Sumario

ÍNDICE: Imágenes y tablas

Índice de las imágenes – Memoria

Índice de las imágenes – Anexo 1 – A1.1

Índice de las imágenes – Anexo 2 – A2.1

Índice de las imágenes – Anexo 2 – A2.2

Índice de las imágenes – Anexo 3 – A3.1

Índice de las imágenes – Anexo 3 – A3.3

Índice de las tablas – Memoria

Índice de las tablas – Anexo 2 – A2.1

Herramienta para la Gestión de accesos para servicios de emergencias en entornos con población dispersa✤

Jon Urteaga Linazasoro TFM - MUSIGT 2017 †

Resumen

El presente trabajo tiene por objeto el desarrollo de una aplicación web para la mejora de la gestión de los accesos durante la asistencia en situaciones de emergencia.

El sistema informático desarrollado pretende cubrir todas las necesidades de información del servicio de emergencias de los Bomberos en un entorno rural y montañoso, y con población dispersa. De forma conjunta desde la Universidad Pública de Navarra, los Bomberos de Navarra y el Sistema de Información Territorial de Navarra se ha visto la conveniencia de mejorar la accesibilidad a zonas de difícil acceso de la comarca del Baztan (merindad de Pamplona, Navarra).

La finalidad principal de este proyecto es la optimización del proceso de obtención de rutas de acceso a las zonas de actuación. Por lo tanto, se desarrollará una herramienta para obtener las rutas más convenientes para acceder en el menor tiempo posible al lugar donde se produzca una emergencia (más concretamente a la vivienda o edificio diseminado[1] en el que se produzca).

De manera complementaria, se pretende también facilitar información de interés asociada tanto al recorrido como a su entorno. De manera que los servicios de emergencia obtengan toda la información relativa al desplazamiento: la composición de las rutas de acceso (tramos y cruces), sus características (tipo de suelo y estado de la carretera, existencia de dificultades como pendientes y curvas, etc.) y la lista de vehículos transitables por dichas rutas.

Asimismo, para poder dar asistencia en una emergencia también será necesario obtener información adicional como sería la del entorno donde se ha producido la

✤ Proyecto de desarrollo tecnológico orientado a la mejora del servicio asistencia de emergencias en un entorno rural con una ortografía montañosa y con la ubicación de la población muy dispersa. Este proyecto ha surgido de SITNA (Sistema de Información Territorial de Navarra).

† Trabajo Fin de Máster (TFM) del Máster en Sistemas de Información Geográfica y Teledetección (MUSIGT) ofertada por la Escuela Técnica Superior de Ingenieros Agrónomos (ETSIA) de la Universidad Pública de Navarra (UPNA) realizada en el curso académico 2016 - 2017.

1 Viviendas o edificios situados fuera de los núcleos urbanos.

incidencia: información referida al propio diseminado (características del edificio, etc.), información de sus habitantes, existencia de vecinos/as, etc.

La aplicación se ha implementado siguiendo una arquitectura informática basada en PostGIS (para el almacenamiento de los datos), GeoServer (para la capa de servicios) y un cliente (para la capa de presentación).

En el desarrollo de este proyecto se han realizado diversos trabajos como: extracción de información desde un archivo KML, el diseño de la base de datos, el almacenamiento de los datos obtenidos en el proceso de carga, la definición del modelo de explotación, la creación de servicios web WMS/WFS y el desarrollo de archivos HTML, CSS y JavaScript.

Como líneas futuras de trabajo se propone el desarrollo de la capa de presentación de la aplicación utilizando la API SITNA, creando funcionalidades más personalizadas de acuerdo a las necesidades del usuario final y con un aspecto más atractivo y más intuitivo.

Palabras clave: Emergencia, Asistencia, Optimización de rutas, Sistemas de información geográfica, SIG, Base de Datos Espacial, Web Mapping.

Memoria del trabajo fin de máster

1. Introducción

En este proyecto de Trabajo Fin de Máster (TFM) se desarrollará un prototipo de herramienta destinada específicamente a los servicios de *Bomberos/ras de Navarra* que trabajan en la Comarca de Baztan, *Baztanaldea*. Aunque esta herramienta también se podría adaptar a las necesidades de otros servicios de emergencia que trabajan en esta misma zona: servicio sanitario de Osasunbidea (ambulancias, etc.), Policía Foral de Navarra, etc.

En Navarra cada año unos 400 incendios afectan a viviendas (en edificios de todo tipo de alturas, garajes, chimeneas y cubiertas): en 2016 fueron un total de 369 incendios de este tipo, y en el año 2015 unas 392. En este sentido, cabe mencionar la casuística especial que requiere una intervención en un entorno como el de Baztan. Concretamente, el parque de bomberos de Oronoz-Mugaire en el año 2015 realizó 55 intervenciones y en 2016 unas 60.

Cabe hacer hincapié en que el municipio donde se sitúa este parque de bomberos tiene una extensión de 376,81 km^2 (es el municipio más extenso de la *Comunidad Foral de Navarra*) y esta compuesto por 15 localidades: Almandoz, Amaiur, Aniz, Arizkun, Arraioz, Azpilikueta, Berroeta, Elizondo, Elbete, Erratzu, Gartzain, Irurita, Lekaroz, Oronoz y Ziga. También contiene un numerosa cantidad de barrios: Aintzialde, Bozate, Ordoki, Pertalats, Mendiola, Iraperri y Aritzakun (barrios de Arizkun), Mardea (barrio de Arraioz), Apaioa, Arribiltoa, Urrasun y Zuaztoi (barrios de Azpilikueta), Antzaborda, Beartzun, Berro y Etxaide (barrios de Elizondo), Gorostopolo, Iñarbil y Iñarbegi (barrios de Erratzu), Aitzano, Ariztegi y Etxerri (barrios de Gartzain), Aroztegia, Oharriz y Uharte (barrios de Lekaroz), Mugairi y Zozaia (barrios de Oronoz), Zigaurre (barrio de Ziga) y el territorio de Quinto Real (también con zonas habitadas). También existen zonas de caserío dispersos como Azkar, Etxartea, Madaria y Olazur. Esto contribuye en gran medida a que en este municipio haya un gran número de carreteras, caminos rurales y pistas forestales.

Por todo ello, al producirse un aviso de urgencia desde un domicilio cuya ubicación está situada en una zona alejada del centro urbano (y en una zona montañosa de difícil acceso), los servicios de emergencia deben seleccionar correctamente la ruta a seguir para llegar en el menor tiempo al destino. Y para poder determinar dicha ruta óptima, será decisivo el acceso que puedan tener a la información espacial del entorno de actuación.

También cabe mencionar que esta unidad de emergencia contiene vehículos con una cierta dificultad para transitar por algunos tramos de un entorno montañoso (por sus dimensiones o capacidades de maniobra), pudiendo darse la posibilidad de que alguno de estos vehículos no sea apto para una determinada emergencia, al no ser capaz de llegar a su destino en el tiempo requerido. Por consiguiente, para elegir correctamente la ruta de acceso será preciso obtener información detallada de las características de las carreteras y los caminos rurales.

En resumen con este proyecto de desarrollo tecnológico se pretende implementar una herramienta que ayude a lograr el éxito del desplazamiento, ofreciendo la información necesaria para elegir correctamente tanto la ruta como el vehículo.

Para ello, la herramienta que se pretende desarrollar tendrá tres funcionalidades reseñables:

1. Visualización del itinerario a seguir.

   Tras la identificación de la ruta óptima la herramienta posibilitará su visualización en un *visor geoweb* o *Geo Visor*[1]. De esta manera, en el transcurso del desplazamiento se podrá consultar dicha ruta, y por lo tanto, se transitará por el recorrido esperado y no habrá equivocaciones.

2. Consultar la información relativa al entorno.

   Este visor posibilitará la consulta de la información del entorno a lo largo del cual se transita, permitiendo anticiparse a las dificultades que podrían darse en el transcurso del itinerario.

3. Consultar la información relativa a la vivienda.

   Esta herramienta también facilitará la consulta de información referida a la vivienda de destino (imagen del edificio, etc.) como información reservada de sus habitantes (ficha médica, etc.). De esta manera, los/las bomberos/ras obtendrán información relevante para actuar adecuadamente ante una emergencia.

1.1. Antecedentes

La necesidad de desarrollar una herramienta de estas características surgió desde la sección de intervención de los/las *Bomberos/ras de Navarra* que trabajan en el parque comarcal de Oronoz-Mugaire (parque de bomberos/ras de la comarca de Baztan). Asimismo, otros servicios de asistencia de emergencias de la misma zona de trabajo venían reclamando una herramienta eficiente que indicara el itinerario a seguir para llegar en el menor tiempo posible al lugar de una incidencia (especialmente en los casos que requerían un desplazamiento alejado de los núcleos urbanos).

Es este contexto, surgió el proyecto desarrollado por varios/as empleados/das del *Servicio de Urgencia Rural* de *Osasunbidea*. En este trabajo se definió un *sistema de geolocalización*[2] para facilitar la obtención de rutas de acceso a viviendas donde se podría producir una emergencia. Su finalidad era solventar en la atención de avisos de urgencias del sistema 112 los problemas surgidos en zonas montañosas con población dispersada, y conseguir desplazarse sin pérdida y en el menor tiempo posible al lugar donde se necesita la asistencia sanitaria.

1 Visualizador de datos geográficos.

2 “Nuevo sistema de geolocalización en Navarra para disminuir los tiempos de respuesta en aviso urgente en zonas de montaña y de gran dispersión”. Artículo realizado por José Miguel Ablitas Muro (miembro de Urgencias rurales del Centro de Salud de Elizondo en el año 2012, en Osasunbidea), con la colaboración de Agurtzane Goienetxe Goñi, Pablo González Lorente y Alberto Istúriz Abadra.

De esta manera, se pretendía ajustar las isócronas sanitarias a los tiempos de respuesta recomendados, y poder así ofrecer a los habitantes de estas zonas rurales una correcta cobertura sanitaria en urgencias extra-hospitalarias.

1.1.1. Estudio realizado por el Servicio de Urgencia Rural de Osasunbidea

En este trabajo, estos/estas sanitarias recalcaban las dificultades de localizar caseríos dispersos en el valle del Baztan, al no disponer éstos una dirección convencional acorde a una vivienda de una zona urbana. Es decir, la dirección de estos caseríos no contenía un nombre de una calle y un número de portal, sino el nombre propio de la vivienda. Por lo tanto, a la hora de atender una emergencia utilizando un software de navegación GPS se producía dificultades al no poder obtener una ruta de acceso a dicho destino. Por ello, hacían hincapié en la necesidad de disponer un dispositivo que facilitase la identificación de estos domicilios, la obtención de su localización (es decir, sus coordenadas geográficas), y posteriormente la ruta de acceso a ella.

Para que este sistema fuese válido en urgencias extra-hospitalarias y pudiesen conseguir las rutas requeridas fue fundamental la creación de una base de datos acorde a las necesidades de un servicio con esta casuística. Por tanto, en esta base de datos fue muy importante una correcta nomenclatura de los edificios (y de la toponimia del entorno), siendo los nombres de las viviendas las mismas que las utilizadas por los/las vecinos/as del lugar.

En la investigación desarrollada para mejorar el tiempo de respuesta, analizaron y compararon diferentes dispositivos informáticos basados en datos georreferenciados (de ubicaciones) y enrutamiento (el establecimiento de una ruta al disponer una ubicación origen y otra de destino), y los aplicaron a las necesidades de la actividad que ejercían en las fechas de mayo a septiembre del 2012 en la zona del Baztan.

Análisis y conclusiones sobre el estudio realizado.

Se observaron las limitaciones de cada una de estas aplicaciones de navegación (implantadas en dispositivos móviles sanitarios) y obtuvieron las siguientes conclusiones:

- Necesidad de una herramienta (integrado en los móviles sanitarios no corporativos) que facilite con precisión el acceso del sanitario al paciente demandante, al no ser los sistemas de navegación GPS habituales válidos para avisos domiciliarios urgentes. Esta herramienta tendría que permitir a los/las profesionales desplazarse sin pérdida y en el mínimo tiempo posible por carreteras y caminos rurales sin requerir un amplio conocimiento geográfico de la zona de trabajo.
- Necesidad de obtener un sistema donde se pueda localizar un destino partiendo del nombre de la casa (caserío o vivienda), del nombre del propietario y/o de la zona en la que se ubique (barrio, parajes especiales, etc.).
    - Por lo tanto, sería fundamental establecer con anterioridad las localizaciones exactas de caseríos y zonas que aglutinen viviendas (sobre todo en zonas montañosas con la

población muy dispersada), y recoger las nomenclaturas adecuadas de estos diseminados.

  - Estos datos (en un aviso domiciliario urgente) serán fundamentales para obtener la localización (y posterior ruta) y realizar correctamente el acceso del/la sanitario/a al paciente.
  - El proceso de generación de datos espaciales recomendada para estos diseminados se realizaría de las siguiente manera: a través de la herramienta *Google Maps*, utilizando imágenes satélite, se procederá a crear puntos que representen a los diseminados (creando de esta manera un archivo KML). A continuación cada uno de ellos se etiquetará utilizando nombres autóctonos (incluyendo el nombre del barrio y el pueblo al que pertenece). Estos datos contendrán coordenadas de longitud y latitud en el sistema WGS 84.

- Esta herramienta también deberá facilitar la ruta (o las rutas) de acceso al punto de destino. Estas rutas tendrán que ser rutas prefijadas, y tendrá que conformar un itinerario válido e inequívoco para el desplazamiento. De esta forma un/una sanitario/a podrá realizar sus servicios sin tener un conocimiento previo del entorno geográfico que trabaja.
- Los paquetes de softwares GPS habituales (al introducir un punto de origen y otra de destino) calculan varias rutas (y establecen diferentes accesos) para llegar al diseminado especificado. A continuación el/la usuario/a de la misma elige entre todas ellas la más óptima (la más rápida, la más cómoda o la que más se adapte a las necesidades del servicio).
- Estos sistemas de navegación GPS pueden calcular rutas no acordes a las utilizadas por los habitantes de la zona, y pueden incluir tramos o sitios de difícil transito. Por ello, la elección correcta entre diferentes rutas dependerá del buen conocimiento del terreno.
- Para evitar esta situación se plantea trabajar con archivos KML de rutas prediseñadas. Estas rutas se definirán por los propios técnicos sanitarios utilizando aplicaciones de navegación GPS que permitan almacenar rutas y posteriormente descargarlos (aplicaciones como *OruxMaps* y *Garmin*).
- Los archivos KML (de rutas) estarán organizados por barrios y pueblos, y cada una de las rutas contendrá el nombre del caserío que representa. Todas ellas estarán ordenadas (alfabéticamente) para poder así buscar más fácilmente en caso de búsqueda.
- Necesidad de definir la forma de utilización de los archivos KML (de diseminados y de rutas): definir cómo hay que compartirlos (entre los/las profesionales sanitarios/as) y en qué dispositivo hay que integrarlos.
  - Definir la opción más adecuada para compartir estos archivos utilizando escasos recursos informáticos (y poca capacidad de memoria). Para ello, se plantea analizar varias opciones: utilizando mapas on-line, a través del correo electrónico, la aplicación *WhatsApp* y el servicio de mensajes *SMS*.

  - El sistema planteado se ha implementado satisfactoriamente en diferentes dispositivos:
    1. Sistema de navegadores GPS: *TOMTOM* y *Garmin*.
    2. Sistemas operativos de teléfonos móviles: *Android*, *iOS* (*iPhone*, *Apple*) y *WP* (*Windows Phone*).
    3. Software de navegación utilizados: *Windows Navigation*, *Sygic*, *Co-pilot*, *IGO*, *Navigon*, etc.
    4. Motor de carga para aplicaciones (de archivos KML): *OruxMaps* (*Android*) y *App KMLmaps* (*iOS*).
  - Ventajas de la utilización de archivos en un formato estandarizado como KML[3]: es totalmente integrable en la mayoría de dispositivos y existen sistemas de conversión (de este formato) a otros formatos específicos. Por lo tanto al no representar ninguna incompatibilidad relevante, se podrá utilizar en *Navegadores GPS*, *Smartphone* y *Tablets* indistintamente.
  - Ventaja relevante al utilizar aplicaciones de navegación GPS que funcionen en modalidad off-line: posibilidad de utilizar el sistema en zonas donde no existe cobertura datos de internet (ni cobertura de llamadas de los teléfonos móviles). Por lo tanto, se asegura la funcionalidad total del dispositivo en todos los tramos de las vías en el valle del Baztan.
- Un sistema así permitiría desplazarse sin ninguna pérdida ni demora en el tiempo, y contribuirá a la disminución de costes y al aumento de la eficiencia de la asistencia y, por tanto, de la calidad asistencial.
- Posibilidad de adjuntar datos adjuntos a los datos espaciales de los diseminados. Estos datos contendrán información reservada (historial clínico, etc.) de personas afines (empadronadas, etc.) a estas viviendas.
- Necesidad de mantener actualizada la base de datos: posibilidad de actualizar los archivos KML in situ e incorporar puntos inexistentes en ella.
- Este sistema, más allá de la atención de urgencia sanitaria, también podrá facilitar el trabajo a otros/as profesionales (del mismo ámbito rural): técnicos/as de ambulancia, repartidores/as de oxígeno a domicilio, protección civil, bomberos/as, etc.
- Este sistema permitirá la distribución de información espacial entre diferentes servicios de emergencia (en los casos como accidentes múltiples o incidencias que requieran la actuación conjunta de varios recursos).

3 Keyhole Markup Language, KML. Es un lenguaje de marcado basado en XML (Extensible Markup Language) para representar datos geográficos en tres dimensiones [definición KML, Wikipedia].

1.1.2. Metrobaserri: mapa de vías simplificadas y señalización de acceso a diseminados

En este contexto también se han desarrollado diferentes iniciativas en Navarra, para facilitar la localización y acceso de caseríos dispersos de servicios de emergencia en zonas rurales. Una iniciativa a recalcar es el proyecto de *Metro-Baserri* [4]. Este proyecto está desarrollada por el *Gobierno de Navarra* para municipios con una orografía que dificulta el acceso a viviendas, como es el caso de Leitza, Zubieta y Goizueta (ver Imagen 1).

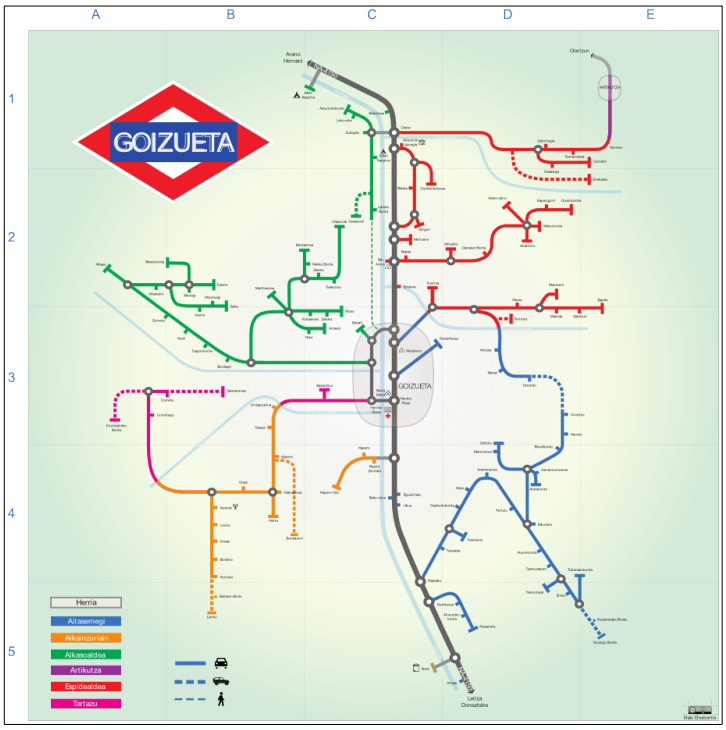

Imagen 1: Mapa metrobaserri del municipio de Goizueta

En el proyecto *Metro-Baserri* principalmente se pretende crear mapas que representan de manera simplificada y clara las vías a transitar para llegar a los diferentes diseminados existentes en un municipio. En este proyecto también se incluye la señalización en campo de la localización de los diseminados, señalizando tanto cruces de carreteras como entradas a las viviendas. Con esta iniciativa se pretende facilitar aún más el acceso a las viviendas y garantizar así la correcta elección de vías en el desplazamiento. Estos mapas están orientados a profesionales del servicios de emergencias como bomberos/as, sanitarios/as o técnicos/as de ambulancias.

1.2. Definición de las bases del proyecto SITNA – Bomberos de Navarra

Teniendo en cuenta lo mencionado en el apartado anterior (en el sistema desarrollado por sanitarios/as de *Osasunbidea* y en el proyecto *Metro-Baserri*) se definieron las bases del proyecto SITNA – Bomberos Navarra. Estos son las bases explicadas en la mesa de coordinación del proyecto (se incluye una versión más extendida está en Anexo A1.1):

4 Proyecto impulsado por el Gobierno de Navarra en el contexto de *Mapas topológicos de ámbito municipal.*

- Necesidad de una adecuada base de datos para la herramienta: centralizada, normalizada y georeferenciada.
- Datos fundamentales en la base de datos: información asociada a los diseminados e información referida a las rutas (de acceso a diseminados). Establecer una metodología para crear la base de datos e insertar y actualizar registros de información.
- Utilización de PostegreSQL con extensión PostGIS como respuesta a la necesidad de utilizar una base de datos espacial gestionada por un sistema que permita representar datos geográficos.
- Desarrollo de una aplicación web de tipo *web mapping* que permita a los/las usuarios/as de esta herramienta el acceso y explotación del contenido de la base de datos.
- La aplicación contendrá un servicio de localización de diseminados donde se obtendrá información espacial asociada a la ubicación de emergencias. Esta aplicación también tendrá un servicio de descarga donde se podrá obtener la ruta de acceso al diseminado localizado anteriormente. Por último la aplicación tendrá un servicio de visualización donde se podrá visualizar los datos espaciales seleccionados y otros servicios web OGC estándar. Por lo tanto, esta aplicación (al contener estos tres servicios) conseguirá ofrecer técnicamente una herramienta útil para resolver el problema propuesto en este proyecto.

2. Análisis de los requisitos del Proyecto

En este apartado se analizan los requisitos de la aplicación a desarrollar, considerando tanto los requisitos funcionales como no funcionales o de calidad del servicio. Estos requisitos se explicarán más detalladamente en el Anexo A1.1.

2.1. Requisitos funcionales

El sistema que se pretende desarrollar con este proyecto contiene dos tipos de usuario: por una parte, existirá el perfil del administrador y por otra los usuarios con perfil de cliente.

1. El administrador será el gestor de la aplicación. Este tipo de usuario podrá añadir, modificar y eliminar información del sistema.
   - Podrá añadir o borrar registros de zonas, destinos, habitantes, salidas, servicios, vehículos, rutas, cruces, puntos de información, tramos, suelos, estados y tramos rurales.
   - Podrá completar o modificar información asociada a:
     - Zonas: el nombre, el tipo de zona, la zona padre que pertenece, etc.
     - Destinos: el nombre, el tipo de diseminado, la dirección, la fotografía de la fachada, la habitabilidad, los riesgos potenciales de edificio, datos de contacto del propietario, los destinos vecinos relacionados, etc.
     - Habitantes: el nombre, los apellidos, el código nif, la fecha de nacimiento, la edad, datos de contacto, dirección donde residen, etc.
     - Salidas: el nombre, tipo de salida, el servicio de emergencia que pertenece, etc.

- Servicios: el nombre, el tipo de servicio, el contacto, etc.
- Vehículos: el nombre, el tipo de vehículo, las limitaciones de tránsito, el servicio de emergencia que pertenece, etc.
- Rutas: la salida de la ruta, el destino la ruta, el número de tramos que contiene, etc.
- Cruces: las observaciones sobre el cruce, la ruta a que pertenece, etc.).
- Puntos de información: (el nombre, la descripción del aviso, la ruta a que pertenece, etc.
- Tramos: el nombre, el tipo de tramo, la longitud, el tiempo de desplazamiento, la ruta a que pertenece y en qué orden, los vehículos que puedan transitar por ella, etc.
- Suelos: el tipo de suelo, la característica del suelo, etc.
- Estados: el tipo de estado, la característica del estado, etc.
- Tramos rurales: el suelo y el estado que contiene el tramo, la fecha de la última clasificación del suelo y estado, la existencia de curvas cerradas y pendientes altas, las advertencias, etc.

- También podrá añadir o borrar tipos de zona, de destino, de salida, de servicio, de vehículo, de tramo, de suelo y de estado.

2. El cliente será un usuario de un servicio de emergencia.
   - Podrá utilizar un buscador para identificar el diseminado que pretende localizar.
   - A continuación podrá visualizar la representación espacial del diseminado, su entorno y la ruta (una o varias) asociada a ésta. Esta visualización la realizará con el Geo Visor, y en ella se podrán añadir capas con otra información de interés.
   - En todo momento se podrá consultar información relacionada con el diseminado (información referida al edificio e información relativa a sus habitantes) y con la ruta: longitud de los tramos, lista de vehículos que pueden transitar por ella y las condiciones de los tramos que conforman las rutas.
   - Al elegir la ruta podrá descargar toda la información a su dispositivo, y lo podrá compartir con otros servicios de emergencia.
   - Podrá ver durante el transcurso del desplazamiento la ubicación del vehículo dentro de la ruta.
   - Al llegar al diseminado podrá consultar la información referida al destino para ayudar a realizar correctamente la asistencia.
   - En un principio, no podrá editar ningún dato ni registrar incidencias con esta herramienta.

2.2. Requisitos no funcionales

Los requerimientos no funcionales del sistema desarrollado se pueden englobar en tres apartados generales:

1. Entornos y herramientas de desarrollo del proyecto:

   La mayor parte del proyecto se ha desarrollado en el sistema operativo Microsoft Windows XP, aunque algunos desarrollos se han realizado en Microsoft Windows 10.

   Todos los programas utilizados son de software libre, y existen versiones compatibles con el sistema operativo GNU/Linux, por lo que el desarrollo se podría haber realizado igualmente sobre este sistema.

2. Requerimientos para desplegar la aplicación en un entorno de producción:

   Se requieren las siguientes piezas de infraestructura:

   - Servidor Base de Datos en el que correrá el gestor de Base de Datos con soporte a tipos de datos espaciales PostgreSQL/PostGIS.
   - Servidor de mapas en el que correrá el servicio de mapas Geoserver.
   - Servidor web para la publicación de la página web del visor incluyendo todos los archivos necesarios (HTML, CSS y JavaScript).

3. Requerimientos del puesto del usuario para acceder a la aplicación:

   El usuario podrá utilizar cualquier tipo de navegar web para acceder a la información existente en la aplicación. Y para ello, podrá utilizar los navegadores web de cualquier tipo de sistema operativo.

   También el usuario podrá utilizar el software QGIS (o otro software GIS) para visualizar el contenido de los servicios de mapas Geoserver.

3. Análisis de datos de Entrada

Lo primero que se realizará en el proyecto como paso previo a la definición del diseño del modelo de datos y de su estructura, será inspeccionar detenidamente la muestra de datos facilitada por los/las bomberos/ras de Baztanaldea. De esta manera, el diseño partirá de la información de entrada (el contenido y la estructura) creada por y para el servicio de emergencia. En este apartado solamente se indicará un resumen del análisis realizado. El análisis completo se puede consultar en el Anexo A2.1.

3.1. Archivo KML de muestra de datos analizado

3.1.1. Elementos Name y Schema

3.1.1.1. Diferentes estructuras de datos para un mismo tipo de diseminado

Al analizar el contenido de los ficheros KML se han encontrado varias estructuras de datos. Entre ellas existen varias para representar la información referida a **Destinos**.

1. Una de ellas es la estructura de datos “Basaburua”, donde se utiliza la etiqueta **Schema** para definirlo.
2. Otra estructura de datos se llama “BasaburuaCompleto”. Pero en este caso no existe ningún etiquetado Schema con este atributo de identificación id. Por lo tanto, esta estructura no está definida en el código KML.
3. También hay otros tipos de diseminados con otro tipo de estructura de datos, pero en este caso no se especifica qué nombre tiene dicha estructura.

En este aspecto cabe mencionar que hubiese sido preferible definir las estructuras de datos al principio del documento, para posteriormente utilizarlos en la asignación de datos a un elemento. De esta manera, el documento KML sería más claro y se podría hacer cambios más fácilmente. También hay que mencionar que los diseminados tendrían que tener una misma estructura de datos, algo que no ocurre al existir varias versiones de dicha estructura. Es decir, los diseminados que representan los destinos de las rutas tendrán que tener la misma información, y por lo tanto deberían tener la misma estructura de datos.

3.1.1.2. Diferentes formas de asignar valores a variables de datos

En este documento KML se aprecia la existencia de diferentes formas para añadir valores a los datos asociados a los diseminados.

1. Sólo en algunos casos la asignación de los valores se realizaba haciendo referencia a una estructura de datos predefinida.
2. Existen otros casos donde se asigna valores siguiendo un orden y manteniendo el mismo tipo de contenido que en estructuras predefinidas, pero haciendo uso de elementos **CDATA** dentro de la etiqueta Descripción.

En este sentido, sería igualmente deseable definir una sola forma de asignación de valores en todo el documento, de manera que si se pretende modificar algún valor de los diseminados se realizaría de la misma forma para todos ellos.

3.1.2. Elementos Style y StyleMap

3.1.2.1. Diferentes ubicaciones para la declaración de los estilos

En el documento KML se aprecia diferentes ubicaciones para la declaración de estilos de objetos. Es decir, la declaración de estilos se sitúa en diferentes niveles del etiquetado.

Esta organización dificulta el análisis y la modificación del contenido del documento.

3.1.2.2. Diferentes tipos de símbolos para diseminados

Se puede observar la existencia de diferentes símbolos (de tipo puntual) para representar los diseminados en el mapa. Más concretamente, se han utilizado dos tipos de iconos. Las imágenes de estos símbolos se pueden consultar en Anexo A2.1.

También se han utilizado diferentes colores para representar dichos iconos. Estos cambios de color se han realizado para agrupar diseminados, y diferenciar así visualmente diferentes grupos o zonas en el mapa.

En este caso, el problema es que este criterio de representación a variado dependiendo de las zonas, y refleja las diferentes campañas realizadas en la toma de datos.

3.1.2.3. Otro tipo de símbolos para representar puntos en el mapa

En este fichero KML también se hace referencia a otros tres tipos de símbolos:

- Simbología de diseminados con datos para emergencias.

Existen dos clases de símbolos de este tipo que se utilizarán para representar los diseminados en el mapa en función de la información contenida/representada. El primer caso (el símbolo representado mediante un icono en forma de H) contiene información referente a sus habitantes, y el segundo caso (icono Arrow) contiene a la vez datos relacionados con el diseminado y los habitantes.

- Símbolo de advertencia.

Este tipo de icono hace referencia a un aviso puntual y no representan un diseminado. Este tipo de icono se utilizará para ubicar un mensaje de advertencia en el mapa. Por lo tanto, este tercer caso (icono Caution) contiene solamente una descripción del mensaje de advertencia.

3.1.2.4. Diferentes colores para representar diferente tipos de tramos

El recorrido que representa las rutas de acceso transcurre por diferentes vías: carreteras y caminos rurales. Por ello, los tramos lineales que representan las carreteras son de un color diferente a los tramos relativos a los caminos rurales.

La parte rural de las rutas pueden tener características diferentes a lo largo de su recorrido, y podrán estar compuestas por varias subtramos. Al analizar el color de dichos tramos se aprecia la existencia de diferentes criterios en función de cada zona o barrio (espacial) o en función de la campaña de toma de datos (temporal). En algunos casos se utiliza el mismo color para representar los tramos rurales de una ruta, y en cambio en otras se utilizan diferentes colores para lo mismo.

3.1.2.5. Carencias y deficiencias en la definición de los estilos

Al analizar la forma que se han realizado las definiciones de estilos se ve claramente las carencias y deficiencias de la metodología definida en la captura de datos y creación del fichero KML. Por lo tanto, se tendrá que definir otra manera de definir las asignaciones de los estilos. Brevemente se mencionarán varias recomendaciones de cómo debería de realizar dichas declaraciones de estilos:

- La declaración de los estilos se debería realizar en el mismo nivel de etiquetado, y uno detrás de otro. Es decir, todas las declaraciones existentes de estilos se deberían agrupar.
- Todos los diseminados tendrían que contener la misma estructura de datos, y también la composición de información que contienen debería ser la misma. Por ello, se recomienda añadir la información de los habitantes en los casos de diseminados donde solamente contienen información de la vivienda o edificio, y añadir los datos de la vivienda en los casos de diseminados que contienen sólo la información de los habitantes. De esta forma, cada caso de diseminados contendrían un mismo tipo de simbología.
- Estos símbolos se podrán colorear de diferente manera si se pretende representar agrupaciones (como barrios o pueblos) o si se quiere representar gráficamente una clasificación de un parámetro (como intervalos de distancia o de tiempo de desplazamiento desde el parque de bomberos).
- Los tramos de carretera y los tramos rurales que componen una ruta se diferenciarán con diferentes colores. Los tramos rurales de una misma ruta contendrán el mismo color. Pero habrá la posibilidad de que estos tramos se representen con diferentes colores siguiendo una clasificación de un parámetro que contiene. Por ejemplo, este parámetro podría ser el grado de dificultad de la transitabilidad de la vía. Este parámetro se podría calcular a partir de las limitaciones de circulación que contienen ciertos vehículos de la unidad de emergencia: lista de características de los tramos (cierta composición del suelo, estado de la vía, existencia de pendientes altas y curvas cerradas, etc.) que impiden el transito de los diferentes vehículos de emergencia.

3.1.3. Elementos Placemark y Folder, marcas de posición y carpetas

Los elementos *Folder* se utilizan para agrupar diferentes elementos. En este caso, se han utilizado para diferentes propósitos:

1. Para agrupar varios elementos Placemark (un elemento o varios de punto y un elemento o varios de cadena de líneas) y componer así un grupo de marcas de posición asociado a un diseminado. En estos casos (carpetas de diseminados), se les asignará como nombre de carpeta el mismo nombre del diseminado.
2. Para agrupar estas carpetas de diseminados, y conformar así zonas o barrios reseñables.
3. Para agrupar diferentes campañas de toma de datos de una misma zona.
4. Para agrupar varios placemarks de tipo punto. Son grupos de diseminados pendientes (de procesado) que no contienen ninguna ruta asociada a ellas.
5. Para agrupar varios placemarks de tipo línea, donde no exista ningún placemark de tipo punto asociadas a ellas. Este caso, conforma un grupo de rutas pendientes (de procesado).

3.1.3.1. Problemas con la nomenclatura de los diseminados

Al analizar la estructura de este documento se han identificado varios problemas asociados a la nomenclatura:

1. Existencia de varios diseminados con el mismo nombre.
   - Por un lado están los diseminados con el mismo nombre pero ubicados en diferente zona, barrio o pueblo.
   - Por otro lado están los que se sitúan en la misma zona. Por lo tanto, pueden surgir confusiones. Para evitar los problemas con el nombre de los diseminados se utilizará el nombre del propietario junto al nombre de la casa. De esta forma se pretende evitar equivocaciones en la identificación del destino.

2. Existencia de varios diseminados sin ningún nombre: se han encontrado tres casos en el documento.

   Por lo tanto, si se crea alguna incidente en un diseminado de este tipo habrá dificultades para identificarlo, acceder a la ruta de acceso y consultar la información que contiene. Por esta razón, en el documento KML no podrá haber ningún diseminado sin un nombre asociado a ella.

4. Confusiones producidas por el desconocimiento del euskera.

   Una persona residente en este valle al realizar una llamada al 112 (para dar aviso de una incidencia) es probable que haga uso del *baztanés* para referirse a una vivienda o zona (aunque esa persona se dirija en castellano). Por lo tanto, habrá que analizar la pronunciación de este dialecto para evitar posibles confusiones en la interpretación del nombre de la vivienda.

5. Confusiones producidas al realizar el proceso de búsqueda.

   Existen diferentes formas de escribir un mismo nombre de una vivienda, por lo tanto habrá que identificar las posibles confusiones que se podrían crear. Para ello, se analizará todos los nombres de los diseminados utilizados en el documento KML, y se propondrá posibles cambios para evitar errores (la modificación de algunas nomenclaturas, la introducción de sinónimos, la utilización de técnicas de búsquedas fonéticas, etc.). De esta forma, se agilizará el proceso de búsqueda.

   Por lo tanto, se ha procedido a analizar los siguientes casos:

   1. La utilización de la palabra *Borda*, *Etxea* y otras referencias a tipos de edificios o edificaciones.
      - Las palabras “Borda” y “Etxea” (o “Etxe”).
      - Las palabras como “Venta” (Benta), “Hotel” (Hotela), “Agroturismo” (Landa Etxea), etc.

- Las nomenclaturas que contienen solamente la tipología del edificio, sin ninguna toponimia.
- Las nomenclaturas que contienen la tipología del edificio y un nombre (la toponimia o el nombre del negocio o propietario).

2. La utilización de la *TX*, *TS*, *TZ* y K.
3. Los nombres de diseminados con terminaciones en – *a*.
   - Primer caso: los nombres con terminaciones de -alde(a), -ondo(a), -buru(a), -leku(a) y -gain(a).
   - Segundo caso: los nombres con terminaciones de -berri(a), -zuri(a) o -txuri(a), -eder(ra), -txar(ra), -garai(a), -txiki(a) y -argi(a).
4. Las nomenclaturas con terminaciones en – *(e)nea*.

Este análisis se puede consultar íntegramente en el Anexo A2.1.

3.1.3.2. Duplicidad de tramos de rutas en el documento

Al analizar los tramos de las rutas se puede observar los siguientes problemas:

- Los tramos (que componen las rutas) están delimitados por cruces de vías. Esta delimitación refleja un cambio en el tipo de vía o características de la calzada, pero no en todo los casos. En el fichero KML analizado se aprecian dos tipos generales de tramos: tramos de carretera y tramos rurales. Los tramos de carretera contienen los mismos datos relativos al tipo de calzada, en cambio los tramos rurales no. Estos tramos rurales a su vez están compuestas por varias subtramos rurales. Pero al analizar sus características se aprecia que contienen una mezcla de composición de suelo y características de calzada.
- En las rutas analizadas, los tramos no están exactamente delimitados por los cruces físicos de las carreteras, y cada tramo empieza y acaba siguiendo diferente criterio.
- Cada una de las rutas contiene sus propios tramos, y son diferentes a los tramos de las otras rutas. Por tanto, existe información redundante al solaparse todos estos tramos. Estos tramos no siguen el mismo eje de la vía, y crean diferentes rutas alrededor de éste.
- Existen casos de rutas donde sus tramos no llegan a conectarse entre sí, y otros casos donde el último tramo no llegan exactamente al punto de destino. Ver ejemplos en Anexo A2.1, en el apartado 1.4.2.
- Para llegar a un diseminado cabe la posibilidad de que existan varias formas de acceso. Por lo tanto, un diseminado puede contener varias rutas de acceso (por ejemplo, una ruta directa y otra ruta alternativa). Para diferenciar los diferentes conjuntos de tramos que forman cada ruta habría que guardarlos en diferentes carpetas del documento KML. En el documento existe un caso donde se ha realizado incorrectamente dicha agrupación: no se ha completado correctamente toda la ruta de acceso y sus tramos no conforman del todo bien el recorrido.

3.1.3.4. Carencias y deficiencias en la delimitación de los tramos

Al analizar la forma que se han delimitado los tramos se ve claramente las carencias y deficiencias de la metodología definida en la captura de datos. Por lo tanto, se tendrá que definir otra manera de definir los límites de los tramos. Brevemente se mencionarán varias recomendaciones de cómo debería de realizar la incorporación de nuevos datos en el fichero KML:

1. En la base de datos no tiene que haber tramos duplicados, y por tanto, un mismo tramo tendrá que ser parte de diferentes rutas de acceso. Con este requisito, en la base de datos habrá menos registros almacenados, y al realizar modificaciones de los tramos en la base de datos habrá menos operaciones que realizar.
2. Se evitará el solapamiento de los tramos, y con ello, se creará una estructura de tramos única. De esta manera, se evitaría en la base de datos información espacial redundante.
3. Los tramos habrá que delimitarlos exactamente hasta los cruces de carreteras.
4. Se crearán subtramos para diferenciar cambios en los datos relativos a la calzada de la vía (el tipo de vía o composición del suelo). De esta manera, los tramos tendrán una características más homogénea, y la clasificación sobre su transitabilidad se realizará más adecuadamente. Pero no se delimitarán subtramos muy pequeños.
5. Las tramos incorrectos o incompletos se corregirán espacialmente. Al disponer información geográfica de la zona de trabajo, en los casos donde el eje de los tramos se desvían excesivamente del eje de las vías se podrán corregir. También se podría completar los tramos donde la ruta no llega al punto de destino. Por lo tanto, habrá un trabajo de edición de datos espaciales de la muestra de datos existente.
6. Se definirá con exactitud el cruce de carretera que refleja el cambio del tipo de tramo de la ruta. Es decir, en cada ruta se tendrá que definir con una cierta precisión el cruce que define el final del tramo de carretera y el inicio de la parte de tramo rural.
7. En una primera versión se creará una única ruta de acceso para cada diseminado, pero en algunos casos habrá que crear una ruta de acceso alternativa para acortar el tiempo de asistencia de emergencia (o ajustar las isócronas sanitarias, etc.). Por lo tanto, si existen varias rutas de acceso para un diseminado habrá que especificar claramente qué vehículos pueden circular por cada una de ellas, y qué delimitaciones contiene cada acceso.

3.2. Proceso de extracción de la muestra de datos

En este apartado del documento se analizará brevemente cómo adaptar la información facilitada por *Bomberos de Navarra* a la estructura y el contenido necesario para el nuevo sistema. A este proceso de reorganización de los datos se le denomina proceso ETL (*Extract*, *Transform*, *Load*), donde se permitirá trasladar los datos desde diferentes fuentes a la base de datos (*cargar datos*), limpiar y modificar su contenido (*transformar datos*) y posteriormente obtener los datos necesarios para la herramienta de explotación (*extraer datos*).

Como paso previo al proceso ETL es necesario crear la base de datos. En ella se crearán las estructuras de datos acordes al modelo de datos diseñado para recibir los datos de entrada. También se definirá el sistema de referencia utilizado por los datos espaciales (coordenadas) de la geodatabase.

Posteriormente se realizará la fase de carga. En esta fase, se obtendrán los datos del fichero KML y se almacenarán en la base de datos. Para ello, se utilizará una herramienta de carga específicamente desarrollada para el proyecto en el lenguaje de programación *Python*.

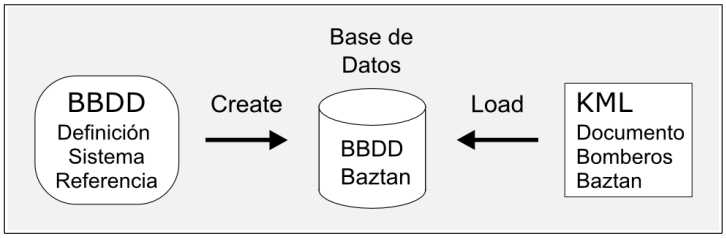

Imagen 2: Creación de la base de datos y posterior carga de datos

La siguiente fase será la fase de transformación. En esta fase, se modificarán valores de algunos registros (cargados en la fase anterior) y se añadirá nueva información (requerida para cubrir las necesidades de este proyecto) usando un lenguaje SQL (para acceder y realizar operaciones en el sistema de gestión de bases de datos relacional utilizado). También se editarán los datos espaciales utilizando un software GIS (en nuestro caso QGIS).

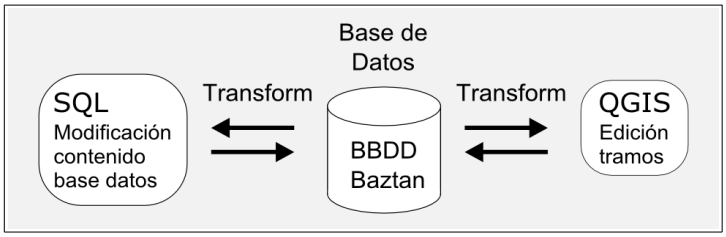

Imagen 3: La realización de la transformación de datos

En esta fase también se transformará la propia estructura de la base de datos para así crear relaciones de datos acorde al modelo de datos diseñada para la herramienta.

La última fase será la fase de extracción. En esta fase se procederá a extraer toda la información almacenada en la base de datos, creando vistas para la publicación. De esta manera, estas vistas se podrán subir al servidor de *Geoserver* y se podrá así compartir toda la información con la aplicación desarrollada.

4. Diseño técnico de la base de datos

El diseño técnico de la base de datos consistirá en la definición de la estructura de datos que sirva como base de este sistema de información geográfica. El proceso se divide en varias fases de diseño donde se crearán sucesivamente el modelo **conceptual**, el modelo **lógico** o relacional y el modelo **físico**.

1. Diseño conceptual: diseño de alto nivel de la estructura de base de datos. Este diseño será independientemente del *Sistema Gestor de Base de Datos* (SGBD) con el que se vaya a implementar. Su objetivo será realizar una descripción de la información que se almacenará en la base de datos, identificando entidades de negocio, sus atributos y las relaciones existentes entre ellas.
2. Diseño lógico: diseño de la estructura de la base de datos conforme al modelo relacional a partir del modelo conceptual: definición de tablas, campos, relaciones existentes entre ellas, normalización de esta estructura, etc.
3. Diseño físico. implementación de la base de datos diseñada en la fase anterior. Para ello se detallarán las estructuras de almacenamiento y los métodos más eficientes para acceder a los datos en el lenguaje de descripción de datos propio del motor de la base de datos (en este caso PostgreSQL).

En este apartado se expone un resumen del modelo conceptual y del modelo relacional de la base de datos a construir. El análisis completo se podrá consultar en el Anexo A2.2.

4.1. Modelo de datos conceptual

El modelo de datos a obtener será una simplificación del problema de gestión de emergencias que se pretende abordar al que se llegará a través de un proceso de abstracción de la información y de los conceptos implicados en dicho problema.

Para definir las variables utilizadas en la base de datos y el tipo de valor que van a contener se hará uso de diagramas **Entidad / Relación** (E/R), donde se reflejará la información que contiene cada entidad (conceptos principales del modelo) y las relaciones existentes entre entidades.

Para facilitar la comprensión del diagrama E/R se han agrupado las distintas relaciones y entidades implicadas en tres apartados: las relacionadas con los diseminados, las relacionadas con las rutas de acceso, y por último las relacionadas con los tramos que componen las rutas.

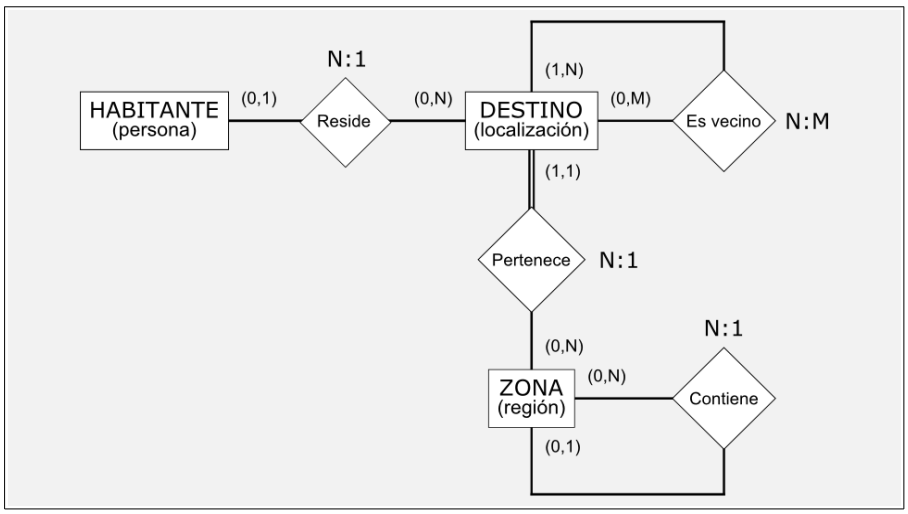

Imagen 4: Diagrama E/R relacionado con los diseminados

4.1.1.- E / R relacionadas con la información de los diseminados

Como se aprecia en la Imagen 4, existen tres entidades relacionadas para el almacenamiento de la información referida a los diseminados y a sus habitantes.

1. Para la información referida al diseminado:
   - La entidad **Destino** contendrá la información referida a su localización.
   - La entidad **Zona** añadirá más especificaciones entorno a su ubicación. Información relacionada con barrios, poblaciones, municipios y agrupaciones de diseminados.
2. Para la información referida a sus habitantes:
   - La entidad **Habitante** contendrá la información referida a los habitantes del diseminado.
   - La relación de **vecindad** de la entidad **Destino** contendrá la información referida a los/las vecinos/as de dichos habitantes.

4.1.1.1. Relación Pertenece: Destino pertenece a Zona

Los destinos son localizaciones de diseminados, y pertenecen a una única zona. Las zonas son regiones que representan barrios, poblaciones, municipios o agrupaciones de diseminados. En una zona puede no contener ningún destino (diseminado), al ser una zona despoblada.

4.1.1.2. Relación Contiene: Zona contiene a Zona

Las zonas pueden contener a otras zonas. En cambio, las subzonas (zonas que son parte de una zona) no pueden estar dentro de varias zonas, es decir, cada subzona sólo estará dentro de una zona.

4.1.1.3. Relación Reside: Habitante reside a Destino

Los habitantes de los diseminados (edificio), y sólo podrán residir en una vivienda de acuerdo al modelo. Esto se debe a que los datos referidos a habitantes provienen del empadronamiento municipal. Dado que la residencia de los/las habitantes varía con el tiempo, puede que existan habitantes (en la base de datos) que actualmente no estén vinculados a ningún **Destino** (de la *Comarca de Baztan*).

De acuerdo al modelo planteado, los diseminados podrán estar habitados por varias personas, pudiendo existir diseminados que actualmente no tengan ningún habitante residiendo en ella o que por su tipología no contendrán ningún habitante: es el caso de los edificios de tipo ermita, cementerio, iglesia, escuela y puente, para los que no hay habitantes empadronados.

4.1.1.4. Relación Es Vecino: Destino es vecino de Destino

Los destinos pueden tener destinos vecinos. Asimismo, estos destinos vecinos (destinos que tienen un vecino/a) pueden tener varios destinos vecinos.

4.1.2.- E / R relacionadas con la información de las rutas de acceso

Como se aprecia en la Imagen 5 y 6, existen siete entidades relacionadas en nuestra modelo de datos en referencia al almacenamiento de la información relacionada con las rutas de acceso.

- La entidad **Ruta** contendrá la información referida a la ruta de acceso.

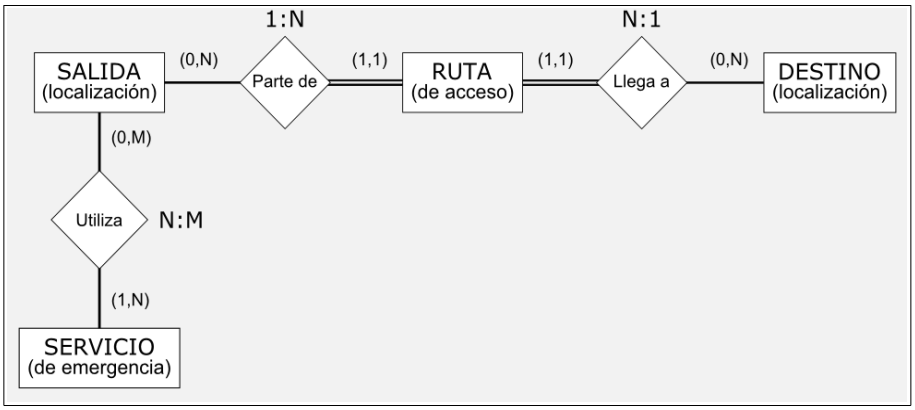

Imagen 5: Primer diagrama E/R relacionado con las rutas de acceso

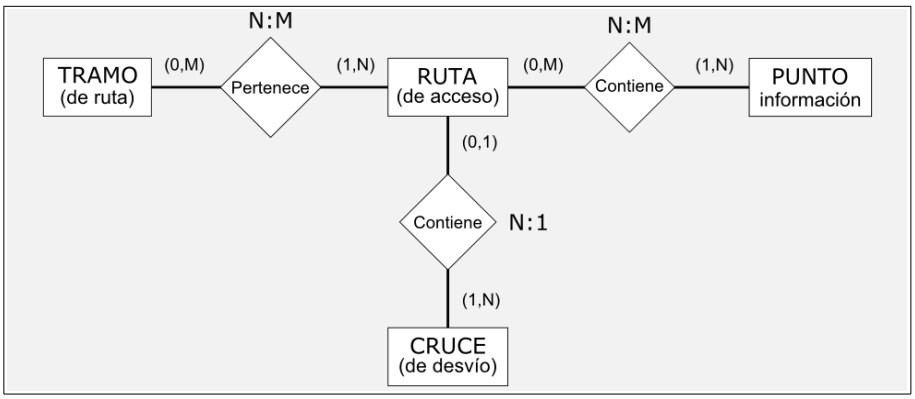

Imagen 6: Segundo diagrama E/R relacionado con las rutas de acceso

- En este sentido, la entidad **Destino** añadirá a este conjunto de información datos relacionados con el punto de destino de dicha ruta. En cambio, la entidad **Salida** añadirá especificaciones entorno al punto de salida.
- La entidad **Servicio** servirá para añadir información referida al equipo de emergencia que utiliza estos puntos de salida. Por lo tanto, existirán rutas personalizadas para cada equipo de emergencia.
- La entidad **Tramo** pretende incorporar información espacial y temporal de la composición de las rutas.
- La entidad **Cruce** pretende añadir información referida a cruces. Representan los desvíos utilizados en el transcurso de la ruta. La razón de incorporar esta entidad será simplemente poner a disposición del usuario la información espacial de este cruce. La obtención de este dato proporcionará el punto de unión para acceder desde cualquier lugar de la carretera a los tramos rurales. De esta forma, un vehículo de emergencia que no esté en el punto de salida podrá acceder a la ruta de acceso sin ningún problema.

- La entidad **Punto de Información** contendrá información sobre advertencias de dificultades de tránsito acaecidas en el recorrido de la ruta.

4.1.2.1. Relación Llega: Ruta llega a Destino

Las rutas representan el recorrido donde se accede a los destinos, y cada ruta llegará a un solo destino. Los destinos son localizaciones de diseminados, y habrá destinos que todavía no contengan asociada ninguna ruta de acceso. Pero en la mayoría de los casos a los destinos llegarán varias rutas, al disponer cada servicio de emergencia una ruta personalizada para cada destino.

4.1.2.2. Relación Parte: Ruta parte de Salida

Las rutas parten de una sola salida para acceder a un solo destino. Estas salidas son localizaciones de equipamientos orientados a los servicios de emergencia, con la finalidad de que estos servicios obtengan rutas que partan desde sus propios equipamientos. Por lo tanto, desde cada salida podrán partir varias rutas de acceso. Pero habrá casos donde no partirá ninguna ruta asociada. Será el caso de salidas que están ubicados fuera de la *Comarca de Baztan* (por ejemplo, el *Parque Central de Bomberos* de Cordovilla).

En estos casos, si algún vehículo de estas salidas tiene que desplazarse a Baztan para asistir una emergencia será interesante la utilización de los cruces de desvío (definidos como puntos de acceso a tramos rurales), y poder así utilizar las ruta de acceso a ese incidente.

4.1.2.3. Relación Utiliza: Servicio utiliza Salida

Los servicios serán equipos de emergencia que operan en la *Comunidad Foral de Navarra*, y podrán tener o no salidas (equipamientos o infraestructuras) en la *Comarca de Baztan*. Por la complejidad operativa de los servicios de emergencia un servicio podrá tener varias salidas (equipamientos) en esta región de trabajo. Cada salida estará utilizada por al menos un servicio de emergencia, y habrá salidas donde compartan instalaciones varios servicios.

4.1.2.4. Relación Pertenece: Tramo pertenece a Ruta

Los tramos son partes del itinerario a recorrer y conforman las rutas de acceso. Por lo tanto, un tramo puede pertenecer a una o varias ruta de acceso. Pero podrán existir tramos (almacenados en la base de datos) que no forman parte de ninguna ruta de acceso. Por ejemplo al actualizar los recorridos de las rutas y existir cambios en el acceso de algún diseminado. Cada ruta tendrá por lo menos un tramo, y también podrá tener varios tramos.

Al crear la posibilidad de que un tramo pueda pertenecer a varios rutas se evita tramos repetidos en la base de datos. De esta manera se reducen los datos redundantes que existían en el documento KML.

4.1.2.5. Relación Contiene: Ruta contiene Cruce

Los tramos estarán delimitados por cruces de desvío. Pero en este caso, el cruce se referirá solamente al desvío donde se deja de transitar por el tramo de carretera y se accede al tra-

mo rural. Por lo tanto, las rutas pueden contener sólo cruce. Pero habrá casos donde no contenga ningún cruce. Éstos serán los casos donde no existe ningún tramo rural en la ruta de acceso.

En general, los cruces pertenecerán a una ruta. Pero también podrá pertenecer a varias rutas. Esta posibilidad evita la existencia de cruces repetidas en nuestra base de datos.

4.1.2.6. Relación Contiene: Tramo contiene Punto de Información

Los puntos de información son puntos de advertencia existentes en las rutas. Por lo tanto, las rutas pueden contener (o no) una o varios puntos de información.

Los puntos de información también podrán pertenecer a varias rutas. Esta posibilidad surge al querer evitar puntos de información repetidas en la base de datos.

4.1.3.- E / R relacionadas con la información de los tramos de rutas

Como se aprecia en la Imagen 7, existen tres entidades relacionadas en el modelo de datos para el almacenamiento de la información relacionada con los tramos de rutas.

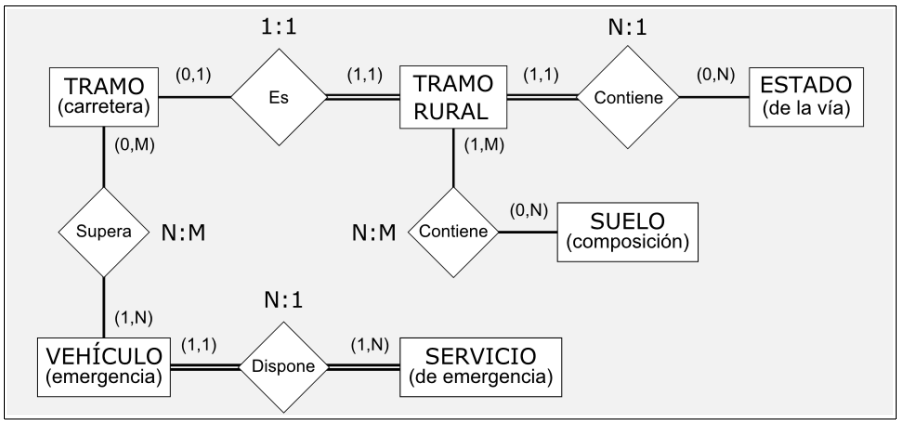

Imagen 7: Diagrama E/R relacionado con los tramos de rutas

- La entidad **Tramo** contendrá la información de cada uno de los tramos que componen las rutas de acceso.
- La entidad **Tramo Rural** será un tipo específico de tramo para las partes de una ruta que transcurren por caminos rurales.
- En relación a los anteriores, la entidad **Suelo** contendrá los datos de las características de composición de los tramos rurales, y la entidad **Estado** informará sobre la transitabilidad de dichos tramos rurales.
- La entidad **Vehículo** permitirá especificar las limitaciones de movilidad de cada vehículo, lo cuál es de los vehículos disponibles en los servicios de emergencia podrán transitar por cada tramo rural.
- La entidad **Servicio** permitirá describir el equipo de emergencia al que pertenece cada vehículo.

4.1.3.1. Relación subclase: Tramo es Tramo Rural y Tramo Carretera

Los tramos rurales y los tramos de carretera son subclases de la entidad Tramo. Al disponer solamente de información sobre tramos rurales, sólo se ha creado la entidad Tramo Rural.

4.1.3.2. Relación Contiene: Tramo rurales contiene Suelo

Los tramos rurales son los tramos más difíciles de transitar. Cada tramo rural contendrá una o varias composiciones del suelo, ya que los tramos rurales no están totalmente delimitado por un cambio de características de la calzada.

Una misma composición del suelo puede estar contenida en varios tramos rurales, y podrá existir alguna característica que no se aplique a ningún tramo rural.

4.1.3.3. Relación Contiene: Tramo rurales contiene Estado

Cada tramo rural contendrá información referida al estado de transitabilidad de la vía, y podrá contener la fecha de esta categorización. Una mismo estado pueden estar referido desde varios tramos, y podrán existir estados sin referir desde ningún tramo.

4.1.3.4. Relación Supera: Vehículo supera Tramo

De acuerdo al modelo, los vehículos de emergencias podrán superar al menos un tramo, y algunos de ellos podrán superar varios tramos. Podrán existir tramos que no sean superables por ningún vehículo de emergencia. También habrá tramos que sean superables por varios vehículos.

4.1.3.5. Relación Dispone: Servicio dispone Vehículo

Los servicios de emergencia dispondrán al menos de un vehículo, y podrán disponer de varios vehículos. Cada vehículo estará asociado a un solo servicio de emergencias.

4.1.4.- Atributos de las Entidades y Relaciones

Para completar el modelado de la base de datos, se deberán definir los atributos asociados a cada una de las entidades descritas en los apartados anteriores. También se especificará el **dominio** de cada atributo (valores posibles para cada uno de ellos), y se mencionará cuáles de estos atributos (o composición de atributos) podrán ser **clave primaria** de cada entidad. De esta forma quedará definida toda la información que debe contener el sistema. Este análisis se puede consultar íntegramente en el Anexo A2.2.

4.2. Modelo de datos relacional

El modelo relacional, necesario para implementación física de la Base de datos relacional, se obtendrá a partir del modelo conceptual. En este proceso se trasladarán las entidades y atributos (definidos en el modelo conceptual) a tablas y columnas (del modelo lógico del sistema). Adicionalmente, se procederá a establecer restricciones que deberán cumplir los valores del conjunto de datos almacenados en los campos de la base de datos (para poder así

mantener la integridad de los datos). Por último, se utilizará toda la información generada en el proceso para plasmar en un diagrama el modelo relacional resultante.

El análisis completo se puede consultar en el Anexo A2.2, apartado 2 (modelo relacional).

