



Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**



Grado en Ingeniería Informática

Trabajo Fin de Grado

DATOS ENLAZADOS sobre BICICLETAS en CALLES de MADRID

Autor: Rubén Rodríguez Álvarez
Tutor(a): Oscar Corcho García

Madrid, Abril 2020

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Título: DATOS ENLAZADOS sobre BICICLETAS en CALLES de MADRID

Abril 2020

Autor: Rubén Rodríguez Álvarez

Tutor: Oscar Corcho García
Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Tabla de contenidos

1. Introducción	1
2. Definición de Vocabularios	3
2.1. Vocabulario de Accidentes de Bicicletas	4
2.1.1. Clases	6
2.1.2. Propiedades de datos	9
2.1.3. Propiedades de objeto	12
2.2. Vocabulario de CicloCarriles	19
2.2.1. Clases	21
2.2.2. Propiedades de datos	23
2.2.3. Propiedades de objeto	25
2.3. Vocabulario de Calles Tranquilas	26
2.4. Vocabulario de Callejero (Propuesta)	27
2.4.1. Propiedades de datos	29
2.4.2. Propiedades de objeto	30
3. Transformaciones en los Datasets	31
3.1. Modificaciones manuales en Datasets	33
3.2. Revisiones de abreviaturas y erratas	36
3.3. Obtención de valores para propiedades	40
3.3.1. Propiedad esCruce	40
3.3.2. Propiedad tipoVia	42
3.3.3. Propiedad typicalAgeRange	43
3.3.4. Palabras Clave	44
3.4. Obtención de Identificadores de Vias	46
4. Generación y búsqueda en ficheros	51
4.1. Generación de ficheros OWL	51
4.2. Consultas SPARQL	53
5. Aplicación Android	55
5.1. Obtención de la ruta	55
5.1.1. Conexión API Google Directions	55
5.1.2. Creación BBDD coordenadas y lista de calles transitadas	55
5.2. Cálculo de Seguridad de ruta	55
6. Resultados y conclusiones	57
Bibliografía	60

Anexo	61
--------------	-----------

Capítulo 1

Introducción

Capítulo 2

Definición de Vocabularios

2.1. Vocabulario de Accidentes de Bicicletas

Para el vocabulario asociado con los accidentes de bicicletas se han tomado como referencia los datos proporcionados por el ayuntamiento de Madrid. [3]. En estos datasets se muestran los accidentes de tráfico con implicación de bicicletas dentro de la jurisdicción del ayuntamiento.

Se han añadido además algunos datos no proporcionados en estos datasets, como es el Municipio, el id de la vía u otros, ya que se han considerado necesarios para la definición de un vocabulario reutilizable y aplicable a otros datasets.

Este vocabulario se ha definido con el objetivo de ser válido tanto para accidentes de tráfico de bicicletas como de automóviles u otros vehículos. Aun habiendo partido de un dataset en el que se representaban los accidentes relativos al primer caso, todos los elementos definidos pueden ser utilizados en cualquier tipo de accidente. Debido a que este trabajo está enfocado a crear una aplicación para la seguridad de bicicletas, se ha partido de esta base, pero podría ser perfectamente reutilizado para otro tipo de accidente añadiéndole propiedades necesarias para los mismos como podrían ser la velocidad, el numero de pasajeros...

La organización del conjunto de datos se hará siguiendo el diagrama 2.1

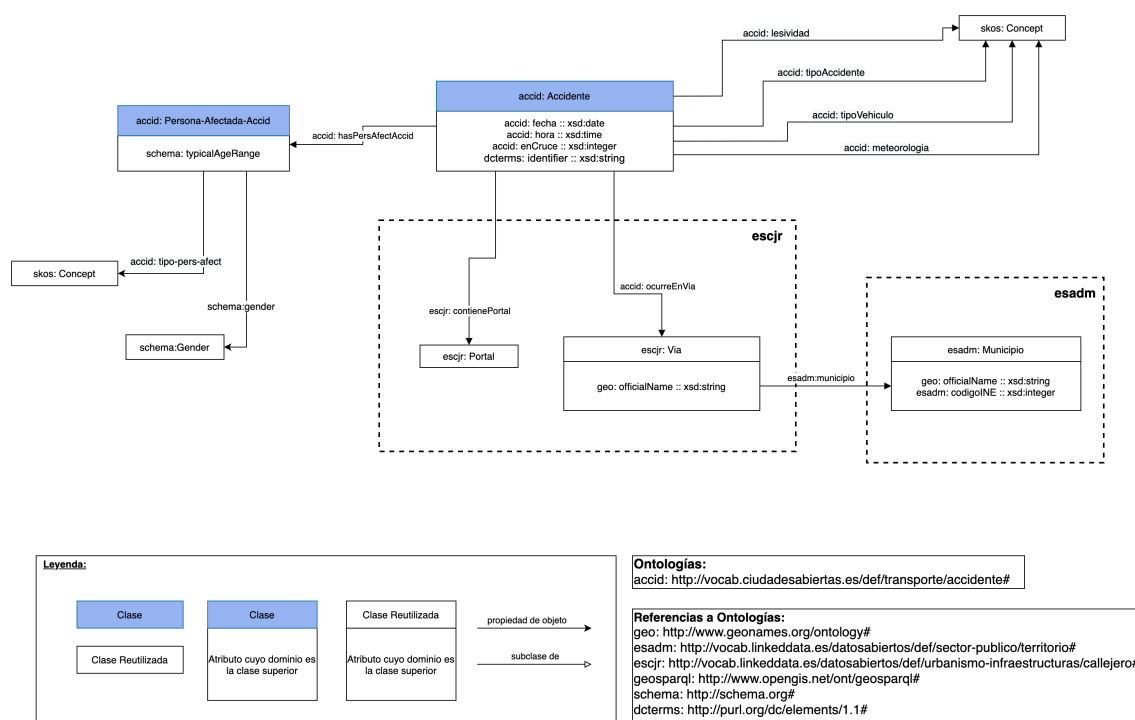


Figura 2.1: Diagrama de Ontología de Accidentes.

Para la representación de los datos de accidentes de tráfico se han definido varias clases y propiedades. Se han reutilizado elementos ya definidos en el vocabulario de Callejero [1], de Territorio [12] y de Schema [2].

En la siguiente tabla se muestran los Namespaces usados.

esadm	http://vocabulary.openstreetmap.org/def/sector-publico/territorio#
escjr	http://vocabulary.openstreetmap.org/def/urbanismo-infraestructuras/callejero#
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
schema	http://schema.org/
www-geonames-org	http://www.geonames.org/
xml	http://www.w3.org/XML/1998/namespace
xsd	http://www.w3.org/2001/XMLSchema#

Figura 2.2: Namespaces usados para Accidentes

Se ha optado por mantener la separación de elementos como fecha y hora, calle y numero debido a que en la fuente de origen están así dispuestos y en la posterior aplicación final que se va a construir será más conveniente tener esa información por separado, para poder disponer de datos a horas con menos luminosidad o calles completas(sin conocer la posición exacta), por ejemplo.

Para este conjunto de datos se ha optado por añadir, además de los ya proporcionados por la fuente de origen del ayuntamiento, nueva información como la propiedad “esCruce“, el municipio, el tipo de vía o el identificador de vía. Son propiedades inferidas de la información proporcionada que permiten que sea más sencillo su tratamiento y uso, para esta u otras aplicaciones que puedan tener estos datos. EsCruce se obtendrá del nombre de la calle, del cual atendiendo a varios patrones se puede determinar si el accidente ha ocurrido en una intersección de dos o más vías. El Municipio se ha añadido para su posible reutilización posterior utilizando otros datasets de otras localidades, para este caso será siempre Madrid. El Identificador de Vía se obtendrá comparando el nombre de la vía y su tipo con el Callejero de Madrid, el cual proporcionará este valor único que represente la vía. El tipo de vía finalmente se ha eliminado del vocabulario ya que no tiene relevancia para los datos obtenidos de éste, más allá de la obtención del identificador de vía. En cualquier caso, si fuese necesario se podría obtener a partir del nombre de la calle, aunque no se ha considerado relevante para añadirlo a la ontología.

En este conjunto de datos se ha hecho un cambio relevante con respecto al original y que será detallada en el capítulo Transformaciones en los vocabularios. Los accidentes que han ocurrido entre un cruce de vías se han separado en tantos registros como vías interfieran. De este modo será mucho más simple la búsqueda de accidentes ocurridos en una calle y se podrá hacer una búsqueda más sencilla de ellas. Se podrá identificar si dos o más registros pertenecen al mismo accidente por el número de expediente, el cual se conserva igual en ambos.

2.1.1. Clases

Accidente

IRI:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente#Accidente>

Siniestro ocurrido en la vía pública con implicación de algún vehículo. El recurso se construirá a partir de su número de expediente.

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

PersonaAfectada

IRI:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente#PersonaAfectada>

La persona perjudicada por el accidente de tráfico.

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Portal

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#Portal>

Ha sido definido por la plataforma ciudadesabiertas [17]. Subacceso independiente exterior (al aire libre) a una misma construcción. Para una misma construcción, con un mismo número de vía, pueden existir varias entradas que pueden estar numeradas con números o letras. [fuente: Modelo de Direcciones de la Administración General del Estado v.2]

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Municipio

IRI: <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#Municipio>

Se ha reutilizado la definición de Municipio proporcionada por vocab.linkeddata.es [12]. Un Municipio es el ente local definido en el artículo 140 de la Constitución española y la entidad básica de la organización territorial del Estado según el artículo 1 de la Ley 7/1985, de 2 de abril, Reguladora de las Bases del Régimen Local. Tiene personalidad jurídica y plena capacidad para el cumplimiento de sus fines. La delimitación territorial de Municipio está recogida del Registro Central de Cartografía del IGN. Su nombre, que se especifica con la propiedad `dct:title`, es el proporcionado por el Registro de Entidades Locales del Ministerio de Política Territorial, en <http://www.ine.es/nomen2/index.do>

Definida por:

<http://purl.org/derecho/vocabulario>

<http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio>

<http://www.ign.es/ign/resources/acercaDe/tablon/ModeloDireccionesAGE>

Esta en rango de: municipio

Via

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#Via>

Se ha reutilizado la definición de Municipio proporcionada por vocab.linkeddata.es [13]. Vía de comunicación construida para la circulación. En su definición según el modelo de direcciones de la Administración General del Estado, Incluye calles, carreteras de todo tipo, caminos, vías de agua, pantanales, etc. Asimismo, incluye la pseudovía., es decir todo aquello que complementa o sustituye a la vía. En nuestro caso, este término se utiliza para hacer referencia a las vías urbanas. Representación numérica de la misma.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Gender

IRI: <https://schema.org/gender>

Género de la persona afectada.

Seguirá el formato definido por Schema.org

Se utilizarán las siguientes definidas en la clase:

<http://schema.org/Male>

<http://schema.org/Female>

<http://schema.org/Mixed>

Definida por:

<https://schema.org/gender>

2.1.2. Propiedades de datos

fecha

IRI: <http://vocab.ciudadesabiertas.es/def/transporte/accidente#fecha>

Fecha en la que ocurre el siniestro. Día, mes y año, sin incluir la hora del accidente.

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio: Accidente

Rango: xsd:date

hora

IRI: <http://vocab.ciudadesabiertas.es/def/transporte/accidente#hora>

Hora en la que ocurre el siniestro.

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio: Accidente

Rango: xsd:time

officialName

IRI: <http://www.geonames.org/ontology#officialName>

Definición reutilizada del Callejero de DatosAbiertos [1].
Un nombre en el idioma oficial local.

Definida por:

<http://www.geonames.org/ontology>

Dominio: Via

Rango: xsd:string

typicalAgeRange

IRI: <https://schema.org/typicalAgeRange>

Rango de edad en el que se encuentra la persona afectada.

Seguirá el siguiente formato definido por Schema.org:

`10-12` [16]

Definida por:

<https://schema.org/typicalAgeRange>

Dominio: PersonaAfectada

Rango: xsd:string

enCruce

IRI: <http://vocab.ciudadesabiertas.es/def/transporte/accidente#enCruce>

Si el accidente ocurrió en un cruce entre 2 o más vías.

Está representado como un integer ya que puede ser un cruce de múltiples calles. En caso de ser un valor booleano solo podría representarse la intersección entre calles. Esta propiedad representa el numero de calles asociadas. En caso de que no fuese cruce se le asignaría el valor 0, en los casos en los que si se asignaría 2, 3 o números sucesivos dependiendo del numero de calles de la intersección.

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio: Accidente

Rango: xsd:integer

identifier

IRI: <http://purl.org/dc/terms/identifier>

An unambiguous reference to the resource within a given context. Recommended practice is to identify the resource by means of a string conforming to an identification system. Examples include International Standard Book Number (ISBN), Digital Object Identifier (DOI), and Uniform Resource Name (URN). Persistent identifiers should be provided as HTTP URIs [18].

Definida por:

<http://purl.org/dc/elements>

Dominio: Accidente

Rango:

<http://www.w3.org/2000/01/rdf-schema#Literal>

codigoINE

IRI: <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#codigoINE>

Indicador de si las bicicletas disponen o no de un carril propio para su circulación.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio>

Dominio: Municipio

Rango: xsd:integer

2.1.3. Propiedades de objeto

hasPersonaAfectada

IRI: [http:](http://vocab.ciudadesabiertas.es/def/transporte/accidente#hasPersonaAfectada)

[//vocab.ciudadesabiertas.es/def/transporte/accidente#hasPersonaAfectada](http://vocab.ciudadesabiertas.es/def/transporte/accidente#hasPersonaAfectada)

Persona que se asocia a un accidente. Esta a su vez puede tener más características como por ejemplo el rol que tuvo (peatón, conductor), edad y género.

