



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 usuario. 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](http://ciudadesAbiertas.es). 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 búsqueda 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.

Tabla de contenidos

1. Introducción	1
2. Desarrollo	3
2.1. Vocabulario de Accidentes de Bicicletas	4
2.1.1. Clases	6
2.1.2. Propiedades de datos	10
2.1.3. Propiedades de objeto	12
2.2. Vocabulario de CicloCarriles	20
2.2.1. Clases	22
2.2.2. Propiedades de datos	24
2.2.3. Propiedades de objeto	25
2.3. Vocabulario de Calles Tranquilas	27
2.3.1. Clases	29
2.3.2. Propiedades de datos	32
2.3.3. Propiedades de objeto	33
2.4. Transformaciones en los vocabularios: Fase 1	37
2.5. Transformaciones en los vocabularios: Fase 2	50
2.5.1. Cambios relativos a CicloCarriles	50
2.5.2. Cambios relativos a Accidentes de Bicicletas	52
2.5.3. Cambios relativos a Calles Tranquilas	54
3. Resultados y conclusiones	57
Bibliografía	60
Anexo	61

Capítulo 1

Introducción

Capítulo 2

Desarrollo

2.1. Vocabulario de Accidentes de Bicicletas

Para el vocabulario asociado con los accidentes de bicicletas en el ayuntamiento de Madrid se ha elegido la siguiente fuente Datos.Madrid [3]. En ella se muestran los accidentes de tráfico con implicación de bicicletas dentro de la jurisdicción del ayuntamiento.

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

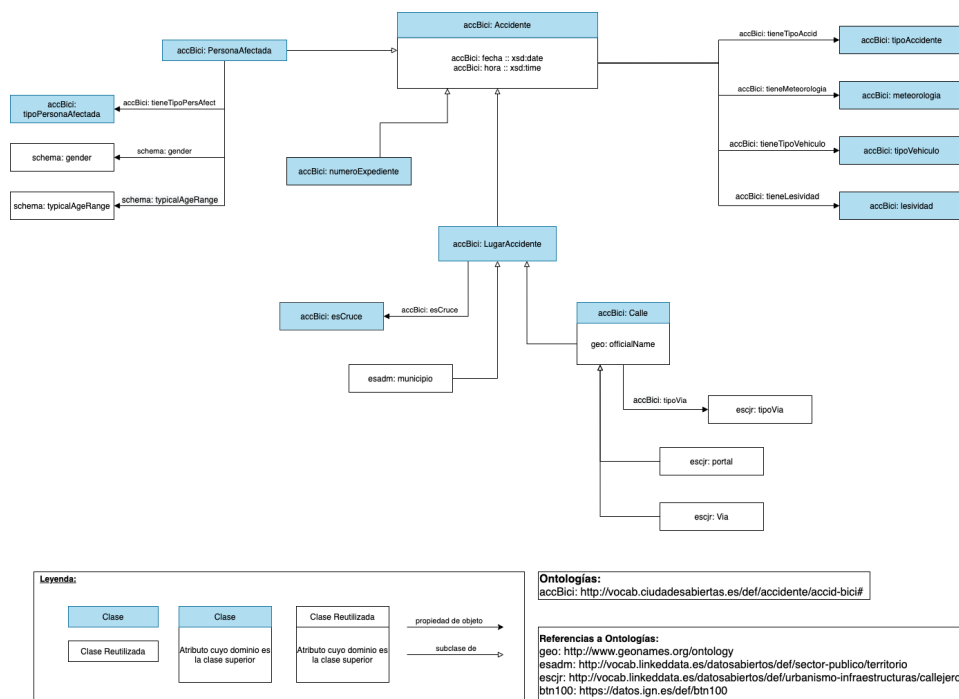


Figura 2.1: Diagrama de Ontología de Accidentes con implicación de bicicletas en Madrid.

Para la representación de los datos de accidentes de tráfico con bicicletas se han definido varias clases y propiedades. Se han reutilizado elementos ya definidos en el vocabulario de Callejero [1] 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 de Bicicletas

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

Para este conjunto de datos se ha optado por añadir, además de los ya proporcionados por la fuente de origen del ayuntamiento, nueva información como la propiedad `.esCruce`, el municipio, el tipo de 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. `EsCruce` se obtendrá del nombre de la calle, del cual atendiendo a varios patrones se puede determinar si el accidente ha ocurrido en una intersección de dos o más vías. El Municipio se ha añadido para su posible reutilización posterior utilizando otros datasets de otras localidades, en este caso será siempre "Madrid". El Tipo de Via de nuevo se obtendrá, a partir de los nombres de las calles, siguiendo unos patrones y palabras clave que determinen si es Calle, Avenida, Plaza... lo cual facilitará además su posterior cruce con otros datasets que también utilicen esta propiedad. 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.

En este conjunto de datos se ha hecho un cambio importante 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 via y se podrá hacer una búsqueda más sencilla de ellas. Para no perder ninguna información el nombre del lugar donde ha ocurrido el accidente se conservará igual (es decir, con la descripción del cruce de vias) y 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/accidente/accid-bici#Accidente>

Siniestro ocurrido con implicación de bicicletas.

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

En dominio de:

PersonaAfectada, LugarAccidente, meteorología,
lesividad, tipoVehiculo, tipoAccidente,
numeroExpediente

Tiene Superclases:

fecha, time

PersonaAfectada

IRI:

<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#PersonaAfectada>

La persona perjudicada por el accidente de tráfico.

Definida por:

<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Subclase de:

Accidente

En dominio de:

tipoPersonaAfectada, gender, typicalAgeRange

LugarAccidente

IRI:

<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#LugarAccidente>

El momento en el que ocurrió el siniestro.

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Subclase de:

Accidente

En dominio de:

esCruce, municipio, Calle

Calle

IRI: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#Calle>

Representación de una vía de una ciudad.

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Tiene subclase:

tipoVía, Vía, portal

Tiene Superclases:

LugarAccidente nombre oficial

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>

Tiene Superclases: LugarAccidente

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>

Tiene Superclases:
[LugarAccidente](#)

Vía

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, pantalanés, 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>

Tiene Superclases:
[Calle](#)

numeroExpediente

IRI: `http:`

`//vocab.ciudadesabiertas.es/def/accidente/accid-bici#numeroExpediente`

El identificador del siniestro que proporciona el ayuntamiento.

Definida por:

`http://vocab.ciudadesabiertas.es/def/accidente/accid-bici`

Tiene Superclases:

Accidente

2.1.2. Propiedades de datos

fecha

IRI: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#fecha>

Fecha en la que ocurre el siniestro.

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio:

Accidente

Rango:

date

hora

IRI: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#hora>

Hora en la que ocurre el siniestro.

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio:

Accidente

Rango:

time

nombre oficial

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: Calle

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

2.1.3. Propiedades de objeto

tienePersonaAfectada

IRI: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tienePersonaAfectada>

Persona que se asocia a un accidente.

Definida por:
<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio:
Accidente

Rango:
PersonaAfectada

tieneLugarAccidente

IRI: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneLugarAccidente>

Lugar de los acontecimientos asociados a un accidente.

Definida por:
<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio:
Accidente

Rango:
LugarAccidente

tieneCalle

IRI: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneCalle>

Calle asociada al lugar donde ocurrió el accidente.

