CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Integración y puesta en valor de conjuntos de datos abiertos

Autor: Jesús Pérez Melero

Director: Raúl García Castro

MADRID, ENERO 2018

Dedicatoria

Agradecimientos

Índice de Contenido

Índice de Ilustraciones

Índice de Ejemplos

Índice de Tablas

1.- Introducción

El presente trabajo pretende realizar un sistema que tome como entrada distintos **conjuntos de datos abiertos** del portal de datos abiertos de la ciudad de **Madrid**.

La información contenida en dichos conjuntos de datos **se enriquecerá** en función de las necesidades oportunas (falta o exceso de información) y, posteriormente, se publicará siguiendo los principios de los datos enlazados, concretamente en el **formato RDF**. Los datos enlazados [1] constituyen un **método de publicación de información estructurada e interconectada**, que además de servir para nutrir páginas web para uso de los humanos, extiende su uso a la **lectura automática** por parte de las máquinas, siendo esto un pilar fundamental de la **Web Semántica**.

Los datos resultantes de dicho método serán la **base** de una aplicación que permita realizar distintos tipos de consultas sobre la ciudad de Madrid, la mayoría de ellas de índole **turístico y cultural**. Esta aplicación será presentada en el **Concurso de Datos Abiertos de la Ciudad de Madrid** [2], que tiene lugar el último trimestre del año 2017.

Nótese que todos los recursos utilizados en la realización de este proyecto (software, documentación, obras, etc.) tendrán **licencias libres** [3].

1.1.- Lista de tareas

Tomando como referencia la propuesta de trabajo proporcionada por el profesor tutor, la lista de tareas en las cuales se dividirá la realización del presente trabajo son:

1. **Estudio del dominio**: Se realizará un estudio previo del dominio de los datos enlazados, es decir, métodos de publicación, tipos de conjuntos de datos, análisis de conjuntos de datos, análisis de software adecuado para el tratamiento de datos enlazados, etc.
2. **Análisis de sistemas relevantes al trabajo**: Se realizará un estudio de los sistemas que pueden ser útiles de cara a la realización de este trabajo.
3. **Extracción de requisitos**: Los requisitos serán la base del sistema que se construya. Se definirá cuáles son y por qué deben considerarse como requisitos.
4. **Diseño del sistema**: Una vez definidos los requisitos se procederá al diseño técnico del sistema, indicando qué herramientas (lenguajes, IDEs, Frameworks…) serán utilizados en la fase de implementación.
5. **Implementación del sistema**: Se realizará la implementación del sistema siguiendo el diseño previo.
6. **Aplicación y evaluación del sistema**: Una vez construido el sistema, este será evaluado utilizando un caso de uso para comprobar su efectividad, y acompañando dicha evaluación de un test de usabilidad para asegurar la calidad del sistema construido, de cara al usuario.
7. **Escritura del documento y preparación de la presentación**: De forma paralela a todas las tareas, se redactará el documento que contendrá toda la información relativa al desarrollo del presente trabajo.

2.- Estudio del dominio

Esta sección tiene como objetivo presentar los conceptos básicos que se van a tratar en la realización del TFG, así como sus características principales.

2.1.- Datos abiertos

Los datos abiertos [4] (*open data*) forman parte de una filosofía que ha ido tomando cada vez más importancia en nuestro tiempo. Esta filosofía tiene como objetivo principal que ciertos tipos de datos estén disponibles y accesibles para todo el mundo, sin ningún tipo de restricción de derechos de autor, patentes u otros mecanismos de control. Este movimiento se identifica con otros similares como pueden ser el del software libre o el de código abierto (*open source*).

Para que los datos puedan considerarse *abiertos*, deben ser accesibles y reutilizables sin necesidad de ningún tipo de permiso. Habitualmente los datos abiertos contienen información sobre geografía, cartografía, meteorología, datos médicos, contabilidad... Históricamente esos datos han sido propiedad de organizaciones públicas o privadas pero ahora gracias a este movimiento esos datos se han convertido en públicos, para la vista y consumo de todos.