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio:

Accidente

Rango:

PersonaAfectada

tipoVehiculo

IRI:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente#tipoVehiculo>

Tipo de vehículo afectado, p.ej. Bicicleta, Bicicleta EPAC (pedaleo asistido). Se han definido los siguientes elementos:

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-vehiculo/BICICLETA>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-vehiculo/BICICLETA-EPAC>

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio: Accidente

Rango: concept

meteorologia

IRI:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente#meteorologia>

Condiciones ambientales que se dan en el momento del siniestro. Se han definido varios tipos posibles:

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/meteorologia/DESPEJADO>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/meteorologia/LLUVIA-DEBIL>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/meteorologia/LLUVIA-INTENSA>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/meteorologia/NUBLADO>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/meteorologia/GRANIZANDO>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/meteorologia/DESCONOCIDO>

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio: Accidente

Rango: concept

tipoAccidente

IRI:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente#tipoAccidente>

Tipo de accidente asociado. Se han definido para ello varios tipos posibles:

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-accidente/COLISION>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-accidente/COLISION-DOBLE>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-accidente/COLISION-MULTIPLE>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-accidente/ALCANCE>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-accidente/CHOQUE-NO-VEHICULO>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-accidente/ATROPELLO-PEATON>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-accidente/VUELCO>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-accidente/CAIDA>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-accidente/OTROS>

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio: Accidente

Rango: concept

lesividad

IRI: <http://vocab.ciudadesabiertas.es/def/transporte/accidente#lesividad>

Código que indica la gravedad del siniestro para la persona afectada.

Para su uso se han definido los siguientes elementos:

01 Atencion en urgencias sin posterior ingreso. - LEVE:

[http:](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/01)

[//vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/01](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/01)

02 Ingreso inferior o igual a 24 horas - LEVE:

[http:](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/02)

[//vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/02](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/02)

03 Ingreso superior a 24 horas. - GRAVE:

[http:](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/03)

[//vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/03](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/03)

04 Fallecido 24 horas - FALLECIDO:

[http:](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/04)

[//vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/04](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/04)

05 Asistencia sanitaria ambulatoria con posterioridad - LEVE:

[http:](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/05)

[//vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/05](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/05)

06 Asistencia sanitaria inmediata en centro de salud o mutua - LEVE:

[http:](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/06)

[//vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/06](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/06)

07 Asistencia sanitaria solo en el lugar del accidente - LEVE:

[http:](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/07)

[//vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/07](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/07)

14 Sin asistencia sanitaria:

[http:](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/14)

[//vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/14](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/14)

77 Se desconoce:

[http:](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/77)

[//vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/77](http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/77)

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio: Accidente

Rango: concept

gender

IRI: <https://schema.org/gender>

Género de la persona afectada.

Seguirá el formato definido por Schema.org [14] Se utilizarán las siguientes definidas en la clase:

<http://schema.org/Male>

<http://schema.org/Female>

<http://schema.org/Mixed>

Definida por:

<https://schema.org/gender>

Dominio: PersonaAfectada

Rango: Gender [15]

tipoPersAfect

IRI:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente#tipoPersAfect>

Persona a la que afecta el accidente. Puede ser Conductor, peatón, testigo o viajero. Se han definido los siguientes elementos:

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-pers-afect/CONDUCTOR>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-pers-afect/PEATON>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-pers-afect/TESTIGO>

<http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-pers-afect/VIAJERO>

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio: PersonaAfectada

Rango: concept

portal

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#portal>

Numero de la calle donde ha ocurrido el accidente, si procede.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Dominio: Accidente

Rango: Portal

ocurrioAccidente

IRI: <http://vocab.ciudadesabiertas.es/def/transporte/accidente#ocurrioAccidente>

Propiedad que permite, a partir de una vía, conocer los accidentes que han ocurrido en ella.

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio: Via

Rango: Accidente

ocurreEnVia

IRI: <http://vocab.ciudadesabiertas.es/def/transporte/accidente#ocurreEnVia>

Propiedad que permite conocer las vías asociadas a un accidente. Puede haber varias en el caso de que haya ocurrido en un cruce.

Definida por:

<http://vocab.ciudadesabiertas.es/def/transporte/accidente>

Dominio: Accidente

Rango: Via

municipio

IRI: <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#municipio>

Municipio al que pertenece un fenómeno geográfico o una entidad administrativa [12].

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio>

Dominio: Via

Rango: Municipio

portal

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#portal>

Portal asociado a un accidente.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Dominio: Accidente

Rango: Via

2.2. Vocabulario de CicloCarriles

Para el vocabulario asociado con los ciclocarriles para ciclistas se han obtenido los datos del portal de datos abiertos del ayuntamiento de Madrid [4], en el cual se muestran las calles que disponen de ciclocarriles y alguna de sus características.

La organización del conjunto de datos se hará siguiendo el diagrama 2.5

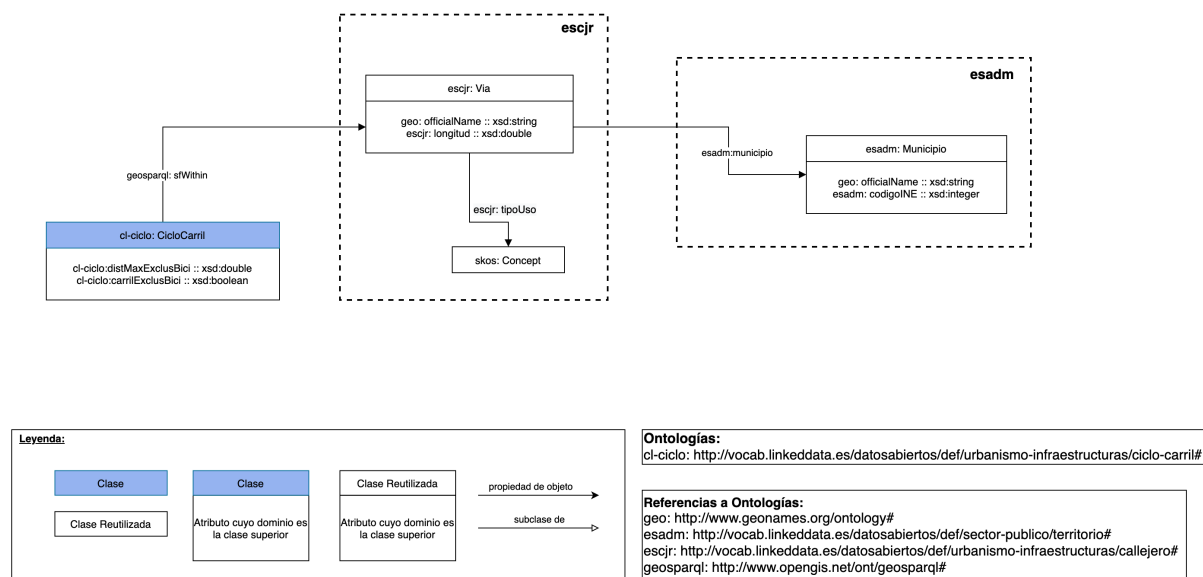


Figura 2.3: Diagrama de Ontología de Ciclocarriles.

Para la representación de los datos de ciclocarriles para ciclistas se han definido varias clases y propiedades. Se han reutilizado elementos ya definidos en el vocabulario de Callejero de ciudades abiertas [1] y de Territorio [12].

Se han añadido elementos como el identificador de vía y el municipio (que siempre será Madrid). El identificador de vía se añadirá para cada caso a partir del nombre. Para ello se hará una reducción del nombre de la vía a palabras clave, proceso detallado en la sección de Transformaciones de Vocabularios, y se cruzará con el dataset del callejero de Madrid [6].

Se ha optado por omitir la propiedad “MinSimpTol” debido a que no aporta valor al conjunto al tener solo 2 valores, 0 para calles sin carril bici y 0.20 para calles que si disponen de él, información que puede inferirse del campo “MaxSimpTol” (renombrada “distMaxExclusBici”), con valor 0 para el primer caso y valor distinto de 0 para el segundo. Para representar esto se ha añadido la propiedad “carrilExclusBici” con valor booleano indicando si dispone de ese carril exclusivo o no.

La fecha proporcionada por el ayuntamiento se ha omitido debido a que no se sabe con exactitud su significado. En caso de que fuese fecha de creación del ciclocarril se debería añadir en futuras actualizaciones del vocabulario, sin embargo al ser en todos los registros la misma cabe la posibilidad de que sea la fecha de inserción en el dataset, lo cual no aporta información relevante y podría inducir a errores.

Para este caso el valor de “tipoUso” será siempre CICLOCARRIL.

Se han transformado los valores del campo longitud a metros (expresados en kilómetros en los datos de origen) para poder reutilizarlos con más facilidad.

En la siguiente tabla se muestran los Namespaces usados.

	http://vocab.ciudadesabiertas.es/def/ciclo-carril#
esadm	http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#
escjr	http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
schema	http://schema.org#
www-geonames-org	http://www.geonames.org/
xml	http://www.w3.org/XML/1998/namespace
xsd	http://www.w3.org/2001/XMLSchema#

Figura 2.4: Namespaces usados para Ciclocarriles

Cabe destacar en este conjunto de datos la ausencia de la propiedad TipoVia. En este caso los nombres de las calles contenían únicamente palabras clave y privaban de la capacidad de obtener este atributo. En cambio si dispone de tipoUso, propiedad que indica si es una calle peatonal o ciclocarril.

Debido a la falta de disponibilidad de una leyenda o información proporcionada por el ayuntamiento de Madrid, para este conjunto de datos no se han podido conocer con exactitud el significado de todos sus datos y por tanto algunos como la dirección no han podido añadirse al modelo. En un futuro si se tratase información de otras fuentes o se añadiese una documentación detallada para este dataset, si se podría añadir esa propiedad.

2.2.1. Clases

CicloCarril

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/ciclo-carril#CicloCarril>

Vía con uno o más carriles destinados al tránsito de ciclistas. No necesariamente exclusivos para el tránsito de bicicletas, pero si con señalización y limitaciones adaptadas para ello.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/ciclo-carril>

Pertenece A: Via

Via

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#Via>

Se ha reutilizado la definición de Municipio proporcionada por vocab.linkeddata.es [13] Vía de comunicación construida para la circulación. En su definición según el modelo de direcciones de la Administración General del Estado, Incluye calles, carreteras de todo tipo, caminos, vías de agua, pantanales, etc. Asimismo, incluye la pseudovía., es decir todo aquello que complementa o sustituye a la vía. En nuestro caso, este término se utiliza para hacer referencia a las vías urbanas. Representación numérica de la misma.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Municipio

IRI: <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#Municipio>

Se ha reutilizado la definición de Municipio proporcionada por vocab.linkeddata.es [12]. Un Municipio es el ente local definido en el artículo 140 de la Constitución española y la entidad básica de la organización territorial del Estado según el artículo 1 de la Ley 7/1985, de 2 de abril, Reguladora de las Bases del Régimen Local. Tiene personalidad jurídica y plena capacidad para el cumplimiento de sus fines. La delimitación territorial de Municipio está recogida del Registro Central de Cartografía del IGN. Su nombre, que se especifica con la propiedad `dct:title`, es el proporcionado por el Registro de Entidades Locales del Ministerio de Política Territorial, en <http://www.ine.es/nomen2/index.do>

Definida por:

<http://purl.org/derecho/vocabulario>

<http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio>

<http://www.ign.es/ign/resources/acercaDe/tablon/ModeloDireccionesAGE>

Esta en rango de:

municipio

2.2.2. Propiedades de datos

longitud

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#longitud>

Longitud de la calle descrita. Esta propiedad está referida a la vía que contiene un ciclocarril (calle completa).

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Dominio: Via

Rango: xsd:double

distMaxExclusBici

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/ciclo-carril#distMaxExclusBici>

Longitud del carril exclusivo de bicicletas dentro de la calle. En caso de que no haya ciclocarril, el valor será 0.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/ciclo-carril>

Dominio: CicloCarril

Rango: xsd:double

officialName

IRI: <http://www.geonames.org/ontology#officialName>

Definido en el callejero de DatosAbiertos [1]. Un nombre en el idioma oficial local.

Definida por:

<http://www.geonames.org/ontology>

Dominio: Via

Rango: xsd:string

carrilExclusBici

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/ciclo-carril#carril-exclus-bici>

Indicador de si las bicicletas disponen o no de un carril propio para su circulación.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/ciclo-carril>

Dominio: CicloCarril

Rango: xsd:boolean

codigoINE

IRI: <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#codigoINE>

Indicador de si las bicicletas disponen o no de un carril propio para su circulación.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio>

Dominio: Municipio

Rango: xsd:integer

2.2.3. Propiedades de objeto

municipio

IRI: <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#municipio>

Municipio al que pertenece un fenómeno geográfico o una entidad administrativa.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio>

Dominio: CicloCarril

Rango: Municipio

tipoUso

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#tipoUso>

Identificador del tipo de uso que puede tener la calle. Se han definido 2 clases para ello:

- <http://vocab.ciudadesabiertas.es/kos/urbanismo-infraestructuras/calle/tipo-uso/CICLOCALLE>
- <http://vocab.ciudadesabiertas.es/kos/urbanismo-infraestructuras/calle/tipo-uso/PEATONAL>

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Dominio: Via

Rango: concept

2.3. Vocabulario de Calles Tranquilas

Para el vocabulario asociado con las calles tranquilas para ciclistas en el ayuntamiento de Madrid se ha elegido la fuente de Datos.Madrid [5]. Proporcionada por el ayuntamiento de Madrid, se muestran las calles más apropiadas para el tránsito de ciclistas. No se dan especificaciones de los criterios utilizados que han llevado a estas calles a formar parte de la lista. Sin embargo, si se puede observar en algunas de ellas ciertos patrones, como que no forman parte de las vías principales de la capital y que son poco transitadas. La misma web ofrece un archivo KML y permite que se puedan mostrar sobre un mapa en Open Street Map [7], lo cual proporciona una idea general de su disposición y posibles criterios utilizados.

Para la representación de los datos de calles tranquilas para ciclistas no se ha definido un modelo sino que se han realizado modificaciones al ya existente para Vías. De esta forma se ha abierto una solicitud al repositorio relativo al vocabulario de Callejero [1] y se han añadido las propiedades necesarias para representar los datos aquí dispuestos. Esta propuesta de modificación se detallará en capítulos posteriores.