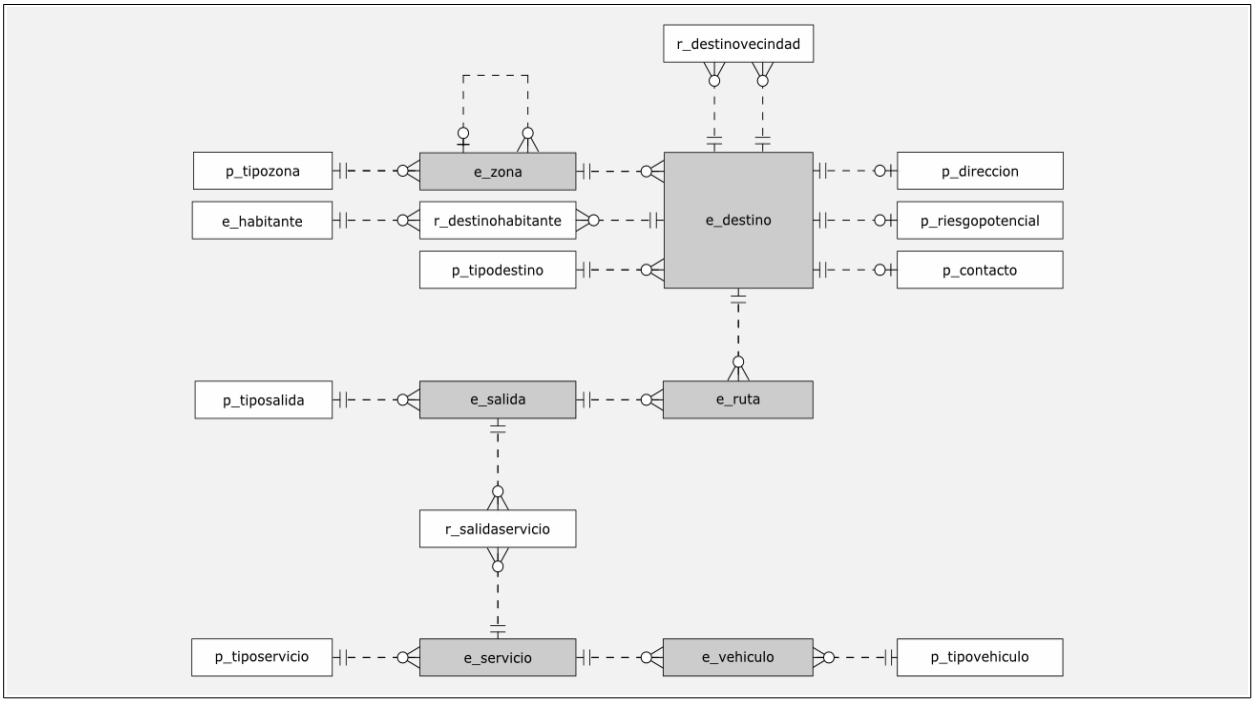

Imagen 8: Diagrama general 1 del modelo relacional

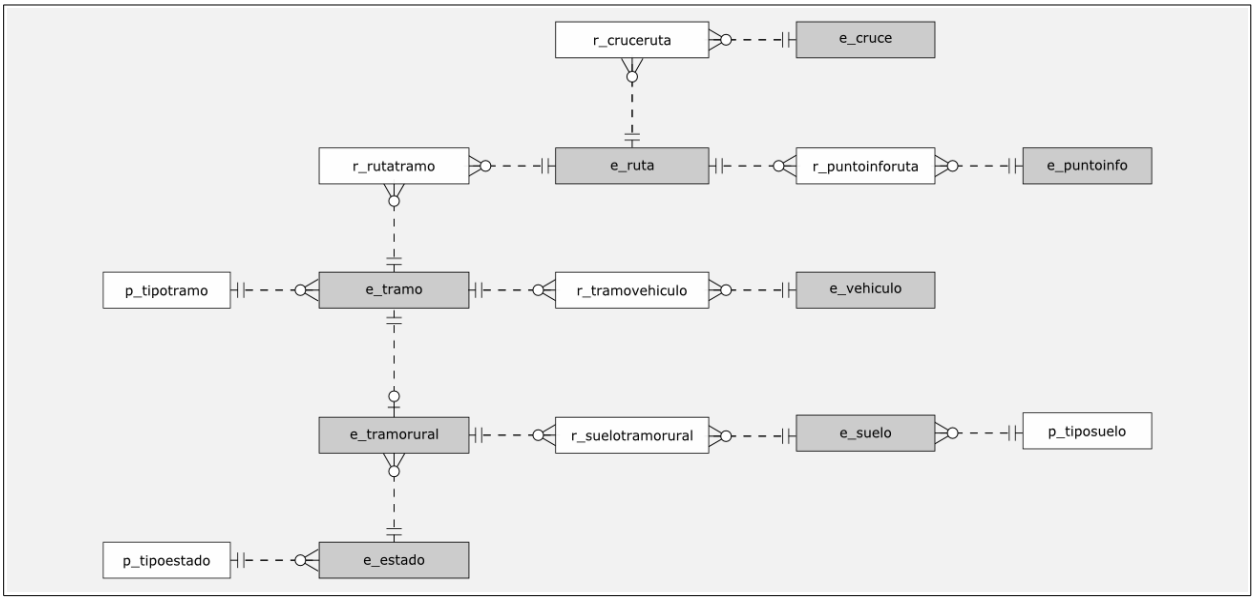

Imagen 9: Diagrama general 2 del modelo relacional

4.2.1. Tablas del modelo relacional

En este apartado se enumerarán las tablas que componen el diseño del modelo relacional que se muestran en la imagen 8 y 9. El análisis completo se puede consultar en el apartado 2.1.18 del Anexo A2.2.

Las tablas del modelos se pueden clasificar en los siguientes grupos: tablas principales (entidades del modelo conceptual), tablas con datos complementarios, tablas de relación entre las anteriores y las tablas de catálogo (valores maestros).

Nota: las claves primarias están en ***color azul*** y subrayadas, y las claves foráneas están en ***negrita***.

1. Tablas principales:
   - Entidad **zona**: tabla e_zona.

*e_zona (**idzona**, nombre, **idtipozona**, **idzonapadre**)*

   - Entidad **destino**: tabla e_destino.

*e_destino (**iddestino**, **idzona**, nombre, **idtipodestino**, fotografia, geom, habitado, observacion)*

   - Entidad **ruta**: tabla e_ruta.

*e_ruta (**idruta**, **idsalida**, **iddestino**, descripcion, numerotramos)*

   - Entidad **vehiculo**: tabla e_vehiculo.

*e_vehiculo(**idvehiculo**, **idservicio**, **idtipovehiculo**, limitacion)*

   - Entidad **salida**: tabla e_salida.

*e_salida (**idsalida**, nombre, **idtiposalida**, geom)*

   - Entidad **servicio**: tabla e_servicio.

*e_servicio (**idservicio**, **idtiposervicio**, contacto)*

   - Entidad **tramo**: tabla e_tramo.

*e_tramo (**idtramo**, **idtipotramo**, nombre, geom, longitud, tiempo)*

   - Entidad **tramo rural**: tabla e_tramorural.

*e_tramorural (**idtramo**, **idestado**, fecha, curvacerrada, pendientealta, advertencia)*

   - Entidad **habitante**: tabla e_habitante.

*e_habitante (**idhabitante**, nombre, apellido1, apellido2, nif, fechanacimiento, edad, contacto)*

- Entidad **suelo**: tabla e_suelo.

> *e_suelo (**idsuelo**, **idtiposuelo**, caracteristica)*

- Entidad **cruce**: tabla e_cruce.

> *e_cruce (**idcruce**, observacion, geom)*

- Entidad **puntoinfo**: tabla e_puntoinfo.

> *e_puntoinfo (**idpuntoinfo**, nombre, descripcion, geom)*

- Entidad **estado**: tabla e_estado.

> *e_estado (**idestado**, **idtipoestado**, caracteristica)*

A modo de ejemplo se detalla la estructura de las tabla e_estado:

La clave primaria de la tabla *e_estado* será la columna *idestado*. La columna *idtipoestado* es una clave foránea referida a la columna *idtipoestado* de la tabla *p_tipoestado*. Esta columna no admite valores nulos.

2. tablas con datos complementarios:
   - Entidades **dirección**, **riesgo potencial** y **contacto**: tablas p_direccion, p_riesgopotencial y p_contacto.

> *p_direccion (**iddestino**, idportal, provincia, municipio, cp, codmun, idpoblacion, nombrebarrio, nombrecalle, idenvia, numeroportal, letraportal, nombrepoligono, nombreparcela)*
>
> *p_riesgopotencial (**iddestino**, idinventario, depositogas, depositogasolina)*
>
> *p_contacto (**iddestino**, idpropietario, contacto1, contacto2)*

   - Entidades **historial clínico** y **ficha policial**: tablas p_historialclinico y p_fichapolicial.

> *p_historialclinico (**idhabitante**, historialclinico, minusvalia, observacion)*
>
> *p_fichapolicial (**idhabitante**, fichapolicial, antecedente, observacion)*

3. Tablas de relación:
   - Relación **destino – vecindad**: tabla r_destinovecindad.

> *r_destinovecindad (**iddestinovecindad**, **iddestino**, **idvecindad**, distanciavecindad)*

- Relación **salida – servicio**: tabla r_salidaservicio.

*r_salidaservicio (**idsalidaservicio**, **idsalida**, **idservicio**)*

- Relación **ruta – tramo**: tabla r_rutatramo.

*r_rutatramo (**idrutatramo**, **idruta**, **idtramo**, orden)*

- Relación **punto info – ruta**: tabla r_puntoinforuta.

*r_puntoinforuta (**idpuntoinforuta**, **idruta**, **idpuntoinfo**)*

- Relación **tramo – vehiculo**: tabla r_tramovehiculo.

*r_tramovehiculo (**idtramovehiculo**, **idtramo**, **idvehiculo**)*

- Relación **suelo – tramo rural**: tabla r_suelotramorural.

*r_suelotramorural (**idsuelotramorural**, **idsuelo**, **idtramo**)*

- Relación **destino – habitante**: tabla r_destinohabitante.

*r_destinohabitante (**iddestinohabitante**, **iddestino**, **idhabitante**)*

- Relación **cruce – ruta**: tabla cruceruta.

*r_cruceruta (**idcruceruta**, **idruta**, **idcruce**)*

A modo de ejemplo se detalla la estructura de las tabla r_cruceruta:

La clave primaria de la tabla *r_cruceruta* será la columna *idcruceruta*. La columna *idcruce* será una clave foránea referida a la columna *idcruce* de la tabla *Cruce*. Esta columna no podrá contener valores nulos. Otra clave foránea será la columna *idruta*, y se referirá a la columna *idruta* de la tabla *Ruta*. Esta columna tampoco podrá contener valores nulos.

4. Tablas de catálogo: estas tablas contienen los valores maestros asociados a los dominios de las distintas entidades (valores válidos).

*p_tipozona (**idtipozona**, descripciontipo)* *p_tipodestino (**idtipodestino**, descripciontipo)* *p_tiposalida (**idtiposalida**, descripciontipo)* *p_tiposervicio (**idtiposervicio**, descripciontipo)* *p_tipotramo (**idtipotramo**, descripciontipo)* *p_tiposuelo (**idtiposuelo**, descripciontipo)*

*p_tipoestado (**idtipoestado**, descripciontipo)*

*p_tipovehiculo (**idtipovehiculo**, descripciontipo)*

4.2.2. Dominio de las columnas creadas

El dominio de cada columna (o atributo) podrá ser de tipo texto, numérico, fecha, tiempo, booleano, espacial y valores predefinidos.

- Números de identificación o referencia: de tipo *Numérico entero positivo*.
- Nombres de entidades: de tipo *Texto (hasta 100 caracteres)*.
- Apellidos de habitantes de diseminados: de tipo *Texto (150)*.
- Teléfonos de contacto: de tipo *Texto (15)*.
- Código del documento de identificación: de tipo *Texto (9)*.
- Direcciones URL de fotografías: de tipo *Texto (1000)*.
- Observaciones y descripciones sobre alguna característica: de tipo *Texto (1000)*.
- Distancias y longitud en metros: de tipo *Numérico real positivo*.
- Código postal, edad de una persona, orden, etc.: de tipo *Numérico entero positivo*.
- Fecha de nacimiento, fecha de actualizaciones, etc.: de tipo *Fecha (año, mes, día)*.
- Tiempo de respuesta (isócronas sanitarias, etc.): de tipo *Intervalo* (de tiempo).
- Existencia de riesgos, dificultades de transito, habitantes, etc.: cde tipo *Booleano*.
- Tipos o características reseñables (*descripciontipo*): cada uno de ellos tendrán valores predefinidos específicos.
- Información espacial: de tipo ***Geometry*** *(de subtipo Point y LineString) conforme al SRS 25830*.

5.2.3. Diagrama del modelo relacional

En el diagrama del modelo relacional se pretende reflejar principalmente varios aspectos del diseño:

- Las tablas. Conformarán la unidad de almacenamiento principal de la base de datos.
- Las columnas de las tablas. Representarán los campos de cada uno de los registros almacenados.
- La clave primaria de la tabla. Representará la columna que desempeñará la función de identificador único de cada registro. Podrán estar compuestos por varias columnas. En el diagrama, las columnas que son claves primarias contendrán un “PK” al principio del nombre, y estarán diferenciadas del resto de columnas.
- Las claves foráneas de la tabla. Representa la clave primaria de la tabla que está relacionada. Es decir, se utilizará la clave foránea para expresar la relación existente entre dos tablas. En el diagrama, las columnas que son claves foráneas contendrán un “FK” al prin-

cipio del nombre, y también contendrán un número para diferenciarse de otras claves foráneas que podrían existir en una misma tabla.

- Las restricciones de las columnas. Representan restricciones referidas a los registros almacenados en una columna. Por ejemplo, restricciones de los dominios que contienen las columnas:
  - La restricción de unicidad existirá en todas las columnas que representen claves primarias.
  - La restricción de existencia evitará la aparición de los valores nulos en las columnas. En este caso, en el diagrama las columnas que representan esta restricción se escribirán en negrita. Todas las columnas que representan claves primarias tendrán esta restricción por defecto.
- Las relaciones entre las tablas. Representan el tipo de relación que existe entre dos tablas.

En el Anexo A2.2, se representará un diagrama del modelo relacional con el objetivo de explicar el diseño relacional realizado.

5. Construcción del sistema de información

En este apartado se explicarán las diferentes etapas de desarrollo realizadas para implementar y cargar una instancia de la base de datos conforme al modelo de datos diseñado. De manera preliminar a la implementación se deberá diseñar el proceso de carga que quedará documentado mediante diagramas de flujos de información; posteriormente se definirá un modelo de datos intermedio asociado al proceso de carga (necesidad detectada durante el proceso de carga); y finalmente se definirán las transformaciones realizadas para trasladar los datos desde el modelo intermedio al modelo de explotación.

Asimismo se mencionarán los problemas surgidos en estos procesos, y las recomendaciones planteadas para evitar o minimizar dichos problemas. Por último, se analizará el modelo de explotación obtenido.

5.1. Proceso de carga de datos

El proceso de carga consistirá en la extracción de la información contenida en los ficheros KML suministrados, para posteriormente añadir dichos datos a las tablas de la base de datos. Para realizar esta fase de carga se ha desarrollado una aplicación de tratamiento y carga de datos específica en el lenguaje de programación Python (ver Imagen 10).

Para realizar la extracción de los datos es necesario recorrer toda la estructura del fichero KML, y poder así acceder todos los elementos de datos del mismo que contienen información relevante. Para realizar este proceso correctamente habrá que ajustar la extracción a las diferentes casuísticas de estructuración de la información hallados dentro de los ficheros KML

descritos en el Anexo A2.1. El proceso de carga realizado se representarán con diagramas de flujos de información (ver apartado 5.1.1.).

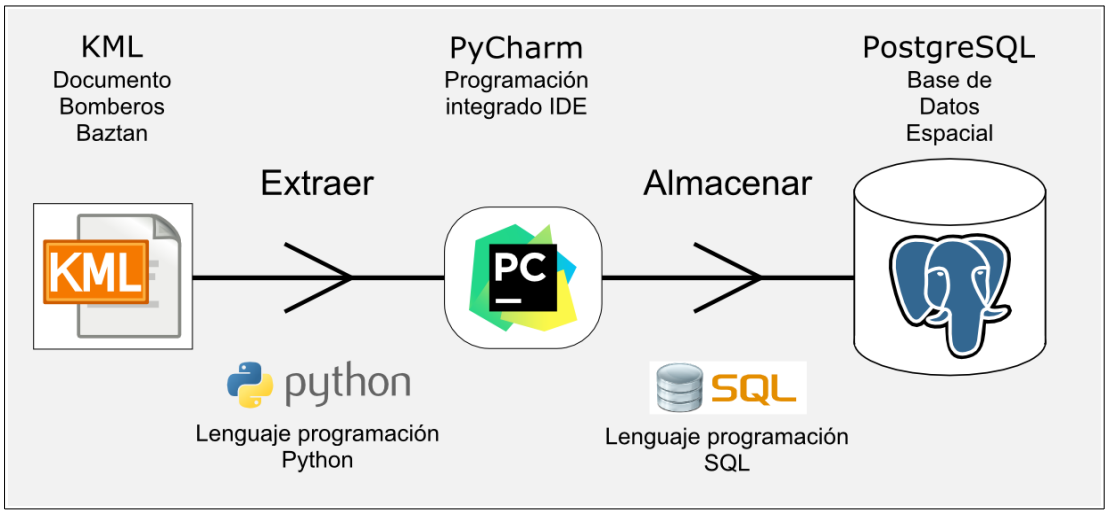

Imagen 10: Arquitectura del proceso de carga de datos

Al realizar este proceso se obtendrá una lista de instrucciones de carga de la información adquirida en la extracción. Dichas instrucciones estarán especificadas en el lenguaje de programación para bases de datos relacionales SQL.

De manera análoga a la generación de estas instrucciones de carga en SQL también se podrán obtener diferentes tipos de productos de salida. En nuestro caso, al realizar el proceso de carga, también se han obtenido varios archivos CSV. De esta forma, utilizando un software de ofimática se ha analizado de forma más cómoda la información extraída, lo que ha permitido validar y completar la información a cargar sobre las propias instrucciones SQL antes de su carga definitiva en PostgreSQL.

5.1.1. Diagrama de flujo

En este apartado se mostrarán los ordinogramas diseñados para el proceso de carga de datos (consultar explicación detallada y ordinogramas de este proceso en el Anexo A.3.1).

Como se aprecia en el diagrama general (ver la Imagen 11) el acceso a la información contenida en el archivo KML se realiza de manera estructurada y progresiva: primero se accede al nodo raíz, luego al sub-elemento **Document**, y por último a los sub-elementos **Folder**.

El procesamiento de los sub-elementos Folder queda detallada en el siguiente diagrama (ver la Imagen 11). En este proceso se analizarán uno por uno cada elemento Folder: primero se obtendrán los datos referidos a una zona y creando un registro para dicha tabla, y a continuación se analizará si el elemento analizado contiene un elemento Placemark, un elemento Document o un elemento Folder. En el caso que contenga uno de estos sub-elementos se procederá a procesar dichos elementos de manera recursiva.

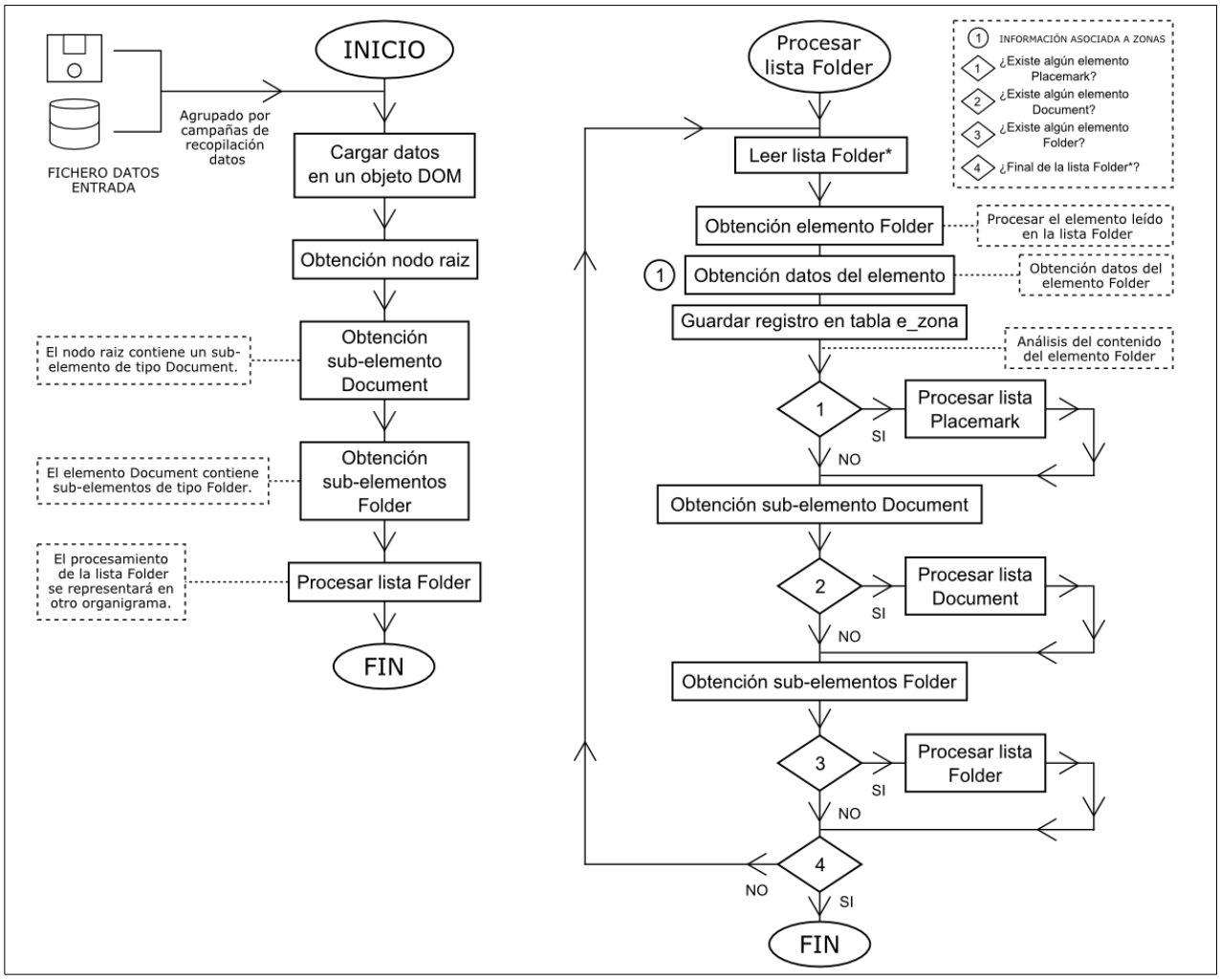

Imagen 11: Diagrama general del proceso de carga y diagrama Procesar Lista Folder

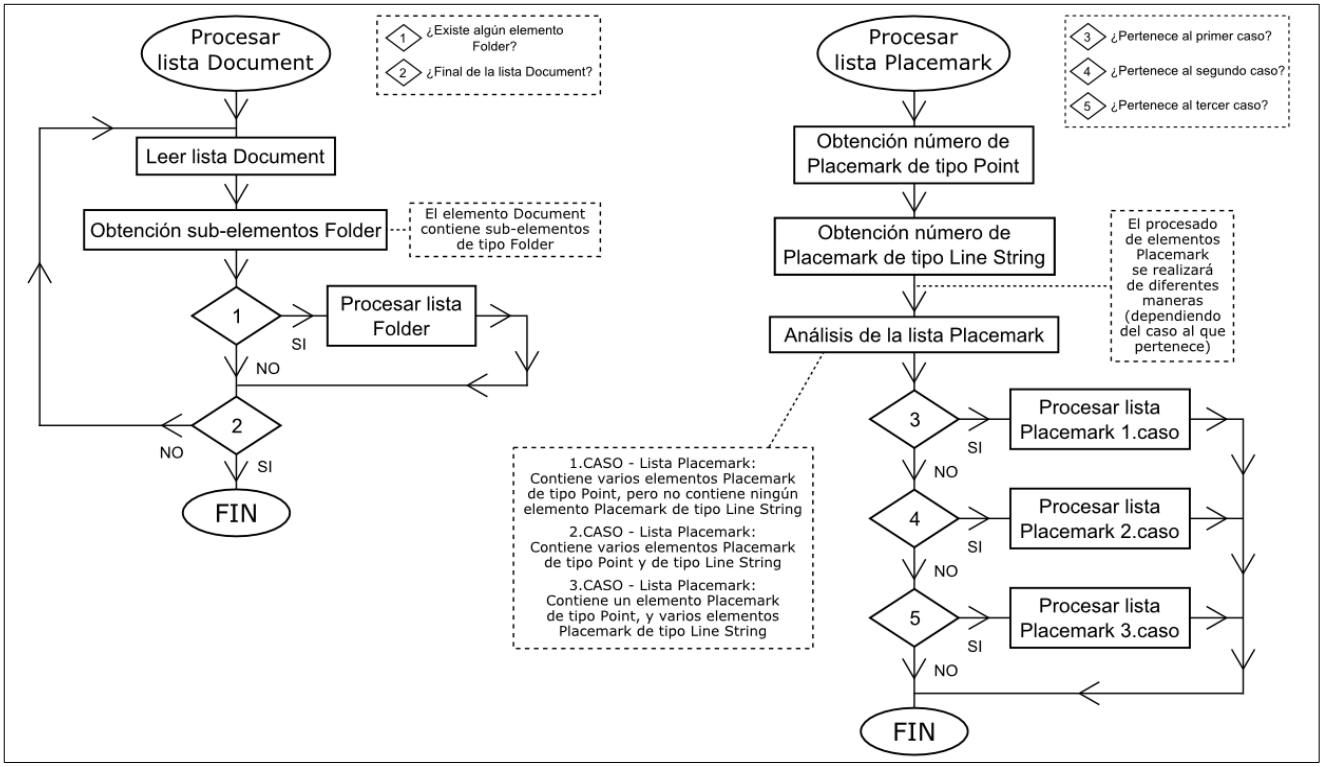

Imagen 12: Diagramas Procesar Lista Document y Procesar Lista Placemark

El procesamiento del elemento Document se detalla en el siguiente diagrama (ver Imagen 12). En este proceso se analiza su contenido y se extraen los sub-elementos Folder que contiene, para su posterior procesamiento.

El procesamiento de elementos Placemark se detalla en el siguiente diagrama (ver Imagen 12). Antes de realizar el procesamiento de los elementos Placemark se analizará la tipología a la que pertenece cada uno de estos elementos: se obtendrán el número de elementos Placemark de tipo Point y de tipo Line String, y dependiendo del número de estos dos tipos de Placemark se determinará el tipo de procesamiento que se le aplicará posteriormente. Se han definido tres casos de procesamiento:

- 1. caso: existencia de varios elementos Placemark de tipo Point. No contiene ningún elemento Placemark de tipo Line String. Ver Imagen 13.
- 2. caso: existencia de varios elementos Placemark de tipo Point y de tipo Line String. Ver Imagen 13.
- 3. caso: existencia de un elemento Placemark de tipo Point, y varios elementos Placemark de tipo Line String. Ver Imagen 14.

En el primer caso de procesamiento de elementos Placemark (ver Imagen 13) todos los Placemark son de tipo Point, por lo tanto se analizará cada elemento Placemark para procesarlos como Puntos Pendientes (puntos o destinos pendientes de ser procesados).

En segundo caso (ver Imagen 13) los elementos Placemark serán de tipo Point y de tipo Line String y se procesarán de diferente manera:

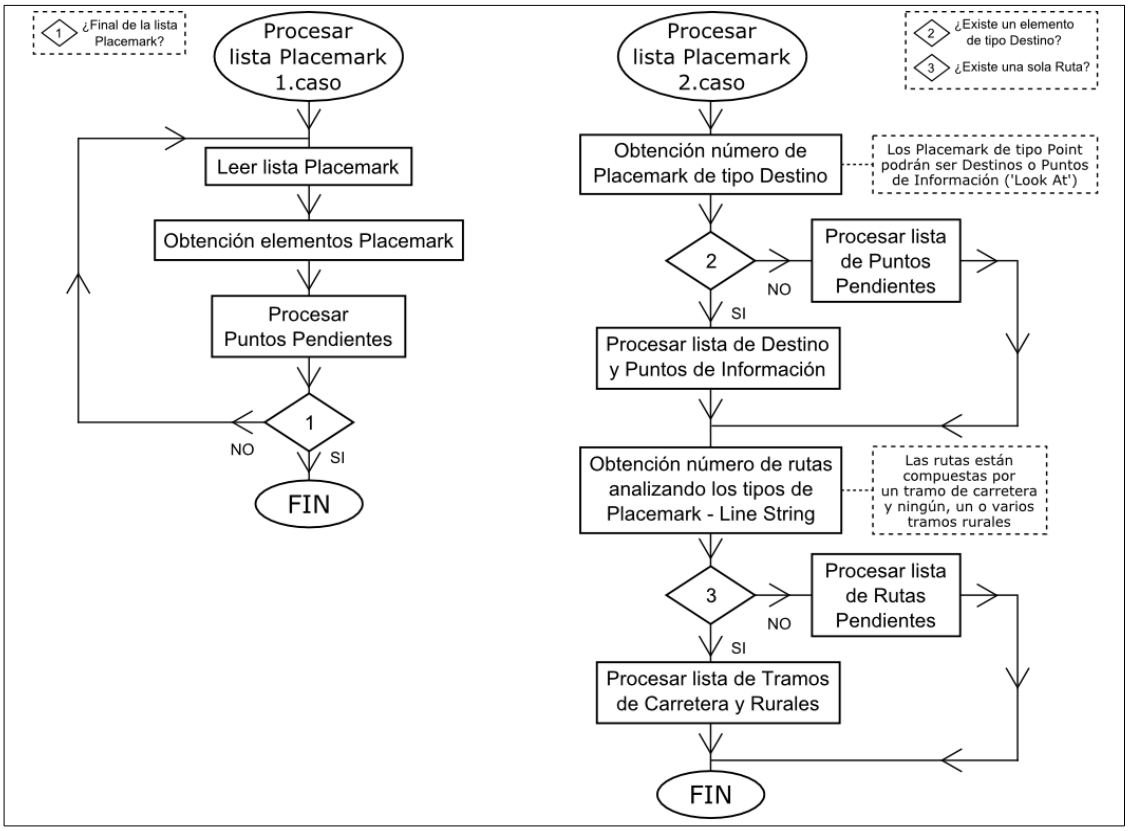

Imagen 13: Diagramas Procesar Lista Placemark 1.caso y 2.caso

- Elementos Placemark de tipo Point: este tipo de elementos Placemark pueden representar Destinos o Puntos de Información. Por tanto, en la lista de elementos Placemark podrán existir dos casos de procesamiento: en el primer caso existirá un Destino y uno o varios Puntos de Información, y en el segundo caso todos los elementos serán Destinos (y representarán Puntos Pendientes). Por lo tanto, el procesamiento a aplicar dependerá del número de Destinos que contenga la lista de elementos Placemark.
- Elementos Placemark de tipo Line String: este tipo de elementos Placemark pueden ser Tramos de Carretera o Tramos Rurales, y también pueden ser Rutas principales. Por tanto, en la lista de elementos Placemark podrá existir dos casos generales de procesamiento: en el primer caso existirá un Tramo Carreteras y ningún, uno o varios Tramos Rurales (conformando todos ellos una Ruta); y en el otro caso, todos los elementos serán Rutas (y representarán Rutas Pendientes). Por lo tanto, el procesamiento a aplicar dependerá del número de Rutas o Tramos de Carretera que contenga la lista de elementos Placemark.

El tercer caso de procesamiento de elementos Placemark se representa en el siguiente diagrama (ver Imagen 14). En este caso primero se procesará el único elemento Placemark de tipo Point que contiene, y posteriormente se procesarán los elementos de tipo Line String. Estos últimos elementos representarán Tramos de Carretera y Tramos Rurales, y se procesarán de manera distinta a la expuesta en los casos anteriores.

El procesamiento de una lista de Puntos Pendientes se representa en el siguiente diagrama (ver Imagen 15). En ella se realiza el procesamiento de una lista de elementos Placemark de tipo Point que representan puntos o destinos pendientes de ser procesados.

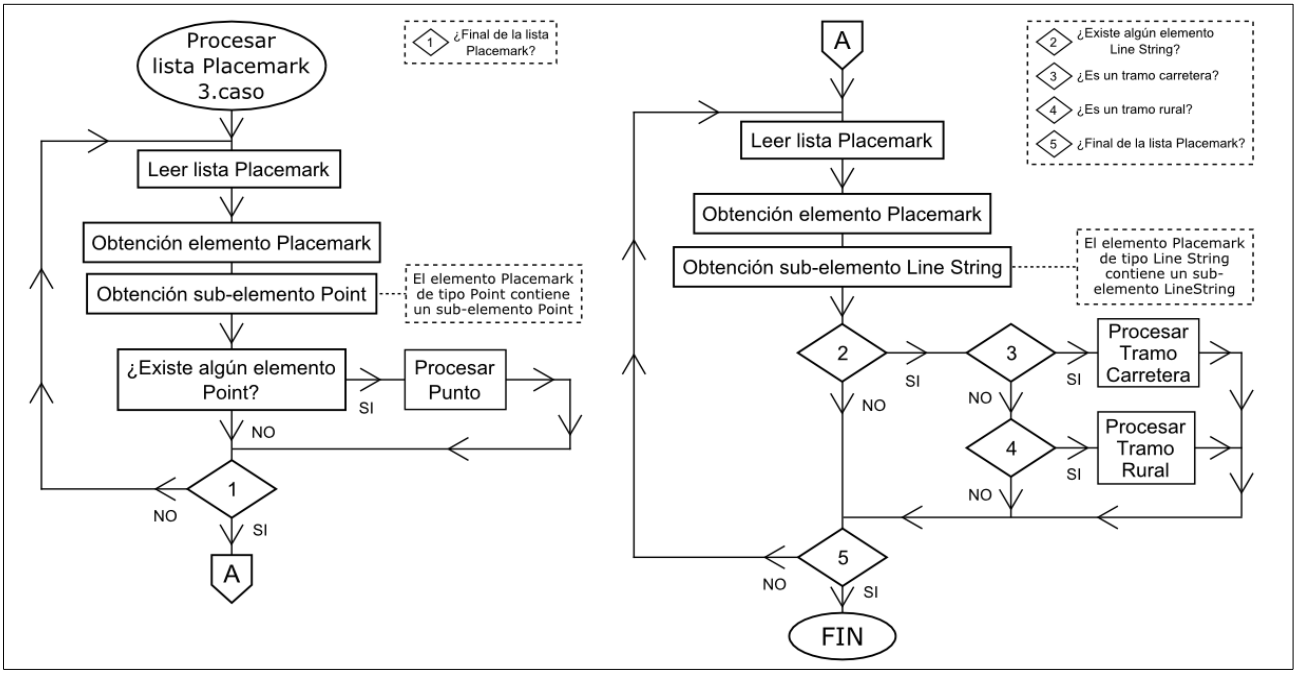

Imagen 14: Diagrama Procesar Lista Placemark 3.caso

El siguiente diagrama (segundo en Imagen 15) representa el procesamiento de una lista de Destinos y Puntos de Información. Como puede verse, los elementos Placemark de tipo Point

pueden representar dos elementos diferentes: Destinos (uno) o Puntos de Información (ninguno, uno o varios). Cada uno de estos elementos se procesará de diferente manera.

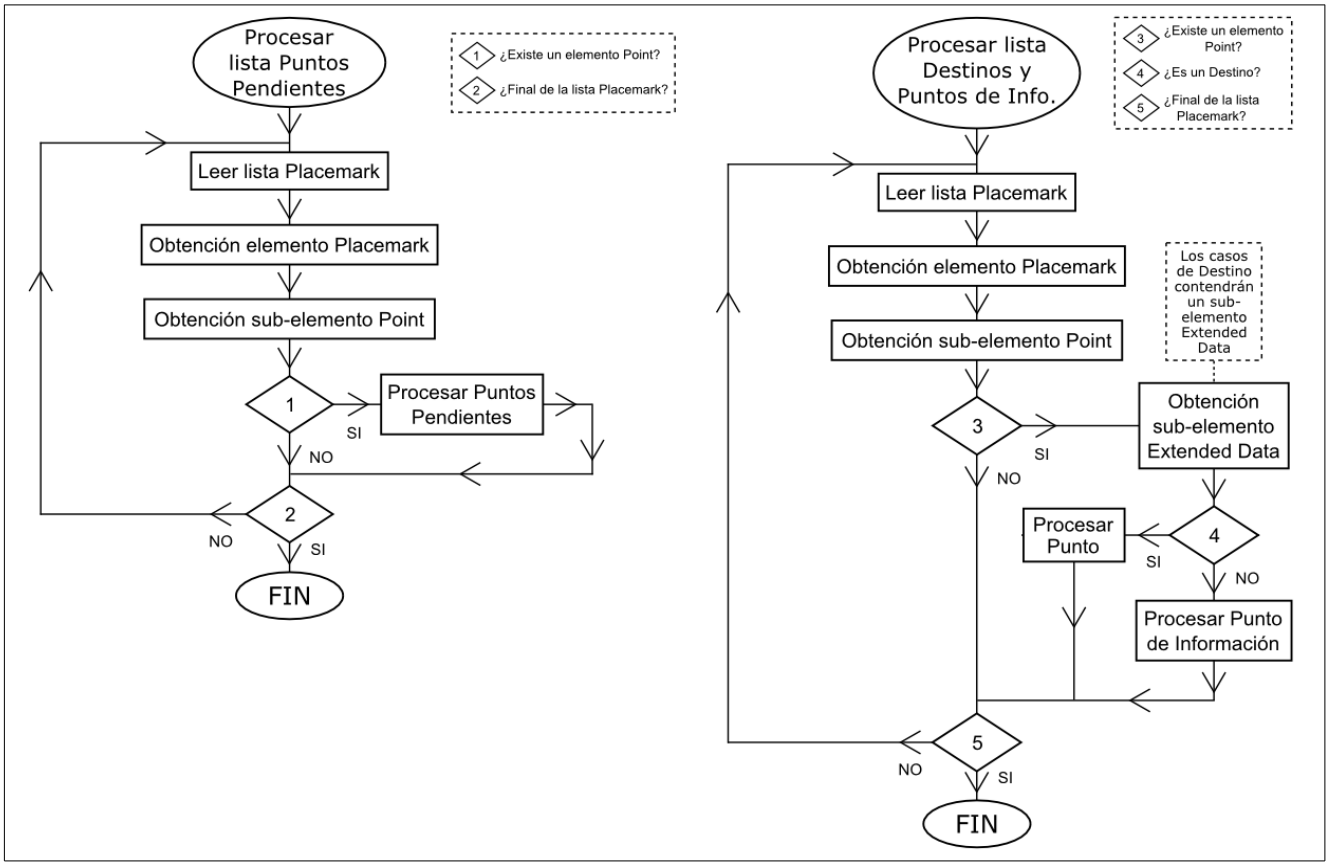

Imagen 15: Diagramas Procesar Lista Puntos Pendientes y Procesar Lista Destinos y Puntos de Info.

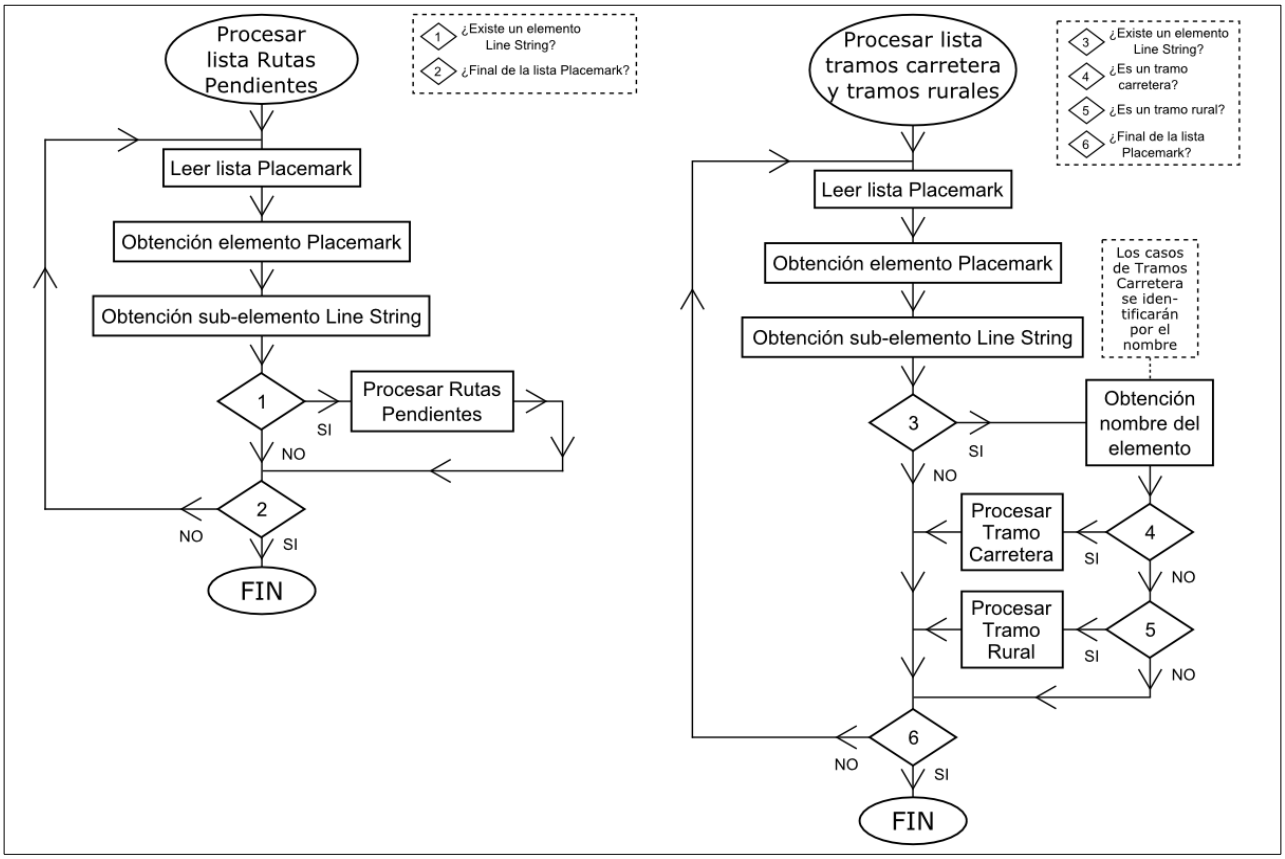

Imagen 16: Diagramas Procesar Lista Rutas Pendientes y Procesar Lista Tramos Carretera y Tramos Rurales

El procesamiento de una lista de elementos Placemark de tipo Line String, se representa en los siguientes dos diagramas (ver Imagen 16). En el primero se representa el procesamiento de rutas pendientes. En el segundo se representa el procesamiento de una lista de tramos de carretera y tramos rurales.

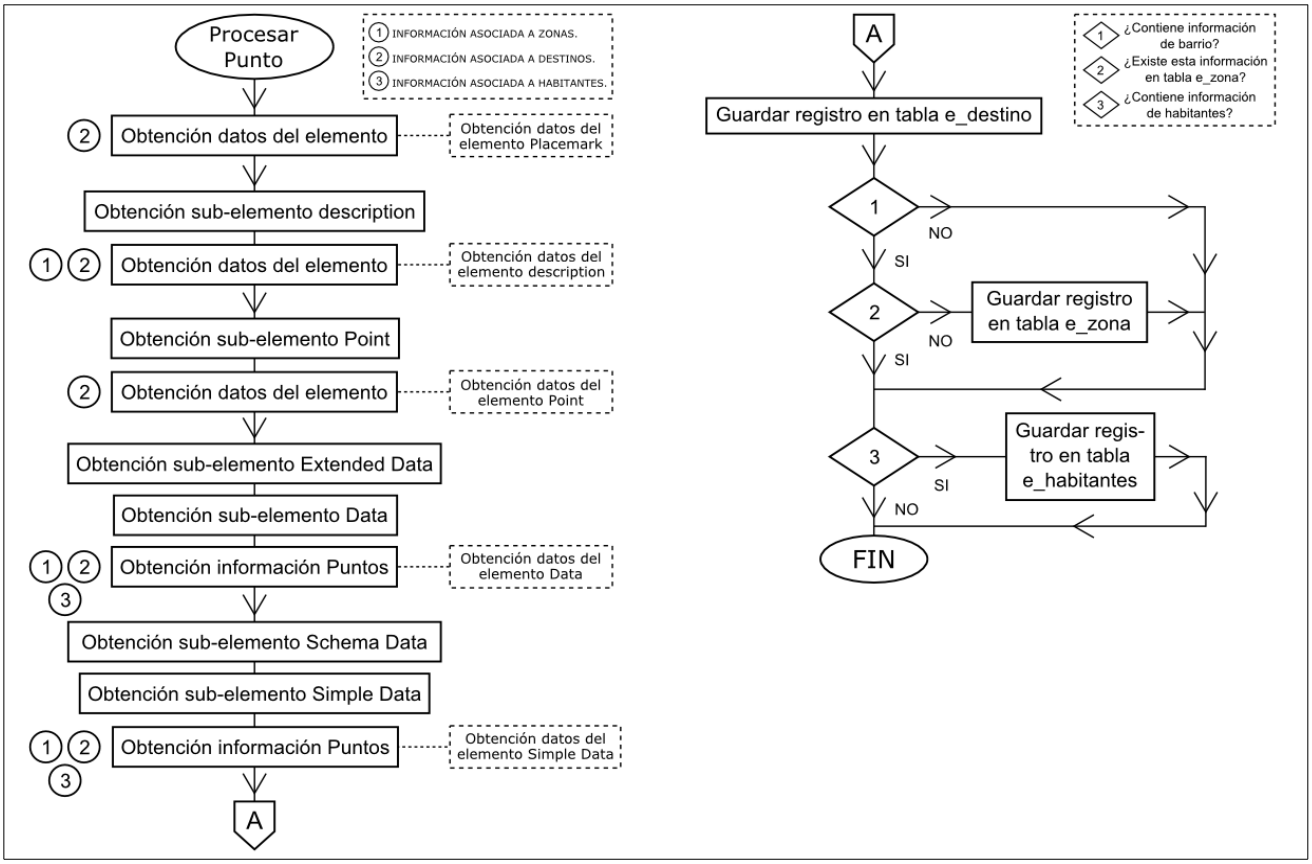

Imagen 17: Diagrama Procesar Punto

El procesamiento de puntos de destinos se representa en el siguiente diagrama (ver Imagen 17). En este proceso se obtendrá toda la información referida a destinos, zonas y habitantes, y se guardarán en las tablas correspondientes.

El procesamiento de puntos pendientes de destinos se representa en el siguiente diagrama (ver Imagen 18). En este proceso se obtendrá toda la información referida a destinos pendientes, zonas y habitantes, y se guardarán en las tablas correspondientes. Al obtener los puntos pendientes existe coincidencia con los puntos de destino, se analizará si contienen diferente información.

El procesamiento de puntos de información se representa en el siguiente diagrama (ver Imagen 19). En este proceso se obtendrá toda la información referida a los puntos de información, y se guardará en la tabla correspondiente.

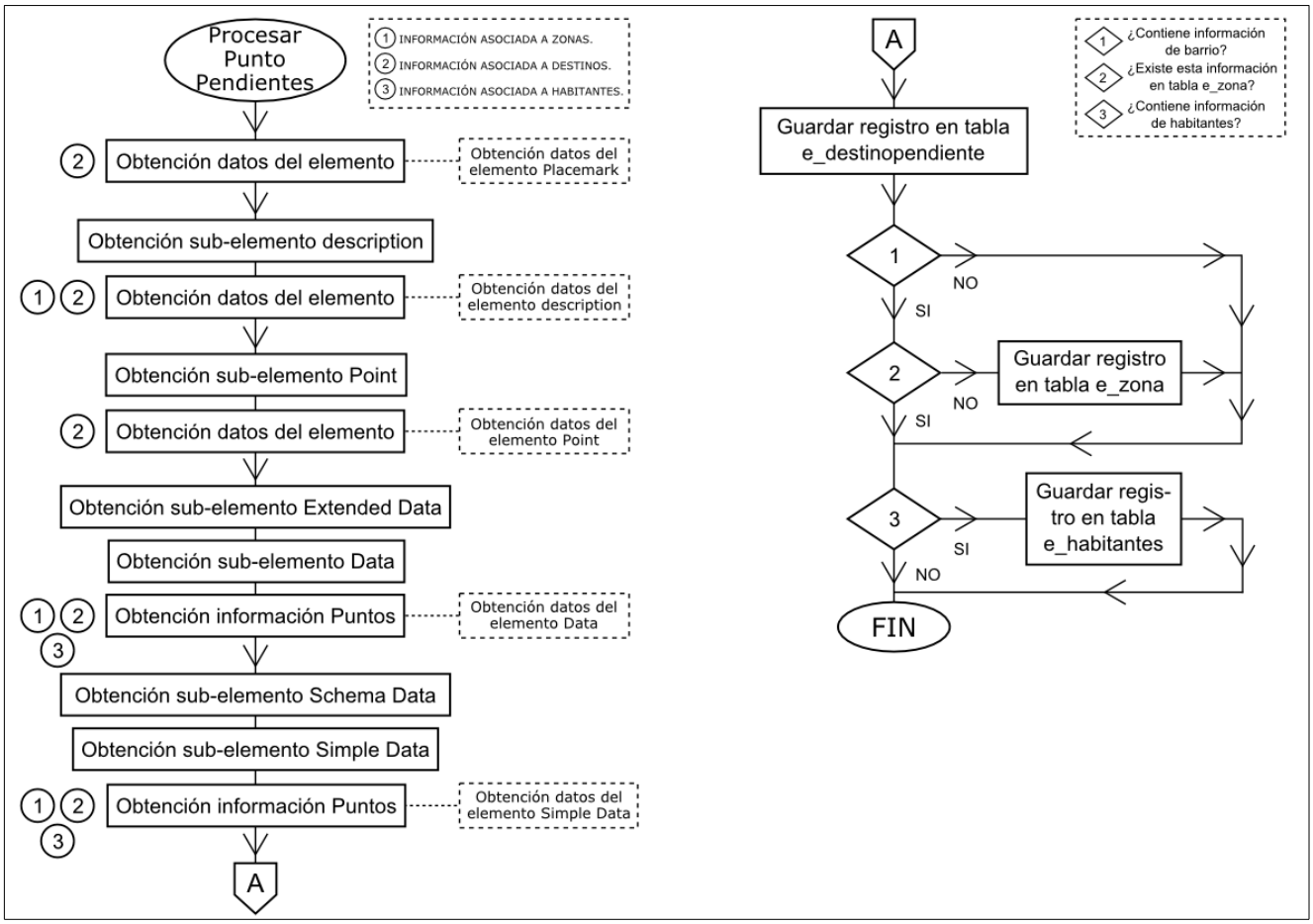

Imagen 18: Diagrama Procesar Punto Pendiente

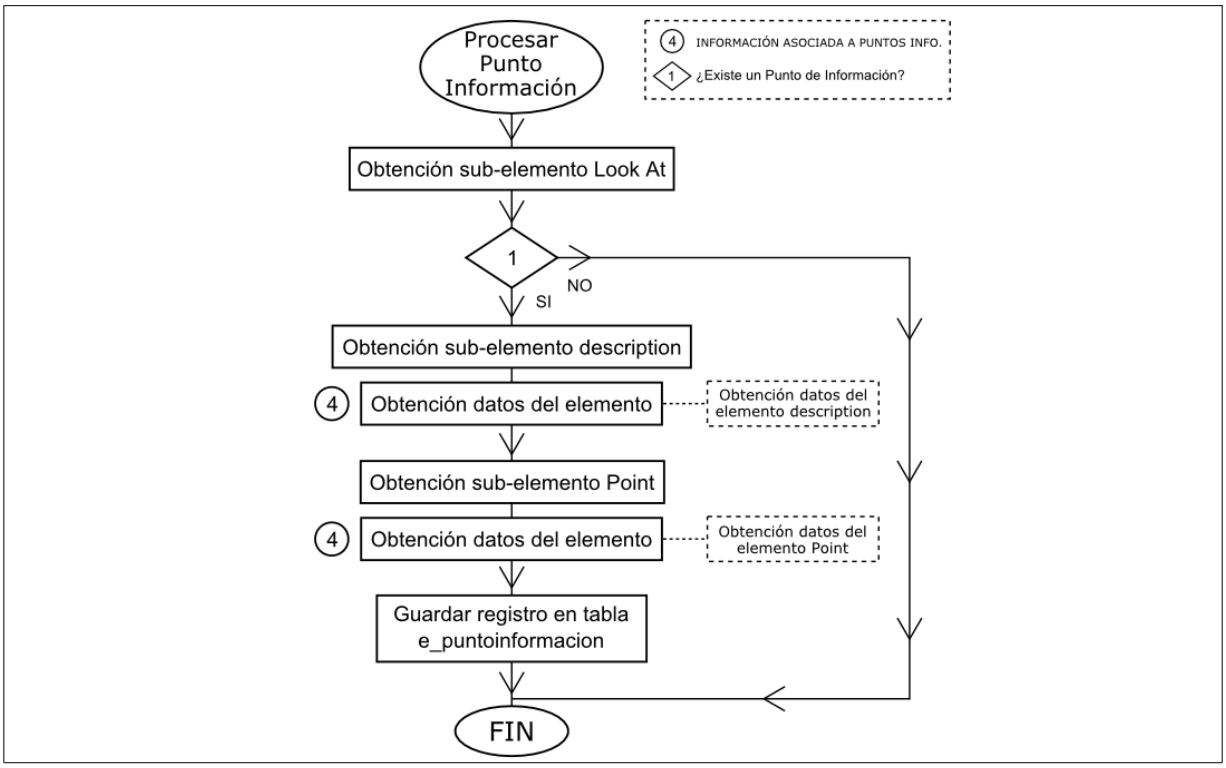

Imagen 19: Diagrama Procesar Punto Información

El procesamiento de un tramo rural se representa en el siguiente diagrama (ver Imagen 20). En este proceso se obtendrá toda la información referida a los tramos rurales, y se guardará

en la tabla correspondiente. En el caso de que los tramos conformen rutas alternativas en vez de las rutas oficiales se procederá de diferente manera, y se guardarán en otra tabla diferente (en las tablas e_tramocarreteraalternativo y e_tramoruralalternativo).

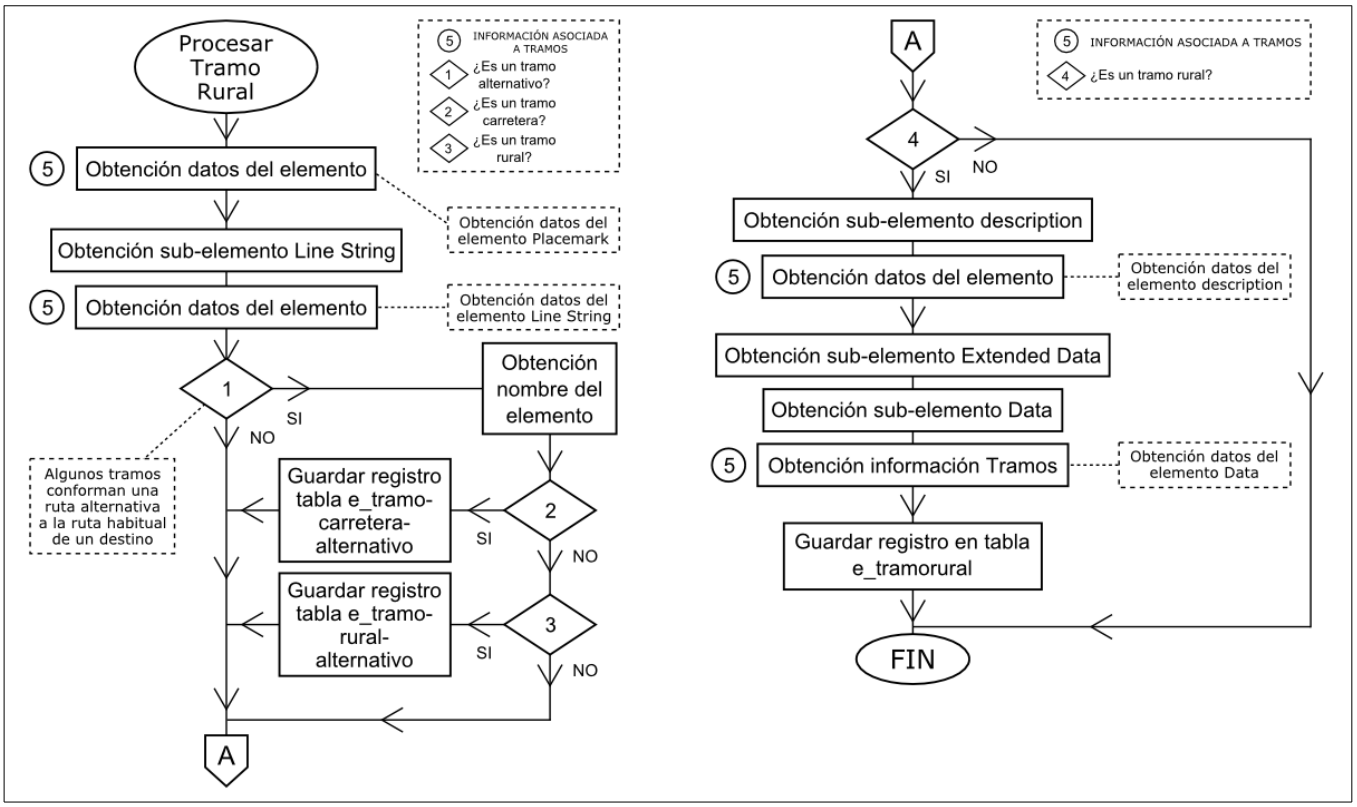

Imagen 20: Diagrama Procesar Tramo Rural

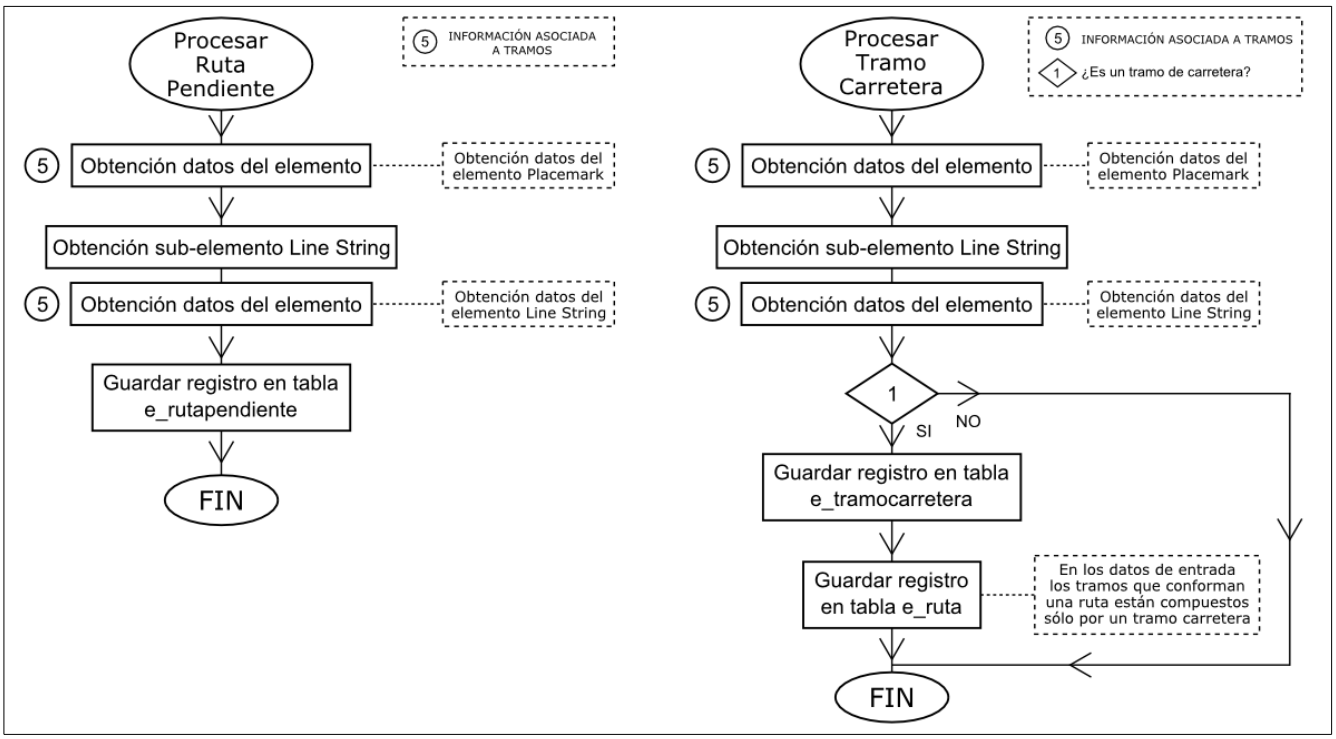

Imagen 21: Diagramas Procesar Ruta Pendiente y Procesar Tramo Carretera

El procesamiento de una ruta pendiente se representa en el siguiente diagrama (ver Imagen 21). En este proceso se obtendrá toda la información referida a las rutas pendientes, y se

guardará en la tabla correspondiente. En la siguiente diagrama (ver Imagen 21) se representa el procesamiento de un tramo de carretera. En este proceso se obtendrá la información referida a los tramos de carretera, y se guardará en la tabla correspondiente.

5.2. Implementación de la estructura obtenida tras la adquisición de datos de entrada

Para implementar la estructura de datos primero se creará la base de datos (BD) utilizando un SGBDR (sistema gestor de base de datos relacional). En este caso, se creará una BD en el servidor local en *PostgreSQL* con extensiones espaciales *PostGIS*.

1. El primer paso a realizar consistirá en la configuración general al crear una instancia nueva de BD:
   - El nombre de la instancia de BD (*Name*). En nuestro caso, *bomberosnavarra.*
   - El usuario propietario (*Owner*). Por ejemplo, *Bomberos de Navarra.*
   - La documentación de la BD (*Comment*). Por ejemplo, "Base de Datos de la herramienta de *Gestión de Accesos a Incidencias* del servicio de emergencias de los Bomberos de Oronoz-Mugaire, del parque comarcal Baztanaldea".

   En esta base de datos se podrán añadir distintos usuarios/as con su nombre, su contraseña de acceso y sus permisos. Con ello se conseguiría que cada usuario pudiese consultar y modificar el contenido de la BD que le estuviese permitido. Estos/estas usuarias podrían ser los diferentes servicios de emergencias que actúan en la misma zona de trabajo y diferentes instituciones públicas implicadas en el mantenimiento de los datos. En el marco de este trabajo, se creará solamente un usuario administrador que gestionará toda la base de datos.
2. A continuación, se creará una nueva extensión (*New Extension*) donde se añadirá la extensión *postgis*. Esto permitirá utilizar en la BD de PostgreSQL los tipos *geometry*, *geography* y *raster spatial* (y su funciones respectivamente).
3. Por último, se crearán un esquema (*New Schema*) donde se agruparán todas las tablas definidas en el modelo relacional. En nuestro caso se llamará *baztanaldea* y en él se crearán todas las tablas del modelo: se indicarán los nombres de las tablas, se crearán columnas (definiendo el tipo de contenido) y se indicarán qué columnas son claves primarias y cuáles son las claves foráneas.

5.3. Diferencias del modelo de datos intermedio y del modelo relacional

A lo largo del desarrollo del proceso de carga se detecta la necesidad de contar con un modelo de datos intermedio que difiere relativamente del modelo diseñado al principio del proyecto. Es decir, hay diferencias entre el modelo de datos diseñado y el modelo de carga (ver Imagen 22). El modelo de explotación tendrá que asemejarse al modelo de datos diseñado, por lo que será necesario identificar los cambios que hay que realizar en el modelo de carga o modelo intermedio.

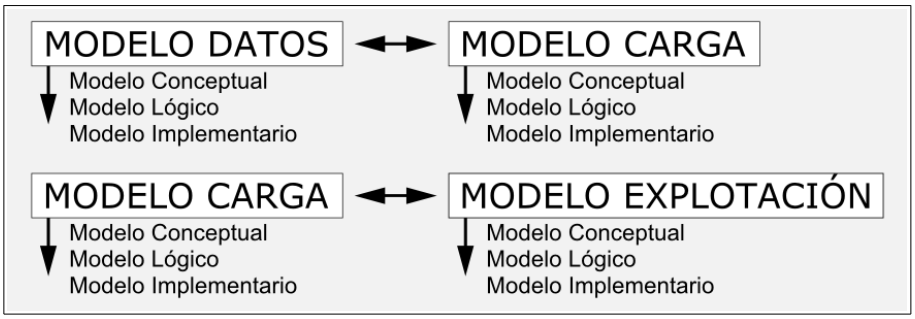

Imagen 22: El modelo de carga o el modelo intermedio de datos

Una vez identificados los cambios entre el modelo intermedio y el modelo de explotación final, se procederá a diseñar y desarrollar los procesos de transformación necesarios entre ambos. Como se aprecia en el esquema de la Imagen 23, este proceso de transformación se realizará utilizando la **herramienta PgAdmin** (donde se realizarán cambios estructurales en la base de datos) y el **software QGIS** (donde se realizarán cambios de la información espacial de la base de datos).