Los datos abiertos pueden presentarse en multitud de formatos, preferiblemente formatos libres como XML, RDF, JSON, CSV, RSS... aunque algunas entidades prefieren utilizar formatos propietarios como DOC(X) o XLS(X).

Con la intención de sumarse a este movimiento, el gobierno de España tiene habilitado un portal, de carácter nacional, con información de diversos tipos, accesible y reutilizable en http://datos.gob.es.

2.2.- Datos enlazados

Actualmente multitud de datos son expuestos en la web en formatos muy conocidos como pueden ser HTML, PDF, XML, etc. Las páginas web y los servidores se nutren de bases de datos de distinta índole para proporcionar dichos datos a los usuarios, formando así los documentos que todos conocemos y dando lugar a la web 2.0. Sin embargo, si queremos realizar consultas complejas sobre varias páginas o fuentes de datos, encontramos muchas dificultades.

Una característica clave de los datos enlazados es utilizar la web como una única base de datos global, de la cual poder leer y escribir información, convirtiendo así la web de documentos en una web de datos.

De acuerdo con *Tim Berners Lee*, los cuatro principios de los datos enlazados son:

- Utilizar URIs como nombres para las cosas.
- Utilizar URIs HTTP para que la gente puede mirar a través de esos nombres.
- Cuándo se accede a una URI, se obtiene información útil gracias a tecnologías como RDF o SPARQL.
- Incluir referencias a otras URIs, de forma que se cree un enlace y la información de un extremo se enriquezca con la información del otro extremo del enlace.

De esta forma y siguiendo estos cuatro principios, los datos enlazados posibilitan la creación de la web semántica, que es a su vez una parte fundamental de la web 3.0.

2.3.- La Web Semántica

La web semántica [5] es una extensión de la web actual, que se basa en dos grandes pilares:

- La descripción del significado
- La manipulación automática de dichas descripciones mediante lógica y motores de inferencia.

La descripción del significado pasa por utilizar la semántica, los metadatos y las ontologías:

- La semántica persigue el estudio del significado de los términos lingüísticos. La web semántica busca dotar de significado a la información existente en la web. Este significado debe ser interpretable por las máquinas, siendo así necesario que la información adicional que se añada pueda ser procesada por un computador.
- Los metadatos son datos que describen otros datos. En el contexto de la web semántica, son los datos que describen los recursos que hay en la web.
- Las ontologías son jerarquías de conceptos, que contienen atributos y relaciones. Definen una terminología concreta para definir redes semánticas.

Algunas de las ventajas de la web semántica son:

- Incorporación de contenido semántico a los datos que existen en la web. Esto hace posible búsquedas más precisas basadas en significado y no en contenido.
- Hace posible que los computadores gestionen el conocimiento (mediante inteligencia artificial).

Algunas desventajas de la web semántica son:

- Adaptar todos los documentos de la web para que sean procesados de forma semántica es muy costoso.

Es necesaria una unificación de términos junto con la definición de estándares semánticos, para evitar posibles conflictos entre idiomas (por ejemplo, definir que IVA es igual a VAT).

2.4.- Resource Description Framework (RDF)

RDF es un *framework* propuesto por la W3C con la idea original de ser un modelo de datos para metadatos [6]. Establece una serie de normas para clasificar la información dentro de un documento utilizando etiquetas, propiedades y relaciones entre la información, haciendo así posible que sea comprensible por las máquinas. Este *framework* ha sido diseñado preferentemente para las máquinas y no para las personas, por lo que no es espera que este tipo de información sea mostrada por pantalla o devuelta a los usuarios finales.

Utiliza como lenguaje XML (entre otros) y desde Febrero de 2004 es considerado como una de las recomendaciones de la W3C en materia de web semántica. RDF utiliza las URIs como mecanismo para identificar los recursos, los cuales describe con propiedades y valores.

