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

La publicación de datos abiertos por parte de ayuntamientos e instituciones públicas está cada vez más extendido y en un futuro se irá incrementando. Estos datos son completamente accesibles y reutilizables para cualquier fin que un usuario quiera darles, lo cual da mucha libertad a la hora de crear aplicaciones y dar valor a esa información. Esta información, en su mayoría, está publicada en formato RDF, CSV u hojas de calculo. La publicación por parte de los ayuntamientos suele contener errores, incoherencias u otros problemas que impiden su correcta utilización. Es por ello que para su uso debe hacerse un análisis y determinar una estrategia y modelización de los mismos.

En trabajo consistirá en crear una aplicación que, a partir de los datos proporcionados por el ayuntamiento de Madrid, pueda hacer una valoración de las distintas calles acorde con la seguridad de las mismas para circular en bicicleta. Todo esto siguiendo los principios de Web Semantica y Linked Data que permita realizar un buen modelado de los datos, para su correcto funcionamiento y para una posible reutilización posterior.

Primero será necesario seleccionar los vocabularios con los que se va a trabajar. Para ello se reutilizarán los ya creados en la plataforma vocab.ciudadesabiertas.es y se crearán nuevos a partir del portal de datos del ayuntamiento de Madrid (datos.madrid.es). Se reutilizará el vocabulario de Callejero, definido en [1]. Y se han diseñado tres nuevos vocabularios a partir de los datasets proporcionados por el ayuntamiento de Madrid. El correspondiente a Accidentes con implicación de Bicicletas, accesible en formato CSV y XLSX en la web [3]. El correspondiente a Ciclocarriles, accesible en formato XLS y KML en la dirección [4]. Y el correspondiente a Calles Tranquilas, accesible en formato XLS y KML en la dirección [6].

En este proyecto se va a desarrollar una aplicación que, a partir de estos datos y conociendo la ruta entre 2 puntos dentro de Madrid, se pueda determinar la seguridad de una ruta para ir en bicicleta. Para ello se hará uso de una API externa, de la cual se obtendrá la ruta entre las posiciones dadas por el ususario. Una vez se tengan las calles por las cuales el navegador GPS guiará al ciclista hasta su destino, se comprobará una a una su seguridad. Para esta comprobación se hará uso de los datos mencionados antes. Para ello, se le asignará un identificador único a cada calle, el cual es proporcionado por el Callejero del ayuntamiento [7] y definido como se ha mencionado anteriormente en el portal ciudadesAbiertas. Este identificador ha sido asignado a los otros tres datasets, cuyos vocabularios han sido definidos en el contexto de este trabajo, y de esta forma es posible hacer una busqueda rápida y concisa de cada calle por la que transitará la ruta, para así conocer mejor las características de ellas y determinar su seguridad.

Más adelante se detallarán los elementos de cada vocabulario utilizado en la aplicación. En términos generales, en la siguiente figura se muestran los vocabularios definidos para este proyecto y sus enlaces con el Callejero y entre ellos.

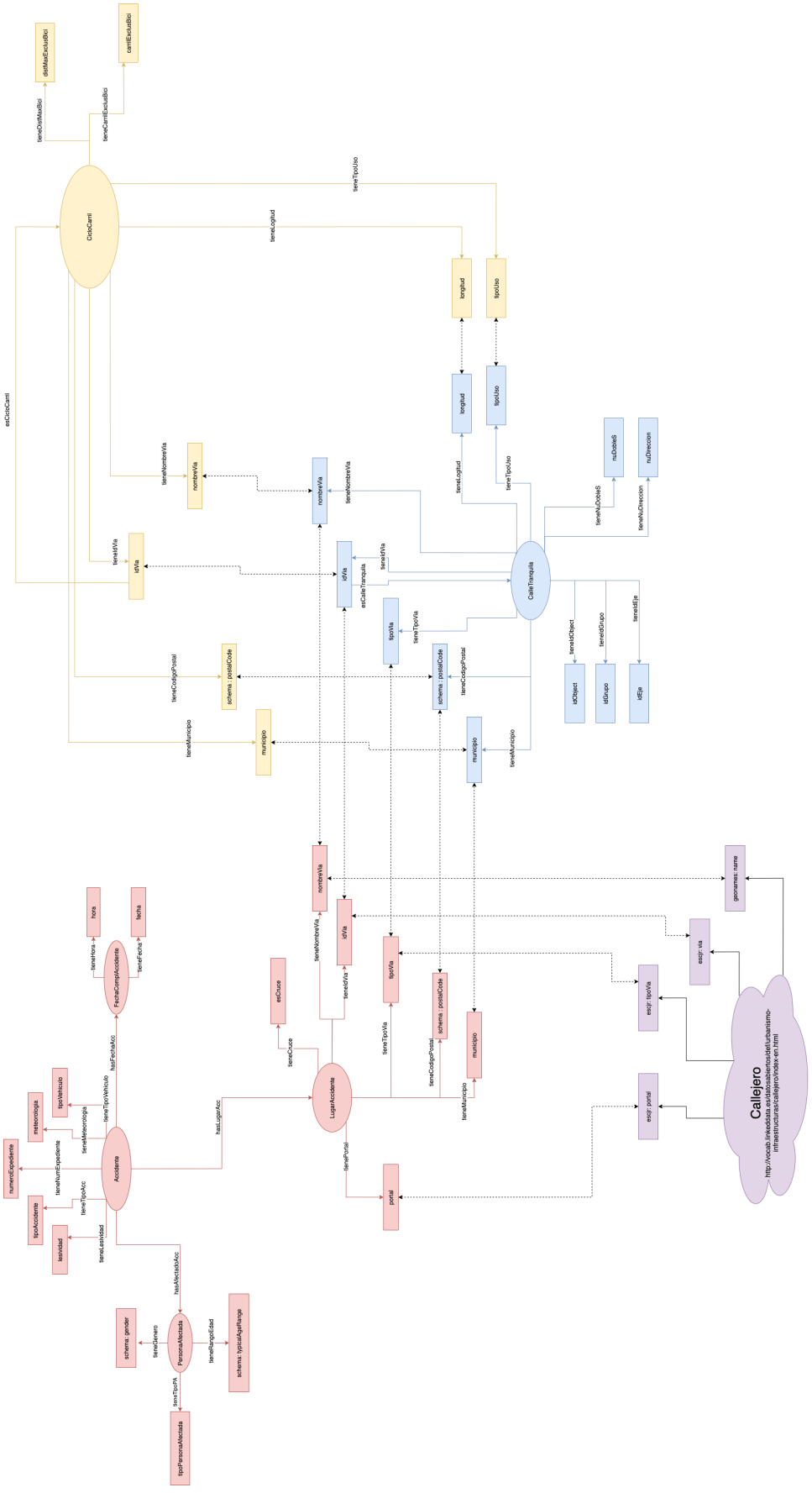

Figura 1: Diagrama de los vocabularios usados en la aplicacion

Tabla de contenidos

Capítulo 1

Introducción

Capítulo 2

Desarrollo

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 via u otros, ya que se han considerado necesarios para la definición de un vocabulario reutlizable 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 los elementos definidos pueden ser utlilizados en cualquier tipo de accidente. Debido a que esta trabajo está enfocado a crear una aplicación para la seguridad de bicicletas, se ha partido de esta base, pero podria ser perfectamente reutilizado para otro tipo de accidente añadiendole propiedades necesarias para los mismos como podrian ser la velocidad, el numero de pasajeros...
La organización del conjunto de datos se hará siguiendo el diagrama 2.1

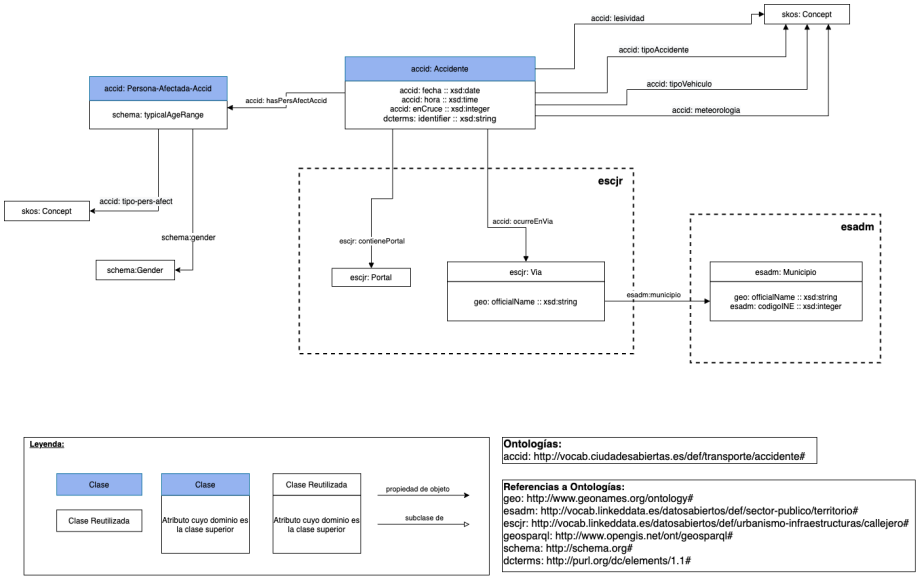

Figura 2.1: Diagrama de Ontología de Accidentes.

Para la representación de los datos de accidentes de trafico se han definido varias clases y propiedades. Se han reutilizado elementos ya definidos en el vocabulario de Callejero [1], de Territorio [13] y de Schema [2].

En la siguiente tabla se muestran los Namespaces usados.