Sin embargo, si se ha hecho uso del dataset proporcionado por el ayuntamiento para la aplicación final que se está construyendo en el contexto de este trabajo. Se han realizado ciertas modificaciones con respecto al dataset original para que puedan utilizarse sus datos más eficazmente.

Se ha optado por la separación del tipo de vía del nombre, conservándola en éste y creando una nueva propiedad que permita saber su clase. Algunos ejemplos serían Calle, Avenida, Plaza... Se ha añadido el identificador de la vía, obtenido a partir del nombre y cruzado con el dataset del callejero de Madrid [6]. El identificador permitirá hacer búsquedas mucho más rápidas sobre los datos en caso de querer hacerla filtrando por la calle, que es el caso de la aplicación final que se desea realizar para este proyecto.

En este caso el municipio será siempre Madrid, pero en caso de que se quisiera reutilizar en otros proyectos a mayor escala sería necesario conocer la zona geográfica donde se encuentra, por tanto también se ha añadido, aunque con el valor fijo de Madrid, que corresponde al código 28079, proporcionado por el Instituto Nacional de Estadística[9].

Estas transformaciones se detallarán más adelante en la sección: Transformaciones en los vocabularios.

Se ha optado por omitir la propiedad ID_TIPO, ya que representa lo mismo que TX_CAPA (el uso que tiene la vía) y se ha optado por la segunda por ser representado con texto, más visual y representativo a la hora de su utilización.

Debido a la falta de disponibilidad de una leyenda o información proporcionada por el ayuntamiento de Madrid, para este conjunto de datos no se han podido conocer con exactitud el significado de sus datos. Ciertos datos no han podido añadirse al modelo por dicho impedimento.

2.4. Vocabulario de Callejero (Propuesta)

La objetivo final de este proyecto es construir una aplicación que permita cruzar datos relativos a bicicletas en Madrid para así obtener la seguridad de una ruta. Muchos de ellos no pueden considerarse propios de las bicicletas sino que forman parte de las vías por las que éstas van a circular. La necesidad de estos datos y las posibles utilidades que podrían tener para muchos otras otras aplicaciones y tratamientos han llevado a realizar una propuesta de modificación al propio vocabulario de callejero ya creado.

Aun siendo cambios menores que no afectan a la estructura ni a la base del mismo, si es necesario realizar esta ampliación para que pueda abarcar más información y pueda tener muchas más aplicaciones.

Para ello se ha abierto una petición en Github para este repositorio y una vez aceptada formaría parte del modelo. El enlace del ISSUE creado es `TODO:inserteaquíurldepropuesta!!!`.

En el diagrama 2.5 se muestran las modificaciones propuestas en rojo. También como parte del proyecto se ha modificado el gráfico para que siga el formato de las nuevas ontologías que se estaban creando en ciudadesabiertas [19]

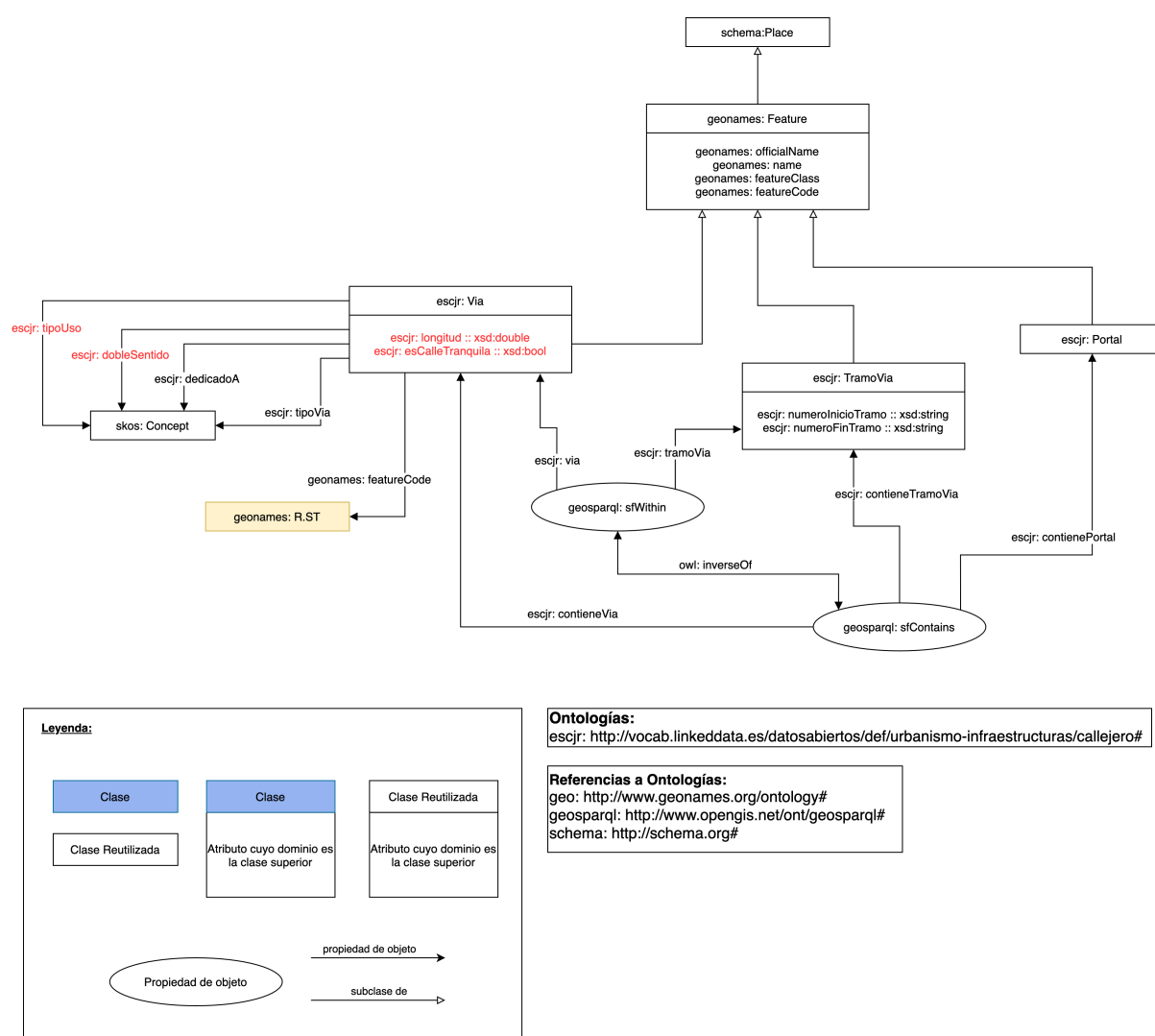


Figura 2.5: Diagrama de Ontología de Callejero

Se han añadadido las propiedades de objeto tipoUso y dobeSentido. La primera de ellas ha sido utilizada en los datasets de CicloCarriles y de CallesTranquilas y se refiere al uso dado (CicloCarril o Peatonal). La segunda únicamente en CallesTranquilas y como su nombre indica, representa el sentido único o doble de una calle.

Se han añadido además dos propiedades de datos para la clase Via: longitud y esCalleTranquila. La primera se representa con un valor numérico decimal y formaba parte del dataset de Ciclocarriles y de CallesTranquilas. La segunda representa con un valor booleano si es o no una calle tranquila para ciclistas siguiendo el criterio del ayuntamiento.

En las siguientes secciones se detallarán estas propiedades incluidas en la propuesta de modificación.

2.4.1. Propiedades de datos

esCalleTranquila

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#esCalleTranquila>

Propiedad que indica si una vía es calle tranquila o no para bicicletas. Vías con poco tráfico, mucha visibilidad o con mucho porcentaje de accidentes pueden ser algunos de los criterios seguidos para esta valoración.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Dominio: Via

Rango: xsd:boolean

longitud

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#longitud>

Longitud de la calle o tramo de la calle descrito. Su unidad de medida es el metro aunque en muchos casos puede venir representado como Shape_leng.

Definida por:

<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Dominio: Via

Rango: xsd:double

2.4.2. Propiedades de objeto

tipoUso

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#tipoUso>

Identificador del tipo de uso que puede tener la calle. Se han definido 2 clases para ello:

- <http://vocab.ciudadesabiertas.es/kos/urbanismo-infraestructuras/calle/tipo-uso/CICLOCALLE>
- <http://vocab.ciudadesabiertas.es/kos/urbanismo-infraestructuras/calle/tipo-uso/PEATONAL>

Definida por: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Dominio: Via

Rango: concept

dobleSentido

IRI: <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#dobleSentido>

Propiedad que determina si una vía es de sentido único o doble.

<http://vocab.ciudadesabiertas.es/kos/urbanismo-infraestructuras/calle/doble-sentido/SENTIDO-UNICO>

<http://vocab.ciudadesabiertas.es/kos/urbanismo-infraestructuras/calle/doble-sentido/DOBLE-SENTIDO>

Definida por:
<http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>

Dominio: Via

Rango: concept

Capítulo 3

Transformaciones en los Datasets

Partiendo de los datos proporcionados por el ayuntamiento de Madrid y con el fin de plasmar las estructuras antes definidas, se han realizados ciertos cambios con respecto al dataset original. Campos añadidos, modificaciones o transformaciones en los ya existentes son algunos de los motivos para realizarlos.

Cabe destacar que antes de hacer cualquier modificación o tratamiento se deben transformar a codificación ISO-8859-1. En los datasets utilizados para este proyecto, obtenidos de la web de datos abiertos del ayuntamiento de Madrid [20], se han observado muchos problemas en torno a su codificación.

Para este proyecto, como ya se ha mencionado anteriormente, se han elegido tres conjuntos de datos para evaluar la seguridad de las rutas: Ciclocarriles [4], Calles tranquilas [5] y accidentes de bicicletas [3]. Todos ellos proporcionados por el ayuntamiento de Madrid.

Algunas de las propiedades que se han definido en los vocabularios antes mencionados no formaban parte de los datasets originales. Dichos datos se han considerado necesarios para la realización del proyecto y han tenido que ser inferidos de la información ya existente.

En este capítulo se detallarán los procesos que se han seguido para obtener dicha información y que podrían ser utilizados para obtener otras propiedades (como es el caso del tipo de vía, necesario para obtener el identificador de vía, aunque finalmente no ha sido requerido en la aplicación final de este proyecto).

Gran parte de este proceso ha sido la transformación de un lenguaje natural o abreviado utilizado para el nombrado de calles (con elementos como C/, Plza, Glta) , en uno estándar que permita poder relacionarlos con otros datasets y otros registros escritos por otras personas.

Para el caso del dataset de Accidentes el proceso ha sido más complejo ya que hay un mayor uso de abreviaturas y más cantidad de erratas (posiblemente no siga un proceso automático, sino que haya sido obtenido de informes policiales escritos manualmente). Además, muchos de los registros presentes indicaban la existencia de accidentes en cruces de vías, propiedad definida en el vocabulario pero no contemplada en el conjunto de datos original. Por lo tanto, se ha inferido esta propiedad a partir del nombre de la calle y se ha separado en las distintas calles que lo componen.

Cabe destacar que todos los datasets han seguido el mismo tratamiento (excepto en la obtención de los cruces, exclusiva de los accidentes). De esta forma hay una probabilidad mucho más alta

de que los distintos elementos de los datasets coincidan entre si y se pueda obtener conocimiento de estos datos inicialmente separados.

En primer lugar, para evitar errores en el tratamiento, se creará una nueva columna para los nombres de las calles. De esta forma se copiarán los nombres originales en esta y se modificarán. Para el cruce entre datasets es preferible utilizar estos nuevos nombres creados, ya que se han realizado los mismos cambios en todos los conjuntos. En cambio, para su representación de cara al público es preferible el original, ya que al segundo se le habrán eliminado los conectores, estará en mayúsculas y puede que contenga errores. El segundo es más útil para su uso en sistemas informáticos y el primero para su visualización de cara a usuarios.

En esta sección se ha utilizado el dataset del callejero de Madrid [6] debido a que es necesario para obtener el identificador de las vías a partir del nombre de estas. Es por ello que, aunque no forme parte de la definición de los vocabularios ni se vaya a usar de forma directa en la aplicación final, se realicen transformaciones sobre él. Ha sido necesario corregir varios errores y realizar las mismas transformaciones que a los demás conjuntos de datos, como ya se ha explicado anteriormente, para que haya una mayor coincidencia entre ellos.

3.1. Modificaciones manuales en Datasets

Como ya se ha descrito anteriormente los ficheros se deben visualizar en codificación ISO-8859-1. Esto no resuelve todos los errores debido a que ciertas palabras siguen mostrándose de manera incorrecta. Para resolver este inconveniente se han realizado modificaciones automáticas (descritas en los siguientes capítulos, para los elementos más comunes) y otras manuales (en elementos no repetitivos y fáciles de cambiar). Estos errores se encuentran mayoritariamente en el dataset de Ciclocarriles, para el cual no fue posible encontrar la codificación adecuada y no contenía demasiados registros (unos 150).

- Cambios realizados de forma manual al conjunto de datos de Ciclocarriles:

- ln 52: M/ndez ?lvaro → Méndez Álvaro
- ln 64-65: Men/ndez Pelayo → Menéndez Pelayo
- ln 72-73: Ortega y Gasset → Jose Ortega y Gasset (Igual al nombre del Callejero de Madrid)
- ln 103: Donoso Cort/s → Donoso Cortés
- ln 112: Gral Moscardæ → Gral Moscardó
- ln 114: Camilo Jos/Cela - Azcona → Camilo José Cela (También eliminado Azcona ya que no esta previsto la existencia de cruces)
- ln 117: Gral Yag>e → Gral Yagüe
- ln 125: MARQUÉS DE VIANA - SOR ANGELA DE LA CRUZ → Dividido en 2 registros con características similares.

