:elementname="nombre" type="xs:string" />
      <xs:elementname="genero" type="Genero" />
      <xs:elementname="fechaNacimiento" type="xs:date" minOccurs='0'/>
    </xs:sequence>
    <xs:attributename="id" type="xs:ID" use='required'/>
  </xs:complexType>
  <xs:simpleTypename="Genero">
    <xs:restrictionbase="xs:token">
      <xs:enumerationvalue="Varon"/>
      <xs:enumerationvalue="Mujer"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

1.1. Validación de documentos XML

Existen diferentes enfoques para indicar cómo se han de validar los documentos XML respecto a un esquema. Estos son algunos de ellos:

1) Esquema integrado. Los DTD se pueden integrar directamente en los documentos XML.

Figura 6. DTD integrado en documento XML

```
<!DOCTYPE curso [
   <!ELEMENT curso (estudiante*)>
   <!ELEMENT estudiante (nombre,genero,fechaNacimiento?)>
   ...
]>
<curso name="Datos">
 ...
</curso>
```

2) Asociación directa con el esquema XML. Se puede hacer utilizando los atributos `xsi: SchemaLocation` o `xsi: noNamespaceSchemaLocation`.

Figura 7. Documento XML con declaración de esquema y localización

```
<curso  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://ejemplo.co.es/ns/Curso
                            http://ejemplo.co.es/curso.xsd">

...
</curso>
```

2. JSON

ECMA Script

ECMA Script es una especificación de lenguaje de programación basado en JavaScript propuesto como estándar por Netscape. Actualmente está aceptado como el estándar ISO 16262.

Douglas Crockford propuso JSON (*JavaScript Object Notation*) en 2001 como un subconjunto de JavaScript. Dicho JSON ha evolucionado hacia un formato de intercambio de datos independiente con su propia especificación de la ECMA (**European Computer Manufacturers Association**).

Un documento JSON puede definirse recursivamente de la manera siguiente:

- `true`, `false` y `null` son valores JSON.
- Cualquier número decimal también es un valor JSON.
- Cualquier cadena de caracteres Unicode entre comillas ("") es también un valor JSON, denominado *valor de cadena*.
- Si k_1, k_2, …, k_n son n valores de cadena distintos y v_1, v_2, …, v_n son valores JSON, entonces {k_1: v_1, k_2: v_2, …, k_n: v_n} son valores JSON, denominados *objetos*. En este caso, cada k_i : v_i es un par atributo-valor. El orden de los pares no es significativo.
- Si v_1, v_2, …, v_n son valores JSON, entonces [v_1, v_2, …, v_n] es un valor JSON denominado *array*. El orden de los elementos del *array* no es significativo.

Hay que tener en cuenta que en el caso de *arrays* y objetos, los valores v_i pueden ser también objetos o matrices, lo que permite a los documentos un nivel arbitrario de anidamiento. De esta forma el modelo de datos JSON se puede representar como un árbol.

Figura 8. Ejemplo de JSON con dos claves: nombre y estudiantes

```
{
     "nombre": "Datos" ,
"estudiantes":[
        { "nombre": "Alicia",
          "genero": "Mujer",
            "edad": 21
        },
        { "nombre": "Benito",
          "genero": "Varón",
 "fechaNacimiento": "1981-11-24"
        }
    ]
}
```

JSON Schema se propuso como un lenguaje de esquema para JSON con funcionalidad parecida a XML Schema para XML. Está escrito utilizando la sintaxis JSON y es independiente del lenguaje de programación. Contiene los si-

guientes tipos de datos predefinidos: `null`, `boolean`, `object`, `array`, `number` y `string`, y permite definir restricciones sobre cada uno de ellos. Las siguientes son palabras reservadas en JSON Schema:

- `$schema`. Indica que el esquema está definido según una versión del estándar.
- `$id`. Define el URI para el esquema.
- `Title` y `description`. No añaden restricciones, solo describen el esquema.
- `Type`. Define la restricción de validación de los datos JSON.

En JSON Schema es posible tener definiciones reutilizables a las que luego se puede referenciar.