Un ejemplo de RDF-XML es el siguiente:

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:book="http://www.libros.example/book#">

<rdf:Description
rdf:about="http://www.libros.example/book/LaCatedral">
  <book:author>Cesar Mallorqui</ book:author>
  <book:title>La Catedral</ book:title>
  <book:price>12.40</ book:price>
  <book:year>1999</ book:year>
</rdf:Description>

<rdf:Description
rdf:about="http://www.libros.example/book/CartasDeInvierno">
  <book:author>Agustin Fernandez Paz</book:author>
  <book:title>Cartas de Invierno</book:title>
  <book:price>9.90</book:price>
  <book:year>1995</book:year>
</rdf:Description>


</rdf:RDF>
```

Ejemplo 1 - RDF-XML

La primera línea del documento es la declaración habitual de XML, que se encuentra seguida por el elemento raíz del documento, `<rdf:RDF>`. En la apertura de este elemento raíz se declaran los diferentes *namespaces* que se van a utilizar a lo largo del documento. Estos *namespaces* son utilizados para acortar la declaración de sucesivos elementos.

Dentro del elemento raíz encontramos elementos de tipo `<rdf:Description>` que contienen la descripción de los elementos que contiene el documento. En el ejemplo propuesto se muestra un fichero RDF que contiene datos sobre libros publicados en España, identificados por el atributo `rdf:about`. Por tanto, al tratarse de libros, dichos

elementos contienen propiedades como el autor, el título, el precio y el año de publicación.

Si bien el ejemplo anterior muestra cómo luce un fichero RDF-XML, existen otros tipos de formatos que se utilizan para serializar RDF, como es el caso de *Turtle* (*Terse RDF Triple Language*). *Turtle* [7] representa la información mediante tripletas, las cuales están formadas por sujeto, predicado y valor.

Cada uno de los elementos que haya se identifica mediante una URI, al igual que en el caso anterior. Un ejemplo de documento RDF con sintaxis *Turtle* sería el siguiente:

```
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix book: <http://www.libros.example/book#/> .

book:LaCatedral a dbo:book ;
      dbo:title "La Catedral"^^<xsd:string> ;
      dbo:author "Cesar Mallorqui"^^<xsd:string> .
```

Ejemplo 2 - RDF-TTL

Como puede observarse, el formato del documento es distinto y algo más comprensible que el anterior. Sin embargo, no debe olvidarse que estos formatos están destinados preferentemente a las máquinas, no a las personas.

Los *namespaces* que definíamos en el anterior ejemplo mediante las etiquetas Xmlns ahora son definidos mediante la anotación **`@prefix`**. La descripción de los elementos contenidos del fichero es en este caso más clarificadora y se puede apreciar claramente el formato de tripletas. En este ejemplo podemos observar dos tripletas distintas:

Sujeto	Predicado	Valor
book:LaCatedral	dbo:title	“La Catedral”
book:LaCatedral	dbo:author	“Cesar Mallorqui”

Tabla 1 - Tripletas del Ejemplo 2

La modelación y descripción de los datos es un proceso complejo ya que la información se modela en un instante de tiempo concreto pero es posible que dicha información cambie en el futuro, siendo así necesario redefinir el modelo. Es necesario por tanto un estudio y esfuerzo considerables para producir un modelo útil que permita obtener los beneficios inherentes a los datos enlazados.

2.5.- SPARQL

SPARQL [8] (*SPARQL Protocol and RDF Query Language*) es un lenguaje utilizado para la consulta de grafos RDF, normalizado por la W3C. Se trata de una parte clave de la web semántica. Una de sus características más atractivas es que permite utilizar varias fuentes de datos, lo que hace que las búsquedas sean más amplias y complejas.

Para ilustrar cómo funciona SPARQL, se muestran a continuación ejemplos de varios tipos de queries sobre la siguiente fuente de datos:

Sujeto	Predicado	Valor
http://www.madridonyou.com/building#reinaSofia	http://www.madridonyou.com/building#buildingType	Museum
http://www.madridonyou.com/building#barRetiro	http://www.madridonyou.com/building# buildingType	Bar
http://www.madridonyou.com/building#plazaColon	http://www.madridonyou.com/building# buildingType	Monument

Tabla 2 - Origen de datos para consultas SPARQL de ejemplo

Las siguientes consultas devuelven información variada tomando como base la fuente anterior:

```
PREFIX building: <http://www.madridonyou.com/building>

SELECT * WHERE

{

   ?sujeto building:buildingType ?valor

}
```

Ejemplo 3 - Query SPARQL: Obtener toda la información

Como se puede observar, las queries en SPARQL utilizan también prefijos para indicar los sitios a los cuáles están haciendo referencia. La sintaxis de la query es similar a la del lenguaje SQL, pero SPARQL tiene por debajo una capa semántica de la cual carece SQL.

La query formulada en el ejemplo 3 muestra el contenido de todos los campos (sujeto, predicado y valor al haber indicado ‘ * ’ en la cláusula `SELECT`) que casen con el patrón especificado en el cuerpo de la cláusula `WHERE`. En dicho cuerpo, se han declarado dos variables, `?sujeto` y `?valor`, que irán tomando distintos valores.

Sin embargo sólo se seleccionarán las tripletas que tengan como predicado `buildingType`. Teniendo en cuenta los datos de partida, el resultado de esta query será un `resultSet` idéntico a la Tabla 2.

```
PREFIX building: <http://www.madridonyou.com/building>

SELECT * WHERE

{

   ?sujeto building:buildingType “Bar”

}
```

Ejemplo 4 - Query SPARQL: Obtener información de un subconjunto de elementos

La query del ejemplo 4 ahora sólo tiene una variable declarada e impone restricciones en el predicado y en el valor. Esta query únicamente devolverá (al igual que en el caso anterior, todos los campos) tripletas que contengan un predicado `buildingType` y un valor “`Bar`”. Es decir, esta query pretende obtener todos los bares presentes en la fuente de datos. Por tanto, se obtendrá el siguiente resultado:

Sujeto	**Predicado**	**Valor**
http://www.madridonyou.com/building #barRetiro	http://www.madridonyou.com/building# buildingType	Bar

Tabla 3 - Resultado del Ejemplo 4

A partir de la versión 1.1 de SPARQL se introdujeron las **quries federadas**, que tienen la siguiente estructura:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>

SELECT ?name

FROM <http://example.org/myfoaf.rdf>

WHERE

{

  <http://example.org/myfoaf/I> foaf:knows ?person .

  SERVICE <http://people.example.org/sparql> {

    ?person foaf:name ?name . }

}
```

Ejemplo 5 - Query Federada

Si bien la cláusula `FROM` no es específica de este tipo de queries (únicamente especifica dónde está la fuente de datos), sí lo es la cláusula `SERVICE`. Esta cláusula establece una nueva fuente de datos y realiza consultas sobre la misma, de forma dinámica, en la misma query, y une los resultados de las queries realizadas antes y después de la misma.

2.6.- Generación y publicación de datos enlazados

El proceso de generación de datos enlazados comprende varias fases que deben ser llevadas a cabo en un estricto orden.

1. **Seleccionar el origen de los datos**: De acuerdo a unos requisitos preestablecidos sobre qué tipo de información se va a tratar, se seleccionan uno o varios datasets como fuente de información. Dichos datasets pueden obtenerse a partir de portales dedicados a tal efecto (de organizaciones, asociaciones, gobiernos…) o de investigaciones propias.
2. **Obtener acceso a los datos**: Una vez se haya seleccionado el conjunto de datos que se quiere tratar, se deberá garantizar que se tienen permisos para acceder a ellos. Esto se hará analizando las licencias que tengan lo(s) dataset(s) en cuestión. Además, se deberá consultar, a través de la licencia, si dichos datos pueden ser usados con los fines planeados para ellos.
3. **Analizar los datos**: Una vez se esté en disposición de los datos, deberán ser analizados en busca de posibles fallos a corregir o ajustes que deban ser aplicados. Esto podrá realizarse con aplicaciones como LOD Refine, Open Refine o Karma.
4. **Desarrollo de una estrategia de nombrado**: Para identificar cada uno de los elementos existentes en los conjuntos de datos, debe definirse una estrategia de nombrado que sea capaz de asignar una URI única a cada elemento, de manera que sea referenciable.
5. **Desarrollo de una ontología**: Se deberá desarrollar una ontología que dote de significado a los elementos de lo(s) dataset(s), y permita generar datos en formato RDF. A la hora de definir la ontología, puede ser útil buscar en ontologías existentes términos ya definidos que se adapten a lo que estamos buscando, como DBpedia o EsDBpedia.
6. **Enlace con otros datasets**: Una vez se tengan los ficheros RDF producidos por la ontología, se deberán enlazar con otros datasets existentes para así generar una red de datos enlazados y generar algún tipo de beneficio.