Definida por:
<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio:
LugarAccidente

Rango:
Calle

tieneTipoVehiculo

IRI: http:

[//vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneTipoVehiculo](http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneTipoVehiculo)

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

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoVehiculo/BICICLETA>

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoVehiculo/BICICLETA-EPAC>

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio:

Accidente

Rango:

concept

tieneMeteorologia

IRI: http:

[//vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneMeteorologia](http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneMeteorologia)

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

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/meteorologia/DEPEJADO>

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/meteorologia/LLUVIA-DEBIL>

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/meteorologia/LLUVIA-INTENSA>

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/meteorologia/NUBLADO>

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/meteorologia/GRANIZANDO>

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/meteorologia/DESCONOCIDO>

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio:

Accidente

Rango:

concept

tieneNumExpediente

IRI: http:

`//vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneNumExpediente`

El identificador del siniestro que proporciona el ayuntamiento.

Definida por:

`http://vocab.ciudadesabiertas.es/def/accidente/accid-bici`

Dominio:

Accidente

Rango:

numeroExpediente

tieneTipoAcc

IRI:

<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneTipoAcc>

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

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/COLISION)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/COLISION](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/COLISION)

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/COLISION-DOBLE>

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/COLISION-MULTIPLE>

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/ALCANCE)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/ALCANCE](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/ALCANCE)

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/CHOQUE-NO-VEHICULO>

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/ATROPELLO-PEATON>

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/VUELCO)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/VUELCO](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/VUELCO)

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/CAIDA)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/CAIDA](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/CAIDA)

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/OTROS)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/OTROS](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/tipoAcc/OTROS)

Definida por:

<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio:

Accidente

Rango:

concept

tieneLesividad

IRI:

<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneLesividad>

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

Para su uso se han definido los siguientes elementos:

01 Atención en urgencias sin posterior ingreso. - LEVE:

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/01)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/01](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/01)

02 Ingreso inferior o igual a 24 horas - LEVE:

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/02)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/02](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/02)

03 Ingreso superior a 24 horas. - GRAVE:

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/03)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/03](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/03)

04 Fallecido 24 horas - FALLECIDO:

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/04)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/04](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/04)

05 Asistencia sanitaria ambulatoria con posterioridad - LEVE:

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/05)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/05](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/05)

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

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/06)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/06](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/06)

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

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/07)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/07](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/07)

14 Sin asistencia sanitaria:

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/14)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/14](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/14)

77 Se desconoce:

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/77)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/77](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/lesividad/77)

Definida por:

<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

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:

genderType [16]

tieneTipoPersAfect

IRI: [http:](http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneTipoPersAfect)

[//vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneTipoPersAfect](http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneTipoPersAfect)

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

<http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/persAfec/CONDUCTOR>

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/persAfec/PEATON)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/persAfec/PEATON](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/persAfec/PEATON)

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/persAfec/TESTIGO)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/persAfec/TESTIGO](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/persAfec/TESTIGO)

[http:](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/persAfec/VIAJERO)

[//vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/persAfec/VIAJERO](http://vocab.ciudadesabiertas.es/recurso/accidente/accid-bici/persAfec/VIAJERO)

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio:

PersonaAfectada

Rango:

concept

tipoVia

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

Se ha reutilizado la definición de tipoVia proporcionada por vocab.linkeddata.es [12]. Tipo de vía, que será representado mediante la clasificación en SKOS de URI <http://vocab.linkeddata.es/datosabiertos/kos/urbanismo-infraestructuras/tipo-via>. Por ejemplo, estas serán las URIs correspondientes a calles y plazas <http://vocab.linkeddata.es/datosabiertos/kos/urbanismo-infraestructuras/tipo-via/CL> <http://vocab.linkeddata.es/datosabiertos/kos/urbanismo-infraestructuras/tipo-via/PL>

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

Dominio: Calle

Rango: concept

tieneMunicipio

IRI: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneMunicipio>

Municipio en el que ocurrió un accidente.

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio: LugarAccidente

Rango: municipio

tieneIdVia

IRI: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tieneIdVia>

Identificador de calle asociado.

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio: Calle

Rango: Via

tienePortal

IRI:

<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#tienePortal>

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

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio: Calle

Rango: Portal

esCruce

IRI: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici#esCruce>

Si el accidente ocurrió en un cruce entre 2 o más vías. Se han definido los siguientes elementos:

<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici/esCruce/TRUE>

<http://vocab.ciudadesabiertas.es/def/accidente/accid-bici/esCruce/FALSE>

Definida por: <http://vocab.ciudadesabiertas.es/def/accidente/accid-bici>

Dominio: LugarAccidente

Rango: concept

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 2.3

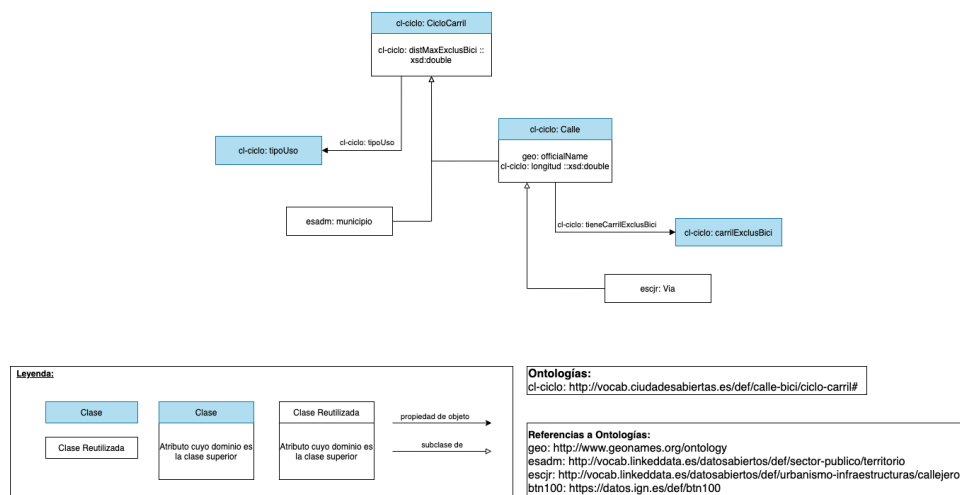


Figura 2.3: Diagrama de Ontología de ciclocarriles en Madrid.

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.es [1]. 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, presente en los demás utilizados. En este caso los nombres de las calles contenían únicamente palabras clave y privaban de la capacidad de obtener este atributo. En cambio si dispone de tipoUso, propiedad que indica si es una calle peatonal o ciclocarril.

Debido a la falta de disponibilidad de una leyenda o información proporcionada por el ayuntamiento de Madrid, para este conjunto de datos no se han podido conocer con exactitud el significado de sus datos. Se ha obtenido de forma aproximada esa información pero podría no ser correcta.

2.2.1. Clases

CicloCarril

IRI:

<http://vocab.ciudadesabiertas.es/def/calle-bici/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.ciudadesabiertas.es/def/calle-bici/ciclo-carril>

Tiene subclase:

Calle, municipio, tipoUso

Tiene Superclase:

distMaxExclusBici

Calle

IRI: <http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#Calle>

Representación de una via de una ciudad.

Definida por:

<http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril>

Tiene subclase:

carrilExclusBici, Via

Tiene Superclases:

CicloCarril, longitud, nombre oficial

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>

Tiene Superclases:
Calle

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>

Tiene Superclases:
CicloCarril

2.2.2. Propiedades de datos

longitud

IRI:

<http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#longitud>

Longitud de la calle descrita. Esta propiedad está referida a la calle que contiene un ciclocarril (calle completa). La longitud del ciclocarril estará representada por la propiedad `distMaxExclusBici`.