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 2.2: 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 .enCruce", el municipio, el tipo de via o el identificador de via. 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. EnCruce 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 Via se obtendrá comparando el nombre de la via y su tipo con el Callejero de Madrid, el cual proporcionará este valor único que represente la via. El tipo de via 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 via. 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 la sección Transformaciones en los vocabularios Fase1. Los accidentes que han ocurrido entre un cruce de vias se han separado en tantos registros como vias 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 via pública con implicación de algún vehíclo. 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 [18]. 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 [13] 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 [14] 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, pantalanes, 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. Dia, 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 [17]

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 podria representarse la intersección entre calles. Esta propiedad representa el numero de calles asociadas. En caso de que no fuese cruce se le asignaria el valor 0, en los casos en los que si se asignaria 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 [19].

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://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 (peaton, 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 vehiculo 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/DEPEJADO
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`

Codigo 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://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/01`
02 Ingreso inferior o igual a 24 horas - LEVE:
`http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/02`
03 Ingreso superior a 24 horas. - GRAVE:
`http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/03`
04 Fallecido 24 horas - FALLECIDO:
`http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/04`
05 Asistencia sanitaria ambulatoria con posterioridad - LEVE:
`http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/05`
06 Asistencia sanitaria inmediata en centro de salud o mutua - LEVE:
`http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/06`
07 Asistencia sanitaria solo en el lugar del accidente - LEVE:
`http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/07`
14 Sin asistencia sanitaria:
`http://vocab.linkeddata.es/datosabiertos/kos/transporte/accidente/lesividad/14`
77 Se desconoce:
`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 [15] 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 [16]

tipoPersAfect

IRI:
`http://vocab.ciudadesabiertas.es/def/transporte/accidente#tipoPersAfect`

Persona a la que afecta el accidente. Puede ser Conductor, peaton, 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 via, 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 vias 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 [13].

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

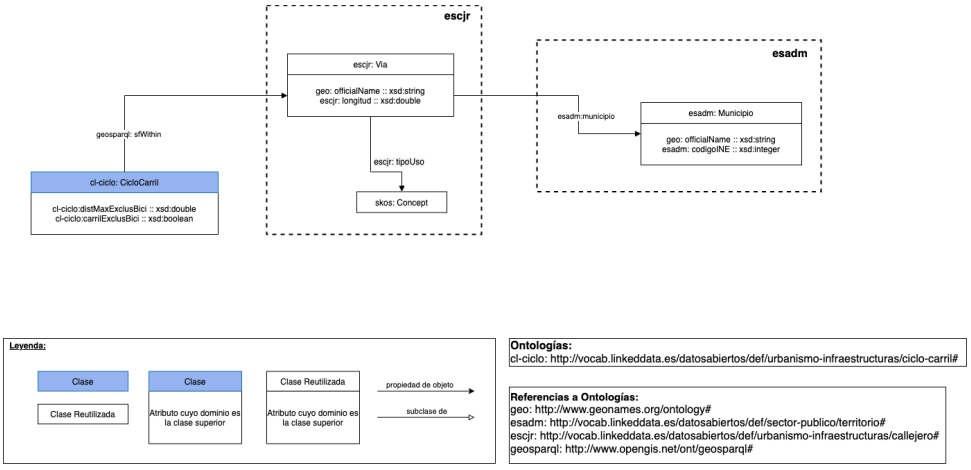

Figura 2.3: Diagrama de Ontología de Ciclocarriles.

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 ciudadesabiertas [1] y de Territorio [13].
Se han añadido elementos como el identificador de via y el municipio (que siempre será Madrid). El identificador de via se añadirá para cada caso a partir del nombre. Para ello se hará una reducción del nombre de la via a palabras clave, proceso detallado en la sección de Transformaciones de Vocabularios, y se cruzará con el dataset del callejero de Madrid [7].
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 ayuntameinto 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`

Via 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 [14] 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, pantalanes, 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 [13] 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 via 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 [6]. Proporcionada por el ayuntamiento de Madrid, se muestran las calles más apropiadas para el tránsito de ciclistas. No se dan especificaciones de los criterios utlizados que han llevado a estas calles a formar parte de la lista. Sin embargo, si se puede observar en algnas de ellas ciertos patrones, como que no forman parte de las vias 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 [8], 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 Vias. 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 aqui dispuestos. Esta propuesta de modificación se detallará en capitulos posteriores.
Sin embargo, si se ha hecho uso del dataset proporcionado por el ayuntamiento para la aplicación final que se esta construyendo en el contexto de este trabajo. Se han realizados ciertas modificaciones con respecto al dataset original para que puedan utilizarse sus datos más eficazmente.
Se ha optado por la separación del tipo de via del nombre, conservandola 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 via, obtenido a partir del nombre y cruzado con el dataset del callejero de madrid [7]. 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 reulizar 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 Estadistica[10].
Estas transformaciones se detallarán más adelante en la seccion: 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 via) 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 asi obtener la seguridad de una ruta. Muchos de ellos no pueden considerarse propios de las bicicletas sino que forman parte de las vias por las que éstas van a circular. La necesidad de estos datos y las posibles utilidades que podrian tener para muchos otras otras aplciaciones 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:inserteaquiurldepropuesta!!!`.

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 ontologias que se estaban creando en ciudadesabiertas [20]

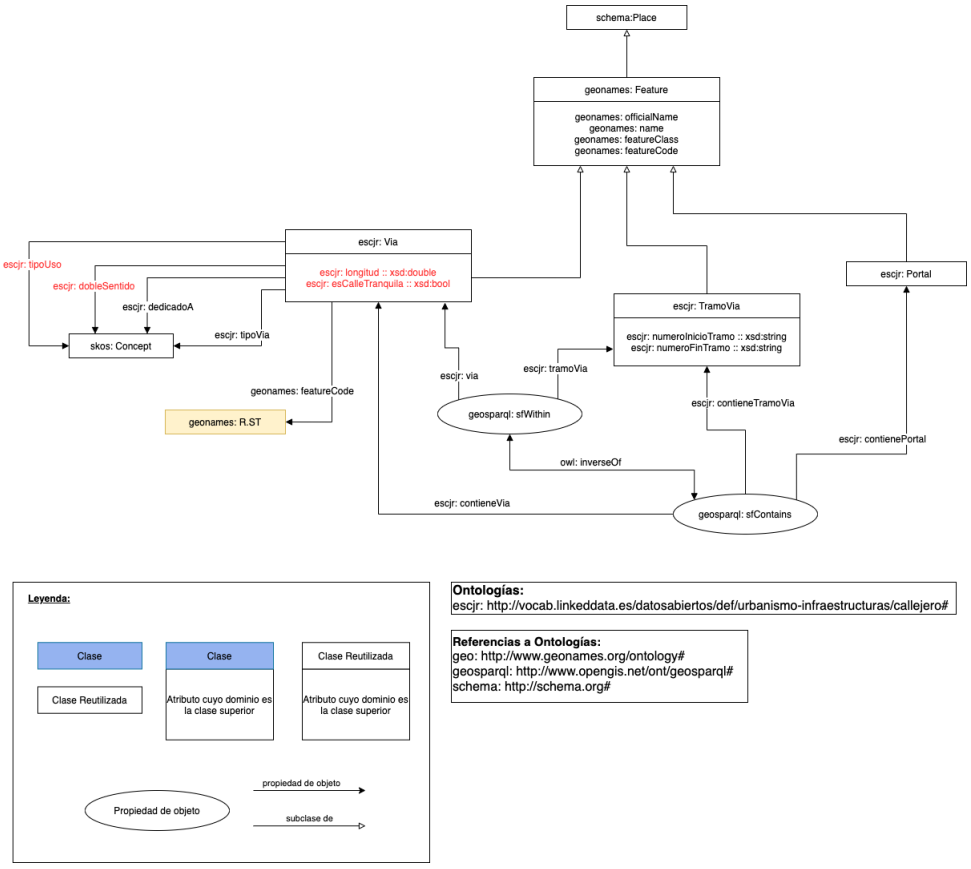

Figura 2.5: Diagrama de Ontología de Callejero

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 unico 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 via es calle tranquila o no para bicicletas. Vias 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

Doble sentido o sentido único de una calle. Puede tomar los siguientes valores definidos en datos.ign.es [11]:
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

2.5. Transformaciones en los vocabularios: Fase 1

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 o modificaciones o transformaciones en los ya existentes son algunos de los motivos para realizarlos.

Con el fin de repetir el proceso en datasets iguales referentes a otros años o posibles similitudes con otros, a continuación se detallan los pasos que se han seguido en este caso para poder replicarlos:

- Transformar a codificación ISO-8859-1.
- Modificar palabras básicas como por ejemplo C/ por calle y demás casuística. Para ello se ha definido el siguiente código para aplicarlo a todos los registros del dataset por igual.