2.7.- Ontologías

Una ontología se define como una definición formal de tipos, propiedades y relaciones entre entidades pertenecientes a un mismo dominio. Las ontologías son utilizadas para categorizar y organizar la información, así como limitar la complejidad de los problemas.

Se usará Protegé para desarrollar la ontología del sistema que se va a construir.

- Falta ahondar ligeramente en el término de ontología -

3.- Análisis de sistemas relevantes

En esta sección se describirán sistemas que van a ser usados en el desarrollo de este proyecto.

3.1.- Sistemas y fuentes de información

El presente sistema tiene como base distintos tipos de conjuntos de datos que proporcionan conocimiento al mismo. Dichos conjuntos serán tomados de diversas fuentes:

3.1.1.- Portal de datos abiertos de Madrid

El portal de datos abiertos de Madrid contiene una gran cantidad de datasets, separados en distintas categorías, destinados al uso público. Para cada *dataset*, se dispone de su sector, fecha de incorporación al catálogo, frecuencia de actualización y los formatos disponibles.

Si bien son varios los formatos que ofrecen (CSV, JSON, XML, RDF, KML, XLSX…) serán los ficheros en formato CSV los que se utilizarán en principio como base de conocimiento del sistema.

De acuerdo con el propósito del sistema que se pretende construir, se tomarán datos de los sectores de turismo (47%), automovilismo (29%), ciclismo (18%) y ocio (6%).

Este portal se encuentra en http://datos.madrid.es/portal/site/egob.

Ilustración 1 - Portal de Datos Abiertos de Madrid

3.1.2.- DBpedia y ESDBpedia

DBpedia es un repositorio online abierto y gratuito. Contiene información estructurada obtenida directamente de Wikipedia mediante procesos de mapeo. Wikipedia está compuesta por documentos, algo que no ocurre en DBpedia, que se compone de datos estructurados. Esto permite que se puedan realizar consultas con alto nivel de complejidad contra DBpedia que no sería posible lanzar a Wikipedia.

ESDBpedia es igual que DBpedia pero orientada a información localizada en España.

Estos dos repositorios serán utilizados en este caso como fuentes que enriquecerán los conjuntos de datos obtenidos del portal de datos abiertos con información adicional que originalmente no está presente en dichos *datasets*.

Nótese que el uso de DBpedia está disponible bajo los términos que indica la licencia Creative Commons Attribution-ShareAlike 3.0.

Ilustración 2 - DBpedia

3.2.- Sistemas para el desarrollo de ontologías. Protégé

Para el desarrollo de la ontología que se usará en este proyecto, se usará la herramienta Protégé, un editor open-source de ontologías para construir sistemas inteligentes. Ha sido desarrollado en la universidad de Stanford.

Ofrece una sencilla interfaz para desarrollar ontologías.

3.3.- Sistemas para el tratamiento de datos

Los datos obtenidos del portal de datos abiertos de Madrid serán tratados con diversas aplicaciones para corregirlos y modificarlos en caso de ser necesario. Además, se usarán dichas aplicaciones para generar los ficheros en formato RDF que enlazarán con otros datasets existentes.

3.3.1.- Open Refine

Open Refine (también conocido como Google Refine) es una herramienta para trabajar con conjuntos de datos. Permite realizar operaciones de limpieza y transformación de los datos, así como generar nuevos datos a partir de los que ya existen.