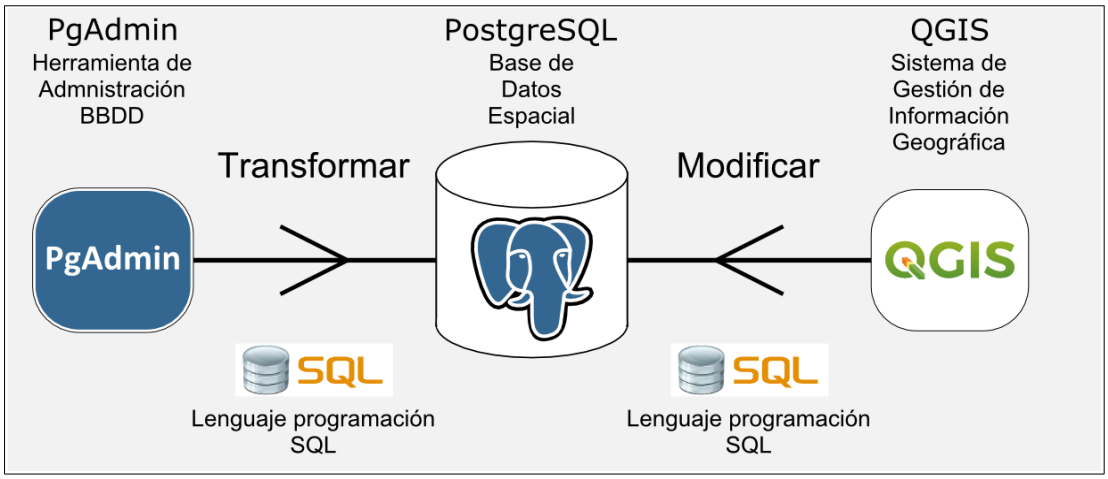

Imagen 23: Arquitectura del proceso de modificación del modelo intermedio

Esta diferencia entre el modelo intermedio y el modelo relacional se ha representado en la Tabla 1. A continuación, se explicará paso a paso la creación de las tablas de la base de datos como las modificaciones realizadas sobre ellas.

Las tablas obtenidas al realizar la carga de datos:

- Tabla **e_zona**, **e_habitante**, **e_puntoinfo**, **r_puntoinforuta**: en el *modelo de explotación* no se ha modificado la estructura de la tabla.
- Tabla **e_destino**: en el *modelo de explotación* se ha reorganizado el contenido de esta tabla, añadiendo tres tablas a la base de datos: p_direccion, p_riesgopotencial y p_contacto.
- Tablas **e_tramocarretera** y **e_tramorural**: en el *modelo de explotación* la mayor parte de la información que contienen estas dos tablas se ha incorporado a la tabla e_tramo, se ha modificado la estructura de la tabla e_tramorural, manteniendo cierta información en ella, y la tabla e_tramocarretera se ha eliminado.

- Tabla **e_ruta**: en el *modelo de explotación* se le ha añadido una columna.

Modelo obtenido en la Carga	Modelo Relacional
e_zona p_tipozona	**e_zona** p_tipozona
e_destino p_tipodestino	**e_destino** p_tipodestino
e_destinopendiente	-
e_habitante - -	**e_habitante** p_historialclinico p_fichapolicial
r_destinohabitante *	**r_destinohabitante**
r_destinopendientehabitante	-
-	**r_destinovecindad**
e_salida * p_tiposalida	**e_salida** p_tiposalida
e_ruta	**e_ruta**
e_rutapendiente	-
e_tramocarretera	-
e_tramorural	**e_tramorural**
p_accv p_estado p_tematico	- - -
- -	**e_tramo** p_tipotramo
r_rutatramo	r_rutatramo
e_tramocarreteraalternativo **e_tramoruralalternativo**	- -
r_rutatramoalternativo	-
e_servicio * p_tiposervicio	**e_servicio** p_tiposervicio
r_salidaservicio	r_salidaservicio
-	**e_cruce**
-	r_curceruta
e_puntoinfo	**e_puntoinfo**
r_puntoinforuta	r_puntoinforuta
- -	**e_suelo** p_tiposuelo
-	r_suelotramorural
- -	**e_estado** p_tipoestado
- -	**e_vehiculo** p_tipovehiculo
-	r_tramovehiculo

Tabla 1: Modelo de carga y modelo relacional

Las tablas temporales creadas en el modelo intermedio:

- Tabla **e_destinopendiente**: en esta tabla se ha almacenado toda la información referida a los destinos denominados “destinos pendientes”. En el *modelo de explotación* algunos de los registros almacenados en esta tabla se han almacenado en la tabla e_destino, y esta tabla se ha eliminado.
- Tabla **r_destinopendientehabitante**: en el *modelo intermedio* en esta tabla se ha almacenado la información referid a la relación de la tabla e_destinopendiente y e_habitante. En el *modelo de explotación* al relacionarse la tabla e_habitante con la tabla e_destino se ha creado otra tabla para representar dicha relación, y esta tabla se ha eliminado.
- Tablas **p_accv**, **p_estado** y **p_tematico**: en esta tabla se ha almacenado cierta información para comparar la información existente en las columnas vehiculotransitable y

observacionacceso de la tabla e_tramorural. Después de analizar toda esta información se ha introducido dicha información se ha almacenado en las tablas e_vehiculo, e_estado y e_suelo.

- Tabla **e_rutapendiente**: en esta tabla se ha almacenado toda la información referida a los destinos denominados “rutas pendientes”. En el *modelo de explotación* algunos de los registros almacenados en esta tabla se han almacenado en las tablas e_tramo, e_ruta y r_rutatramo, y esta tabla se ha eliminado.
- Tablas **e_tramocarreteraalternativo** y **e_tramoruralalternativo**: en esta tabla se ha almacenado la información relativa a las rutas alternativas. En el *modelo de explotación* algunos de los registros almacenados en esta tabla se han almacenado en las tablas e_tramo, e_ruta y r_rutatramo, y esta tabla se ha eliminado.

Las tablas creadas utilizando instrucciones de programación en SQL:

- Tablas **e_salida**, **e_servicio**, **r_salidaservicio**: se han introducido manualmente en la base de datos.
- Tablas **p_tipozona**, **p_tipodestino**, **p_tipotramo**, **p_tipovehiculo**, **p_tipoestado**, **p_tiposuelo**, **p_tiposalida**, **p_tiposervicio**: se han introducido manualmente en la base de datos.
- Tabla **r_destinovecindad**, **e_cruce**, **r_cruceruta**: esta tabla no se ha obtenido en el proceso de carga. Esta tabla se formará al realizar consultas de programación en SQL.
- Tabla **r_destinohabitante**: esta tabla se ha creado a partir de la tabla r_destinopendientehabitante, al obtener las coincidencias entre e_destino y e_destinopendiente.
- Tabla **r_tramoruta**: la relación de los tramos con las rutas se ha obtenido a partir de la relación de la tabla de rutas con las tablas de tramo carretera y tramo rural.

5.4. Cambios en el contenido del modelo de datos intermedio

En el proceso de transformación del modelo de carga después de realizar la reestructuración se han realizado operaciones de depuración de la información tal y como se detalla a continuación. En el Anexo A3.2 existen varios ejemplos de estas modificaciones.

5.4.1. Tabla e_destino

En la tabla e_destino existen registros duplicados que deberán ser eliminados. Para ello, se realizarán diferentes consultas de programación en SQL: se identificarán los casos, se procederá a modificar las relaciones existentes con estos destinos y finalmente se eliminarán esta duplicidades existentes.

5.4.2. Tabla e_habitante

La información de habitantes está asociada a los Destinos de la tabla e_destinopendientes. Por lo tanto, se asociará esta información a los Destinos coincidentes de la tabla e_destino. Por ejemplo, los destinos iddestino 90, 91 y 92 de la tabla e_destinopendiente son coinciden-

tes con los destinos iddestino 77, 78 y 79 de la tabla e_destino, por lo que se asociará a estos destinos la información de habitantes.

5.4.3. Tabla r_destinovecindad

La tabla de vecindad se creará considerando como vecinos/as los diseminados existentes a menos de 500 m. Al analizar la distancia entre diseminados (realizando consultas SQL) se observa que existen casos donde diferentes diseminados pueden ubicarse en un mismo edificio.

5.4.4. Tabla e_salida

Al analizar el tramo inicial de todas las rutas se ha identificado dos puntos de salida en vez de uno (ver la Imagen 24). La existencia de un segundo punto de salida puede ser debido a que existen dos garajes o emplazamientos donde estacionan los vehículos del servicio de bomberos en el Parque de Oronoz-Mugaire. Al no disponer suficiente información para confirmar esta suposición no se añadirá un segundo registro a la tabla e_salida.

Imagen 24: Las dos ubicaciones del punto de salida de las rutas

5.4.5. Tablas e_tramocarretera y e_tramorural

Existe duplicidad en los tramos, por lo tanto se modificará el contenido de las tablas e_tramocarretera y e_tramorural para que no haya tramos duplicados en la base de datos. Para ello, se realizarán diferentes consultas de programación en SQL: se identificarán el número de casos, se procederá a modificar las relaciones existentes con estos tramos y al final se eliminará esta duplicidad.

También se ha analizado la composición de las rutas, es decir, la agrupación de los tramos que conforman las rutas: qué tramos componen cada ruta y en qué orden. Para ello primero ha sido necesario identificar los distintos casos de agrupación de tramos: agrupación de Tra-

mo de Carretera y Destino; agrupación de Tramo de carretera y Tramo rural; agrupación de Tramo rural y Tramo rural; y agrupación de Tramo rural y Destino.

5.4.6. Tablas e_tramocarreteraalternativo y e_tramoruralalternativo

Como se ha realizado en las tablas anteriores, en estas tablas también se ha analizado la duplicidad de los tramos y la agrupación de los tramos (identificando las diferentes tipologías). En este caso también se obtendrá la composición de cada ruta. Toda esta información se incluirá en las tablas e_ruta, e_tramocarretera y e_tramorural.

5.4.7. Tabla e_cruce

Los cruces se han definido como el punto de agrupación de un tramo de carretera con un tramo rural, el punto de transición de una carretera a una vía rural. Por lo tanto, en esta tabla se almacenarán los cruces obtenidos de las rutas normales y alternativas, utilizando para ello las tablas temporales creadas en la composición de rutas.

5.4.8. Tabla e_puntoinfo

Al analizar los tramos de las rutas se ha identificado un caso donde en una misma carpeta existen dos destinos, y a su vez dos rutas. Por lo tanto, en el proceso de carga uno de estos dos destinos se ha incorporado a la base de datos como un punto de información, y los tramos de las dos rutas se ha incorporado como parte de una sola ruta. Este destino tiene el nombre de “Borda finde” y está asociado al iddestino 75 de la tabla e_destino. Al corregir este error de asignación se ha creado la tabla r_puntoinforuta.

5.4.9. Tabla e_tramo

Esta tabla se ha utilizado para reestructurar la información relativa a los tramos: primero se ha creado la tabla p_tipotramo; a continuación se ha añadido la información de las tablas de los tramos de carretera y los tramos rurales, especificando el tipo de tramo; y después se ha eliminado la tabla e_tramocarretera y se ha modificado la estructura de e_tramorural. Al realizar esta reestructuración se ha modificado la tabla r_rutatramo.

5.4.10. Tablas e_vehiculo, e_suelo y e_estado

Toda la información relativa a los vehículos, las características del suelo y el estado de la calzada se han analizado detenidamente, identificando los diferentes casos que se dispone. Posteriormente, toda esta información se ha reclasificado y se ha reestructurado creando nuevas tablas para ello.

5.5. Conclusiones sobre el proceso de carga

Tras realizar los procesos de carga y de transformación, se ha analizado el origen de los problemas detectados en estos procesos. En el siguiente apartado se proponen recomendaciones y criterios a seguir para minimizar o evitar dichos problemas, utilizando una metodología de carga y de transformación específica.

5.5.1. Datos redundantes: destinos

En los datos de entrada se han identificado duplicidades en la información de destinos. Pero al analizar el origen de esta duplicidad, se ha llegado a la conclusión de que no ha existido información redundante en la obtención de datos en campo. Sino que esta duplicidad ha surgido al realizar pruebas de asignación de cierta información. Es decir, se han creado destinos duplicados para obtener diferentes ejemplos que contengan información referida al propio destino y/o información de las personas que habitan en ella.

Por lo tanto, se recomienda introducir en la base de datos el conjunto de datos relacionados con los destinos de forma independiente a la información relativa a sus habitantes. De esta forma, se evitarán procesos de transformación posteriores a la carga.

También se recomienda en el proceso de captura de información de los destinos almacenar cada destino en una misma carpeta del KML. De esta forma, se pretende evitar errores de asignación en el proceso de carga.

5.5.2. Datos redundantes: tramos

En los datos de entrada también se han identificado duplicidades en los tramos. En este caso, se observa que la duplicidad ha sido creada conscientemente y se ha realizado al obtener y digitalizar los datos de los tramos. Por un lado existe duplicidad en las rutas del recorrido obtenidas mediante mediciones en campo, y por otro lado se ha duplicado tramos al realizar la edición del recorrido con algún software de navegación GPS. Por ejemplo, al crear tracks de los itinerarios con un móvil existen numerosos casos donde se han repetido mediciones varias veces en las mismas carreteras y caminos rurales; también se han dado varios casos donde se han medido sólo tracks de una parte del itinerario (a partir de un cruce de carretera por ejemplo) y posteriormente en la edición de la ruta se les ha añadido copias de tramos de otros tracks.

Por lo tanto, se recomienda aplicar otra metodología para la obtención de datos en campo. Los criterios o recomendaciones a seguir serían las siguientes:

1. Medición de los tracks: solamente se medirán tramos de carretera y de caminos rurales que no existen en la base de datos. Es decir, las mediciones se indicarán desde un cruce de carretera de un tramo ya existente en otro track y se procederá a medir sólo la parte del itinerario de la ruta que no existe en la base de datos.
2. Edición de los tracks: se editarán todos los tracks medidos, con la finalidad de obtener tramos delimitados por cruces de tramos. De este modo, dos rutas que compartan una parte del recorrido estarán asociados a los mismos tramos, y no habrá en la base de datos tramos con una parte del itinerario duplicado o solapado.
3. Almacenamiento de los tramos: al realizar correctamente la medición y la edición de los tracks se evitarán crear tramos duplicados en la base de datos.
4. Modificación de los tramos: al realizar mediciones de un track de una ruta puede producirse el caso de que la medición parta de un cruce de carretera que no se había tenido

en cuenta anteriormente. En este caso el punto de inicio del tramo medido no coincidirá con el punto de inicio o final de ningún tramo existente en la base de datos, sino con un punto intermedio. Por lo tanto, después de medir y editar el nuevo track se procederá a editar el tramo existente en la base de datos, y se crearán dos tramos a partir de este tramo. Estos dos nuevos tramos heredarán la misma información asociada al tramo original.

5. Obtención de información sobre tramos de carretera: el desarrollo de esta herramienta se ha enfocado principalmente a la visualización de datos referidos a tramos rurales y a los propios diseminados, pero resultaría interesante reflejar la información referida a los tramos de carretera. En este contexto, se podrían crear subtramos con características asociadas. Por ejemplo, alguna incidencia como una obra, un deslizamiento de tierras, algún corte de carretera por alguna actividad deportiva-cultural, etc.

También se recomienda utilizar una metodología para crear las polilíneas que conforman los tramos:

- El punto de inicio de la polilínea deberá estar en la dirección del punto de salida de la ruta, y a su vez, el punto final deberá estar en la dirección del punto de destino de la ruta.
- Todos los tramos asociados a una ruta estarán enlazados espacialmente entre sí.
- El identificador idtramo de los tramos se adjudicará de menos a más según la dirección del recorrido de la ruta. Por ejemplo, en el caso de que en una ruta dos tramos que están unidos por el punto del final de un tramo y el punto del inicio del siguiente tramo se le adjudicará el número identificador del tramo (idtramo) de la siguiente forma: el tramo que esté más cerca del punto de salida de la ruta contendrá un idtramo más pequeño que el tramo que está más cerca del punto de destino.
- Todos los puntos de inicio de los tramos que representa el tramo inicial de carretera de las rutas serán el mismo punto espacial (el punto oficial de salida definido en este proyecto). Es decir, todas las rutas partirán del edificio central del parque de bomberos de Oronoz-Mugaire. Posteriormente, al desarrollar más la aplicación, se podría definir diferentes localizaciones de los equipamientos que pueden conformar este parque de bomberos u otro servicio de emergencia. De este modo, se definirían nuevos tramos iniciales de carretera y se podría obtener rutas relativas a estos nuevos puntos de salida.

Los tramos que conforman rutas alternativas se crearán y se editarán de la misma forma que los tramos de las rutas normales de los diseminados.

Al contener tramos no duplicados en la base de datos se podrá analizar la calidad gráfica y precisión de los mismos. De manera que si algún tramo no se ajusta a los parámetros de calidad exigidos se realizará las modificaciones y ajustes necesarios para obtener dicha calidad.

También se recomienda definir correctamente la representación espacial de los cruces de carretera para cada ruta. De esta manera, en un caso de emergencia grave los vehículos que provienen de otros parques de bomberos de otra comarca de Navarra podrán acceder direc-

tamente y sin equivocaciones a la ruta de acceso utilizando para ello la ubicación del cruce (por ejemplo, se podrá utilizar un navegador GPS de carretera para acceder al inicio de los tramos rurales de una ruta).

5.5.3. Datos pendientes de editar: destinos y rutas pendientes

Se recomienda no almacenar en la base de datos destinos y rutas pendientes de edición. Y también se recomienda almacenar organizadamente toda la información asociada a los destinos y los tramos. De esta forma, se ahorrará tiempo y trabajo de modificaciones en la base de datos, y se asegurará que la información que contenga la base de datos sea homogénea.

5.5.5. Datos asociados a los diseminados y a los tramos de las rutas

El grado de complejidad y de detalle de otros datos complementarios asociados a los diseminados y a los tramos determinarán la potencialidad y/o la eficiencia de esta herramienta en los servicios de emergencia. Por lo tanto, se recomienda recopilar e integrar una variedad importante y relevante de información en esta base de datos.

5.5.6. Declaración y asignación de estilos

En los datos de entrada existen diferentes modelos de declaración y asignación de estilos de los elementos lineales y puntuales de tramos, rutas y diseminados. Esta información no se ha incluido en el proceso de carga, por lo tanto en esta aplicación se visualizarán los datos con un criterio distinto al del fichero de entrada KML. Este problema que se identificó en el análisis del archivo de entrada no será pues relevante y no se ha tenido en cuenta en el proceso de desarrollo de la base de datos.

5.5.7. Sistema de referencia de coordenadas

Las coordenadas que se obtengan en campo estarán en el sistema de referencia europeo ETRS89, evitando así realizar transformaciones de coordenadas. Si se utiliza un dispositivo de captura que obtiene datos utilizando el sistema de referencia WGS84, posteriormente en PostGIS se realizará la transformación.

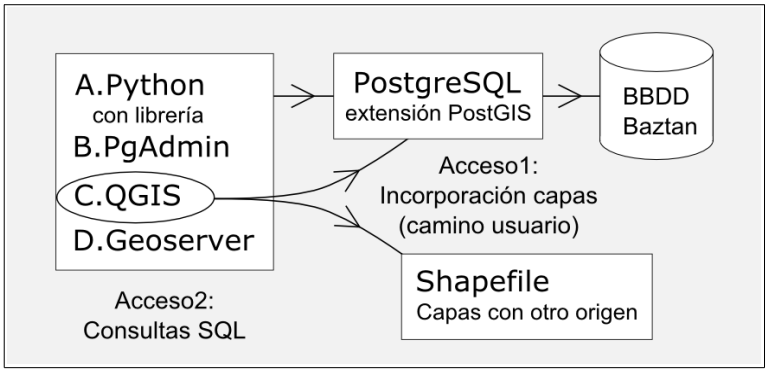

Imagen 25: Diferentes opciones de acceso a la BBDD

5.6. Obtención del modelo de explotación

Después de realizar la migración del modelo de carga, al modelo de datos de explotación diseñado se procederá a realizar diferentes pruebas para comprobar si el modelo construido se ajusta a las necesidades del proyecto mediante el análisis de los datos que se han almacenado en la base de datos.

Para acceder y analizar los datos de la base de datos se han definido cuatro opciones (ver Imagen 25):

1. **Python**: para acceder a la base de datos se podrá utilizar un entorno de programación integrado o IDE basado en Python utilizando librerías específicas para ello. Por ejemplo se podría utilizar PyCharm con la librería psycopg2.
2. **PgAdmin**: para acceder a la base de datos se podrá utilizar esta herramienta de administración de base de datos PostgreSQL. En nuestro caso, se ha utilizado pgAdmin 3 para gestionar la base de datos.
3. **QGIS**: este software GIS está desarrollado para visualizar datos almacenados en una base de datos espacial PostGIS, y posibilita la edición de dicho contenido. Esta herramienta contiene la opción de realizar consultas espaciales SQL y de representar los resultados de estas consultas. Estos resultados se podrán visualizar en el entorno GIS como una capa o se incorporarán a la base de datos como un tabla o una vista.
4. **Geoserver**: este servidor permite utilizar los datos de la base de datos para crear servicios WMS, WFS y WCS. Este sistema permitirá a un/una usuario/a encontrar, visualizar, utilizar y combinar la información geográfica existente en la base de datos.

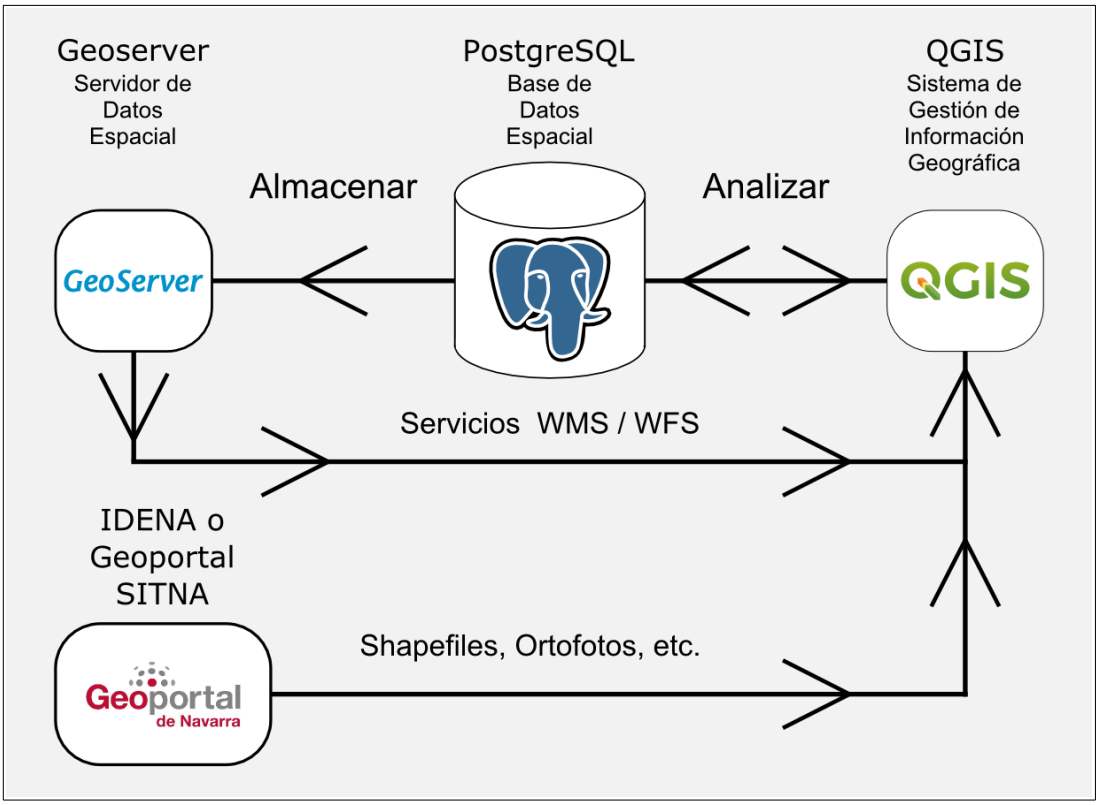

Imagen 26: Arquitecto del proceso del análisis del contenido de BBDD

En el análisis del modelo de explotación realizado el acceso a la base de datos espacial se ha realizado de dos formas (ver Imagen 26): una mediante el software QGIS y otra utilizando Geoserver. Los ejemplos de consultas SQL y procesos realizados se han explicado en el Anexo A3.3.

6. Explotación del sistema

6.1. Arquitectura del sistema

En este apartado se explicará la composición de la arquitectura del sistema de información necesario para el acceso al sistema de información desarrollado que permitirá a un/una usuaria acceder a la información geográfica utilizando un navegador web (desde un ordenador, un móvil o una tablet).

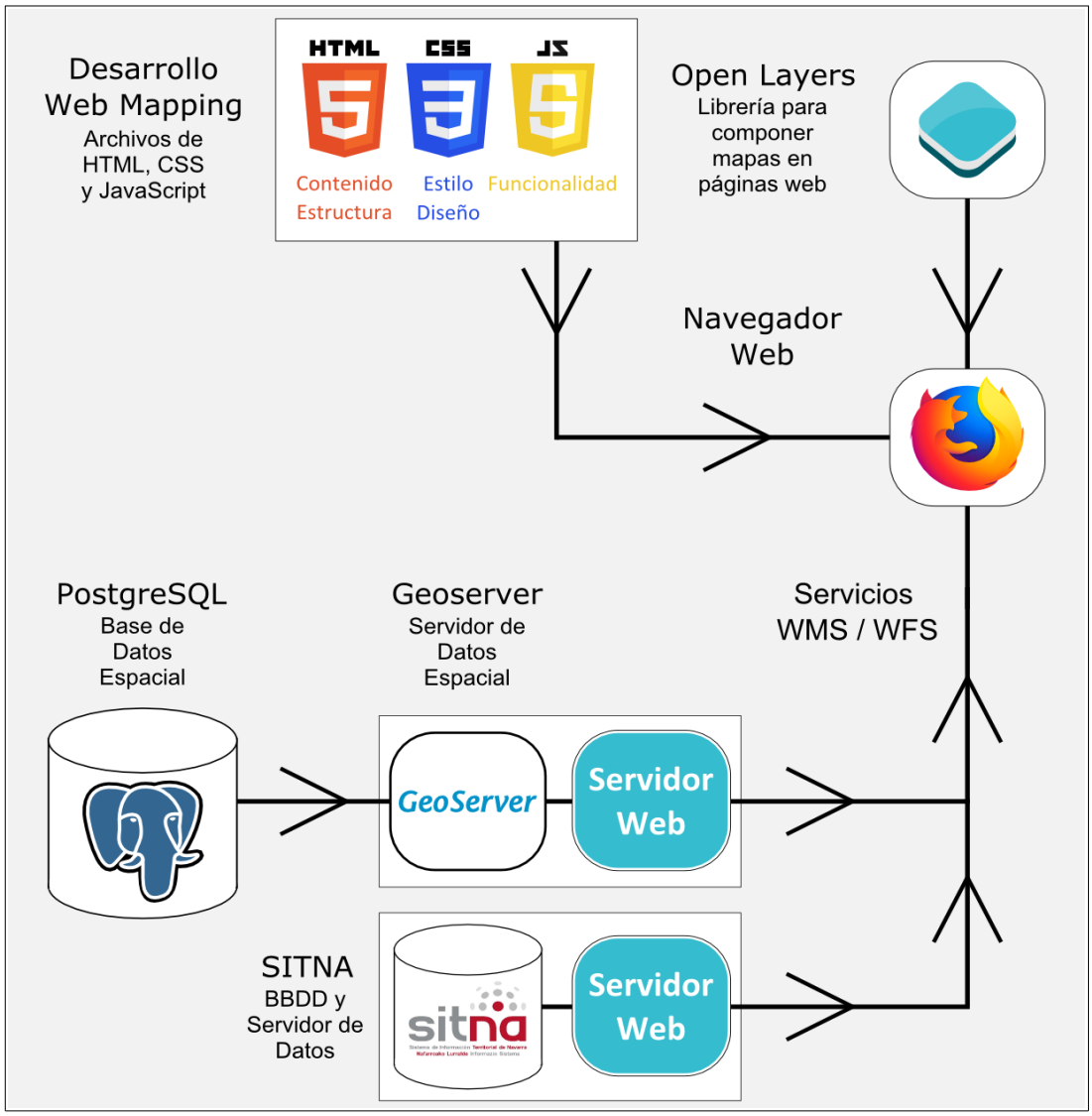

Imagen 27: Arquitectura del sistema de la aplicación

Como se aprecia en la Imagen 27, la arquitectura de acceso al sistema será la siguiente:

- Back-end: esta capa contendrá los componentes necesarios para el acceso a los datos. En este caso, la parte que procesa la entrada lo compondrán el Servidor de Base de Datos y el Servidor Web.
- Servidor BBDD: en este nivel de arquitectura se ubican la base de datos PostgreSQL creada en este proyecto y los repositorios del SITNA.
- Servidor Web: en este nivel ubica el servidor Geoserver y el servidor de SITNA que sirven los datos desde las bases de datos, y utilizan servicios web para crear servicios web de tipo WMS y WFS.
- Frontal o cliente: en esta capa se despliegan los componentes y herramientas con los que interactúa el/la usuaria, que consistirá en un navegador web para acceder al mapa incrustado en una página web. Este mapa estará definido (su estructura, aspecto y funcionalidad) por los archivos HTML, CSS y Javascript, y se utilizará la librería OpenLayers o una API de Javascript (como la API SITNA) para facilitar toda la funcionalidad requerida en la interacción con el mapa. La información geográfica se obtendrá utilizando los servicios WMS/WFS de Geoserver y SITNA.

6.2. Herramientas de explotación

En este apartado se explican dos aproximaciones principales para el análisis y explotación de los datos almacenados en la base de datos espacial del sistema: consultas espaciales SQL desde QGIS; confección de mapas WMS; y acceso a través de servicios WFS.

Estas aproximaciones se realizarán mostrando dos ejemplos básicos: un ejemplo de explotación de datos en QGIS mediante consutlas espaciales y otro ejemplo de explotación de datos en Geoserver (WMS y WFS). En el Anexo A3.3. existen ejemplos más desarrollados.

6.2.1. Explotación de datos en QGIS mediante consultas espaciales

Este ejemplo se han realizado con el software QGIS, idónea para visualizar y editar el contenido de una base datos espacial PostGIS. Se ha utilizado la herramienta de Administración de Base de Datos contenida en el software QGIS, y con ella se han realizado diferentes consultas espaciales SQL, y se han creado tablas y vistas en la base de datos, y posteriormente se han visualizado en este entorno GIS.

Ejemplo 1: tramos rurales clasificados por el grado de transitabilidad

En este ejemplo se combinarán cuatro tablas de la base de datos para obtener un resultado relacional que contenga la información espacial de los tramos rurales y el tipo de estado que se encuentran los tramos rurales.

Como se aprecia en la imagen 28 en el Administrador de BBDD se realizarán las siguientes consultas SQL para obtener dicho resultado:

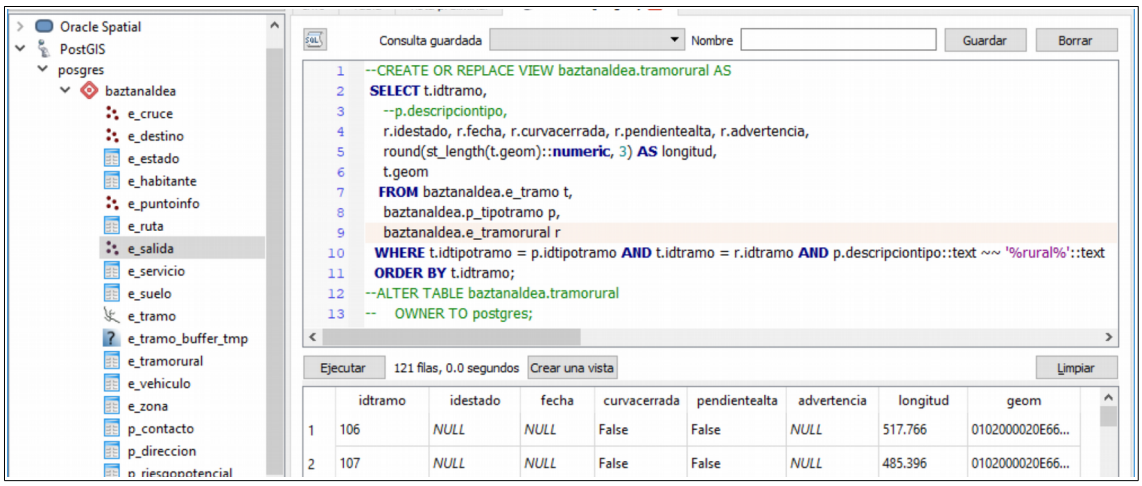

Imagen 28: Consultas SQL para combinar la información de los tramos rurales y tipo de estado

Este es el código que se ha utilizado:

```
CREATE OR REPLACE VIEW baztanaldea.v_tramorural_estado AS
 SELECT t.idtramo, p.descripciontipo, r.idestado, r.fecha,
        r.curvacerrada, r.pendientealta, r.advertencia, e.idtipoestado,
        e.caracteristica, st_length (t.geom) AS longitud, t.geom
FROM baztanaldea.e_tramo t, baztanaldea.p_tipotramo p,
        baztanaldea.e_tramorural r, baztanaldea.e_estado e
WHERE t.idtipotramo = p.idtipotramo AND t.idtramo = r.idtramo
        AND r.idestado = e.idestado AND p.descripciontipo::text = 'tramo rural'::text;
```

En este caso, con el resultado de esta consulta se ha creado una vista en SQL, y a continuación se ha añadido al mapa de QGIS (como se puede observar en la imagen 29).

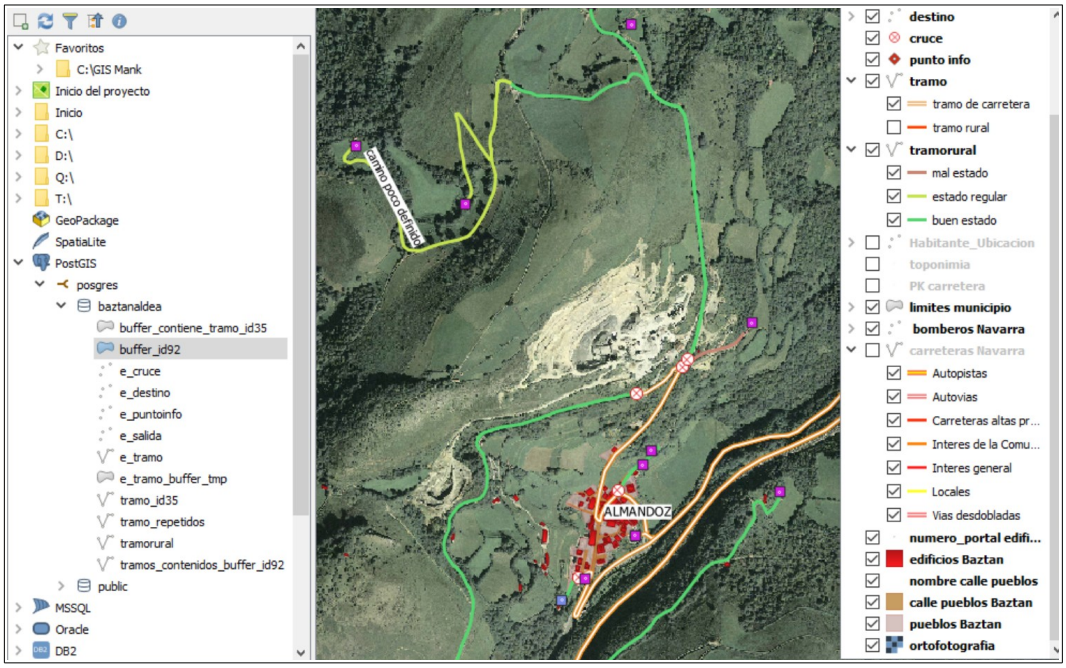

Imagen 29: Visualización de los tramos rurales clasificados por el tipo de estado que contienen

6.2.2. Explotación de datos en Geoserver (WMS y WFS)

Este ejemplo se han realizado con el servidor Geoserver, es decir, se ha utilizado este software para visualizar información geográfica de la base de datos.

Ejemplo 2: identificación de tramos de carretera importantes

En este ejemplo se analizará los tramos de carretera dependiendo del número de rutas que pasan por ellas. De esta forma, se identificarán los tramos más importantes según su acceso a los diseminados.

Como se aprecia en la imagen 30 se ha utilizado el Ejecutor de consultas SQL de PgAdmin III para crear una vista SQL con la lista de tramos de carretera con la información del número de rutas.

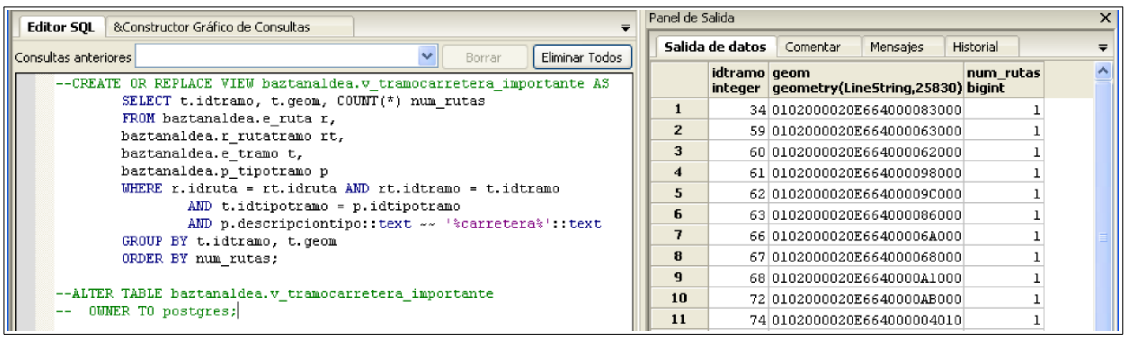

Imagen 30: Consulta realizada para obtener los tramos de carretera importantes

Este es el código que se ha utilizado:

```
CREATE OR REPLACE VIEW baztanaldea.v_tramocarretera_importante AS
    SELECT t.idtramo, t.geom, COUNT(*) num_rutas
    FROM baztanaldea.e_ruta r, baztanaldea.r_rutatramo rt, baztanaldea.e_tramo t,
            baztanaldea.p_tipotramo p
    WHERE r.idruta = rt.idruta AND rt.idtramo = t.idtramo
            AND t.idtipotramo = p.idtipotramo
            AND p.descripciontipo::text ~~ '%carretera%'::text
    GROUP BY t.idtramo, t.geom
    ORDER BY num_rutas;
```

A continuación, en el servidor Geoserver se creará un almacén de datos y una capa con esta información. Como se puede observar en la imagen 31 se ha especificado un estilo SLD para esta capa.

Imagen 31: Asignación del estilo a la capa de servicios WMS de tramos de carretera importante

Finalmente se ha añadido como un servicio de WMS al mapa del software QGIS. Como se puede observar en la imagen 32, esta vista SQL también se ha añadido directamente de la base de datos (para ver la diferencia de visualización).

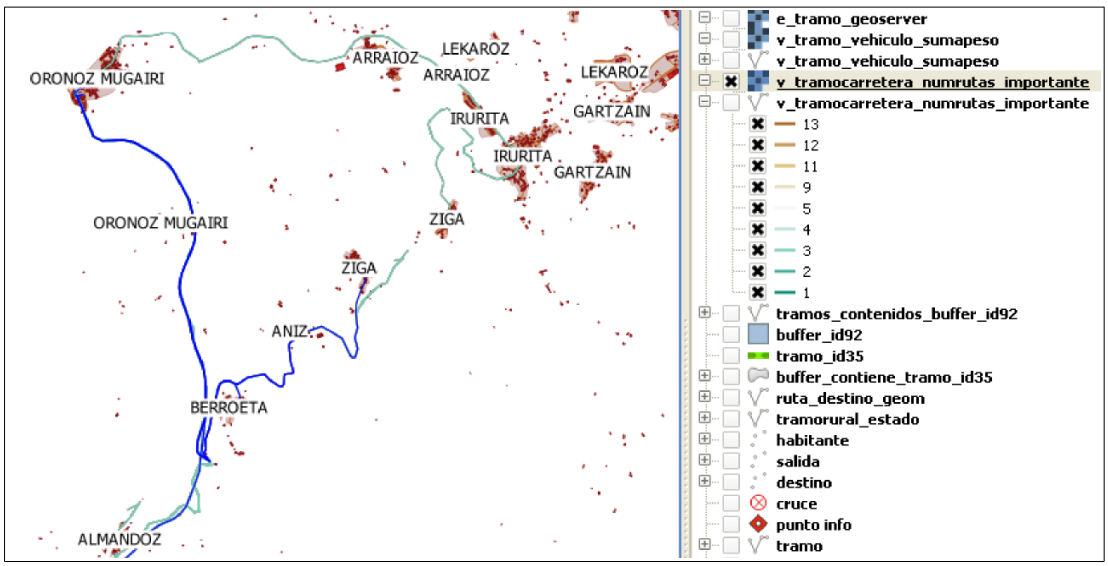

Imagen 32: Visualización de los tramos de carretera importantes en el mapa del software QGIS

7. Programas y lenguajes de programación utilizados

En este apartado se mencionarán brevemente las principales lenguajes de programación y herramientas utilizadas en el desarrollo del proyecto:

1. Lenguaje **KML**: es un lenguaje de marcado basado en XML (Extensible Markup Language). Se utiliza para crear archivos para representar datos geográficos en tres dimensiones.
2. Lenguaje **Python**: es un lenguaje de programación interpretado. Se ha utilizado la versión Python 3, y la librería **minidom**.

3. Lenguaje **SQL**: es un lenguaje de consulta estructurado. Está diseñado para administrar sistemas de gestión de bases de datos relacionales.
4. Lenguaje de programación **HTML**: es un lenguaje de marcado para la elaboración de páginas web.
5. Lenguaje de programación **JavaScript**: es un lenguaje de programación interpretado. Se ha utilizado la librería **OpenLayers**, y **API SITNA** para realizar diferentes pruebas.
6. **PyCharm**: programación integrado o IDE para el desarrollo con Python.
7. **PostgreSQL**: sistema de gestión de bases de datos relacional orientado a objetos. Se le ha añadido la extensión espacial PostGIS, para obtener una base de datos espacial.
8. **PgAdmin**: herramienta de administración de base de datos PostgreSQL.
9. **SQL Power Architect**: software para el diseño de diagramas de estructura de datos. Permite obtener el modelo de datos automáticamente a partir de la conexión de una base de datos.
10. **QGIS**: herramienta de análisis y gestión de información geográfica. Este software permite trabajar con base de datos espaciales, como PostGIS.
11. **Geoserver**: es un servidor de datos espaciales conforme a los principales estándares del OGC. Se ha utilizado para la publicación de información geográfica contenida en la base de datos.
12. **Apache OpenOffice** y **LibreOffice**: software de ofimática utilizado para realizar trabajos de documentación del proyecto.
13. **Inkscape**: software de edición de gráficos vectoriales. Este programa se ha utilizado para realizar los diagramas de flujo del proceso ETL de carga y realizar diferentes gráficos de la documentación.

8. Valoraciones y conclusiones

En este apartado se realizarán valoraciones y se extraerán ciertas conclusiones sobre la metodología utilizada en el desarrollo del proyecto. Se describen a continuación:

1. Obtención de datos espaciales: Es necesario establecer una metodología de obtención de datos en campo (ver el apartado 6.5).
2. Obtención de información relacionado con los datos espaciales. Es importante el trabajo de obtención de información asociada a elementos principales del sistema: salidas, destinos, tramos, cruces y puntos de información. También será importante la información relativa a habitantes, servicios, vehículos, suelos y estados. Por lo tanto, la complejidad y detalle de la información recopilada determinará el grado de potencialidad de la herramienta desarrollada.
3. Extracción de información desde un archivo KML. La metodología utilizada para el proceso de carga ha sido satisfactorio. La utilización del lenguaje de programación Python y el

software PyCharm han cumplido las exigencias del proceso de extracción de un archivo con una estructura KML. Pero la complejidad de los datos de entrada ha requerido mucho análisis del contenido del archivo KML y un diseño elaborado del algoritmo de extracción de datos. Por ello, se ha invertido mucho tiempo de trabajo en la creación de diagramas de flujo.

4. Diseño de la base de datos. El diseño del modelo de datos conceptual, del modelo relacional y del modelo físico ha sido esencial en la creación del sistema. Por ello, se ha analizado en profundidad el modelo de datos necesario para este proyecto. La creación de los diagramas de cada modelo ha sido muy importante para definir los diferentes procesos realizados en este proyecto. En especial, para el proceso de carga de datos y el proceso de transformación de la base de datos.
5. Almacenamiento de los datos obtenidos en el proceso de carga. La utilización de la herramienta de administración PgAdmin de PostgreSQL para los trabajos de creación de la estructura de la base de datos y los trabajos de almacenamiento de datos extraídos también ha sido satisfactorio. Al realizar la extracción datos con el software PyCharm se han obtenido código SQL para insertar registros de información en las tablas de la base de datos. Esta metodología ha facilitado la obtención del modelo y datos de carga.
6. Transformación del modelo intermedio de datos. La migración del modelo ha sido compleja. Los cambios en la estructura de la base de datos, la identificación y eliminación de los datos redundantes, y la modificación de los datos espaciales ha requerido mucho trabajo. Para el tratamiento de datos ha sido fundamental el uso de consultas en SQL.
7. Análisis del modelo de explotación. La utilización de la herramienta de Administración de BBDD incluida en el software QGIS ha cumplido todas las expectativas para analizar la base de datos espacial que se ha desarrollado. La posibilidad de incluir como capas consultas SQL realizadas a la base de datos, convierte a los software GIS un elemento esencial en la gestión de una base de datos espacial.
8. Creación de servicios web WMS /WFS. La utilización del servidor de datos espaciales Geoserver será importante para que el/la usuaria de la aplicación desarrollada pueda acceder a la información espacial almacenada en la base de datos. Por lo tanto, la elección de la información que se quiere representar y su tipología tendrán gran importancia en este punto.
9. Desarrollo de archivos HTML, CSS y JavaScript. La creación de estos archivos ha permitido diseñar un entorno web. Esta fase de creación del sistema ha sido la menos desarrollada en el proyecto.

8.1. Lineas futuras de desarrollo

- Utilizar la API SITNA para la composición de la aplicación.
- Programar en Javascript, y desarrollar una aplicación con más funcionalidades. Por ejemplo, contener un buscador donde el resultado de la búsqueda se represente en el mapa.

- Mejorar el aspecto visual del entorno web, para que sea más intuitivo y atractivo para el/la usuaria.
- Utilizar la librería GDAL de Python para analizar la relación espacial de los datos almacenados en la base de datos. Por ejemplo, si los tramos de una ruta son continuas o contienen espacios entre ellos.

9. Bibliografía

ABLITAS MURO, José Miguel (2012). Nuevo sistema de geolocalización en Navarra para disminuir los tiempos de respuesta en aviso urgente en zonas de montaña y de gran dispersión.

ELMASRI, Ramez y NAVATHE Shamkant (2007). Fundamentos de Sistemas de Base de Datos. Addison-Wesley, 5ª edición.

ULLMAN, Jeffrey D. y J. WIDOM, Jennifer (1999). Introducción a los Sistemas de Bases de Datos.

DE MIGUEL CASTAÑO, Adoración y PIATTINI, Mario (1997). Fundamentos y modelos de bases de datos.

GAUCHAT, Juan Diego (2013). El gran libro de HTML5, CSS3 & Javascript. Ediciones Marcombo.

Tutorial de KML: *https://developers.google.com/kml/documentation/kml_tut*

Tutorial de Python: *http://docs.python.org.ar/tutorial/3/index.html*

https://www.w3schools.com/python

Tutorial Pycharm: *https://www.jetbrains.com/help/pycharm/quick-start-guide.html*

Tutorial de SQL: *https://www.w3schools.com/sql*

Manual PostgreSQL: *https://www.postgresql.org/docs/manuals*

Manual Geoserver: *http://docs.geoserver.org/stable/en/user/index.html*

Tutorial de HTML: *https://www.w3schools.com/html*

Tutorial de CSS: *https://www.w3schools.com/css*

Tutorial Javascript: *https://www.w3schools.com/js*

Tutorial OpenLayers: *http://openlayers.org/en/latest/doc/tutorials*

API SITNA 1.5.0 *http://sitna.navarra.es/api/doc*

ANEXO 1: Documentación de entrada.

Índice del Anexo 1 - Documentación de entrada:

A1.1 Bases del proyecto SITNA – Bomberos de Navarra

Las bases del proyecto definidas en la mesa de coordinación de SITNA – Bomberos Navarra son:

1. Es necesario la creación de una adecuada base de datos: centralizada, normalizada y georeferenciada.
   - En este proyecto se creará una base de datos espacial, y en ella se agregarán los datos de forma estructurada. Estos datos serán información explotable y se podrán adquirir o modificar.
   - Los datos se añadirán a la base de datos con un grado de calidad preestablecido y se evitarán datos redundantes.
   - Esta base de datos será centralizada y podrá ser utilizada en un mismo momento por distintas aplicaciones. Por ejemplo, si se desarrolla una aplicación para otro tipo de servicio de emergencia (como puede ser para técnicos/as sanitarios/as de *Osasunbidea* o para otros/as profesionales) esta nueva herramienta podrá acceder a los datos de la misma base de datos.
2. Datos fundamentales en la base de datos: la información asociada a los diseminados.
   - Será importante la correcta identificación y localización de estos caseríos y viviendas dispersas. También será importante una correcta recopilación de la información relativa a estos diseminados.
   - En la base de datos sólo se incluirán los diseminados que son vivienda y estén habitados, pero habrá alguna excepción. En este conjunto de diseminados también se han incluido viviendas utilizadas temporalmente como edificios de alojamiento (agroturismos, albergues de montaña, etc.) y otros diseminados como edificios e instalaciones públicas (escuelas, ermitas, cementerios, etc.), de patrimonio cultural (molinos, puentes, etc.) y de interés turístico (palacios, torres medievales, etc.) y empresas o establecimientos relacionados con la agricultura y la ganadería (cuadras, granjas, etc.).
   - Una adecuada nomenclatura de los diseminados será esencial. Por lo tanto, los nombres de los diseminados serán las mismas que las utilizadas por los habitantes de estos municipios, y podrán contener nombres no acordes a la toponimia oficial. La recopilación de los nombres lo realizarán el equipo de Bomberos del parque comarcal de Oronoz-Mugaire, y a esta labor podrán contribuir otros organismos e instituciones como *Euskarabidea*[5] y los ayuntamientos de *Baztan*, *Urdazubi/Urdax* y *Zugarramurdi*.

5 Euskarabidea es el Instituto Navarro del Euskera. Es un organismo adscrito en el Gobierno de Navarra.

3. Datos fundamentales en la base de datos: la información referida a las rutas.
   - La simple obtención de rutas no cubrirán totalmente las necesidades de los diferentes equipos de emergencia, porque no está garantizado que todos los vehículos de estos servicios de emergencia puedan transitar por dichas rutas. Las dificultades de transito por vías de un entorno rural y de orografía montañosa son importantes, y no todos los vehículos podrán circular por todas las vías. Por consiguiente, en algunos casos como tramos con pendientes elevadas o con curvas muy cerradas, los vehículos con un volumen grande o con poca maniobrabilidad no podrán transitar. Por esta razón, la herramienta a desarrollar tendrá que permitir consultar la información sobre la accesibilidad de las rutas: en cada ruta se especificará claramente qué tipo de vehículo puede transitar y se indicará las ubicación de los tramos donde es difícil transcurrir.
   - El mantenimiento de esta información será clave para el correcto funcionamiento de la aplicación. Al modificarse alguna vía o camino en el terreno, se identificarán las rutas que transcurren por estos tramos, y estas nuevas características de transitabilidad se introducirán en la base de datos.
4. Metodología para crear la base de datos, y fundamentos para insertar y actualizar registros.
   - Los datos de entrada serán una muestra de datos recopilado por los/las bomberos/ras, de una zona reducida del *Baztan*. Estos datos se analizarán y se modificarán para incluirlo en la base de datos. Para ello, se utilizarán procesos ETL (procesos “Extract, Transform, Load”).
     1. El archivo que contiene la muestra de datos es un fichero con formato KML.
     2. Se creará una primera aproximación de la base de datos: una base de datos acorde al contenido y estructura del fichero KML.
     3. Los datos contenidos en el archivo KML se extraerán utilizando el lenguaje de programación Python, y se añadirán a la base de datos creada.
     4. La estructura de la base de datos se modificará teniendo en cuenta la base de datos diseñada con el modelo de datos conceptual y el modelo relacional.
     5. En el contenido de la base de datos se realizarán cambios: se eliminará los datos redundantes, se modificarán los datos incompletos y erróneos y se reorganizarán los datos. Para ello, se utilizará el lenguaje SQL y el software GIS.
   - Se definirán los pasos a seguir para incluir más registros en la base de datos. También se definirán una serie de indicaciones para actualizar los registros de la base de datos.
5. Utilización del PostegreSQL con extensión PostGIS: se utilizará una base de datos espacial gestionada por un sistema que permita representar datos geográficos. De esta forma, este proyecto se ajustará a los requerimientos del usuario y de la herramienta.

- El modelo de la aplicación web desarrollada contendrá una *arquitectura Cliente – Servidor*. En nuestro caso, el cliente será cualquier dispositivo que tenga un navegador web donde el usuario (bombero/ra, sanitario/a, etc.) podrá realizar las solicitudes estableciendo así un diálogo con el servidor. Posteriormente este cliente esperará la respuesta del servidor. Por lo tanto, el servidor será el componente encargado de procesar las solicitudes de los clientes y enviar la respuesta. Este sistema permitirá realizar peticiones de varios clientes a un mismo servidor, poder así acceder varios/as usuarios/as al mismo contenido de la base de datos. Si así se requiere este servidor podrá llamar a otros servidores para obtener otros datos.

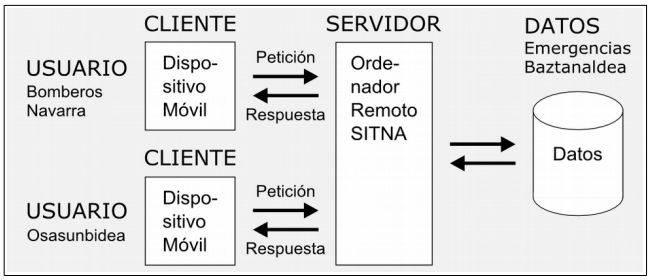

Imagen 33: Modelo Cliente – Servidor utilizado en el proyecto

- En esta red cliente-servidor los datos estarán conectados a los servidores por un sistema gestor de base de datos. En nuestro caso este sistema será un PostegreSQL[6] con extensión PostGIS[7].

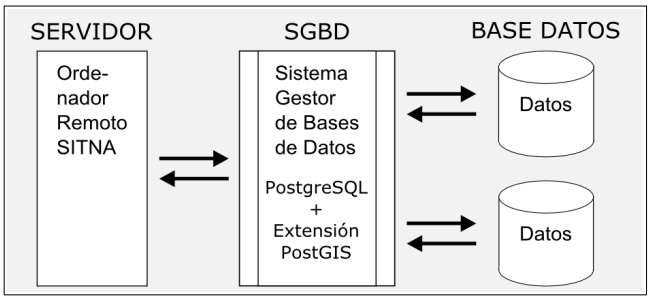

Imagen 34: El SGDB se interpone entre los datos y el servidor

6. Desarrollo de una aplicación web de tipo *web mapping*. De esta forma los/las usuarios/as de esta herramienta podrán acceder al contenido de la base de datos.
   - Al producirse una emergencia los/las bomberos/ras accederán a esta aplicación web. Por lo tanto, el diseño y funcionalidad de esta aplicación tendrá que estar orientado a las exigencias requeridas por este servicio de emergencia.
   - Esta aplicación se desarrollará a partir del *API SITNA*[8]. Por lo tanto, el *API SITNA* ofrecerá una interfaz sencilla de utilizar que incluirá un visualizador con mapas interactivos.
     - Este visualizador de mapas contendrá distintos mapas de fondo de IDENA (*Infraestructura de Datos Espaciales de Navarra*).

6 *PostgreSQL* es un Sistema de gestión de bases de datos relacional orientado a objetos [definición PostgreSQL, Wikipedia].
7 *PostGIS* es un módulo que añade soporte de objetos geográficos a la base de datos objeto-relacional *PostgreSQL*, convirtiéndola en una base de datos espacial para su utilización en un *Sistema de Información Geográfica* [definición PostGIS, Wikipedia].
8 API SITNA es un API (interfaz de programación de aplicaciones) basada en el lenguaje de programación JavaScript y concretamente en la librería libre de visualización de mapas OpenLayers 3.

- También tendrá varias funcionalidades de navegación: zoom, mapa de situación, medición, opción de búsquedas, etc.
- En ella se podrá incluir información geográfica (en forma de capas), procedente de IDENA o de otras fuentes (mediante servicios estándar *OGC*[9]).
- También habrá la posibilidad de incluir ficheros *KML*[10].

7. Necesidad de un buscador: la aplicación contendrá un servicio de localización de diseminados donde se obtendrá información espacial asociada a la ubicación de emergencias. De esta manera, al surgir un aviso de emergencia se podrá ubicar con exactitud la zona de actuación.

- La principal finalidad de esta aplicación es encontrar sin confusiones un determinado diseminado. Esta búsqueda se realizará de forma sencilla y los resultados de la búsqueda contendrán información relativa a la vivienda y sus habitantes.
- Al seleccionar una de los resultados de la búsqueda se podrá consultar la información geográfica y el metadato asociada a cada resultado.
- Esta lista de diseminados contendrán mínimamente una ruta de acceso. Podrán existir varias rutas de acceso, y para elegir la ruta más óptima se podrá consultar la información asociada a cada una de ellas.
  - En cada una de estas rutas se reflejará claramente qué vehículos de emergencia pueden transitar.
  - En su conjunto, esta aplicación facilitará al usuario/a una amplia información:
    1. Información relativa al diseminado: el nombre de la vivienda; nombre referido a su entorno (donde se especificará a qué barrio y pueblo pertenece); coordenadas en el sistema de referencia *ETRS89* (donde se especificará su ubicación); fotografía de la fachada del edificio; información asociada a las instalaciones de la vivienda (por ejemplo la existencia o no de un depósito de gas); información asociada a almacenes o pabellones aledaños a la vivienda (por ejemplo la existencia de un depósito de gasolina para maquinaria agraria).
    2. Información de sus habitantes: número de personas que habita la vivienda, sus nombres, edad y teléfono de contacto; existencia o no de vecinos/as donde se especificará su proximidad y sus teléfonos de contacto; enlaces externos donde se permitirá consultar contenido procedente de historial clínico, antecedentes, etc.
    3. Información relacionada con la accesibilidad: lista de tipo de vehículos (del parque de bomberos) que puede transitar por la ruta para acceder al diseminado; características y limitaciones de la transitabilidad para este acceso.

9 Open Geospatial Consortium, OGC. Su finalidad es la definición de estándares abiertos e interoperables dentro de los Sistemas de Información Geográfica (SIG / GIS) y de la Word Wide Web (W3) [definición OGC, Wikipedia].

10 Keyhole Markup Language. KML es un lenguaje de marcado basado en la estructura XML (Extensible Markup Language) para representar datos geográficos en tres dimensiones [definición KML, Wikipedia]. Los ficheros KMZ serán la versión comprimida de los ficheros KML.

- Toda esta información podrá estar disponible utilizando un servicio CSW[11] (obtenido desde el servidor GeoNetwork).

8. Esta aplicación también tendrá un servicio de descarga donde se podrá obtener la ruta de acceso del diseminado localizado anteriormente.
   - Después de realizar la búsqueda del diseminado donde se ha producido la incidencia y seleccionar la ruta de acceso se podrá obtener esta ruta de la base de datos: se obtendrá el componente espacial como la información relacionada a ella. Toda esta información se podrá consultar en el trascurso del desplazamiento y en todo momento (independientemente de la existencia de la conexión a Internet).

9. Esta aplicación tendrá un servicio de visualización donde se podrá visualizar el compone espacial de los datos utilizados en la aplicación y otros servicios web OGC estándar.
   - La aplicación contendrá un *Geo Visor*, basado en el visor de *API SITNA*. Por lo tanto su funcionalidad estará basada (entre otras opciones) en *OpenLayers*[12].
   - En este visor el usuario podrá superponer capas, hacer búsquedas, consultar información, añadir mapas externos y hacer operaciones básicas de navegación.
   - Habrá un listado de capas disponibles donde se podrá visualizar *información espacial*[13] de la zona donde transcurre la ruta y de los alrededores del diseminado: ortofotos (cacheadas), límites administrativos, límites de los núcleos rurales (y de barrios), ubicación de instalaciones relacionadas con emergencias (parque de bomberos, centros sanitarios, etc.), carreteras, ríos, etc.
   - La información relativa a emergencias estará disponible en el visor utilizando *servicios WMS*[14] o *WFS*[15] obtenidos desde el servidor *GeoServer*[16] del proyecto.

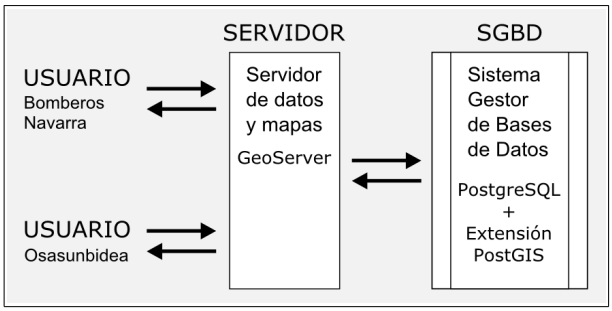

Imagen 35: Servicio WFS y WMS a partir del servidor GeoServer

- Este visor al estar basado en el visor de *API SITNA* se podrá utilizar en modo desconectado (sin conexión a internet).

11 Catalog Service Web, CSW. Servicio obtenido desde un servidor de catalogación de recursos referenciados al espacio geográfico GeoNetwork.
12 Es una biblioteca de JavaScript de código abierto para mostrar mapas interactivos en los navegadores web [definición OpenLayers, Wikipedia].
13 La información geoespacial utilizada es pública (su contenido es Open Data), procedente de IDENA (Infraestructura de Datos Espaciales de Navarra).
14 Web Map Service (Servicio de Visualización de Mapas), WMS. Servicio obtenido desde un servidor de datos y mapas GeoServer.
15 Web Features Service, WFS. Servicio obtenido desde un servidor de datos y mapas GeoServer.
16 Diseñado para la interoperabilidad, publica datos de las principales fuentes de datos espaciales usando estándares abiertos. Permite a los usuarios compartir y editar datos geoespaciales [definición GeoServer, Wikipedia].

10. Esta aplicación al contener estos tres servicios conseguirá ofrecer técnicamente una herramienta útil para resolver el problema propuesto en este proyecto.

ANEXO 2: Análisis del archivo de muestra de datos

Índice del Anexo 2 – Análisis y diseño del sistema:

A2.1. Análisis del archivo de muestra de datos

Los datos recopilados por los *Bomberos de Navarra* están en un archivo con formato KMZ. Este formato es un fichero KML comprimido en formato ZIP, y por tanto habrá que realizar su descompresión para así poder analizar su contenido.

Al obtener el fichero KML se utilizará cualquier programa de edición de texto plano para analizar su contenido. El lenguaje KML sigue una estructura XML y es un lenguaje de etiquetas, por tanto se podrá analizar su estructura de la información con un editor de texto. Para realizar esta operación, en nuestro caso se ha utilizado el software *Notepad++*.

El formato de fichero KML es utilizado habitualmente para mostrar datos geográficos en softwares de navegación. Por ello, también se utilizará el software *Google Earth* para analizar la estructura y el contenido del fichero.

1. Análisis del archivo KML

Para facilitar la comprensión y el análisis del código KML se recomienda modificar la configuración de la visualización del software *Notepad++*. A continuación, se indicarán las opciones del menú utilizadas para ello:

- Cerrar los niveles de etiquetado:

  View > *Collapse Level* > 3

  View > *Collapse Level* > 4

  View > *Collapse Level* > 5

- Facilitar la visualización del código:

  View > *Show Symbol* > *Show Indent Guide*

  View > *Word Wrap*

- Comparar el código de diferentes partes de etiquetado:

  View > *Move/Clone Current Document* > *Clone* to *Other View*

1.1. Cabecera / declaración del documento KML

La información situada en la primera línea del documento KML indica la versión del estándar XML utilizada: en este caso el valor es de “1.0”. También se indica el juego de caracteres del documento: en este caso es “UTF-8”.

```
1     <?xml version="1.0" encoding="UTF-8"?>
2     <kml xmlns="http://www.opengis.net/kml/2.2" xmlns:gx="
      http://www.google.com/kml/ext/2.2" xmlns:kml="http://www.opengis.net/kml/2.2"
      xmlns:atom="http://www.w3.org/2005/Atom">
3     <Document>
16759 </kml>
```

Imagen 36: Cabecera y declaración del documento KML en *Notepad++*

En la siguiente línea se especifica la versión estándar del KML. Esta información indica que este documento contiene un formato de estándar abierto para realizar el intercambio de información geográfica. Como se aprecia la versión KML utilizada es “2.2”.