Listing 2.1: Función cambiosBasicos

```
import csv
import os
from re import findall

def cambiosBasicos(nombreCarpeta="", nombreSinCsv="", nRowNombre=-1, nFileFin = "-1"):
    with open(nombreCarpeta + "/" + nombreSinCsv + ".csv") as csvfile:
        csvreader = csv.reader(csvfile, delimiter=";")
        file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
        primeraLinea = True
        for row in csvreader:
            if (primeraLinea):
                fila = ";".join(row)
                primeraLinea = False
            else:
                nombreVia = row[nRowNombre].upper()
                if (nombreVia.__contains__("C/")):
                    nombreVia = nombreVia.replace("C/", "C/ ")
                if (findall('CRUCE\s.*\sCON\s', nombreVia) != []):  # Escritura estandar de "Cruce"
                    nombreVia = nombreVia.replace(" CON ", " / ")
                    nombreVia = nombreVia.replace("CRUCE ", "")
                if(nombreVia.__contains__("S/N")):  # P.ej. SAN FRANCISCO S/N, casos que generan error
                    nombreVia = nombreVia.replace('S/N', "")
                if (findall('([A-z, C!, Ã¨, 0-9]{1})/[A-z]{1}', nombreVia) != []):  # Escritura estandar de
                    "Cruce"
                    nombreVia = nombreVia.replace("/", " / ")
                # C/  MONASTERIO DE ARLANZA-AV. SANTUARIO DE VALVERDE
                if (findall('(?!\d)[\-{1}](?!\d)', nombreVia) != []): # Evitar palabras con numeros como M-30
                    arr1 = nombreVia.replace("-", " - ").split()
                    if(getTipoVia(palablasMalEscritas(arr1[arr1.index("-")+1])) != ""):  # Si es palabra clave
                        indicadora de nueva via, es Cruce
                        nombreVia = nombreVia.replace("-", " / ")
                if (findall('(?!\d)[\-{1}](\d)', nombreVia) != []): # Palabras con numero como M-30
                    nombreVia = nombreVia.replace("-", "")

                # ;19:10;"CAMINO EN ZONA DENOMIDADA COMO ""ESPACIO MÉXICO""";
                if (nombreVia.__contains__('"')): #Quitar comillas
                    nombreVia = nombreVia.replace('"', ' ')
                if (nombreVia.__contains__("PASO ELEVADO")):
                    nombreVia = nombreVia.replace('PASO ELEVADO', 'PASO_ELEVADO')
                elif (nombreVia.__contains__("SENDA CICLABLE")):
                    nombreVia = nombreVia.replace('SENDA CICLABLE', 'SENDA_CICLABLE')

                #Cambios varios en nueva funcion
                nombreVia = palablasMalEscritas(nombreVia)

                row[nRowNombre] = nombreVia
                fila = ";".join(row)
            file.write(fila + os.linesep)
        file.close()
```

La finalidad de la función definida en Listing 2.1 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 çalle1 / calle2". Para conseguir ese formato han de modificarse otros nombres como por ejemplo ÇRUCE 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.
Otras transformaciones serian en palabras que consideramos clave, por ejemplo Paso elevado o Senda Ciclable, a las cuales se les considerará tipo de via, que sean formadas como una única palabra, facilitando asi su posterior búsqueda y que no se cometa el error de incluirlas en las palabras clave del nombre de la via.
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 guión, considerandolos de esa forma una única palabra.
Por último, se llama a la función palabrasMalEscritas, definida en Listing 2.2, la cual es en parte una continuación de esta aunque para casos más concretos.

Listing 2.2: Función palabrasMalEscritas

```
import csv
import os
from re import findall

def palablasMalEscritas(nombreCalle = ""):  #Arreglar errores de codificación y abreviaturas
    if(nombreCalle == ""):
        return ""
    nuevaPalabra = ""
    for palabra in nombreCalle.split():
        palabra = palabra.replace(" ", "").upper()
        if(palabra == "PNTE."):
            palabra = "PUENTE"
        elif (palabra == "CÑADA." or palabra == "CÑADA"):
            palabra = "CAÑADA"
        elif (palabra == "AVDA." or palabra == "AVDA" or palabra == "AV." or palabra == "AV"):
            palabra = "AVENIDA"
        elif (palabra == "JARDÍN" or palabra == "JARDINES"):
            palabra = "JARDIN"
        elif (palabra == "CUSTA." or palabra == "CUSTA"):
            palabra = "CUESTA"
        elif (palabra == "POLÍGONO"):
            palabra = "POLIGONO"
        elif (palabra == "GALERÍA"):
            palabra = "GALERIA"
        elif (palabra == "PLAZA."):
            palabra = "PLAZA"
        elif (palabra == "PISTA."):
            palabra = "PISTA"
        elif (palabra == "CMNO." or palabra == "CMNO"):
            palabra = "CAMINO"
        elif (palabra == "RONDA."):
            palabra = "RONDA"
        elif (palabra == "GTA." or palabra == "GTA"):
            palabra = "GLORIETA"
        elif (palabra == "CUSTA." or palabra == "CUSTA"):
            palabra = "CUESTA"
        elif (palabra == "PQUE." or palabra == "PQUE"):
            palabra = "PARQUE"
        elif (palabra == "CTRA." or palabra == "CTRA"):
            palabra = "CARRETERA"
        elif (palabra == "AUTOV." or palabra == "AUTOV"):
            palabra = "AUTOVIA"
        elif (palabra == "CRUCE."):
            palabra = "CRUCE"
        elif (palabra == "ANILLO."):
            palabra = "ANILLO"
        elif (palabra == "PASEO."):
            palabra = "PASEO"
        elif(palabra == "TRVA."):
            palabra = "TRAVESIA"
        elif(palabra == "ESTFE."):
            palabra = "ESTACION_FERROCARRIL"
        elif (palabra == "P?"):
            palabra = "PLAZA"
        elif (palabra == "PZA."):
            palabra = "PLAZA"
        elif (palabra == "CÃŹ" or palabra == "CALL." or palabra == "CALL" or palabra == "C/" or palabra ==
            "C?"):
            palabra = "CALLE"
        elif (palabra == "GRAL"):
            palabra = "GENERAL"
        elif (palabra == "STA." or palabra == "STA"):
            palabra = "SANTA"

        # Mayoritariamente errores de codificación:
        elif(palabra.__contains__("Ý")):
            palabra = palabra.replace("Ý", "Í")
        elif (palabra.__contains__("Ãą")):
```

```
68              palabra = palabra.replace("Ãą", "Ñ")
69          elif (findall('([B-DF-HJ-NP-TV-Z]{1})SS', palabra) != [] or findall('SS([B-DF-HJ-NP-TV-Z]{1})',
                palabra) != []):
70              #P.ej. BSSRBARA ó NARVSSEZ ó GUZMSSN Ó "Ortega Y GASSET" Ó ALCALSS
71              palabra = palabra.replace("SS", "Á")
72          elif (findall('([A-z]{1})Ã˜N', palabra) != []):  #P.ej. SAHAGÃ˜N
73              palabra = palabra.replace("Ã˜N", "ÚN")
74          elif (palabra[0] == '?'):  #P.ej. ?LVAREZ --> ? ASCII 63
75              palabra = palabra.replace("?", "Á")
76          elif (palabra.__contains__(chr(190))):  #P.ej. MOSCARDÃž --> ASCII 190, PERÃžN
77              palabra = palabra.replace(chr(190), "Ó")
78          elif (palabra.__contains__(chr(179))):  #P.ej. MOSCARDÃž --> ASCII 179, YAGÃŁE
79              palabra = palabra.replace(chr(179), "Ú")
80          elif (findall('QUÚ', palabra) != []):  # P.ej MarquÚs --> Marqués
81              palabra = palabra.replace("QUÚ", "QUÉ")
82          elif (palabra == "JESÃ˜S"):
83              palabra = "JESÚS"
84          elif (findall('KM-(\d)', palabra) != []):  # Palabras con numero como KM0
85              palabra = ""
86
87          nuevaPalabra = nuevaPalabra + " " + palabra
88      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 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 con la 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 mayoria de forma automática, y por si volviera a ocurrir con otro dataset.

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

  Debido a los errores de codificación en sus registros, además de los cambios ya expuestos anteriormente en Listing 2.2, se han tenido que realizar manualmente. Son los siguientes:

  - 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 (Tambéin elminado 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 –>Divido en 2 registros con caracteristicas 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 MartÝn ArÚvalo –>Doctor Martín Arévalo

  - ln 1919-1921: Errores en el formato del csv o de codificación. Mismo registro en varias lineas.

  - ln 2040: ENLACE CALLE AMERICIO CON MADRID RÍO –>CALLE AMERICIO

  Para la realización de estos cambios se ha observado el mapa proporcionado por el ayuntamiento <https://datos.madrid.es/egob/new/detalle/auxiliar/mapa.j

  4-calles-tranquilas.kml>y se ha determinado la mejor forma de representar los datos.

- Vias añadidas 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 via posterior a la realizacion 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 ultimos 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 en `https://elpais.com/ccaa/2017/04/28/madrid/1493369660_675682.html` y se han añadido tanto los cambios ya realizados, como los aprobados aun no actualizados en el dataset, para que estén ambos nombres.

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;;;;

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

- Añadir la propiedad esCruce únicamente al dataset de Accidentes.Inferida a partir del nombre de la via aplicando el siguiente código:

Listing 2.3: Función annadirEsCruceAccidentes

```
import csv
import os
from re import findall
def annadirEsCruceAccidentes(nombreCarpeta = "", nombreSinCsv = "", nRowNombre = -1, nFileIni = "-1", nFileFin = "-1"):
    with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
        csvreader = csv.reader(csvfile, delimiter=";")
        file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
        primeraLinea = True
        for row in csvreader:
            if (primeraLinea):
                fila = ";".join(row)
                fila = fila + ";" + "ES_CRUCE"
                primeraLinea = False
            else:
                fila = ";".join(row)
                if (row[nRowNombre].__contains__(" / ")):
                    fila = fila + ";" + "1"
                else:
                    fila = fila + ";" + "0"
            file.write(fila + os.linesep)
        file.close()
```

En el caso de que el accidente haya ocurrido en la intersección entre varias vias, se añaden al nombre siguiendo ciertos patrones (antes englobados en uno: calle1 / calle2) y en esta propiedad se detalla si se ha detectado o no que sea cruce.
Únicamente se realiza para el dataset de accidentes ya que los demás no dispondrán de esta propiedad.

- Añadir la propiedad Tipo de Via a los conjuntos de datos. Inferida a partir del nombre de la via aplicando el siguiente código:

Listing 2.4: Función annadirTipoVia