- Cambios realizados de forma manual al dataset de CallesTranquilas:

- ln 1292-1292: Calle de la Cooperativa ElÚctrica → Calle de la Cooperativa Eléctrica
- ln 1822-1823: Doctor MartYn ArÚvalo → Doctor Martín Arévalo
- ln 1919-1921: Errores en el formato del csv o de codificación. Mismo registro en varias lineas.
- ln 1673: AVENIDA ALBUFERA CON FELIPE ÁLVAREZ → AVENIDA ALBUFERA
- ln 2040: ENLACE CALLE AMERICIO CON MADRID RÍO → CALLE AMERICIO - ln 1716: PARQUE LINEAL DE PALOMERAS CON GONZÁLEZ DÁVILA → Eliminada (peatonal) -ln 1846: MARMOLINA CON AVENIDA COMUNIDADES → Eliminada

Para la realización de estos cambios se ha observado el mapa proporcionado por el ayuntamiento [21] y se ha determinado la mejor forma de representar los datos. En los casos que han sido eliminadas o que formaban partes de cruces y se ha mantenido únicamente una calle, se ha tomado en consideración el mapa proporcionado en el url anterior y se ha considerado que era la mejor manera de representar esos datos o que no eran relevantes.

- Cambios realizados de forma manual al dataset del Callejero:

- 201600;CALLE;DEL;COMANDANTE ZORITA;AVIADOR ZORITA;6;1;59;2;50 → Igual que el registro “Aviador Zorita”
- 334200;CALLE;DE;GENERAL YAGUE;GENERAL YAGÜE;6;1;57;4;76 → Igual que el registro “San German”. Cambio de nombre de la vía posterior a la realización de varios dataserts https://es.wikipedia.org/wiki/Calle_de_San_Germán.
- 331600;CALLE;DE;GENERAL MOSCARDO;GENERAL MOSCARDÓ;6;1;39;2;34 → Igual que el registro “Edgar Neville”. Cambio de nombre de la via posterior a la realizacion de varios dataserts <https://www.elmundo.es/madrid/2017/05/31/592dbf00e2704ed5058b4688.html>.
- 765800;RONDA;DE;RONDA VALENCIA;RONDA VALENCIA;1;;2;18 → Se considera nombre completo “Ronda de Valencia”, y no separado como muestra inicialmente

Estos cambios se realizan directamente en el dataset del callejero ya que pueden ser aplicables a todos los datasets. Elementos que se consideran básicos en casos concretos, calles nuevas o nuevos nombres (como es el caso de algunos referidos a personajes militares o políticos) cambiados en los últimos años, deben añadirse por si no han sido actualizados en algunos casos, conservando ambos.

Para ello se ha seguido la lista proporcionada por El Pais [22] y se han añadido tanto los cambios ya realizados, como los aprobados aun no actualizados en el dataset, para que

estén ambos nombres.

Los elementos añadidos son:

96200;CALLE;;BATALLA DE BELCHITE;BATALLA DE BELCHITE;2;1;15;2;22
917;PASEO;DEL;DOCTOR VALLEJO-NAJERA;DOCTOR VALLEJO-NÁJERA;2;1;61;2;56
356700;PLAZA;DE LOS;HERMANOS FALCO Y ALVAREZ;HERMANOS FALCÓ Y ÁLVAREZ;21;1;25;2;24
526000;PASEO;DE;MUÑOZ GRANDES;MUÑOZ GRANDES;11;1;53;2;64
329900;CALLE;DEL;GARCIA DE LA HERRANZ;GARCÍA DE LA HERRANZ;11;1;19;2;10
329700;CALLE;DEL;GENERAL FRANCO;GENERAL FRANCO;11;1;15;2;12
73600;PLAZA;;ARRIBA ESPAÑA;ARRIBA ESPAÑA;5;1;13;2;12
123600;CALLE;;CAIDOS DE LA DIVISION AZUL;CAÍDOS DE LA DIVISIÓN AZUL;5;1;15;2;28
82000;PLAZA;;AUNOS;AUNÓS;5;1;11;2;10
328950;CALLE;DE LA;POETA ANGELA FIGUERA;POETA ÁNGELA FIGUERA;7;1;41;2;22
329400;CALLE;DE;GENERAL DAVILA;GENERAL DÁVILA;7;1;15;2;12
419300;CALLE;DE;JUAN VIGON;JUAN VIGÓN;7;1;25;2;10
332950;CALLE;DEL;GENERAL RODRIGO;GENERAL RODRIGO;7;1;17;2;12
417850;PLAZA;;JUAN PUJOL;JUAN PUJOL;1;1;1;;
402600;CALLE;DE;JOSE LUIS DE ARRESE;JOSÉ LUIS DE ARRESE;15;1;91;2;66
48900;CALLE;DEL;ANGEL DEL ALCAZAR;ÁNGEL DEL ALCÁZAR;15;1;7;2;8
330300;CALLE;DEL;GENERAL KIRKPATRICK;GENERAL KIRKPATRICK;15;1;37;2;46
158300;PLAZA;DEL;CAUDILLO;CAUDILLO;8;1;5;2;4
609700;CALLE;;PRIMERO DE OCTUBRE;PRIMERO DE OCTUBRE;8;1;15;2;20
772400;PLAZA;DEL;VEINTIOCHO DE MARZO;VEINTIOCHO DE MARZO;8;1;11;2;10
137100;CALLE;DEL;CAPITAN CORTES;CAPITÁN CORTÉS;16;1;13;2;14
31000067;AVENIDA;DEL;ALCALDE CONDE MAYALDE;ALCALDE CONDE MAYALDE;8;;;;
28150;CALLE;DEL;ALGABEÑO;ALGABEÑO;16;1;125;2;192
329500;AVENIDA;DEL;GENERAL FANJUL;GENERAL FANJUL;10;1;185;2;144
331250;CALLE;DEL;GENERAL MILLAN ASTRAY;GENERAL MILLÁN ASTRAY;10;1;81;2;72
333250;CALLE;DEL;GENERAL SALIQUET;GENERAL SALIQUET;10;1;109;2;54
325200;CALLE;DE;GARCIA MORATO;GARCÍA MORATO;10;5;9;22;26
329850;CALLE;DEL;GENERAL GARCIA ESCAMEZ;GENERAL GARCÍA ESCÁMEZ;10;3;27;2;52
333000;CALLE;DEL;GENERAL ROMERO BASART;GENERAL ROMERO BASART;10;1;149;2;90
67700;AVENIDA;DEL;ARCO DE LA VICTORIA;ARCO DE LA VICTORIA;9;1;3;2;4
333200;PASEO;DEL;GENERAL SAGARDIA RAMOS;GENERAL SAGARDÍA RAMOS;9;1;7;2;24
31004081;GLORIETA;DE;RAMON GAYA;RAMÓN GAYA;9;;;;
144900;CALLE;DE;CARLOS RUIZ;CARLOS RUIZ;9;1;3;2;10
33025;CALLE;DEL;ALMIRANTE FRANCISCO MORENO;ALMIRANTE FRANCISCO MORENO;9;1;13;;
263650;PLAZA;DE;EMILIO JIMENEZ MILLAS;EMILIO JIMÉNEZ MILLAS;9;1;1;2;4
1887;CALLE;DEL;PUERTO DE LOS LEONES;PUERTO DE LOS LEONES;9;1;61;2;92
360800;CALLE;DE LOS;HEROES DEL ALCAZAR;HÉROES DEL ALCAZAR;13;;;2;12
166500;CALLE;DEL;CERRO DE GARABITAS;CERRO DE GARABITAS;13;1;17;2;12
220600;CALLE;DEL;CRUCERO BALEARES;CRUCERO BALEARES;13;1;25;2;16
338200;PLAZA;DEL;GOBERNADOR CARLOS RUIZ;GOBERNADOR CARLOS RUIZ;13;1;7;2;8
256300;CALLE;DE;EDUARDO AUNOS;EDUARDO AUNÓS;4;1;41;2;56
331500;PASAJE;DEL;GENERAL MOLA;GENERAL MOLA;4;1;9;2;6
357000;CALLE;DE LOS;HERMANOS GARCIA NOBLEJAS;HERMANOS GARCÍA NOBLEJAS;15;;;2;198
331800;CALLE;DEL;GENERAL ORGAZ;GENERAL ORGAZ;6;1;31;2;18
333900;CALLE;DEL;GENERAL VARELA;GENERAL VARELA;6;1;37;2;38
328800;CALLE;DEL;GENERAL ARANDA;GENERAL ARANDA;6;1;55;2;98
328900;ESCALINATA;DEL;GENERAL ARANDA;GENERAL ARANDA;6;;;;

Transformaciones en los Datasets

466800;CALLE;DE;MANUEL SARRION;MANUEL SARRIÓN;6;1;13;2;12

137400;CALLE;;CAPITAN HAYA;CAPITAN HAYA;6;1;65;2;66

293200;PLAZA;DE;FERNANDEZ LADREDA;FERNÁNDEZ LADREDA;11;3;5;;

293200;PLAZA;DE;FERNANDEZ LADREDA;FERNÁNDEZ LADREDA;12;1;1;2;2

Para su adición se han obtenido las propiedades correspondientes a su nombre antiguo de forma que correspondan a la misma vía.

3.2. Revisiones de abreviaturas y erratas

En primer lugar, es necesario que todas las palabras que signifiquen lo mismo tengan la misma nomenclatura. Para ello se ha observado el contenido de los datasets y se realizado un listado de las distintas abreviaturas que pueden ser utilizadas, las distintas formas de nombrado y ciertos elementos con significados similares que puedan ser nombrados de igual forma para un enlazado más eficaz.

Se han eliminado elementos considerados innecesarios y que podrían causar problemas a la hora de hacer cruces entre elementos. Por ejemplo, se dan casos de detallar el kilómetro de la vía donde ocurrió el accidente o el número de la farola más cercana. En calles con gran longitud puede que sea interesante esta información, pero se ha decidido omitirla del nombrado debido a que, al ser un proceso semi-automático, era considerado como una vía y generaba muchos problemas. Además, son elementos que pertenecen al conjunto de datos de accidentes, el cual ya contenía una propiedad “Portal” y podría considerarse que representa lo mismo que el número de farola donde ocurrió. Al no haber muchos registros que contenían esta nomenclatura se ha optado por eliminarlo y no crear una nueva propiedad que represente dicho valor, ya que se puede observar que es algo atípico y que en la gran mayoría de registros no se podría obtener.

En esta sección también se detalla parte de las transformaciones para la propiedad “esCruce”. Debido a la distinta nomenclatura utilizada en el nombrado de los accidentes, en muchos casos los cruces se representan con un guion “-”, la expresión “calle1 CRUCE calle2”, una barra “/”, etc. Esto dificulta su tratamiento y obtención de las distintas vías implicadas, por lo tanto para estos casos se ha optado por expresarlos de la forma: “calle1 / calle2”. Parte de estas transformaciones se encuentran en este apartado debido a que son cambios muy básicos y deben ejecutarse antes de algunos otros, sin embargo más adelante se detallará su tratamiento.

Para dichos cambios se ha definido el siguiente código:

Listing 3.1: Función cambiosBasicos

```
1
2 def cambiosBasicos(nombreCarpeta="", nombreSinCsv="", nRowNombre=-1, nFileIni = "-1", nFileFin = "-1", separarCruces
  = False):
3     with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
4         csvreader = csv.reader(csvfile, delimiter=";")
5         file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
6         primeraLinea = True
7         for row in csvreader:
8             if (primeraLinea):
9                 fila = ";".join(row)
10                primeraLinea = False
11            else:
12                nombreVia = row[nRowNombre].upper()
13                if (nombreVia.__contains__("C/")):
14                    nombreVia = nombreVia.replace("C/", "C/ ")
15                if (separarCruces and findall('CRUCE\s.*\sCON\s', nombreVia) != []): # Escritura estandar de "Cruce"
16                    nombreVia = nombreVia.replace(" CON ", " / ")
17                    nombreVia = nombreVia.replace("CRUCE ", "")
18                if (separarCruces and findall('\sCON\s', nombreVia) != []): # Escritura estandar de "Cruce"
19                    nombreVia = nombreVia.replace(" CON ", " / ")
20                if (nombreVia.__contains__("", CALLE")):
21                    nombreVia = nombreVia.replace(" ", CALLE, " / CALLE")
22                if (nombreVia.__contains__(" -AV. ")):
23                    nombreVia = nombreVia.replace("-AV. ", " / AVENIDA")
24                if (nombreVia.__contains__(" -CARRET. ")):
25                    nombreVia = nombreVia.replace("-CARRET. ", " / CARRET.")
26
27                if (nombreVia.__contains__("", FRENTE)): # Solo si contiene coma
28                    posIni = nombreVia.index(" ", FRENTE)
29                    nombreVia = nombreVia.replace(nombreVia[posIni:], "") # Eliminarlo
30                if (nombreVia.__contains__("FAROLA ") or nombreVia.__contains__("FAROLAS ")):
31                    posIni = 0
32                    if (nombreVia.__contains__("FAROLA ")):
33                        posIni = nombreVia.index("FAROLA ")
34                    if (nombreVia.__contains__("FAROLAS ")):
35                        posIni = nombreVia.index("FAROLAS ")
36                    if (nombreVia[posIni:].__contains__(" ")):
37                        posFin = nombreVia[posIni:].index(" ")
38                    nombreVia = nombreVia.replace(nombreVia[posIni:posFin], "")
```



```

39         elif(nombreVia[posIni:].__contains__(" / ")):
40             posFin = nombreVia[posIni:].index("/")
41             nombreVia = nombreVia.replace(nombreVia[posIni:posFin], "")
42         else:
43             nombreVia = nombreVia.replace(nombreVia[posIni:], "") # Eliminarlo
44         if(nombreVia.__contains__("S/N")): # P.ej. SAN FRANCISCO S/N, casos que generan error
45             nombreVia = nombreVia.replace('S/N', "")
46         if (nombreVia.__contains__("KM-0")): # Palabras con numero como KM-0 (quitar)
47             nombreVia = nombreVia.replace('KM-0', "")
48         if (nombreVia.__contains__("PKM")): # Eliminar PKM
49             nombreVia = nombreVia.replace('PKM', "")
50         if (nombreVia.__contains__("C/ ")):
51             nombreVia = nombreVia.replace("C/ ", "CALLE ")
52         if (findall('[A-z, Ã, 0-9, Ã-Ãz]{1}[/[A-z]{1}]', nombreVia) != []): # Escritura estandar de "Cruce"
53             nombreVia = nombreVia.replace("/", " / ")
54         # C/ MONASTERIO DE ARLANZA-AV. SANTUARIO DE VALVERDE
55         if (findall('[A-z, Ã, 0-9, Ã-Ãz]{1}[/]', nombreVia) != []): # Escritura estandar de "Cruce"
56             nombreVia = nombreVia.replace("/", " / ")
57         if (findall('[A-z, Ã-Ãz]{1}', nombreVia) != []): # Escritura estandar de "Cruce"
58             nombreVia = nombreVia.replace("/", " / ")
59         if (findall('[?!\\d][\\-]{1}[?!\\d]', nombreVia) != []): # Evitar palabras con numeros como M-30
60             arr1 = nombreVia.replace("-", " - ").split()
61             if(separarCruces and getTipoVia(palabrasMalEscritas(arr1[arr1.index("-")+1])) != ""): # Si es
62                 palabra clave indicadora de nueva via, es Cruce
63                 nombreVia = nombreVia.replace("-", " / ")
64             if (findall('[?!\\d][\\-]{1}[?!\\d]', nombreVia) != []): # Palabras con numero como M-30
65                 nombreVia = nombreVia.replace("-", " ")
66
67         if (nombreVia.__contains__("' '")): #Quitar comillas
68             nombreVia = nombreVia.replace(' ', ' ')
69         if (nombreVia.__contains__("PASO ELEVADO")):
70             nombreVia = nombreVia.replace('PASO ELEVADO', 'PASO_ELEVADO')
71         elif (nombreVia.__contains__("SENDA CICLABLE")):
72             nombreVia = nombreVia.replace('SENDA CICLABLE', 'SENDA_CICLABLE')
73
74         #Cambios varios en nueva funcion
75         nombreVia = palabrasMalEscritas(nombreVia)
76
77         row[nRowNombre] = nombreVia
78         fila = ",".join(row)
79         file.write(fila + os.linesep)
80     file.close()