1.2. Elementos Name y Schema

Los primeros elementos que se pueden identificar en el documento son los elementos delimitados por las etiquetas *<name>* y *<Schema>*.

```
<name>CBasaburua Disem.Caserios.kmz</name>
<Schema name="basaburua" id="basaburua">
    <SimpleField type="string" name="CODMUN"></SimpleField>
    <SimpleField type="string" name="ID"></SimpleField>
    <SimpleField type="string" name="POLIGONO"></SimpleField>
    <SimpleField type="string" name="PARCELA"></SimpleField>
    <SimpleField type="string" name="IDENVIA"></SimpleField>
    <SimpleField type="float" name="X"></SimpleField>
    <SimpleField type="float" name="Y"></SimpleField>
    <SimpleField type="string" name="CALLE"></SimpleField>
    <SimpleField type="float" name="NUMERO"></SimpleField>
    <SimpleField type="string" name="LETRA"></SimpleField>
    <SimpleField type="string" name="CASA"></SimpleField>
    <SimpleField type="string" name="PUEBLO"></SimpleField>
</Schema>
```

Imagen 37: El código KML de los elementos Name y Schema en *Notepad++*

El primer elemento *Name* se utilizará para definir el nombre del documento: *C Basaburua Diseminados Caserios*. El elemento *Schema* se utilizará para definir la estructura de los datos almacenados en este documento. En este caso, esta estructura será el de la información relativa a los diseminados de la zona de *Basaburua*.

En la Imagen 37 se representa un diseminado que utiliza esta estructura de datos “Basaburua”. En ella se puede apreciar las variables definidas en el elemento *Schema*:

CASA: nombre de la casa.

CODMUN: código del municipio al que pertenece la entidad.

ID: identificador de la población a que pertenece.

POLIGONO, *PARCELA*: nombre del polígono y nombre de parcela.

IDENVIA: identificador de la vía.

X, *Y*: coordenadas X e Y en ETRS 89 – UTM Huso 30 N.

CALLE, *NUMERO*, *LETRA*: nombre de la calle, número y letra del portal.

PUEBLO: denominación oficial del pueblo a que pertenece dicha entidad.

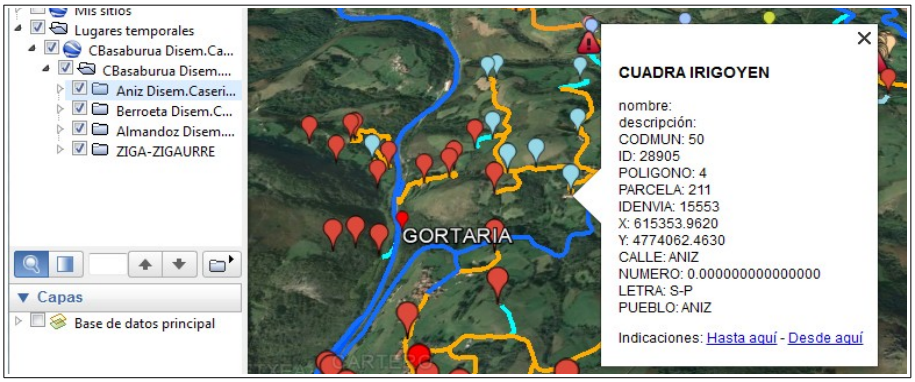

Imagen 38: Datos de un diseminado con Schema Basaburua en *Google Earth*

1.2.1. Diferentes estructuras de datos para un mismo tipo de diseminado

Al analizar el contenido del documento KML se han encontrado varias estructuras de datos utilizados para representar la información de Destinos. En el apartado anterior se ha hecho referencia a la estructura de datos "Basaburua", donde se utiliza la etiqueta Schema para definirlo.

En la Imagen 39 se aprecia la existencia de otra estructura de datos llamada "Basaburua-Completo". Pero en este caso no existe ningún etiquetado Schema con el atributo de identificación "BasaburuaCompleto", por lo tanto no está definida.

Al comparar las variables que contiene la estructura *BasaburuaCompleto* y la estructura *Basaburua* se aprecian ciertas diferencias. La estructura "Basaburua Completo" contiene: coordenadas en dos sistemas de referencia: 25830 se refiere al sistema ETRS 89 / UTM huso 30 N; y 4326 se refiere al sistema WGS 84; una variable referida al tipo de diseminado; y no incluye el nombre de la casa.

```
<open>1</open>
<styleUrl>#StyleBerroeta2</styleUrl>
<ExtendedData>
    <SchemaData schemaUrl="#basaburua">
        <SimpleData name="CODMUN">50
        </SimpleData>
        <SimpleData name="ID">28944
        </SimpleData>
        <SimpleData name="POLIGONO">4
        </SimpleData>
        <SimpleData name="PARCELA">243
        </SimpleData>
        <SimpleData name="IDENVIA">15559
        </SimpleData>
        <SimpleData name="X">614464.2220
        </SimpleData>
        <SimpleData name="Y">4773885.3650
        </SimpleData>
        <SimpleData name="CALLE">BERROETA
        </SimpleData>
        <SimpleData name="NUMERO">
        0.000000000000000</SimpleData>
        <SimpleData name="LETRA">S-P
        </SimpleData>
        <SimpleData name="CASA">GORTARIA
        </SimpleData>
        <SimpleData name="PUEBLO">BERROETA
```

```
6899    <ExtendedData>
6900        <SchemaData schemaUrl=
            "#CBasaburuaCompleto">
6901            <SimpleData name="Pol¡gono">2
                </SimpleData>
6902            <SimpleData name="Parcela">143
                </SimpleData>
6903            <SimpleData name="Idenvia">15568
                </SimpleData>
6904            <SimpleData name="X_25830">612585.437
                </SimpleData>
6905            <SimpleData name="Y_25830">4771842.528
                </SimpleData>
6906            <SimpleData name="IDPueblo">170
                </SimpleData>
6907            <SimpleData name="Pueblo">Almandoz
                </SimpleData>
6908            <SimpleData name="Calle">IRAPERRIALDE
                </SimpleData>
6909            <SimpleData name="longitud_4326">
                -1.61669235234598</SimpleData>
6910            <SimpleData name="latitud_4326">
                43.0909465402234</SimpleData>
6911            <SimpleData name="Tipo">Diseminado
                </SimpleData>
6912        </SchemaData>
6913    </ExtendedData>
```

Imagen 39: Contenido de dos diseminados con Schema BasaburuaCompleto en *Notepad++*

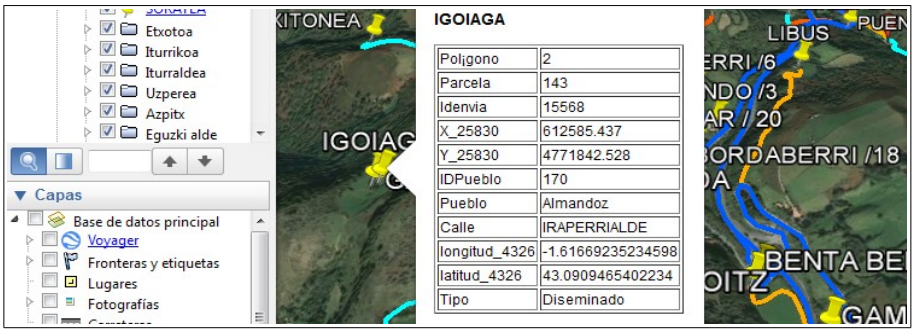

Imagen 40: Visualización de datos con estructura BasaburuaCompleto en *Google Earth*

Existen diseminados con otro tipo de estructura de datos, pero en este caso no se especifica el nombre de la estructura: son los diseminados del barrio de Zigaurre del pueblo de Ziga. En la Imagen 41 se aprecia la estructura de datos de estos diseminados, e incluyen una variable que contiene la imagen de la fachada de la vivienda. Estas imágenes se han añadido al mapa utilizando direcciones URL públicas ubicadas en una cuenta Google.:

Imagen 41: Visualización del tipo de estructura de datos utilizada en la zona Zigaurre

Conclusión: necesidad de definir una estructura única para todos los datos asociados a diseminados, y por consiguiente, cada uno de los diseminados del documento KML tendría que utilizar esta estructura definida.

1.2.2. Diferentes formas de asignar valores a variables de datos

Existen diferentes formas para asignar valores a los datos asociados a los diseminados. En la Imagen 39 se apreciaba cómo se asignaba los valores haciendo referencia a una estructura de datos, pero en todos los casos no se realiza así. En la Imagen 42 se aprecia el caso del diseminado donde la asignación se realiza dentro del elemento CDATA en la etiqueta Descripción. Esta asignación de los valores sigue el orden y mantiene el mismo tipo de contenido que la estructura de datos “Basaburua”. En la Imagen 43 se puede apreciar otro ejemplo de este tipo de asignación pero en este caso la asignación de valores siguie otro tipo de estructura de datos, no predefinida.

Imagen 42: Código KML referente a la visualización de datos del diseminado en *Notepad++*

Imagen 43: Otro ejemplo referente a la visualización de datos del diseminado en *Notepad++*

Conclusión: necesidad de definir una sola forma de asignación de valores en todo el documento.

1.3. Elementos Style y StyleMap

Estos elementos son los relacionados con la visualización de los objetos de marca de posición que representan diseminados y rutas de acceso.

1. Estilo del diseminado: se especificará el aspecto del icono en el mapa.

   En la Imagen 44 se puede apreciar el código del estilo *StyleBerroeta2* donde el estilo de un punto está definido por dos tipos de estilos: *StyleBerroeta0* y *StyleBerroeta1*.

```
<Folder>
    <name>Gortaria</name>
    <Placemark>
        <name>GORTARIA</name>
        <open>1</open>
        <styleUrl>#StyleBerroeta2</styleUrl>
        <ExtendedData>
            <SchemaData schemaUrl="#basaburua">
                <SimpleData name="CODMUN">50
                </SimpleData>
                <SimpleData name="ID">28944
                </SimpleData>
                <SimpleData name="POLIGONO">4
                </SimpleData>
                <SimpleData name="PARCELA">243
                </SimpleData>
                <SimpleData name="IDENVIA">15559
                </SimpleData>
                <SimpleData name="X">614464.2220
                </SimpleData>
                <SimpleData name="Y">4773885.3650
                </SimpleData>
                <SimpleData name="CALLE">BERROETA
                </SimpleData>
                <SimpleData name="NUMERO">
                0.000000000000000</SimpleData>
                <SimpleData name="LETRA">S-P
                </SimpleData>
                <SimpleData name="CASA">GORTARIA
                </SimpleData>
                <SimpleData name="PUEBLO">BERROETA
                </SimpleData>
            </SchemaData>
        </ExtendedData>
        <Point>
            <coordinates>
            -1.593193918674978,43.10905525136183,0
            </coordinates>
        </Point>
    </Placemark>
    <Placemark>
```

```
1287    <Style id="sn_503-wht-blank_maps00">
1299    <Style id="sh_ylw-pushpin62">
1312    <Style id="StyleBerroeta0">
1313        <IconStyle>
1314            <color>ff0000ff</color>
1315            <scale>0.75</scale>
1316            <Icon>
1317                <href>
                    http://www.gstatic.com/mapspro/images/stock/503-wht
                    -blank_maps.png</href>
1318            </Icon>
1319            <hotSpot x="0.5" y="0.5" xunits="fraction" yunits=
                "fraction"/>
1320        </IconStyle>
1321    </Style>
1322    <Style id="line-1267FF-5-nodesc-highlight56">
2994    <Style id="StyleBerroeta1">
2995        <IconStyle>
2996            <color>ff0000ff</color>
2997            <scale>0.75</scale>
2998            <Icon>
2999                <href>
                    http://www.gstatic.com/mapspro/images/stock/503-wht
                    -blank_maps.png</href>
3000            </Icon>
3001            <hotSpot x="0.5" y="0.5" xunits="fraction" yunits=
                "fraction"/>
3002        </IconStyle>
3003    </Style>
3004    <Style id="line-1267FF-5-nodesc-normal30110">
3367    <StyleMap id="StyleBerroeta2">
3368        <Pair>
3369            <key>normal</key>
3370            <styleUrl>#StyleBerroeta1</styleUrl>
3371        </Pair>
3372        <Pair>
3373            <key>highlight</key>
3374            <styleUrl>#StyleBerroeta0</styleUrl>
3375        </Pair>
3376    </StyleMap>
3377    <Style id="sn_ylw-pushpin03">
```

Imagen 44: Código KML referente a la visualización de un diseminado en *Notepad++*

En las etiquetas de estos dos estilos existe una etiqueta llamada *IconStyle*. En esta etiqueta se definirá el estilo del símbolo del diseminado: la escala y la imagen del símbolo que se va a utilizar en el mapa. En la Imagen 45 se representa el aspecto del objeto espacial definido con esta etiqueta.

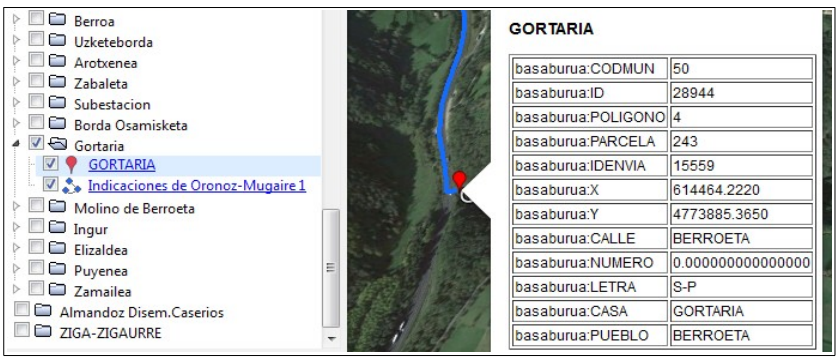

Imagen 45: Aspecto de un diseminado y su ruta de acceso en *Google Earth*

2. Estilo de la ruta: se especificará el aspecto de la ruta en el mapa.

   En la Imagen 46 se puede apreciar el código del estilo *line-1267FF-5-nodesc7014* donde el estilo de una polilínea está definido por dos tipos de estilos: *line-1267FF-5-nodesc-normal130153* y *line-1267FF-5-nodesc-highlight140139*.

```
<Folder>
<Folder>
<Folder>
<Folder>
<Folder>
<Folder>
<Folder>
<Folder>
<Folder>
<Folder>
<Folder>
<Folder>
<Folder>
    <name>Gortaria</name>
    <Placemark>
        <name>GORTARIA</name>
        <open>1</open>
        <styleUrl>#StyleBerroeta2</styleUrl>
        <ExtendedData>
        <Point>
    </Placemark>
    <Placemark>
        <name>Indicaciones de Oronoz-Mugaire 1</name>
        <open>1</open>
        <styleUrl>#line-1267FF-5-nodesc7014</styleUrl>
        <LineString>
            <tessellate>1</tessellate>
            <coordinates>
        </LineString>
    </Placemark>
</Folder>
```

```
261     <StyleMap id="line-1267FF-5-nodesc70141">
262         <Pair>
263             <key>normal</key>
264             <styleUrl>#line-1267FF-5-nodesc-normal30153</styleUrl>
265         </Pair>
266         <Pair>
267             <key>highlight</key>
268             <styleUrl>#line-1267FF-5-nodesc-highlight140139
                </styleUrl>
269         </Pair>
270     </StyleMap>
271     <StyleMap id="msn_ylw-pushpin41">
779     <Style id="line-1267FF-5-nodesc-normal30153">
780         <BalloonStyle>
781             <text><![CDATA[<h3>$[name]</h3>]]></text>
782         </BalloonStyle>
783         <LineStyle>
784             <color>ff00aaff</color>
785             <width>4</width>
786         </LineStyle>
787     </Style>
788     <Style id="s_ylw-pushpin0000">
1214    <Style id="line-1267FF-5-nodesc-highlight140139">
1215        <BalloonStyle>
1216            <text><![CDATA[<h3>$[name]</h3>]]></text>
1217        </BalloonStyle>
1218        <LineStyle>
1219            <color>ff00aaff</color>
1220            <width>4</width>
1221        </LineStyle>
1222    </Style>
```

Imagen 46: Código KML referente a la visualización de una ruta en *Notepad++*

Como se aprecia, en las etiquetas de estos dos estilos existe una etiqueta llamada *LineStyle*. En esta etiqueta se definirá el estilo de línea de los tramos. que va tener la ruta en el mapa. En la Imagen 45 se representa el aspecto del objeto espacial definido con esta etiqueta.

1.3.1. Diferentes ubicaciones para la declaración de los estilos

Existen diferentes ubicaciones para la declaración de estilos de objetos. En la Imagen 47 se puede apreciar la declaración de estilos situado en diferentes niveles del etiquetado.

```
<Style id="sn_ylw-pushpin03">
<Style id="line-1267FF-5-nodesc-highlight14017">
<StyleMap id="line-1267FF-5-nodesc7010">
<Style id="sh_ylw-pushpin27">
<Folder>
    <name>CBasaburua Disem.Caserios</name>
    <Folder>
    <Folder>
    <Folder>
    <Folder>
        <name>ZIGA-ZIGAURRE</name>
        <Document>
            <name>zigaurre-ziga urbano</name>
            <Style id="sh_503-wht-blank_maps20">
            <StyleMap id="msn_503-wht-blank_maps21">
            <StyleMap id="msn_ylw-pushpin19">
            <Style id="sh_503-wht-blank_maps110">
            <Style id="icon-503-F4B400-normal">
```

Imagen 47: Diferentes ubicaciones de la declaración de estilos

Conclusión: esta forma de declarar estilos dificulta el análisis y la modificación de su contenido.

1.3.2. Diferentes tipos de símbolos para diseminados

Existen diferentes símbolos para representar los diseminados en el mapa. Las imágenes de los símbolos se han definido utilizando direcciones URL públicas.

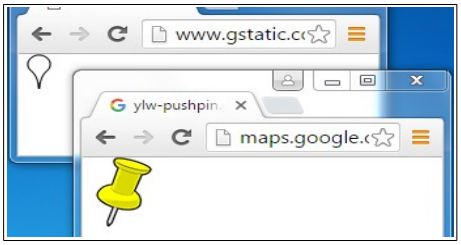

Imagen 48: Tipos de iconos utilizados para representar diseminados

Para representar a la mayoría de los diseminados se han utilizado dos imágenes de símbolos. Estas imágenes se pueden apreciar en la Imagen 48, y sus direcciones URL son las siguientes:

http://www.gstatic.com/mapspro/images/stock/503-wht-blank_maps.png

http://maps.google.com/mapfiles/kml/pushpin/ylw-pushpin.png

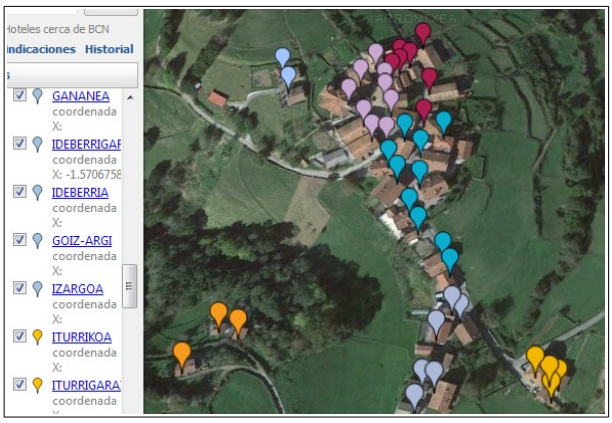

Imagen 49: Colores para representar iconos y reflejar agrupaciones

En la Imagen 49 se aprecia la utilización de diferentes colores para representar estos iconos. Estos cambios de color son debidos a diferentes campañas de captura de datos, pero en algunos casos representarán diferentes agrupaciones de diseminados:diferentes grupos o zonas que pertenecen los diseminados.

1.3.3. Otro tipo de símbolos para representar puntos en el mapa

Existen otros tipos de simbología. En la Imagen 50 se aprecia la existencia de tres tipos de iconos que no se han mencionado hasta ahora.

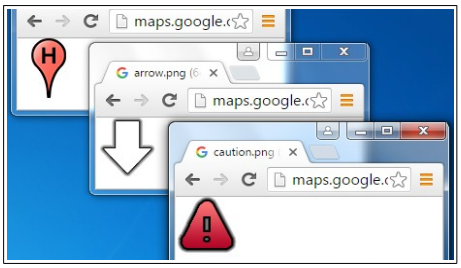

Imagen 50: Tres tipo de imágenes de iconos

Estos tres iconos representan diferentes objetos espaciales: dos de ellos se utilizarán para representar otros tipos de diseminados, al contener diferente información que los diseminados existentes; y el tercer tipo de símbolo se utilizará para representar avisos de emergencia, e indicará posibles incidencias en la ruta de acceso.

- **Iconos de diseminados con datos para emergencias**

En el mapa existen dos tipos de iconos que representan datos relacionados con emergencias. En este caso, las imágenes de los símbolos se han definido utilizando direcciones URL:

- Primer caso, icono H:

  http://maps.google.com/mapfiles/kml/paddle/H.png

- Segundo caso, icono Arrow:

  http://maps.google.com/mapfiles/kml/shapes/arrow.png

En la Imagen 51 se puede observar el primer caso. Este objeto espacial contiene asociada información referente a sus habitantes: *Casa*, *Coordenadas X e Y*, *Zona*, *Identificador*, *Grupo convivencia individuos*, *Residentes*, *Teléfono* y *Observaciones*.

Imagen 51: La información asociada al icono H

En la Imagen 51 se puede observar el segundo caso. Este objeto contiene asociada datos relacionados con los habitantes y el diseminado: *Casa*, *Coordenadas X e Y*, *Población*, *Barrio*, *Identificador*, *Grupo convivencia individuos*, *Residentes*, *Teléfono*, *Observaciones* y *Fotografía*.

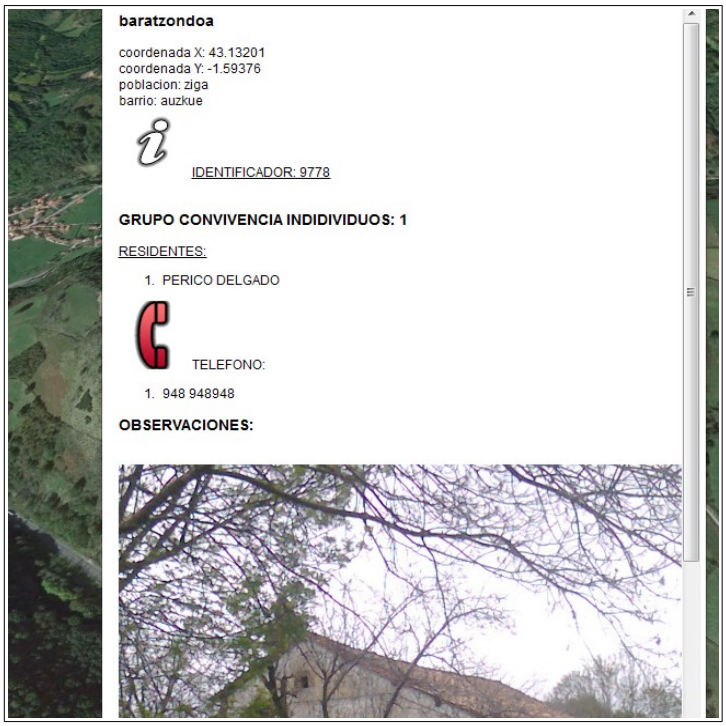

Imagen 52: La información asociada al icono Arrow

La información de los diseminados representada en la Imagen 52 se asemeja a los datos de los diseminados mencionados en el apartado 1.2.1, los diseminados del barrio de *Zigaurre*. Estos diseminados contenían esta estructura de datos (ver Imagen 41): *Casa*, *Coordenadas X e Y*, *Vía*, *Nº portal*, *Código identif.* y *Fotografía*.

- **Icono de advertencia**

Este tipo de icono hace referencia a un aviso puntual: mensaje de advertencia. En este caso, la dirección URL pública utilizada para este tipo de icono es la siguiente:

- Tercer caso, icono Caution:

  http://maps.google.com/mapfiles/kml/shapes/caution.png

Este objeto espacial sólo contiene asociada una descripción del mensaje de advertencia (ver Imagen 53 y Imagen 54).

Imagen 53: Información asociada al icono Caution

Las rutas de acceso se componen por tramos para reflejar los diferentes tipos de vías que componen el recorrido, y estos tramos contendrán información sobre el estado, la composición del suelo y la existencia de pendientes altas y curvas cerradas. Pero para ubicar las incidencias más relevantes se utilizarán puntos de aviso con mensajes de advertencia.

```
    <Placemark>
    <Placemark>
    <Placemark>
    <Placemark>
        <name>¡¡!!</name>
        <description>Fuerte pendiente y curva muy
        cerrada</description>
        <LookAt>
            <longitude>-1.578649601050197
            </longitude>
            <latitude>43.12549950145579</latitude>
            <altitude>0</altitude>
            <heading>-0.006351740152391305
            </heading>
            <tilt>0</tilt>
            <range>1304.289936349974</range>
            <gx:altitudeMode>relativeToSeaFloor
            </gx:altitudeMode>
        </LookAt>
        <styleUrl>#msn_caution0</styleUrl>
        <Point>
            <coordinates>
            -1.583111084102795,43.12545395379991,0
            </coordinates>
        </Point>
    </Placemark>
</Folder>
```

```
11963   <StyleMap id="line-F8971B-141">
11973   <Style id="line-F8971B-1-highlight73">
11979   <StyleMap id="line-1267FF-5-nodesc91">
11989   <Style id="line-F8971B-1-highlight59120">
11995   <Style id="sn_caution0">
11996       <IconStyle>
11997           <scale>1.2</scale>
11998           <Icon>
11999               <href>
                    http://maps.google.com/mapfiles/kml/sh
                    apes/caution.png</href>
12000           </Icon>
12001           <hotSpot x="0.5" y="0" xunits="fraction"
                yunits="fraction"/>
12002       </IconStyle>
12003       <LabelStyle>
12006   </Style>
12789   <StyleMap id="msn_caution0">
12790       <Pair>
12791           <key>normal</key>
12792           <styleUrl>#sn_caution0</styleUrl>
12793       </Pair>
12794       <Pair>
12795           <key>highlight</key>
12796           <styleUrl>#sh_caution0</styleUrl>
12797       </Pair>
12798   </StyleMap>
```

Imagen 54: Código KML referente a la visualización de un punto de advertencia

1.3.4. Diferentes colores para representar diferentes tipos de tramos

Los tramos que representen las carreteras contendrán un tipo de color diferente a los tramos relativos a caminos rurales. Pero entre los tramos rurales existen diferentes criterios de asignación de colores: reflejan cambios de criterio en cada campaña de obtención de datos.

Estos tramos pueden tener características y composiciones diferentes. Por ello, la parte rural del recorrido de la ruta puede estar compuestos por varias subtramos. Por ejemplo, una ruta puede estar compuesta de esta manera:

1. TRAMO CARRETERA: Indicaciones desde Oronoz-Mugaire.
2. TRAMO PISTA 1: Firme con recubrimiento de ASFALTO en buen estado.
3. TRAMO PISTA 2: Firme con recubrimiento de HORMIGÓN en buen estado.
4. TRAMO PISTA 3: Firme con recubrimiento de TODO-UNO (de granulometría muy fina). Últimos 100 metros mucha pendiente y mal estado de firme.

En algunos casos para representar todos los tramos rurales de una ruta se ha utilizado un color determinado, y en otros en cambio cada tramo contiene diferente color.

Estos tramos contienen información asociada. En la Imagen 55 se puede apreciar que un tramo incluye información sobre: las características de la vía (la composición, la información asociada a pendiente y curvas y el estado actual), la lista de vehículos que pueden transitar por el tramo, y una imagen Truck o camión (simbolizando el vehículo que puede transitar por este tramo).

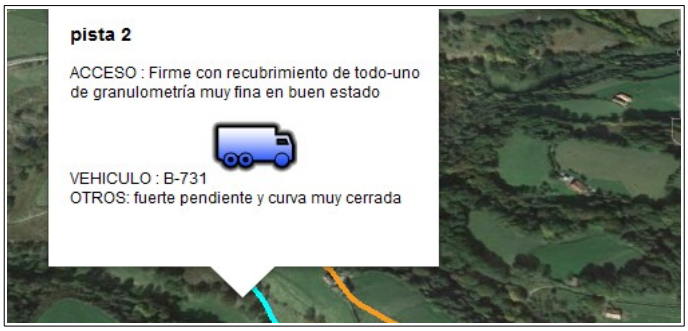

Imagen 55: Información asociada a subtramos

```
                    <name>Perotxenea</name>
                    <Placemark>
                        <name>Desde parque</name>
                        <styleUrl>#m_ylw-pushpin0001</styleUrl>
                        <LineString>
                            <tessellate>1</tessellate>
                            <coordinates>
                        </LineString>
                    </Placemark>
                    <Placemark>
                        <name>Ruta sin título</name>
                        <description><![CDATA[ACCESO: Firme con
                        recubrimiento de ASFALTO  en buen estado<br>
<img src="http://maps.google.com/mapfiles/kml/shapes/truck.png"/>
<br>VEHICULO: B-731]]></description>
                        <styleUrl>#msn_ylw-pushpin202</styleUrl>
                        <LineString>
                            <tessellate>1</tessellate>
                            <coordinates>
                        </LineString>
                    </Placemark>
                    <Placemark>
                        <name>Ruta sin título</name>
                        <description><![CDATA[ACCESO: Firme con
                        recubrimiento de HORMIGON  en buen
                        estado<br>
<img src="http://maps.google.com/mapfiles/kml/shapes/truck.png"/>
<br>VEHICULO: B-731]]></description>
                        <styleUrl>#msn_ylw-pushpin202</styleUrl>
                        <LineString>
                            <tessellate>1</tessellate>
                            <coordinates>
```

Imagen 56: Código KML referente a los subtramos de las rutas

1.3.5. Conclusiones sobre la definición de los estilos

- La declaración de los estilos se realizará en el mismo nivel de etiquetado, y uno detrás de otro. Por lo tanto, se agruparán todas las declaraciones existentes de estilos.
- Todos los diseminados tendrán que contener la misma estructura de datos, y por consiguiente contendrán el mismo tipo de simbología. Por lo tanto, se les añadirá información a los diseminados que contienen solamente información de la vivienda y a los que contienen solamente información de sus habitantes. De esta forma todos los diseminados contendrán la misma información y el mismo icono.
- Se podrá colorear de diferente manera estos iconos para representar agrupaciones (como barrios o pueblos) o para representar gráficamente una clasificación de un parámetro (como intervalos de distancia o de tiempo de desplazamiento desde el parque de bomberos).
- Los tramos que componen las rutas se colorearán dependiendo del grado de dificultad de la transitabilidad de la vía. La clasificación del grado de dificultad se establecerá a partir de

las limitaciones de maniobrabilidad y transitabilidad de los vehículos de la unidad de emergencia.

1.4. Elementos Placemark y Folder, marcas de posición y carpetas

Estos elementos *Placemark* son los que permiten marcar una posición en el mapa, y se han utilizado para representar diseminados (elementos placemark de punto) y rutas de acceso (elementos placemark de cadena de líneas).

Los elementos *Folder* se han utilizado para agrupar diferentes elementos:

1. Para agrupar varias carpetas. Existen cuatro carpetas padre que están compuestos por multitud de carpetas: *Aniz*, *Berroeta*, *Almandoz* y *Ziga-Zigaurre*. Estas carpetas conforman zonas o barrios reseñables de Baztan.
2. Para agrupar en una misma zona diferentes campañas de toma de datos. Existe la carpeta *Ziga-Zigaurre* compuesta por dos carpetas de diferentes campañas: "*Zigaurre-Ziga urbano*" y "*Ziga + Salud*".
3. Para agrupar varios elementos Placemark. Existen carpetas asociadas a un diseminado donde se agrupan todas las marcas de posición relacionadas con este diseminado. Estas carpetas contendrán el mismo nombre que la marca de posición del diseminado que contienen.

1. Carpeta (Tipo Barrio o Pueblo): Aniz Disem.Caserios.

1.1. Placemark (Point):
Jaureguia, Caseta Cazadores, Cocochea.

1.2. Carpetas (Tipo Diseminados):
Gortaria, Bordachuri, Alizkoa, Aezkoa, Marmitz, Cementerio, Sin Nombre, Gananea, Olazar, Olazarborda, Borda, Cuadra Irigoyen.

2. Carpeta (Tipo Barrio o Pueblo): Berroeta Disem.Caserios.

2.1. Placemark (Point):
Aitzurdoki, Indakoa, Arotzenea, Elizaldea, Mendicoa, Gortaria 1, Puyenea, Gamioa.

2.2. Carpetas (Tipo Diseminados):
Ontsaria, Aintzinenea, Aintze/Eneko berri, Santxenea, Cementerio, Borda berri, Maxenekoborda, Uztartekoborda, Larraburu, Lizurieta, Arrazketa, Cuadra aizkoa, Behitiko etxea, Cartero, Sin Nombre, Berroa, Uzketeborda, Arotxenea, Zabaleta, Subestacion, Borda Osamisketa, Gortaria, Molino de Berroeta, Ingur, Elizaldea, Puyenea.

3. Carpeta (Tipo Barrio o Pueblo): Almandoz Disem.Caserios.

3.1. Carpeta: Diseminados caserios.

3.1.1. Placemark (Point):
Igoiaga, Garaiko Etxeko Borda, Soratea.

3.1.2. Carpetas (Tipo Diseminados):
Etxotoa, Iturrikoa, Iturraldea, Uzperea, Azpitx, Eguzki alde, Perotxenea, Elsonea, Indakoa, Goldaburu, San Blas Benta, San Blas Txar, Ariztizelai, Ariztizelai gaina, Benta Errea/Quemada, Odolaga Benta, Taberneria, Libus, Gamioa, Toki Eder, Benta Berri, Etxe-txar, Bordaberri, Itxondo, Txokoberri, Barrendegia, Mikelperizena, Puente Marin, Abreiko, Gontxea, Etxeberria, Bertatxea, Igoroitz, Gartxitonea, Apurtxi, Azkurrieta.

Imagen 57: Estructura de carpetas y placemarks del fichero KML

En la Imagen 57 y la Imagen 58 se puede apreciar la organización o jerarquía de las carpetas de este fichero KML, y se observa en qué zonas se aglutinan las carpetas que almacenan placemarks de diseminados. También se observa en qúe zonas existen las carpetas que agrupan placemarks de un mismo tipo (de tipo punto o de tipo línea).

4. Carpeta (Tipo Campañas): Ziga-Zigaurre.

4.1. Document: Zigaurre-Ziga urbano.

4.1.1. Placemark (Point):
Eliza, Iruingoa, Berekoetxea, Sutegia, Apezetxea, Indaberea, Seroategia, Gontxea, Etxenikea, Sin texto, Mendibel, Etxezuria, Amosenea, Marinea, Baratzondoa, Arrontxea, Indakoa, Legarrea, Iturriondoa, Ostatua, Legarre-Txiki, Mayora, Arotzenea, Aldekoa, Artxea, Aldekotxeberria, Eskola, Aroztegia, Etxeberria, Agerreberea, Agerrea, Gamioa, Gorrienea, Sasternea, Gure Ametsa, Gananea, Ideberrigaraia, Ideberria, Goiz-Argi, Izargoa, Iturrikoa, Iturrigaraia, Garbilekua, Garaikoa, Garazialde, Eguzkizain-Etxea, Ziganda-Enea, Arrillo, Arotzenea Borda 2, Arotzenea Borda, Iturraldea, Garagarrea, Gamioa, Migelenea, Barrenetxe, Karakotxea, Etxandia 1, Etxandia 2-I, Etxandia 2-D, Kalainenea, Indartea, Katea, Etxeberria, Petrinenea, Gaikoetxea, Etxezaharrea, Ermita, Iturriondoa A, Iturriondoa B, Dendarienea, Elizaldea, Ideburu, Kalainenea, Mendialdea, Lerabidea.

4.1.2. Placemark (LineString):
Ruta sin título, Ruta sin título, Ruta sin título, Ruta sin título, Ruta sin título, Ruta sin título, Ruta sin título, Ruta sin título.

4.2. Document: Ziga + Salud.

4.2.1. Document: Salud.

4.2.1.1. Placemark:
Baratzondoa, Gontxea, Sabatenea.

4.2.2. Carpetas (Tipo Diseminados):
Baratzondoa unificado, Baratzondoa, Gontxea, Sabatenea, Eguzkialdea, Segonea, Sutegia, Ideberria, Garaikoa, Olariaga, Indakoa, Berrondoa, Larraldea, Egozkue, Soraburua, Elizaldea, Larregia, Munoaldea, Munoa, Haritzondoa, Behin batean, Gamioa, Bixtaeder, Arginzu, Argiñaga.

Imagen 58: Estructura de carpetas y placemarks del fichero KML

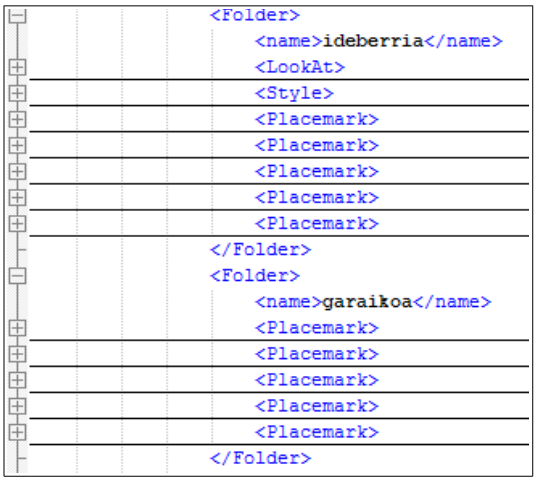

Imagen 59: Agrupación de varios elementos Placemark en *Notepad++*

En Imagen 59 se puede observar el contenido de una de las carpetas que almacenan placemarks de diseminados. En ella se aglutinan etiquetas de nombre, estilo, y varios elementos Placemark de diferente tipo: una marca de posición Point que representa el punto geográfico del diseminado, y varias marcas de posición LineString que representan los subtramos que

componen la ruta de acceso a este diseminado. También puede contener otra marca de posición Point para avisos de advertencia que representará un punto de información.

1.4.1. Problemas con la nomenclatura de los diseminados

Al analizar la estructura de este documento se han identificado varios problemas asociados a la nomenclatura:

1.4.1.1. Existencia de varios diseminados con el mismo nombre

Arotzenea, *Baratzondoa*, *Eguzkialde*, *Elizaldea*, *Etxeberria*, *Gamioa*, *Garaikoa*, *Gontxea*, *Gortaria*, *Indakoa*, *Ideberria*, *Iturrikoa*, *Iturriondoa*, etc.

Imagen 60: Dos caseríos con el mismo nombre en diferente zona

- Existen diseminados con el mismo nombre pero ubicados en diferente zona, barrio o pueblo (ver Imagen 60).
- Existen diseminados con el mismo nombre pero ubicados en la misma zona. En la Imagen 61 se puede apreciar tres casos de diseminado con el nombre de *Gortaria* ubicado cerca del pueblo *Berroeta*.

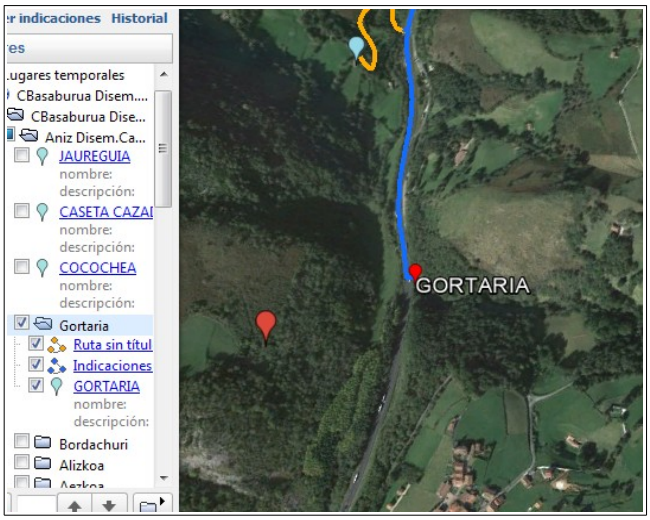

Imagen 61: Tres caseríos con el mismo nombre en una misma zona

Para evitar confusiones con el nombre de los diseminados se utilizará el nombre del propietario junto al nombre de la casa.

1.4.1.2. Existencia de varios diseminados sin ningún nombre alguno

Se han encontrado tres casos donde no existe ningún nombre de diseminado. En la Imagen 62 se puede observar uno de estos diseminados no contiene un nombre pero tiene asociada una ruta de acceso y contiene otro tipo de información. Por lo tanto, si se crea alguna incidente en este diseminado habrá dificultades para identificarlo y acceder a ella.

Imagen 62: Diseminado sin ningún nombre asociado

- La palatalización detrás de -i generalmente no se da.
  dituzte > *tuzte*

- En la declinación, -ea y -oa suenan -ia y -ua.
  etx**ea** > *etxia* ; ast**oa** > *astua*

- La armonía vocálica está muy extendida. También existe en el interior de la palabra.
  diru**a** > *dirue* ; ogi**a** > *ogie* ; negu**a** > *negue* ; uk**a**tu > *uketu*

- Como en otros dialectos navarros también aquí son frecuentes los diptongos ascendentes.
  andr**ea**k > *ándriak*

- Es muy corriente (y es exclusivo de las hablas altonavarras) que los verbos pierdan la primera vocal (aféresis). A veces se da también en sustantivos.
  ebaki > *baki* ; **e**karri > *karri* ; **i**kusi > *kusi* ; **e**man > *man* ; **e**torri > *torri* ;
  emakume > *makome* ; **e**mazte > *mazte*

- También es corriente que se pierda una vocal en el interior de la palabra (sincopa). Sucede en un espacio similar al de la aféresis.
  hirugarr**e**na > *hirugarna* ; abis**a**tu > *abistu* ; probetx**a**tu > *probextu* ;
  ekarr**i**ko > *karko* ; ikus**i**ko > *kusko* ; engain**a**tu > *engaintu*

- El diptongo -eu- evoluciona a -au- en algunas ocasiones.
  d**eu**s > *daus*

- En palabras como joan, jarri, jasa, jendea, jaten... utilizan el sonido y-.
  joan > *yan* ; **j**arri > *yarri* ; **j**asa > *yasa* ; **j**endea > *yendia* ; **j**aten > *yaten*

Imagen 63: Características de la fonética del baztanés

1.4.1.3. Confusiones producidas por el desconocimiento del euskera

La toponimia del entorno de trabajo proviene del vascuence, y es muy probable que las nomenclaturas de las viviendas contengan palabras compuestas provenientes del euskera. Concretamente, en el Valle de Baztan se utiliza un dialecto del vascuence llamado baztanés.

Este dialecto se habla en todas las localidades de este valle, y la pronunciación de las palabras es diferente del resto de zonas vascófonas de Navarra.

Por ello, una persona residente en este valle al realizar una llamada al 112 para dar aviso de una incidencia es probable que haga uso del *baztanés* para referirse a un diseminado o zona, aunque esa persona se dirija en castellano. Por lo tanto, para evitar posibles confusiones habrá que tener en cuenta las características de la *fonética del baztanés*[17] (ver Imagen 63).

1.4.1.4. Confusiones producidas al realizar el proceso de búsqueda

Existen diferentes formas de escribir un mismo nombre de una vivienda, por lo tanto habrá que identificar las posibles confusiones que se podrían suceder. En este apartado se analizarán todos los nombres de los diseminados utilizados en el fichero KML, y se propondrán posibles cambios para evitar errores: la modificación de algunas nomenclaturas, la introducción de sinónimos, la utilización de técnicas de búsquedas fonéticas, etc.

La utilización de la palabra *Borda*, *Etxea* y otras referencias a tipos de edificios o edificaciones

- La palabra "Borda" se utiliza tanto en castellano como en euskera, con el mismo significado.

En el fichero KML, los casos de nomenclaturas de diseminados que utilizan esta palabra son los siguientes: *Bordachuri*, *Olazarborda*, *Borda*, *Borda berri*, *Maxenkoborda*, *Uztartekoborda*, *Uzketeborda*, *Borda Osamisketa*, *Garaiko Etxeko Borda*, *Bordaberri*.

Estos nombres de diseminados son composiciones de varias palabras, y existen formas diferentes de escribirlos, como *Borda Osamisketa* y *Olazarborda*. Por tanto, para evitar errores en el proceso de búsqueda de diseminados se propone modificar las nomenclaturas de la siguiente manera (ver Tabla 2):

1. Si la nomenclatura contiene "Borda" al principio del nombre se escribirá todo junto.

   Por ejemplo, *Borda berri* > *Bordaberri*.

2. Si la nomenclatura contiene "Borda" al final del nombre se escribirán en dos partes diferenciadas.

   Por ejemplo, *Uzketaborda* > *Uzketa Borda*.

REGISTRO	PROPUESTA	Barrio o Pueblo
Bordachuri	Bordatxuri	Aniz
Olazarborda	Olazar Borda	Aniz
Borda	Borda	Aniz
Borda berri	Bordaberri	Berroeta
Maxenekoborda	Manexeneko Borda	Berroeta
Uztartekoborda	Uztarteko Borda	Berroeta
Uzketeborda	Uzkete Borda	Berroeta
Borda Osamisketa	Osamisketa Borda	Berroeta
Garaiko Etxeko Borda	Garaiko Etxeko Borda	Almandoz
Bordaberri	Bordaberri	Almandoz

Tabla 2: Modificaciones propuestas para utilizar el término Borda en las nomenclaturas

17 Información recogida del *Mediateka_* (Mediateca del Vascuence de Navarra). Con licencia Copy Right 2017 de Eskarabidea (Gobierno de Navarra).

- La palabra “Etxea” (o “Etxe”): significa “Casa” en euskera.

Las nomenclaturas de diseminados que contienen esta palabra se escriben de diferente manera. Por lo tanto, se propone modificarlas con el mismo criterio que en el caso de “Borda”:

1. Si “Etxe(a)” lo contiene al principio del nombre se escribirá todo junto.
2. Si “Etxe(a)” lo contiene al final se escribirá en dos partes diferenciadas.

REGISTRO	PROPUESTA	Barrio o Pueblo
Behitiko etxea	Behitiko Etxea	Berroeta
Garaiko Etxeko Borda	Garaiko Etxeko Borda	Almandoz
Etxe-txar	Etxetxar	Almandoz
Etxeberria	Etxeberria	Almandoz
Berekoetxea	Bereko Etxea	Ziga-Zigaurre
Apezetxea	Apez Etxea	Ziga-Zigaurre
Etxenikea	Etxenikea	Ziga-Zigaurre
Etxezuria	Etxezuria	Ziga-Zigaurre
Etxeberria	Etxeberria	Ziga-Zigaurre
Eguzkizain-Etxea	Eguzkizain Etxea	Ziga-Zigaurre
Etxeberria	Etxeberria	Ziga-Zigaurre
Gaikoetxea	Garaiko Etxea	Ziga-Zigaurre
Etxezaharrea	Etxezaharrea	Ziga-Zigaurre

Tabla 3: Modificaciones propuestas para utilizar el término Etxea en las nomenclaturas

Hay que tener en cuenta que esta palabra en el dialecto baztanés se pronuncia *Etxia*, en vez de *Etxea*.

- Las palabras como “Venta” (Benta), “Hotel” (Hotela), “Agroturismo” (Landa Etxea), etc.

Se propone escribir de la misma manera que en las nomenclaturas que contienen “Borda” y “Etxea”.

REGISTRO	PROPUESTA	Barrio o Pueblo
San Blas Benta	San Blas Benta	Almandoz
Odolaga Benta	Odolaga Benta	Almandoz
Benta Errea / Quemada	Errea / Quemada Benta	Almandoz
Benta Berri	Bentaberri	Almandoz

Tabla 4: Modificaciones propuestas para utilizar el término Benta en las nomenclaturas

- Las nomenclaturas que contienen solamente la tipología del edificio, y no contienen un nombre.

En el fichero KML existen referencias a diferentes tipologías de edificios sin ninguna nomenclatura asociada. En algunos casos, estas palabras sólo están escritas en castellano y otras sólo en euskera (ver Tabla 5). En estos casos se propone escribir en bilingüe (castellano y baztanés).

REGISTRO	PROPUESTA	Barrio o Pueblo
Caseta Cazadores	Caseta Cazadores / Ehizilari Etxola	Aniz
Cementerio	Cementerio / Hilerria	Aniz
Cementerio	Cementerio / Hilerria	Berroeta
Cartero	Cartero / Postaria	Berroeta
Subestacion	Subestacion	Berroeta
Molino de Berroeta	Molino / Errota	Berroeta
Eliza	Iglesia / Eliza	Ziga-Zigaurre
Ostatua	Posada / Ostatua	Ziga-Zigaurre
Eskola	Escuela / Eskola	Ziga-Zigaurre
Ermita	Ermita	Ziga-Zigaurre

Tabla 5: Modificaciones propuestas para la utilización de la tipología de los edificios

- Las nomenclaturas que contienen la tipología del edificio y un nombre (la toponimia o el nombre del negocio o propietario).

En estos casos se propone primero escribir el nombre y luego la palabra referente a su tipología.

REGISTRO	PROPUESTA	Barrio o Pueblo
Cuadra Irigoyen	Irigoien Cuadra / Ukuilua	Aniz
Cuadra aizkoa	Aizkoa Cuadra / Ukuilua	Berroeta
Puente Marin	Marin Puente / Zubia	Almandoz

Tabla 6: Modificaciones propuestas para utilización nombres negocio y emplazamientos

La utilización de la *TX*, *TS*, *TZ* y *K*

- La utilización del TX:

El sonido TX del euskera se pronuncia igual que el CH del castellano. En el fichero KML, los nombres de los diseminados están escritos en algunos casos con TX y en otros con CH (ver Tabla 7). Esto puede dificultar la búsqueda de estos diseminados. Por lo tanto, habrá que decidir cuál de estas dos formas de escribir es la más adecuada para la base de datos. Se recomienda escribir de la misma manera en todos los registros.

- La utilización del TS:

El sonido TS del euskera se pronuncia muy parecido al TX del euskera. Por ello es fácil confundir los dos casos y utilizar erróneamente TX en la búsqueda. En este caso, no se propone modificaciones (ver la Tabla 8), pero habrá que tener en cuenta la posible confusión de TX y TS en la búsqueda.

REGISTRO	PROPUESTA	Barrio o Pueblo
Cocochea	Kokotxea	Aniz
Bordachuri	Bordatxuri	Aniz
Santxenea	Santxenea	Berroeta
Arotxenea	Gontxea	Berroeta
Etxotoa	Bertatxea	Almandoz
Perotxenea	Arrontxea	Almandoz
San Blas Txar	San Blas Txar	Almandoz
Etxe-txar	Etxetxar	Almandoz
Itxondo	Itxondo	Almandoz
Txokoberri	Txokoberri	Almandoz
Gontxea	Gontxea	Almandoz
Bertatxea	Bertatxea	Almandoz
Gartxitonea	Gartxitonea	Almandoz
Apurtxi	Apurtxi	Almandoz
Gontxea	Gontxea	Ziga-Zigaurre
Arrontxea	Arrontxea	Ziga-Zigaurre
Legarre-Txiki	Legarretxiki	Ziga-Zigaurre
Artxea	Artxea	Ziga-Zigaurre
Aldekotxeberria	Aldekotxeberria	Ziga-Zigaurre
Gontxea	Gontxea	Ziga-Zigaurre

Tabla 7: Modificaciones propuestas para utilizar el TX

REGISTRO	PROPUESTA	Barrio o Pueblo
Ontsaria	Ontsaria	Berroeta
Gure Ametsa	Gure Ametsa	Ziga-Zigaurre

Tabla 8: No se proponen modificaciones al utilizar el TS

- La utilización del TZ:

El sonido TZ del euskera se pronuncia igual que el TZ del alemán. Aunque este sonido no exista en castellano es fácil de identificarlo. Entre los registros analizados se propone modificar algunos casos donde se prevé que la nomenclatura debería llevar TZ (ver la Tabla 9). Es el caso de Azpitx y Arginzu.

REGISTRO	PROPUESTA	Barrio o Pueblo
Marmitz	Marmitz	Aniz
Aitzurdoki	Aitzurdoki	Berroeta
Arotzenea	Arotzenea	Berroeta
Aintzinenea	Aintzinenea	Berroeta
Aintze/Eneko berri	Aintze/Eneko berri	Berroeta
Azpitx	Azpitz	Almandoz
Igoroitz	Igoroitz	Almandoz
Baratzondoa	Baratzondoa	Ziga-Zigaurre
Arotzenea	Arotzenea	Ziga-Zigaurre
Baratzondoa	Baratzondoa	Ziga-Zigaurre
Haritzondoa	Haritzondoa	Ziga-Zigaurre
Arginzu	Argintzu	Ziga-Zigaurre

Tabla 9: Modificaciones propuestas para utilizar el TZ

- La utilización del K:

En el fichero KML existen nombres de diseminados escritas con K y otras con la letra C: existen dos casos donde se escriben con la letra *C,* y en 29 casos donde se ha utilizado la letra *K*.

Para evitar confusiones en el proceso de búsqueda habrá que escribirlos de la misma forma.

REGISTRO	PROPUESTA	Barrio o Pueblo
Cocochea	Kokotxea	Aniz
Alizkoa	Alizkoa	Aniz
Aezkoa	Aezkoa	Aniz
Aitzurdoki	Aitzurdoki	Berroeta
Indakoa	Indakoa	Berroeta
Mendicoa	Mendikoa	Berroeta
Aintze/Eneko berri	Aintze/Enekoberri	Berroeta
Maxenekoborda	Maxeneko Borda	Berroeta
Uztartekoborda	Uztarteko Borda	Berroeta
Cuadra aizkoa	Cuadra Aizkoa	Berroeta
Behitiko etxea	Behitiko Etxea	Berroeta
Borda Osamisketa	Osamisketa Borda	Berroeta
Garaiko Etxeko Borda	Garaiko Etxeko Borda	Almandoz
Iturrikoa	Iturrikoa	Almandoz
Eguzki alde	Eguzkialde	Almandoz
Indakoa	Indakoa	Almandoz
Toki Eder	Tokieder	Almandoz
Txokoberri	Txokoberri	Almandoz
Abreiko	Abreiko	Almandoz
Berekoetxea	Berekoetxea	Ziga-Zigaurre
Etxenikea	Etxenikea	Ziga-Zigaurre
Indakoa	Indakoa	Ziga-Zigaurre
Aldekoa	Aldekoa	Ziga-Zigaurre
Aldekotxeberria	Aldekotxeberria	Ziga-Zigaurre
Iturrikoa	Iturrikoa	Ziga-Zigaurre
Garbilekua	Garbilekua	Ziga-Zigaurre
Garaikoa	Garaikoa	Ziga-Zigaurre
Eguzkialdea	Eguzkialdea	Ziga-Zigaurre
Garaikoa	Garaikoa	Ziga-Zigaurre
Indakoa	Indakoa	Ziga-Zigaurre
Egozkue	Egozkue	Ziga-Zigaurre

Tabla 10: Modificaciones propuestas para utilizar el K

Los nombres de diseminados con terminaciones en – *a*

- En el fichero KML existen numerosos nombres de diseminados que terminan o no con la letra -a (ver Tablas 11–14). Es el caso de -alde(a), -ondo(a), -buru(a), -leku(a) y -gain(a).

REGISTRO	PROPUESTA	Barrio o Pueblo
Eguzki alde	Eguzkialde	Almandoz
Garazialde	Garazialde	Ziga-Zigaurre
Elizaldea	Elizaldea	Berroeta
Elizaldea	Elizaldea	Berroeta
Iturraldea	Iturraldea	Almandoz
Elizaldea	Elizaldea	Ziga-Zigaurre
Mendialdea	Mendialdea	Ziga-Zigaurre
Eguzkialdea	Eguzkialdea	Ziga-Zigaurre
Larraldea	Larraldea	Ziga-Zigaurre
Elizaldea	Elizaldea	Ziga-Zigaurre
Munoaldea	Munoaldea	Ziga-Zigaurre

Tabla 11: Nomenclaturas de los diseminados que contienen -alde(a)

REGISTRO	PROPUESTA	Barrio o Pueblo
Baratzondoa	Baratzondoa	Ziga-Zigaurre
Iturriondoa	Iturriondoa	Ziga-Zigaurre
Iturriondoa A	Iturriondoa A	Ziga-Zigaurre
Iturriondoa B	Iturriondoa B	Ziga-Zigaurre
Berrondoa	Berrondoa	Ziga-Zigaurre
Haritzondoa	Haritzondoa	Ziga-Zigaurre

Tabla 12: Nomenclaturas de los diseminados que contienen -ondo(a)

REGISTRO	PROPUESTA	Barrio o Pueblo
Larraburu	Larraburu	Berroeta
Goldaburu	Goldaburu	Almandoz
Ideburu	Ideburu	Ziga-Zigaurre
Soraburua	Soraburua	Ziga-Zigaurre

Tabla 13: Nomenclaturas de los diseminados que contienen -buru(a)

REGISTRO	PROPUESTA	Barrio o Pueblo
Garbilekua	Garbilekua	Ziga-Zigaurre
Ariztizelai gaina	Ariztizelaigaina	Almandoz

Tabla 14: Nomenclaturas de los diseminados que contienen -leku(a) y gain(a)

- Existen otros nombres compuestos con otras terminaciones, en este caso adjetivos, que terminan o no con la letra -a (ver Tablas 15 – 20). Es el caso de -berri(a), -zuri(a) o -txuri(a), -eder(ra), -txar(ra), -garai(a), -txiki(a), -argi(a), etc.

REGISTRO	PROPUESTA	Barrio o Pueblo
Aintze / Eneko berri	Aintze / Enekoberri	Berroeta
Borda berri	Bordaberri	Berroeta
Benta Berri	Bentaberri	Almandoz
Bordaberri	Bordaberri	Almandoz
Txokoberri	Txokoberri	Almandoz
Etxeberria	Etxeberria	Almandoz
Aldekotxeberria	Aldekotxeberria	Ziga-Zigaurre
Etxeberria	Etxeberria	Ziga-Zigaurre
Ideberria	Ideberria	Ziga-Zigaurre
Etxeberria	Etxeberria	Ziga-Zigaurre
Ideberria	Ideberria	Ziga-Zigaurre

Tabla 15: Nomenclaturas de los diseminados que contienen -berri(a)

REGISTRO	PROPUESTA	Barrio o Pueblo
Bordachuri	Bordatxuri	Aniz
Etxezuria	Etxezuria	Ziga-Zigaurre

Tabla 16: Nomenclaturas de los diseminados que contienen -txuri(a) y -zuri(a)

REGISTRO	PROPUESTA	Barrio o Pueblo
Toki Eder	Tokieder	Almandoz
Bixtaeder	Bixtaeder	Ziga-Zigaurre

Tabla 17: Nomenclaturas de los diseminados que contienen -eder(ra)

REGISTRO	PROPUESTA	Barrio o Pueblo
San Blas Txar	San Blas Txar	Almandoz
Etxe-txar	Etxetxar	Almandoz

Tabla 18: Nomenclaturas de los diseminados que contienen -txar(ra)

REGISTRO	PROPUESTA	Barrio o Pueblo
Ideberrigaraia	Ideberrigaraia	Ziga-Zigaurre
Iturrigaraia	Iturrigaraia	Ziga-Zigaurre

Tabla 19: Nomenclaturas de los diseminados que contienen -garai(a)

REGISTRO	PROPUESTA	Barrio o Pueblo
Legarre-Txiki	Legarretxiki	Ziga-Zigaurre
Goiz-Argi	Goizargi	Ziga-Zigaurre

Tabla 20: Nomenclaturas de los diseminados que contienen -txiki(a) y -argi(a)

- En todos estos casos, no serán un obstáculo para realizar correctamente el proceso de búsqueda. Por lo tanto, no se prevén cambios en los registro de la base de datos.

Las nomenclaturas con terminaciones en – *(e)nea*

- En el fichero KML existen numerosas nomenclaturas de diseminados que contienen la terminación -(e)nea. La terminación de estos nombres en algunos casos se escribe -enea y en otras -nea. En estos casos, en el proceso de búsqueda se podría producir alguna confusión.
- Al analizar estos casos, se observa que la utilización de -enea o -nea depende de letra que le precede:

  Arotz + enea = Arotzenea Gana + enea = Gananea

  Dendari + enea = Dendarienea Elso + enea = Elsonea

- Más concretamente, al añadir la terminación -enea a una palabra que termina en los vocales e, a y o la palabra resultante perderá la -e-. Por lo tanto, la variación seguirá el siguiente esquema:

  $\text{-e} + \text{-enea} \rightarrow \text{-enea}$ $\text{-i} + \text{-enea} \rightarrow \text{-ienea}$

  $(\text{-a}, \text{-o}) + \text{-enea} \rightarrow \text{-nea}$ $\text{-u} + \text{-enea} \rightarrow ?$

- Al analizar estos casos se ha identificado un nombre que contiene una variación por la influencia del dialecto *Baztanés*. Es el caso de “Sasternea” donde el nombre originario es “Sasterrenea”. Esta variación seguirá el siguiente esquema:

  $\text{-rre} + \text{-enea} \rightarrow \text{-rnea}$

- En todos estos casos, para evitar confusiones se propone añadir como nombre de destino las diferentes formas de escribir de estas nomenclaturas (ver Tabla 21).

REGISTRO	PROPUESTA	Barrio o Pueblo
Arotzenea	Arotzenea	Berroeta
Puyenea	Pujenea	Berroeta
Aintzinenea	Aintzinenea	Berroeta
Santxenea	Santxenea	Berroeta
Arotxenea	Arotxenea (? Arotzenea)	Berroeta
Puyenea	Pujenea	Berroeta
Perotxenea	Perotxenea	Almandoz
Amosenea	Amosenea	Ziga-Zigaurre
Arotzenea	Arotzenea	Ziga-Zigaurre
Gorrienea	Gorrienea	Ziga-Zigaurre
Ziganda-Enea	Ziganda-Enea / Zigandanea	Ziga-Zigaurre
Petrinenea	Petrinenea	Ziga-Zigaurre
Dendarienea	Dendarienea	Ziga-Zigaurre
Kalainenea	Kalainenea	Ziga-Zigaurre
Sabatenea	Sabatenea	Ziga-Zigaurre
Gananea	Gananea / Gana-Enea	Aniz
Elsonea	Elsonea / Elso-Enea	Almandoz
Gartxitonea	Gartxitonea / Gartxito-Enea	Almandoz
Sasternea	Sasternea / Sasterrenea	Ziga-Zigaurre
Gananea	Gananea / Gana-Enea	Ziga-Zigaurre
Segonea	Segonea / Sego-Enea	Ziga-Zigaurre

Tabla 21: Nomenclaturas de los diseminados que contienen -(e)nea

- Por último, hay que mencionar un caso especial que se han identificado: dos diseminados de una misma zona contienen nombres muy parecidas, por lo tanto son fáciles de confundir. Es el caso, del diseminado “Arotxenea” (de Berroeta) y el diseminado “Arotzenea” (de Berroeta).

1.4.2. Duplicidad de tramos de rutas en el fichero

Al analizar los tramos de las rutas se puede observar los siguientes problemas:

- Los tramos que componen las rutas están delimitados por cruces de vías. Esta delimitación refleja un cambio en el tipo de vía o características de la calzada, pero no en todo los casos. En el fichero KML se aprecia dos tipos generales de tramos: tramos de carretera y tramos rurales. Los tramos de carretera contienen las mismas características de la calzada, en cambio los tramos rurales no. Estos tramos a su vez están compuestas por varias subtramos rurales. Pero al analizar sus características se aprecia que contienen una mezcla de composición de suelo y características de calzada. En la Imagen 64 se puede observar tres tramos rurales con tres colores diferentes.