Definida por:

<http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril>

Dominio: Calle

Rango: double

distMaxExclusBici

IRI: [http:](http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#distMaxExclusBici)

[//vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#distMaxExclusBici](http://vocab.ciudadesabiertas.es/def/calle-bici/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.ciudadesabiertas.es/def/calle-bici/ciclo-carril>

Dominio: CicloCarril

Rango: double

nombre oficial

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: Calle

2.2.3. Propiedades de objeto

tieneCalle

IRI:

<http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#tieneCalle>

Calle asociada a un CicloCarril.

Definida por:

<http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril>

Dominio: CicloCarril

Rango: Calle

tipoUso

IRI: <http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#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>

Dominio:

CicloCarril

Rango: concept

tieneIdVia

IRI:

<http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#tieneIdVia>

Identificador de calle asociado.

Definida por:

<http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril>

Dominio: Calle

Rango: Via

tieneMunicipio

IRI: [http:](http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#tieneMunicipio)

[//vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#tieneMunicipio](http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#tieneMunicipio)

Municipio que contiene una calle.

Definida por:

<http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril>

Dominio: CicloCarril

Rango: municipio

tieneCarrilExclusBici

IRI: [http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#](http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#tieneCarrilExclusBici)
[tieneCarrilExclusBici](http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril#tieneCarrilExclusBici)

Indicador de si las bicicletas disponen o no de un carril propio para su circulación. Se han definido 2 clases para ello:

- <http://vocab.ciudadesabiertas.es/recurso/calle-bici/ciclo-carril/carrilExlus/EXCLUSIVO-BICI>
- <http://vocab.ciudadesabiertas.es/recurso/calle-bici/ciclo-carril/carrilExlus/NO-EXCLUSIVO-BICI>

Definida por:

<http://vocab.ciudadesabiertas.es/def/calle-bici/ciclo-carril>

Dominio: CicloCarril

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 utilizados que han llevado a estas calles a formar parte de la lista. Sin embargo, si se puede observar en algunas de ellas ciertos patrones, como que no forman parte de las vías principales de la capital y que son poco transitadas. La misma web ofrece un archivo KML y permite que se puedan mostrar sobre un mapa en Open Street Map [8], lo cual proporciona una idea general de su disposición y posibles criterios utilizados.

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

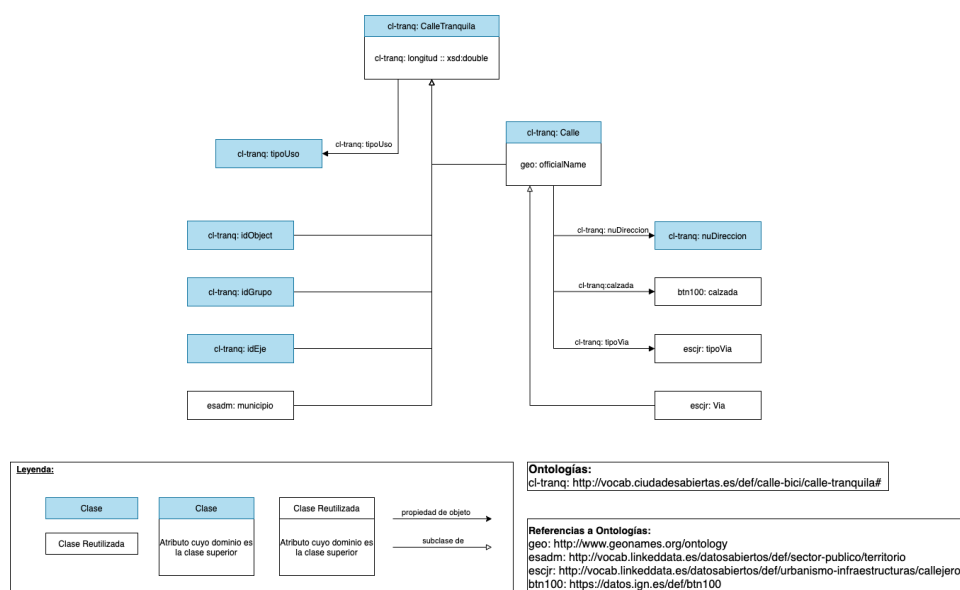


Figura 2.5: Diagrama de Ontología de Calles Tranquilas en Madrid.

Para la representación de los datos de calles tranquilas para ciclistas se han definido varias clases y propiedades. Se han tomado como ejemplo el vocabulario definido en ciudades abiertas de Callejero [1] y el definido en datos.ign sobre calzadas [9].

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 vía del nombre, conservándola en éste y creando una nueva propiedad que permita saber su clase. Algunos ejemplos serían Calle, Avenida, Plaza... Se ha añadido el identificador de la vía, obtenido a partir del nombre y cruzado con el dataset del callejero de Madrid [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 reutilizar en otros proyectos a mayor escala sería necesario conocer la zona geográfica donde se encuentra, por tanto también se ha añadido, aunque con el valor fijo de Madrid, que corresponde al código 28079, proporcionado por el Instituto Nacional de Estadística [10].

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

Se ha optado por omitir la propiedad ID_TIPO, ya que representa lo mismo que TX_CAPA

(el uso que tiene la via) y se ha optado por la segunda por ser representado con texto, más visual y representativo a la hora de su utilización.

El tipo de uso se ha definido en <http://vocab.ciudadesabiertas.es/def/calle-bici#tipoUso> debido a que es común para todas las calle-bici (tanto calles tranquilas como ciclocarriles), por lo tanto se ha optado porque los valores sean definidos en una categoría que englobe a las dos. La dirección se ha definido también en esta categoría común ya que podría ser añadida a ciclocarriles por el ayuntamiento o ser común con otros dataset de una categoría similar(<http://vocab.ciudadesabiertas.es/def/calle-bici#nuDireccion>).

En la siguiente tabla se muestran los Namespaces usados.

	http://vocab.ciudadesabiertas.es/def/calles-tranquilas#
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 2.6: Namespaces usados para Calles Tranquilas

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. Se ha obtenido de forma aproximada esa información pero podría no ser correcta.

2.3.1. Clases

CalleTranquila

IRI: `http:`

`//vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#CalleTranquila`

Calle apropiada para el tránsito de ciclistas siguiendo los criterios del ayuntamiento de Madrid.

Definida por:

`http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila`

Tiene subclase:

`idObject, idGrupo, idEje, tipoUso,`
`municipio, Calle`

Tiene Superclase:

`longitud`

Calle

IRI:

`http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#Calle`

Representación de una vía de una ciudad.

Definida por:

`http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila`

Tiene subclase:

`longitud, nuDireccion,`
`calzada, tipoVia`

Tiene Superclases:

`CalleTranquila`
`nombre oficial`

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>

Tiene Superclases:
Calle

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>

Tiene Superclases:
CalleTranquila

idObject

IRI:

<http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#idObject>

Identificador propio del ayuntamiento.

Definida por:

<http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila>

Tiene Superclases:

CalleTranquila

idGrupo

IRI:

<http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#idGrupo>

Identificador propio del ayuntamiento.

Definida por:

<http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila>

Tiene Superclases:

CalleTranquila

idEje

IRI:

<http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#idEje>

Identificador propio del ayuntamiento.