Figura 9. Ejemplo de JSON Schema

```
{ "$schema": "http://json-schema.org/draft-04/schema#",

    "definiciones": {
        "estudiante": { "type": "object",
            "properties":{
                "nombre": {"type": "string" },
                "genero":{"type": "string", "enum":["Mujer","Varón"]},
                "fechaNacimiento":{"type": "string", "format": "date" },
                "edad": {"type": "integer","minimum": 1 }
                },
                "required":["nombre","genero"]

            }
        },

    "type": "object",
    "properties":{
        "nombre":{ "type": "string" },
        "estudiantes" : { "type": "array",
            "items":{ "$ref": "#/definiciones/estudiante" }
        }
    },
    "required":["nombre","estudiantes"]
}
```

2.1. JSON-LD

JSON-LD (*JSON for Linking Data*) es un formato para estructurar datos en páginas web, análogo a Microdata y RDFa. Para ello JSON-LD se basa en la notación JSON y la amplía con una sintaxis mediante la cual los datos se anotan en función de esquemas de validez universal. La especificación de JSON-LD figura como recomendación oficial del W3C desde 2014.

Figura 10. Anotación con JSON-LD

```
<script type="application/ld+json">{
    "@context": "https://schema.org",
    "@type": "Course",
    "name": "Introducción a los datos",
    "description": "Introducción a los tipos y orígenes de datos.",
    "provider": {
        "@type": "Organization",
        "name": "Universitat Oberta de Catalunya",
        "sameAs": "http://www.uoc.edu"
    }
</script>
```

JSON-LD no se asocia en principio a ningún vocabulario en particular, si bien el proyecto schema.org está considerado *de facto* como el estándar para la anotación.

Los siguientes elementos son de JSON-LD:

- Etiqueta `<script>`. Indica que el contenido del *script* es JSON-LD (entre llaves {}). Aun cuando JSON se anota en etiquetas *script*, no constituye código ejecutable.

- `@context`. Define el vocabulario que se utiliza para marcar los datos, en este caso schema.org.

- `@type`. Especifica el tipo de elemento que se está anotando.

- **Pares propiedad-valor**. Constituyen las propiedades de tipo de elemento descrito. Las componentes son:
  - **Propiedad**: proviene del vocabulario del contexto y debe ir entre comillas dobles (""). Debe pertenecer a las propiedades permitidas para el tipo de elemento.
  - **Valor**: es el valor que tiene la propiedad y que debe estar en concordancia con el tipo de la propiedad; así, valores simples se asignan por separado mientras que los valores múltiples son *arrays* que van entre corchetes ([]). Los números enteros, punto flotante y en doble precisión no necesitan comillas, y los objetos van entre llaves ({}).

La ventaja que ofrece JSON-LD frente a otros formatos es que los datos no se integran en el código HTML, sino que se pueden implementar como un bloque de código autónomo. Esto contribuye a la legibilidad del código y a su independencia respecto al contenido visible.

Google y otros buscadores dan soporte a las anotaciones JSON-LD para las extensiones siguientes:

1) Datos de contacto corporativos: si las organizaciones marcan semánticamente su información de contacto, Google y otros buscadores pueden mostrarlas en las SERP.

2) Logotipos: cuando se marcan los logotipos el buscador sabe qué gráfico ha de utilizar como imagen corporativa oficial.

3) Vínculos a perfiles sociales: si se han marcado los enlaces a los perfiles sociales como datos estructurados, los buscadores amplían los resultados de personas y organizaciones con los botones correspondientes a las redes sociales (Facebook, Twitter, Instagram, YouTube, LinkedIn, etc.).

Google y otros buscadores soportan las anotaciones JSON-LD para los siguientes tipos: artículos periodísticos, libros, música, cursos, ofertas de trabajo, podcasts, vídeos, películas, recetas de cocina, críticas, productos, conjuntos de datos y más.

SERP

SERP (*Search Engine Results Page*) hace referencia a los resultados que muestra una página de buscador como Google, Bing o Yahoo! entre los más destacados.

3. Datos integrados en HTML

3.1. HTML5 Microdata

Microdata es una alternativa para integrar datos estructurados en HTML 5. Microdata utiliza etiquetas específicas para añadir contenido semántico:

Microdata

En el enlace «Cómo funcionan los datos estructurados» descubriréis cómo usa Google los microdatos en los resultados de búsqueda.