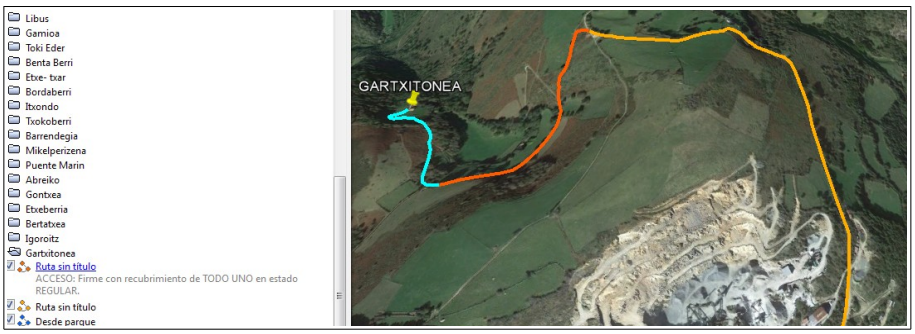

Imagen 64: Ruta compuesta por cuatro subtramos

- En las rutas analizadas, los tramos no están exactamente delimitados por los cruces físicos de las carreteras, y cada tramo empieza y acaba siguiendo un diferente criterio. En la Imagen 65 se pueden observar estas variaciones: los tramos de carreteras en azul no finalizan en los cruces de carretera.

Imagen 65: Diferentes criterios de delimitación de tramos

- Cada una de las rutas contiene sus propios tramos, y son diferentes a los tramos de las otras rutas. Por tanto, al solaparse todos estos tramos existe información redundante. Esta duplicidad de datos es habitual en las zonas donde se agrupan varios diseminados. En la Imagen 66 y en la Imagen 67 se aprecia cómo estos tramos no siguen el mismo eje de la vía, y crean diferentes rutas alrededor de este eje.

Imagen 66: Duplicidad y solapamiento de tramos

Imagen 67: Variación de la trayectoria de los tramos ante el eje de la vía

Imagen 68: Tramos de la ruta que no se conectan entre sí

- Existen casos de rutas donde los tramos no llegan a conectarse entre sí (ver Imagen 68), y existen otros casos donde el último tramo no llegan exactamente al punto de destino (ver Imagen 69).

Imagen 69: Tramos que no llegan exactamente al diseminado

- Para acceder a un diseminado cabe la posibilidad de que existan varias formas de acceso. Por lo tanto, un diseminado puede contener varias rutas de acceso: una ruta directa y otra ruta alternativa. En el fichero KML se ha identificado un caso donde existen varias rutas de acceso de un mismo diseminado almacenadas en una misma carpeta. Es el caso del diseminado "*Elizalde*". En la Imagen 70 se puede apreciar que no se ha completado correctamente estas rutas de acceso.

Imagen 70: Tramos que no conforman exactamente una ruta

- Como conclusión se propone varios cambios en el registro de datos para solventar estos problemas. Estas conclusiones se ha enumerado en el apartado 4.1.3.4. de la memoria.

A2.2. Diseño técnico de la base de datos

El diseño técnico de la base de datos que se quiere realizar se basará en la definición de la estructura de datos de nuestro sistema de información geográfica. Para ello se crearán diferentes fases de diseño donde se crearán el modelo conceptual, el modelo lógico o relacional y el modelo físico.

1. Modelo de datos conceptual

Para definir las variables utilizadas en la base de datos y el tipo de valor que van a contener se procederá a realizar un diagrama Entidad / Relación (E/R), donde se reflejará la información que contiene cada elemento (cada entidad y cada relación creada entre entidades).

1.1. Relaciones de las Entidades y Relaciones

1.1.1. Relación Contiene: Zona contiene a Zona

Las zonas pueden contener (o no) una o varias zonas. En cambio, las subzonas (zonas que son parte de una zona) no pueden estar dentro de varias zonas. Es decir, cada subzona sólo estará dentro de una zona.

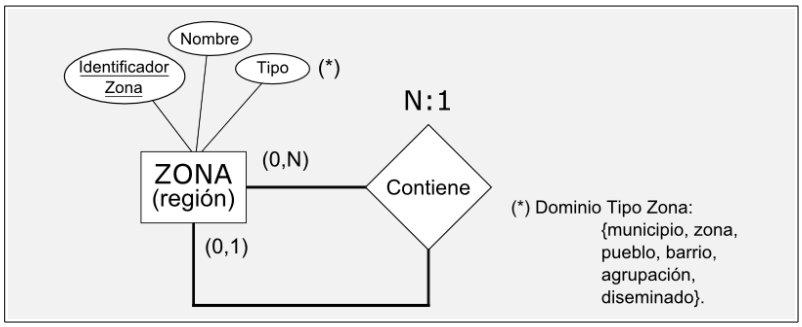

Imagen 71: La relación involutiva Contiene y la entidad Zona

1.1.2. Relación Pertenece: Destino pertenece a Zona

Los destinos son localizaciones de diseminados, y pertenecen al menos a una zona. Las zonas son regiones (que representan barrios, poblaciones, municipios o agrupaciones de diseminados), y un destino sólo podrá pertenecer a una zona. En una zona puede no contener ningún destino (diseminado), al ser una zona despoblada.

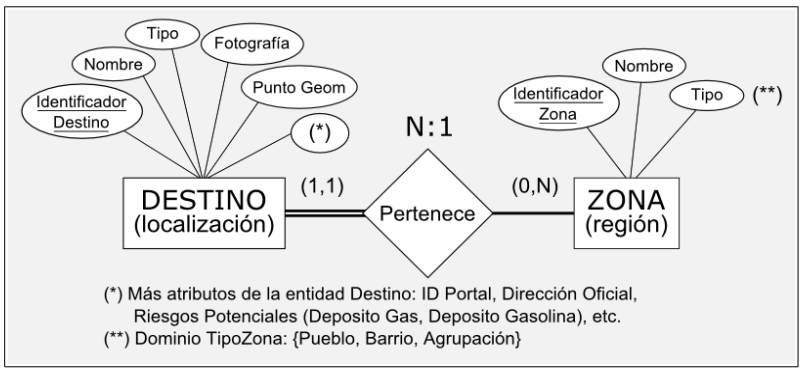

Imagen 72: La relación Pertenece y las entidades Destino y Zona

1.1.3. Relación Reside: Habitante reside a Destino

Los habitantes son personas que viven en los diseminados (edificio), y un habitante sólo podrá residir en una vivienda. Esto se debe a que los datos referidos a habitantes provienen del empadronamiento municipal. Como la residencia de los/las habitantes varía con el tiempo, puede que existan habitantes (en la base de datos) que actualmente no estén vinculados a ningún destino (de la *Comarca de Baztan*).

Los diseminados podrán estar habitados por varias personas, y también podrán existir diseminados (en la base de datos) que actualmente no tengan ningún habitante residiendo en ella. También existen diseminados que no contendrán ningún habitante (al no tener ningún habitante empadronamiento en ella), es el caso de los edificios de tipo ermita, cementerio, iglesia, escuela y puente.

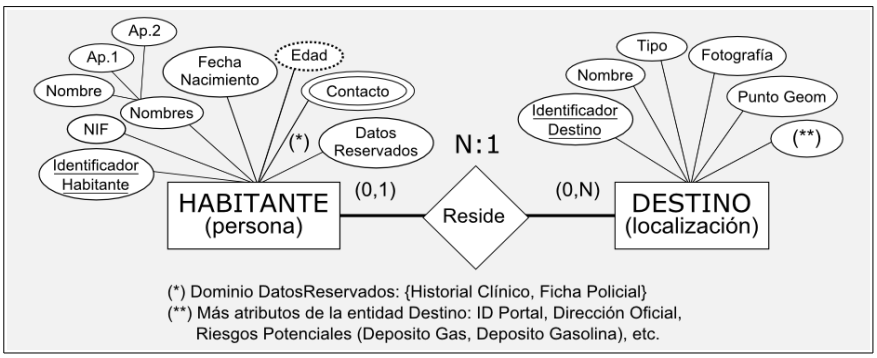

Imagen 73: La relación Reside y las entidades Habitante y Destino

1.1.4. Relación Es Vecino: Destino es vecino de Destino

Los destinos pueden tener vecinos/as (o no) uno o varios destinos. En cambio, estos destinos vecinos (destinos que tienen un vecino/a) pueden tener varios destinos vecinos.

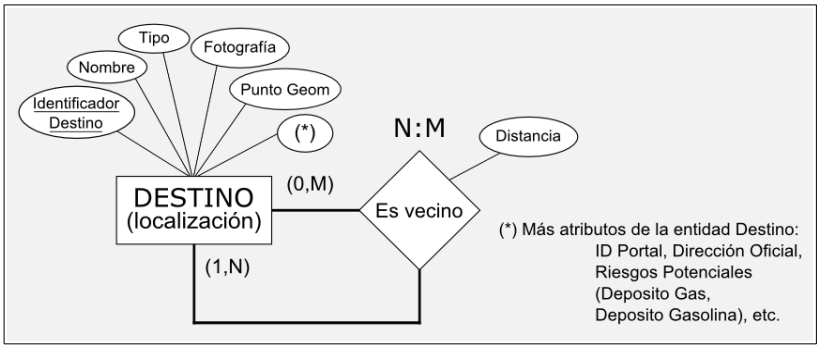

Imagen 74: La relación involutiva Es Vecino y la entidad Destino

1.1.5. Relación Llega: Ruta llega a Destino

Las rutas representan el recorrido donde se accede a los destinos, y cada ruta llegará a un solo destino. Los destinos son localizaciones de diseminados, y habrá destinos que todavía no contengan asociada ninguna ruta de acceso. Pero en la mayoría de los casos a los destinos llegarán varias rutas, al disponer cada servicio de emergencia una ruta personalizada para cada destino.

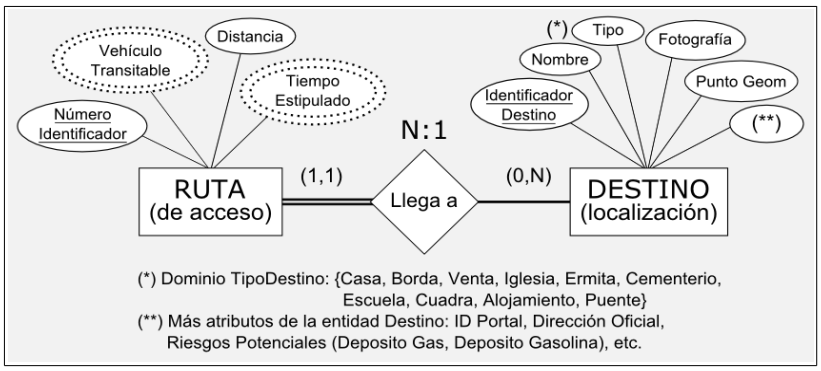

Imagen 75: La relación Llega A y las entidades Ruta y Destino

1.1.6. Relación Parte: Ruta parte de Salida

Las rutas parten de una sola salida para acceder a un solo destino. Estas salidas son localizaciones de equipamientos orientados a los servicios de emergencia, con la finalidad de que estos servicios obtengan rutas que partan desde sus propios equipamientos. Por lo tanto, desde cada salida podrán partir varias rutas de acceso. Pero habrá casos donde no partirá ninguna ruta asociada. Será el caso de salidas que están ubicados fuera de la *Comarca de Baztan* (por ejemplo, el *Parque Central de Bomberos* de Cordovilla).

En estos casos, si algún vehículo de estas salidas tiene que desplazarse a Baztan para asistir una emergencia será interesante la utilización de los cruces de desvío (definidos como puntos de acceso a tramos rurales), y poder así utilizar las ruta de acceso a ese incidente.

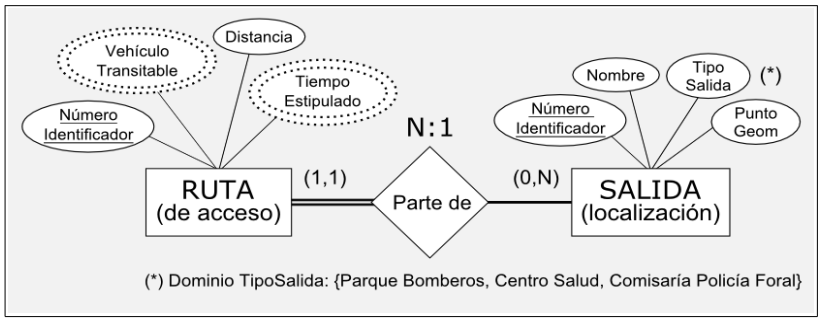

Imagen 76: La relación Parte De y las entidades Ruta y Salida

1.1.7. Relación Utiliza: Servicio utiliza Salida

Los servicios serán equipos de emergencia que operan en la *Comunidad Foral de Navarra*, y podrán tener o no salidas (equipamientos o infraestructuras) en la *Comarca de Baztan*. Por la complejidad operativa de los servicios de emergencia un servicio podrá tener varias salidas (equipamientos) en esta región de trabajo. Cada salida estará utilizada por al menos un servicio de emergencia, y habrá salidas donde compartan instalaciones varios servicios.

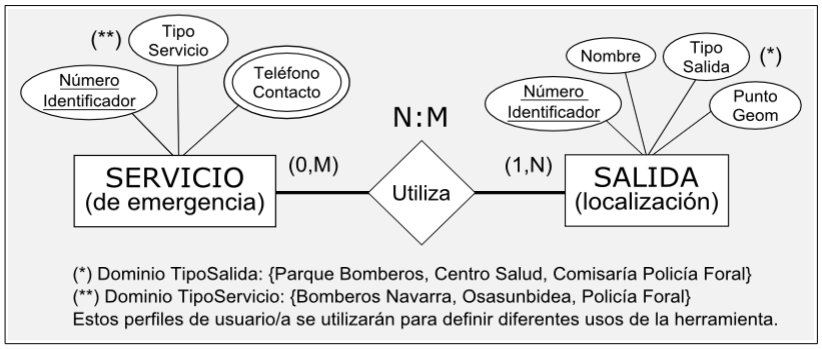

Imagen 77: La relación Utiliza y las entidades Servicio y Salida

1.1.8. Relación Pertenece: Tramo pertenece a Ruta

Los tramos son partes del itinerario a recorrer y conforman las rutas de acceso. Por lo tanto, un tramo puede pertenecer a una o varias ruta de acceso. Pero podrán existir tramos (almacenados en la base de datos) que no forman parte de ninguna ruta de acceso. Por ejemplo al actualizar los recorridos de las rutas y existir cambios en el acceso de algún diseminado. Cada ruta tendrá por lo menos un tramo, y también podrá tener varios tramos.

Al crear la posibilidad de que un tramo pueda pertenecer a varias rutas se evita tramos repetidos en la base de datos. De esta manera se reducen los datos redundantes que existían en el documento KML.

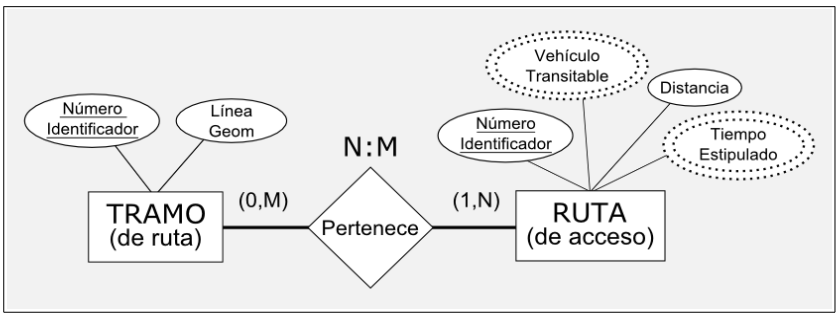

Imagen 78: La relación Pertenece y las entidades Tramo y Ruta

1.1.9. Relación Contiene: Ruta contiene Cruce

Los tramos estarán delimitados por cruces de desvío. Pero en este caso, el cruce se referirá solamente al desvío donde se deja de transitar por el tramo de carretera y se accede al tramo rural. Por lo tanto, las rutas pueden contener sólo cruce. Pero habrá casos donde no contenga ningún cruce. Éstos serán los casos donde no existe ningún tramo rural en la ruta de acceso.

En general, los cruces pertenecerán a una ruta. Pero también podrá pertenecer a varias rutas. Esta posibilidad evita la existencia de cruces repetidas en nuestra base de datos.

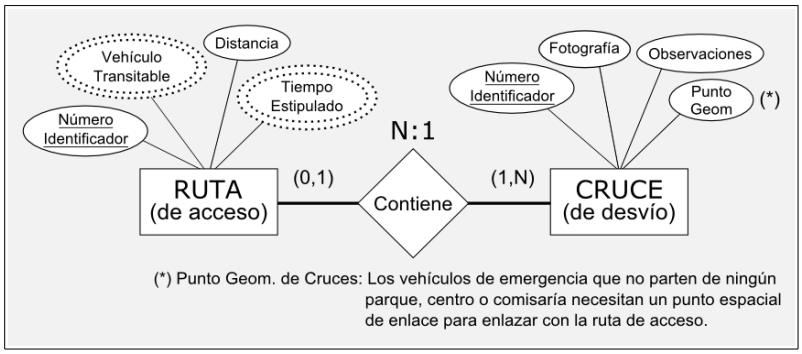

Imagen 79: La relación Delimita y las entidades Ruta y Cruce

1.1.10. Relación Contiene: Ruta contiene Punto Información

Los puntos de información son puntos de advertencia existentes en las rutas. Por lo tanto, las rutas pueden contener (o no) una o varios puntos de información.

Los puntos de información también podrán pertenecer a varias rutas. Esta posibilidad surge al querer evitar puntos de información repetidas en la base de datos.

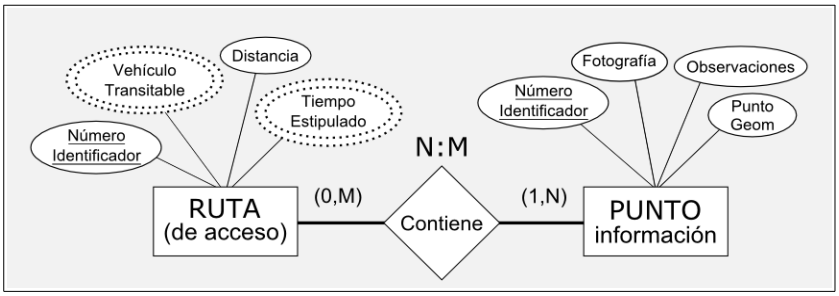

Imagen 80: La relación Contiene y las entidades Ruta y Punto Información

1.1.11. Relación subclase: Tramo es Tramo Rural y Tramo Carretera

Los tramos rurales como los tramos carretera son subclases de la entidad Tramo.

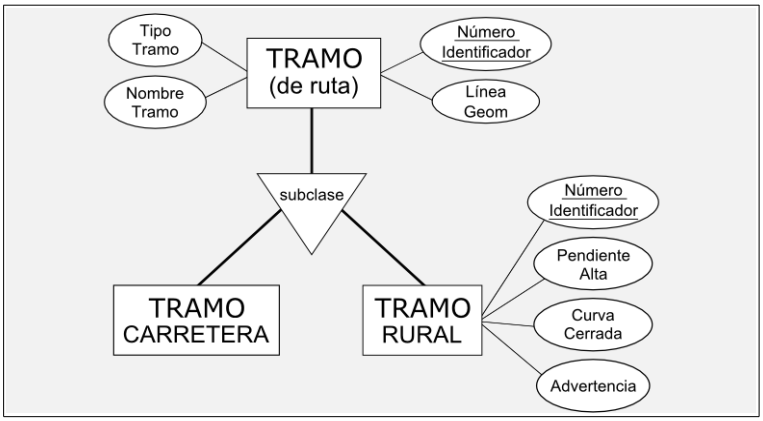

Imagen 81: Creación de dos subclase de la entidad Tramo

Al disponer solamente de información sobre tramos rurales, sólo se ha creado la entidad Tramo Rural.

1.1.12. Relación Contiene: Tramo rural contiene Suelo

Los tramos rurales son los tramos más difíciles de transitar de la ruta de acceso. Cada tramo rural contendrá una o varias composiciones del suelo. La posibilidad de que un tramo rural pueda contener varias composiciones de suelo es porque los tramos rurales no están totalmente delimitado por un cambio de características de la calzada.

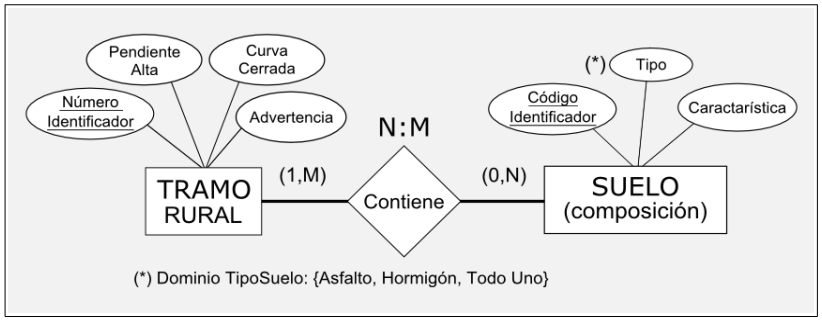

Imagen 82: La relación Contiene y las entidades Tramo y Suelo

Una composición del suelo pueden estar contenidas en varios tramos rurales, y podrá existir alguna característica que no contenga ningún tramo rural.

1.1.13. Relación Contiene: Tramo rural contiene Estado

Cada tramo rural contendrá una característica referida al estado de transito de la vía, y podrá contener la fecha de esta categorización. Una estado de la vía pueden estar contenidas en varios tramos, y podrá existir alguna característica que no contenga ningún tramo.

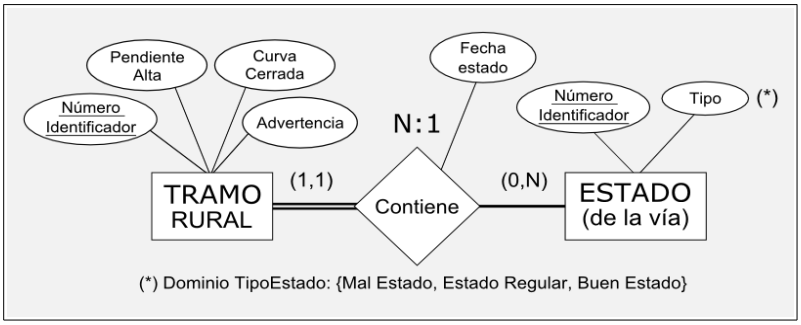

Imagen 83: La relación Contiene y las entidades Tramo y Estado

1.1.14. Relación Supera: Vehículo supera Tramo

Los vehículos de emergencia son los vehículos disponibles para realizar los servicios de emergencia. Cada vehículo podrá superar al menos un tramo, y algunos podrán superar varios de ellos. Podrán existir tramos que no sean superados por ningún vehículo de emergencia. También habrá tramos que sean superadas por varios vehículos.

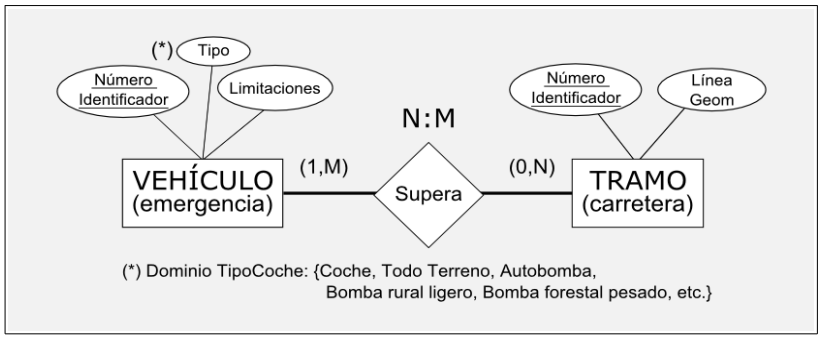

Imagen 84: La relación Supera y las entidades Vehículo y Dificultad

1.1.15. Relación Dispone: Servicio dispone Vehículo

Los servicios de emergencia dispondrán al menos de un vehículo, y podrán disponer de varios vehículos. Cada vehículo estará asociado a un solo servicio de emergencia.

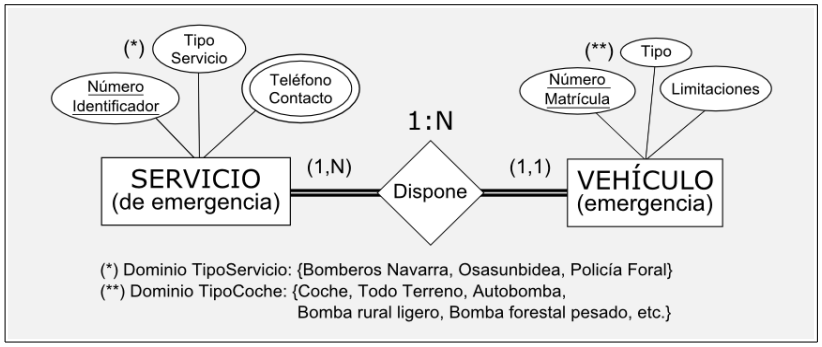

Imagen 85: La relación Dispone y las entidades Servicio y Vehículo

1.2. Atributos de las Entidades y Relaciones

Para terminar de realizar el modelado de nuestra base de datos (y acabar de especificar cómo estructurar la información), se definirán los atributos que podrán utilizar cada entidad descrita anteriormente. También se especificará el dominio de cada atributo, y se mencionará cuál de estos atributos (o qué composición de atributos) podrán ser clave primaria de la entidad. De esta forma se terminará de definir qué información debe contener nuestro sistema y cómo se van a relacionar.

1.2.1. Atributos de la entidad Zona

Como se observa en las imágenes 71 y 72, la entidad *Zona* contiene los siguientes atributos:

- *idzona*. Es un atributo univalorado que contiene el número identificador de esta entidad. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *nombre*. Es un atributo univalorado que contiene el nombre de la zona. Su dominio también será de tipo alfanumérico y con extensión de 100 caracteres.
- *tipozona*. Es un atributo univalorado que contiene el tipo de zona. Su dominio tendrá los siguientes valores definidos: agrupación, barrio, pueblo, municipio.

1.2.2. Atributos de la entidad Destino

Como se observa en las imágenes 72, 73, 74 y 75, la entidad *Destino* contiene los siguientes atributos:

- *iddestino.* Es un atributo univalorado que contiene el número identificador de esta entidad. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *nombre.* Es un atributo univalorado que contiene el nombre del destino. Su dominio será de tipo alfanumérico y con extensión de 100 caracteres.
- *tipodestino*. Es un atributo univalorado que contiene el tipo de destino. Su dominio tendrá los siguientes valores definidos: Casa, Borda, Venta, Iglesia, Ermita, Cementerio, Escuela, Cuadra, Alojamiento, Puente.
- *fotografia.* Es un atributo univalorado donde se almacena la dirección URL donde está almacenado la imagen de la fachada del destino (diseminado). Su dominio también será de tipo texto y con una longitud de 1000 caracteres.
- *geom*. Este atributo contiene la información espacial de la entidad. Su dominio será de tipo Geometry (con un subtipo Point). Toda la información espacial de la base de datos estará en el sistema de referencia ETRS-89 y proyectada en UTM en el huso 30 N.
- *idportal*. Este atributo será el número identificador de una referencia externa del Gestor de Direcciones Postales de Navarra, HelpBidea. Su dominio será de tipo numérico entero positivo.
- *direccionoficial*. Atributo compuesto relacionado con la dirección del diseminado. Contendrá atributos compuestos como los siguientes ejemplos:
  - *direccionmunicipio*. Contiene atributos como provincia, municipio, cp, codmun, idpoblacion.
  - *direccionvia*. Contiene atributos como nombrecalle, idenvia.
  - *direccionportal*. Contiene atributos como numeroportal, letraportal.
  - *direccionparcela*. Contiene atributos como nombrepoligono, nombreparcela.
- *riesgospotenciales*. Atributo compuesto relacionado con la información sobre riesgos potenciales de viviendas elaborado por los Bomberos de Navarra. Contendrá atributos como depositogas y depositogasolina. Cada uno de ellos tendrá un dominio de tipo booleano.

Para organizar mejor el conjunto de atributos de la entidad *Destino* se ordenarán los atributos respecto a la procedencia de la información. De esta manera será más claro su contenido (por ejemplo en las actualizaciones).

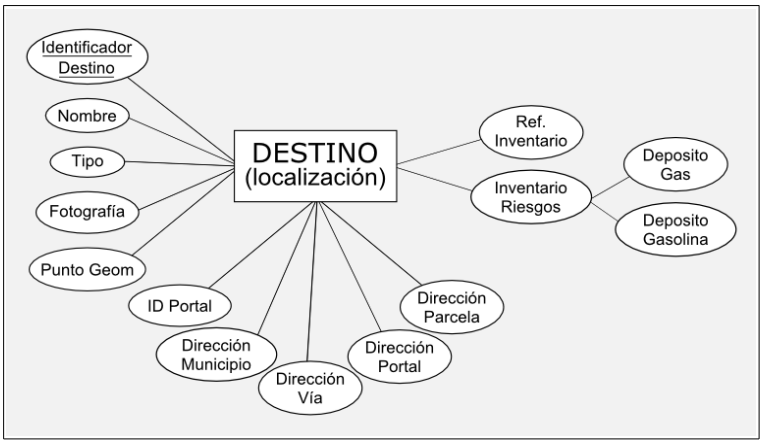

Imagen 86: Atributos de la entidad Destino distribuidos según su origen

Como se puede apreciar en la Imagen 86, los atributos proceden en general de tres orígenes distintos:

- Información General. Contendrá la información general de la vivienda (tipología del edificio, fotografía de la fachada, ubicación en coordenadas ETRS-89, etc.).
- Información Direcciones. Contendrá información referida a direcciones (definida en *Helpbidea*).
- Información Bomberos. Contendrá la información referida al inventario de riesgos potenciales de los edificios realizada por los Bomberos de Navarra (referencia del identificador del inventario, existencia de depósitos de gas o gasolina, etc.).

1.2.3. Atributos de la entidad Habitante

Como se observa en la Imagen 73, la entidad *Habitante* contiene los siguientes atributos:

- *idhabitante*. Es un atributo univalorado que contiene el número identificador de esta entidad. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *nombrehabitante*. Es un atributo compuesto que contiene el nombre compuesto del habitante. Contendrá atributos como nombre, apellido1, apellido2. Cada uno de ellos tendrá un dominio de tipo alfanumérico y una extensión de 150 caracteres.
- *nif*. Es un atributo univalorado que contiene el código NIF de una persona. Su dominio será de tipo texto con una extensión de 9 caracteres.
- *fechanacimiento*. Es un atributo univalorado que contiene la fecha de nacimiento del habitante. Su dominio será de tipo fecha (año, mes, día).
- *edad*. Es un atributo univalorado que contiene la edad del habitante. Este atributo es un atributo derivado, y contiene el valor calculado a partir del atributo fechanacimiento. Su dominio será de tipo intervalo (numérico entero positivo).
- *contacto*. Es un atributo multivalorado que contiene el número de contacto del habitante. Su dominio será de tipo alfanumérico con una extensión de 15 caracteres.

- *datosreservados*. Es un atributo compuesto que contiene información reservada de los habitantes. Contendrá atributos como historialclinico (información sanitaria obtenida de Osasunbidea), fichapolicial (información de antecedentes obtenida de la Policía Foral). Cada uno de ellos tendrá un dominio de tipo texto con una longitud de 1500 caracteres.

1.2.4. Atributos de la relación Es Vecino

Como se observa en la Imagen 74, la relación *Es_Vecino* contiene el siguiente atributo:

- *distanciavecindad*. Es un atributo univalorado que contiene la distancia existente (en metros) entre dos destinos. Su dominio será de tipo numérico real positivo. Se considerará destinos vecinos a los destinos que se sitúen a una distancia menor a medio kilómetro.

1.2.5. Atributos de la entidad Ruta

Como se observa en las imágenes 75, 76, 78, 79 y 80, la entidad *Ruta* contiene los siguientes atributos:

- *idruta*. Es un atributo univalorado que contiene el número identificador de la ruta. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *vehiculotransitable.* Es un atributo multivalorado que contiene los nombres de los tipos de vehículos que pueden transitar por esta ruta. También este atributo es un atributo derivado, y su valor se calculará a partir de las relaciones entre entidades ruta, tramo, y vehiculo. Su dominio será de tipo texto con una extensión de 50 caracteres.
- *longitud*. Es un atributo univalorado que contiene la distancia de la ruta. Este atributo no será un atributo derivado, y no se calculará a partir del atributo geom de la entidad Tramo. Este valor también se utilizará como parámetro para el cálculo del tiempo de transito del tramo. Su dominio es de tipo numérico real positivo.
- *tiempo*. Es un atributo multivalorado que contiene el tiempo estipulado para cada tipo de vehículo que puede transitar por esta ruta. Este atributo es un atributo derivado, y se utilizarán los parámetros de distancia y características de suelo y dificultad de transito para calcular su valor. Su dominio es de tipo tiempo (hora, minuto, segundo).

  La existencia del atributo tiempo podrá ser un dato interesante para los servicios de emergencia. Se podría obtener un tiempo aproximado de transito de cada ruta de acceso, y también se podría obtener un intervalo de tiempo para cada vehículo de emergencia. Por lo tanto, habría información suficiente para elegir el vehículo y el tiempo de transporte conveniente para cada servicio de emergencia.

1.2.6. Atributos de la entidad Salida

Como se observa en las imágenes 76 y 77, la entidad *Salida* contiene los siguientes atributos:

- *idsalida*. Es un atributo univalorado que contiene el número identificador de la salida. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.

- *nombre*. Es un atributo univalorado que contiene el nombre de la salida. Por ejemplo Parque de Bomberos Oronoz-Mugaire, Parque de Bomberos Cordovilla, Centro de Salud Elizondo, Hospital de Pamplona, etc. Su dominio será de tipo texto con una extensión de 100 caracteres.
- *tiposalida*. Es un atributo univalorado que contiene el tipo de salida. Su dominio tendrá los siguientes valores definidos: parque bomberos, centro salud, hospital, comisaria policía.
- *geom*. Este atributo contiene la información espacial de la entidad. Su dominio será de tipo Geometry (con un subtipo Point).

1.2.7. Atributos de la entidad de Servicio

Como se observa en las imágenes 77 y 84, la entidad *Servicio* contiene los siguientes atributos:

- *idservicio*. Es un atributo univalorado que contiene el número identificador del servicio. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *tiposervicio*. Es un atributo univalorado que contiene el tipo de servicio de emergencia. Su dominio tendrá los siguientes valores definidos: Bomberos Navarra, Osasunbidea, Policía Foral.
- *contacto*. Es un atributo multivalorado que contiene el número de contacto del servicio de emergencia. Su dominio será de tipo alfanumérico con una extensión de 15 caracteres.

1.2.8. Atributos de la entidad de Tramo

Como se observa en las imágenes 78, 81 y 84, la entidad *Tramo* contiene los siguientes atributos:

- *idtramo*. Es un atributo univalorado que contiene el número identificador del tramo. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *geom*. Este atributo contiene la información espacial de la entidad. Su dominio será de tipo Geometry (con un subtipo LineString).

1.2.9. Atributos de la entidad de Tramo Rural

Como se observa en las imágenes 81, 82 y 83, la entidad *Tramo Rural* contiene los siguientes atributos:

- *idtramo*. Es un atributo univalorado que contiene el número identificador del tramo. Este atributo será la clave primaria y la clave foránea. Su dominio será de tipo numérico entero positivo.
- *pendientealta*. Es un atributo univalorado que contiene información sobre la existencia de pendientes altas en el recorrido de la ruta. Su dominio será de tipo booleano.

- *curvacerrada*. Es un atributo univalorado que contiene información sobre la existencia de curvas cerradas en el recorrido de la ruta. Su dominio será de tipo booleano.
- *advertencia*. Es un atributo univalorado que contiene advertencias de las dificultades de transito del tramo rural. Su dominio será de tipo texto con una longitud de 1000 caracteres.

1.2.10. Atributos de la entidad Cruce

Como se observa en la Imagen 79, la entidad *Cruce* contiene los siguientes atributos:

- *idcruce*. Es un atributo univalorado que contiene el número identificador de esta entidad. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *fotografia.* Es un atributo univalorado donde se almacena la dirección URL donde está almacenado la imagen del cruce. Su dominio también será de tipo texto y con una longitud de 1000 caracteres.
- *observacion*. Es un atributo univalorado que contiene las observaciones del cruce de desvío. Su dominio será de tipo alfanumérico con una extensión de 1000 caracteres.
- *geom*. Este atributo contiene la información espacial de la entidad. Su dominio será de tipo Geometry (con un subtipo Point).

1.2.11. Atributos de la entidad Punto Información

Como se observa en la Imagen 80, la entidad *Punto Información* contiene los siguientes atributos:

- *idpuntoinfo*. Es un atributo univalorado que contiene el número identificador de esta entidad. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *descripcion*. Es un atributo univalorado que contiene la descripción del punto de información. Su dominio será de tipo alfanumérico con una extensión de 1000 caracteres.
- *geom*. Este atributo contiene la información espacial de la entidad. Su dominio será de tipo Geometry (con un subtipo Point).

1.2.12. Atributos de la entidad Suelo

Como se observa en la Imagen 82, la entidad *Suelo* contiene los siguientes atributos:

- *idsuelo.* Es un atributo univalorado que contiene el número identificador del suelo. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *tiposuelo.* Es un atributo univalorado que contiene el tipo de suelo. Su dominio tendrá los siguientes valores definidos: asfalto, hormigón, todo uno, hierba, tierra.
- *caracteristica*. Es un atributo univalorado que contiene las características del suelo. Su dominio será de tipo alfanumérico con una extensión de 1000 caracteres.

1.2.13. Atributos de la entidad Estado

Como se observa en la Imagen 83, la entidad *Estado* contiene los siguientes atributos:

- *idestado*. Es un atributo univalorado que contiene el número identificador de estado. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *tipoestado*. Es un atributo univalorado que contiene el tipo de estado. Su dominio tendrá los siguientes valores definidos: mal estado, estado regular, buen estado.
- *caracteristica.* Es un atributo univalorado que contiene las características del estado. Su dominio será de tipo alfanumérico con una extensión de 1000 caracteres.

1.2.14. Atributos de la entidad Vehículo

Como se observa en las imágenes 84 y 85, la entidad *Vehiculo* contiene los siguientes atributos:

- *idvehiculo*. Es un atributo univalorado que contiene el número de identificador del vehículo. Este atributo será la clave primaria. Su dominio será de tipo numérico entero positivo.
- *tipovehiculo*. Es un atributo univalorado que contiene el tipo de vehículo de emergencia. Su dominio tendrá los siguientes valores definidos: coche, todo-terreno, autobomba, ambulancia, bomba rural ligero, bomba forestal pesado.
- *limitacion.* Es un atributo univalorado que contiene las limitaciones del vehículo de emergencia. Su dominio será de tipo alfanumérico con una extensión de 1000 caracteres.

2. Modelo relacional, estructura de datos

Este apartado recoge el modelo relacional obtenido a partir del diseño del modelo conceptual. Por lo tanto, para poder diseñar la estructura física de la base de datos de nuestra herramienta, se trasladarán las entidades y atributos (definidos en el modelo conceptual) a tablas y columnas (del modelo lógico del sistema).

En este sentido, también se procederá a establecer restricciones que deberán cumplir los valores del conjunto de datos almacenados en dicha base de datos (para poder así mantener la integridad de los datos).

2.1. Tablas y columnas del modelo relacional

A continuación se explicará cómo se han definido las tablas y columnas en el modelo relacional. En dichas tablas, las columnas subrayadas representarán las claves primarias de las tablas, y las comunas que se representen en negrita se referirán a las claves foráneas de las tablas.

2.1.1. Relación Contiene: zona contiene a zona

Como se observa en la Imagen 71 existe una relación reflexiva con una cardinalidad de tipo N:1 (de muchos a una). Pero al contener esta relación un valor mínimo de participación de

cero (con un MIN y MAX de cardinalidad de 0,N y 0,1) no será una relación obligatoria. Este tipo de relación nos indica que será preferible definir dos tablas: una tabla de la entidad *Zona* y otra tabla de la propia relación.

1.Tabla: Entidad Zona

Al analizar los atributos definidos para la entidad *Destino*, la primera tabla se definirá de esta manera:

e_zona (idzona, nombre, tipozona)

2.Tabla: Relación reflexiva de Zona

La segunda tabla se definirá de la siguiente manera:

r_zonazona (idzonazona, **idzona**, **idcontiene**)

Tablas creadas: zona y tipozona

Pero al crear la base de datos se ha decidido cambiar de criterio y crear una sola tabla. Este cambio se ha realizado para reflejar mejor que una zona solamente puede pertenecer a una zona padre, y también para ser más económicos en la creación de tablas (y ahorrarnos una tabla).

Otro cambio reseñable se ha realizado en la tabla **e_zona**, en la columna **tipozona**. Ésta al tener valores predefinidos se realizará un cambio para evitar que en ella se incluyan nombres de tipos que no estén dentro de esta lista de valores. Este cambio se fundamentará en crear una nueva tabla para contener estos valores y convertir la columna **tipozona** en una clave foránea referida al identificador de la tabla creada (y no contendrá valores nulos).

Por lo tanto, las tablas **e_zona** y **p_tipozona** contendrán estas columnas:

*e_zona (**idzona**, nombre, **idtipozona**, **idzonapadre**)*

*p_tipozona (**idtipozona**, descripciontipo)*

Como se observa, en la tabla **e_zona** la clave primaria lo conformará la columna **idzona**. Esta columna al ser una clave primaria no podrá contener valores nulos. La columna **idzonapadre** será una clave foránea, y se referirá a la columna **idzona** de esta tabla **e_zona**. Esta columna podrá contener valores nulos. Otra clave foránea será **idtipozona**, y se referirá al identificador de la tabla **p_tipozona**. Esta columna no podrá contener valores nulos.

FK **idzonapadre** → e_zona.idzona FK **idtipozona** → p_tipozona.idtipozona

Como se aprecia, en la tabla **p_tipozona** la clave primaria será la columna **idtipozona**. La columna **descripciontipo** contendrá valores predefinidos para el tipo de zona, y estos valores predefinidos serán *municipio*, *zona*, *pueblo*, *barrio*, *agrupacion* y *diseminado*.

2.1.2. Relación Pertenece: destino pertenece a zona

Como se observa en la Imagen 72 existe una relación binaria con una cardinalidad de tipo N:1 (de muchos a una), y contiene un valor mínimo de participación de uno (al tener un MIN

y MAX de cardinalidad de 1,1 y 0,N). Por lo tanto, habrá una relación obligatoria. Este tipo de relación nos indica que será preferible definir dos tablas: una tabla de la entidad *Destino y* otra tabla de la entidad *Zona*.

1.Tabla: Entidad Destino

Al analizar los atributos definidos para la entidad *Destino*, la primera tabla se definirá de esta manera:

e_destino (iddestino, **idzona**, nombre, **idtipodestino**, fotografia, geom, idportal, provincia, municipio, cp, codmun, idpoblacion, nombrebarrio, nombrecalle, idenvia, numeroportal, letraportal, nombrepoligono, nombreparcela, idinventario, depositogas, depositogasolina, idpropietario, contacto1, contacto, habitado, observacion)

2.Tabla: Entidad Zona

Esta tabla se definirá de la misma manera que en la relación anterior:

e_zona (idzona, nombre, **idtipozona**, **idzonapadre**)

Tablas creadas: destino, tipodestino, direccion, riesgopotencial y contacto

Pero al crear la base de datos se ha decidido realizar varios cambios en referencia a la tabla **e_destino**.

Un cambio se realizará en referencia a la tabla **e_destino**, a la columna **tipodestino**. Como en el caso anterior (en la tabla **e_zona**) se creará una tabla (la tabla **p_tipodestino**) que contenga la lista de valores predefinidos de la columna **tipodestino**. Esta columna **tipodestino** se convertirá en una clave foránea (referida al identificador de la tabla creada) y no contendrá valores nulos.

Otro cambio se realizará en referencia a las columnas que representan atributos compuestos. Es el caso de las columnas que aglutinan la información de direcciones, de los riesgos potenciales y del contacto con el propietario de la vivienda. En estos casos se plantea crear tablas que contengan estas columnas. De esta manera, se estructurará mejor la información, y se realizará cambios en su contenido sin acceder a la tabla **e_destino**.

Por lo tanto, las tablas **e_destino**, **p_tipodestino** y las tablas referidas a la reestructuración de la información contendrán estas columnas:

*e_destino (**iddestino**, **idzona**, nombre, **idtipodestino**, fotografia, geom, habitado, observacion)*

*p_tipodestino (**idtipodestino**, descripciontipo)*

*p_direccion (**iddestino**, idportal, provincia, municipio, cp, codmun, idpoblacion, nombrebarrio, nombrecalle, idenvia, numeroportal, letraportal, nombrepoligono, nombreparcela)*

*p_riesgopotencial (**iddestino**, idinventario, depositogas, depositogasolina)*

*p_contacto (**iddestino**, idpropietario, contacto1, contacto2)*

Como se observa, en la tabla **e_destino** la clave primaria lo conformará la columna **iddestino**. La columna **idzona** será una clave foránea, y se referirá a la columna **idzona** de esta tabla **e_zona**. Esta columna no podrá contener valores nulos. Otra clave foránea será la columna **idtipodestino**, y se referirá al identificador de la tabla **p_tipodestino**. Esta columna tampoco podrá contener valores nulos.

FK **idzona** → e_zona.idzona FK **idtipodestino** → p_tipodestino.idtipodestino

La columna **geom** de la tabla **e_destino** será un campo geométrico de tipo puntual.

Como se aprecia, en la tabla **p_tipodestino** la clave primaria será la columna **idtipodestino**. La columna **descripciontipo** contendrá valores predefinidos para el tipo de destino, y estos valores predefinidos serán *casa*, *borda*, *venta*, *iglesia*, *ermita*, *cementerio*, *escuela*, *cuadra*, *alojamiento* y *puente*.

En el caso, de las tablas **p_direccion**, **p_riesgopotencial** y **p_contacto** la clave primaria de cada una de ellas será la columna **iddestino**. Esta columna a su vez será clave foránea, y se referirá a la columna **iddestino** de la tabla **e_destino**. Las columnas **idportal**, **idpoblacion**, **idenvia**, **idinventario** e **idpropietario** serán claves foráneas a tablas externas (de Helpbidea, Catastro, Bomberos de Navarra, etc.).

FK **iddestino** → e_destino.iddestino

2.1.3. Relación Reside: habitante reside a destino

Como se observa en la Imagen 73 existe una relación binaria con una cardinalidad de tipo 1:N (de uno a muchos). Pero al contener esta relación un valor mínimo de participación de cero (con un MIN y MAX de cardinalidad de 0,N y 0,1) no será una relación obligatoria. Este tipo de relación nos indica que será preferible definir tres tablas: una tabla de la entidad *Habitante*, otra tabla de la entidad *Destino* y por último una tabla de la propia relación.

1.Tabla: Entidad Habitante

Al analizar los atributos definidos para la entidad *Habitante*, la primera tabla se definirá de esta manera:

e_habitante (idhabitante, nif, nombre, apellido1, apellido2, fechanacimiento, edad, contacto, historialclinico, fichapolicial)

2.Tabla: Entidad Destino

Esta tabla se definirá de la misma manera que en la relación anterior:

e_destino (iddestino, **idzona**, nombre, **idtipodestino**, fotografia, geom, habitado, observacion)

3.Tabla: Relación entre Habitante y Destino

La tercera y última tabla de esta relación 1:N se definirá de esta manera:

r_destinohabitante (iddestinohabitante, **iddestino**, **idhabitante**)

Tablas creadas: habitante y destinohabitante

En este caso al crear la base de datos se ha decidido realizar sólo un cambio en la tabla **e_habitante**. Este cambio se realizará en referencia a las columnas que representan atributos compuestos. Es el caso de las columnas que aglutinan la información del historial clínico y de la ficha policial del habitante de la vivienda. En estos casos se plantea crear tablas que contengan estas columnas. De esta manera, como en el caso anterior se pretende reestructurar mejor la información.

Por lo tanto, la tabla **e_habitante** y las tablas referidas a la reestructuración de la información contendrán estas columnas:

*e_habitante (**idhabitante**, nombre, apellido1, apellido2, nif, fechanacimiento, edad, contacto)*

*p_historialclinico (**idhabitante**, historialclinico, minusvalia, observacion)*

*p_fichapolicial (**idhabitante**, fichapolicial, antecedente, observacion)*

La tabla **r_habitantedestino** contendrá las mismas columnas:

*r_destinohabitante (**iddestinohabitante**, **iddestino**, **idhabitante**)*

Como se observa, en la tabla **e_habitante** la clave primaria lo conformará la columna **idhabitante**.

Como se observa, en la tabla **r_destinohabitante** la clave primaria será la columna **iddestinohabitante**. La columna **iddestino** será una clave foránea, y se referirá a la columna **iddestino** de esta tabla **e_destino**. Esta columna no podrá contener valores nulos. Otra clave foránea será la columna **idhabitante**, y se referirá al identificador de la tabla **e_habitante**. Esta columna tampoco podrá contener valores nulos.

FK **iddestino** → e_destino.iddestino FK **idhabitante** → e_habitante.idhabitante

En referencia a las tablas **p_historialclinico** y **p_fichapolicial** la clave primaria de cada una de ellas será la columna **idhabitante**. Esta columna a su vez será clave foránea, y se referirá a la columna **idhabitante** de la tabla **e_habitante**. Pero al referirse a datos reservados, no se dispone información necesaria para saber exactamente qué columnas hay que crear. Por lo tanto, en nuestra base de datos no se crearán dichas tablas.

FK **idhabitante** → e_habitante.idhabitante

2.1.4. Relación Es Vecino: destino es vecino de destino

Como se observa en la Imagen 74 existe una relación reflexiva con una cardinalidad de tipo N:M (de muchos a muchos), por lo tanto no será una relación obligatoria. Este tipo de relación nos indica que será preferible definir dos tablas: una tabla de la entidad *Destino*, y otra tabla para la propia relación.

1.Tabla: Entidad Destino

Esta tabla se definirá de la misma manera que en la anterior vez:

e_destino (iddestino, **idzona**, nombre, **idtipodestino**, fotografia, geom, habitado, observacion)

2.Tabla: Relación reflexiva de Destino

La segunda tabla se definirá de la siguiente manera:

r_destinovecindad (iddestinovecindad, **iddestino**, **idvecindad**, distanciavecindad)

Tablas creadas: destinovecindad

En este caso no habrá modificaciones en las tablas preestablecidas.

Por lo tanto, la tabla que se creará será la **r_destinovecindad** y contendrá las mismas columnas:

*r_destinovecindad (**iddestinovecindad**, **iddestino**, **idvecindad**, distanciavecindad)*

Como se aprecia, la clave primaria será la columna **iddestinovecindad**. La columna **iddestino** será una clave foránea, y se referirá a la columna **iddestino** de esta tabla **e_destino**. Esta columna no podrá contener valores nulos. Otra clave foránea será la columna **idvecindad**, y se referirá también a la columna **iddestino** de la tabla **e_destino**. Esta columna tampoco podrá contener valores nulos.

FK **iddestino** → e_destino.iddestino FK **idvecindad** → e_destino.iddestino

2.1.5. Relación Llega: ruta llega a destino

Como se observa en la Imagen 75 existe una relación binaria con una cardinalidad de tipo 1:N (de uno a muchos), y contiene un valor mínimo de participación de uno (al tener un MIN y MAX de cardinalidad de 0,N y 1,1). Por lo tanto, habrá una relación obligatoria. Este tipo de relación nos indica que será preferible definir dos tablas: una tabla de la entidad *Ruta* y otra tabla de la entidad *Destino*.

1.Tabla: Entidad Ruta

Al analizar los atributos definidos para la entidad *Habitante*, la primera tabla se definirá de esta manera:

e_ruta (idruta, **idsalida**, **iddestino**, descripcion, numerotramos, vehiculotransitable, distancia, tiempo)

2.Tabla: Entidad Destino

Esta tabla se definirá de la misma manera que en la anterior vez:

e_destino (iddestino, **idzona**, nombre, **idtipodestino**, fotografia, geom, habitado, observacion)

Tablas creadas: ruta

En este caso al crear la base de datos se ha decidido realizar un cambio en la tabla **e_ruta**. Este cambio ha sido la eliminación de las columnas **vehiculotransitable**, **distancia** y **tiempo**. Este cambio se ha realizado para que esta información se recopile en la tabla **e_tramo**, y no en esta entidad.

Por lo tanto, la tabla **e_ruta** contendrá estas columnas:

*e_ruta (**idruta**, **idsalida**, **iddestino**, descripcion, numerotramos)*

Como se observa, en la tabla **e_ruta** la clave primaria lo conformará la columna **idruta**. La columna **idsalida** será una clave foránea, y se referirá a la columna **idsalida** de esta tabla **e_salida**. Otra clave foránea será la columna **iddestino**, y se referirá a la columna **iddestino** de esta tabla **e_destino**. Estas dos columnas no podrán contener valores nulos.

FK **idsalida** → e_salida.idsalida FK **iddestino** → e_destino.iddestino

2.1.6. Relación Parte: ruta parte de salida

Como se observa en la Imagen 76 existe una relación binaria con una cardinalidad de tipo 1:N (de uno a muchos), y contiene un valor mínimo de participación de uno (al tener un MIN y MAX de cardinalidad de 0,N y 1,1). Por lo tanto, habrá una relación obligatoria. Este tipo de relación nos indica que será preferible definir dos tablas: una tabla de la entidad *Ruta* y otra tabla de la entidad *Salida*.

1.Tabla: Entidad Ruta

Esta tabla se definirá de la misma manera que en la anterior vez:

e_ruta (idruta, **idsalida**, **iddestino**, descripcion, numerotramos)

2.Tabla: Entidad Salida

Al analizar los atributos definidos para la entidad *Salida*, la segunda tabla se definirá de esta manera:

e_salida (idsalida, nombre, tiposalida, geom)

Tablas creadas: salida y tiposalida

Pero al crear la base de datos se ha decidido realizar un cambio en la tabla **e_salida**.

Este cambio se realizará en referencia a la tabla **e_salida**, a la columna **tiposalida**. Como en los casos anteriores (en las tablas **e_zona** y **e_destino**) se creará una tabla (la tabla **p_tiposalida**) que contenga la lista de valores predefinidos de la columna **tiposalida**. Esta columna **tiposalida** se convertirá en una clave foránea (referida al identificador de la tabla creada) y no contendrá valores nulos.

Por lo tanto, las tablas **e_salida** y **p_tiposalida** contendrán estas columnas:

*e_salida (**idsalida**, nombre, **idtiposalida**, geom)*

*p_tiposalida (**idtiposalida**, descripciontipo)*

Como se observa, en la tabla **e_salida** la clave primaria lo conformará la columna **idsalida**. La columna **idtiposalida**, y se referirá a la columna **idtiposalida** de la tabla **p_tiposalida**. Esta columna no podrá contener valores nulos.

FK **idtiposalida** → p_tiposalida.idtiposalida

La columna **geom** de la tabla **e_salida** será un campo geométrico de tipo puntual.

Como se aprecia, en la tabla **p_tiposalida** la clave primaria será la columna **idtiposalida**. La columna **descripciontipo** contendrá valores predefinidos para el tipo de salida, y estos valores predefinidos serán *parque bomberos*, *centro salud*, *hospital* y *comisaria policia*.

2.1.7. Relación Utiliza: servicio utiliza salida

Como se observa en la Imagen 77 existe una relación binaria con una cardinalidad de tipo N:M (de muchos a muchos), y por lo tanto no será una relación obligatoria. Este tipo de relación nos indica que será preferible definir tres tablas: una tabla de la entidad *Servicio*, otra tabla de la entidad *Salida* y por último una tabla de la propia relación.

1.Tabla: Entidad Servicio

Al analizar los atributos definidos para la entidad *Servicio*, la primera tabla se definirá de esta manera:

e_servicio (idservicio, tiposervicio, contacto)

2.Tabla: Entidad Salida

Esta tabla se definirá de la misma manera que en la anterior vez:

e_salida (idsalida, nombre, **idtiposalida**, geom)

3.Tabla: Relación entre Servicio y Salida

La tercera y última tabla de esta relación N:M se definirá de esta manera:

r_salidaservicio (idsalidaservicio, **idsalida**, **idservicio**)

Tablas creadas: servicio, tiposervicio y salidaservicio

Pero al crear la base de datos se ha decidido realizar un cambio en la tabla **e_salida**.

Este cambio se realizará en referencia a la tabla **e_servicio**, a la columna **tiposervicio**. Como en los casos anteriores (en las tablas **e_zona**, **e_destino** y **e_salida**) se creará una tabla (la tabla **p_tiposervicio**) que contenga la lista de valores predefinidos de la columna **tiposervicio**. Esta columna **tiposervicio** se convertirá en una clave foránea (referida al identificador de la tabla creada) y no contendrá valores nulos.

Por lo tanto, las tablas **e_servicio** y **p_tiposervicio** contendrán estas columnas:

*e_servicio (**idservicio**, **idtiposervicio**, contacto)*

*p_tiposervicio (**idtiposervicio**, descripciontipo)*

La tabla **r_salidaservicio** contendrá las mismas columnas:

*r_salidaservicio (**idsalidaservicio**, **idsalida**, **idservicio**)*

Como se observa, en la tabla **e_servicio** la clave primaria lo conformará la columna **idservicio**. La columna **idtiposervicio**, y se referirá a la columna **idtiposervicio** de la tabla **p_tiposervicio**. Esta columna tampoco podrá contener valores nulos.

FK **idtiposervicio** → p_tiposervicio.idtiposervicio

Como se aprecia, en la tabla **p_tiposervicio** la clave primaria será la columna **idtiposervicio**. La columna **descripciontipo** contendrá valores predefinidos para el tipo de servicio, y estos valores predefinidos serán *bomberos navarra*, *osasunbidea*, *policia foral*, *proteccion civil* y *policia municipal*.

Como se observa, en la tabla **r_salidaservicio** la clave primaria será la columna **idsalidaservicio**. La columna **idsalida** será una clave foránea, y se referirá a la columna **idsalida** de esta tabla **e_salida**. Esta columna no podrá contener valores nulos. Otra clave foránea será la columna **idservicio**, y se referirá a la columna **idservicio** de la tabla **e_servicio**. Esta columna tampoco podrá contener valores nulos.

FK **idsalida** → e_salida.idsalida FK **idservicio** → e_servicio.idservicio

2.1.8. Relación Pertenece: tramo pertenece a ruta

Como se observa en la Imagen 78 existe una relación binaria con una cardinalidad de tipo N:M (de muchos a muchos), y por lo tanto no será una relación obligatoria. Este tipo de relación nos indica que será preferible definir tres tablas: una tabla de la entidad *Tramo*, otra tabla de la entidad *Ruta* y por último una tabla de la propia relación.

1.Tabla: Entidad Tramo

Al analizar los atributos definidos para la entidad *Tramo*, la primera tabla se definirá de esta manera:

e_tramo (idtramo, tipotramo, nombre, geom)

2.Tabla: Entidad Ruta

Esta tabla se definirá de la misma manera que en la anterior vez:

e_ruta (idruta, **idsalida**, **iddestino**, descripcion, numerotramos)

3.Tabla: Relación entre Tramo y Ruta

La tercera y última tabla de esta relación N:M se definirá de esta manera:

r_rutatramo (idrutatramo, **idruta**, **idtramo** , orden)

Tablas creadas: tramo, tipotramo y rutatramo

Pero al crear la base de datos se ha decidido realizar dos cambios en la tabla **e_tramo**.

Un cambio se realizará en referencia a la tabla **e_tramo**, a la columna **tipotramo**. Como en los casos anteriores se creará una tabla (la tabla **p_tipotramo**) que contenga la lista de valores predefinidos de la columna **tipotramo**. Esta columna **tipotramo** se convertirá en una clave foránea (referida al identificador de la tabla creada) y no contendrá valores nulos.

El otro cambio se realizará añadiendo las columnas **longitud** y **tiempo**. Este cambio se ha realizado para recabar información de distancia y tiempo a nivel de tramo. Posteriormente facilitará el cálculo a nivel de ruta.

Por lo tanto, la tasblas **e_tramo** y **p_tipotramo** contendrán estas columnas:

*e_tramo (**idtramo**, **idtipotramo**, nombre, geom, longitud, tiempo)*

*p_tipotramo (**idtipotramo**, descripciontipo)*

Y la tabla **r_rutatramo** contendrá las mismas columnas:

*r_rutatramo (**idrutatramo**, **idruta**, **idtramo**, orden)*

Como se observa, en la tabla **e_tramo** la clave primaria lo conformará la columna **idtramo**. La columna **idtipotramo**, y se referirá a la columna **idtipotramo** de la tabla **p_tipotramo**. Esta columna no podrá contener valores nulos.

FK **idtipotramo** → p_tipotramo.idtipotramo

La columna **geom** de la tabla **e_tramo** será un campo geométrico de tipo lineal.

Como se aprecia, en la tabla **p_tipotramo** la clave primaria será la columna **idtipotramo**. La columna **descripciontipo** contendrá valores predefinidos para el tipo de tramo, y estos valores predefinidos serán *tramo carretera* y *tramo rural*.

Como se observa, en la tabla **r_rutatramo** la clave primaria será la columna **idrutatramo**. La columna **idruta** será una clave foránea, y se referirá a la columna **idruta** de esta tabla **e_ruta**. Esta columna no podrá contener valores nulos. Otra clave foránea será la columna **idtramo**, y se referirá a la columna **idtramo** de la tabla **e_tramo**. Esta columna tampoco podrá contener valores nulos.

FK **idruta** → e_ruta.idruta FK **idtramo** → e_tramo.idtramo

2.1.9. Relación Contiene: ruta contiene cruce

Como se observa en la Imagen 79 existe una relación binaria con una cardinalidad de tipo 1:N (de uno a muchos). Pero al contener esta relación un valor mínimo de participación de cero (con un MIN y MAX de cardinalidad de 1,N y 0,1) no será una relación obligatoria. Este tipo de relación nos indica que será preferible definir tres tablas: una tabla de la entidad *Ruta*, otra tabla de la entidad *Cruce* y por último una tabla de la propia relación.

1.Tabla: Entidad Ruta

Esta tabla se definirá de la misma manera que en la anterior vez:

e_ruta (idruta, **idsalida**, **iddestino**, descripcion, numerotramos)

2.Tabla: Entidad Cruce

Al analizar los atributos definidos para la entidad *Cruce*, la segunda tabla se definirá de esta manera:

e_cruce (idcruce, fotografia, observacion, geom)

3.Tabla: Relación entre Ruta y Cruce

La tercera y última tabla de esta relación N:M se definirá de esta manera:

r_cruceruta (idcruceruta, **idruta**, **idcruce**)

Tablas creadas: cruce y cruceruta

Pero al crear la base de datos se ha decidido realizar un cambio en la tabla **e_cruce**.

El cambio será eliminar la columna **fotografia**. Este cambio se ha realizado por no disponer información sobre este aspecto.

Por lo tanto, la tabla **e_cruce** contendrá estas columnas:

*e_cruce (**idcruce**, observacion, geom)*

Y la tabla **r_cruceruta** contendrá las mismas columnas:

*r_cruceruta (**idcruceruta**, **idruta**, **idcruce**)*

Como se observa, en la tabla **e_cruce** la clave primaria lo conformará la columna **idcruce**. La columna **geom** será un campo geométrico de tipo puntual.

Como se observa, en la tabla **r_cruceruta** la clave primaria será la columna **idcruceruta**. La columna **idcruce** será una clave foránea, y se referirá a la columna **idcruce** de esta tabla **e_cruce**. Esta columna no podrá contener valores nulos. Otra clave foránea será la columna **idruta**, y se referirá a la columna **idruta** de la tabla **e_ruta**. Esta columna tampoco podrá contener valores nulos.

FK **idruta** → e_ruta.idruta FK **idcruce** → e_cruce.idcruce

2.1.10. Relación Contiene: ruta contiene punto información

Como se observa en la Imagen 80 existe una relación binaria con una cardinalidad de tipo N:M (de muchos a muchos), y por lo tanto no será una relación obligatoria. Este tipo de relación nos indica que será preferible definir tres tablas: una tabla de la entidad *Ruta*, otra de la entidad *Punto Información* y por último una tabla de la propia relación.

1.Tabla: Entidad Ruta

Esta tabla se definirá de la misma manera que en la anterior vez:

e_ruta (idruta, **idsalida**, **iddestino**, descripcion, numerotramos)

2.Tabla: Entidad Punto Información

Al analizar los atributos definidos para la entidad *Punto Información*, esta tabla se definirá de esta manera:

e_puntoinfo (idpuntoinfo, nombre, descripcion, geom)

3.Tabla: Relación entre Ruta y Punto Información

La tercera y última tabla de esta relación N:M se definirá de esta manera:

r_puntoinforuta (idpuntoinforuta, **idruta**, **idpuntoinfo**)

Tablas creadas: puntoinfo y puntoinforuta

Al crear la base de datos no se han realizado ningún cambio.