Definida por:

<http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila>

Tiene Superclases:

CalleTranquila

2.3.2. Propiedades de datos

nombre oficial

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: Calle

longitud

IRI:
<http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#longitud>

Longitud de la calle o tramo de la calle descrito. Su medida es Shape_leng. Es propiedad de CalleTranquila y no de calle porque cabe la posibilidad de que no coincidan ambas medidas debido a tramos de vías más inseguros. A falta de documentación sobre el dataset se ha supuesto esta hipótesis.

Definida por:
<http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila>

Dominio: CalleTranquila

Rango: double

2.3.3. Propiedades de objeto

tieneCalle

IRI: http:

`//vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#tieneCalle`

Calle asociada a una Calle Tranquila.

Definida por:

`http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila`

Dominio: CalleTranquila

Rango: Calle

tipoUso

IRI:

`http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#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`

Dominio:

CalleTranquila

Rango: concept

tieneIdEje

IRI: http:

`//vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#tieneIdEje`

Identificador de Eje asociado a una calle.

Definida por:

`http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila`

Dominio: CalleTranquila

Rango: idEje

tieneIdGrupo

IRI: http:

`//vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#tieneIdGrupo`

Identificador de Grupo asociado a una calle.

Definida por:

`http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila`

Dominio: CalleTranquila

Rango: idGrupo

tieneIdObject

IRI: http:

`//vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#tieneIdObject`

Identificador de Objeto asociado a una calle.

Definida por:

`http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila`

Dominio: CalleTranquila

Rango: idObject

tieneIdVia

IRI: http:

`//vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#tieneIdVia`

Identificador de calle asociado.

Definida por:

`http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila`

Dominio: Calle

Rango: Via

tieneMunicipio

IRI: [http:](http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#tieneMunicipio)

[//vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#tieneMunicipio](http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila#tieneMunicipio)

Municipio que contiene una calle.

Definida por:

<http://vocab.ciudadesabiertas.es/def/calle-bici/calle-tranquila>

Dominio: CalleTranquila

Rango: municipio

nuDireccion

IRI: <http://vocab.ciudadesabiertas.es/def/calle-bici#nuDireccion>

Dirección y sentido de la calle o tramo de la calle a la que se refiere. Puede tomar los siguientes valores:

<http://vocab.ciudadesabiertas.es/recurso/calle-bici#SOL> - Dirección y sentido a Sol

<http://vocab.ciudadesabiertas.es/recurso/calle-bici#paraleloSOL> - Dirección paralela a Sol

<http://vocab.ciudadesabiertas.es/recurso/calle-bici#contrarioSOL> - Dirección a sol y sentido opuesto.

Esta definición esta basada en datos aproximados proporcionados por mapas y otros elementos que han podido verse en los ejemplos del dataset. Pueden ser erróneos o no reflejar con exactitud la realidad del dato ya que no existe una leyenda que poder consultar.

Definida por: <http://vocab.ciudadesabiertas.es/def/calle-bici>

Dominio: Calle

Rango:
concept

calzada

IRI: <https://datos.ign.es/def/btn100#calzada>

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

<https://datos.ign.es/kos/transportes/calzada/convencional>

<https://datos.ign.es/kos/transportes/calzada/doble>

<https://datos.ign.es/kos/transportes/calzada/sentido-unico>

Definida por: <https://datos.ign.es/kos/transportes/calzada/>

Dominio: Calle

Rango:
concept

tipoVia

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

Se ha reutilizado la definición de tipoVia proporcionada por vocab.linkeddata.es [12].

Tipo de vía, que será representado mediante la clasificación en SKOS de URI

<http://vocab.linkeddata.es/datosabiertos/kos/urbanismo-infraestructuras/tipo-via>. Por ejemplo, estas serán las URIs correspondientes a calles y plazas

<http://vocab.linkeddata.es/datosabiertos/kos/urbanismo-infraestructuras/tipo-via/CL>

<http://vocab.linkeddata.es/datosabiertos/kos/urbanismo-infraestructuras/tipo-via/PL>

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

Dominio: Calle

Rango: concept

2.4. 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

```
1 import csv
2 import os
3 from re import findall
4
5 def cambiosBasicos(nombreCarpeta="", nombreSinCsv="", nRowNombre=-1, nFileFin = "-1"):
6     with open(nombreCarpeta + "/" + nombreSinCsv + ".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                primeraLinea = False
14            else:
15                nombreVia = row[nRowNombre].upper()
16                if (nombreVia.__contains__("C/")):
17                    nombreVia = nombreVia.replace("C/", "C/ ")
18                if (findall('CRUCE\s.*\sCON\s', nombreVia) != []): # Escritura estandar de "Cruce"
19                    nombreVia = nombreVia.replace(" CON ", " / ")
20                    nombreVia = nombreVia.replace("CRUCE", " ")
21                if (nombreVia.__contains__("S/N")): # P.ej. SAN FRANCISCO S/N, casos que generan error
22                    nombreVia = nombreVia.replace('S/N', " ")
23                if (findall('[A-z, C!, Ã, 0-9]{1}[/][A-z]{1}', nombreVia) != []): # Escritura estandar de
24                    "Cruce"
25                    nombreVia = nombreVia.replace("/", " / ")
26                # C/ MONASTERIO DE ARLANZA-AV. SANTUARIO DE VALVERDE
27                if (findall('(!\d)[\{-1\}](?!d)', nombreVia) != []): # Evitar palabras con numeros como M-30
28                    arr1 = nombreVia.replace("-", " - ").split()
29                    if (getTipoVia(palabrasMalEscritas(arr1[arr1.index("-")+1])) != ""): # Si es palabra clave
30                        indicadora de nueva via, es Cruce
31                        nombreVia = nombreVia.replace("-", " / ")
32                if (findall('(!\d)[\{-1\}](\d)', nombreVia) != []): # Palabras con numero como M-30
33                    nombreVia = nombreVia.replace("-", " ")
34
35                # ;19:10;"CAMINO EN ZONA DENOMINADA COMO "ESPACIO MÉXICO""";
36                if (nombreVia.__contains__('\"')): #Quitar comillas
37                    nombreVia = nombreVia.replace('"', ' ')
38                if (nombreVia.__contains__("PASO ELEVADO")):
39                    nombreVia = nombreVia.replace('PASO ELEVADO', 'PASO_ELEVADO')
40                elif (nombreVia.__contains__("SENDA CICLABLE")):
41                    nombreVia = nombreVia.replace('SENDA CICLABLE', 'SENDA_CICLABLE')
42
43                #Cambios varios en nueva funcion
44                nombreVia = palabrasMalEscritas(nombreVia)
45
46                row[nRowNombre] = nombreVia
47                fila = ";".join(row)
48                file.write(fila + os.linesep)
49            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 `calle1 / calle2`". Para conseguir ese formato han de modificarse otros nombres como por ejemplo `CRUCE calle1 CON calle1`"para que posteriormente las funciones sean aplicables a estos casos.

También se eliminan elementos como "S/N"(Sur/Norte) que no tienen excesiva relevancia, no forman parte del nombre, pero en cambio si pueden llegar a producir errores graves. 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 así 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