```

La finalidad de la función definida en 3.1 es modificar ciertas palabras para que posteriormente se puedan entender mejor e inferir información a partir de ellas sin llegar a errores.

Es el caso por ejemplo de los cruces. Se pueden escribir de múltiples formas, pero en el tratamiento que vamos a realizar será del modo “calle1 / calle2“. Para conseguir ese formato han de modificarse otros nombres como por ejemplo “CRUCE calle1 CON calle1“ para que posteriormente las funciones sean aplicables a estos casos.

También se eliminan elementos como “S/N“ (Sur/Norte) que no tienen excesiva relevancia, no forman parte del nombre, pero en cambio si pueden llegar a producir errores graves al contener una barra y poder considerarse un cruce o información relevante en el nombrado de una calle. Otras transformaciones serian en palabras que consideramos clave, por ejemplo Paso elevado o Senda Ciclable, a las cuales se les considerará tipo de vía, que sean formadas como una única palabra, facilitando así su posterior búsqueda y que no se cometa el error de incluirlas en las palabras clave del nombre de la vía.

También, de nuevo debido a la importancia que tienen guiones o barras en la detección de elementos inusuales o cruces, para carreteras como M-30, M-40 o elementos como KM-0 se les eliminará el guion, considerándolos de esa forma una única palabra. Del mismo modo también se han eliminado los indicadores de la farola donde ocurre el acontecimiento.

Por último, se llama a la función palabrasMalEscritas, definida en 3.2 la cual es en parte una continuación de esta, aunque para casos más concretos.

Listing 3.2: Función palabrasMalEscritas

```

1
2 def palabrasMalEscritas(nombreCalle = ""): #Arreglar errores de codificación y abreviaturas
3     if(nombreCalle == ""):
4         return ""
5     if (nombreCalle.__contains__(" ")):
6         nombreCalle = nombreCalle.replace(" ", " ")
7     if (nombreCalle.__contains__("(")):
8         nombreCalle = nombreCalle.replace("(", " ( ")
9
10    nuevaPalabra = ""
11    for palabra in nombreCalle.split():
12        palabra = palabra.replace(" ", "").upper()
13        if(palabra == "PNTE."):
14            palabra = "PUENTE"
15        elif (palabra == "CÑADA." or palabra == "CÑADA"):
16            palabra = "CAÑADA"
17        elif (palabra == "AVDA." or palabra == "AVDA" or palabra == "AV." or palabra == "AV"):
18            palabra = "AVENIDA"
19        elif (palabra.__contains__("AV.")):
20            palabra = palabra.replace("AV.", "AVENIDA")
21        elif (palabra == "JARDÍN" or palabra == "JARDINES"):
22            palabra = "JARDIN"
23        elif (palabra == "CUSTA." or palabra == "CUSTA"):
24            palabra = "CUESTA"
25        elif (palabra == "POLÍGONO" or palabra == "POLIG."):
26            palabra = "POLIGONO"
27        elif (palabra == "GALERÍA"):
28            palabra = "GALERIA"
29        elif (palabra == "PLAZA."):
30            palabra = "PLAZA"
31        elif (palabra == "PISTA."):
32            palabra = "PISTA"
33        elif (palabra == "CMNO." or palabra == "CMNO"):
34            palabra = "CAMINO"
35        elif (palabra == "BULEV."):
36            palabra = "BULEVAR"
37        elif (palabra == "RONDA."):
38            palabra = "RONDA"
39        elif (palabra == "GTA." or palabra == "GTA"):
40            palabra = "GLORIETA"
41        elif (palabra == "CUSTA." or palabra == "CUSTA"):
42            palabra = "CUESTA"
43        elif (palabra == "PQUE." or palabra == "PQUE"):
44            palabra = "PARQUE"
45        elif (palabra == "CTRA." or palabra == "CTRA" or palabra == "CRA." or palabra == "CARRET."):
46            palabra = "CARRETERA"
47        elif (palabra.__contains__("CARRET.")):
48            palabra.replace("CARRET.", "CARRETERA")
49        elif (palabra == "AUTOV." or palabra == "AUTOV"):
50            palabra = "AUTOVIA"
51        elif (palabra == "CRUCE."):
52            palabra = "CRUCE"
53        elif (palabra == "ANILLO."):
54            palabra = "ANILLO"
55        elif (palabra == "PASEO."):
56            palabra = "PASEO"
57        elif(palabra == "TRVA."):
58            palabra = "TRAVESIA"
59        elif(palabra == "ESTFE."):
60            palabra = "ESTACION_FERROCARRIL"
61        elif (palabra == "P?"):
62            palabra = "PLAZA"
63        elif (palabra == "PZA."):
64            palabra = "PLAZA"
65        elif (palabra == "CÁZ" or palabra == "CALL." or palabra == "CALL" or palabra == "C/" or palabra == "C?"):
66            palabra = "CALLE"
67        elif (palabra == "GRAL."):
68            palabra = "GENERAL"
69        elif (palabra == "STA." or palabra == "STA"):
70            palabra = "SANTA"
71        elif (palabra == "PTA."):
72            palabra = "PUERTA"
73        elif (palabra == "PALAC."):
74            palabra = "PALACIO"
75        elif (palabra == "METRO."):
76            palabra = "METRO"
77
78        elif(palabra.__contains__("Ŷ")):
79            palabra = palabra.replace("Ŷ", "Í")
80        elif (palabra.__contains__("Âa")):
81            palabra = palabra.replace("Âa", "Ñ")
82        elif (findall('([B-DF-HJ-NP-TV-Z]{1})SS', palabra) != [] or findall('SS([B-DF-HJ-NP-TV-Z]{1})', palabra) !=
83            []):
84            #P.ej. BSSRBARA ó NARVSSEZ ó GUZMSSN Ó "Ortega Y GASSET" Ó ALCALSS
85            palabra = palabra.replace("SS", "Â")
86        elif (findall('([a-z]{1})Â~N', palabra) != []): #P.ej. SAHAGÂ~N
87            palabra = palabra.replace("Â~N", "ÜN")
88        elif (palabra[0] == '?'): #P.ej. ?LVAREZ --> ? ASCII 63
89            palabra = palabra.replace("?", "Â")
90        elif (palabra.__contains__(chr(190))): #P.ej. MOSCARDÂž --> ASCII 190, PERÂžN
91            palabra = palabra.replace(chr(190), "Ô")
92        elif (palabra.__contains__(chr(179))): #P.ej. MOSCARDÂž --> ASCII 179, YAGÂžE

```

```
92     palabra = palabra.replace(chr(179), "Ü")
93     elif (findall('QUÜ', palabra) != []): # P.ej MarquÜs --> Marqués
94         palabra = palabra.replace("QUÜ", "QUÉ")
95     elif (palabra == "JESÁ'S"):
96         palabra = "JESÚS"
97
98     #Quitar numeros (posiblemente de portales, los nombres propios son en numeros romanos)
99     elif (findall('[0-9]{1}', palabra) != []):
100         for a in findall('[0-9]{1}', palabra):
101             palabra = palabra.replace(a, '')
102         if (findall('[A-Z, Á-Àz]{3}', palabra) == []):
103             # En el caso de que le acompañen 1 o 2 letras (letras de portal)
104             palabra = ""
105     nuevaPalabra = nuevaPalabra + " " + palabra
106     return nuevaPalabra
```

En este caso se vuelven a transformar palabras de forma que sea más clara y fácil su utilización. En primer lugar, se eliminan abreviaturas que han sido detectadas y se sustituyen por las palabras completas.

En segundo lugar, se corrigen errores de codificación. Aunque esto a priori no debería ocurrir, el dataset de ciclocarriles no fue posible descargarlo y tratarlo de forma correcta, por tanto muchos de sus elementos estaban corruptos. Aunque se tuvo que hacer algún cambio manual si fue posible determinar los elementos más comunes que habían sido modificados y de esta forma es posible hacer la gran mayoría de forma automática, y por si volviera a ocurrir con otro dataset.

Por último, se eliminan números y los portales de las calles. Además, se separan los paréntesis para que posteriormente se pueda eliminar la información contenida en ellos.

3.3. Obtención de valores para propiedades

En esta segunda sección ya se tienen los nombres de las calles con las palabras sin abreviaturas, erratas, errores de codificación y con los cruces escritos de forma estándar. A partir de este punto se comenzará a refinar esta información para obtener de ella las propiedades necesarias.

Para esta sección se van a detallar las transformaciones y comprobaciones que se han realizado para los cruces, la obtención del tipo de vía y obtención de palabras clave (necesarias para la obtención del identificador).

3.3.1. Propiedad esCruce

Esta propiedad es exclusiva para los accidentes. Como su nombre indica determina si ha ocurrido en una intersección de vías, detallando el número de ellas que lo componen.

Se ha realizado en dos pasos. En el primero se determina si contiene 2 o más vías. Esto se puede saber gracias a las transformaciones anteriores donde se han modificado los nombres para que sigan el formato deseado. Se le asignará el valor 1 en caso de ser así y 0 en caso contrario. Se ha utilizado el código dispuesto en 3.3.

Listing 3.3: Función `annadirEsCruceAccidentes`

```
1 import csv
2 import os
3 from re import findall
4
5 def annadirEsCruceAccidentes(nombreCarpeta = "", nombreSinCsv = "", nRowNombre = -1, nFileIni = "-1", nFileFin =
   "-1"):
6     with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
7         csvreader = csv.reader(csvfile, delimiter=";")
8         file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
9         primeraLinea = True
10        for row in csvreader:
11            if (primeraLinea):
12                fila = ";".join(row)
13                fila = fila + ";" + "esCruce"
14                primeraLinea = False
15            else:
16                fila = ";".join(row)
17                if (row[nRowNombre].__contains__(" / ")):
18                    fila = fila + ";" + "1"
19                else:
20                    fila = fila + ";" + "0"
21            file.write(fila + os.linesep)
22        file.close()
```

En el segundo paso se analiza el nombre del lugar donde ha ocurrido el accidente y se crea una lista con las distintas calles que lo componen. Posteriormente se crean tantas entradas como vías lo compongan y se les asignan las mismas propiedades (es el mismo accidente con el mismo número de expediente), únicamente se diferenciarán por la vía en la que ocurrió (aunque también se conservará el nombre inicial con el cruce). Posteriormente se modificará la propiedad `esCruce` asignándole el número de calles que componen esa intersección. De esta forma, en caso de necesitar conocer todos los registros que componen ese accidente, se podrá saber el número que buscar. Se ha utilizado el código dispuesto en 3.4 y 3.5.