Por lo tanto, la tabla **e_puntoinfo** contendrá estas columnas:

*e_puntoinfo (**idpuntoinfo**, nombre, descripcion, geom)*

La tabla **r_puntoinforuta** contendrá las mismas columnas:

*r_puntoinforuta (**idpuntoinforuta**, **idruta**, **idpuntoinfo**)*

Como se observa, en la tabla **e_puntoinfo** la clave primaria lo conformará la columna **idpunto-info**. La columna **geom** será un campo geométrico de tipo puntual.

Como se aprecia, en la tabla **r_puntoinforuta** la clave primaria será la columna **idpuntoinforuta**. La columna **idpuntoinfo** será una clave foránea, y se referirá a la columna **idpuntoinfo** de esta tabla **e_puntoinfo**. Otra clave foránea será la columna **idruta**, y se referirá a la columna **idruta** de la tabla **e_ruta**. Estas dos columnas no podrán contener valores nulos.

FK **idruta** → e_ruta.idruta FK **idpuntoinfo** → e_puntoinfo.idpuntoinfo

2.1.11. Relación subclase: tramo es tramo rural y tramo carretera

Como se observa en la Imagen 81 existe una relación de subclase. Por lo tanto, la tabla que representa una clase inferior, heredará de la tabla padre la columna que representa la clave primaria.

Tabla creada: tramorural

La tabla **e_puntoinfo** contendrá estas columnas:

*e_tramorural (**idtramo**, **idestado**, fecha, curvacerrada, pendientealta, advertencia)*

Como se observa, en la tabla **e_tramorural** la clave primaria lo conformará la columna **idtramo**. Estas columnas también serán claves foráneas. Esta columna se referirá a la columna **idtramo**

de la tabla **e_tramo**. La columna **idestado**, y se referirá a la columna **idestado** de la tabla **e_estado**. Esta columna tampoco podrá contener valores nulos.

FK **idtramo** → e_tramo.idtramo FK **idestado** → e_estado.idestado

2.1.12. Relación Contiene: tramo rural contiene suelo

Como se observa en la Imagen 82 existe una relación binaria con una cardinalidad de tipo N:M (de muchos a muchos), y por lo tanto no será una relación obligatoria. Este tipo de relación nos indica que será preferible definir tres tablas: una tabla de la entidad *Tramo Rural*, otra tabla de la entidad *Suelo* y por último una tabla de la propia relación.

1.Tabla: Entidad Tramo Rural

Esta tabla se definirá de la misma manera que en la anterior vez:

e_tramorural (**idtramo**, **idestado**, fecha, curvacerrada, pendientealta, advertencia)

2.Tabla: Entidad Suelo

Al analizar los atributos definidos para la entidad *Suelo*, la segunda tabla se definirá de esta manera:

e_suelo (idsuelo, tiposuelo, caracteristica)

3.Tabla: Relación entre Tramo Rural y Suelo

La tercera y última tabla de esta relación N:M se definirá de esta manera:

r_suelotramorural (idsuelotramorural, **idsuelo**, **idtramo**)

Tablas creadas: suelo, tiposuelo y suelotramorural

Pero al crear la base de datos se ha decidido realizar un cambio en la tabla **e_suelo**.

Este cambio se realizará en referencia a la tabla **e_suelo**, a la columna **tiposuelo**. Como en los casos anteriores se creará una tabla (la tabla **p_tiposuelo**) que contenga la lista de valores predefinidos de la columna **tiposuelo**. Esta columna **tiposuelo** se convertirá en una clave foránea (referida al identificador de la tabla creada) y no contendrá valores nulos.

Por lo tanto, las tablas **e_suelo** y **p_tiposuelo** contendrán estas columnas:

*e_suelo (**idsuelo**, **idtiposuelo**, caracteristica)*

*p_tiposuelo (**idtiposuelo**, descripciontipo)*

La tabla **r_suelotramorural** contendrá las mismas columnas:

*r_suelotramorural (**idsuelotramorural**, **idsuelo**, **idtramo**)*

Como se observa, en la tabla **e_suelo** la clave primaria lo conformará la columna **idsuelo**. La columna **idtiposuelo**, y se referirá a la columna **idtiposuelo** de la tabla **p_tiposuelo**. Esta columna tampoco podrá contener valores nulos.

FK **idtiposuelo** → p_tiposuelo.idtiposuelo

Como se aprecia, en la tabla **p_tiposuelo** la clave primaria será la columna **idtiposuelo**. La columna **descripciontipo** contendrá valores predefinidos para el tipo de suelo, y estos valores predefinidos serán *hormigón*, *asfalto*, *todo uno*, *tierra*, *hierba* y *sin recubrimiento*.

Como se observa, en la tabla **r_suelotramorural** la clave primaria será la columna **idsuelotramorural**. La columna **idsuelo** será una clave foránea, y se referirá a la columna **idsuelo** de esta tabla **e_suelo**. Otra clave foránea será la columna **idtramo**, y se referirá a la columna **idtramo** de la tabla **e_tramorural**. Estas dos columnas no podrán contener valores nulos.

FK **idsuelo** → e_suelo.idsuelo FK **idtramo** → e_tramorural.idtramo

2.1.13. Relación Contiene: tramo rural contiene estado

Como se observa en la Imagen 83 existe una relación binaria con una cardinalidad de tipo N:1 (de muchos a una), y contiene un valor mínimo de participación de uno (al tener un MIN y MAX de cardinalidad de 1,1 y 0,N). Por lo tanto, habrá una relación obligatoria. Este tipo de relación nos indica que será preferible definir tres tablas: una tabla de la entidad *Tramo Rural* y otra tabla de la entidad *Estado*.

1.Tabla: Entidad Tramo Rural

Esta tabla se definirá de la misma manera que en la anterior vez:

e_tramorural (**idtramo**, **idestado**, fecha, curvacerrada, pendientealta, advertencia)

2.Tabla: Entidad Estado

Al analizar los atributos definidos para la entidad *Estado*, la segunda tabla se definirá de esta manera:

e_estado (idestado, tipoestado, caracteristica)

Tablas creadas: estado y tipoestado

Pero al crear la base de datos se ha decidido realizar un cambio en la tabla **e_estado**.

Este cambio se realizará en referencia a la tabla **e_estado**, a la columna **tipoestado**. Se creará una tabla (la tabla **p_tipoestado**) que contenga la lista de valores predefinidos de la columna **tipoestado**. Esta columna **tipoestado** se convertirá en una clave foránea (referida al identificador de la tabla creada) y no contendrá valores nulos.

Por lo tanto, las tablas **e_estado** y **p_tipoestado** contendrán estas columnas:

*e_estado (**idestado**, **idtipoestado**, caracteristica)*

*p_tipoestado (**idtipoestado**, descripciontipo)*

Como se observa, en la tabla **e_estado** la clave primaria lo conformará la columna **idestado**. La columna **idtipoestado**, y se referirá a la columna **idtipoestado** de la tabla **p_tipoestado**. Esta columna no podrá contener valores nulos.

FK **idtipoestado** → p_tipoestado.idtipoestado

Como se aprecia, en la tabla **p_tipoestado** la clave primaria será la columna **idtipoestado**. La columna **descripciontipo** contendrá valores predefinidos para el tipo de estado, y estos valores predefinidos serán *mal estado*, *estado regular* y *buen estado*.

2.1.14. Relación Supera: vehículo supera tramo

Como se observa en la Imagen 84 existe una relación binaria con una cardinalidad de tipo N:M (de muchos a muchos), y por lo tanto no será una relación obligatoria. Este tipo de relación nos indica que será preferible definir tres tablas: una tabla de la entidad *Vehiculo*, otra tabla de la entidad *Tramo* y por último una tabla de la propia relación.

1.Tabla: Entidad Vehiculo

Al analizar los atributos definidos para la entidad *Vehiculo*, la segunda tabla se definirá de esta manera:

e_vehiculo (idvehiculo, **idservicio**, tipovehiculo, limitacion)

2.Tabla: Entidad Tramo

Esta tabla se definirá de la misma manera que en la anterior vez:

e_tramo (idtramo, **idtipotramo**, nombre, geom, longitud, tiempo)

3.Tabla: Relación entre Vehiculo y Tramo

La tercera y última tabla de esta relación N:M se definirá de esta manera:

r_tramovehiculo (idtramovehiculo, **idtramo**, **idvehiculo**)

Tablas creadas: vehiculo, tipovehiculo y tramovehiculo

Pero al crear la base de datos se ha decidido realizar un cambio en la tabla **e_vehiculo**.

Este cambio se realizará en referencia a la tabla **e_vehiculo**, a la columna **tipovehiculo**. Se creará una tabla (la tabla **p_tipovehiculo**) que contenga la lista de valores predefinidos de la columna **tipovehiculo**. Esta columna **tipovehiculo** se convertirá en una clave foránea (referida al identificador de la tabla creada) y no contendrá valores nulos.

Por lo tanto, las tablas **e_vehiculo** y **p_tipovehiculo** contendrán estas columnas:

*e_vehiculo(**idvehiculo**, **idservicio**, **idtipovehiculo**, limitacion)*

*p_tipovehiculo (**idtipovehiculo**, descripciontipo)*

La tabla **r_tramovehiculo** contendrá las mismas columnas:

*r_tramovehiculo (**idtramovehiculo**, **idtramo**, **idvehiculo**)*

Como se observa, en la tabla **e_vehiculo** la clave primaria lo conformará la columna **idvehiculo**. La columna **idservicio** será una clave foránea, y se referirá a la columna **idservicio** de esta ta-

bla **e_servicio**. Otra clave foránea será la columna **idtipovehiculo**, y se referirá a la columna **idtipovehiculo** de la tabla **p_tipovehiculo**. Estas dos columnas no podrán contener valores nulos.

FK **idservicio** → e_servicio.idservicio FK **idtipovehiculo** → p_tipovehiculo.idtipovehiculo

Como se aprecia, en la tabla **p_tipovehiculo** la clave primaria será la columna **idtipovehiculo**. La columna **descripciontipo** contendrá valores predefinidos para el tipo de vehiculo, y estos valores predefinidos serán *vehiculo B-741*, *vehiculo B-731*, *vehiculo B-752*, *vehiculo 4X4*, *camion de monte sin remolque*, *autobomba forestal*, *bomba rural ligero*, *bomba forestal pesado* y *ambulancia*.

Como se observa, en la tabla **r_tramovehiculo** la clave primaria será la columna **idtramovehiculo**. La columna **idtramo** será una clave foránea, y se referirá a la columna **idtramo** de esta tabla **e_tramo**. Otra clave foránea será la columna **idvehiculo**, y se referirá a la columna **idvehiculo** de la tabla **e_vehiculo**. Estas dos columnas no podrán contener valores nulos.

FK **idtramo** → e_tramo.idtramo FK **idvehiculo** → e_vehiculo.idvehiculo

2.1.15. Relación Dispone: servicio dispone vehículo

Como se observa en la Imagen 85 existe una relación binaria con una cardinalidad de tipo 1:N (de uno a muchos). Al contener esta relación un valor mínimo de participación de uno (con un MIN y MAX de cardinalidad de 1,N y 1,1) será una relación obligatoria. Este tipo de relación nos indica que será preferible definir dos tablas: una tabla de la entidad *Servicio* y otra de la entidad *Vehiculo*.

1.Tabla: Entidad Servicio

Esta tabla se definirá de la misma manera que en la anterior vez:

e_servicio (idservicio, **idtiposervicio**, contacto)

2.Tabla: Entidad Vehiculo

Esta tabla se definirá de la misma manera que en la anterior vez:

e_vehiculo(idvehiculo, **idservicio**, **idtipovehiculo**, limitacion)

Tablas creadas: ninguna

En esta relación no se crearán ninguna nueva tabla.

2.1.16. Resumen de las tablas a crear

En definitiva, éstas son las 32 tablas que conformarán la estructura de nuestra base de datos:

1. Tablas creadas a partir de entidades:

   e_zona (idzona, nombre, **idtipozona**, **idzonapadre**)

   e_destino (iddestino, **idzona**, nombre, **idtipodestino**, fotografia, geom, habitado, observacion)

   e_habitante (idhabitante, nombre, apellido1, apellido2, nif, fechanacimiento, edad, contacto)

e_ruta (idruta, **idsalida**, **iddestino**, descripcion, numerotramos)

e_salida (idsalida, nombre, **idtiposalida**, geom)

e_servicio (idservicio, **idtiposervicio**, contacto)

e_tramo (idtramo, **idtipotramo**, nombre, geom, longitud, tiempo)

e_cruce (idcruce, observacion, geom)

e_puntoinfo (idpuntoinfo, nombre, descripcion, geom)

e_tramorural (**idtramo**, **idestado**, fecha, curvacerrada, pendientealta, advertencia)

e_suelo (idsuelo, **idtiposuelo**, caracteristica)

e_estado (idestado, **idtipoestado**, caracteristica)

e_vehiculo(idvehiculo, **idservicio**, **idtipovehiculo**, limitacion)

2. Tablas creadas a partir de la reestructuración de las columnas:

p_tipozona (idtipozona, descripciontipo)

p_tipodestino (idtipodestino, descripciontipo)

p_tiposalida (idtiposalida, descripciontipo)

p_tiposervicio (idtiposervicio, descripciontipo)

p_tipotramo (idtipotramo, descripciontipo)

p_tiposuelo (idtiposuelo, descripciontipo)

p_tipoestado (idtipoestado, descripciontipo)

p_tipovehiculo (idtipovehiculo, descripciontipo)

p_direccion (**iddestino**, idportal, provincia, municipio, cp, codmun, idpoblacion, nombrebarrio, nombrecalle, idenvia, numeroportal, letraportal, nombrepoligono, nombreparcela)

p_riesgopotencial (**iddestino**, idinventario, depositogas, depositogasolina)

p_contacto (**iddestino**, idpropietario, contacto1, contacto2)

3. Tablas creadas a partir de relaciones:

r_destinohabitante (iddestinohabitante, **iddestino**, **idhabitante**)

r_destinovecindad (iddestinovecindad, **iddestino**, **idvecindad**, distanciavecindad)

r_salidaservicio (idsalidaservicio, **idsalida**, **idservicio**)

r_rutatramo (idrutatramo, **idruta**, i**dtramo**, orden)

r_cruceruta (idcruceruta, **idruta**, **idcruce**)

r_puntoinforuta (idpuntoinforuta, **idruta**, **idpuntoinfo**)

r_suelotramorural (idsuelotramorural, **idsuelo**, **idtramo**)

r_tramovehiculo (idtramovehiculo, **idtramo**, **idvehiculo**)

2.1.17. Dominio de las comunas creadas

El dominio de cada columna (o atributo) definida será de tipo texto, numérico, fecha, tiempo, booleano, espacial y valores predefinidos.

- Número de identificación o referencia: cada uno de ellos serán de tipo *Numérico entero positivo*.

  idzona, idtipozona, iddestino, idtipodestino, iddestinovecindad, idhabitante, iddestinohabitante, idsalida, idtiposalida, idruta, idservicio, idtiposervicio, idsalidaservicio, idtramo, idtipotramo, idrutatramo, idcruce, idcruceruta, idpuntoinfo, idpuntoinforuta, idestado, idtipoestado, idsuelo, idtiposuelo, idsuelotramorural, idvehiculo, idtipovehiculo, idtramovehiculo, idzonapadre (e_zona), idvecindad (r_destinovecindad), idportal, idpoblacion, idenvia, idinventario y idpropietario (p_direccion).

- Nombres de entidades: cada uno de ellos serán de tipo *Texto (100)*.

  nombre (e_zona, e_destino, e_habitante, e_salida, e_tramo, e_puntoinfo),
  provincia, municipio, nombrebarrio, nombrecalle, letraportal, nombrepoligono,
  nombreparcela (p_direccion).

- Apellidos de habitantes de diseminados: cada uno de ellos serán de tipo *Texto (150)*.

  apellido1, apellido2 (e_habitante).

- Teléfono de contacto: cada uno de ellos serán de tipo *Texto (15)*.

  contacto1, contacto2 (p_contacto), contacto (e_habitante, e_servicio).

- Código del documento de identificación: este atributo será de tipo *Texto (9)*.

  nif (e_habitante).

- Direcciones URL de fotografías: cada uno de ellos serán de tipo *Texto (1000)*.

  fotografia (e_destino).

- Observaciones y descripciones sobre alguna característica: cada uno de ellos serán de tipo *Texto (1000)*.

  observacion (e_destino, e_cruce), descripcion (e_ruta, e_puntoinfo),
  caracteristica (e_estado, e_suelo), advertencia (e_tramorural), limitacion (e_vehiculo).

- Distancias y longitud en metros: cada uno de ellos serán de tipo *Numérico real positivo*.

  distanciavecindad (r_destinovecindad), longitud (e_tramo).

- Código postal, edad de una persona, orden, etc.: este atributo será de tipo *Numérico entero positivo*.

  cp, codmun, numeroportal (p_direccion), edad (e_habitante), numerotramos (e_ruta),
  orden (r_rutatramo).

- Fecha de nacimiento, fecha de actualizaciones, etc.: este atributo será de tipo *Fecha (año, mes, día)*.

  fechanacimiento (e_habitante), fecha (e_tramorural).

- Tiempo de respuesta (isócronas sanitarias, etc.): este atributo será de tipo *Intervalo* (de tiempo).

  tiempo (e_tramo).

- Existencia de riesgos, dificultades de transito, habitantes, etc.: cada uno de ellos serán de tipo *Booleano*.

  depositogas, depositogasolina (p_riesgopotencial), pendientealta, curvacerrada (e_tramorural), habitado (e_destino).

- Tipos o características reseñables (*descripciontipo*): cada uno de ellos tendrán valores predefinidos.

descripciontipo (p_tipozona): { *diseminado, agrupacion, barrio, pueblo, zona, municipio* }

descripciontipo (p_tipodestino): { *casa, borda, venta, iglesia, ermita, cementerio, escuela, cuadra, alojamiento, puente* }

descripciontipo (p_tiposalida): { *parque bomberos, centro salud, hospital, comisaria policia* }

descripciontipo (p_tiposervicio): { *bomberos navarra, osasunbidea, policia foral, proteccion civil, policia municipal* }

descripciontipo (p_tipotramo): { *tramo carretera, tramo rural* }

descripciontipo (p_tipoestado): { *buen estado, estado regular, mal estado* }

descripciontipo (p_tiposuelo): { *asfalto, hormigon, todo uno, tierra, hierba, sin recubrimiento* }

descripciontipo (p_tipovehiculo): { *vehiculo B-741, vehiculo B-731, vehiculo B-752, vehiculo 4X4, camion monte sin remolque, autobomba forestal, bomba rural ligero, bomba forestal pesado, ambulancia* }

- Información espacial: cada uno de ellos serán de tipo *Geometry (de subtipo Point y LineString)*.

  geom de tipo puntual (e_destino, e_salida, e_cruce, e_puntoinfo) y geom de tipo lineal (e_tramo).

  Hay que mencionar que la información espacial de esta base de datos podrá ser modificada para incluir información espacial en 3D. Es decir, los diseminados, tramos, cruces y puntos de información de esta base de datos podrán contener subtipos de *Geometry* que incluyan la coordenada Z. De esta manera, se modificarán los tipos *Point* por *PointZ* y *LineString* por *LineStringZ*. Esta información puede ser interesante para trabajar con desniveles o tener en cuenta predicciones de cotas de nieve.

2.1.18. Diagrama del modelo relacional

En el diagrama del modelo relacional se pretende reflejar principalmente varios aspectos del diseño:

- Las tablas. Conformarán la unidad de almacenamiento principal de la base de datos.
- Las columnas de las tablas. Representarán los campos de cada uno de los registros almacenados.
- La clave primaria de la tabla. Representará la columna que desempeñará la función de identificador único de cada registro. Podrán estar compuestos por varias columnas, y no habrá registros con el mismo valor en esta composición de columnas. En el diagrama, las columnas que son claves primarias contendrán un “PK” al principio del nombre, y estarán diferenciadas del resto de columnas.
- Las claves foráneas de la tabla. Representa la clave primaria de la tabla que está relacionada. Es decir, se utilizará la clave foránea para expresar la relación existente entre dos tablas. En el diagrama, las columnas que son claves foráneas contendrán un “FK” al principio del nombre, y también contendrán un número para diferenciarse de otras claves foráneas que podrían existir en una misma tabla.
- Las restricciones de las columnas. Representan restricciones referidas a los registros almacenados en una columna. Por ejemplo, restricciones de los dominios que contienen las columnas:
  - La restricción de unicidad existirá en todas las columnas que representen claves primarias.
  - La restricción de existencia evitará la aparición de los valores nulos en las columnas. En este caso, en el diagrama las columnas que representan esta restricción se escribirán en negrita. Todas las columnas que representan claves primarias tendrán esta restricción por defecto.
- Las relaciones entre las tablas. Representan el tipo de relación que existe entre dos tablas.

A continuación, se utilizará un diagrama para explicar el modelo relacional de nuestro proyecto. Más concretamente, se utilizarán cuatro imágenes que conformarán partes diferenciadas de este diagrama.

Todas las tablas representadas contienen una sola columna que representa la clave primaria. Todas las columnas representadas como clave primaria evitan tener registros duplicados (al tener valores únicos), y no contienen valores nulos. Las columnas que representan claves foráneas hacen referencia a columnas externas (de otras tablas) que son claves primarias. Por lo tanto, contendrán el mismo dominio que las columnas referidas. Pero no contendrán las mismas restricciones. Es decir, las columnas que representan claves foráneas no contienen la restricción de unicidad ni la restricción de existencia que contiene la columna externa que hace referencia.

Primera parte del diagrama

Como se aprecia en la Imagen 87, existen diez tablas (relacionadas entre si) que representan la información referida a los diseminados: por un lado están las tablas e_destino, p_tipodes-

tino, p_direccion, p_riesgopotencial y p_contacto; por otro lado las tablas e_zona y p_tipozona; también está la tabla e_habitante; y por último existen las referidas a las relaciones entre tablas r_destinovecindad y r_habitantedestino.

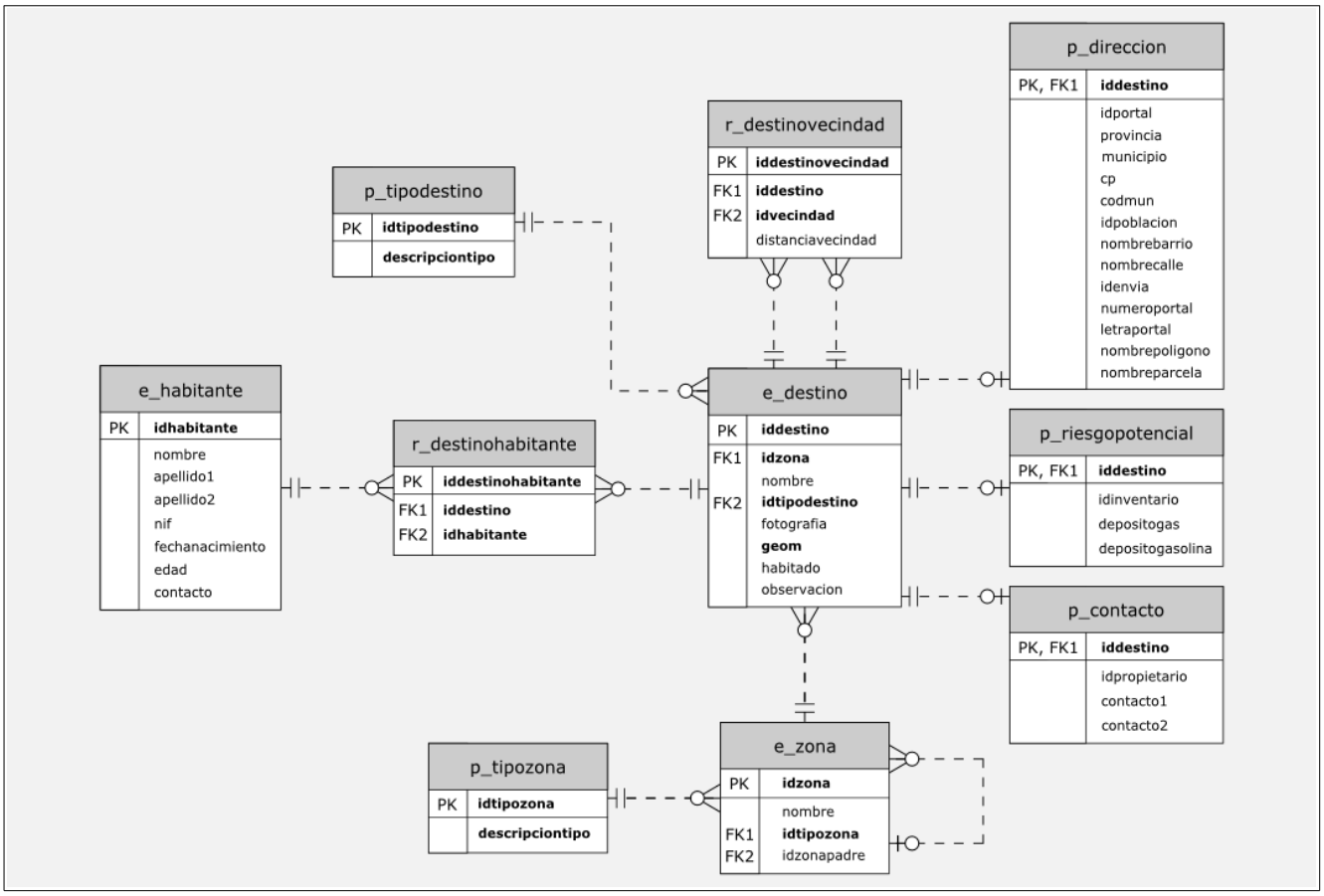

Imagen 87: Modelo relacional donde se aprecia la información sobre diseminados

En estas tablas, las columnas que no contienen valores nulos son las siguientes:

- Las columnas que representan claves primarias (PK): iddestino, idtipodestino, idzona, idtipozona, idhabitante, iddestinovecindad y iddestinohabitante.
- Las columnas que representan claves foráneas (FK): iddestino, idvecindad, idtipodestino, idzona, idtipozona y idhabitante. Existe una columna FK que puede contener valores nulos: idzonapadre.
- Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tipodestino y p_tipozona) y geom (de la tabla e_destino).

  En este caso, las relaciones entre tablas son de tipo N:M, N:1 y 1:1.
- La relación reflexiva de tipo N:M de la entidad Destino se representa realizando dos uniones de tipo (N,0) y (1,1) entre la tabla r_destinovecindad y e_destino.
- La relación reflexiva de tipo N:1 de la entidad Zona se representa realizando una unión de tipo (N,0) y (0,1) en la misma tabla e_zona.

- La relación de tipo 1:N entre la entidad Destino y la entidad Habitante se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_destinohabitante y e_habitante; y otra de tipo (0,N) y (1,1) entre la tabla r_destinohabitante y e_destino.
- Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a la relación existente entre la tabla e_destino y la tabla p_tipodestino, y la relación entre la tabla e_zona y la tabla p_tipozona. En estos dos casos se representará realizando una unión de tipo (0,N) y (1,1).
- Las relaciones de tipo 1:1 referidas a la reestructuración de la información del diseminado. En este caso se referirá a las relaciones existentes entre la tabla e_destino y la tabla p_dirección, entre la tabla e_destino y la tabla p_riesgopotencial, y entre la tabla e_destino y la tabla p_contacto. En estos tres casos se representará realizando una unión de tipo (1,1) y (0,1).

Segunda parte del diagrama

Como se aprecia en la Imagen 88 existen nueve tablas (relacionadas entre si) que representan la información referida a las rutas de acceso: por un lado está la tabla *e_ruta*; por otro lado las tablas *e_salida* y *p_tiposalida*; también está la tabla *e_destino* (antes mencionada); las tablas *e_servicio* y *p_tiposervicio*; las tablas *e_vehiculo* y *p_tipovehiculo*; y por último existen las referidas a las relaciones entre tablas en este caso la tabla *r_salidaservicio*.

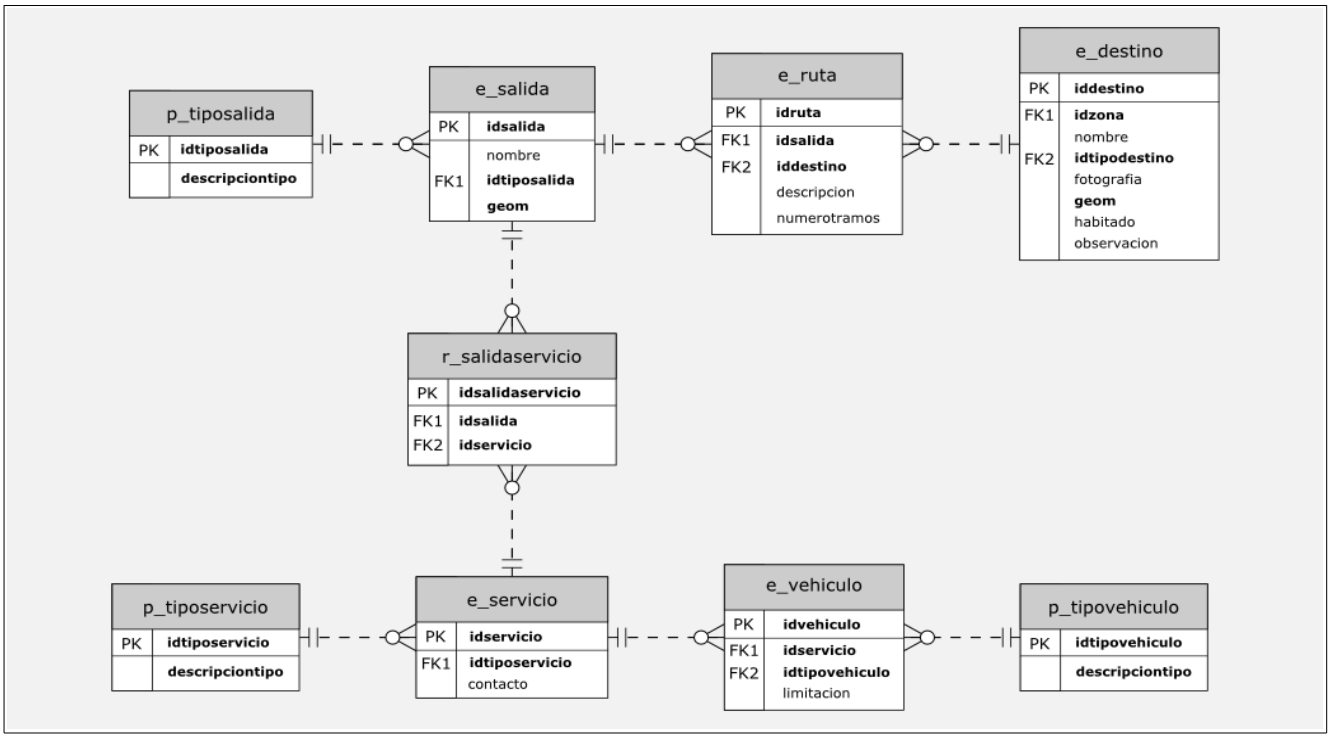

Imagen 88: Modelo relacional donde se aprecia la información sobre rutas de acceso

En estas tablas, las columnas que no contienen valores nulos son las siguientes:

- Las columnas que representan claves primarias (PK): idruta, idsalida, idtiposalida, iddestino, idservicio, idtiposervicio, idvehiculo, idtipovehiculo y idsalidaservicio.

- Las columnas que representan claves foráneas (FK): idsalida, idtiposalida, iddestino, idtipodestino, idservicio, idtiposervicio y idtipovehiculo.
- Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tiposalida, p_tiposervicio, y p_tipovehiculo) y geom (de la tabla e_salida y e_destino).

En este caso, las relaciones entre tablas son de tipo N:M y N:1.

- La relación de tipo N:M entre la entidad Salida y la entidad Servicio se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_salidaservicio y e_salida; y otra de tipo (0,N) y (1,1) entre la tabla r_salidaservicio y e_servicio.
- La relación de tipo N:1 entre la entidad Ruta y la entidad Salida se representa realizando una unión de tipo (0,N) y (1,1) entre la tabla e_ruta y e_salida.
- La relación de tipo N:1 entre la entidad Ruta y la entidad Destino se representa realizando una unión de tipo (0,N) y (1,1) entre la tabla e_ruta y e_destino.
- La relación de tipo N:1 entre la entidad Vehiculo y la entidad Servicio se representa realizando una unión de tipo (0,N) y (1,1) entre la tabla e_vehiculo y e_servicio.
- Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a la relación existente entre la tabla e_salida y la tabla p_tiposalida, e_servicio y la tabla p_tiposervicio, y la relación entre la tabla e_vehiculo y la tabla p_tipovehiculo. En estos tres casos se representará realizando una unión de tipo (0,N) y (1,1).

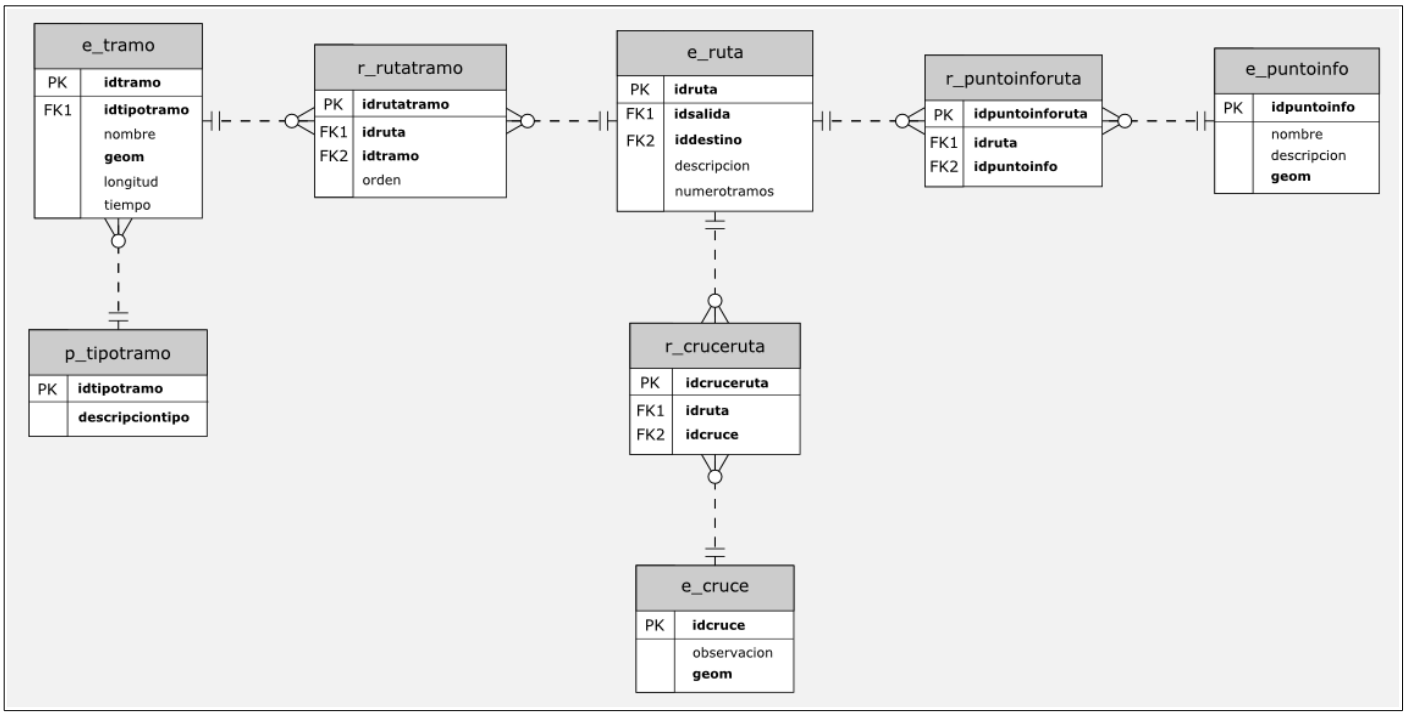

Imagen 89: Modelo relacional donde se aprecia la información sobre rutas de acceso

Tercera parte del diagrama

Como se aprecia en la Imagen 89 existen ocho tablas (relacionadas entre si) que representan también la información referida rutas de acceso: por un lado está la tabla *e_ruta* (antes men-

cionada); por otro lado las tablas *e_tramo* y *p_tipotramo*; también está la tabla *e_puntoinfo* y la tabla *e_cruce*; y por último existen las referidas a las relaciones entre tablas *r_rutatramo*, *r_puntoinforuta* y *r_cruceruta*.

En estas tablas, las columnas que no contienen valores nulos son las siguientes:

- Las columnas que representan claves primarias (PK): idruta, idtramo, idtipotramo, idpuntoinfo, idcruce, idrutatramo, idpuntoinforuta y idcruceruta.
- Las columnas que representan claves foráneas (FK): idsalida, iddestino, idruta, idtramo, idtipotramo, idpuntoinfo y idcruce.
- Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tipotramo) y geom (de la tabla e_tramo, e_puntoinfo y e_cruce).

En este caso, las relaciones entre tablas son de tipo N:M y N:1.

- La relación de tipo N:M entre la entidad Ruta y la entidad Tramo se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_rutatramo y e_ruta; y otra de tipo (0,N) y (1,1) entre la tabla r_rutatramo y e_tramo.
- La relación de tipo N:M entre la entidad Ruta y la entidad Punto Información se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_puntoinforuta y e_ruta; y otra de tipo (0,N) y (1,1) entre la tabla r_puntoinforuta y e_puntoinfo.
- La relación de tipo N:1 entre la entidad Ruta y la entidad Cruce se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_cruceruta y e_ruta; y otra de tipo (0,N) y (1,1) entre la tabla r_cruceruta y e_cruce.
- Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a la relación existente entre la tabla e_tramo y la tabla p_tipotramo. En este caso se representará realizando una unión de tipo (0,N) y (1,1).

Cuarta parte del diagrama

Como se aprecia en la Imagen 90 existen nueve tablas (relacionadas entre si) que representan la información referida a los tramos de rutas: por un lado está la tabla *e_tramo* (antes mencionada); por otro lado las tablas *e_vehiculo* (antes mencionada); también está la tabla *e_tramorural*; las tablas *e_estado* y *p_tipoestado*; las tablas *e_suelo* y *p_tiposuelo*; y por último existen las referidas a las relaciones entre tablas *r_tramovehiculo* y *r_suelotramorural*.

En estas tablas, las columnas que no contienen valores nulos son las siguientes:

- Las columnas que representan claves primarias (PK): idtramo, idvehiculo, idestado, idtipoestado, idsuelo, idtiposuelo, idtramovehiculo y idsuelotramorural.
- Las columnas que representan claves foráneas (FK): idtramo, idvehiculo, idtipovehiculo, idservicio, idestado, idtipoestado, idsuelo y idtiposuelo.
- Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tipoestado y p_tiposuelo) y geom (de la tabla e_tramo).

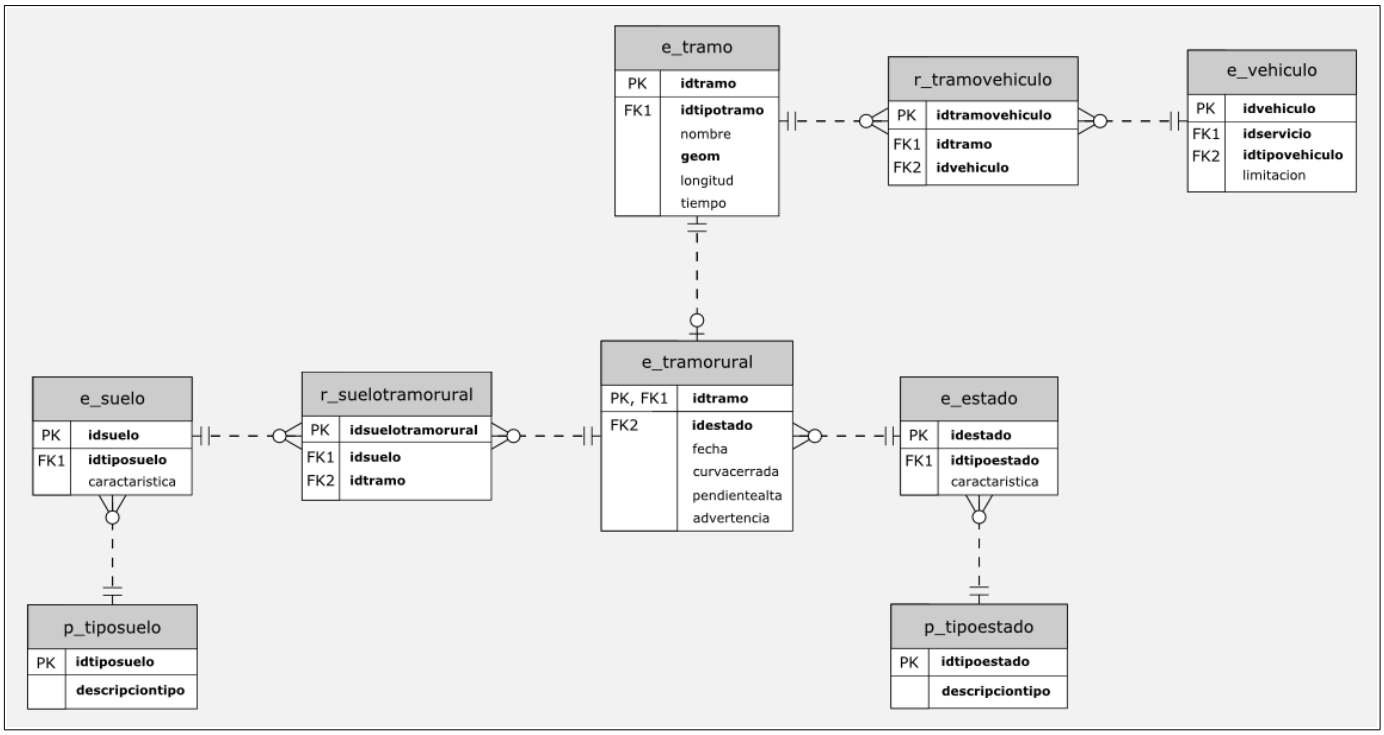

Imagen 90: Modelo relacional donde se aprecia la información sobre tramos de rutas

En este caso, las relaciones entre tablas son de tipo N:M, N:1 y 1:1.

- La relación de tipo N:M entre la entidad Tramo y la entidad Vehiculo se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_tramovehiculo y e_tramo; y otra de tipo (0,N) y (1,1) entre la tabla r_tramovehiculo y e_vehiculo.
- La relación de tipo N:M entre la entidad Tramorural y la entidad Suelo se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_suelotramorural y e_tramorural; y otra de tipo (0,N) y (1,1) entre la tabla r_ suelotramorural y e_suelo.
- La relación de tipo N:1 entre la entidad Tramorural y la entidad Estado se representa realizando una unión de tipo (0,N) y (1,1) entre la tabla e_tramorural y e_estado.
- Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a las relaciones existente entre la tabla e_estado y la tabla p_tipoestado y entre la tabla e_suelo y la tabla p_tiposuelo. En este caso se representará realizando una unión de tipo (0,N) y (1,1).
- Las relaciones de tipo 1:1 referidas a la relación de subclase de la entidad Tramo y entidad Tramorural. En este caso se representará utilizando una unión de tipo (1,1) y (0,1).

3. Modelo físico

En este apartado se explicará el modelo físico creado en PostgreSQL para implementar el modelo relacional diseñado. Para ello, se ha creado un diagrama del modelo físico utilizando un software *SQL Power Architect*.

Esta herramienta realiza el modelado de datos automáticamente a partir de la conexión de una base de datos. Por lo tanto, esta herramienta utilizará la jerarquía existente en nuestra base de datos para crear un diagrama de la estructura de datos.

Si se observa el diagrama del modelo físico creado por esta herramienta se podrá comentar varios aspectos del diseño:

- Las tablas. Conformarán la unidad de almacenamiento principal de la base de datos. En el diagrama se reflejarán las tablas como las relaciones que existen entre ellas. Serán las mismas que en el modelo relacional.
- Las columnas de las tablas. Representarán los campos de cada uno de los registros almacenados. En este diagrama se añadirá el dominio al nombre de cada columna.
- La clave primaria de la tabla. Representará la columna (o las columnas) que desempeñará la clave candidata para identificar las tuplas de una tabla. En este diagrama, la columna (o las columnas) que representa la clave primaria estará diferenciada de las demás columnas, y contendrán la palabra “PK” entre corchetes. Pero si estas columnas también representen claves foráneas (a la vez que claves primarias), en este caso contendrán la palabra “PFK” entre corchetes.
- Las claves foráneas de la tabla. Representa la clave primaria de la tabla que está relacionada. Es decir, se utilizará la clave foránea para expresar la interrelación existente entre dos tablas. En este diagrama las columnas que son claves foráneas contendrán la palabra “FK” entre corchetes. Pero si estas columnas también representen claves primarias (a la vez que claves foráneas) contendrán la palabra “PFK” entre corchetes.
- Las restricciones de las columnas. Representan restricciones referidas a los registros almacenados en una columna. En este diagrama solamente se representarán las restricciones referidas a los dominios que contienen las columnas, más concretamente a las restricciones de existencia. En estos casos, se reflejará las columnas que no contienen valores nulos, y para ello se incluirá la palabra “NOT NULL” al dominio de la columna. Esta herramienta añade por defecto esta restricción a todas las columnas que representan claves primarias y claves foráneas. Por lo tanto, habrá que modificar los casos donde no contengan esta restricción.

A continuación, se explicará el diagrama obtenido con esta herramienta. Para ello, se utilizarán varias imágenes de las partes representativas de este diagrama.

- Como se aprecia en la Imagen 91, existen siete tablas (relacionadas entre si) que representan la información referida a los diseminados: por un lado están las tablas e_destino, p_tipodestino, p_direccion, p_riesgopotencial y p_contacto; y por otro lado las tablas e_zona y p_tipozona.

  En estas tablas, las columnas que no contienen valores nulos son las siguientes:

  1. Las columnas que representan claves primarias (PK): iddestino, idtipodestino, idzona y idtipozona.
  2. Las columnas que representan claves foráneas (FK): iddestino, idtipodestino, idzona y idtipozona.

3. Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tipodestino y p_tipozona) y geom (de la tabla e_destino).
4. En este caso, las relaciones entre tablas son de tipo N:1 y 1:1.
5. La relación reflexiva de tipo N:1 de la entidad Zona se representa realizando una unión de tipo (N,0) y (0,1) en la misma tabla e_zona.
6. Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a la relación existente entre la tabla e_destino y la tabla p_tipodestino, y la relación entre la tabla e_zona y la tabla p_tipozona. En estos dos casos se representará realizando una unión de tipo (0,N) y (1,1).
7. Las relaciones de tipo 1:1 referidas a la reestructuración de la información del diseminado. En este caso se referirá a las relaciones existentes entre la tabla e_destino y la tabla p_dirección, entre la tabla e_destino y la tabla p_riesgopotencial, y entre la tabla e_destino y la tabla p_contacto. En estos tres casos se representará realizando una unión de tipo (1,1) y (0,1).

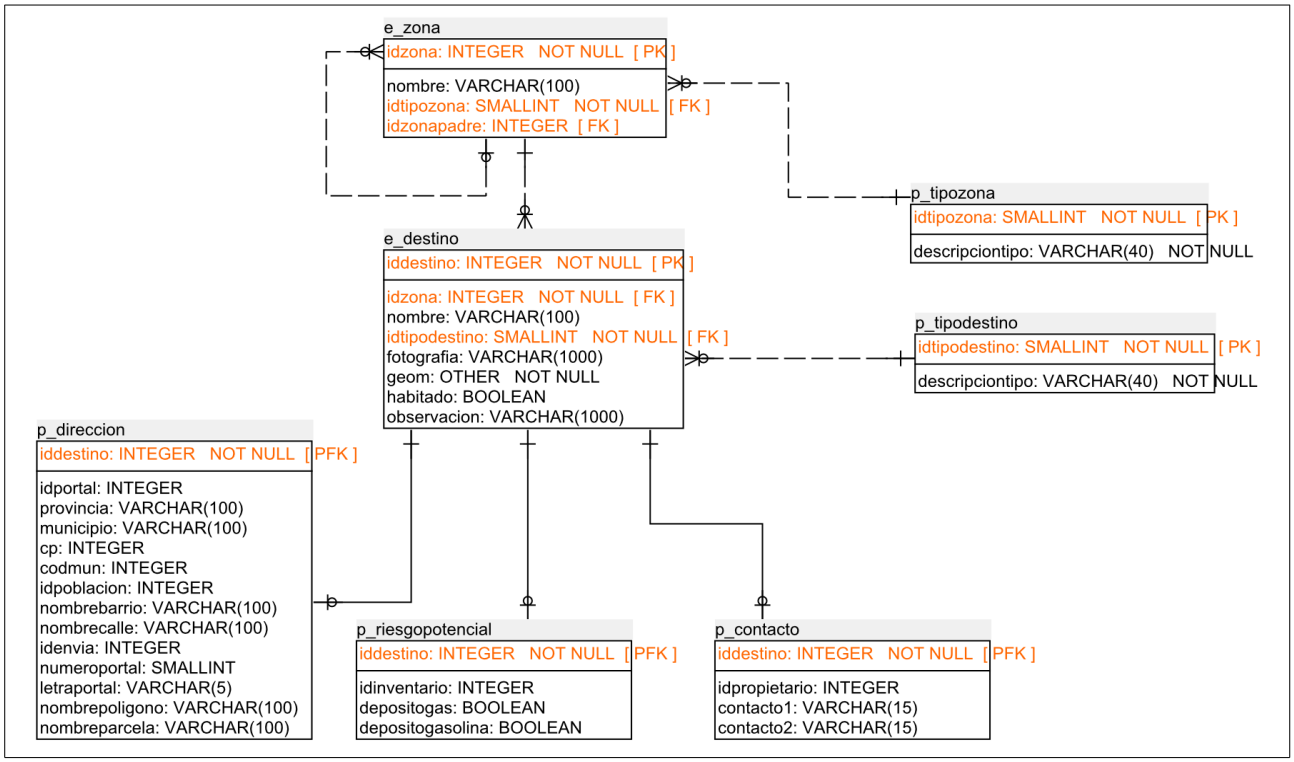

Imagen 91: Tablas y relaciones creadas entorno a Destino y Zona

- Como se aprecia en la Imagen 92, existen ocho tablas (relacionadas entre si) que representan la información referida a los diseminados: por un lado están las tablas e_destino, p_tipodestino, p_direccion, p_riesgopotencial y p_contacto; por otro lado la tabla e_habitante; y por último las referidas a las relaciones entre tablas r_destinovecindad y r_habitantedestino.

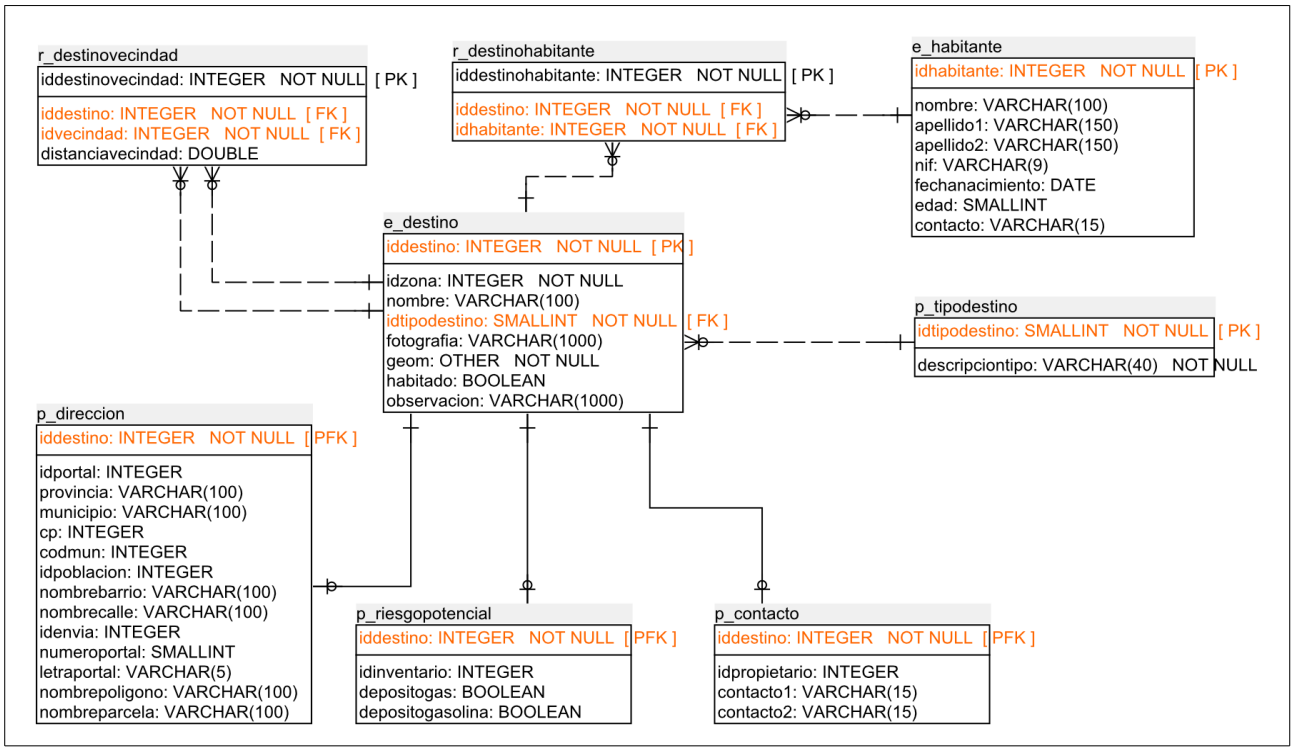

Imagen 92: Tablas y relaciones creadas entorno a Destino y Habitante

En estas tablas, las columnas que no contienen valores nulos son las siguientes:

1. Las columnas que representan claves primarias (PK): iddestino, idtipodestino, idhabitante, iddestinovecindad y iddestinohabitante.
2. Las columnas que representan claves foráneas (FK): iddestino, idvecindad, idtipodestino, idzona y idhabitante.
3. Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tipodestino) y geom (de la tabla e_destino).

En este caso, las relaciones entre tablas son de tipo N:M, N:1 y 1:1.

1. La relación reflexiva de tipo N:M de la entidad Destino se representa realizando dos uniones de tipo (N,0) y (1,1) entre la tabla r_destinovecindad y e_destino.
2. La relación de tipo 1:N entre la entidad Destino y la entidad Habitante se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_destinohabitante y e_habitante; y otra de tipo (0,N) y (1,1) entre la tabla r_destinohabitante y e_destino.
3. Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a la relación existente entre la tabla e_destino y la tabla p_tipodestino. En este caso se representará realizando una unión de tipo (0,N) y (1,1).
4. Las relaciones de tipo 1:1 referidas a la reestructuración de la información del diseminado. En este caso se referirá a las relaciones existentes entre la tabla e_destino y la tabla p_dirección, entre la tabla e_destino y la tabla p_riesgopotencial, y entre la tabla e_destino y la tabla p_contacto. En estos tres casos se representará realizando una unión de tipo (1,1) y (0,1).

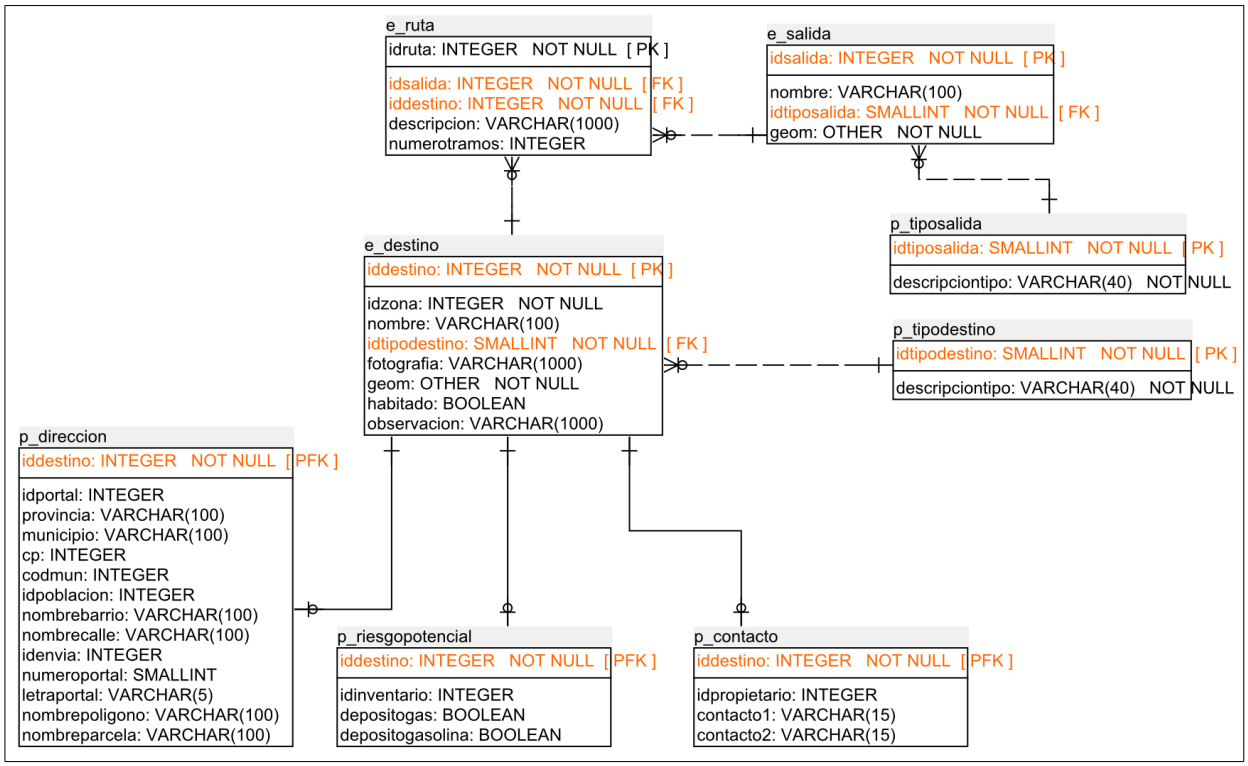

Imagen 93: Tablas y relaciones creadas entorno a Destino, Salida y Ruta

- Como se aprecia en la Imagen 93, existen ocho tablas (relacionadas entre si) que representan la información referida a los diseminados: por un lado están las tablas e_destino, p_tipodestino, p_direccion, p_riesgopotencial y p_contacto; por otro lado la tabla e_ruta; y por último las *e_salida* y *p_tiposalida*; también está la tabla *e_destino* (antes mencionada).

  En estas tablas, las columnas que no contienen valores nulos son las siguientes:

  1. Las columnas que representan claves primarias (PK): idruta, idsalida, idtiposalida, iddestino y idtipodestino.
  2. Las columnas que representan claves foráneas (FK): idsalida, idtiposalida, iddestino y idtipodestino.
  3. Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tiposalida y p_tipodestino) y geom (de la tabla e_salida y e_destino).

  En este caso, las relaciones entre tablas son de tipo N:M y N:1.

  1. La relación de tipo N:1 entre la entidad Ruta y la entidad Salida se representa realizando una unión de tipo (0,N) y (1,1) entre la tabla e_ruta y e_salida.
  2. La relación de tipo N:1 entre la entidad Ruta y la entidad Destino se representa realizando una unión de tipo (0,N) y (1,1) entre la tabla e_ruta y e_destino.
  3. Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a la relación existente entre la tabla e_salida y la tabla p_tiposalida, y la relación entre la tabla e_destino y la tabla p_tipodestino . En estos dos casos se representará realizando una unión de tipo (0,N) y (1,1).

- Como se aprecia en la Imagen 94 existen siete tablas (relacionadas entre si) que representan la información referida a las rutas de acceso: por un lado están las tablas *e_salida* y *p_tiposalida*; también están las tablas *e_servicio* y *p_tiposervicio*; las tablas *e_vehiculo* y *p_tipovehiculo*; y por último existen las referidas a las relaciones entre tablas en este caso la tabla *r_salidaservicio*.

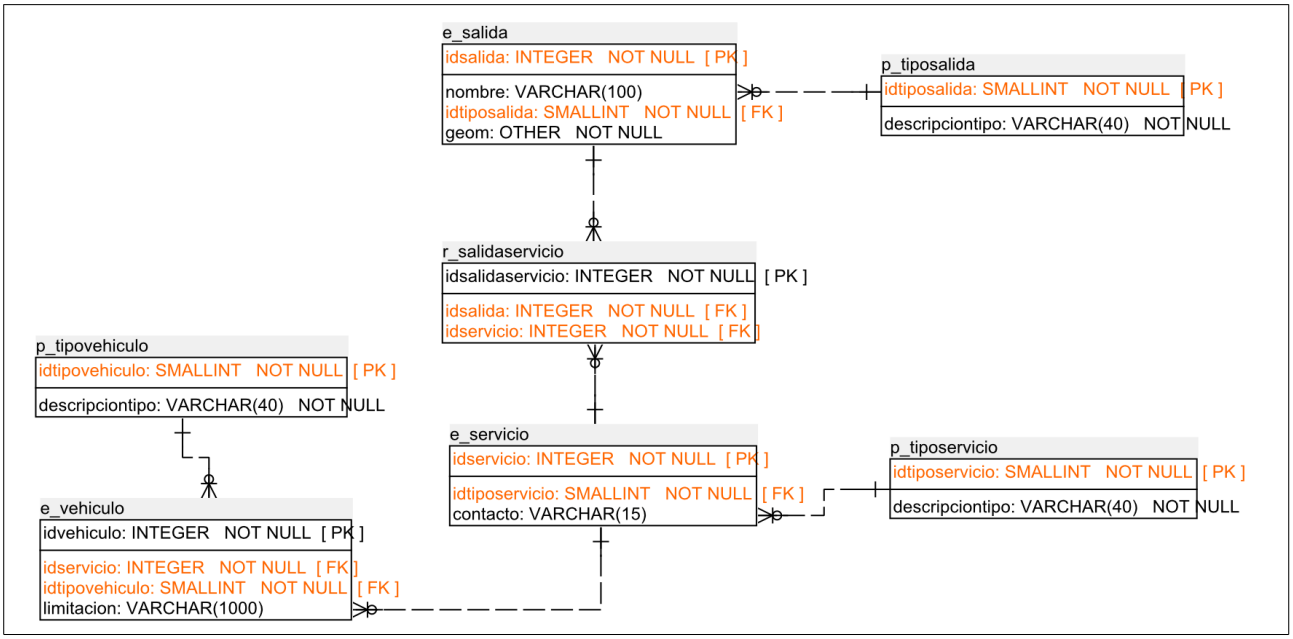

Imagen 94: Tablas y relaciones creadas entorno a Salida, Servicio y Vehiculo

En estas tablas, las columnas que no contienen valores nulos son las siguientes:

1. Las columnas que representan claves primarias (PK): idsalida, idtiposalida, idservicio, idtiposervicio, idvehiculo, idtipovehiculo y idsalidaservicio.
2. Las columnas que representan claves foráneas (FK): idsalida, idtiposalida, idservicio, idtiposervicio y idtipovehiculo.
3. Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tiposalida, p_tiposervicio, y p_tipovehiculo) y geom (de la tabla e_salida).

En este caso, las relaciones entre tablas son de tipo N:M y N:1.

1. La relación de tipo N:M entre la entidad Salida y la entidad Servicio se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_salidaservicio y e_salida; y otra de tipo (0,N) y (1,1) entre la tabla r_salidaservicio y e_servicio.
2. La relación de tipo N:1 entre la entidad Vehiculo y la entidad Servicio se representa realizando una unión de tipo (0,N) y (1,1) entre la tabla e_vehiculo y e_servicio.
3. Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a la relación existente entre la tabla e_salida y la tabla p_tiposalida, e_servicio y la tabla p_tiposervicio, y la relación entre la tabla e_vehiculo y la tabla p_tipovehiculo. En estos tres casos se representará realizando una unión de tipo (0,N) y (1,1).

- Como se aprecia en la Imagen 95 existen siete tablas (relacionadas entre si) que representan también la información referida rutas de acceso: por un lado están las tablas *e_tramo* y *p_tipotramo*; por otro lado las tablas *e_servicio* y *p_tiposervicio*; las tablas *e_vehiculo* y *p_tipovehiculo*; y por último existen las referidas a las relaciones entre tablas *r_tramovehiculo*.