```

1 import csv
2 import os
3 from re import findall
4
5 def palabrasMalEscritas(nombreCalle = ""): #Arreglar errores de codificación y abreviaturas
6     if(nombreCalle == ""):
7         return ""
8     nuevaPalabra = ""
9     for palabra in nombreCalle.split():
10        palabra = palabra.replace(" ", "").upper()
11        if(palabra == "PNTE."):
12            palabra = "PUENTE"
13        elif (palabra == "CÑADA." or palabra == "CÑADA"):
14            palabra = "CAÑADA"
15        elif (palabra == "AVDA." or palabra == "AVDA" or palabra == "AV." or palabra == "AV"):
16            palabra = "AVENIDA"
17        elif (palabra == "JARDÍN" or palabra == "JARDINES"):
18            palabra = "JARDIN"
19        elif (palabra == "CUSTA." or palabra == "CUSTA"):
20            palabra = "CUESTA"
21        elif (palabra == "POLÍGONO"):
22            palabra = "POLIGONO"
23        elif (palabra == "GALERÍA"):
24            palabra = "GALERIA"
25        elif (palabra == "PLAZA."):
26            palabra = "PLAZA"
27        elif (palabra == "PISTA."):
28            palabra = "PISTA"
29        elif (palabra == "CMNO." or palabra == "CMNO"):
30            palabra = "CAMINO"
31        elif (palabra == "RONDA."):
32            palabra = "RONDA"
33        elif (palabra == "GTA." or palabra == "GTA"):
34            palabra = "GLORIETA"
35        elif (palabra == "CUSTA." or palabra == "CUSTA"):
36            palabra = "CUESTA"
37        elif (palabra == "PQUE." or palabra == "PQUE"):
38            palabra = "PARQUE"
39        elif (palabra == "CTRA." or palabra == "CTRA"):
40            palabra = "CARRETERA"
41        elif (palabra == "AUTOV." or palabra == "AUTOV"):
42            palabra = "AUTOVIA"
43        elif (palabra == "CRUCE."):
44            palabra = "CRUCE"
45        elif (palabra == "ANILLO."):
46            palabra = "ANILLO"
47        elif (palabra == "PASEO."):
48            palabra = "PASEO"
49        elif (palabra == "TRVA."):
50            palabra = "TRAVESIA"
51        elif (palabra == "ESTFE."):
52            palabra = "ESTACION_FERROCARRIL"
53        elif (palabra == "P?"):
54            palabra = "PLAZA"
55        elif (palabra == "PZA."):
56            palabra = "PLAZA"
57        elif (palabra == "CÁZ" or palabra == "CALL." or palabra == "CALL" or palabra == "C/" or palabra ==
58            "C?"):
59            palabra = "CALLE"
60        elif (palabra == "GRAL"):
61            palabra = "GENERAL"
62        elif (palabra == "STA." or palabra == "STA"):
63            palabra = "SANTA"
64
65        # Mayoritariamente errores de codificación:
66        elif (palabra.__contains__("Ÿ")):
67            palabra = palabra.replace("Ÿ", "í")
68        elif (palabra.__contains__("Ãq")):

```

```

68     palabra = palabra.replace("Ãa", "Ñ")
69     elif (findall('([B-DF-HJ-NP-TV-Z]{1})SS', palabra) != [] or findall('SS([B-DF-HJ-NP-TV-Z]{1})',
70         palabra) != []):
71         #P.ej. BSSRBARA ó NARVSSEZ ó GUZMSSN Ó "Ortega Y GASSET" Ó ALCALSS
72         palabra = palabra.replace("SS", "Ã")
73     elif (findall('([A-Z]{1})Ã`N', palabra) != []): #P.ej. SAHAGÃ`N
74         palabra = palabra.replace("Ã`N", "ÜN")
75     elif (palabra[0] == '?'): #P.ej. ?LVAREZ --> ? ASCII 63
76         palabra = palabra.replace("?", "Ã")
77     elif (palabra.__contains__(chr(190))): #P.ej. MOSCARDÃž --> ASCII 190, PERÃžN
78         palabra = palabra.replace(chr(190), "0")
79     elif (palabra.__contains__(chr(179))): #P.ej. MOSCARDÃž --> ASCII 179, YAGÃžE
80         palabra = palabra.replace(chr(179), "Ü")
81     elif (findall('QUÜ', palabra) != []): # P.ej MarquÜs --> Marqués
82         palabra = palabra.replace("QUÜ", "QUÉ")
83     elif (palabra == "JESÃ`S"):
84         palabra = "JESÜS"
85     elif (findall('KM-(\d)', palabra) != []): # Palabras con numero como KMO
86         palabra = ""
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 mayoría 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 (También eliminado Azcona ya que no esta previsto la existencia de cruces)
- ln 117: Gral Yag>e →Gral Yagüe
- ln 125: MARQUÉS DE VIANA - SOR ANGELA DE LA CRUZ →Divido en 2 registros con características similares.

■ Cambios realizados de forma manual al dataset de CallesTranquilas:

ln 1292-1292: Calle de la Cooperativa ElÚctrica →Calle de la Cooperativa Eléctrica

ln 1822-1823: Doctor MartYn ArÚvalo →Doctor Martín Arévalo

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

- ln 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.j4-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.

2.4. Transformaciones en los vocabularios: Fase 1

331600;CALLE;DE;GENERAL MOSCARDÓ;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 datasets <https://www.elmundo.es/madrid/2017/05/31/592dbf00e2704ed5058b4688.html>.
765800;RONDA;DE;RONDA VALENCIA;RONDA VALENCIA;1;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

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

En el caso de que el accidente haya ocurrido en la intersección entre varias vías, 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

```
1 import csv
2 import os
3 from re import findall
4
5 def annadirTipoVia(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 + ";" + "TIPO_VIA"
14            primeraLinea = False
15        else:
16            tipoVia = getTipoVia(row[nRowNombre])
17            fila = ";".join(row)
18            fila = fila + ";" + tipoVia
19        file.write(fila + os.linesep)
20    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`

```

1 import csv
2 import os
3 from re import findall
4
5 def getTipoVia(nombreViaIni = ""):
6     if(nombreViaIni == ""):
7         return ""
8     arr = nombreViaIni.upper().split()
9     # Para solo tener las primeras palabras y no por ejemplo Gran VIA
10    nombreVia = arr[0]
11    if (nombreVia.__contains__("CALLE")):
12        return "Calle"
13    elif (nombreVia.__contains__("PASEO")):
14        return "Paseo"
15    elif (nombreVia.__contains__("PLAZA")):
16        return "Plaza"
17    elif (nombreVia.__contains__("GLORIETA")):
18        return "Glorieta"
19    elif (nombreVia.__contains__("RONDA")):
20        return "Ronda"
21    elif (nombreVia.__contains__("CAMINO")):
22        return "Camino"
23    elif (nombreVia.__contains__("PISTA")):
24        return "Pista"
25    elif (nombreVia.__contains__("ANILLO")):
26        return "Anillo"
27    elif (nombreVia.__contains__("CRUCE")):
28        return "Cruce"
29    elif (nombreVia.__contains__("AUTOVIA")):
30        return "Autovia"
31    elif (nombreVia.__contains__("CARRETERA")):
32        return "Carretera"
33    elif (nombreVia.__contains__("PARQUE")):
34        return "Parque"
35    elif (nombreVia.__contains__("CUESTA")):
36        return "Cuesta"
37    elif (nombreVia.__contains__("CAÑADA")):
38        return "Cañada"
39    elif (nombreVia.__contains__("AVENIDA")):
40        return "Avenida"
41    elif (nombreVia.__contains__("BULEVAR")):
42        return "Bulevar"
43    elif (nombreVia.__contains__("JARDIN")):
44        return "Jardin"
45    elif (nombreVia.__contains__("PARTICULAR")):
46        return "Particular"
47    elif (nombreVia.__contains__("POLIGONO")):
48        return "Poligono"
49    elif (nombreVia.__contains__("GALERIA")):
50        return "Galeria"
51    elif (nombreVia.__contains__("ESCALINATA")):
52        return "Escalinata"
53    elif (nombreVia.__contains__("VIA")):
54        return "Via"
55    elif (nombreVia.__contains__("PASARELA")):
56        return "Pasarela"
57    elif (nombreVia.__contains__("PASAJE")):
58        return "Pasaje"
59    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 de 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 vía para así comparar de forma más satisfactoria con los registros del Callejero de la ciudad.