- `Itemscope`. Indica que se va a definir una entidad o contenido en un bloque e informa al navegador que todas las propiedades definidas dentro de este elemento le pertenecen.
- `Itemtype`. Indica el tipo de contenido; por ejemplo, una persona, un evento, etc.
- `Itemprop`. Indica las propiedades para ese tipo de contenido.
- `Itemid`. Indica un identificador único para un elemento; por ejemplo, un URI o URN.

3.1.1. LocalBusiness

Para facilitar a los motores de búsqueda la ubicación de negocios resulta útil marcar los datos con microdatos que añadan más información y, por lo tanto, que aumenten las posibilidades de mostrar esta información adicional junto con el listado de los resultados de la búsqueda.

Figura 11. Ejemplo de microdato para LocalBusiness

```
<div itemscope itemtype="http://schema.org/LocalBusiness"></div>
```

Algunas propiedades que se pueden incluir serían:

Propiedades

La lista completa de propiedades está en http://schema.org/LocalBusiness.

- `name`. El nombre del negocio.
- `description`. Contiene una breve descripción del negocio.
- `image`. Una imagen del negocio o el logo.
- `url`. Dirección del sitio web.
- `address`. Incluye la dirección en la que además se puede marcar separadamente la localidad o el código postal.
- `map`. Mapa de la localización (por ejemplo, enlace con Google Maps).
- `telephone`. El número de teléfono.
- `openingHours`. El horario.

Figura 12. Microdatos Local Business

```
<div itemscope itemtype="http://schema.org/LocalBusiness"></div>
    <span itemprop="name">Ejemplo Co.</span>
    <p itemprop="description">Una breve descripción de Ejemplo Co. </p>
    <img itemprop="image" src="ejemplo_co.png" alt="Foto Ejemplo Co.">
    <a itemprop="url" href="https://ejemplo.co.es"> Ejemplo Co.</a>
    <div itemprop="address" itemscope
         itemtype="http://schema.org/PostalAddress">
        <span itemprop="streetAddress">Calle Calle 1</span>
        <span itemprop="addressLocality">Ciudad</span>,
        <span itemprop="addressRegion">Provincia</span>,
        <span itemprop="postalCode">99123</span>
    </div>
    <a itemprop="map" href="http://maps.google.com">Ver en mapa</a>
    <span itemprop="telephone">999 012345</span>
    <a itemprop="email" href="mailto:ejemplo@ejemplo.co.es">Correo</a>
    <time itemprop="openingHours" datetime="Mo-Fr 09:00-21:00">
            De lunes a viernes de 9h a 21h.
    </time>
</div>
```

3.2. Microformatos

Los microformatos (µF) es una especificación de patrones HTML para publicar datos estructurados sobre conceptos frecuentes, como personas, eventos, publicaciones de blogs o reseñas.

Los **microformatos** son elementos de marcado semántico que utilizan POSH (*Plain Old Semantic HTML*) con un conjunto de valores para las etiquetas `class`, `rel` y `rev`. Los microformatos son abiertos y están disponibles libremente para que cualquiera los pueda usar.

Tabla 1. Algunos tipos de microformatos

Nombre	Utilidad
`hCalendar`	Marcar eventos.
`hCard`	Representa a personas, compañías, organizaciones y lugares.
`XFN`	Representa relaciones sociales.
`hReview`	Representa reseñas de productos, negocios, eventos, etc.
`hAtom`	Representa entradas de blogs.
`rel-license`	Representa licencias de uso.
`hResume`	Representa *curriculum vitae*.
`hRecipe`	Representa recetas de cocina.
`Geo`	Representa localizaciones geográficas.

Microformatos

En microformats.org/wiki/Main_Page se pueden encontrar todos los microformatos existentes.

El principal inconveniente de los microformatos es que no pueden representar cualquier tipo de datos.