```
import csv
import os
from re import findall

def annadirTipoVia(nombreCarpeta = "", nombreSinCsv = "", nRowNombre = -1, nFileIni = "-1", nFileFin = "-1"):
    with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
        csvreader = csv.reader(csvfile, delimiter=";")
```

```
        file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
        primeraLinea = True
        for row in csvreader:
            if (primeraLinea):
                fila = ";".join(row)
                fila = fila + ";" + "TIPO_VIA"
                primeraLinea = False
            else:
                tipoVia = getTipoVia(row[nRowNombre])
                fila = ";".join(row)
                fila = fila + ";" + tipoVia
            file.write(fila + os.linesep)
        file.close()
```

En el código mostrado en Listing 2.4 se observa la función annadirTipoVia. Aplcada a los datasets de CallesTranquilas y Accidentes, pero no a Ciclocarriles debido a la naturaleza de los nombres de sus calles, los cuales no contienen estas palabras que determinan el tipo que es (Solo disponen de la parte esencial del nombre).
Esta función recorre las filas del dataset analizando el nombre (ya formateado para que se más sencilla su lectura) y se añade la columna tipoCalle con esta información. Para obtener el tipo se llamará a la función getTipoVia, la cual se muestra en Listing 2.5.

Listing 2.5: Función getTipoVia

```
import csv
import os
from re import findall

def getTipoVia(nombreViaIni = ""):
    if(nombreViaIni == ""):
        return ""
    arr = nombreViaIni.upper().split()
    # Para solo tener las primeras palabras y no por ejemplo Gran VIA
    nombreVia = arr[0]
    if (nombreVia.__contains__("CALLE")):
        return "Calle"
    elif (nombreVia.__contains__("PASEO")):
        return "Paseo"
    elif (nombreVia.__contains__("PLAZA")):
        return "Plaza"
    elif (nombreVia.__contains__("GLORIETA")):
        return "Glorieta"
    elif (nombreVia.__contains__("RONDA")):
        return "Ronda"
    elif (nombreVia.__contains__("CAMINO")):
        return "Camino"
    elif (nombreVia.__contains__("PISTA")):
        return "Pista"
    elif (nombreVia.__contains__("ANILLO")):
        return "Anillo"
    elif (nombreVia.__contains__("CRUCE")):
        return "Cruce"
    elif (nombreVia.__contains__("AUTOVIA")):
        return "Autovia"
    elif (nombreVia.__contains__("CARRETERA")):
        return "Carretera"
    elif (nombreVia.__contains__("PARQUE")):
        return "Parque"
    elif (nombreVia.__contains__("CUESTA")):
        return "Cuesta"
    elif (nombreVia.__contains__("CAÑADA")):
        return "Cañada"
    elif (nombreVia.__contains__("AVENIDA")):
        return "Avenida"
    elif (nombreVia.__contains__("BULEVAR")):
        return "Bulevar"
    elif (nombreVia.__contains__("JARDIN")):
        return "Jardin"
    elif (nombreVia.__contains__("PARTICULAR")):
        return "Particular"
    elif (nombreVia.__contains__("POLIGONO")):
        return "Poligono"
    elif (nombreVia.__contains__("GALERIA")):
        return "Galeria"
    elif (nombreVia.__contains__("ESCALINATA")):
        return "Escalinata"
    elif (nombreVia.__contains__("VIA")):
        return "Via"
    elif (nombreVia.__contains__("PASARELA")):
        return "Pasarela"
    elif (nombreVia.__contains__("PASAJE")):
        return "Pasaje"
    elif (nombreVia.__contains__("PUENTE")):
```

```
60          return "Puente"
61      elif (nombreVia.__contains__("COSTANILLA")):
62          return "Costanilla"
63      elif (nombreVia.__contains__("COLONIA")):
64          return "Colonia"
65      elif (nombreVia.__contains__("CARRERA")):
66          return "Carrera"
67      elif (nombreVia.__contains__("PLAZUELA")):
68          return "Plazuela"
69      elif (nombreVia.__contains__("ACCESO")):
70          return "Acceso"
71      elif (nombreVia.__contains__("POBLADO")):
72          return "Poblado"
73      elif (nombreVia.__contains__("PASADIZO")):
74          return "Pasadizo"
75      elif (nombreVia.__contains__("TRASERA")):
76          return "Trasera"
77      elif (nombreVia.__contains__("SENDA")):
78          return "Senda"
79      elif (nombreVia.__contains__("ARROYO")):
80          return "Arroyo"
81      elif (nombreVia.__contains__("VALLE")):
82          return "Valle"
83      elif (nombreVia.__contains__("AEROPUERTO")):
84          return "Aeropuerto"
85      elif (nombreVia.__contains__("PASO_ELEVADO")):
86          return "Paso Elevado"
87      elif (nombreVia.__contains__("SENDA_CICLABLE")):
88          return "Senda Ciclable"
89      elif (nombreVia.__contains__("PASAJE")):
90          return "Pasaje"
91      else:
92          return ""
```

A pesar de su simpleza actual esta función esta creada además para que si en futuras ocasiones se observasen nuevos tipos o abreviaturas sean sencillas da añadir sin necesidad de modificar el dataset desde el origen.

- Crear una nueva propiedad llamada "palabras clave"que nos permita reducir al máximo los elementos del nombre de la via para asi comparar de forma más satisfactoria con los registros del Callejero de la ciudad.

Listing 2.6: Función annadirPalabrasClave

```
1  import csv
2  import os
3  from re import findall
4
5  def annadirPalabrasClave(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 + ";" + "PALABRAS_CLAVE"
14                 primeraLinea = False
15             else:
16                 palabrasClave = quitarConectores(row[nRowNombre])
17                 palabrasClave = quitarPalabrasConflicto(palabrasClave)
18                 fila = ";".join(row)
19                 fila = fila + ";" + palabrasClave
20             file.write(fila + os.linesep)
21         file.close()
```

En el código de Listinf 2.6 se añade a este nuevo campo una palabra obtenida a partir del nombre a la que le han sido eliminados los conectores y palabras que se consideran genéricas o conflictivas (por ejemplo todas las referidas a tipos de via, elementos ya añadidos a este nuevo campo y que pueden estar en un dataset y no en otro por su caracter genérico).

Listing 2.7: Función quitarConectores

```python
import csv
import os
from re import findall

def quitarConectores(nombreVia = ""):
    nombreVia = nombreVia.upper()
    listaPosibles = [' DEL ', ' DE ', ' Y ', ' LAS ', ' LA ', ' LOS ', ' A ', ' POR ', ' CON ',
                     ' EL ', ' EN ', ' O ', ' I ', ' AL ', ',', '-', 'S/N', ' JUNTO ', '_']
    for a in listaPosibles:
        if(nombreVia.__contains__(a)):
            nombreVia = nombreVia.replace(a, ' ')
    return nombreVia
```

Listing 2.8: Función quitarPalabrasConflicto

```python
import csv
import os
from re import findall


def quitarPalabrasConflicto(nombreVia = ""):
    nombreVia = nombreVia.upper()
    if(nombreVia == ""):
        return ""
    listaPalabrasQuitar = ["CRUCE. ", "CRUCE ", "CALL. ", "CALL ", "CALLE ", "C/", "C/ ", "PASEO. ", "PASEO ",
         "PLAZA.", "PLAZA ", "GTA. ",
    "GTA ", "GLORIETA ", "RONDA. ", "RONDA ",
    "CMNO. ", "CMNO ", "CAMINO ", "PISTA. ", "PISTA ", "AUTOVIA ", "CTRA. ",
    "CTRA ", "CARRETERA ", "PQUE. ", "PQUE ", "PARQUE ", "CUSTA. ", "CUSTA ", "CUESTA ", "CÑADA. ", "CÑADA ",
         "CAÑADA ", "AVDA. ",
    "AVDA ", "AV. ", "AV ", "AVENIDA ", "VIA ", "PASARELA ", "PASAJE ", "PUENTE ", "COSTANILLA ", "COLONIA ",
         "CARRERA ",
    "PLAZUELA ", "BULEVAR ", "BULEV. ", "ESCALINATA ", "JARDÍN ", "JARDIN ", "JARDINES ", "PARTICULAR ",
         "POLÍGONO ", "POLIGONO ", "ACCESO ",
    "POBLADO ", "PASADIZO ", "TRASERA ", "SENDA ", "GALERÍA ", "GALERIA ", "ARROYO ", "VALLE ", "AEROPUERTO ",
         "PASO_ELEVADO", "SENDA_CICLABLE"]

    for a in listaPalabrasQuitar:
        if(nombreVia.__contains__(a) and findall('[A-z]{1}' + a, nombreVia) == []): #Para evitar errores por
             ejemplo en SegoVIA
            arr = nombreVia.split()
            if (arr.index(a.replace(' ', '')) == 0):  # Para evitar suprimir Gran VIA p.ej.
                nombreVia = nombreVia.replace(a, ' ')
            if (nombreVia.__contains__('/') and findall('[A-z]{1}' + '/', nombreVia) == []):
                if (arr.index(a.replace(' ', '')) == arr.index('/') + 1):  # Para cuando es un cruce
                    nombreVia = nombreVia.replace(a, ' ')