Listing 2.6: Función añadirPalabrasClave

```

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

```

En el código de Listing 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 vía, elementos ya añadidos a este nuevo campo y que pueden estar en un dataset y no en otro por su carácter genérico).

Listing 2.7: Función quitarConectores

```

1 import csv
2 import os
3 from re import findall
4
5 def quitarConectores(nombreVia = ""):
6     nombreVia = nombreVia.upper()
7     listaPosibles = [' DEL ', ' DE ', ' Y ', ' LAS ', ' LA ', ' LOS ', ' A ', ' POR ', ' CON ',
8                     ' EL ', ' EN ', ' O ', ' I ', ' AL ', ' ', ' - ', ' S/N ', ' JUNTO ', ' _ ']
9     for a in listaPosibles:
10         if(nombreVia.__contains__(a)):
11             nombreVia = nombreVia.replace(a, ' ')
12     return nombreVia

```

Listing 2.8: Función quitarPalabrasConflicto

```

1 import csv
2 import os
3 from re import findall
4
5
6 def quitarPalabrasConflicto(nombreVia = ""):
7     nombreVia = nombreVia.upper()
8     if(nombreVia == ""):
9         return ""
10    listaPalabrasQuitar = ["CRUCE. ", "CRUCE ", "CALL. ", "CALL ", "CALLE ", "C/", "C/ ", "PASEO. ", "PASEO ",
11                          "PLAZA.", "PLAZA ", "GTA. ",
12                          "GTA ", "GLORIETA ", "RONDA. ", "RONDA ",
13                          "CMNO. ", "CMNO ", "CAMINO ", "PISTA. ", "PISTA ", "AUTOVIA ", "CTRA. ",
14                          "CTRA ", "CARRETERA ", "PQUE. ", "PQUE ", "PARQUE ", "CUSTA. ", "CUSTA ", "CUESTA ", "CÑADA. ", "CÑADA ",
15                          "CAÑADA ", "AVDA. ",
16                          "AVDA ", "AV. ", "AV ", "AVENIDA ", "VIA ", "PASARELA ", "PASAJE ", "PUENTE ", "COSTANILLA ", "COLONIA ",
17                          "CARRERA ",
18                          "PLAZUELA ", "BULEVAR ", "BULEV. ", "ESCALINATA ", "JARDÍN ", "JARDIN ", "JARDINES ", "PARTICULAR ",
19                          "POLÍGONO ", "POLIGONO ", "ACCESO ",
20                          "POBLADO ", "PASADIZO ", "TRASERA ", "SENDA ", "GALERÍA ", "GALERIA ", "ARROYO ", "VALLE ", "AEROPUERTO ",
21                          "PASO_ELEVADO", "SENDA_CICLABLE"]
22
23    for a in listaPalabrasQuitar:
24        if(nombreVia.__contains__(a) and findall('[A-z]{1}' + a, nombreVia) == []): #Para evitar errores por
25            ejemplo en Segovia
26            arr = nombreVia.split()
27            if (arr.index(a.replace(' ', '')) == 0): # Para evitar suprimir Gran VIA p.ej.
28                nombreVia = nombreVia.replace(a, ' ')
29            if (nombreVia.__contains__('/') and findall('[A-z]{1}' + '/', nombreVia) == []):
30                if (arr.index(a.replace(' ', '')) == arr.index('/') + 1): # Para cuando es un cruce
31                    nombreVia = nombreVia.replace(a, ' ')
32
33    # Podria ser M11 por ejemplo
34    if(nombreVia.strip()[0] == '/'):
35        # Para eliminar descripciones al inicio y su / p.ej. SENDA_CICLABLE / AV.ROSALES / CARRET.VILLAV. A
36        VALLECA
37        nombreVia = nombreVia.replace('/', ' ', 1)
38    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 vías 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

```

1 import csv
2 import os
3 import difflib
4 import unicodedata
5
6 def crearFichNombresId(nombreCarpeta = "", nombreSinCsv = "", nFileIni = "-1", nFileFin = "_IDs_000",
7                       nRowEsCruce = -1, nRowPalabrasClave = "-1", nRowTipoVia="-1", tieneTipoCalle = False):
8     with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
9         csvreader = csv.reader(csvfile, delimiter=";")
10        file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
11        primeralinea = True
12        nCorrect = 0
13        nIncorrect = 0
14        contadorFila = 0
15        for row in csvreader:
16            if (primeralinea):
17                fila = ";".join(row)
18                fila = fila + ";" + "ID_CALLE"
19                primeralinea = False
20            else:
21                arrCalles = []
22                if (nRowEsCruce != -1 and row[nRowEsCruce] == "1"): # Si es cruce tendra varios ids
23                    arrCallesCruce = getArrCalles(row[nRowPalabrasClave].upper())
24                else:
25                    arrCallesCruce = [row[nRowPalabrasClave].upper()]
26
27                listIds = []
28                nElem = 0
29                for nombre in arrCallesCruce:
30                    #TODO: quitar
31                    '''if(not(nombre.__contains__("EUGENIA"))):
32                        continue'''
33                    #TODO: quitar
34                    segundaVuelta = False
35                    nVuelta = 0
36                    while (nVuelta == 0 or segundaVuelta):
37                        nVuelta = nVuelta + 1
38                        with open("callejeroCSV" + "/" + "nombres_IDS" + ".csv") as csvCallejero:
39                            csvreaderCallej = csv.reader(csvCallejero, delimiter=";")
40                            primeralinea = True
41                            encontrado = False
42                            for rowCallj in csvreaderCallej:
43                                if (primeralinea):
44                                    primeralinea = False
45                                else:
46                                    try:
47                                        # TODO: quitar
48                                        '''if (rowCallj[11].__contains__("EUGENIA")):
49                                            print("OK2")'''
50                                        # TODO: quitar
51                                        if((nElem > 0 or not(tieneTipoCalle) or
52                                           (segundaVuelta and esTipoCalleOmitible(row[nRowTipoVia].upper()))
53                                           or chequearPalabras(rowCallj[1].upper(), row[nRowTipoVia].upper(), False))
54                                           and
55                                           #Porque si es la segunda entrada no contiene el tipo de via
56                                           chequearPalabras(rowCallj[11].upper(), nombre.upper(), segundaVuelta)):
57                                            listIds.append(rowCallj[0])
58                                            encontrado=True
59                                            nCorrect = nCorrect+1
60                                            segundaVuelta=False
61                                            break
62                                    except IndexError:
63                                        print("Error")
64                                        # Posible error de que haya una linea vacia extra al final
65                                        if (not(encontrado) and not(segundaVuelta)):
66                                            segundaVuelta = True
67                                        continue
68                                    if(not(encontrado) and segundaVuelta):
69                                        segundaVuelta = False
70                                        listIds.append("-1")
71                                        nIncorrect = nIncorrect + 1
72                            csvCallejero.close()
73                            nElem = nElem + 1
74                            fila = ";".join(row)
75                            fila = fila + ";" + listIds.__str__()
76                            contadorFila = contadorFila+1
77                            if(contadorFila % 20 == 0):
78                                print("Fila: ", contadorFila)
79                            file.write(fila + os.linesep)
80        file.close()
81        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 función `getArrCalles` (Listing 2.10) para obtener las dos o más vías 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 vía, 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 vía 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 vía 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`

```

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