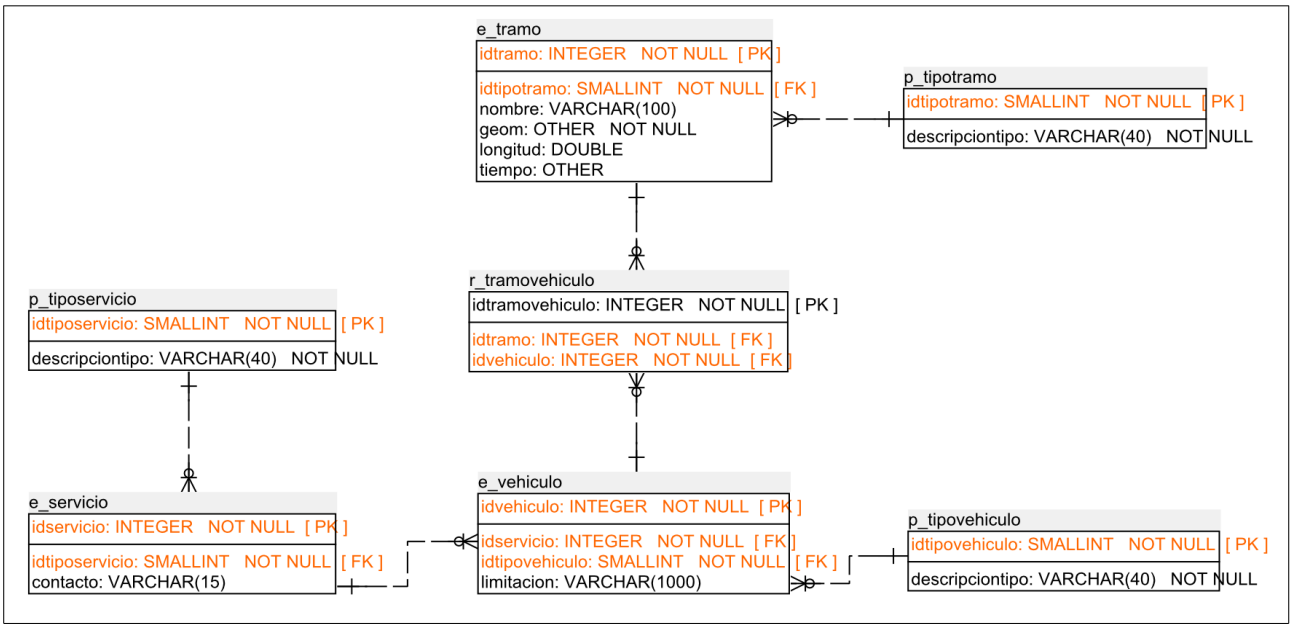

Imagen 95: Tablas y relaciones creadas entorno a Servicio, Vehiculo y Tramo

En estas tablas, las columnas que no contienen valores nulos son las siguientes:

1. Las columnas que representan claves primarias (PK): idtramo, idtipotramo, idservicio, idtiposervicio, idvehiculo, idtipovehiculo y idtramovehiculo.
2. Las columnas que representan claves foráneas (FK): idtramo, idtipotramo, idservicio, idtiposervicio, idvehiculo y idtipovehiculo.
3. Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tipotramo, p_tiposervicio, y p_tipovehiculo) y geom (de la tabla e_tramo).

En este caso, las relaciones entre tablas son de tipo N:M y N:1.

1. La relación de tipo N:M entre la entidad Tramo y la entidad Vehiculo se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_tramovehiculo y e_tramo; y otra de tipo (0,N) y (1,1) entre la tabla r_tramovehiculo y e_vehiculo.
2. La relación de tipo N:1 entre la entidad Vehiculo y la entidad Servicio se representa realizando una unión de tipo (0,N) y (1,1) entre la tabla e_vehiculo y e_servicio.
3. Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a la relación existente entre la tabla e_tramo y la tabla p_tipotramo, e_servicio y la tabla p_tiposervicio, y la relación entre la tabla e_vehiculo y la tabla p_tipovehiculo. En estos tres casos se representará realizando una unión de tipo (0,N) y (1,1).

- Como se aprecia en la Imagen 96 existen ocho tablas (relacionadas entre si) que representan la información referida a los tramos de rutas: por un lado están las tablas *e_tramo* y *p_tipotramo*; por otro lado la tabla *e_tramorural*; las tablas *e_estado* y *p_tipoestado*; las tablas *e_suelo* y *p_tiposuelo*; y por último existen las referidas a las relaciones entre tablas *r_suelotramorural*.

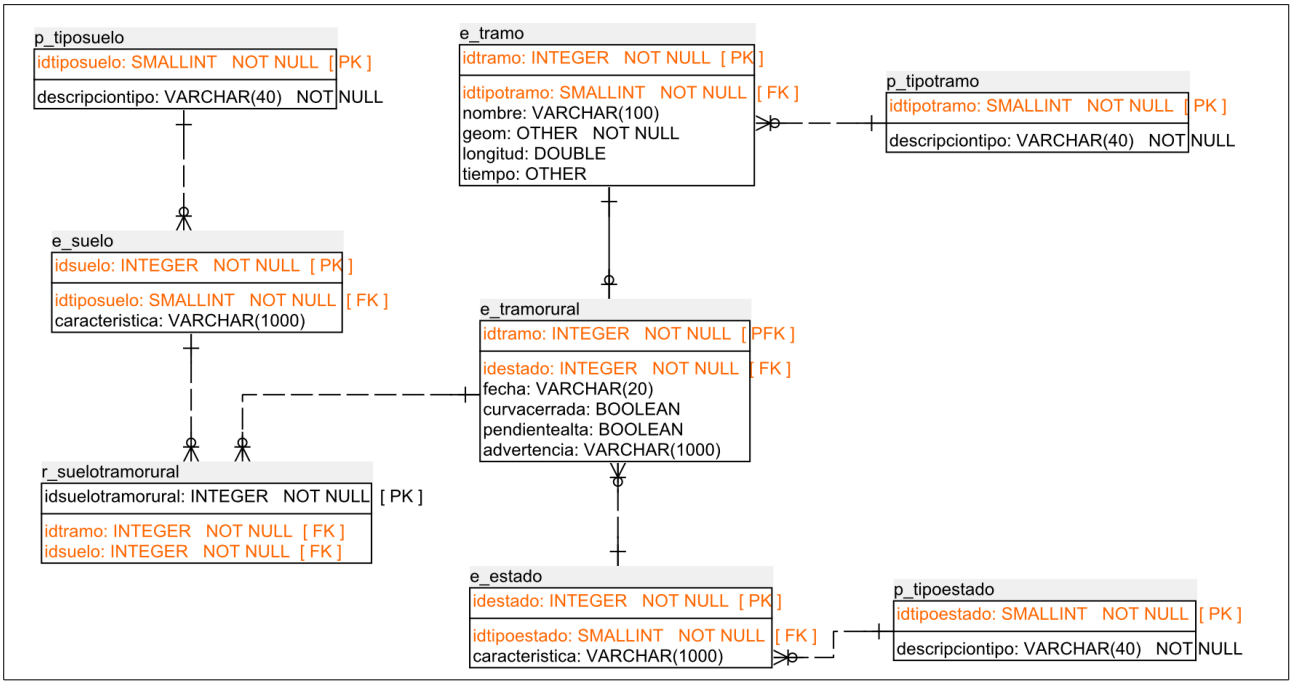

Imagen 96: Tablas y relaciones creadas entorno a Tramo, Tramorural, Estado y Suelo

En estas tablas, las columnas que no contienen valores nulos son las siguientes:

1. Las columnas que representan claves primarias (PK): idtramo, idtipotramo, idestado, idtipoestado, idsuelo, idtiposuelo y idsuelotramorural.
2. Las columnas que representan claves foráneas (FK): idtramo, idtipotramo, idestado, idtipoestado, idsuelo y idtiposuelo.
3. Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tipotramo, p_tipoestado y p_tiposuelo) y geom (de la tabla e_tramo).

En este caso, las relaciones entre tablas son de tipo N:M, N:1 y 1:1.

1. La relación de tipo N:M entre la entidad Tramorural y la entidad Suelo se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_suelotramorural y e_tramorural; y otra de tipo (0,N) y (1,1) entre la tabla r_ suelotramorural y e_suelo.
2. La relación de tipo N:1 entre la entidad Tramorural y la entidad Estado se representa realizando una unión de tipo (0,N) y (1,1) entre la tabla e_tramorural y e_estado.
3. Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a las relaciones existente entre la tabla e_tramo y la tabla p_tipotramo , entre la tabla e_estado y la tabla p_tipoestado y entre la tabla e_suelo y la tabla p_tiposuelo. En este caso se representará realizando una unión de tipo (0,N) y (1,1).

4. Las relaciones de tipo 1:1 referidas a la relación de subclase de la entidad Tramo y entidad Tramorural. En este caso se representará utilizando una unión de tipo (1,1) y (0,1).

- Como se aprecia en la Imagen 97 existen ocho tablas (relacionadas entre si) que representan también la información referida rutas de acceso: por un lado está la tabla *e_ruta*; por otro lado las tablas *e_tramo* y *p_tipotramo*; también está la tabla *e_puntoinfo* y la tabla *e_cruce*; y por último existen las referidas a las relaciones entre tablas *r_rutatramo*, *r_puntoinforuta* y *r_cruceruta*.

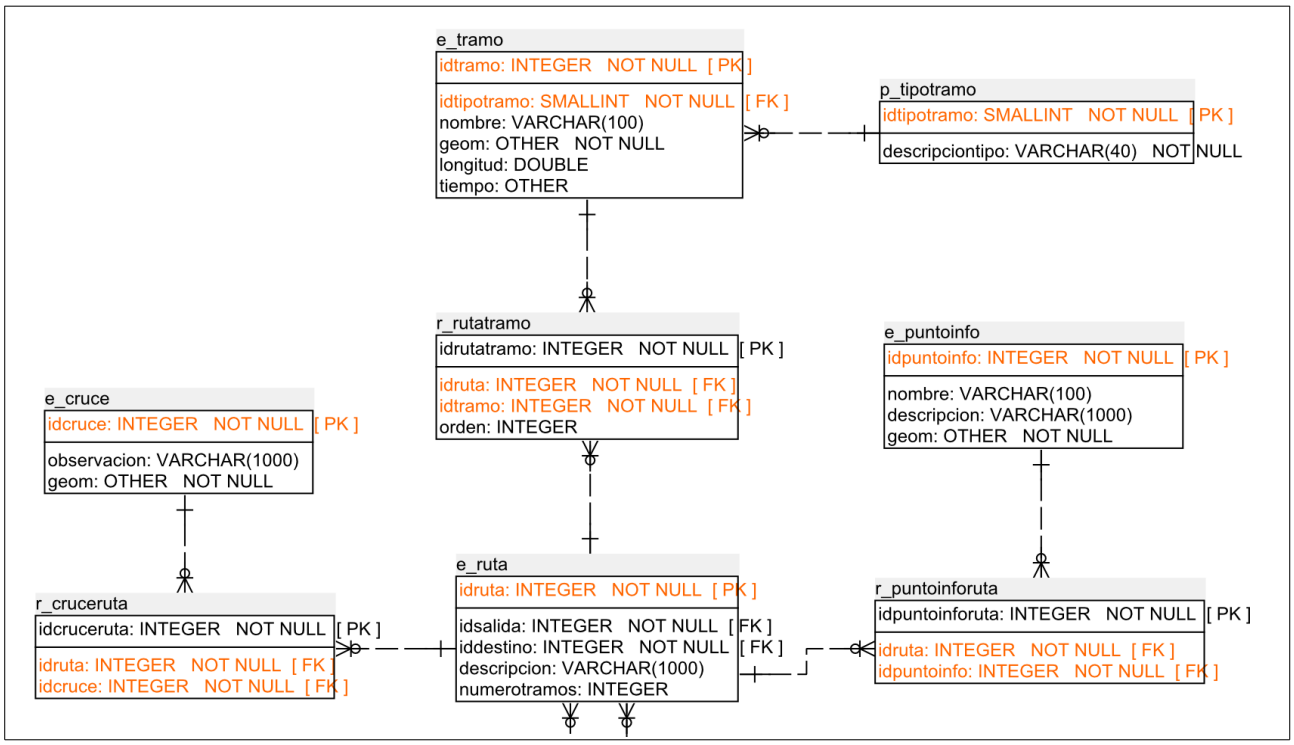

Imagen 97: Tablas y relaciones creadas entorno a Tramo, Ruta, Cruce y Puntoinfo.

En estas tablas, las columnas que no contienen valores nulos son las siguientes:

1. Las columnas que representan claves primarias (PK): idruta, idtramo, idtipotramo, idpuntoinfo, idcruce, idrutatramo, idpuntoinforuta y idcruceruta.
2. Las columnas que representan claves foráneas (FK): idruta, idtramo, idtipotramo, idpuntoinfo y idcruce. Las columnas idsalida y iddestino también son claves foráneas en esta imagen no se han incluido las tablas e_destino y e_salida.
3. Las columnas que no son claves primarias ni claves foráneas: descripciontipo (de la tabla p_tipotramo) y geom (de la tabla e_tramo, e_puntoinfo y e_cruce).

En este caso, las relaciones entre tablas son de tipo N:M y N:1.

1. La relación de tipo N:M entre la entidad Ruta y la entidad Tramo se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_rutatramo y e_ruta; y otra de tipo (0,N) y (1,1) entre la tabla r_rutatramo y e_tramo.

2. La relación de tipo N:M entre la entidad Ruta y la entidad Punto Información se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_puntoinforuta y e_ruta; y otra de tipo (0,N) y (1,1) entre la tabla r_puntoinforuta y e_puntoinfo.
3. La relación de tipo N:1 entre la entidad Ruta y la entidad Cruce se representa realizando dos uniones: una de tipo (0,N) y (1,1) entre la tabla r_cruceruta y e_ruta; y otra de tipo (0,N) y (1,1) entre la tabla r_cruceruta y e_cruce.
4. Las relaciones de tipo N:1 referidas a la información del tipo de una entidad. En este caso se referirá a la relación existente entre la tabla e_tramo y la tabla p_tipotramo. En este caso se representará realizando una unión de tipo (0,N) y (1,1).

ANEXO 3: Desarrollo del sistema

Índice del Anexo 3 – Desarrollo del sistema:

A3.1 Proceso de carga de datos

En este apartado del proyecto se definirá la metodología utilizada para obtener la información contenida en el documento KML para posteriormente añadirla a las tablas de la BD (creadas anteriormente). Para realizar esta fase de carga se utilizará el lenguaje de programación Python. Más concretamente se utilizará la versión Python 3.

Por lo tanto, en este apartado se procederá a explicar el código fuente (en el lenguaje programación Python) utilizado para realizar esta carga de la geodatabase. Para obtener este código fuente se realizarán procesos de diseño, codificación, depuración y mantenimiento.

1. Estudio de las librerías de Python

Antes de empezar a programar en el lenguaje de programación Python se ha realizado un breve estudio sobre qué librerías existentes de Python podrían servir para acceder al contenido de un archivo de tipo KML.

Para realizar dicho estudio se han analizado diferentes artículos encontrados en internet: artículos de blogs de programadores, de portales digitales especializados en Python, y de foros asociados a la página oficial de python.

La primera búsqueda se centró en librerías creadas para el acceso del contenido de archivos KML:

- Librerías KML. Existen varios módulos que permiten procesar archivos KML (leer, escribir y manipular). Entre otras se analizaron las librerías Fastkml, pyKML, simplekml y Keytree.

Para obtener cada una de estas librerías se accedió a las páginas web oficiales de estas librerías. En ellas se consultó la información existente de las últimas versiones, y se analizaron los ejemplos de script de codificación.

Pero al utilizar estas librerías en un software intérprete de Python (en nuestro caso PyCharm) se produjeron errores que impedían su uso. Esto se debía a la versión Python utilizada en nuestro ordenador. Es decir, todos estos módulos se desarrollaron para la versión Python 2, y todavía no se han actualizado a la versión Python 3.

Por lo tanto, se procedió a realizar una segunda búsqueda de librerías. En este caso, se centró la búsqueda en librerías creadas para el acceso del contenido de archivos XML:

- Librerías XML. Existen varios módulos que permiten procesar archivos XML (leer, escribir y manipular). Entre otros existen: xml.etree.ElementTree, lxml, xml.dom, xml.dom.minidom, xml.dom.pulldom, xml.sax y xml.parsers.expat.

Al utilizar varias de ellas en un software intérprete de Python, se decantó por la librería xml.dom.minidom. Por lo tanto, esta librería facilitará la obtención de los datos del contenido del documento KML, para posteriormente almacenarla en la geodatabase. Es decir, esta librería será apta para realizar la carga de datos de tipo Rolling.

1.1. Métodos de acceso a documentos XML

La librería minidom facilitará la obtención de datos en un archivo con estructura XML, utilizando para ello una metodología basada en el acceso a atributos en árbol DOM.

El método DOM crea una representación tipo árbol del documento XML. Por lo tanto, al utilizar este método se lee de golpe todo el contenido de dicho archivo, para posteriormente acceder a la información deseada.

Existe otro tipo de metodología de acceso a documentos XML llamado SAX. Pero en este método se accede a la información por eventos, es decir, se accede a la información mientras se lee el documento.

1.1.1. Metodología de tipo DOM

Este método realiza una representación con una organización de árbol estándar de la estructura de datos de un archivo XML. El resultado de esta representación será un objeto DOM donde se podrá acceder a las partes del contenido XML utilizando propiedades y métodos propios de la metodología. Este método es útil para aplicaciones que analizan el documento XML de forma aleatoria.

1.1.2. Metodología de tipo SAX

Este método facilita el acceso a un elemento SAX para procesar la información que contiene en ella, y no se puede acceder varios elementos SAX a la vez. Por lo tanto, si un elemento SAX contiene un elemento no se podrá acceder a la vez al elemento y al sub-elemento.

1.2. Librería minidom

La librería minidom facilitará el acceso a la información existente en un mismo nivel de granularidad o jerarquía en el archivo KML. Es decir, al utilizar esta librería se podrá obtener los datos almacenados en una misma agrupación o tipo de contenido del archivo KML. Este acceso de datos se realizará utilizando el método de acceso en árbol DOM. Por lo tanto, al utilizar este método, la información se reestructurará para facilitar el acceso a ella.

Con esta reestructuración se creará un objeto DOM donde los datos se organizarán en nodos y nodo hijos (almacenando en estos nodos el nombre de la etiqueta, sus atributos y su contenido o valor).

Como se observa en la Imagen 98 los elementos del archivo KML se representan en etiquetas que contienen atributos (características o propiedades de los elementos XML). Esta estructura de datos XML se convertirá en una estructura en árbol DOM.

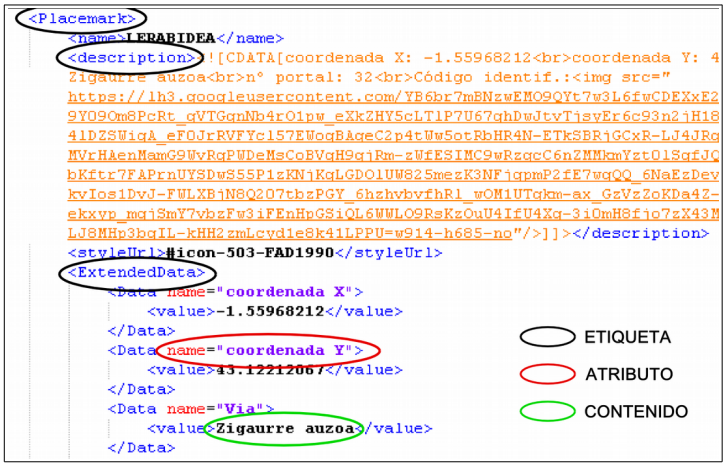

Imagen 98: Nodos, atributos y elementos del árbol DOM

Para crear el objeto DOM y para crear los nodos hijos de este árbol se procederá de la siguiente manera:

1. Función parse. Esta función creará un objeto de árbol DOM. Al utilizar esta función se creará una variable de tipo objeto DOM donde se almacenará todo el contenido del documento KML. Por ejemplo:

```
from xml.dom import minidom
domKml = minidom.parse("C:\\Documents and Settings\\Escritorio\\document.kml")
```

2. Función documentElement. Al utilizar esta función se obtendrá el nodo raíz. Por ejemplo:

```
nodo = domKml.documentElement
```

3. Función childNodes. Esta función se utilizará para navegar a través de la estructura del objeto DOM. El resultado de la utilización de esta función será la creación de una lista de nodos hijos de la raíz del árbol. Por ejemplo: `nodohijos = nodo.childNodes`

Para realizar el acceso de datos (después de esta reestructuración) se podrá realizar de varias formas:

1. Mediante referencia directa. Al crear la lista de nodos hijos se podrá acceder a ellas directamente. Por ejemplo: `nodohijos [número indice]`
2. Buscando el nombre del nodo. Habrá la opción de buscar nodos a partir del nombre de la etiqueta. En nuestro caso, se han creado funciones para realizar esta búsqueda. Por ejemplo:

```
documentKML = buscarSubElemento(docKml, 'Document')
folders = buscarSubElementosTipo(documentKML, 'Folder')
```

3. Navegando por el árbol DOM. Habrá la posibilidad de acceder a nodos hijos creando funciones recursivas que naveguen por el árbol DOM.

2. Metodología de carga utilizando Python

En este apartado se explicará el conjunto de instrucciones creados en Python para realizar la carga de datos. Para facilitar la comprensión de la programación realizada se utilizarán diagramas de flujo. Es decir, se utilizarán partes del ordinograma para explicar diferentes etapas del script generado.

Como se puede observar en el ordinograma de la Imagen 99 se obtienen los datos de entrada a partir de una base de datos o un fichero externo. Dicha información se almacenará en una variable de tipo objeto DOM, utilizando la función *minidom.parse*. A continuación, a partir de esta variable se obtendrá un nodo raíz (donde surgirán los nodos hijos sucesivamente). Para ello, se utilizará la función *documentElement*.

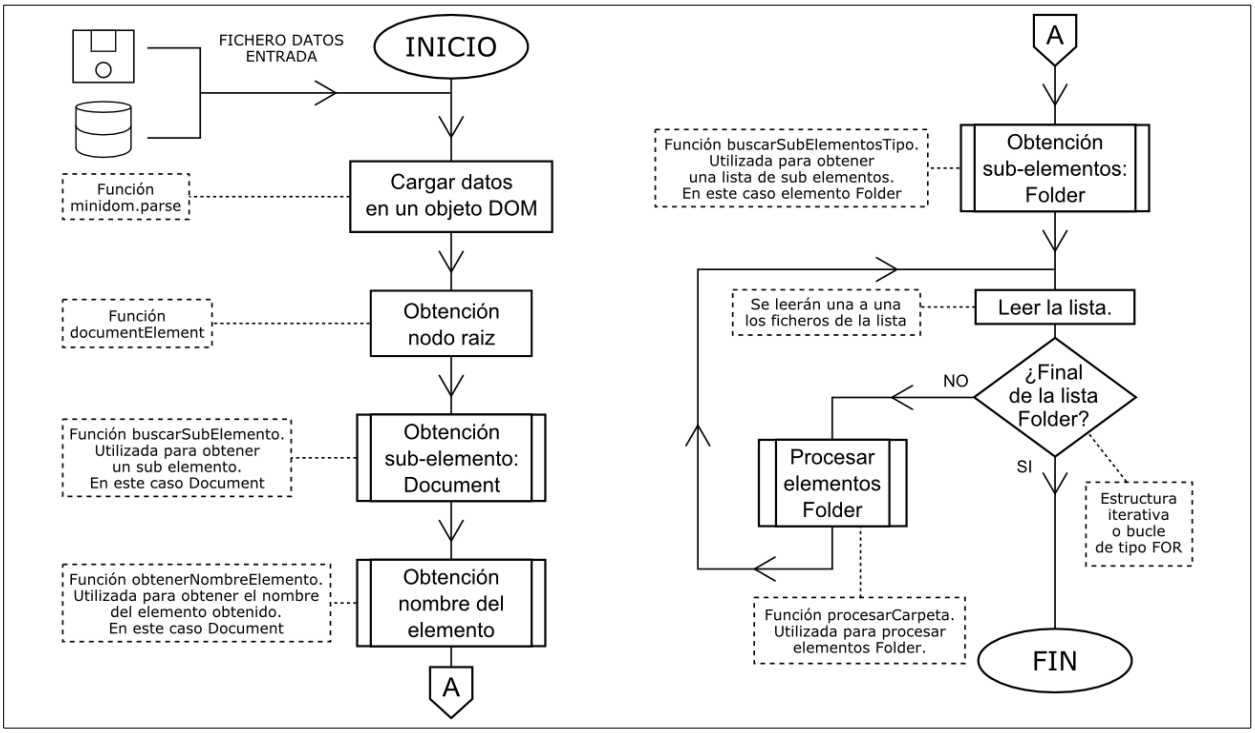

Imagen 99: Ordinograma del procesamiento de carpetas del documento KML

Para identificar los nodos hijos que nos interesará procesar, se analizará la jerarquía o organización del documento KML. Para ello, se visualizará su contenido con un editor de texto, y de esta manera se identificará uno a uno los nodos hijos que contengan la información que se desea cargar. Por lo tanto, el nodo hijo (del nodo raíz) que nos interesa será el nodo que representa al elemento Document. Es decir, el primer paso a realizar será el de acceder a este nodo Document. Para ello se utilizará la función *buscarSubElemento* (ver el apartado 2.1.).

Como se aprecia en la imagen, el siguiente paso será el de obtener el nombre del nodo obtenido. En los casos de los nodos que representen los elementos XML de tipo Document, Folder y Placemark se podrá conseguir el nombre al contener todos éstos un nodo hijo de tipo Name (que contienen el texto del nombre). Para ello se utilizará la función *obtenerNombreElemento* (ver el apartado 2.3.).

Después de obtener dicho nombre se analizará el contenido del nodo Document. Como se aprecia, dentro de este nodo contendrá nodos hijos de diferente tipo: de tipo Name, Schema,

Style, StyleMap y Folder. En nuestro caso, la información que nos interesa está en los nodos hijos de tipo Folder. Por lo tanto, se procederá a procesar dichos nodos. Para esta labor primero se obtendrá una lista de elementos Folder utilizando la función *buscarSubElementosTipo* (ver el apartado 2.2.), y posteriormente se procederá a procesar cada uno de estos elementos. Para procesar esta lista de elementos Folder se utilizará la función *procesarCarpeta* de forma iterativa (ver el apartado 2.5.).

2.1. Función buscarSubElemento

Como se aprecia en la Imagen 100, la función *buscarSubElemento* contendrá dos parámetros de entrada: un elemento DOM y un tipo de elemento. El primer parámetro será el nodo (el elemento DOM) donde se realizará la búsqueda entre los nodos hijos que contiene. Esta búsqueda lo determinará el segundo parámetro, el tipo de elemento. Es decir, este parámetro definirá el tipo de nodo hijo que se pretende encontrar.

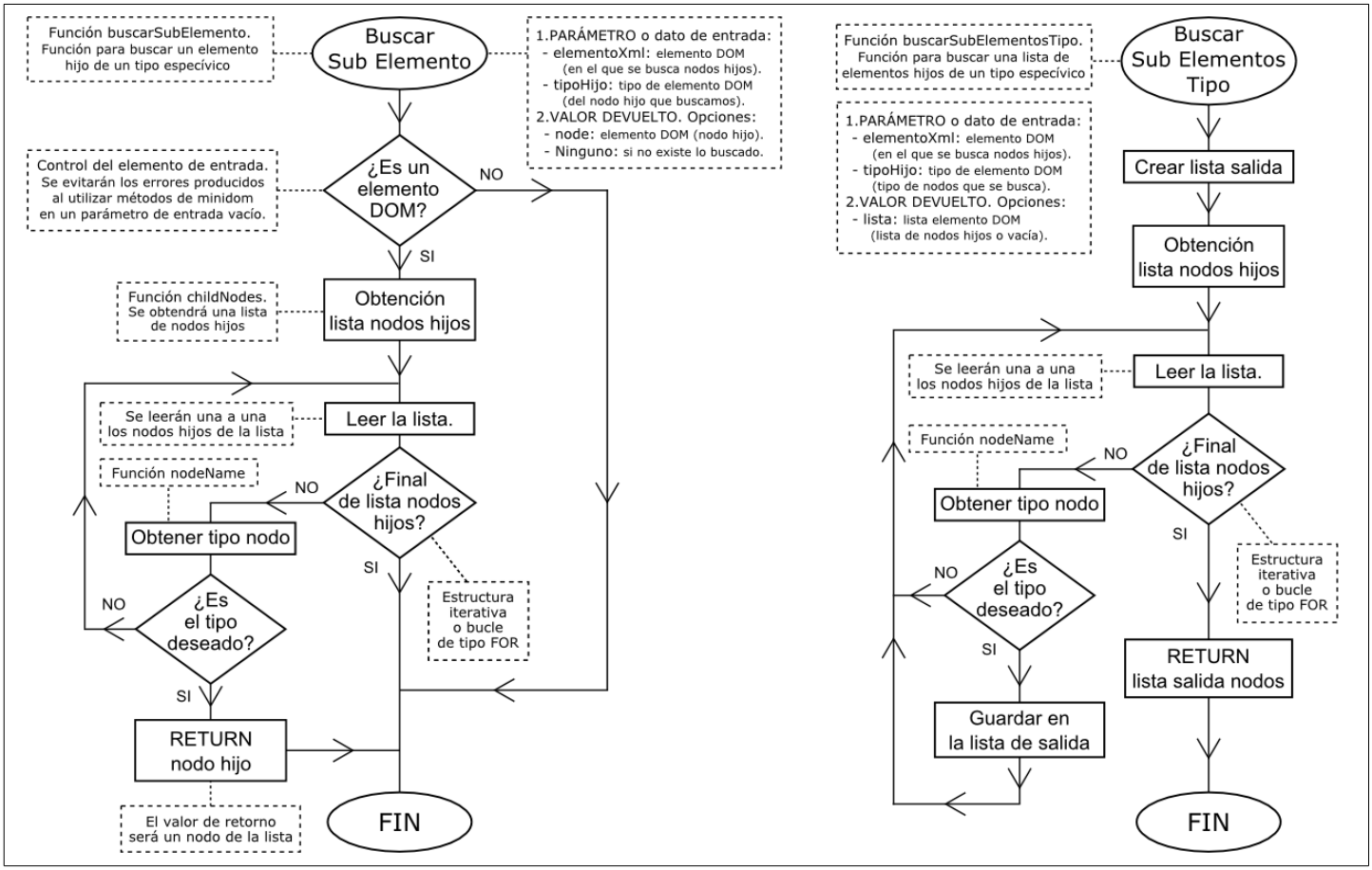

Imagen 100: Ordinograma de la función de Búsqueda de sub-elementos y de sub-elementos tipo

Si se analiza este diagrama de flujo se observa que esta función contiene los siguientes procesos:

1. Control de los datos de entrada. Es decir, se comprobará si es correcto el primer parámetro de entrada.
2. Obtención de los nodos hijos del elemento DOM de entrada. El resultado será una lista de elementos.

3. Análisis del tipo de elemento que contiene la lista. Para ello, se realizará un búsqueda iterativa. Cada elemento será analizado, si su tipo coincide con el tipo de elemento buscado. Si es así se acabará la búsqueda.

Al utilizar esta función si se encuentra el nodo hijo que cumpla la condición preestablecida se devolverá un parámetro de salida. En este caso este parámetro será el nodo hijo coincidente, un elemento DOM.

2.2. Función buscarSubElementosTipo

Como se aprecia en la Imagen 100, la función *buscarSubElementosTipo* es muy similar a la función anterior. Contiene los mismos parámetros de entrada, pero busca uno o varios elementos DOM que contengan un tipo de elemento determinado. Por lo tanto, al realizar la búsqueda los elementos coincidentes se guardarán en una lista, y al finalizar las instrucciones de esta función se devolverá (como parámetro de salida) esta lista de nodos coincidentes.

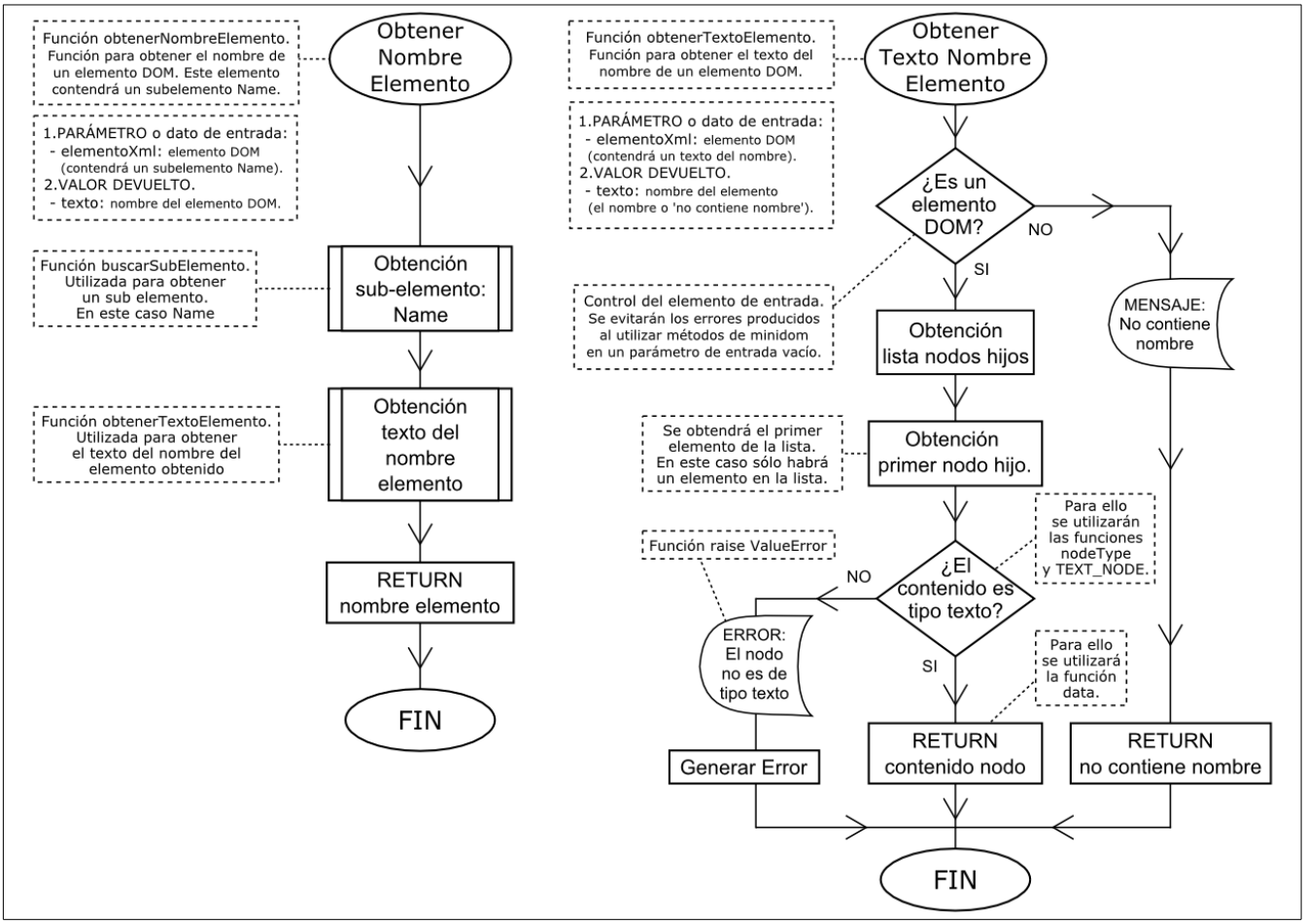

Imagen 101: Ordinograma de la función de Obtención del nombre del elemento

2.3. Función obtenerNombreElemento

Como se aprecia en la Imagen 101, la función *obtenerNombreElemento* contendrá un parámetro de entrada: un elemento DOM. Este elemento será un nodo donde contendrá varios

nodos hijos, entre ellos uno de tipo Name. Si no es así, no se podrá obtener el nombre del elemento.

Como se aprecia en este diagrama de flujo, al utilizar esta función en realidad se ejecutan dos funciones sucesivas: la función *buscarSubElemento* (explicada en el apartado 2.1.) y la función *obtenerTextoElemento* (ver el apartado 2.4.). La primera función se utilizará para obtener un nodo hijo de tipo Name (del elemento de entrada), y la segunda función se utilizará para obtener el nombre de este elemento.

El parámetro de salida de esta función será el propio nombre del elemento.

2.4. Función obtenerTextoElemento

Como se aprecia en la Imagen 101, la función *obtenerTextoElemento* contendrá un parámetro de entrada: un elemento DOM. Este elemento será un nodo, un elemento DOM de tipo Name. Si no es así, no se podrá obtener el texto del nombre del elemento.

Como se aprecia en este diagrama de flujo, el primer proceso que se realizará es la función *ChildNodes*. Esta función creará una lista de nodos hijos. En este caso al ser un nodo de tipo Name se obtendrá un único nodo hijo. A continuación, el siguiente proceso será utilizar la función *data* y de esta manera se obtendrá el valor de este nodo.

Como se aprecia en el diagrama de flujo, el resultado de esta función podrá ser de tres tipos:

1. No contiene nombre. Este caso se producirá al disponer como parámetro de entrada un elemento no acorde a un objeto DOM.
2. Contiene una información no acorde al exigido. Este caso se producirá al obtener el valor del nodo hijo, cuando la información obtenida no sea de texto.
3. Contiene nombre. Este caso se producirá al obtener la información correctamente, y sea de texto.

2.5. Función procesarCarpeta

Para representar el ordinograma de la función *procesarCarpeta* se ha diferenciado en ocho partes. Todas ellas están agrupados en las siguientes cuatro imágenes: imágenes 102, 103, 104 y 105.

Esta función *procesarCarpeta* contiene dos parámetros de entrada: un elemento DOM y un número identificador de un elemento DOM. El primer parámetro será un elemento DOM (un nodo de tipo Folder) donde se pretende analizar su contenido. En cambio, el segundo parámetro será el número identificador del elemento padre del elemento DOM mencionado (el primer parámetro de entrada). Es decir, este parámetro representará el identificador del nodo (de tipo Folder) del nodo hijo (de tipo Folder) que se va a analizar. Esta función no tiene ningún parámetro de salida.

Como se aprecia en la Imagen 102, existen dos hojas que representan los primeros procesos realizados en esta función. En la primera hoja se representará el procesamiento de la información de zonas. Esta información estará relacionada directamente con los elementos XML

que representan las carpetas (del documento de entrada) donde se almacenan información de diseminados y sus tramos de ruta.

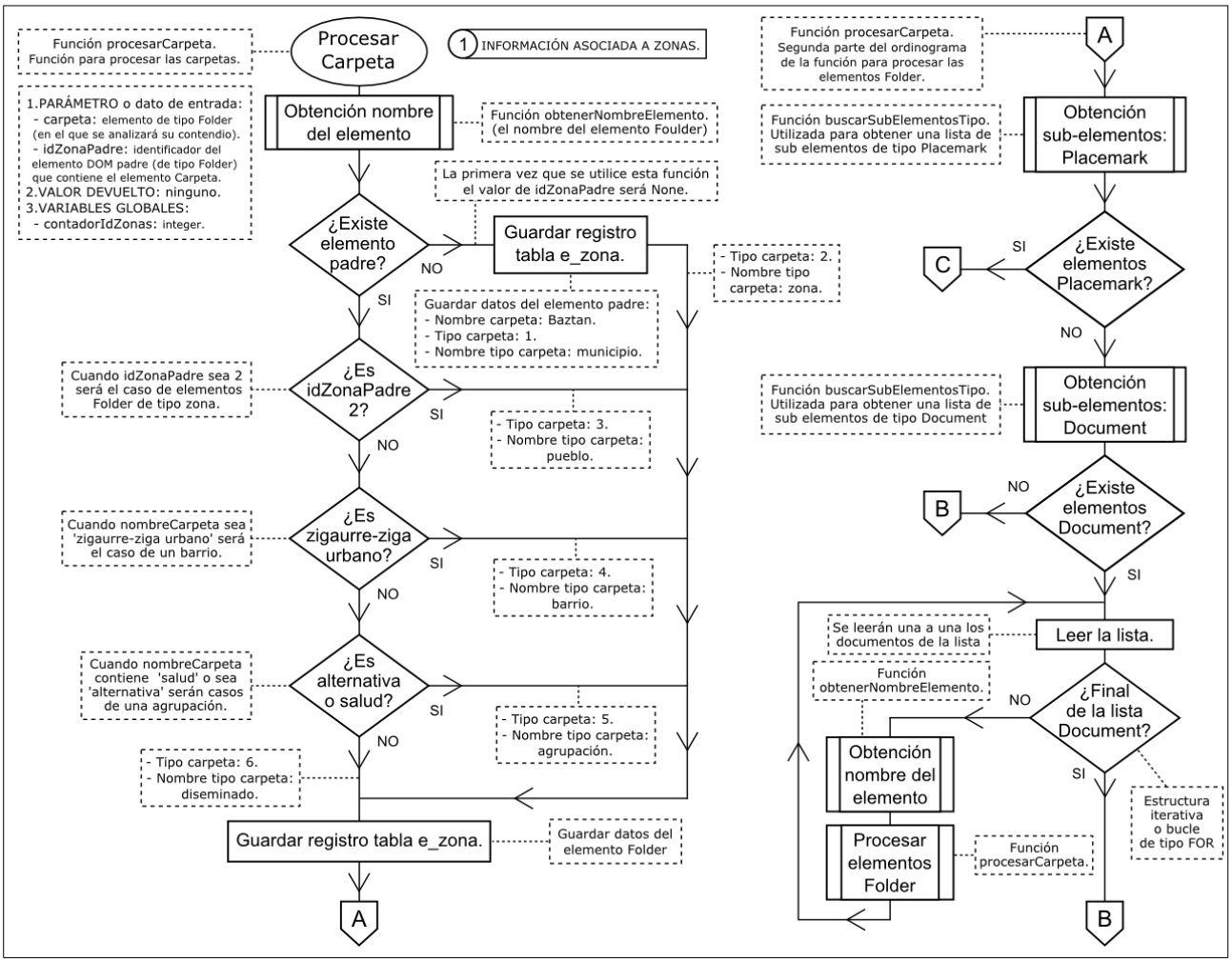

Imagen 102: Ordinograma de la función de Procesar elementos Folder

Por lo tanto, en primer lugar, se procederá a acceder a la información de las zonas. En nuestro caso, esta información estará representada en los nodos de tipo Folder. Este nodo se obtendrá como parámetro de entrada.

1. Primero se obtendrá el nombre del nodo Folder. Para ello se utilizará la función *obtenerNombreElemento*.
2. Posteriormente se clasificará el tipo de información de nodo que se dispone, es decir, el tipo de zona que es. Se clasificará como diseminado, agrupacion, barrio, pueblo, zona y municipio. También se definirá el número identificador de cada zona procesada, y para ello se utilizará la variable global *contadorIdZonas*.
3. Finalmente se guardará toda el conjunto de información obtenida: el identificador de la zona, el nombre de la zona, el tipo de zona y el identificador de la zona padre.

Pero hay que tener en cuenta que la primera vez que se accede a esta función se procederá de diferente manera. Este caso será el del nodo Folder que engloba todo el entorno de trabajo y se almacena todos los datos de entrada, y por lo tanto representará la zona que no con-

tiene ninguna zona padre. En este caso, se procederá a crear un registro de una zona que represente el municipio y que contenga esta zona del entorno de trabajo. Como se aprecia en el ordinograma, primero se guardará el registro de tipo “municipio” (con los datos del municipio de Baztan), y a continuación se guardará el registro de tipo “zona”.

En la segunda hoja de la función se procederá a analizar los sub-elementos que contiene el nodo Folder analizado anteriormente. Este análisis se realizará de la siguiente manera:

1. Primero se analizará si contiene elementos de tipo Placemarks. Si es así se procesará la información que contiene (como se explica en la *cuarta hoja* del ordinograma de la función).
2. A continuación, se analizará si contiene elementos de tipo Document. En este caso se presupone que los elementos de tipo Document contiene unos sub-elementos de tipo Folder. Por lo tanto, se procederá a analizar estos sub-elementos de tipo Folder.
3. Finalmente se analizará si contiene elementos de tipo Folder (como se explica en la *tercera hoja* del ordinograma de la función).

Como se aprecia en la Imagen 103, también existen dos hojas que representan los siguientes procesos realizados en esta función. En la tercera hoja se representará el procesamiento de los sub-elementos Folder antes mencionados. Como se aprecia en el ordinograma, cuando se termine de analizar estos nodos Folder se terminarán los procesos de esta función.

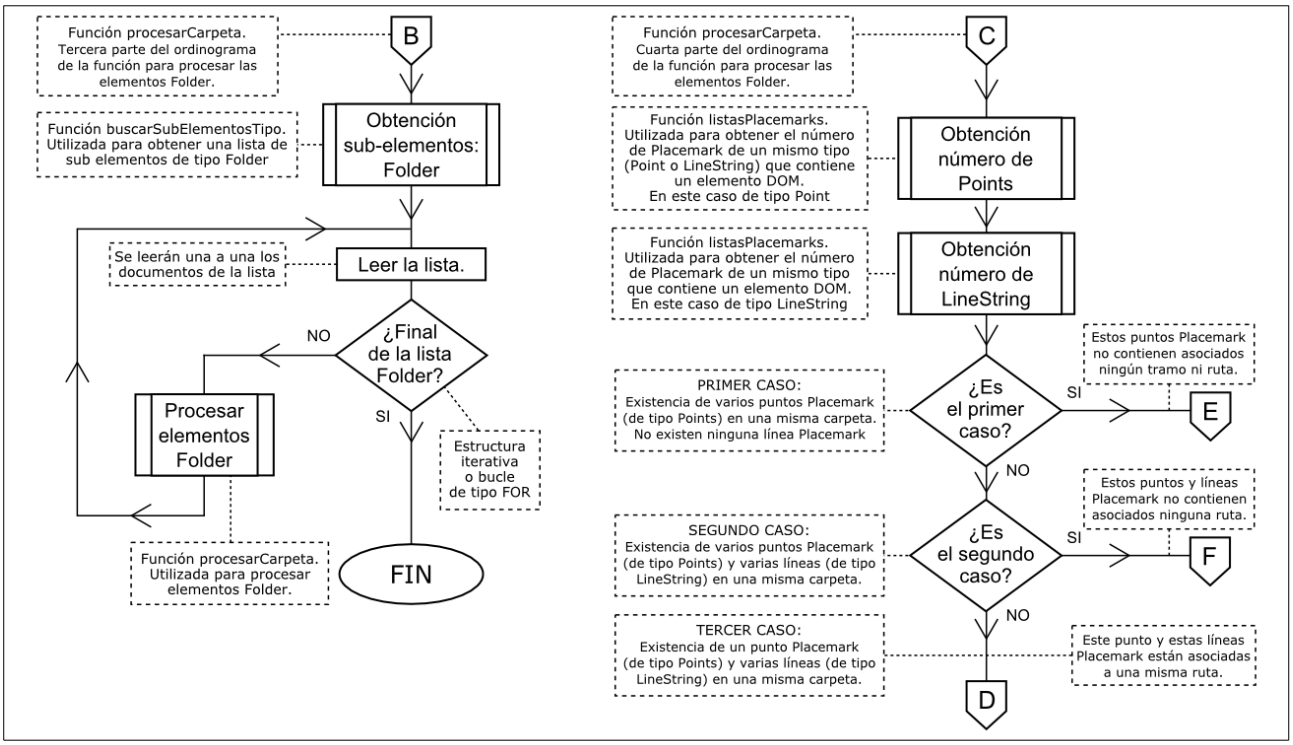

Imagen 103: Ordinograma de la función de Procesar elementos Folder

En la cuarta hoja se representará el procesamiento de los sub-elementos Placemarks. Más concretamente, en esta hoja se representa la clasificación de los tres casos de procesamiento que se plantean para sub-elementos Placemarks. De esta forma, en las siguientes hojas se

procesará los diferentes casos de sub-elementos Placemarks (contenidos en un elemento Folder) encontrados en en el documento KML:

1. Existencia de varios elementos Placemark de tipo Points (de un elementos padre Folder), y ausencia de nodos Placemark de tipo LineString.
2. Existencia de varios elementos Placemarks de tipo Points y de tipo LineString (de un elementos padre Folder).
3. Existencia de un elemento Placemark de tipo Points y varios elementos Placemark de tipo LineString (de un elementos padre Folder).

Para realizar esta clasificación primero se obtendrá el número de casos de elementos Placemark de tipo Point y LineString. Para ello se utilizará la función *listasPlacemarks* (ver el apartado 2.6.).

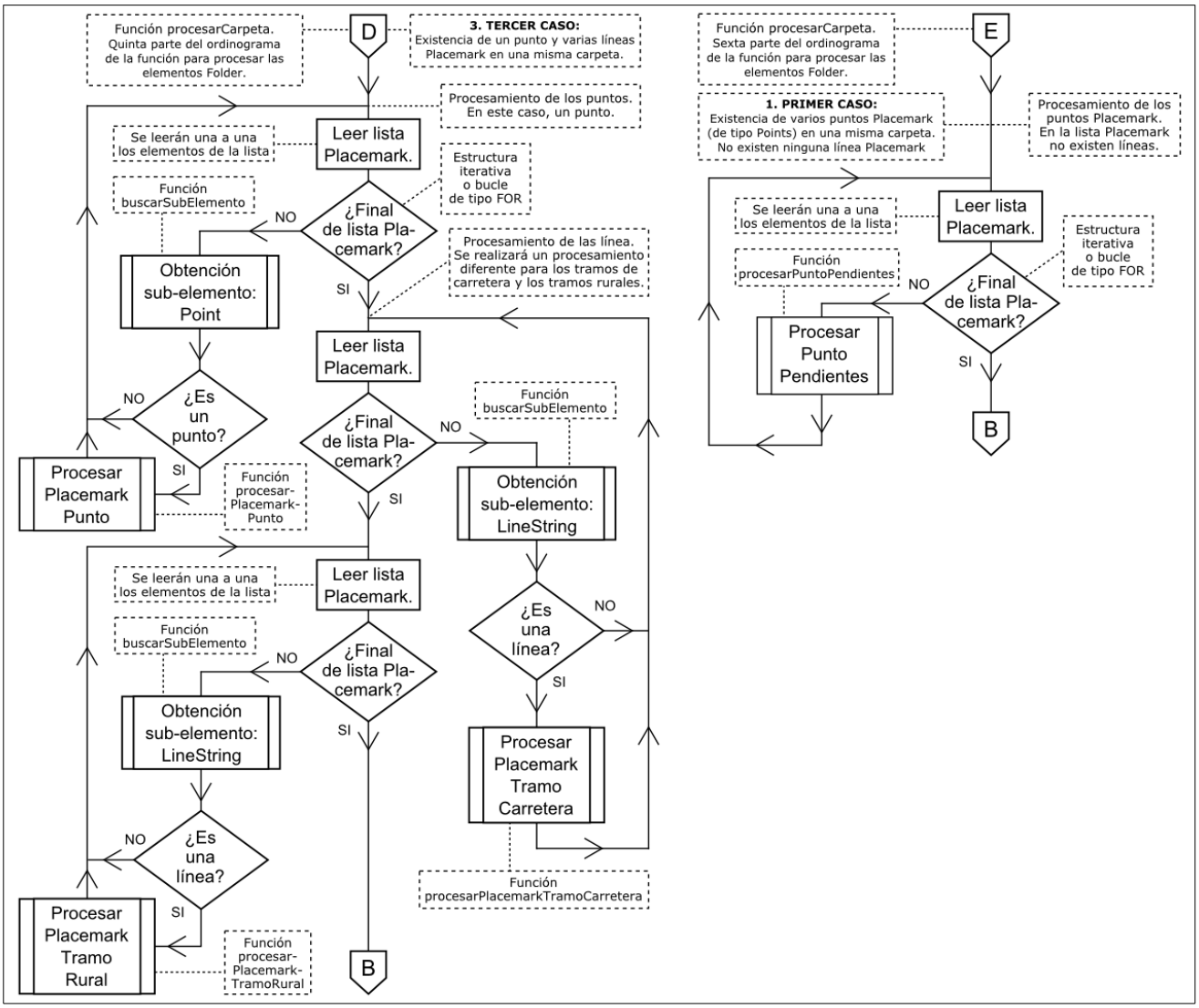

Imagen 104: Ordinograma de la función de Procesar elementos Folder

Como se aprecia en la Imagen 104, también existen dos hojas que representan los siguientes procesos realizados en esta función. En la quinta hoja se representará el tercer caso clasificado anteriormente. Este caso es el que contiene toda la información relacionada con los dise-

minados: la información propia del diseminado y la información relacionada con los tramos asociados a este diseminado.

Por un lado se procesará el elemento Placemark de tipo Point. Primero se obtendrá el elemento del tipo Point de la lista de sub-elementos Placemark y a continuación se procesará utilizando la función *procesarPlacemarkPunto* (ver el apartado 2.9.).

Por otro lado se procesará los elementos Placemark de tipo LineString. En este caso también se procederá de la misma manera: primero se obtendrán los elementos del tipo LineString y a continuación se procesarán. Pero al procesar estos elementos se ha decidido realizarlos de diferente manera los elementos LineString que representan a tramos carretera y a tramos rurales. En estos casos para procesar estos elementos se han utilizado las funciones *procesarPlacemarkTramoCarretera* y *procesarPlacemarkTramoRural* respectivamente (ver los apartados 2.13. y 2.14.).

En la sexta hoja se representa el primer caso clasificado anteriormente (en la cuarta hoja). En este caso sólo existirá la información asociada a los diseminados, y no contendrá ninguna información sobre los tramos. Por lo tanto, se procederá a la obtención de cada uno de estos elementos de tipo Point, y se procesarán utilizando la función *procesarPuntoPendientes* (ver el apartado 2.12.).

Y por último se explicará cómo se ha procesado el segundo caso clasificado anteriormente. Este caso está representado en la Imagen 105 (en la octava y novena hoja). Este caso es más complejo para procesar, ya que existen diferentes casos a tener en cuenta. Al contener un elemento de tipo Folder varios sub-elementos de tipo Point (más de uno) y de tipo LineString pueden existir dos casos principales:

1. Elementos de tipo Point que representan un diseminado y puntos de información. Existen varios tramos asociados a este diseminado (un tramo de carretera y varios tramos rurales).
2. Elementos de tipo Point que representan varios diseminados pendientes de añadir una ruta de acceso, y a su vez existen un conglomerado de tramos (pendientes de añadir información relativa al recorrido).

Por lo tanto, en este procesamiento se han diferenciado los siguientes etapas de procesos:

1. Procesamiento de los elementos de tipo Point que representan a los diseminados.

   1.1. Los casos donde existen elementos Point que representan a un diseminado y a puntos de información.

   1.2. Los casos donde existen elementos Point que representan a diseminados pendientes.

2. Procesamiento de los elementos de tipo LineString que representan a los tramos.

   2.1. Los casos donde existen elementos LineString que representan a un tramo de carretera y a tramos rurales.

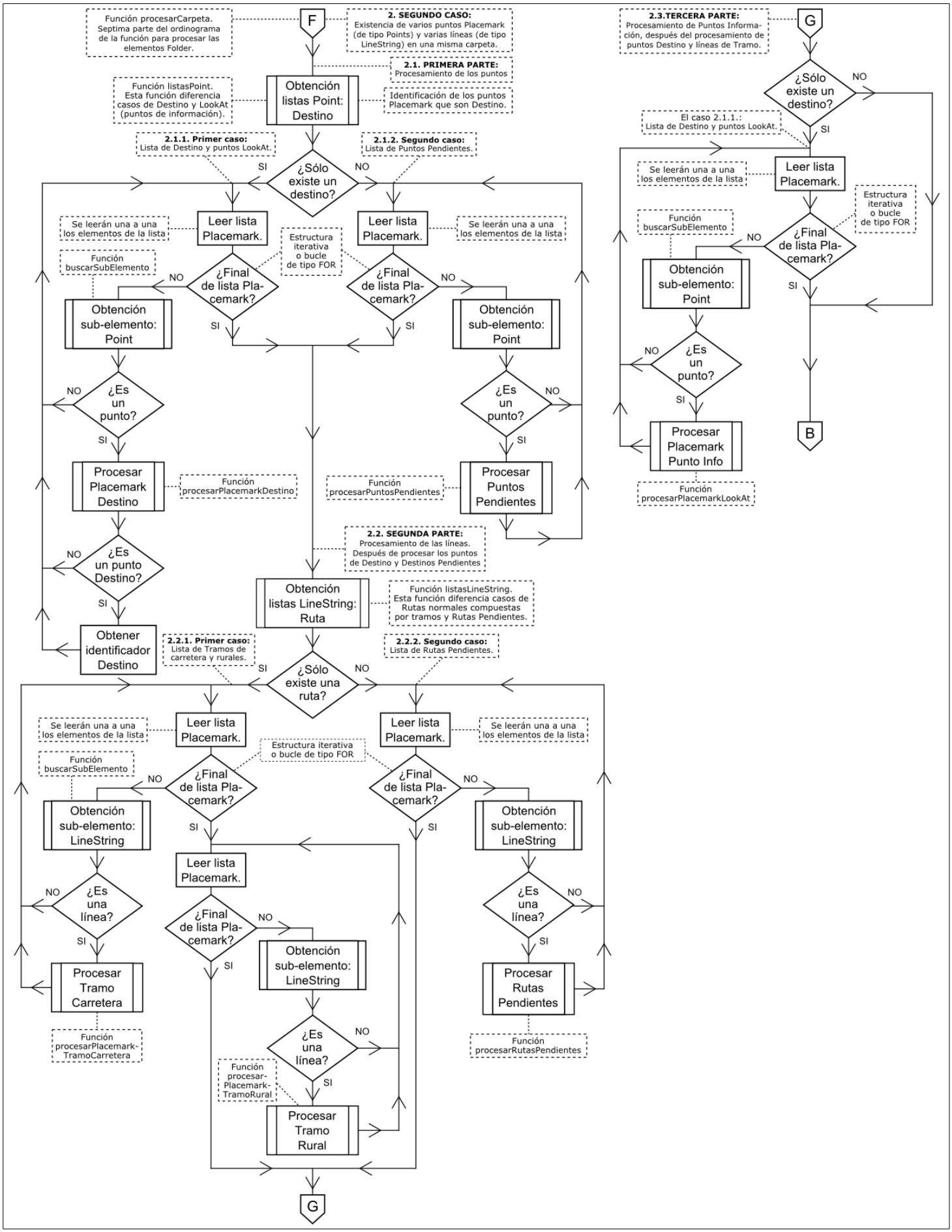

Imagen 105: Ordinograma de la función de Procesar elementos Folder

2.2. Los casos donde existen elementos LineString que representan a tramos pendientes.

3. Procesamiento de los elementos de tipo Point que representan a los puntos de información.

En todos estos casos se procederá de la misma forma: primero se obtendrá los elementos de tipo Point o de tipo LineString (según el caso), y para ello se utilizarán las funciones *listasPoint* (ver el apartado 2.7.) y *listasLineString* (ver el apartado 2.8.). A continuación se procesarán estos elementos utilizando una función acorde a la etapa de procesos planteada anteriormente. De esta forma, se utilizarán las siguientes funciones: *procesarPlacemarkDestino* (ver el apartado 2.10.) y *procesarPlacemarkPuntosPendientes* (ver el apartado 2.12.); *procesarPlacemarkTramoCarretera* (ver el apartado 2.13.), *procesarPlacemarkTramoRural* (ver el apartado 2.14.) y *procesarRutasPendientes* (ver el apartado 2.15.); y *procesarPlacemarkLookAt* (ver el apartado 2.11.).

Hay que mencionar que en el caso de los elementos de tipo Point que representan un diseminado y puntos de información se le añade una operación relevante: al procesar la información relacionada con el diseminado se obtiene el identificador de este diseminado. Por lo tanto, al procesar los elementos de tipo LineString y obtener la información asociada a los tramos se le añadirá este identificador del diseminado. También al procesar los elementos de tipo Point que representan puntos de información se le añadirá este identificador.

Por lo tanto, se ve la importancia que tiene el orden de procesamiento de esta información. Por ejemplo, al procesar la información relacionada con los puntos de información (o advertencia) después de procesar la información de los tramos crea la posibilidad de combinar diferentes informaciones. Es decir, se podrá añadir información obtenida de los tramos a la información obtenida de los puntos de advertencia (por ejemplo la ruta o tramo a que pertenece dicha punto). Pero como se verá más adelante, sólo se le ha añadido el identificador del diseminado.

2.6. Función listaPlacemarks

Como se observa en el ordinograma de la función *listaPlacemarks*, esta función contendrá dos parámetros de entrada: una lista de elementos DOM y un tipo de elemento. El primer parámetro será una lista de nodos de tipo Placemark donde se realizará la búsqueda. Esta búsqueda lo determinará el segundo parámetro, el tipo de elemento. Este parámetro será de tipo Point o de tipo LineString.

Como se puede observar en el primer ordinograma de la Imagen 106 se analizará cada elemento de esta lista para ver si existe coincidencia con el tipo de elemento a buscar. Por lo tanto, para analizar la coincidencia de la tipología se tendrá que analizar los sub-elementos de estos elementos Placemark. De esta forma, se utilizará la variable *contadorPlacemark* para contar las coincidencias. Esta variable será el parámetro de salida de la función.

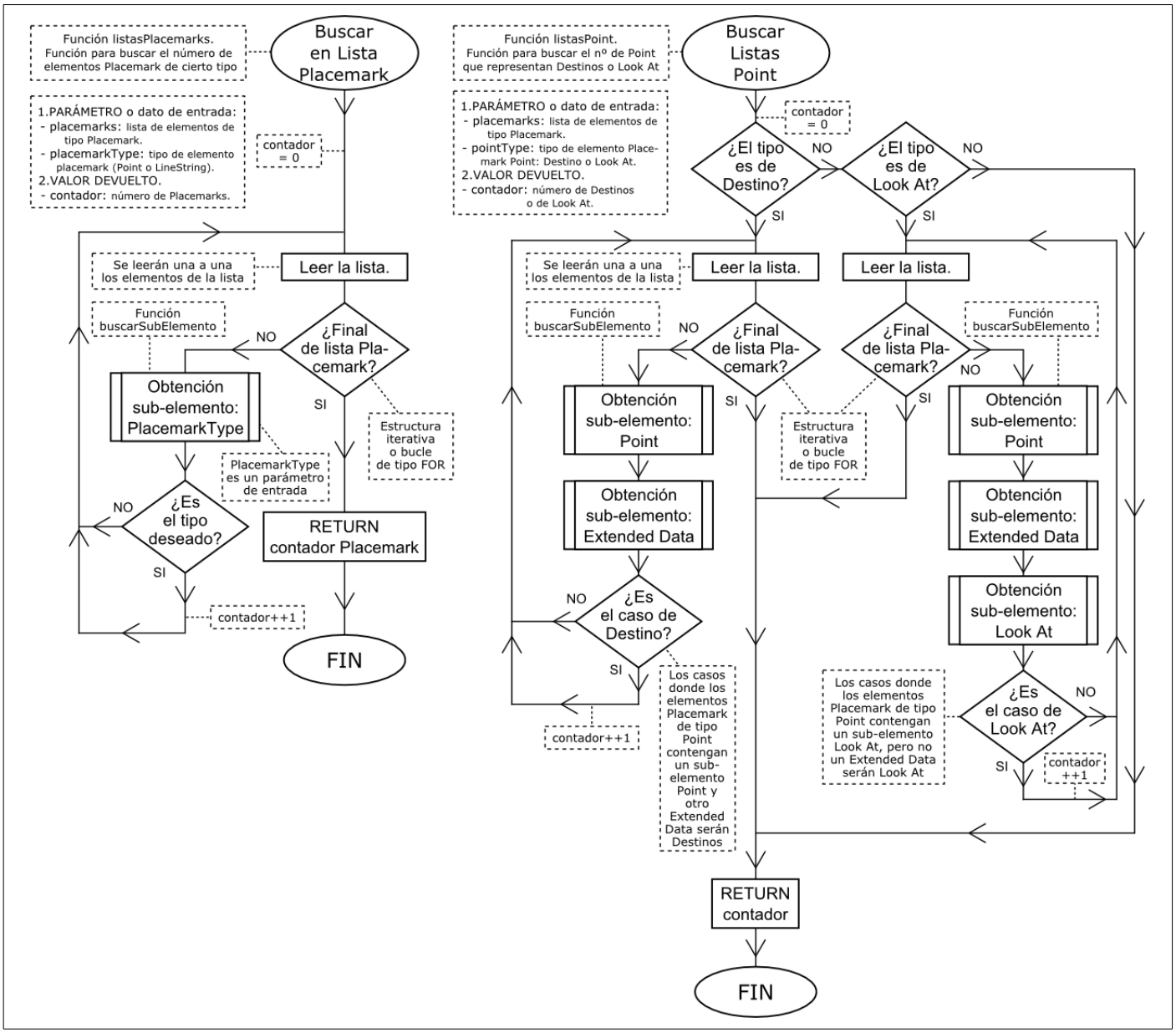

Imagen 106: Ordinograma de la función para obtener el número de Destinos o de Look At

2.7. Función listasPoint

Como se observa en el ordinograma de la función *listasPoint*, esta función contendrá dos parámetros de entrada: una lista de elementos DOM y un tipo de elemento. El primer parámetro será una lista de nodos de tipo Placemark donde se realizará la búsqueda. Esta búsqueda lo determinará el segundo parámetro, el tipo de elemento. Este parámetro será de tipo Destino o de tipo Puntos de Información (en este caso, denominado como Look At).