        # Podria ser M11 por ejemplo
    if(nombreVia.strip()[0] == '/'):
        # Para eliminar descripciones al inicio y su / p.ej. SENDA_CICLABLE / AV.ROSALES / CARRET.VILLAV. A
             VALLECA
        nombreVia = nombreVia.replace('/', '', 1)
    return nombreVia
```

En los códigos mostrado en Listing 2.7 y 2.8 se observan las funciones antes mencionadas. Los conectores han sido obtenidos de los propios datasets aquí utilizados por lo tanto pueden ocurrir errores puntuales en el caso de que apareciesen nuevos no esperados. Se ha optado por esta opción en vez de buscar palabras monosílabas por ejemplo debido a que da mayor seguridad en cuanto a errores, y es preferible detectar un conector sin omitir que una palabra eliminada que si era parte esencial.
En el caso de las palabras conflicto, la función está destinada a eliminar los tipos de vias como ya se mencionó anteriormente.

Cabe destacar que esta función es llamada por todos los datasets incluido el Callejero de Madrid. Ésto permite que todos por igual hayan recibido las mismas transformaciones para obtener sus palabras clave, de tal forma que sea más sencillo su cruce para obtener los identificadores.

- Para finalizar esta primera fase, todos los vocabularios son comparados con el Callejero de Madrid. Para ello se comparan las palabras claves de sus nombres, anteriormente obtenidas, y se comprueba que sean las mismas para así asignarles el identificador. Éste identificador lo proporciona el ayuntamiento, por ello es importante este cruce con el

Callejero, del cual se obtendrán.

Listing 2.9: Función crearFichNombresId

```
import csv
import os
import difflib
import unicodedata

def crearFichNombresId(nombreCarpeta = "", nombreSinCsv = "", nFileIni = "-1", nFileFin = "_IDs_000",
                   nRowEsCruce = -1, nRowPalabrasClave = "-1", nRowTipoVia="-1", tieneTipoCalle = False):
with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
    csvreader = csv.reader(csvfile, delimiter=";")
    file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
    primeraLinea = True
    nCorrect = 0
    nIncorrect = 0
    contadorFila = 0
    for row in csvreader:
        if (primeraLinea):
            fila = ";".join(row)
            fila = fila + ";" + "ID_CALLE"
            primeraLinea = False
        else:
            arrCalles = []
            if (nRowEsCruce != -1 and row[nRowEsCruce] == "1"):  # Si es cruce tendra varios ids
                arrCallesCruce = getArrCalles(row[nRowPalabrasClave].upper())
            else:
                arrCallesCruce = [row[nRowPalabrasClave].upper()]

            listIds = []
            nElem = 0
            for nombre in arrCallesCruce:
                #TODO: quitar
                '''if(not(nombre.__contains__("EUGENIA"))):
                    continue'''
                #TODO: quitar
                segundaVuelta = False
                nVuelta = 0
                while (nVuelta == 0 or segundaVuelta):
                    nVuelta = nVuelta + 1
                    with open("callejeroCSV" + "/" + "nombres_IDs" + ".csv") as csvCallejero:
                        csvreaderCallej = csv.reader(csvCallejero, delimiter=";")
                        primeraLinea = True
                        encontrado = False
                        for rowCallj in csvreaderCallej:
                            if (primeraLinea):
                                primeraLinea = False
                            else:
                                try:
                                    # TODO: quitar
                                    '''if (rowCallj[11].__contains__("EUGENIA")):
                                        print("OK2")'''
                                    # TODO: quitar
                                    if((nElem >0 or not(tieneTipoCalle) or
                                    (segundaVuelta and esTipoCalleOmitible(row[nRowTipoVia].upper()))
                                    or checkearPalabras(rowCallj[1].upper(), row[nRowTipoVia].upper(),False))
                                        and
                                    #Porque si es la segunda entrada no contiene el tipo de via
                                    checkearPalabras(rowCallj[11].upper(), nombre.upper(), segundaVuelta)):
                                        listIds.append(rowCallj[0])
                                        encontrado=True
                                        nCorrect = nCorrect+1
                                        segundaVuelta=False
                                        break
                                except IndexError:
                                    print("Error")
                                    # Posible error de que haya una linea vacia extra al final
                        if (not(encontrado) and not(segundaVuelta)):
                            segundaVuelta = True
                            continue
                        if(not(encontrado) and segundaVuelta):
                            segundaVuelta = False
                            listIds.append("-1")
                            nIncorrect = nIncorrect + 1
                        csvCallejero.close()
                        nElem = nElem + 1
            fila = ";".join(row)
            fila = fila + ";" + listIds.__str__()
            contadorFila = contadorFila+1
            if(contadorFila % 20 == 0):
                print("Fila: " , contadorFila)
        file.write(fila + os.linesep)
    file.close()
    print("Incorrectas: ", nIncorrect, " || Correctas: ", nCorrect)
```

En Listing 2.9 se muestra la funcion crearFichNombresId. Ésta funcion realiza numerosas vueltas para cada registro. Primero selecciona el elemento del dataset que se va a analizar

(que aun no tenga id). Segundo comprueba que no sea un cruce, y en caso de serlo llama a la funcion getArrCalles (Listing 2.10) para obtener las dos o más vias que lo componen (y comprobar los ids para cada una de ellas). Tercero recorre el dataset del callejero para comparar las palabras clave y obtener coincidencias. Para ello sigue los siguientes pasos:
1. Compara el tipo de via, en caso de no coincidir será distinto tipo de calle y por tanto diferente (esto no es válido para el caso de cruces ya que la via secundaria no tendría tipo, por tanto se omite).
2. En caso de no coincidir en ningún elemento, vuelve a dar otra vuelta al callejero rebajando las restricciones. No se comprueba que sea del mismo tipo de via y se aplica una fórmula que se detallará posteriormente para que se permitan letras añadidas, suprimidas o modificadas (por erratas, plurales, femeninos y masculinos...)
Por último se imprime por pantalla los conteos de los elementos que han coincidido y los que no para determinar la eficacia del proceso (aproximado, ya que pueden haberse cometido errores no detectados y que se hayan dado paso a identificadores mal asociados).

Listing 2.10: Función getArrCalles

```
import csv
import os
import difflib
import unicodedata

def getArrCallesRec(nombreDataset = ""):

    listaNombres = []
    if (nombreDataset.__contains__('/')):
        n1 = nombreDataset.split().index('/')
        txtRestante = " ".join(nombreDataset.split()[n1 + 1:])
        nombre = " ".join(nombreDataset.split()[0:n1])
        elemRecursiv = getArrCallesRec(txtRestante)
        listaNombres = [nombre, elemRecursiv]
    else:
        listaNombres = [nombreDataset]

    return listaNombres

def getArrCalles(nombreDataset = ""):
    list1 = getArrCallesRec(nombreDataset)
    txt = list1.__str__().replace('[', '').replace(']', '')
    list3 = []
    nombre=""

    for elem in txt.split():
        if(elem.__contains__("'") and nombre != ""):
            nombre = nombre + " " + elem.replace("'", "").replace(",", "")
            list3.append(nombre)
            nombre = ""
        elif (elem.__contains__("'") and nombre == "" and elem.replace("'", "", 1).__contains__("'")):
            nombre = elem.replace("'", "").replace(",", "")
            list3.append(nombre)
            nombre = ""
        elif (elem.__contains__("'") and nombre == ""):
            nombre = elem.replace("'", "").replace(",", "")
        else:
            nombre = nombre + " " + elem
    return list3
```

En este codigo se obtiene un array de calles a partir del nombre. Siguiendo el patrón "calle1 / calle2 " que se ha utilizado para los cruces, se separan sus elementos devolviendo un array con los nombres completos de cada una de ellas separados.

Listing 2.11: Función checkearPalabras

```
import csv
import os
import difflib
import unicodedata
def checkearPalabras(nombreCallej = "", nombreDataset = "", segundaVuelta = False):
    # En la segunda vuelta también se omite el tipo de calle
```