Listing 3.4: Función `separarCucesAccidentes`

```
1
2 def separarCucesAccidentes(nombreCarpeta = "", nombreSinCsv = "", nRowNombre = -1, nFileIni = "-1", nFileFin = "-1"):
3     with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
4         csvreader = csv.reader(csvfile, delimiter=";")
5         file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
6         primeraLinea = True
7         nColPalClav = -1
8         nColEsCruce = -1
9         for row in csvreader:
```

```
10     if (primeraLinea):
11         fila = ";".join(row)
12         nColPalClav = row.index("CALLE")
13         nColEsCruce = row.index("esCruce")
14         primeraLinea = False
15         fila = ";".join(row)
16         file.write(fila + os.linesep)
17     elif(row[nColEsCruce] == "1"):
18         arrCalles = getArrCalles(row[nColPalClav])
19         # Se le asigna a cada registro una de las calles
20         for calle in arrCalles:
21             row[nColPalClav] = calle
22             row[nColEsCruce] = arrCalles.__len__().__str__()
23             fila = ";".join(row)
24             file.write(fila + os.linesep)
25     else:
26         fila = ";".join(row)
27         file.write(fila + os.linesep)
28 file.close()
```

Listing 3.5: Función getArrCalles

```
1 def getArrCallesRec(nombreDataset = ""):
2     listaNombres = []
3     if (nombreDataset.__contains__('/')):
4         n1 = nombreDataset.split().index('/')
5         txtRestante = " ".join(nombreDataset.split()[n1 + 1:])
6         nombre = " ".join(nombreDataset.split()[0:n1])
7         elemRecursiv = getArrCallesRec(txtRestante)
8         listaNombres = [nombre, elemRecursiv]
9     else:
10         listaNombres = [nombreDataset]
11
12     return listaNombres
13
14 def getArrCalles(nombreDataset = ""):
15     list1 = getArrCallesRec(nombreDataset)
16     txt = list1.__str__().replace('[', '').replace(']', '')
17     list3 = []
18     nombre=""
19
20     for elem in txt.split():
21         if(elem.__contains__(" ") and nombre != ""):
22             nombre = nombre + " " + elem.replace("'", "").replace(",", "")
23             list3.append(nombre)
24             nombre = ""
25         elif (elem.__contains__(" ") and nombre == "" and elem.replace("'", "", 1).__contains__(" ")):
26             nombre = elem.replace("'", "").replace(", ", " ")
27             list3.append(nombre)
28             nombre = ""
29         elif (elem.__contains__(" ") and nombre == ""):
30             nombre = elem.replace("'", "").replace(", ", " ")
31         else:
32             nombre = nombre + " " + elem
33     return list3
```

Para el dataset de accidentes siempre han de ejecutarse estas funciones antes ya que las siguientes parten de una calle única. Por ejemplo, en el caso del tipo de vía, no se puede obtener el tipo de un cruce, ha de hacerse a partir de la calle única.

3.3.2. Propiedad tipoVia

El tipo de vía es obtenido de nuevo a partir del nombre de la calle. Como se ha mencionado anteriormente, para el caso de accidentes, se tendrá que utilizar el nombre de la vía y no del cruce.

Listing 3.6: Función annadirTipoVia

```
1 def annadirTipoVia(nombreCarpeta = "", nombreSinCsv = "", nRowNombre = -1, nFileIni = "-1", nFileFin = "-1"):  
2 with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:  
3     csvreader = csv.reader(csvfile, delimiter=";")  
4     file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")  
5     primeraLinea = True  
6     for row in csvreader:  
7         if (primeraLinea):  
8             fila = ";".join(row)  
9             fila = fila + ";" + "tipoVia"  
10            primeraLinea = False  
11        else:  
12            tipoVia = getTipoVia(row[nRowNombre])  
13            fila = ";".join(row)  
14            fila = fila + ";" + tipoVia  
15            file.write(fila + os.linesep)  
16        file.close()
```

En primer lugar, se ejecuta la función 3.6. En ella es llamada getTipoVia. Esta segunda únicamente realiza una comprobación sobre el nombre para conocer si contiene las palabras clave: “CALLE”, “PASEO”, “PLAZA”, “GLORIETA”, “RONDA”, “CAMINO”, “PISTA”, “ANILLO”, “CRUCE”, “AUTOVIA”, “CARRETERA”, “PARQUE”, “CUESTA”, “CAÑADA”, “AVENIDA”, “BULEVAR”, “JARDIN”, “PARTICULAR”, “POLIGONO”, “GALERIA”, “ESCALINATA”, “VIA”, “PASARELA”, “PASAJE”, “PUENTE”, “COSTANILLA”, “COLONIA”, “CARRERA”, “PLAZUELA”, “ACCESO”, “POBLADO”, “PASADIZO”, “TRASERA”, “SENDA”, “ARROYO”, “VALLE”, “AEROPUERTO”, “PASO_ELEVADO”, “SENDA_CICLABLE”, “PASAJE”. En el caso de que contenga algunas de ellas, serán asignadas a esta columna. En caso contrario se añadirá un valor vacío.

3.3.3. Propiedad typicalAgeRange

Esta propiedad ha de seguir un formato definido por schema.org. Dicha representación debería ser por ejemplo 30-34, en cambio en el dataset original de accidentes viene representado como “DE 30 A 34 AÑOS”. Para dicha transformación se ha definido el siguiente código.

Listing 3.7: Función getTypicalAgeRangeOk

```
1 def getTypicalAgeRangeOk(txtOld = ""):
2     if(txtOld.replace(" ", "") == ""):
3         return ""
4     if(txtOld.upper().__contains__("DESCONOCIDA")):
5         return ""
6     arrPal = txtOld.split()
7     # "DE 30 A 34 AÑOS"
8     try:
9         fIni = arrPal[1]
10        fFin = arrPal[3]
11        if (fIni > fFin):
12            fIni, fFin = fFin, fIni
13        txtFin = fIni + "-" + fFin
14    except IndexError:
15        txtFin = ""
16        print("Distinto formato Rango Edad: ", txtOld)
17    # En caso de que no siga el formato estandar
18    if(findall('[0-9]{1}-([0-9]{1})', txtFin) == []):
19        fIni = -1
20        fFin = -1
21        i = 1
22        for elem in arrPal:
23            if(fIni == fFin and findall('[0-9]{1}', elem) != []):
24                if (fIni == -1):
25                    fIni = elem
26                    fFin = elem
27            if(fIni != fFin):
28                if(fIni > fFin):
29                    fIni, fFin = fFin, fIni
30            return fIni + "-" + fFin
31        else:
32            return ""
33    return txtFin
```

3.3.4. Palabras Clave

El paso previo a cruzar los nombres de vías entre datasets será reducirlos a las palabras imprescindibles. Para ello se eliminan espacios, conectores y palabras conflicto. Esto último se compone sobre todo por elementos ya tratados anteriormente como son los tipos de vía. Una calle puede tener esa nomenclatura en un conjunto de datos y en otro contener su nombre sin su tipo, es por ello que se elimina para evitar posibles errores.

El código utilizado se detalla a continuación.

Listing 3.8: Función annadirPalabrasClave

```
1 def annadirPalabrasClave(nombreCarpeta = "", nombreSinCsv = "", nRowNombre = -1, nFileIni = "-1", nFileFin = "-1"):  
2     with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:  
3         csvreader = csv.reader(csvfile, delimiter=";")  
4         file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")  
5         primeraLinea = True  
6         for row in csvreader:  
7             if (primeraLinea):  
8                 fila = ";".join(row)  
9                 fila = fila + ";" + "palabrasClave"  
10                primeraLinea = False  
11            else:  
12                palabrasClave = quitarConectores(row[nRowNombre])  
13                palabrasClave = quitarPalabrasConflicto(palabrasClave)  
14                # Eliminar espacios innecesarios:  
15                palabrasClave = palabrasClave.replace(" ", "").replace(" ", " ")\  
16                .replace(" ", " ").replace(" ", " ")\  
17                #Eliminar espacios al principio y al final  
18                try:  
19                    if (len(palabrasClave)>1 and palabrasClave[0] == " "):  
20                        palabrasClave = palabrasClave.replace(" ", "", 1)  
21                    if (len(palabrasClave)>1 and palabrasClave[len(palabrasClave)-1] == " "):  
22                        palabrasClave = palabrasClave[0: len(palabrasClave)-1]  
23                except IndexError:  
24                    print("annadirPalabrasClave: Fuera de rango :: ", palabrasClave)  
25                fila = ";".join(row)  
26                fila = fila + ";" + palabrasClave  
27                file.write(fila + os.linesep)  
28            file.close()
```

Listing 3.9: Función quitarConectores

```
1 def quitarConectores(nombreVia = ""):  
2     nombreVia = nombreVia.upper()  
3     listaPosibles = ['DEL', 'DE', 'Y', 'LAS', 'LA', 'LOS', 'A', 'POR', 'CON',  
4                     'EL', 'EN', 'O', 'I', 'AL', ' ', '-', 'S/N', 'JUNTO', '_', 'SOBRE',  
5                     'ENTRE', 'FRENTE']  
6     for a in listaPosibles:  
7         if (nombreVia.__contains__(a)):  
8             if (nombreVia.__contains__(' ' + a + ' ')):  
9                 nombreVia = nombreVia.replace(' ' + a + ' ', " ")  
10            elif( # Para evitar que pertenezca a una palabra  
11                (findall('[A-z, 0-9, \s, Ã-Äž]{1}'+ a, nombreVia) == [] or findall('[A-z, 0-9, \s, Ã-Äž]{1}'+ a,  
12                    nombreVia) == [' '])  
13                and (findall(a + '[A-z, 0-9, \s, Ã-Äž]{1}', nombreVia) == [] or findall(a + '[A-z, 0-9, \s, Ã-Äž]{1}',  
14                    nombreVia) == [' '])):  
15                nombreVia = nombreVia.replace(a, " ")  
16     return nombreVia
```

Listing 3.10: Función quitarPalabrasConflicto

```
1 def quitarPalabrasConflicto(nombreVia = ""):  
2     nombreVia = nombreVia.upper()  
3     if (nombreVia == ""):  
4         return ""  
5     if (nombreVia.__contains__("\\")):  
6         nombreVia = nombreVia.replace("\\"", "")  
7     nombreVia = quitarTextoEntreParentesis(nombreVia)  
8  
9     listaPalabrasQuitar = ["CRUCE", "CALLE", "PASEO", "PLAZA", "GLORIETA", "CAMINO", "PISTA",  
10        "AUTOVIA", "CARRETERA", "CUESTA", "AVENIDA", "VIA", "PASARELA",  
11        "PASAJE", "COSTANILLA", "COLONIA", "POLIGONO", "CARRERA", "PLAZUELA", "BULEVAR",  
12        "ESCALINATA", "JARDIN", "PARTICULAR", "ACCESO", "POBLADO",  
13        "PASADIZO", "TRASERA", "SENDIA", "GALERIA", "VALLE",  
14        "PASO_ELEVADO", "SENDIA_CICLABLE", "ANILLO", "TRAVESIA", "ESTACION_FERROCARRIL",  
15        "CAÑADA", "AUTOPISTA", "RONDA", "AEROPUERTO", "PUENTE", "CALLEJON",  
16        "COLPB.", "INS.", "IDB."]
```



```

17
18     for a in listaPalabrasQuitar:
19
20         if(nombreVia.__contains__(a)):
21             existePalQuit = False
22             if (nombreVia.__contains__(' ' + a + ' ')):
23                 existePalQuit = True
24                 arr = nombreVia.split()
25
26             elif ( # Para evitar que pertenezca a una palabra
27                   (findall('[A-z, 0-9, \s, À-Ãž]{1}'+ a, nombreVia) == [] or findall('[A-z, 0-9, \s, À-Ãž]{1}'+ a,
28                     nombreVia) == [' '])
29                   and (findall(a + '([A-z, 0-9,\s, À-Ãž]{1})', nombreVia) ==[] or findall(a + '([A-z, 0-9,\s, À-Ãž]{1}',
30                     nombreVia) ==[' ' ])):
31                 #Para evitar errores por ejemplo en SegoVIA
32                 #Para evitar errores por ejemplo en PUENTECESURES
33                 arr = nombreVia.split()
34                 existePalQuit = True
35
36             if (existePalQuit and arr.index(a) == 0): # Para evitar suprimir Gran VIA p.ej. (que este al inicio) ó
37                 FRANCISCO JOSÉ ARROYO
38                 nombreVia = nombreVia.replace(a, ' ')
39             if (existePalQuit and nombreVia.__contains__(' /') and findall('[A-z, À-Ãž]{1}' + ' /', nombreVia) == []):
40                 if (arr.index(a) == arr.index('/') + 1): # Para cuando es un cruce
41                     nombreVia = nombreVia.replace(a + ' ', ' ')
42
43             # Podria ser M11 por ejemplo
44             if(nombreVia.replace(" ", "") != "" and nombreVia.strip()[0] == '/'):
45                 # Para eliminar descripciones al inicio y su / p.ej. SENDA_CICLABLE / AV.ROSALES / CARRET.VILLAV. A VALLECA
46                 nombreVia = nombreVia.replace('/', ' ', 1)
47     return nombreVia

```

Listing 3.11: Función quitarTextoEntreParentesis

```

1 def quitarTextoEntreParentesis(nombreVia = ""):
2     if(nombreVia.__contains__("(") and nombreVia.__contains__(")")):
3         posIni = nombreVia.index("(")
4         posFin = nombreVia.index(")")
5         txtElim = nombreVia[posIni:posFin+1]
6         return nombreVia.replace(txtElim, "")
7     if (nombreVia.__contains__("(")):
8         posIni = nombreVia.index("(")
9         txtElim = nombreVia[posIni:]
10        return nombreVia.replace(txtElim, "")
11    else:
12        return nombreVia

```

Como se puede observar no se detectan las palabras comprobando si el nombre las contiene, sino que se busca que estén separadas por espacios para evitar errores, por ejemplo eliminar VIA en “SegoVIA“. Se elimina el texto entre paréntesis ya que se considera que no forma parte de las palabras esenciales del nombre de la vía. Parte de las transformaciones necesarias para esto ya fueron realizadas en el capítulo relativo a cambios básicos.

3.4. Obtención de Identificadores de Vías

Los identificadores de las vías serán los que permitan la cohesión entre los distintos conjuntos de datos que aquí se están tratando. Todos ellos tienen en común propiedades relativas a vías, aunque no es posible enlazarlos por el nombre de un modo simple. La obtención de las palabras clave de estos títulos permitiría un enlazado bastante eficaz entre ellos, aunque con un coste computacional muy elevado y sería muy ineficaz para una aplicación móvil como es el caso de este proyecto.

El identificador de las vías es necesario para el enlazado de las mismas (es parte de la URI de estos recursos) y sería conveniente que datasets posteriores tuviesen dicha información, ya que seguirían las pautas de los vocabularios definidos para ellos.

Al estar utilizando conjuntos de datos que aun no han tenido en cuenta estos requerimientos y al ser necesario para la creación del recurso de las vías, se ha creado una función que lo obtenga a partir del dataset de callejero [6], el cual si contiene tanto nombres como identificadores y ha sufrido las mismas transformaciones, es decir, para una misma calle debería tener las mismas palabras clave. Para una mayor eficacia en la búsqueda se comprueba el tipo de vía que es y su nombre en varias vueltas, aumentando el margen de error en cada una de ellas. Se ha utilizado el código mostrado en 3.12 y posteriormente se detallará su funcionamiento.