3.2.1. Microformato `hCalendar`

El microformato `hCalendar` se utiliza para entradas de calendario para eventos deportivos, aniversarios, recordatorios, reuniones, conferencias y otros eventos. La clase principal de `hCalendar` es `vCalendar` y la clase para eventos es `vEvent`, que es necesaria para todos los eventos. Las propiedades se representan con los elementos de `hCalendar` que necesita las clases `dtstart` y `summary`.

dtstart

Debería estar en formato de fecha ISO 8601 para representaciones de fecha y hora.

Figura 13. Conferencia representada en `hCalendar`

```
<div class="vevent">
    <h1 class="summary">Conferencia Datos 2020</h1>
    <div class="description">La conferencia Datos 2020 ha sido
anunciada.</div>
    <div>Publicado :
        <abbr class="dtstamp" title="20200525T000000Z">
           25 de Mayo de 2020
        </abbr>
    </div>
    <div>Organizada por:
       <a class="organizer" href="mailto:datos2020@ejemplo.co.es">
          datos2020@ejemplo.co.es
       </a>
    </div>
    <div>Fechas:
         <abbr class="dtstart" title="20200525T093000Z">
              25 de Mayo de 2020, 9.30am
        </abbr> -
        <abbr class="dtend" title="20200526T200000Z">
              26 de Mayo de 2020, 8.00pm
        </abbr>
      </div>
</div>
```

Como propiedades opcionales, entre otras, podrían añadirse `location`, `duration` (en formato duración de fechas ISO), `category`, `description`, `geo` (latitud y longitud), `contact`, `organizer` o `status`.

3.2.2. Microformato `hCard`

El microformato `hCard` se utiliza para representar datos de contactos de personas, compañías u organizaciones y se basa en el estándar `vCard` (RFC 2426), por lo que el marcado `vCard` es convertible en `hCard`.

vCard

El estándar `vCard` se utiliza para las tarjetas de visita de empresas. Muchos móviles usan `vCard` para almacenar los contactos en el dispositivo.

La clase base de `hCard` es `vCard` y los únicos atributos obligatorios en `hCard` son `fn` y `n`.[1]

(1)Si se omite `n` pero está definido `fn`, entonces el valor de `n` será el de `fn`.

Figura 14. Tarjeta de visita con `hCard`

```
<link rel="profile" href="http://microformats.org/profile/hcard" />
…
<div id="hcard-Persona-Ficticia" class="vcard">
    <img src="http://ejemplo.co.es/persona_ficticia.png"
         alt="Foto de Persona Ficticia" class="photo" />
    <a class="url fn" href="http://ejemplo.co.es">Persona Ficticia</a>
    <div class="org">Empresa Ficticia</div>
    <a class="email" href="mailto:persona_ficticia@ejemplo.co.es">
         persona_ficticia@ejemplo.co.es
    </a>
    <div class="adr">
        <div class="street-address">Avenida Principal</div>
        <span class="locality">Ciudad</span>,
        <span class="region">Provincia</span>,
        <span class="postal-code">99123</span>
        <span class="country-name">España</span>
    </div>
    <div class="tel">+99 123456789</div>
</div>
```

3.2.3. `rel-license`

Hay millones de recursos web con derechos reservados y también muchos tipos de licencias asociadas a documentos y objetos. Con el microformato `rel="license"` se pueden añadir hipervínculos que apuntan a la descripción de la licencia, lo que es especialmente útil para imágenes, aunque puede usarse para cualquier tipo de recurso.

Figura 15. Imagen con licencia Creative Commons Attribution-Share Alike

```
<link rel="profile" href="http://microformats.org/profile/rel-license" />
…
<img src="imagen.jpg" alt="Mi imagen"
     rel="license"
     ref="http://creativecommons.org/licenses/by-sa/4.0/" />
```

El valor de atributo `href` proporciona el URI asociada al recurso que describe la licencia. Algunas de las licencias más frecuentes son:

- Creative Commons Attribution (cc by)
- Creative Commons Attribution Share Alike (cc by-sa)
- Creative Commons Attribution No Derivatives (cc by-nd)
- Creative Commons Attribution Non-Commercial (cc by-nc)
- Creative Commons Attribution Non-Commercial Share Alike (cc by-nc-sa)
- Creative Commons Attribution Non-Commercial No Derivatives (cc by-nc-nd)

3.3. RDFa

RDFa (*RDF in Attributes*) permite poner tripletas RDF en HTML como valores de atributos. La sintaxis completa (RDFa *Core*) proporciona características para expresar datos estructurados complejos como relaciones humanas, lugares o eventos. Existe un subconjunto mínimo (RDFa *Lite*) más sencillo y adecuado para muchos propósitos.