```
7     nombreDataset1 = nombreDataset.upper().replace("Á", "A").replace("É", "E").replace("Í", "I") \
8         .replace("Ó", "O").replace("Ú", "U").replace("Ü", "U").replace(" ", "")
9     nombreCallej1 = nombreCallej.upper().replace("Á", "A").replace("É", "E").replace("Í", "I") \
10        .replace("Ó", "O").replace("Ú", "U").replace("Ü", "U").replace(" ", "")
11    if(nombreDataset1 == nombreCallej1):    # Para aumentar la velocidad al ser cambios menores
12        return True
13    if(unicodedata.normalize('NFKD', nombreCallej).encode('ASCII', 'ignore').strip().upper() \
14       == unicodedata.normalize('NFKD',nombreDataset).encode('ASCII', 'ignore').strip().upper()):
15        return True
16    #       ------------------------------------------------------------------------------------------------------------------------------------------
17
18    if(segundaVuelta):   # Excepciones cuando hay pequeños cambios que pueden deberse a errores ortograficos
19        nombreDataset = nombreDataset.upper().replace("Á", "A").replace("É", "E").replace("Í", "I") \
20            .replace("Ó", "O").replace("Ú", "U").replace("Ü", "U")
21        nombreCallej = nombreCallej.upper().replace("Á", "A").replace("É", "E").replace("Í", "I") \
22            .replace("Ó", "O").replace("Ú", "U").replace("Ü", "U")
23        # ------------------------------------------------------------------------------
24        # P.ej. COMANDANTE ZORITA
25        nombreDataset1 = nombreDataset
26        nombreCallej1 = nombreCallej
27        if(nombreDataset.__contains__("COMANDANTE")):
28            nombreDataset1 = nombreDataset.replace("COMANDANTE", "AVIADOR")
29        if (nombreCallej.__contains__("COMANDANTE")):
30            nombreCallej1 = nombreCallej.replace("COMANDANTE", "AVIADOR")
31        if(nombreCallej1.__contains__("AVIADOR") and
32            (unicodedata.normalize('NFKD', nombreCallej1).encode('ASCII', 'ignore').strip().upper() \
33           == unicodedata.normalize('NFKD',nombreDataset1).encode('ASCII', 'ignore').strip().upper())):
34            return True
35        # ------------------------------------------------------------------------------
36        if(nombreCallej.replace(" ", "").__contains__("M30") or
37            nombreCallej.replace(" ", "").__contains__("M40")):
38            return False #Si no la ha detectado ya es que hay problemas y las Bicicletas no pueden circular
                  por ahi
39        # ------------------------------------------------------------------------------
40        # Por ejemplo FERNADO VI --> FERNANDO VI
41        longCadena1 = nombreDataset.__len__()
42        longCadena2 = nombreCallej.__len__()
43        if(longCadena1 < (longCadena2*0.7) or longCadena1 > (longCadena2 * 1.3)) :
44            # Si el tamaño de la cadena difiere mucho
45            return False
46        diff = difflib.ndiff(nombreCallej1.replace(" ", ""), nombreDataset1.replace(" ", ""))
47        diferenciastxt = ''.join(diff)
48        # ------------------------------------------------------------------------------
49        #Primero comprobar 1 solo error en total
50        if ((diferenciastxt.count('+') + diferenciastxt.count('-')) <= 1
51            and not (nombreCallej.__contains__("BARROS"))  #  BARRIOS  ::   BARROS
52            and not (nombreCallej.__contains__("OLIVAR"))  # Bolivar - Olivar
53            and not (nombreCallej.__contains__("MANZANAR"))  # MANZANARES  ::   MANZANAR
54            ):
55            # Añadiendo una S por ejemplo
56            print("Posible Conicidencia Simple: ", nombreDataset, " :: ", nombreCallej)
57            return True
58        # ------------------------------------------------------------------------------
59        # Segundo comprobar 1 sustitucion
60        if (diferenciastxt.count('+') <= 1 and diferenciastxt.count('-') <=1
61            and not(nombreDataset.__contains__("HORTALEZA")) #Hortaleza - Fortaleza
62            and not (nombreCallej.__contains__("GALENA"))  # Gilena - Galena
63            and not (nombreCallej.__contains__("PEAL"))  # Real - Peal
64            and not (nombreCallej.__contains__("OLIVAR"))  # Bolivar - Olivar
65            and not (nombreCallej.__contains__("CRUCES"))  # RUICES  ::   CRUCES
66            and not (nombreCallej.__contains__("HAYA"))  # RAYA  ::   HAYA
67            and not (nombreCallej.__contains__("OCA"))  # OÑA  ::   OCA
68            and not (nombreCallej.__contains__("MANZANAR"))  # MANZANARES  ::   MANZANAR
69            and not (nombreDataset.__contains__("CANDILEJAS") and nombreCallej.__contains__("CANALEJAS"))  #
                  CANDILEJAS  ::   CANALEJAS
70            and not (nombreCallej.__contains__("LIMON"))  # JUAN VIGON  ::   JUAN LIMON
71            and not (nombreCallej.__contains__("CENICIENTOS"))  # CENICIENTA  ::   CENICIENTOS
72            and not (nombreCallej.__contains__("CANTERAS"))  # MANOTERAS  ::   CANTERAS
73            and not (nombreDataset.__contains__("MANOTERAS"))  # MANOTERAS  ::   SANTERAS
74            and not (nombreCallej.__contains__("GOR"))  # SIERRA GADOR  ::   SIERRA GOR
75            and not (nombreDataset.__contains__("GADOR"))  # SIERRA GADOR  ::   SIERRA GUDAR
76            and not (nombreCallej.__contains__("ERASMO"))  # SANZ RASO  ::   SAN ERASMO
77            and not (nombreCallej.__contains__("ALIO"))  # SANTIAGO AMON  ::   SANTIAGO ALIO
78            and not (nombreDataset.__contains__("PARVILLAS"))  # PARVILLAS  ::   MARAVILLA
79            and not (nombreDataset.__contains__("MARMOLINA"))  # MARMOLINA  ::   CAROLINA
80            and not (nombreCallej.__contains__("SAMANIEGO"))  # SAN DIEGO  ::   SAMANIEGO
81            and not (nombreDataset.replace(" ", "").__contains__("SANDIEGO"))  # SAN DIEGO  ::   SAN DACIO
82            and not (nombreCallej.__contains__("BARROS"))  # BARRIOS  ::   BARROS
83            and not (nombreDataset.replace(" ", "").__contains__("MADRIDRIO"))  #  MADRID RIO  ::   MADRE DIOS
84            ):
85            print("Posible Conicidencia Sustitucion: ", nombreDataset, " :: ", nombreCallej)
86            return True
87        # ------------------------------------------------------------------------------
88        # Dependiendo de la longitud de la palabra acepta 1 error o más
89        if (diferenciastxt.count('+') <= (longCadena1/10+1) and diferenciastxt.count('-') <=(longCadena1/10+1)
90            and not (nombreDataset.__contains__("HORTALEZA")) #Hortaleza - Fortaleza
91            and not (nombreCallej.__contains__("GALENA"))  # Gilena - Galena
92            and not (nombreCallej.__contains__("PEAL"))  # Real - Peal
93            and not (nombreCallej.__contains__("OLIVAR"))  # Bolivar - Olivar
94            and not (nombreCallej.__contains__("CRUCES"))  # RUICES  ::   CRUCES
95            and not (nombreCallej.__contains__("HAYA"))  #  RAYA  ::   HAYA
96            and not (nombreCallej.__contains__("OCA"))  #  OÑA  ::   OCA
97            and not (nombreCallej.__contains__("MANZANAR"))  #  MANZANARES  ::   MANZANAR
98            and not (nombreDataset.__contains__("CANDILEJAS") and nombreCallej.__contains__("CANALEJAS") )  #
```

```
                CANDILEJAS  ::   CANALEJAS
            and not (nombreCallej.__contains__("LIMON"))  # JUAN VIGON  ::   JUAN LIMON
            and not (nombreCallej.__contains__("CENICIENTOS"))  # CENICIENTA  ::   CENICIENTOS
            and not (nombreCallej.__contains__("CANTERAS"))  #  MANOTERAS  ::   CANTERAS
            and not (nombreDataset.__contains__("MANOTERAS"))  # MANOTERAS  ::   SANTERAS
            and not (nombreCallej.__contains__("GOR"))  # SIERRA GADOR  ::   SIERRA GOR
            and not (nombreDataset.__contains__("GADOR"))  # SIERRA GADOR  ::   SIERRA GUDAR
            and not (nombreCallej.__contains__("ERASMO"))  #  SANZ RASO  ::   SAN ERASMO
            and not (nombreCallej.__contains__("ALIO"))  #  SANTIAGO AMON  ::   SANTIAGO ALIO
            and not (nombreDataset.__contains__("PARVILLAS"))  # PARVILLAS  ::   MARAVILLA
            and not (nombreDataset.__contains__("MARMOLINA"))  # MARMOLINA  ::   CAROLINA
            and not (nombreCallej.__contains__("SAMANIEGO"))  # SAN DIEGO  ::   SAMANIEGO
            and not (nombreDataset.replace(" ", "").__contains__("SANDIEGO"))  # SAN DIEGO  ::   SAN DACIO
            and not (nombreCallej.__contains__("BARROS"))  #  BARRIOS  ::   BARROS -
            and not (nombreDataset.replace(" ", "").__contains__("MONTEAYA"))  # MONTE AYA  ::   MONTANA -
            and not (nombreCallej.__contains__("SANTERAS"))  # SANZ RASO  ::   SANTERAS -
            and not (nombreDataset.replace(" ", "").__contains__("PALOROSA"))  # PALO ROSA  ::   ALCORISA
            and not (nombreDataset.__contains__("SACROMONTE"))  #  SACROMONTE  ::   CERRO MONTE -
            and not (nombreDataset.replace(" ", "").__contains__("EDUARDOAUNOS"))  #  EDUARDO AUNOS  ::
                 EDUARDO UROSA -
            and not (nombreDataset.replace(" ", "").__contains__("MADRIDRIO"))  # MADRID RIO  ::   MADRE DIOS
            ):
            # Si se han añadido o quitado 2 o menos letras, se puede considerar igual
            print("-------Posible Conicidencia: ", nombreDataset, " :: ", nombreCallej)
            return True
        # ------------------------------------------------------------------------------------
    return False
```

En esta función primero se sustituyen los elementos como tildes y se comparan para obtener coincidencias.
En caso de que ésta no se diese, el programa daría una segunda vuelta, a la cual la función actuaria de diferente manera. Primero igual, ya que podría darse coincidencia por haber eliminado el tipo de via de la restricción. Si tampoco se diese el caso se comprobaría primero con una letra añadida o eliminada, despues con 2 (lo que permitiría letras cambiadas) y despúes con una diferencia variable dependiendo de su dimensión.
Se han añadido numerosas restricciones observadas en el proceso y también se imprime por pantalla los cambios realizados para que puedan ser vistos y modificados en caso de errores.

Listing 2.12: Función esTipoCalleOmitible