Listing 3.12: Función crearFichNombresId

```
1 def crearFichNombresId(nombreCarpeta = "", nombreSinCsv = "", nFileIni = "-1", nFileFin = "_IDs_000",
2     nRowPalabrasClave = "-1", nRowTipoVia="-1", tieneTipoCalle = False):
3     with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
4         csvreader = csv.reader(csvfile, delimiter=";")
5         file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
6         primeraLinea = True
7         nCorrect = 0
8         nIncorrect = 0
9         contadorFila = 0
10        for row in csvreader:
11            if (primeraLinea):
12                fila = ";".join(row)
13                fila = fila + ";" + "idVia"
14                primeraLinea = False
15            else:
16                idVia = "-1"
17                nombre = row[nRowPalabrasClave].upper()
18                nVuelta = 1
19                while (nVuelta <=8 ):
20                    with open( RUTA_CALLEJERO ) as csvCallejero:
21                        csvreaderCallej = csv.reader(csvCallejero, delimiter=";")
22                        primeraLinea = True
23                        encontrado = False
24                        for rowCallj in csvreaderCallej:
25                            if (primeraLinea):
26                                primeraLinea = False
27                            else:
28                                #----- Bloque desplazado para mejor comprensión -----
29            try:
30                if(
31                    not(rowCallj[1].upper().__contains__("AUTOVIA") or rowCallj[2].upper().__contains__("AUTOVÍA")
32                    or row[nRowTipoVia].upper() == "Autovia" or row[nRowTipoVia].upper() == "Autopista"
33                    ) # Si es autovia no debe comprobarlo porque no pueden circular bicicletas
34                    and (
35                        (not(tieneTipoCalle) or
36                        chequearPalabras(rowCallj[POS_TIPVIA_CALL].upper(), row[nRowTipoVia].upper(), -1))
37                        or
38                        (nVuelta > 4 and tieneTipoCalle
39                        and esTipoCalleOmitible(row[nRowTipoVia].upper())
40                        and not(chequearPalabras(rowCallj[POS_TIPVIA_CALL].upper(), row[nRowTipoVia].upper(), -1)))
41                    ) # En la 5a, 6a, 7a y 8a vuelta se comprobará sin tipo de via y de nuevo con las mismas comprobaciones
42                    and ((nVuelta<=4 and chequearPalabras(rowCallj[POS_PALCLAV_CALL].upper(), nombre.upper(), nVuelta))
43                    or(nVuelta>=5 and chequearPalabras(rowCallj[POS_PALCLAV_CALL].upper(), nombre.upper(), nVuelta-4)))
44                ):
45                    idVia = rowCallj[POS_IS_CALL]
46                    encontrado=True
47                    nCorrect = nCorrect+1
48                    nVuelta = 99
49                break
50            except IndexError:
51                print("Error")
52                # Posible error de que hay una linea vacia extra al final
53                #-----
```

```

54         if(not(encontrado) and nVuelta < 6):
55             nVuelta = nVuelta + 1
56             continue
57         if(not(encontrado) and nVuelta>=6):
58             nVuelta = 99
59             idVia = "-1"
60             nIncorrect = nIncorrect + 1
61             csvCallejero.close()
62             fila = ";".join(row)
63             fila = fila + ";" + idVia.__str__()
64             contadorFila = contadorFila+1
65             if(contadorFila % 10 == 0):
66                 print("Fila: " , contadorFila)
67             file.write(fila + os.linesep)
68         file.close()
69         print("Incorrectas: " , nIncorrect , " || Correctas: " , nCorrect)

```

En este código se hace un recorrido por dos datasets, primero en el que se quieren añadir los identificadores (Accidentes, CallesTranquilas o Ciclocarriles), el cual tiene únicamente los nombres de sus calles. El segundo recorrido se hace a través del Callejero de Madrid antes mencionado. Éste último contiene tanto los nombres como los identificadores y en el caso de que alguno de sus nombres coincida, el id de ese registro será copiado en el otro conjunto. Para ello, por cada registro del dataset al que queremos añadir este valor, se hará un recorrido completo al callejero hasta encontrar ese nombre.

Debido a la cantidad de errores que pueden ocurrir en este proceso se realiza la búsqueda varias veces. Primeramente, se dan 4 vueltas comprobando que el tipo de vía coincida en ambos datasets. En cada iteración se aumenta el margen de error permitido (como se observa en el código 3.13). En las 4 siguientes iteraciones se sigue el mismo proceso, aunque sin comprobar el tipo de vía en el caso de que sea omitible. Hay ciertos tipos que no pueden ser obviados de esta comprobación: “AUTOVIA”, “POBLADO”, “VALLE”, “ESCALINATA”, “PASO ELEVADO”, “SENDA CICLABLE”, “GALERIA”, “CAÑADA”, “AUTOPISTA”, “POLIGONO”, “RONDA”, “AEROPUERTO”, “PUENTE”, “TRAVESIA”, “PLAZUELA”, “CALLEJON”, “COSTANILLA”, “JARDIN”, “ARROYO”, “PARTICULAR”, “TRASERA”, “COLONIA”. Estos casos se han considerado como posibles fuentes de nombres y se ha decidido que sea obligatorio que coincida la tipología en ambos conjuntos (por ejemplo, no puede ser igual la Calle del Atazar que el Poblado del Atazar).

Todas estas iteraciones se realizan haciendo una búsqueda por todo el callejero de Madrid en cada una de ellas. Es decir, se van comprobando todas las vías de menor a mayor margen de error, de forma que se cruce con la que tenga más similitud, o por lo menos, no menos coincidencia que otra.

Listing 3.13: Función chequearPalabras

```

1 def chequearPalabras(nombreCallej = "", nombreDataset = "", nVuelta = 1):
2     if(nombreDataset.replace(" ", "") == "" or nombreCallej.replace(" ", "") == ""):
3         return False
4     nombreDataset = nombreDataset.upper().replace("Á", "A").replace("É", "E").replace("Í", "I") \
5         .replace("Ó", "O").replace("Ú", "U").replace("Ü", "U").replace(",", " ").replace("-", " ")
6     nombreCallej = nombreCallej.upper().replace("Á", "A").replace("É", "E").replace("Í", "I") \
7         .replace("Ó", "O").replace("Ú", "U").replace("Ü", "U").replace(",", " ").replace("-", " ")
8
9     # Comprobaciones básicas
10    # -----
11    if(unicodedata.normalize('NFKD', nombreCallej.replace(" ", "")).encode('ASCII', 'ignore').strip().upper() \
12       == unicodedata.normalize('NFKD', nombreDataset.replace(" ", "")).encode('ASCII', 'ignore').strip().upper()):
13        return True
14    elif(nVuelta ==1):
15        return False #La primera vuelta solo hace estas comprobación
16    # -----
17    longCadena1 = nombreDataset.__len__()
18    longCadena2 = nombreCallej.__len__()
19    if (longCadena1 < (longCadena2 * 0.7) or longCadena1 > (longCadena2 * 1.3)):
20        return False # Si el tamaño de la cadena difiere mucho
21    diff = difflib.ndiff(nombreCallej.replace(" ", ""), nombreDataset.replace(" ", ""))
22    diferenciastxt = ''.join(diff)
23    # -----
24    # -----

```

3.4. Obtención de Identificadores de Vías

```
25 # -----
26 if(nVuelta == 2): # Excepciones cuando hay pequeños cambios que pueden deberse a errores ortograficos
27     # Por ejemplo FERNADO VI --> FERNANDO VI
28     # Añadiendo una S por ejemplo
29     # Vuelta2: comprobar 1 solo error en total
30     if ((diferenciastxt.count('+') + diferenciastxt.count('-')) <= 1
31         and not (nombreCallej.__contains__("BARROS") and nombreDataset.__contains__("BARRIOS")) # BARRIOS ::
32             BARROS
33         and not (nombreCallej.__contains__("OLIVAR") and nombreDataset.__contains__("BOLIVAR")) # Bolivar -
34             Olivar
35         and not (nombreCallej.__contains__("ESTE") and nombreDataset.__contains__("OESTE")) # OESTE:: ESTE
36         and not (nombreCallej.__contains__("OESTE") and nombreDataset.__contains__("ESTE"))
37         and not (nombreCallej.__contains__("VIAR") and nombreDataset.__contains__("VIA")) # VIA :: VIAR
38         and not (nombreDataset.__contains__("VIA") and not (nombreCallej.__contains__("VIA")))
39     ):
40         print("Vuelta2: Posible Conicidencia: ", nombreDataset, " :: ", nombreCallej)
41         return True
42 # -----
43 if (nVuelta == 3):
44     # Vuelta3: comprobar 1 sustitucion (quitar 1 letra y añadir otra)
45     if (diferenciastxt.count('+') <= 1 and diferenciastxt.count('-') <=1
46         and not((diferenciastxt.count('+') + diferenciastxt.count('-')) <= 1)
47         and not (nombreDataset.__contains__("HORTALEZA") and nombreCallej.__contains__("FORTALEZA")) #Hortaleza -
48             Fortaleza
49         and not (nombreCallej.__contains__("GALENA") and nombreDataset.__contains__("GILENA")) # Gilena - Galena
50         and not (nombreCallej.__contains__("PEAL") and nombreDataset.__contains__("REAL")) # Real - Peal
51         and not (nombreCallej.__contains__("HAYA") and nombreDataset.__contains__("RAYA")) # RAYA :: HAYA
52         and not (nombreCallej.__contains__("OCA") and nombreDataset.__contains__("OÑA")) # OÑA :: OCA
53         and not (nombreDataset.__contains__("CANDILEJAS") and nombreCallej.__contains__("CANALEJAS")) #
54             CANDILEJAS :: CANALEJAS
55         and not (nombreDataset.__contains__("PASO") and nombreCallej.__contains__("MASO")) # PASO:: MASO
56         and not (nombreCallej.__contains__("VID") and nombreDataset.__contains__("VIA")) # VIA :: VID
57         and not (nombreDataset.__contains__("VIA") and not (nombreCallej.__contains__("VIA")))
58     ):
59         print("Vuelta3: Posible Conicidencia: ", nombreDataset, " :: ", nombreCallej)
60         return True
61 # -----
62 if(nVuelta == 4):
63     # Si se han añadido o quitado 2 o menos letras, se puede considerar igual
64     # Dependiendo de la longitud de la palabra acepta 1 error o más
65     if (diferenciastxt.count('+') <= (longCadena1/10+1) and diferenciastxt.count('-') <=(longCadena1/10+1)
66         and not(diferenciastxt.count('+') <= 1 and diferenciastxt.count('-') <=1)
67         and not((diferenciastxt.count('+') + diferenciastxt.count('-')) <= 1)
68         and not (nombreDataset.__contains__("HORTALEZA")) #Hortaleza - Fortaleza
69         and not (nombreCallej.__contains__("GALENA")) # Gilena - Galena
70         and not (nombreCallej.__contains__("PEAL")) # Real - Peal
71         and not (nombreCallej.__contains__("OLIVAR")) # Bolivar - Olivar
72         and not (nombreCallej.__contains__("CRUCES") and nombreDataset.__contains__("RUICES")) # RUICES ::
73             CRUCES
74         and not (nombreCallej.__contains__("HAYA")) # RAYA :: HAYA
75         and not (nombreCallej.__contains__("OCA")) # OÑA :: OCA
76         and not (nombreCallej.__contains__("MANZANAR") and nombreDataset.__contains__("MANZANARES")) #
77             MANZANARES :: MANZANAR
78         and not (nombreDataset.__contains__("CANDILEJAS") and nombreCallej.__contains__("CANALEJAS")) #
79             CANDILEJAS :: CANALEJAS
80         and not (nombreCallej.__contains__("CENICIENTOS") and nombreDataset.__contains__("CENICIENTA")) #
81             CENICIENTA :: CENICIENTOS
82         and not (nombreCallej.__contains__("CANTERAS") and nombreDataset.__contains__("MANOTERAS")) # MANOTERAS
83             :: CANTERAS
84         and not (nombreDataset.__contains__("MANOTERAS") and nombreDataset.__contains__("SANTERAS")) # MANOTERAS
85             :: SANTERAS
86         and not (nombreCallej.__contains__("GOR") and nombreDataset.__contains__("GADOR")) # SIERRA GADOR ::
87             SIERRA GOR
88         and not (nombreDataset.__contains__("GADOR") and nombreDataset.__contains__("GUDAR")) # SIERRA GADOR ::
89             SIERRA GUDAR
90         and not (nombreCallej.__contains__("ERASMO") and nombreDataset.__contains__("RASO")) # SANZ RASO ::
91             SAN ERASMO
92         and not (nombreCallej.__contains__("ALIO") and nombreDataset.__contains__("AMON")) # SANTIAGO AMON ::
93             SANTIAGO ALIO
94         and not (nombreDataset.__contains__("PARVILLAS") and nombreCallej.__contains__("MARAVILLA")) # PARVILLAS
95             :: MARAVILLA
96         and not (nombreDataset.__contains__("MARMOLINA") and nombreCallej.__contains__("CAROLINA")) # MARMOLINA
97             :: CAROLINA
98         and not (nombreCallej.__contains__("SAMANIEGO") and nombreDataset.replace(" ",
99             "").__contains__("SANDIEGO")) # SAN DIEGO :: SAMANIEGO
100         and not (nombreDataset.replace(" ", "").__contains__("SANDIEGO")
101             and nombreCallej.replace(" ", "").__contains__("SANDACIO")) # SAN DIEGO :: SAN DACIO
102         and not (nombreDataset.replace(" ", "").__contains__("MONTEAYA") and
103             nombreCallej.__contains__("MONTANA")) # MONTE AYA :: MONTANA -
104         and not (nombreCallej.__contains__("SANTERAS") and nombreDataset.__contains__("SANTERAS")) # SANZ RASO
105             :: SANTERAS -
106         and not (nombreCallej.__contains__("ALCORISA") and nombreDataset.replace(" ",
107             "").__contains__("PALOROSA")) # PALO ROSA :: ALCORISA
108         and not (nombreCallej.replace(" ", "").__contains__("CERROMONTE") and
109             nombreDataset.__contains__("SACROMONTE")) # SACROMONTE :: CERRO MONTE -
110         and not (nombreCallej.replace(" ", "").__contains__("EDUARDOUROSA")
111             and nombreDataset.replace(" ", "").__contains__("EDUARDOAUNOS")) # EDUARDO AUNOS ::
112             EDUARDO UROSA -
113         and not (nombreCallej.replace(" ", "").__contains__("MADREDIOS")
114             and nombreDataset.replace(" ", "").__contains__("MADRIDRIO")) # MADRID RIO :: MADRE DIOS
115         and not (nombreCallej.__contains__("FORTEA") and nombreDataset.replace(" ", "").__contains__("ZORITA"))
116         # COMANDANTE ZORITA :: COMANDANTE FORTEA
```

Transformaciones en los Datasets

```
97         and not (nombreCallej.__contains__("LIMON") and nombreDataset.__contains__("VIGON")) # JUAN VIGON ::
98             JUAN LIMON
99         and not (nombreCallej.__contains__("ACUARELA") and nombreDataset.__contains__("PASARELA")) # PASARELA
100             :: ACUARELA
101         and not (nombreCallej.__contains__("PASA") and nombreDataset.__contains__("PASO")) # PASO :: PASA
102         and not (nombreCallej.__contains__("PASO") and nombreDataset.__contains__("PASA"))
103         and not (nombreDataset.__contains__("VIA") and not(nombreCallej.__contains__("VIA")))
104     ):
105         print("-----Vuelta4: Posible Conicidencia: ", nombreDataset, " :: ", nombreCallej)
106         return True
107     # -----
108     return False
```

En la función 3.13 se detalla el funcionamiento de la comprobación de nombres al que antes se hacia referencia. Primero se transforma a mayúsculas y se eliminan caracteres especiales y espacios para una comparación más eficaz.