Una de las ventajas de Open Refine es que permite buscar en distintos datasets ocurrencias similares a los datos que se encuentran en nuestros datasets, permitiéndonos de esta forma realizar un enlazado de forma sencilla entre datasets.

4.- Extracción de requisitos

Se han identificado diferentes *Features* o características que la aplicación deberá tener. Para cada característica o *feature* se detallan uno o más requisitos software con su correspondiente numeración. Por tanto, cada requisito podrá identificarse mediante el patrón *feature (número)*. Por tanto, el requisito 2(1) haría referencia al punto 1 de la *feature* 2.

4.1.- Feature #1 – Inicio de sesión

La aplicación dispondrá de un mecanismo de *login* para poder autenticar al usuario en el sistema. Esta funcionalidad debe estar siempre disponible para cualquier usuario que no esté autenticado previamente.

Descripción:

1. El mecanismo de *login* debe fundamentarse al menos en un nombre de usuario y una contraseña.
2. Si el *login* es correcto, se redirigirá al usuario a la página principal de la aplicación.
3. Si el *login* es incorrecto, se mostrará un mensaje de error (en texto rojo) indicando que alguno de los parámetros era incorrecto (no se debe especificar cuál).
4. Las contraseñas deben almacenarse en una base de datos relacional habiendo sido tratadas previamente por una función hash.
5. Si un usuario ya está autenticado en el sistema, será redirigido a la página principal si intenta volver a realizar un *login*. Esto será así hasta que se realice una operación de *logout*, en cuyo caso dejará de estar autenticado en el sistema.

4.2.- Feature #2 – Registro en la aplicación

Dado que es necesario un *login* será necesario disponer de un formulario de registro de nuevos usuarios que puedan utilizar el sistema. Esta funcionalidad, al igual que el *login*, deberá estar disponible para todos los usuarios no autenticados en el sistema.

Descripción:

1. El formulario de registro debe contener los campos mínimos para recopilar información útil del usuario, es decir, nombre de usuario, contraseña, e-mail y nombre.
2. La contraseña deberá tener al menos 6 caracteres, siendo necesario que contenga letras y números.
3. El nombre de usuario elegido no puede ser idéntico a otro ya existente.

4.3.- Feature #3 – Ruta por edificios monumentales

El usuario podrá solicitar un recorrido por distintos edificios de carácter monumental o turístico de Madrid. Esta funcionalidad es exclusiva para los usuarios que estén autenticados en el sistema.

Descripción:

1. El usuario deberá poder elegir si quiere que el recorrido sea por todo Madrid o únicamente por uno o varios distritos.
2. El usuario deberá poder elegir las horas entre las que se va a realizar el recorrido, en cuyo caso sólo se mostrarán edificios que tengan un horario de apertura al público incluido en el rango de horas. En caso de que no se defina un intervalo de horas, se mostrarán las múltiples opciones junto con su horario de apertura al público.
3. Las rutas se podrán personalizar haciendo que se planifiquen recorridos basados en el estilo arquitectónico o el año de construcción, entre otros.
4. Junto con cada edificio perteneciente a la ruta generada, se deberá mostrar una breve descripción del mismo, acompañada de datos útiles para el usuario (teléfono, dirección, e-mail…) y una imagen que se ajuste a la actualidad del edificio.
5. La ruta resultante se formará sobre un mapa con cada elemento perteneciente señalado en él.

4.4.- Feature #4 – Rutas completas por la ciudad

El usuario podrá solicitar un recorrido más completo que en la Feature#3, en el cual no sólo obtendrá información sobre edificios y monumentos sino también sobre cafeterías y restaurantes.

Descripción:

1. El usuario deberá poder elegir si quiere que el recorrido sea por todo Madrid o únicamente por uno o varios distritos.
2. El usuario deberá poder elegir las horas entre las que se va a realizar el recorrido, en cuyo caso sólo se mostrarán elementos que tengan un horario de apertura al público incluido en el rango de horas. En caso de que no se defina un intervalo de horas, se generará un recorrido aleatorio teniendo en cuenta la distancia entre cada uno de los elementos pertenecientes al mismo.
3. Junto con cada elemento perteneciente a la ruta generada, se deberá mostrar una breve descripción del mismo, acompañada de datos útiles para el usuario (teléfono, dirección, e-mail…) y una imagen que se ajuste a la actualidad del edificio.
4. La ruta resultante se formará sobre un mapa con cada elemento perteneciente señalado en él.