```

En este código 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`

```

1 import csv
2 import os
3 import difflib
4 import unicodedata
5 def checkearPalabras(nombreCallej = "", nombreDataset = "", segundaVuelta = False):
6     # 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
39         por ahi
40     # -----
41     # Por ejemplo FERNADO VI --> FERNANDO VI
42     longCadena1 = nombreDataset.__len__()
43     longCadena2 = nombreCallej.__len__()
44     if(longCadena1 < (longCadena2*0.7) or longCadena1 > (longCadena2 * 1.3)) :
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")) #
70        CANDILEJAS :: CANALEJAS
71        and not (nombreCallej.__contains__("LIMON")) # JUAN VIGON :: JUAN LIMON
72        and not (nombreCallej.__contains__("CENICIENTOS")) # CENICIENTA :: CENICIENTOS
73        and not (nombreCallej.__contains__("CANTERAS")) # MANOTERAS :: CANTERAS
74        and not (nombreDataset.__contains__("MANOTERAS")) # MANOTERAS :: SANTERAS
75        and not (nombreCallej.__contains__("GOR")) # SIERRA GADOR :: SIERRA GOR
76        and not (nombreDataset.__contains__("GADOR")) # SIERRA GADOR :: SIERRA GUDAR
77        and not (nombreCallej.__contains__("ERASMO")) # SANZ RASO :: SAN ERASMO
78        and not (nombreCallej.__contains__("ALIO")) # SANTIAGO AMON :: SANTIAGO ALIO
79        and not (nombreDataset.__contains__("PARVILLAS")) # PARVILLAS :: MARAVILLA
80        and not (nombreDataset.__contains__("MARMOLINA")) # MARMOLINA :: CAROLINA
81        and not (nombreCallej.__contains__("SAMANIEGO")) # SAN DIEGO :: SAMANIEGO
82        and not (nombreDataset.replace(" ", "").__contains__("SANDIEGO")) # SAN DIEGO :: SAN DACIO
83        and not (nombreCallej.__contains__("BARROS")) # BARRIOS :: BARROS
84        and not (nombreDataset.replace(" ", "").__contains__("MADRIDRIO")) # MADRID RIO :: MADRE DIOS
85        ):
86         print("Posible Conicidencia Sustitucion: ", nombreDataset, " :: ", nombreCallej)
87         return True
88     # -----
89     # Dependiendo de la longitud de la palabra acepta 1 error o más
90     if (diferenciastxt.count('+') <= (longCadena1/10+1) and diferenciastxt.count('-') <=(longCadena1/10+1)
91        and not (nombreDataset.__contains__("HORTALEZA")) #Hortaleza - Fortaleza
92        and not (nombreCallej.__contains__("GALENA")) # Gilena - Galena
93        and not (nombreCallej.__contains__("PEAL")) # Real - Peal
94        and not (nombreCallej.__contains__("OLIVAR")) # Bolivar - Olivar
95        and not (nombreCallej.__contains__("CRUCES")) # RUICES :: CRUCES
96        and not (nombreCallej.__contains__("HAYA")) # RAYA :: HAYA
97        and not (nombreCallej.__contains__("OCA")) # OÑA :: OCA
98        and not (nombreCallej.__contains__("MANZANAR")) # MANZANARES :: MANZANAR
99        and not (nombreDataset.__contains__("CANDILEJAS") and nombreCallej.__contains__("CANALEJAS")) #

```

```

199         CANDILEJAS :: CANALEJAS
200         and not (nombreCallej.__contains__("LIMON")) # JUAN VIGON :: JUAN LIMON
201         and not (nombreCallej.__contains__("CENICIENTOS")) # CENICIENTA :: CENICIENTOS
202         and not (nombreCallej.__contains__("CANTERAS")) # MANOTERAS :: CANTERAS
203         and not (nombreDataset.__contains__("MANOTERAS")) # MANOTERAS :: SANTERAS
204         and not (nombreCallej.__contains__("GOR")) # SIERRA GADOR :: SIERRA GOR
205         and not (nombreDataset.__contains__("GADOR")) # SIERRA GADOR :: SIERRA GUDAR
206         and not (nombreCallej.__contains__("ERASMO")) # SANZ RASO :: SAN ERASMO
207         and not (nombreCallej.__contains__("ALIO")) # SANTIAGO AMON :: SANTIAGO ALIO
208         and not (nombreDataset.__contains__("PARVILLAS")) # PARVILLAS :: MARAVILLA
209         and not (nombreDataset.__contains__("MARMOLINA")) # MARMOLINA :: CAROLINA
210         and not (nombreCallej.__contains__("SAMANIEGO")) # SAN DIEGO :: SAMANIEGO
211         and not (nombreDataset.replace(" ", "").__contains__("SANDIEGO")) # SAN DIEGO :: SAN DACIO
212         and not (nombreCallej.__contains__("BARROS")) # BARRIOS :: BARROS -
213         and not (nombreDataset.replace(" ", "").__contains__("MONTEAYA")) # MONTE AYA :: MONTANA -
214         and not (nombreCallej.__contains__("SANTERAS")) # SANZ RASO :: SANTERAS -
215         and not (nombreDataset.replace(" ", "").__contains__("PALOROSA")) # PALO ROSA :: ALCORISA
216         and not (nombreDataset.__contains__("SACROMONTE")) # SACROMONTE :: CERRO MONTE -
217         and not (nombreDataset.replace(" ", "").__contains__("EDUARDOAUNOS")) # EDUARDO AUNOS ::
218         EDUARDO UROSA -
219         and not (nombreDataset.replace(" ", "").__contains__("MADRIDRIO")) # MADRID RIO :: MADRE DIOS
220         ):
221         # Si se han añadido o quitado 2 o menos letras, se puede considerar igual
222         print("-----Posible Coincidencia: ", nombreDataset, " :: ", nombreCallej)
223         return True
224     # -----
225     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 actuaría de diferente manera. Primero igual, ya que podría darse coincidencia por haber eliminado el tipo de vía de la restricción. Si tampoco se diese el caso se comprobaría primero con una letra añadida o eliminada, después con 2 (lo que permitiría letras cambiadas) y después 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

```

1 import csv
2 import os
3 import difflib
4 import unicodedata
5
6 def esTipoCalleOmitible(nombreVia = ""):
7     if(nombreVia == ""):
8         return True
9     nombreVia = nombreVia.upper().replace("Á", "A").replace("É", "E").replace("Í", "I") \
10         .replace("Ó", "O").replace("Ú", "U").replace("Ü", "U").replace(" ", "")
11     if (nombreVia.__contains__("CALLE")):
12         return True
13     elif (nombreVia.__contains__("PASEO")):
14         return True
15     elif (nombreVia.__contains__("PLAZA")):
16         return True
17     elif (nombreVia.__contains__("GLORIETA")):
18         return True
19     elif (nombreVia.__contains__("RONDA")):
20         return False
21     elif (nombreVia.__contains__("CAMINO")):
22         return True
23     elif (nombreVia.__contains__("PISTA")):
24         return True
25     elif (nombreVia.__contains__("ANILLO")):
26         return True
27     elif (nombreVia.__contains__("CRUCE")):
28         return True
29     elif (nombreVia.__contains__("AUTOVIA")):
30         return False
31     elif (nombreVia.__contains__("CARRETERA")):
32         return True
33     elif (nombreVia.__contains__("PARQUE")):
34         return False
35     elif (nombreVia.__contains__("CUESTA")):
36         return True
37     elif (nombreVia.__contains__("CAÑADA")):
38         return False
39     elif (nombreVia.__contains__("AVENIDA")):
40         return True
41     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.