Se puede observar que está dividido en 4 secciones que corresponderían a las rondas de error que se aplicaban anteriormente. En la primera se comprueba que sea exactamente igual. La gran mayoría de elementos son cruzados en esta primera iteración ya que ha sido refinado anteriormente y reducido a palabras clave. Hay una alta probabilidad de que los elementos de un dataset y otro sean iguales.

En caso contrario, para la segunda vuelta aplicada se le permitirá un carácter añadido o eliminado. Este es el caso por ejemplo de palabras acabas en “s” u otras erratas. Para la tercera vuelta se le permitirá la sustitución de un carácter por otro, es decir, añadir y eliminar un carácter. Esto permite errores como cambios de vocales o erratas de sustitución de letras. Para la cuarta y última vuelta se tiene en cuenta la longitud de la cadena. Para nombres compuestos y extensos es más probable que ocurran erratas, por lo tanto, a mayor nombre mayor margen de error permitido.

Como se puede observar en el código 3.13, tras los pasos 2, 3 y 4 se imprime por terminal la transformación realizada. Esto permite al desarrollador comprobar el buen funcionamiento del programa y, en caso de revisarlo y encontrar elementos mal emparejados, poder modificarlos manualmente antes de que se añadan a la aplicación final. Para el paso 1 no es necesario ya que el nombre es exacto, pero las siguientes, al permitir margen de error, pueden ocurrir fallos fácilmente detectables de forma manual. También se observan multitud de restricciones. Esto es debido a lo mencionado anteriormente, errores detectados de forma manual han sido añadidos a las distintas fases de las comprobaciones para que no los considere en posteriores ejecuciones y así poder refinar el emparejamiento de los dataset.

Capítulo 4

Generación y búsqueda en ficheros

Una vez definidos los vocabularios y transformados los datasets acorde con las necesidades que detallaron, se deben generar sus correspondientes ficheros OWL. Para este proyecto, como viene siendo habitual en la web semántica, se utiliza SPARQL para hacer la búsqueda a través de los conjuntos de datos y utilizar así los recursos. Para ello, en primer lugar, se deben generar los ficheros en formato TTL o RDF, con sus correspondientes separaciones por clases y propiedades de cada uno de sus componentes. Una vez se consiga esta organización de los elementos, se podrán hacer consultas utilizando las queries antes mencionadas y se podrán obtener los elementos requeridos para cada recurso.

Este proceso, como se ha podido observar, se divide en dos secciones bien definidas: la generación de los ficheros en el formato deseado y las búsquedas sobre estos.

4.1. Generación de ficheros OWL

Para esta primera sección se ha utilizado el programa OpenRefine [23]. En él se han cargado los datasets en formato CSV antes transformados y se han generado estos ficheros OWL necesarios para las siguientes consultas.

En este proceso se ha partido de la estructura de los vocabularios definidos anteriormente y se ha replicado con los datos disponibles. Para el caso de CallesTranquilas se ha generado un fichero diferente al definido. Dado que no se ha creado un vocabulario específico para ello sino que se han añadido propiedades relativas a ello en el Callejero, no se podía generar del modo en que estaba en el dataset original. Para el vocabulario definido debería existir una propiedad en el dataset de callejero que indicase si una calle es tranquila o no. Dado que dicha propiedad aun no esta creada y que es proporcionada por un conjunto de datos paralelo, se deben obtener esos valores de este último. Para la aplicación que se quiere desarrollar solo es necesario conocer las propiedades de las calles tranquilas y no de todas las vías de Madrid, siguiendo las especificaciones estrictas del vocabulario se tendría que haber obtenido esta propiedad del callejero, sin embargo, se ha tomado la decisión de revisar únicamente el dataset de calles tranquilas, reduciendo así a un 10 % aproximadamente el total de calles, haciendolo más eficiente y partiendo de la clase vía con sus propiedades (como se especifica en la definición de la ontología). En caso de que años posteriores los datasets siguiesen las recomendaciones expuestas en este proyecto sobre la exposición de los datos, se tendrían que hacer ciertas modificaciones en las queries relativas a este datasets, aunque como se ha mencionado, se ha partido del nodo raíz Vía y se han consultado sus propiedades, por lo tanto los cambios serían mínimos.

Para la generación de los ficheros de calles tranquilas se ha tomado como nombre de la vía el título original, es decir, la columna relativa al nombre sin corrección de erratas ni eliminación de palabras que provenía de la fuente origen. En el caso de CicloCarriles y Accidentes se ha elegido el nombre de la vía. En el primer caso ya que contenía multitud de erratas y se considera más legible el nombre modificado. En el segundo caso debido a que el nombre original contiene el título del cruce en muchos casos, y el nombrado de la vía debe ser único de la misma.

En 4.1 se ve un ejemplo de la distribución de los elementos del dataset de Ciclocarriles en el fichero OWL.

Listing 4.1: ciclocarriles.ttl

```

1 <http://vocab.ciudadesabiertas.es/recurso/callejero/madrid/Via/560600> a escjr:Via;
2   <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#Municipio>
3   <https://datos.ign.es/recurso/municipio/28079>;
4   escjr:longitud "299.0"^^<xsd:double>;
5   escjr:tipoUso <http://vocab.ciudadesabiertas.es/kos/urbanismo-infraestructuras/calle/tipo-uso/CICLOCALLES>;
6   <http://www.geonames.org/ontology#officialName> " PALOS DE LA FRONTERA" .
7
8 <https://datos.ign.es/recurso/municipio/28079> a
9   <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#Municipio>;
10  <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#codigoINE>
11  "28079"^^<http://www.w3.org/2001/XMLSchema#int>;
12  <http://www.geonames.org/ontology#officialName> "Madrid" .
13
14 <http://vocab.ciudadesabiertas.es/recurso/callejero/madrid/ciclo-carril/560600> a
15   cl-ciclo:Ciclocarril;
16   cl-ciclo:carrilExclusBici true;
17   cl-ciclo:distMaxExclusBici "298.76"^^<xsd:double>;
18   geosparql:sfWithin <http://vocab.ciudadesabiertas.es/recurso/callejero/madrid/Via/560600> .

```

En 4.2 se ve un ejemplo de la distribución de los elementos del dataset de Calles Tranquilas en el fichero OWL.

Listing 4.2: callesTranquilas.ttl

```

1 <http://vocab.ciudadesabiertas.es/recurso/callejero/madrid/Via/75300> a escjr:Via;
2   escjr:dobleSentido
3   <http://vocab.ciudadesabiertas.es/kos/urbanismo-infraestructuras/calle/doble-sentido/SENTIDO-UNICO>;
4   escjr:longitud "846"^^<xsd:double>;
5   escjr:municipio <https://datos.ign.es/recurso/municipio/28079>;
6   escjr:tipoUso <http://vocab.ciudadesabiertas.es/kos/urbanismo-infraestructuras/calle/tipo-uso/CICLOCALLES>;
7   geo:officialName "Calle del Arroyo de la Media Legua" .
8
9 <https://datos.ign.es/recurso/municipio/28079> a escjr:Municipio;
10  esadm:codigoINE "28079"^^<http://www.w3.org/2001/XMLSchema#int>;
11  geo:officialName "Madrid" .

```

En 4.3 se ve un ejemplo de la distribución de los elementos del dataset de Accidentes de bicicletas en el fichero OWL.

Listing 4.3: accidentes.ttl

```

1 <http://vocab.ciudadesabiertas.es/recurso/transporte/accidente/2019S000659> a accid:Accidente;
2   dct:identifier "2019S000659";
3   accid:enCruce "2"^^<http://www.w3.org/2001/XMLSchema#int>;
4   accid:fecha "01/01/2019"^^<xsd:date>;
5   accid:hasPersAfectadAccid
6   <http://vocab.ciudadesabiertas.es/recurso/transporte/accidente/persona-afectada/2019S000659>;
7   accid:hour "14:00"^^<xsd:hour>;
8   accid:lesividad <http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/01>;
9   accid:meteorologia <http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/meteorologia/Despejado>;
10  accid:ocurreEnVia <http://vocab.ciudadesabiertas.es/recurso/callejero/madrid/Via/153800>;
11  <http://vocab.ciudadesabiertas.es/recurso/callejero/madrid/Via/248700>;
12  accid:tipoAccidente <http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-accidente/Alcance>;
13  accid:tipoVehiculo <http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-vehiculo/Bicicleta>;
14  <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#contienePortal>
15  <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero/madrid/153800/->,
16  <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero/madrid/248700/-> .
17
18 <http://vocab.ciudadesabiertas.es/recurso/transporte/accidente/persona-afectada/2019S000659>
19   a accid:PersonaAfectada;
20   schema:gender schema:Male;

```



```
20   schema:typicalAgeRange "25-29";
21   accid:tipoPersAfect <http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/tipo-pers-afect/Conductor>
22   .
23 <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero/madrid/153800/->
24   a <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#Portal> .
25
26 <http://vocab.ciudadesabiertas.es/recurso/callejero/madrid/Via/153800> a
27   <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero#Via>;
28   accid:ocurrioAccidente <http://vocab.ciudadesabiertas.es/recurso/transporte/accidente/2019S000659>;
29   <http://www.geonames.org/ontology/officialName> "CALLE CASTELLO";
30   geosparql:sfWithin <https://datos.ign.es/recurso/municipio/28079> .
31
32 <https://datos.ign.es/recurso/municipio/28079> a
33   <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio/Municipio>;
34   <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio#codigoINE>
35   "28079"^^<http://www.w3.org/2001/XMLSchema#int>;
36   <http://www.geonames.org/ontology#officialName> "Madrid" .
```

4.2. Consultas SPARQL

Capítulo 5

Aplicación Android

5.1. Obtención de la ruta

5.1.1. Creación BBDD coordenadas

5.1.2. Conexión API Google Directions y lista de calles transitadas

5.2. Cálculo de Seguridad de ruta

Capítulo 6

Resultados y conclusiones

Resumen de resultados obtenidos en el TFG. Y conclusiones personales del estudiante sobre el trabajo realizado.

Bibliografía

- [1] <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero/index-en.html>
- [2] <http://schema.org>
- [3] <https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=20f4a87ebb65b510VgnVCM1000001d4a900aRCRD&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnnextfmt=default>
- [4] <https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=435a7cd5de319410VgnVCM1000000b205a0aRCRD&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnnextfmt=default>
- [5] <https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=a320f5ac548f4410VgnVCM1000000b205a0aRCRD&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnnextfmt=default>
- [6] <https://datos.madrid.es/sites/v/index.jsp?vgnextoid=b3c41f3cf6a6c410VgnVCM2000000c205a0aRCRD&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD>
- [7] <https://datos.madrid.es/egob/new/detalle/auxiliar/mapa.jsp?geoUrl=/egob/catalogo/205115-4-calles-tranquilas.kml>
- [8] <https://datos.ign.es/def/btn100/index-es.html#calzada>
- [9] <https://www.ine.es/daco/daco42/codmun/codmunmapa.htm>
- [10] <https://datos.ign.es/def/btn100/index-es.html#calzada>
- [11] <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero/index-en.html#tipoVia>
- [12] <http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio/index-en.html#Municipio>
- [13] <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero/index-en.html#Via>
- [14] <https://schema.org/gender>
- [15] <https://lists.w3.org/Archives/Public/public-schemaorg/2019Oct/0013.html>
- [16] <https://schema.org/typicalAgeRange>

- [17] <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero/index-en.html#Portal>
- [18] <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms/identifier>
- [19] <https://ciudadesabiertas.es/vocabularios/#CatálogoVocabularios>
- [20] <https://datos.madrid.es/portal/site/egob>
- [21] <https://datos.madrid.es/egob/new/detalle/auxiliar/mapa.jsp?geoUrl=/egob/catalogo/205115-4-calles-tranquilas.kml>
- [22] https://elpais.com/ccaa/2017/04/28/madrid/1493369660_675682.html
- [23] <https://openrefine.org>

Anexo

Este capítulo es opcional, y se escribirá de acuerdo con las indicaciones del Tutor.