4.5.- Feature #5 – Información sobre transporte

Como opción adicional a cualquier ruta que se vaya a diseñar, se habilitará la opción de combinar el recorrido con información sobre cómo aparcar cerca de cada uno de los elementos del recorrido. Se distinguirán 4 tipos de estacionamientos: aparcamientos públicos y SER, aparcamientos para movilidad reducida, puntos de recarga eléctrica y bases y áreas de descanso para bicicletas (*BiciMad*).

Descripción:

1. El usuario podrá elegir un tipo de estacionamiento concreto, o varios.
2. Una vez generada la ruta, se señalará en el mapa (de forma que se distinga de los elementos de la ruta) los puntos de aparcamiento asociados a cada edificio.
3. Un aparcamiento se considerará válido si se encuentra a menos de 500 metros del edificio al cual está asociado. Si se encuentra a más distancia, se obviará el resultado.
4. Un aparcamiento puede estar asociado a varios elementos de la ruta.
5. En el caso de que se seleccionen varios tipos de estacionamiento, se mostrarán en la pantalla de forma que puedan hacerse visibles y ocultos dando así la posibilidad de ver todos los tipos de aparcamientos a la vez (para hacer comparaciones) o solo uno de ellos para no sobrecargar visualmente el mapa.

4.6.- Feature #6 – Accesibilidad y usabilidad

El sistema debe ser intuitivo y usable, de forma que no sea necesario una guía de uso en profundidad del mismo.

Descripción:

1. Ninguna operación dentro del sistema deberá superar las 10 acciones elementales.

5.- Diseño del sistema

Esta sección contiene todo lo relativo al diseño del sistema que se pretende construir. Contiene por tanto las tecnologías que se van a utilizar así como la arquitectura que presentará el sistema.

5.1.- Tecnologías utilizadas

La siguiente tabla muestra las tecnologías que han sido utilizadas en cada parte específica del sistema.

Elemento	Categoría	Componente
Java	Lenguaje de Programación	Back-end
SPARQL	Lenguaje de Programación	Back-end
Angular	Lenguaje de Programación	Front-end
HTML 5	Lenguaje de Marcas	Front-end
CSS 3	Hojas de estilo	Front-end
Amazon DynamoDB	AWS	Base de datos en la nube
Amazon S3	AWS	Sistema de almacenamiento en la nube
Amazon EC2	AWS	Contenedor en la nube
Amazon SES	AWS	Mecanismo de envío de mail en la nube
Amazon SQS	AWS	Mecanismo para colas FIFO en la nube
Eclipse	IDE	Entorno de desarrollo integrado utilizado en el desarrollo del código fuente

Tabla 4 - Tecnologías utilizadas

5.2.- Arquitectura del sistema

El sistema seguirá una arquitectura basada en microservicios o MSA (*MicroServices Architecture)*. Esto quiere decir que el sistema estará compuesto por un conjunto de pequeños servicios que se comunicarán entre sí mediante APIs bien definidas con peticiones HTTP. Cada uno de los microservicios se encargará de dar soporte a un área concreta del sistema.

6.- Implementación del sistema

7.- Aplicación y evaluación del sistema

8.- Bibliografía

1. Wikipedia (2017). *Datos enlazados*. Enlace
2. MediaLab Prado (2017). *Datatón Ciudad de Madrid 2017*. Enlace
3. Wikipedia (2017). *Contenido libre*. Enlace
4. Wikipedia (2017). *Datos abiertos*. Enlace
5. W3C (2017). *Web Semántica*. Enlace
6. W3C (2017). *RDF-XML*. Enlace
7. Wikipedia (2017). *Turtle (Syntax)*. Enlace
8. W3C (2017). *SPARQL Query Language for RDF*. Enlace