2.5. 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.

2.5.1. 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

```
1 import csv
2 import os
3 from re import findall
4
5 def variosCambiosEnCicloCarriles(nombreCarpeta="", nombreSinCsv="", 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        nColLong = -1
11        nColDistMin = -1
12        for row in csvreader:
13            if (primeraLinea):
14                fila = ";".join(row)
15                nColLong = row.index("Longitud")
16                nColDistMin = row.index("MinSimpTol")
17                # Cambio nombres: tipoUso, nombreVia, distMaxExclusBici y longitud
18                fila = fila.replace("Longitud", "longitud").replace("CAPA", "tipoUso")\
19                    .replace("TX_NOMBRE", "nombreVia").replace("MaxSimpTol", "distMaxExclusBici")
20                #Añadir carrilExclusBici:
21                fila = fila + ";" + "carrilExclusBici"
22                # Añadir municipio:
23                fila = fila + ";" + "municipio"
24            primeraLinea = False
```



```

25         else:
26             # Cambiar la longitud de KM a metros
27             row[nColLong] = (float(row[nColLong].replace(',', '.')) * 1000).__str__()
28             fila = ";" + row[nColLong]
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 kilómetros en metros y en el segundo se añade la palabra Madrid, ya que siempre será el municipio para los datos aquí 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 = ""):
7     with open(nombreCarpeta + "/" + nombreSinCsv + nFileIni + ".csv") as csvfile:
8         csvreader = csv.reader(csvfile, delimiter=";")
9         file = open(nombreCarpeta + "/" + nombreSinCsv + nFileFin + ".csv", "w")
10        primeraLinea = True
11        nRowColumna = -1
12        for row in csvreader:
13            if (primeraLinea):
14                nRowColumna = row.index(nombreColumna)
15                del row[nRowColumna]
16                fila = ";" + row[nRowColumna]
17                primeraLinea = False
18            else:
19                del row[nRowColumna]
20                fila = ";" + row[nRowColumna]
21                file.write(fila + os.linesep)
22        file.close()

```

En el código 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", "MinSimplTol")

```

2.5.2. Cambios relativos a Accidentes de Bicicletas

Los relativos a Accidentes de Bicicletas serán los siguientes:

- Resumen de cambios:
out: N° 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 N°Expediente, 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 modificado 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

```
1 import csv
2 import os
3 from re import findall
4
5 def variosCambiosEnAccidentesBici(nombreCarpeta="", nombreSinCsv="", 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        nColTypicalAge = -1
11        nColGender = -1
12        for row in csvreader:
13            if (primeraLinea):
14                fila = ";".join(row)
15                nColTypicalAge = row.index("RANGO EDAD")
16                nColGender = row.index("SEXO")
17
18                # Cambio nombres: N° Expediente, FECHA, HORA, CALLE, NÚMERO, DISTRITO, TIPO ACCIDENTE,
19                # ESTADO METEREOLÓGICO, TIPO VEHÍCULO, TIPO PERSONA, RANGO EDAD, SEXO,
20                # LESIVIDAD*
21                fila = fila.replace("N° EXPEDIENTE", "numeroExpediente").replace("FECHA", "fecha")\
22                    .replace("HORA", "hora").replace("CALLE", "nombreVia").replace("NÚMERO", "portal")\
23                    .replace("DISTRITO", "distrito").replace("TIPO ACCIDENTE", "tipoAccidente")\
24                    .replace("ESTADO METEREOLÓGICO", "meteorologia").replace("TIPO VEHÍCULO",
25                    "tipoVehiculo")\
26                    .replace("TIPO PERSONA", "tipoPersonaAfectada").replace("RANGO EDAD",
27                    "typicalAgeRange")\
28                    .replace("SEXO", "gender").replace("LESIVIDAD*", "lesividad")
29
30                # Añadir municipio:
31                fila = fila + ";" + "municipio"
32                primeraLinea = False
33            else:
34                # Modificar el Genero:
35                generoEspan = row[nColGender]
36                if(generoEspan == "Hombre"):
37                    row[nColGender] = "Male"
38                elif(generoEspan == "Mujer"):
39                    row[nColGender] = "Female"
40
41                # Modificar TypicalAgeRange
42                row[nColTypicalAge] = getTypicalAgeRange0k(row[nColTypicalAge])
43
44                fila = ";".join(row)
45                # Añadir municipio:
46                fila = fila + ";" + "Madrid"
```

```

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

```

En 2.16 se muestra el código 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 posibilidades en esta propiedad se deberían añadir. En [1] <https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=20f4a87ebb65b510VgnVCM1000001d4a900aRCRD&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnnextfmt=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.

2.5.3. 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; palabras-
Clave; idVia
in: idObject; idEje; idGrupo; tipoUso; nuDireccion; nuDobleS; nombreVia; nombre-
ViaReduc; 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 necesario cambiarlos.

Se ha eliminado la columna ID_TIPO ya que representa con un código 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 más recomendable conservar si la segunda. Se añade la columna municipio asignándole a cada registro el valor "Madrid" para esta propiedad.

Listing 2.18: Función variosCambiosEnCallesTranquilas

```
1 import csv
2 import os
3 from re import findall
4
5 def variosCambiosEnCallesTranquilas(nombreCarpeta="", nombreSinCsv="", 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                # Cambio nombres: OBJECTID, ID_EJE, ID_GRUPO, TX_CAPA, NU_DIRECCI, NU_DOBLE_S,
14                # TX_NOMBRE, TX_NOMBRE_ y Shape_Leng
15                fila = fila.replace("OBJECTID", "idObject").replace("ID_EJE", "idEje")\
16                    .replace("ID_GRUPO",
17                        "idGrupo").replace("TX_CAPA", "tipoUso").replace("NU_DIRECCI", "nuDireccion")\
18                    .replace("NU_DOBLE_S", "nuDobleS").replace("TX_NOMBRE_", "nombreViaReduc")\
19                    .replace("TX_NOMBRE", "nombreVia").replace("Shape_Leng", "longitud")
20                # Añadir municipio:
21                fila = fila + ";" + "municipio"
22                primeraLinea = False
23            else:
24                fila = ";".join(row)
25                # Añadir municipio:
26                fila = fila + ";" + "Madrid"
27        file.write(fila + os.linesep)
28    file.close()
```

En 2.18 se muestra el código 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 función 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&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnnextfmt=default>
- [4] <https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=435a7cd5de319410VgnVCM1000000b205a0aRCRD&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnnextfmt=default>
- [5] <https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=a320f5ac548f4410VgnVCM1000000b205a0aRCRD&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnnextfmt=default>
- [6] <https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=a320f5ac548f4410VgnVCM1000000b205a0aRCRD&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnnextfmt=default>
- [7] <https://datos.madrid.es/sites/v/index.jsp?vgnextoid=b3c41f3cf6a6c410VgnVCM2000000c205a0aRCRD&vgnnextchannel=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>

Anexo

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