RDFa *Lite* tiene los atributos siguientes:

- `vocab`. Define el vocabulario en el que se basa el marcado de elementos con RDFa; por ejemplo, schema.org.
- `typeof`. Indica de qué tipo es el elemento descrito.
- `property`. Proporciona una propiedad del elemento.
- `resource`. Permite asignar identificadores a los elementos.
- `prefix`. Permite prefijos para representar recursos y abreviar.

En RDFa la relación entre dos recursos se puede expresar con el atributo `rel`, mientras que las relaciones inversas se expresan con el atributo `rev`.

Figura 16. RFDa para anotar personas

```
<div vocab="http://xmlns.com/foaf/0.1/" typeof="Person">
    <p>
        <span property="name">Alicia</span>,
        Correo: <a property="mbox" href="mailto:alicia@ejemplo.co.es">
                   alicia@ejemplo.co.es
                </a>,
        Teléfono: <a property="phone" href="tel:+99 12345678">
                    +99 12345678
                </a>
    </p>
    <ul rel="knows">
        <li resource="http://ejemplo.co.es/benito/" typeof="Person">
             <span property="name">Benito</span>
        </li>
        <li resource="http://ejemplo.co.es/carlos/" typeof="Person">
            <span property="name">Carlos</span>
        </li>
    </ul>
</div>
```

Tabla 2. Vocabularios de uso frecuente en RDFa

Prefijo	URI	Vocabulario
cc	http://creativecommons.org/ns#	Creative Commons Rights Expression Language
dcterms	http://purl.org/dc/terms/	Dublin Core Metadata Terms
dc	http://purl.org/dc/elements/1.1/	Dublin Core Metadata Element Set, V. 1.1
foaf	http://xmlns.com/foaf/0.1/	Friend of a Friend (FOAF)
og	http://ogp.me/ns#	Facebook OpenGraph

Prefijo	URI	Vocabulario
sioc	http://rdfs.org/sioc/ns#	SIOC Core
vcard	http://www.w3.org/2006/vcard/ns#	vCard in RDF
schema	http://schema.org/	schema.org

RDFa proporciona una API (*Application Programming Interface*) que accede al DOM (*Document Object Model*) para extraer y utilizar datos estructurados en la página web.

4. CSV

CSV (siglas de *comma-separated values*) es un archivo de texto que se utiliza para **transferir datos** de una aplicación a otra. Los archivos CSV no tienen un esquema predeterminado. Normalmente para CSV (y similares, como TSV, *tab-separated values*) se exporta desde las tablas de una base de datos relacional. La cabecera y el significado de las columnas suelen aparecer como una fila dentro del archivo CSV, aunque pueden omitirse.

Las características siguientes son propias de un archivo CSV:

1) Un carácter para **separar registros** individuales (normalmente retorno de carro `CRLF`).

Figura 17. Ejemplo de registros en CSV

```
aaa,bbb,ccc CRLF
zzz,yyy,xxx CRLF
```

2) Un carácter para **separar columnas** individuales (comas, tabulaciones o espacios).

Figura 18. Ejemplo de campos separados por coma

```
aaa,bbb,ccc
```

3) Delimitador de campos que es un carácter para evitar confusiones con los separadores (normalmente comillas dobles o simples).

Figura 19. Ejemplo de campos delimitados por comillas

```
"aaa","bbb","ccc" CRLF
"zzz","yyy","xxx" CRLF
```

4) Opcionalmente, en la primera línea del archivo puede aparecer una **cabecera** con los nombres de los campos en el mismo formato que las líneas de registros.

Figura 20. Ejemplo de cabecera en CSV

```
campo1,campo2,campo3 CRLF
aaa,bbb,ccc CRLF
zzz,yyy,xxx CRLF
```