```
import csv
import os
import difflib
import unicodedata

def esTipoCalleOmitible(nombreVia = ""):
    if(nombreVia == ""):
        return True
    nombreVia = nombreVia.upper().replace("Á", "A").replace("É", "E").replace("Í", "I") \
            .replace("Ó", "O").replace("Ú", "U").replace("Ü", "U").replace(" ", "")
    if (nombreVia.__contains__("CALLE")):
        return True
    elif (nombreVia.__contains__("PASEO")):
        return True
    elif (nombreVia.__contains__("PLAZA")):
        return True
    elif (nombreVia.__contains__("GLORIETA")):
        return True
    elif (nombreVia.__contains__("RONDA")):
        return False
    elif (nombreVia.__contains__("CAMINO")):
        return True
    elif (nombreVia.__contains__("PISTA")):
        return True
    elif (nombreVia.__contains__("ANILLO")):
        return True
    elif (nombreVia.__contains__("CRUCE")):
        return True
    elif (nombreVia.__contains__("AUTOVIA")):
        return False
    elif (nombreVia.__contains__("CARRETERA")):
        return True
    elif (nombreVia.__contains__("PARQUE")):
        return False
    elif (nombreVia.__contains__("CUESTA")):
        return True
    elif (nombreVia.__contains__("CAÑADA")):
        return False
    elif (nombreVia.__contains__("AVENIDA")):
        return True
    elif (nombreVia.__contains__("BULEVAR")):
```

```
42          return True
43      elif (nombreVia.__contains__("JARDIN")):
44          return False
45      elif (nombreVia.__contains__("PARTICULAR")):
46          return False
47      elif (nombreVia.__contains__("POLIGONO")):
48          return False
49      elif (nombreVia.__contains__("GALERIA")):
50          return False
51      elif (nombreVia.__contains__("ESCALINATA")):
52          return False
53      elif (nombreVia.__contains__("VIA")):
54          return True
55      elif (nombreVia.__contains__("PASARELA")):
56          return True
57      elif (nombreVia.__contains__("PASAJE")):
58          return True
59      elif (nombreVia.__contains__("PUENTE")):
60          return False
61      elif (nombreVia.__contains__("COSTANILLA")):
62          return False
63      elif (nombreVia.__contains__("COLONIA")):
64          return False
65      elif (nombreVia.__contains__("CARRERA")):
66          return True
67      elif (nombreVia.__contains__("PLAZUELA")):
68          return False
69      elif (nombreVia.__contains__("ACCESO")):
70          return True
71      elif (nombreVia.__contains__("POBLADO")):
72          return False
73      elif (nombreVia.__contains__("PASADIZO")):
74          return False
75      elif (nombreVia.__contains__("TRASERA")):
76          return False
77      elif (nombreVia.__contains__("SENDA")):
78          return True
79      elif (nombreVia.__contains__("ARROYO")):
80          return False
81      elif (nombreVia.__contains__("VALLE")):
82          return False
83      elif (nombreVia.__contains__("AEROPUERTO")):
84          return False
85      elif (nombreVia.__contains__("PASO_ELEVADO")):
86          return False
87      elif (nombreVia.__contains__("SENDA_CICLABLE")):
88          return False
89      return True
```

Esta función es llamada en la segunda vuelta del bucle por si el tipo de via al que hace referencia no pudiese ser obviado. Esto es por ejemplo en Parque, Polígono... No es posible comparar un parque con una calle que tengan el mismo nombre y aceptar que es válido, en cambio una avenida con una calle si. Para ello se debe especificar los tipos que pueden ser comparables y los que no.

2.6. Transformaciones en los vocabularios: Fase 2

En esta segunda fase de transformaciones se realizarán cambios menores. Una vez ya se han realizado los cambios necesarios para obtener las palabras clave y los identificadores, se realizarán cambios de nombres en las propiedades de los datasets y se añadirán nuevos elementos como por ejemplo municipio.
Estos cambios se realizarán a partir de lo generado en la Fase1.

En esta sección los cambios serán diferentes por cada Dataset por tanto se detallarán por separado.

2.6.1. Cambios relativos a CicloCarriles

Los relativos a CicloCarriles serán los siguientes:

- Resumen de cambios:
  out: CAPA; TX_NOMBRE; MaxSimpTol; MinSimpTol; FECHA; Longitud; TIPO_VIA; PALABRAS_CLAVE; ID_CALLE
  in: tipoUso; nombreVia; distMaxExclusBici; longitud; TIPO_VIA; PALABRAS_CLAVE; ID_CALLE; carrilExclusBici; municipio

  Para el caso de tipoUso, nombreVia, distMaxExclusBici y longitud no se realizará ninguna modificación en los registros. Únicamente se ha cambiado el nombrado.
  Se han eliminado las propiedades MinSimpTol y FECHA, junto con sus datos, debido a que no aportan información añadida y por ejemplo en el caso de la distancia minima, se sustituirá por la propiedad carrilExclusBici que dará la misma información de forma más clara con otros tipos de datos.
  Se han añadido carrilExclusBici y municipio.
  Se modificarán los datos de Longitud, ya que están en kilómetros y es preferible, para la longitud de las calles en una ciudad, que se representen en metros por ser el formato estándar.

- Cambios en los nombres y añadir carrilExclusBici y municipio:

Listing 2.13: Función variosCambiosEnCicloCarriles

```
import csv
import os
from re import findall

def variosCambiosEnCicloCarriles(nombreCarpeta="", nombreSinCsv="", nFileIni = "-1", nFileFin = "-1"):
    with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
        csvreader = csv.reader(csvfile, delimiter=";")
        file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
        primeraLinea = True
        nColLong = -1
        nColDistMin = -1
        for row in csvreader:
            if (primeraLinea):
                fila = ";".join(row)
                nColLong = row.index("Longitud")
                nColDistMin = row.index("MinSimpTol")
                # Cambio nombres: tipoUso, nombreVia, distMaxExclusBici y longitud
                fila = fila.replace("Longitud", "longitud").replace("CAPA", "tipoUso")\
                           .replace("TX_NOMBRE", "nombreVia").replace("MaxSimpTol","distMaxExclusBici")
                #Añadir carrilExclusBici:
                fila = fila + ";" + "carrilExclusBici"
                # Añadir municipio:
                fila = fila + ";" + "municipio"
                primeraLinea = False
```

```
25              else:
26                  # Cambiar la longitud de KM a metros
27                  row[nColLong] = (float(row[nColLong].replace(',', '.')) * 1000).__str__()
28                  fila = ";".join(row)
29
30                  #Añadir carrilExclusBici:
31                  if(float(row[nColDistMin].replace(',', '.')) == 0.0):
32                      fila = fila + ";" + "0" # No tiene carril exclusivo para bicicletas
33                  else:
34                      fila = fila + ";" + "1" # Si tiene
35
36                  # Añadir municipio:
37                  fila = fila + ";" + "Madrid"
38
39              file.write(fila + os.linesep)
40          file.close()
```

En el código mostrado primero se cambian los nombres de las propiedades a tipoUso, nombreVia, distMaxExclusBici y longitud. Posteriormente se añaden las de carrilExclusBici y municipio. Para su cálculo, en el primer caso se multiplica por 1.000 la longitud (para convertir kilometros en metros y en el segundo se añade la palabra Madrid, ya que siempre será el municipio para los datos aqui utilizados.

Listing 2.14: Función eliminarColumnaCompleta

```
1  import csv
2  import os
3  from re import findall
4
5  def eliminarColumnaCompleta(nombreCarpeta="", nombreSinCsv="", nFileIni = "-1", nFileFin = "-1", nombreColumna
       = ""):
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         nRowColumna = -1
11         for row in csvreader:
12             if (primeraLinea):
13                 nRowColumna = row.index(nombreColumna)
14                 del row[nRowColumna]
15                 fila = ";".join(row)
16                 primeraLinea = False
17             else:
18                 del row[nRowColumna]
19                 fila = ";".join(row)
20             file.write(fila + os.linesep)
21         file.close()
```

En el codigo de Listing 2.14 se muestra una función que se reutilizará para todos los datasets. En ella se elimina una columna completa, tanto su nombre como sus datos en todos los registros.
En el caso del dataset de CicloCarriles se eliminarán las columnas FECHA y MinSimplTol, como se observa en las siguientes llamadas a esta función.

Listing 2.15: Llamadas a eliminarColumnaCompleta

```
1 eliminarColumnaCompleta(nombreCarpeta, nombreSinCsv, "2", "3", "FECHA")
2 eliminarColumnaCompleta(nombreCarpeta, nombreSinCsv, "3", "4", "MinSimpTol")
```

2.6.2. Cambios relativos a Accidentes de Bicicletas

Los relativos a Accidentes de Bicicletas serán los siguientes:

- Resumen de cambios:
  out: Nº EXPEDIENTE; FECHA; HORA; CALLE; NÚMERO; DISTRITO; TIPO ACCIDENTE; ESTADO METEREOLÓGICO; TIPO VEHÍCULO; TIPO PERSONA; RANGO EDAD; SEXO; LESIVIDAD*;esCruce;tipoVia;palabrasClave;idVia
  in: numeroExpediente; fecha; hora; nombreVia; portal; distrito; tipoAccidente; meteorologia; tipoVehiculo; tipoPersonaAfectada; typicalAgeRange; gender; lesividad; esCruce; tipoVia; palabrasClave; idVia; municipio

  Se modifican el nombre de las columnas NºExpediente, FECHA, HORA, CALLE, NÚMERO , DISTRITO, TIPO ACCIDENTE, ESTADO METEREOLÓGICO, TIPO VEHÍCULO, TIPO PERSONA, RANGO EDAD, SEXO, LESIVIDAD* para que sea más entendible y coincida con la definición antes desarrollada.
  Se han modificado los datos de las columnas typicalAgeRange (RANGO EDAD) para que seguir el formato definido por schema.org (añoInicio - añoFin) (p.ej. 10-12)
  Se han mofificado los datos de gender (SEXO) de para que también concuerden con los definidos en schema.org. En este caso el dataset solo proporciona los valores Hombre y Mujer, por lo tanto solo se sustituirán por Male y Female.
  Se añade la columna municipio asignandole a cada registro el valor "Madrid"para esta propiedad.

Listing 2.16: Función variosCambiosEnAccidentesBici

```
import csv
import os
from re import findall

def variosCambiosEnAccidentesBici(nombreCarpeta="", nombreSinCsv="", nFileIni = "-1", nFileFin = "-1"):
    with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
        csvreader = csv.reader(csvfile, delimiter=";")
        file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
        primeraLinea = True
        nColTypicalAge = -1
        nColGender = -1
        for row in csvreader:
            if (primeraLinea):
                fila = ";".join(row)
                nColTypicalAge = row.index("RANGO EDAD")
                nColGender = row.index("SEXO")
                # Cambio nombres: Nº Expediente, FECHA, HORA, CALLE, NÚMERO , DISTRITO, TIPO ACCIDENTE,
                #                 ESTADO METEREOLÓGICO, TIPO VEHÍCULO, TIPO PERSONA, RANGO EDAD, SEXO,
                      LESIVIDAD*
                fila = fila.replace("Nº  EXPEDIENTE", "numeroExpediente").replace("FECHA", "fecha")\
                           .replace("HORA", "hora").replace("CALLE","nombreVia").replace("NÚMERO","portal")\
                           .replace("DISTRITO", "distrito").replace("TIPO ACCIDENTE","tipoAccidente")\
                           .replace("ESTADO METEREOLÓGICO","meteorologia").replace("TIPO VEHÍCULO",
                                "tipoVehiculo")\
                           .replace("TIPO PERSONA", "tipoPersonaAfectada").replace("RANGO EDAD",
                                "typicalAgeRange")\
                           .replace("SEXO", "gender").replace("LESIVIDAD*", "lesividad")
                # Añadir municipio:
                fila = fila + ";" + "municipio"
                primeraLinea = False
            else:
                # Modificar el Genero:
                generoEspan = row[nColGender]
                if(generoEspan == "Hombre"):
                    row[nColGender] = "Male"
                elif(generoEspan == "Mujer"):
                    row[nColGender] = "Female"

                # Modificar TypicalAgeRange
                row[nColTypicalAge] = getTypicalAgeRangeOk(row[nColTypicalAge])

                fila = ";".join(row)
                # Añadir municipio:
                fila = fila + ";" + "Madrid"
```

```
42            file.write(fila + os.linesep)
43        file.close()
```

En 2.16 se muestra el codigo de los cambios en el dataset de Accidentes de Bicicleta.
Primero se cambian los nombres antes mencionados por los nuevos para que coincidan con los definidos en el vocabulario.
Segundo se realizan las transformaciones para typicalAgeRange y para gender. Para Gender únicamente el dataset tiene 2 valores posibles, Hombre y Mujer, por lo que se ha optado por considerar únicamente esas posibilidades y sustituirlos por sus nombres en inglés, lo definido por schema.org. Si se diese el caso de nuevas posiblidades en esta propiedad se deberían añadir. En [1] `https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=20f4a87ebb65b510VgnVCM1000001d4a900aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default` se puede consultar la documentación asociada a este dataset y hasta el momento solo puede haber estas 2 opciones y no asignado.
Para typicalAgeRange se ha creado una nueva función a la que llama para obtener el rango de edad en el formato estandar definido por schema.org. Se muestra en el siguiente código:

Listing 2.17: Función getTypicalAgeRangeOk

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

En 2.17 comprueba primero el formato en el que vienen la totalidad de los datos de esta columna (De 10 A 20 AÑOS) y lo devuelve en el formato estandar de schema.org (10-20). En caso de que el dataset contenga la información con distinta representación, se comprueba de forma básica si es posible encontrar en todo el texto dos números separados. En este caso se considerará que el primero es la fecha inicio y el segundo la final. Si es posible esta combinación se devolverán con el formato antes mencionado.
Por si se diera el caso de que la fecha estuviese cambiada de sentido, se comprueba que el inicio sea menor que el fin, en caso contrario se invierten para seguir el estandar.

2.6.3. Cambios relativos a Calles Tranquilas

Los relativos a Calles Tranquilas serán los siguientes:

- Resumen de cambios:
  out: OBJECTID; ID_EJE; ID_GRUPO; ID_TIPO; TX_CAPA; NU_DIRECCI; NU_DOBLE_S; TX_NOMBRE; TX_NOMBRE_; Shape_Leng; tipoVia; palabrasClave; idVia
  in: idObject; idEje; idGrupo; tipoUso; nuDireccion; nuDobleS; nombreVia; nombreViaReduc; longitud; tipoVia; palabrasClave; idVia; municipio

  Se modifican el nombre de las columnas OBJECTID, ID_EJE, ID_GRUPO, TX_CAPA, NU_DIRECCI, NU_DOBLE_S, TX_NOMBRE, NU_DOBLE_S y Shape_Leng para que sean más entendibles y coincida con la definición antes desarrollada.
  En este caso los datos de la longitud si están en metros por lo tanto no es necsario cmabiarlos.
  Se ha eliminado la columna ID_TIPO ya que representa con un codigo numérico lo mismo que TX_CAPA (llamada a partir de ahora tipoUso). Se ha optado por esta última propiedad ya que comparte valores con tipoUso del dataset CicloCarriles, y por lo tanto se entiende que es más utilizado en otros dataset que la primera propiedad y que, al representar lo mismo, es mñas recomendable conservar si la segunda.
  Se añade la columna municipio asignandole a cada registro el valor "Madrid"para esta propiedad.

Listing 2.18: Función variosCambiosEnCallesTranquilas

```
import csv
import os
from re import findall

def variosCambiosEnCallesTranquilas(nombreCarpeta="", nombreSinCsv="", nFileIni = "-1", nFileFin = "-1"):
    with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
        csvreader = csv.reader(csvfile, delimiter=";")
        file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
        primeraLinea = True
        for row in csvreader:
            if (primeraLinea):
                fila = ";".join(row)
                # Cambio nombres: OBJECTID, ID_EJE, ID_GRUPO, TX_CAPA, NU_DIRECCI, NU_DOBLE_S,
                #                 TX_NOMBRE, TX_NOMBRE_ y Shape_Leng
                fila = fila.replace("OBJECTID", "idObject").replace("ID_EJE", "idEje")\
                           .replace("ID_GRUPO", "idGrupo").replace("TX_CAPA","tipoUso").replace("NU_DIRECCI","nuDireccion")\
                           .replace("NU_DOBLE_S", "nuDobleS").replace("TX_NOMBRE_","nombreViaReduc")\
                           .replace("TX_NOMBRE","nombreVia").replace("Shape_Leng","longitud")
                # Añadir municipio:
                fila = fila + ";" + "municipio"
                primeraLinea = False
            else:
                fila = ";".join(row)
                # Añadir municipio:
                fila = fila + ";" + "Madrid"
            file.write(fila + os.linesep)
        file.close()
```

En 2.18 se muestra el codigo de los cambios en el dataset de Calles Tranquilas
En este caso se cambian los nombres antes mencionados por los nuevos para que coincidan con los definidos en el vocabulario y se añade la columna municipio con el valor "Madrid.en todos los registros.

En el siguiente código se muestra la llamada a la funcion eliminarColumnaCompleta

?? para suprimir del dataset ID_TIPO.

Listing 2.19: Llamadas a eliminarColumnaCompleta

```
1 eliminarColumnaCompleta(nombreCarpeta, nombreSinCsv, "2", "3", "ID_TIPO")
```

Capítulo 3

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&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default

[4] https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=435a7cd5de319410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default

[5] https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=a320f5ac548f4410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default

[6] https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=a320f5ac548f4410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default

[7] https://datos.madrid.es/sites/v/index.jsp?vgnextoid=b3c41f3cf6a6c410VgnVCM2000000c205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD

[8] https://datos.madrid.es/egob/new/detalle/auxiliar/mapa.jsp?geoUrl=/egob/catalogo/205115-4-calles-tranquilas.kml

[9] https://datos.ign.es/def/btn100/index-es.html#calzada

[10] https://www.ine.es/daco/daco42/codmun/codmunmapa.htm

[11] https://datos.ign.es/def/btn100/index-es.html#calzada

[12] http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero/index-en.html#tipoVia

[13] http://vocab.linkeddata.es/datosabiertos/def/sector-publico/territorio/index-en.html#Municipio

[14] http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero/index-en.html#Via

[15] https://schema.org/gender

[16] https://lists.w3.org/Archives/Public/public-schemaorg/2019Oct/0013.html

[17] https://schema.org/typicalAgeRange

[18] http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero/index-en.html#Portal

[19] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms/identifier

[20] https://ciudadesabiertas.es/vocabularios/#CatálogoVocabularios

Anexo

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