Como se puede observar en el ordinograma de la Imagen 107 se analizarán cada elemento de la lista Placemark para obtener el número de caso de Destino o de Look At (puntos de información). Los casos coincidentes se representarán en el contador, es decir, se utilizará la variable *contadorPoint* para contar las coincidencias. Esta variable será el parámetro de salida de la función.

Los casos de Destino serán elementos Placemark (de tipo Point) que contengan un sub-elemento ExtendedData, en cambio los casos de Puntos de Información (Look At) serán los ele-

mentos que no contengan un sub-elemento Extended Data y contengan un sub-elemento LookAt.

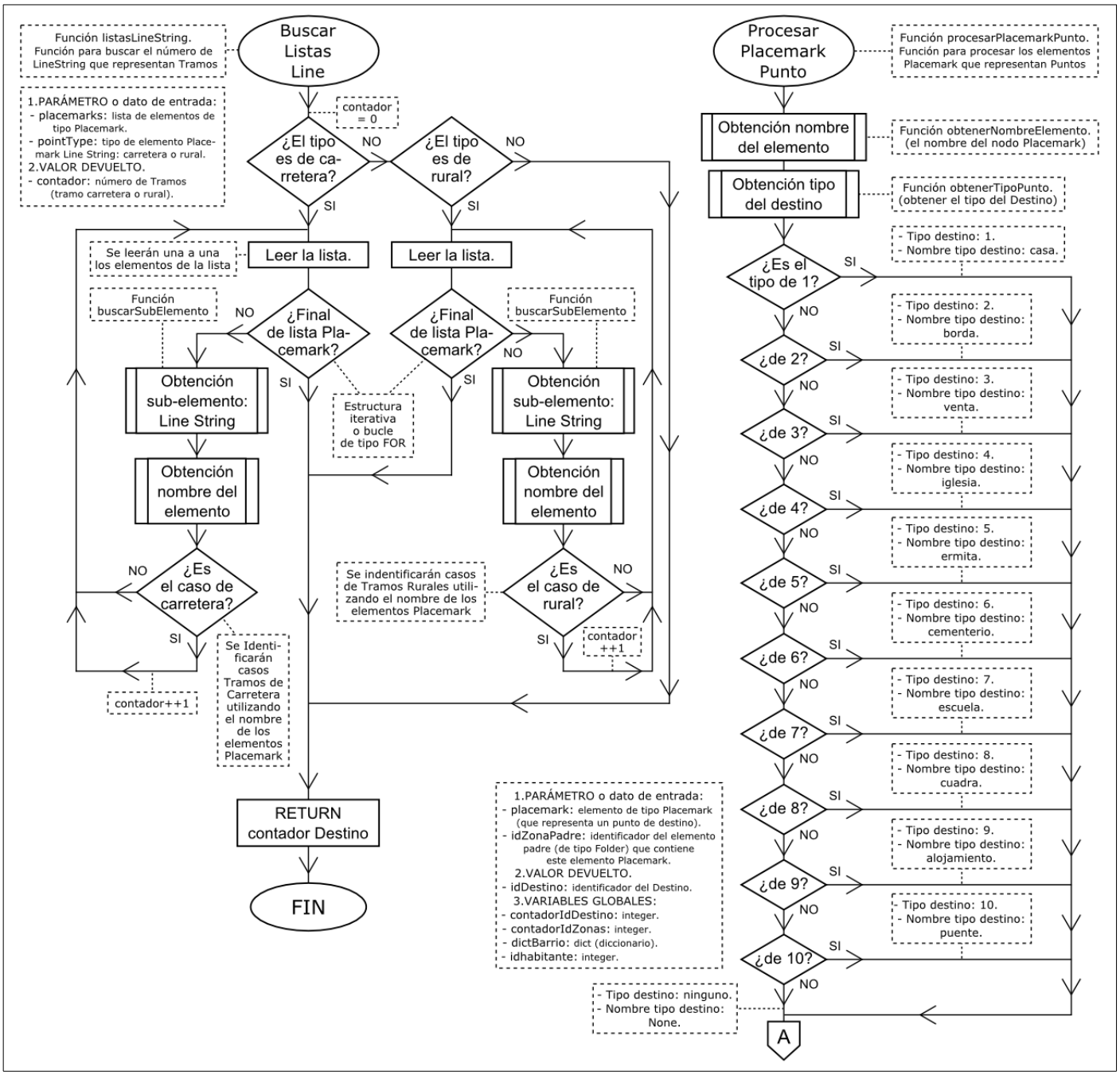

Imagen 107: Ordinograma de las funciones de listar Placemarks y de procesar Placemark Puntos

2.8. Función listasLineString

Como se observa en el ordinograma de la función *listasLineString*, esta función contendrá dos parámetros de entrada: una lista de elementos DOM y un tipo de elemento. El primer parámetro será una lista de nodos de tipo Placemark donde se realizará la búsqueda. Esta búsqueda lo determinará el segundo parámetro, el tipo de elemento. Este parámetro será de tipo Tramo Carretera o de tipo Tramo Rural.

Para diferenciar estos dos casos de Tramos se utilizarán los nombres de los elementos Placemark.

Como se puede observar en el ordinograma de la Imagen 108 se analizarán cada elemento de la lista Placemark para obtener el número de caso de Carretera o de Rural. Los casos coincidentes se representarán en el contador, es decir, se utilizará la variable *contadorLineString* para contar las coincidencias. Esta variable será el parámetro de salida de la función.

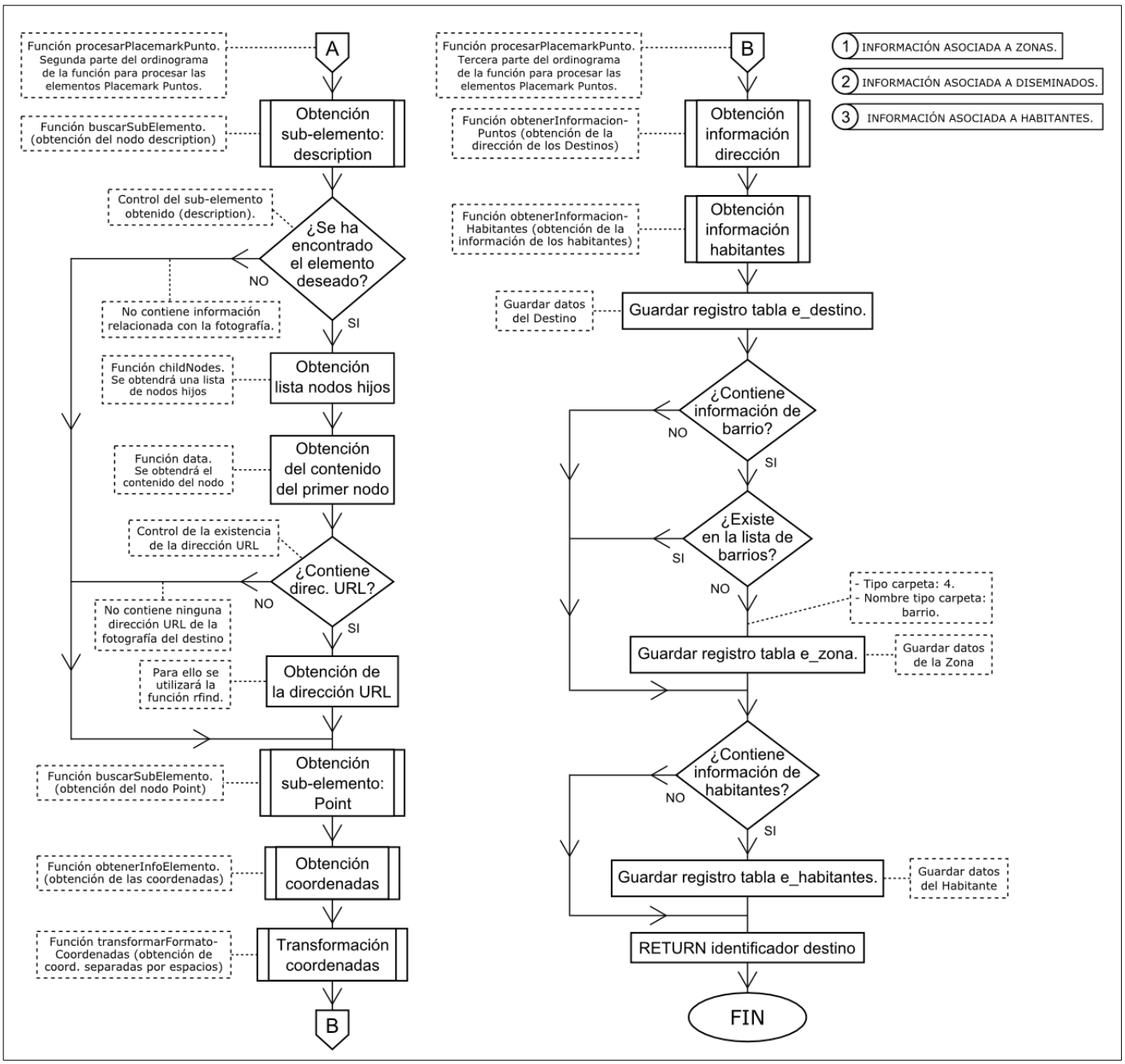

Imagen 108: Ordinograma de la función para procesar Placemark Puntos

Más concretamente se utilizarán las siguientes funciones:

- Para obtener el nombre del destino se utilizará la función *obtenerNombreElemento* (ver el apartado 2.3.).
- Para obtener el tipo del destino se utilizará la función *obtenerTipoPunto* (ver el apartado 2.16.).
- La dirección URL de la fotografía estará almacenada en el nodo Description. Por lo tanto, para obtener este nodo se utilizará la función *buscarSubElemento* (ver el apartado 2.1.).

- Las coordenadas estarán almacenadas en el nodo Point. Por lo tanto, para obtener este nodo se utilizará la misma función *buscarSubElemento*. Y a continuación, se utilizará la función *obtenerInfoElemento* (ver el apartado 2.17.) para obtener las coordenadas. Se cambiará el formato de estas coordenadas, y para ello se utilizará la función *transformarFormatoCoordenadas* (ver el apartado 2.18.).
- Para obtener la información de la dirección del destino se utilizará la función *obtenerInformacionPuntos* (ver el apartado 2.19.).
- Para obtener la información de los habitantes del destino se utilizará la función *obtenerInformacionHabitantes* (ver el apartado 2.20.).

2.9. Función procesarPlacemarkPunto

Como se observa en el ordinograma de la función *procesarPlacemarkPunto* se han creado diferentes partes de este ordinograma para explicar el procesamiento de la información de los Destinos. En la primera parte del ordinograma podemos observar que esta función contendrá dos parámetros de entrada: un elemento DOM de tipo Placemark y el identificador del elemento DOM de tipo Folder. Este segundo parámetro será el elemento padre del primer parámetro, es decir, del elemento Placemark obtenido.

Como se observa en los ordinogramas de la Imagen 107 e Imagen 108 se procederá a obtener la siguiente información: el nombre, el tipo, la dirección URL de la fotografía, las coordenadas y la dirección del destino, y la información referida a los habitantes. Y al obtener toda esta información se guardará en la tabla referida al destino.

A continuación se analizará si en este conjunto de información contiene información referida al barrio (el barrio donde se sitúa este destino). Si dispone esta información, se consultará si este barrio existe en los registros de zonas obtenidos al utilizar la función *procesarCarpeta* (ver la Imagen 102 del apartado 2.5.). Para realizar esta consulta se utilizará la variable global *dictBarrio* (diccionario). Y por lo tanto, si existe algún caso que no esté en estos registros se almacenará en la tabla de zonas.

También se analizará si en este conjunto de información existe información relacionada con los/las habitantes. Si existe se almacenará en la tabla de habitantes (cada uno de sus miembros).

2.10. Función procesarPlacemarkDestino

Esta función se utilizará en los casos donde existan más de un elemento Placemark de tipo Point en un elemento padre Folder. Por lo tanto, entre estos elementos se tendrá que obtener el que representa al Destino, y posteriormente procesarlo. Para hacer esta discretización se ha analizado la estructura del documento KML, y se ha observado que los elementos Placemarks que representan Destinos contienen un sub-elemento Extended Data (sub-elemento que contiene información referida al diseminado). Por lo tanto, los elementos Placemark (de tipo Point) que no contengan dicho sub-elemento (y contengan un sub-elemento LookAt) se catalogarán como puntos de información.

Como se observa en el ordinograma de la función *procesarPlacemarkDestino* esta función contendrá dos parámetros de entrada: un elemento DOM de tipo Placemark y el identificador del elemento DOM de tipo Folder. Este segundo parámetro será el elemento padre del primer parámetro, es decir, del elemento Placemark obtenido.

Como se observa en la Imagen 109 primero se procederá a obtener el sub-elemento Extended Data. De esta forma se utilizará la función *procesarPlacemarkPunto* solamente en los casos de Destino (ver apartado 2.9.).

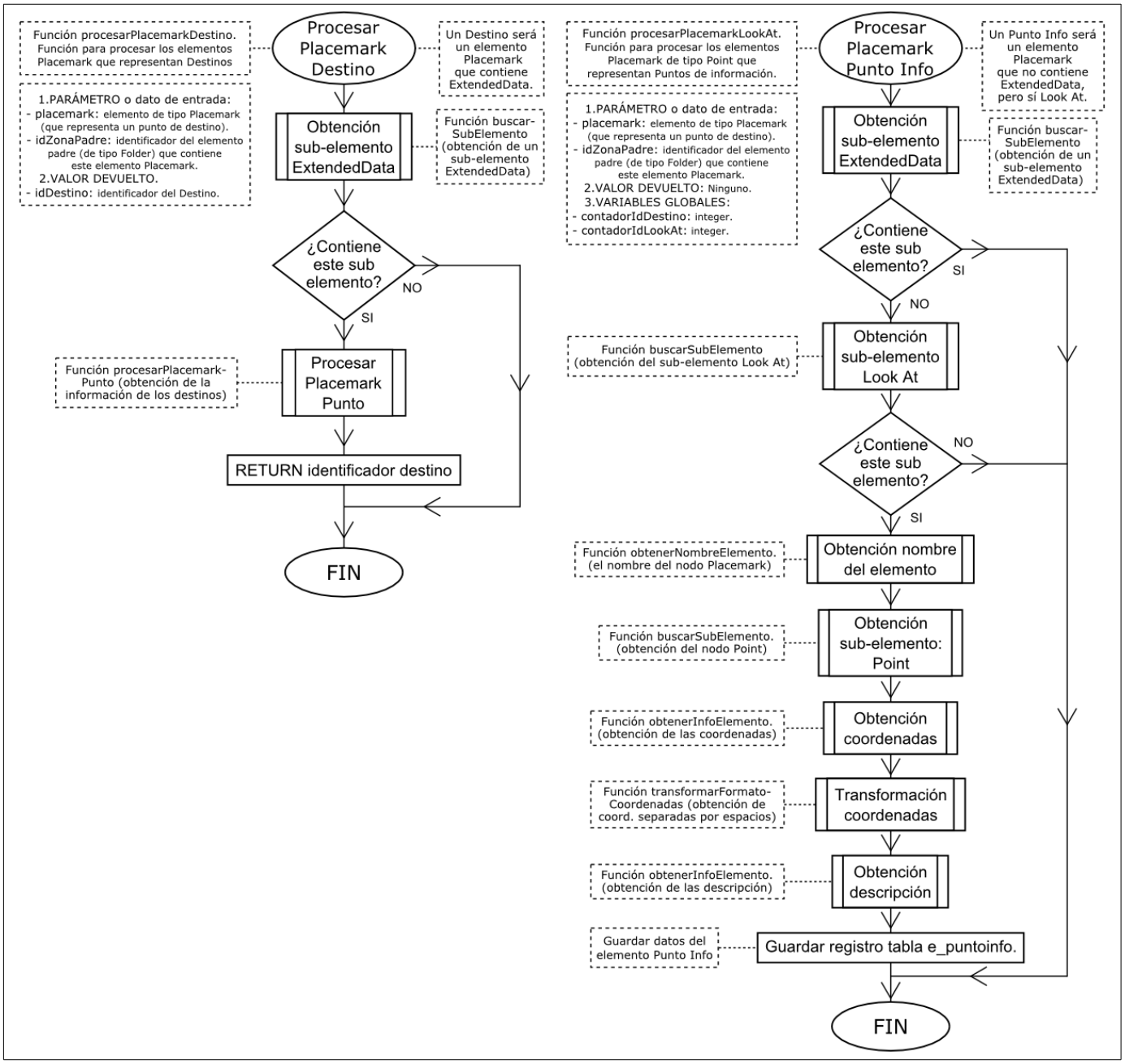

Imagen 109: Ordinogramas de las funciones para procesar Placemark Destinos y Puntos de Información

2.11. Función procesarPlacemarkLookAt

Como se ha mencionado anteriormente los casos donde los elementos Placemark (de tipo Point) que no contengan un sub-elemento ExtendedData y contengan un sub-elemento

LookAt se considerarán puntos de información. Para procesar estos casos se utilizará la función *procesarPlacemarkLookAt*.

Como se observa en el ordinograma de la Imagen 109 esta función contendrá dos parámetros de entrada: un elemento DOM de tipo Placemark y el identificador del elemento DOM de tipo Folder. Este segundo parámetro será el elemento padre del primer parámetro, es decir, del elemento Placemark obtenido.

Primero se procederá a obtener los sub-elementos Extended Data y Look At, para poder así utilizar la función *procesarPlacemarkLookAt* solamente en los casos de Puntos de Información. Para ello, se utilizará la función *buscarSubElemento* (ver apartado 2.1.).

Por lo tanto, en estos casos se obtendrán las siguientes informaciones: el nombre, las coordenadas y la descripción (las indicaciones o los avisos) de los puntos de información.

Más concretamente se utilizarán las siguientes funciones:

- Para obtener el nombre del destino se utilizará la función *obtenerNombreElemento* (ver el apartado 2.3.).
- Las coordenadas estarán almacenadas en el nodo Point. Por lo tanto, para obtener este nodo se utilizará la función *buscarSubElemento*. Y a continuación, se utilizará la función *obtenerInfoElemento* (ver el apartado 2.17.) para obtener las coordenadas. Se cambiará el formato de estas coordenadas, y para ello se utilizará la función *transformarFormatoCoordenadas* (ver el apartado 2.18.).
- Para obtener la información de la descripción del destino se utilizará la función *obtenerInfoElemento* (ver el apartado 2.17.).

Al obtener toda esta información se guardará en la tabla referida a los puntos de información.

2.12. Función procesarPuntosPendientes

Como se observa en el ordinograma de la función *procesarPuntosPendientes* esta función contendrá dos parámetros de entrada: un elemento DOM de tipo Placemark y el identificador del elemento DOM de tipo Folder. Este segundo parámetro será el elemento padre del primer parámetro, es decir, del elemento Placemark obtenido.

Como se observa en la Imagen 110 e Imagen 111, el ordinograma de la función *procesarPuntoPendientes* contiene la misma estructura que el ordinograma de la función *procesarPlacemarkPunto* (ver apartado 2.9.). Es decir, para obtener la información de los destinos pendientes se procederá de la misma manera, y para ello se utilizarán las mismas funciones.

De esta forma, se obtendrán con esta función el nombre, el tipo, la dirección URL de la fotografía, las coordenadas y la dirección de los destinos pendientes, y también la información referida a los habitantes. Toda esta información se guardará en la tabla referida a los Puntos Pendientes.

En este caso también se analizará si contienen información referida al barrio e información relacionada con los/las habitantes. Y si existe, en los dos casos se almacenará en la tabla zonas y en la tabla habitantes.

Después de incorporar estas tablas a la base de datos, habrá que analizar el contenido de la tabla de Puntos Pendientes y la tabla Destino: se analizarán los casos de coincidencia (si contienen los mismos datos) y se incorporarán a la tabla Destino los Puntos Pendientes que no coincidan.

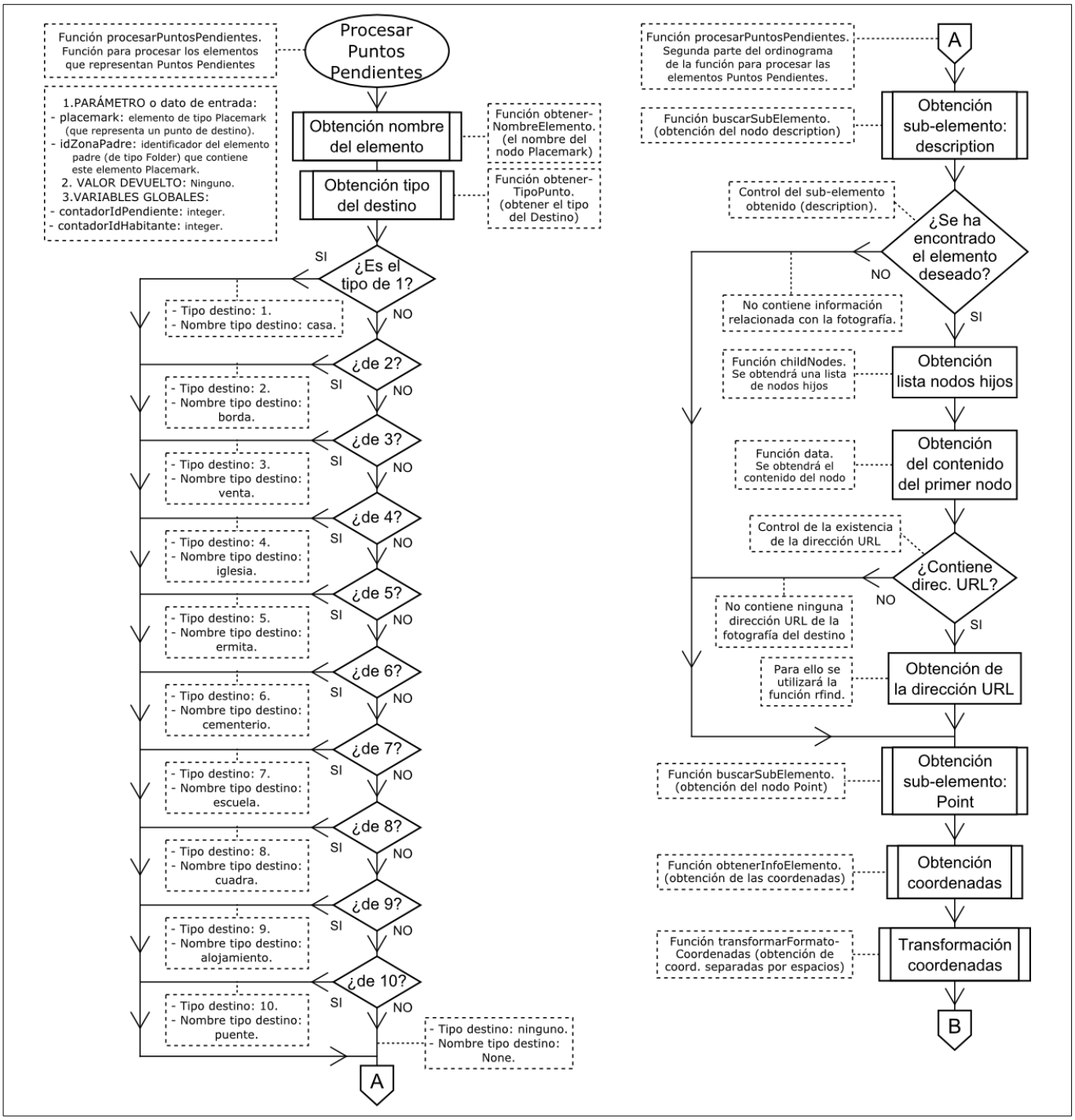

Imagen 110: Ordinograma de la función para procesar Puntos Pendientes

2.13. Función procesarPlacemarkTramoCarretera

Como se aprecia en la Imagen 111 la función *procesarPlacemarkTramoCarretera* contendrá dos parámetros de entrada: un elemento DOM de tipo Placemark y el identificador de este elemento DOM. Este segundo parámetro será el identificador de destino del elemento Placemark (primer parámetro).

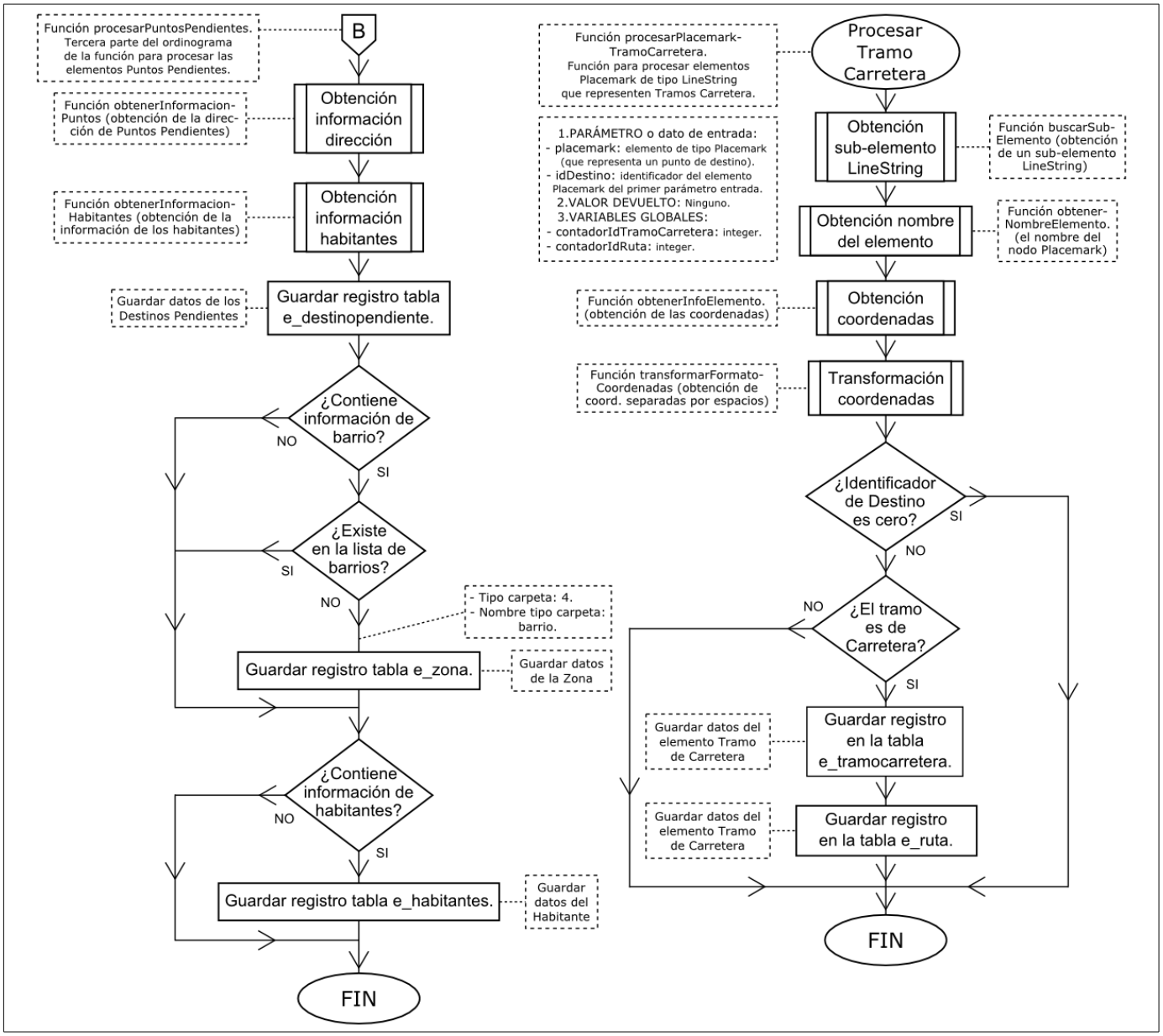

Imagen 111: Ordinogramas de las funciones para procesar Puntos Pendientes y Tramos de Carretera

Por lo tanto, con esta función se procederá a recabar la siguiente información: el nombre del tramo y las coordenadas de los polilínea del tramo.

Para ello, se utilizarán las siguientes funciones:

- Para obtener el nombre del destino se utilizará la función *obtenerNombreElemento* (ver el apartado 2.3.).
- Las coordenadas estarán almacenadas en el nodo Point. Por lo tanto, para obtener este nodo se utilizará la función *buscarSubElemento*. Y a continuación, se utilizará la función *obtenerInfoElemento* (ver el apartado 2.17.) para obtener las coordenadas. Se cambiará

el formato de estas coordenadas, y para ello se utilizará la función *transformarFormato-Coordenadas* (ver el apartado 2.18.).

Al obtener toda esta información se guardará en la tabla referida a los Tramos de Carretera.

2.14. Función procesarPlacemarkTramoRural

Como se aprecia en la Imagen 112 e Imagen 113 la función *procesarPlacemarkTramoRural* contendrá los mismos parámetros de entrada que la función anterior.

Como se aprecia en el ordinograma el primer paso será recabar la información del nombre del tramo y las coordenadas de los polilínea del tramo. También se cambiará el formato de estas coordenadas.

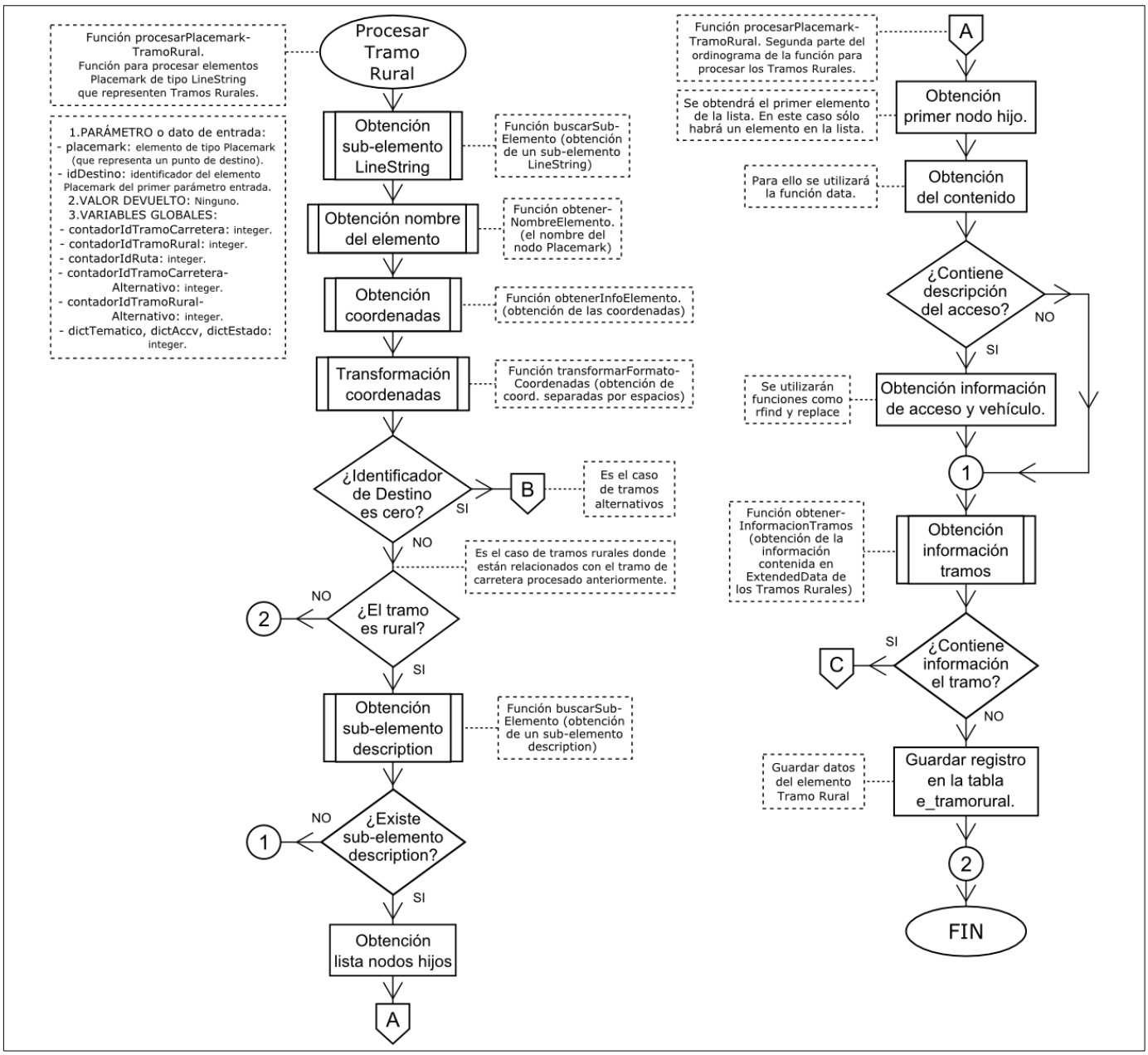

Imagen 112: Ordinograma de la función para procesar Tramos Rurales

A continuación, se analizará si el segundo parámetro de entrada contiene un valor. Si es así, significará que este tramo está asociada a un identificador de un destino. Por lo tanto, será un caso de un tramo rural (de la ruta oficial). Si por el contrario el segundo parámetro de entrada es cero, significará que es el caso de un tramo de ruta alternativa. Estos tramos estarán agrupados en un nodo de tipo Folder que está situado dentro del nodo que representa un destino.

En los casos de tramos oficiales se procederá a obtener la información contenida en el nodo descripción (información de acceso y vehículo) y la contenida en el nodo ExtendedData (fecha de los datos, longitud del tramo, datos del vehículo de acceso, datos del suelo de los tramos, etc.). Para obtener la segunda fuente de información se ha utilizado la función *obtenerInformacionTramos* (ver el apartado 2.21). Al obtener toda esta información se guardará en la tabla referida a los Tramos Rurales.

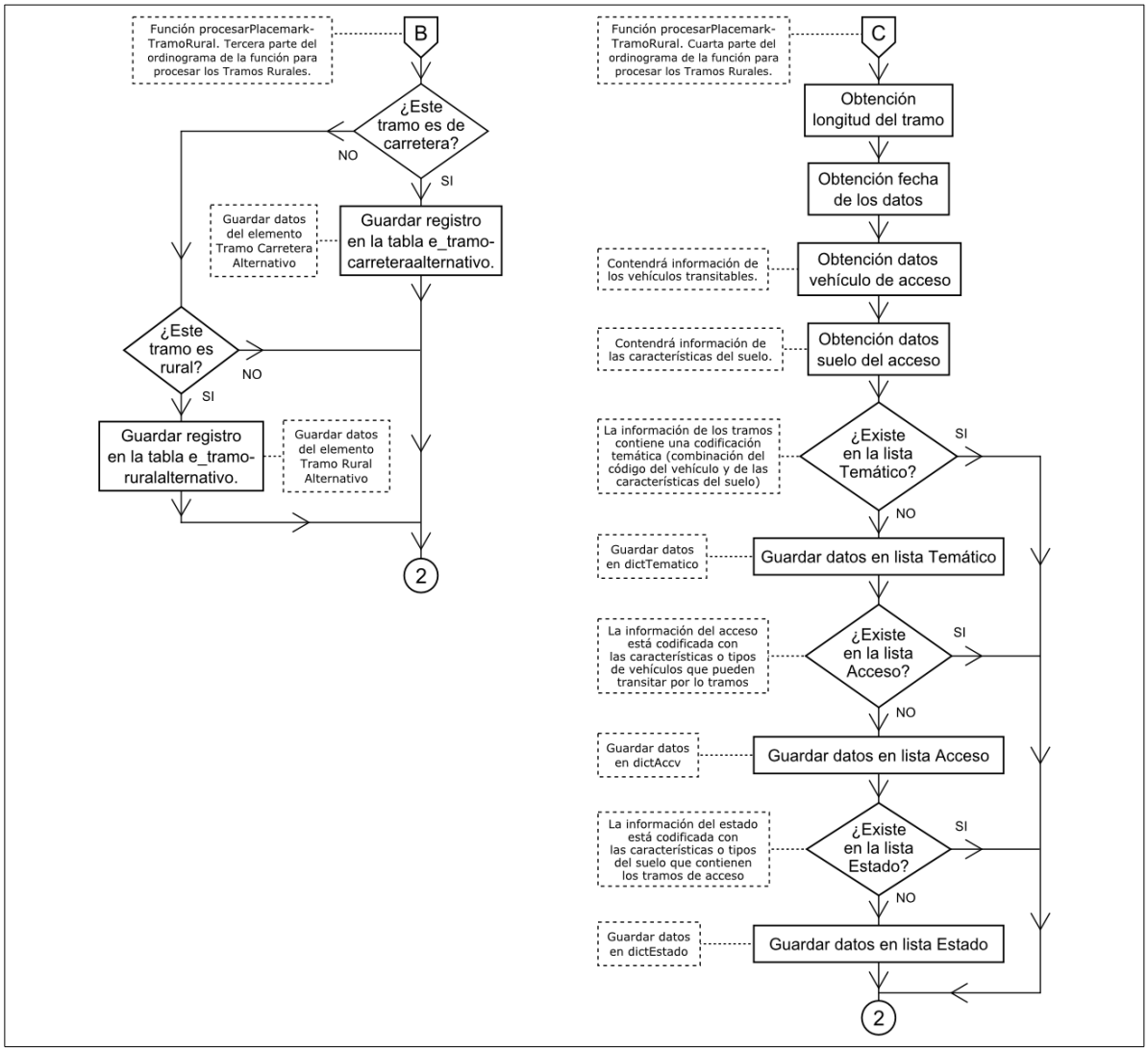

Imagen 113: Ordinograma de la función para procesar Tramos Rurales

En los casos de tramos alternativos se analizará si son un tramo de carretera o un tramo rural, y posteriormente se guardará la información que contiene en las tablas de los Tramos de

Carretera Alternativos y de los Tramos Rurales Alternativos. En estos casos heredarán el identificador del destino de la ruta oficial (procesada antes que la ruta alternativa).

2.15. Función procesarRutasPendientes

Como se aprecia en la Imagen 114 la función *procesarRutasPendientes* contendrá los mismos parámetros de entrada que las dos funciones anteriores.

Como en el ordinograma de la función *procesarTramoCarretera* (ver apartado 2.13.), en éste también se obtendrán el nombre del tramo y las coordenadas de los polilínea del tramo. Y también se cambiará el formato de estas coordenadas. Toda esta información se guardará en la tabla referida a los Rutas Pendientes.

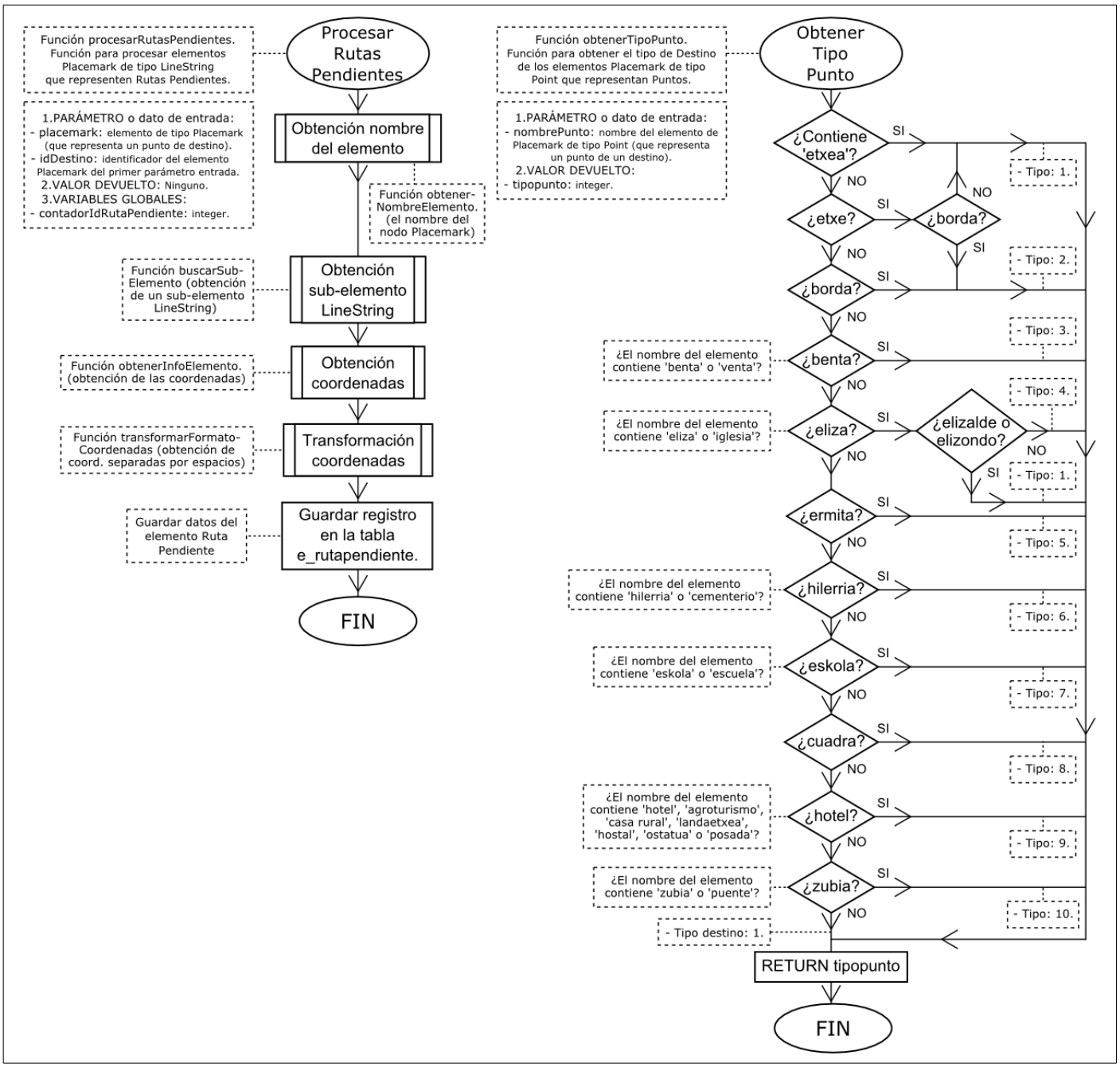

Imagen 114: Ordinogramas de las funciones para procesar Rutas Pendientes y para obtener Tipo Punto

Después de incorporar esta tabla a la base de datos, habrá que analizar el contenido de la tabla de Rutas Pendientes y la tabla referida a tramos: se analizará si existen casos de coincidencia.

2.16. Función obtenerTipoPunto

Como se aprecia en la Imagen 115 la función *obtenerTipoPunto* contendrá un parámetro de entrada: un nombre de un elemento DOM (Placemark de tipo Point).

Con esta función se analizará el contenido del nombre, y se le adjudicará un valor numérico al tipo de punto. De esta forma, se clasificarán los destinos dependiendo de si representan a casas, bordas, ventas, iglesias, cementerios, escuelas, cuadras, alojamientos y puentes. Este valor numérico será el parámetro de salida de la función.

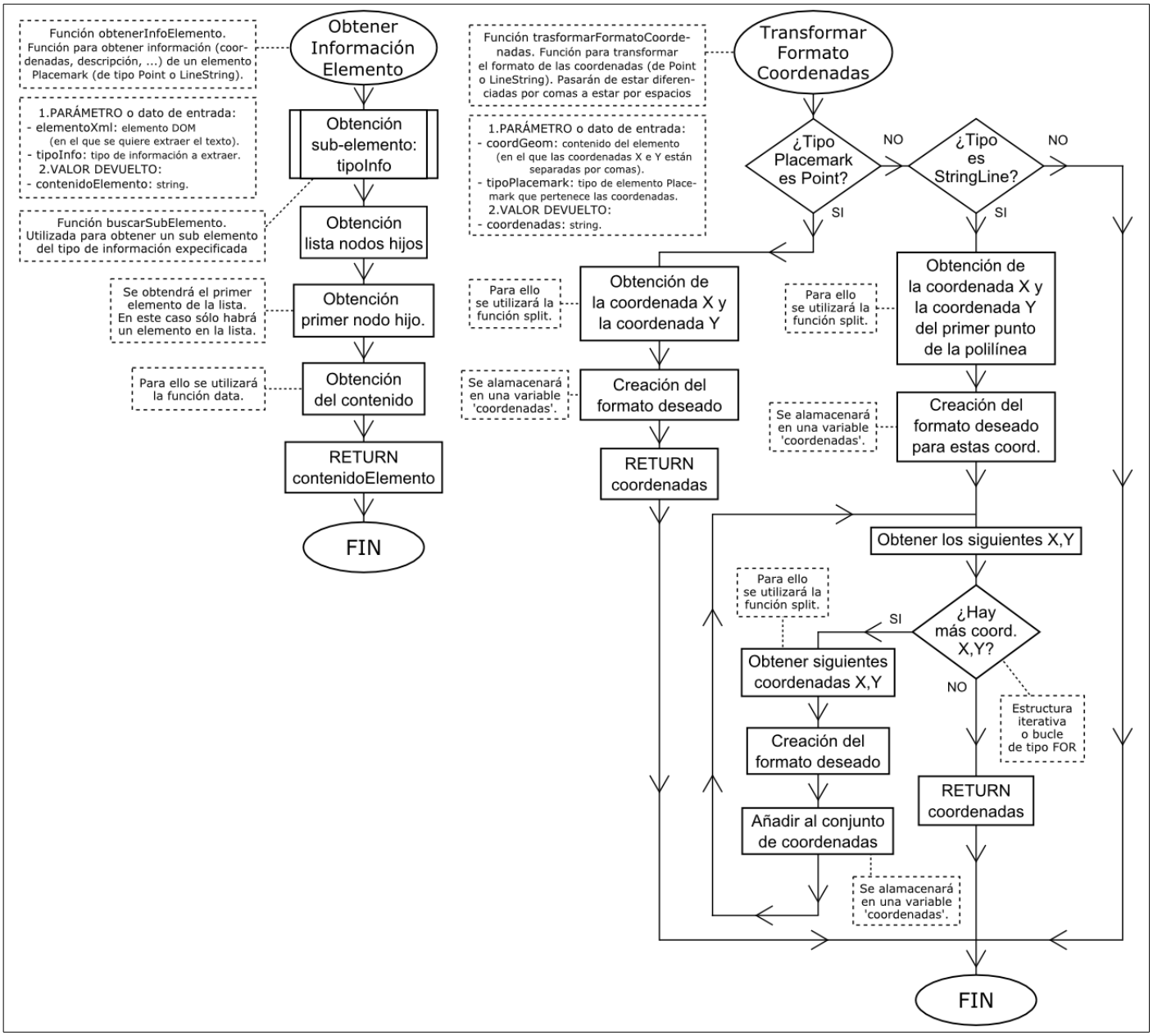

Imagen 115: Ordinogramas de las funciones para obtener información del elemento y de las coordenadas

2.17. Función obtenerInfoElemento

Como se aprecia en la Imagen 115 la función *obtenerInfoElemento* contendrá dos parámetros de entrada: un elemento DOM de tipo Placemark y el tipo de contenido. Este segundo parámetro será el contenido que se quiere extraer del elemento Placemark (primer parámetro).

Con esta función primero se obtendrá el sub-elemento (del elemento DOM especificado en el primer parámetro) que contenga el tipo especificado (en el segundo parámetro). En nuestro caso, este tipo será de coordinates o de description. Y a continuación se obtendrá la información que contiene este sub-elemento. Esta información será el parámetro de salida.

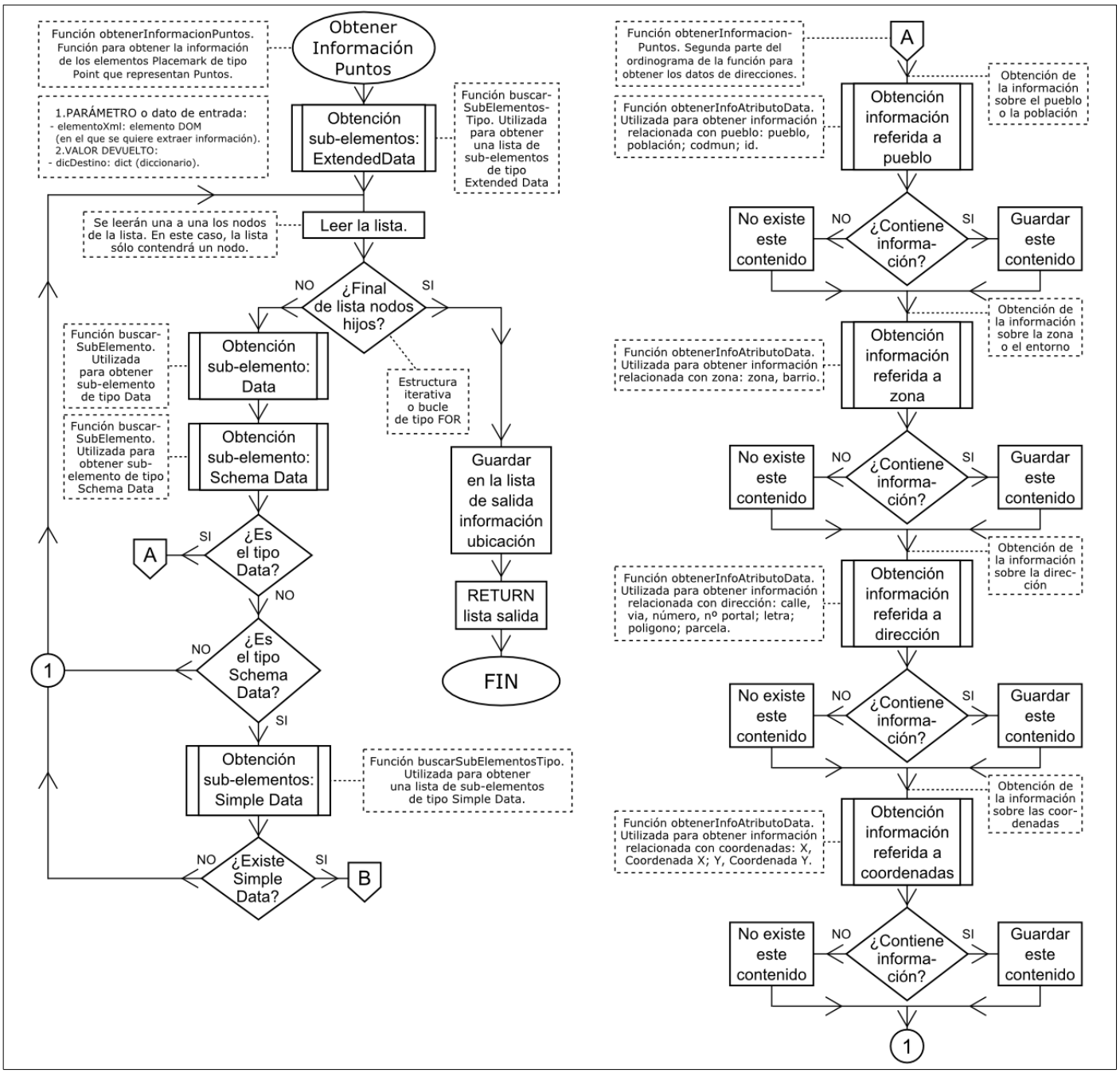

Imagen 116: Ordinograma de la función para obtener la información del elemento Puntos

2.18. Función transformarFormatoCoordenadas

Como se aprecia en la Imagen 115 la función *transformarFormatoCoordenadas* contendrá dos parámetros de entrada: un conjunto de coordenadas y el tipo de elemento Placemark. El primer parámetro contendrá el texto obtenido con la función *obtenerInfoElemento* (del apartado anterior), y las coordenadas estarán en un formato que se desea transformar (para utilizarlo en el código SQL). El segundo parámetro será el tipo de Placemark del elemento DOM que pertenecen las coordenadas. En nuestro caso los tipos utilizados serán Point y String Line.

Por lo tanto, en el caso de obtener las coordenadas de un elemento de tipo Point se procederá de la siguiente manera: primero se obtendrán las coordenadas X e Y, posteriormente se creará el formato que se quiere obtener, y a continuación, se procederá a devolver este texto de coordenadas (como parámetro de salida).

En el caso de las coordenadas de una polilínea, se tendrá que trabajar una a una todas las coordenadas. Para ello, se transformará el formato de cada una de las coordenadas X e Y, y se guardarán en un variable. Al acabar de transformar todas las coordenadas, se procederá a devolver el valor de la variable donde se han guardado todas estas coordenadas.

2.19. Función obtenerInformacionPuntos

Como se aprecia en la Imagen 116 e Imagen 117 la función *obtenerInformacionPuntos* contendrá un parámetro de entrada: un elemento DOM (Placemark de tipo Point).

La información que se desea acceder con esta función estará contenida en el sub-elemento (del elementos Placemark de tipo Point) de tipo Extended Data. Más concretamente en los sub-elementos que contiene este elemento Extended Data. En algunos casos estos sub-elementos serán de tipo Data y en otras de tipo Schema Data. Por lo tanto, en estos dos casos se procederá de diferente manera para obtener la información de los puntos.

En el primer caso, para obtener esta información habrá que acceder al elemento Extended Data y a continuación a sus sub-elementos Data (como se aprecia en la Imagen 98). Cada uno de estos sub-elementos contendrá diferente información, y por ello para obtener cada información se utilizará el valor del atributo name de cada elemento Data. De esta forma, se accederá a cada uno de estos elementos Data para obtener el sub-elemento Value, y posteriormente extraer su contenido. La información obtenida se guardará en su variable correspondiente.

En el segundo caso, el elemento Extended Data contendrá un sub-elemento Schema Data, y a su vez contendrá el conjunto de sub-elementos Simple Data que contendrán dicha información. Como en el caso anterior la información se clasificará a partir del atributo Name que contienen estos elementos. Pero en este caso, la información estará ubicada dentro del elemento Simple Data como el contenido.

En los dos casos para obtener cada información se ha utilizado las funciones *obtenerInfoAtributoData* (ver apartado 2.22.) y *obtenerInfoAtributoSimpleData* (ver apartado 2.23.) respectivamente. De esta manera se intentará obtener la siguiente información:

1. Información sobre el pueblo donde se sitúa: pueblo, población; codmun; IdPueblo, id.
2. Información sobre el entorno donde se sitúa: zona, barrio.
3. Vía donde se ubica: calle, vía; idenvia.
4. Portal donde se ubica: número, nº portal; letra.
5. Polígono donde se ubica: poligono.
6. Parcela donde se ubica: parcela.
7. Posicionamiento espacial: X, Coordenada X, X_25830; Y, Coordenada Y, Y_25830.

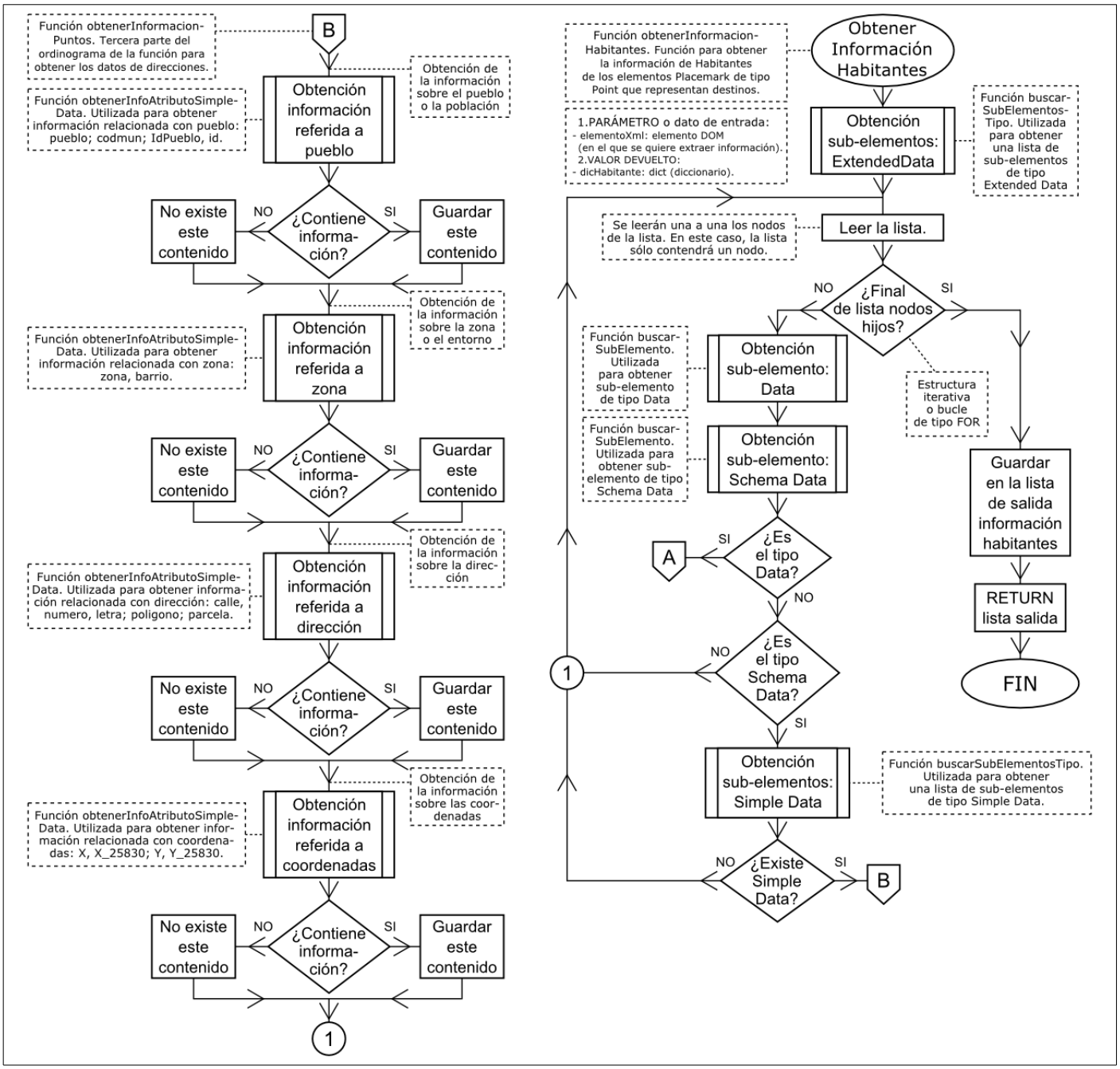

Imagen 117: Ordinograma de las funciones para obtener información sobre puntos y sobre habitantes

Al obtener toda la información contenida en cada punto, se almacenará en una lista, y ésta será el parámetro de salida de la función.

2.20. Función obtenerInformacionHabitantes

Como se aprecia en la Imagen 117 e Imagen 118 la función *obtenerInformacionHabitantes* contendrá un parámetro de entrada: un elemento DOM (Placemark de tipo Point).

La información de Habitantes está almacenada en los mismo elementos DOM que la información de los puntos (de la ubicación). Por lo tanto, para obtener esta información se procederá de la misma manera que la función *obtenerInformacionPuntos* (ver apartado anterior), y también se utilizarán las funciones *obtenerInfoAtributoData* (ver apartado 2.22.) y *obtenerInfoAtributoSimpleData* (ver apartado 2.23.).

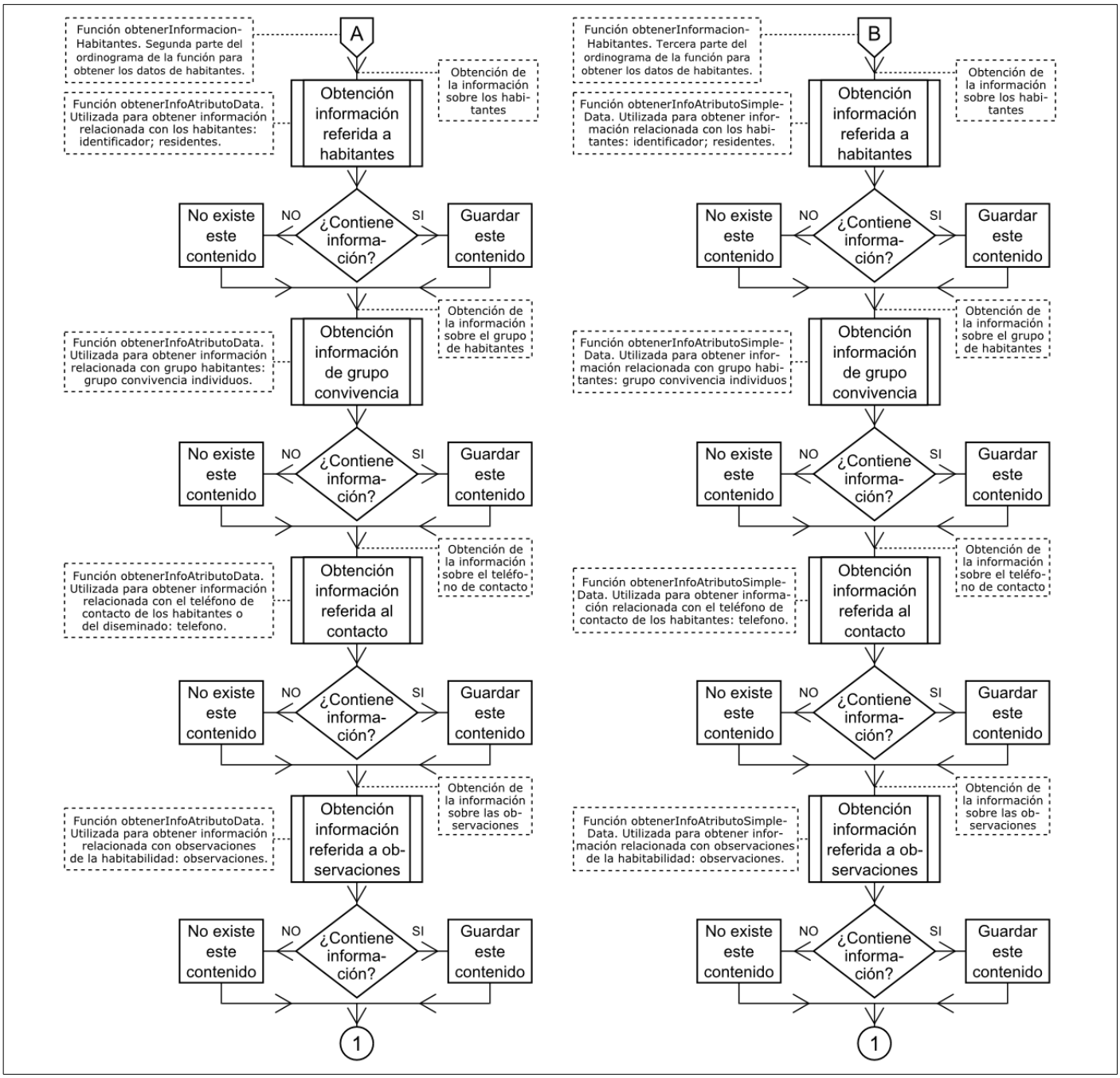

Imagen 118: Ordinograma de la función para obtener la información sobre habitantes

Como se aprecia en este caso se procederá a obtener la siguiente información:

1. Información referida a los habitantes: identificador de los habitantes; nombre de los residentes.
2. Información relacionada con el grupo de habitantes: grupo de convivencia de individuos.
3. Información referida al contacto con los habitantes: teléfono de contacto.
4. Información relacionada con las observaciones de la habitabilidad: observaciones.

Al obtener toda la información contenida en cada punto, se almacenará en una lista, y ésta será el parámetro de salida de la función.

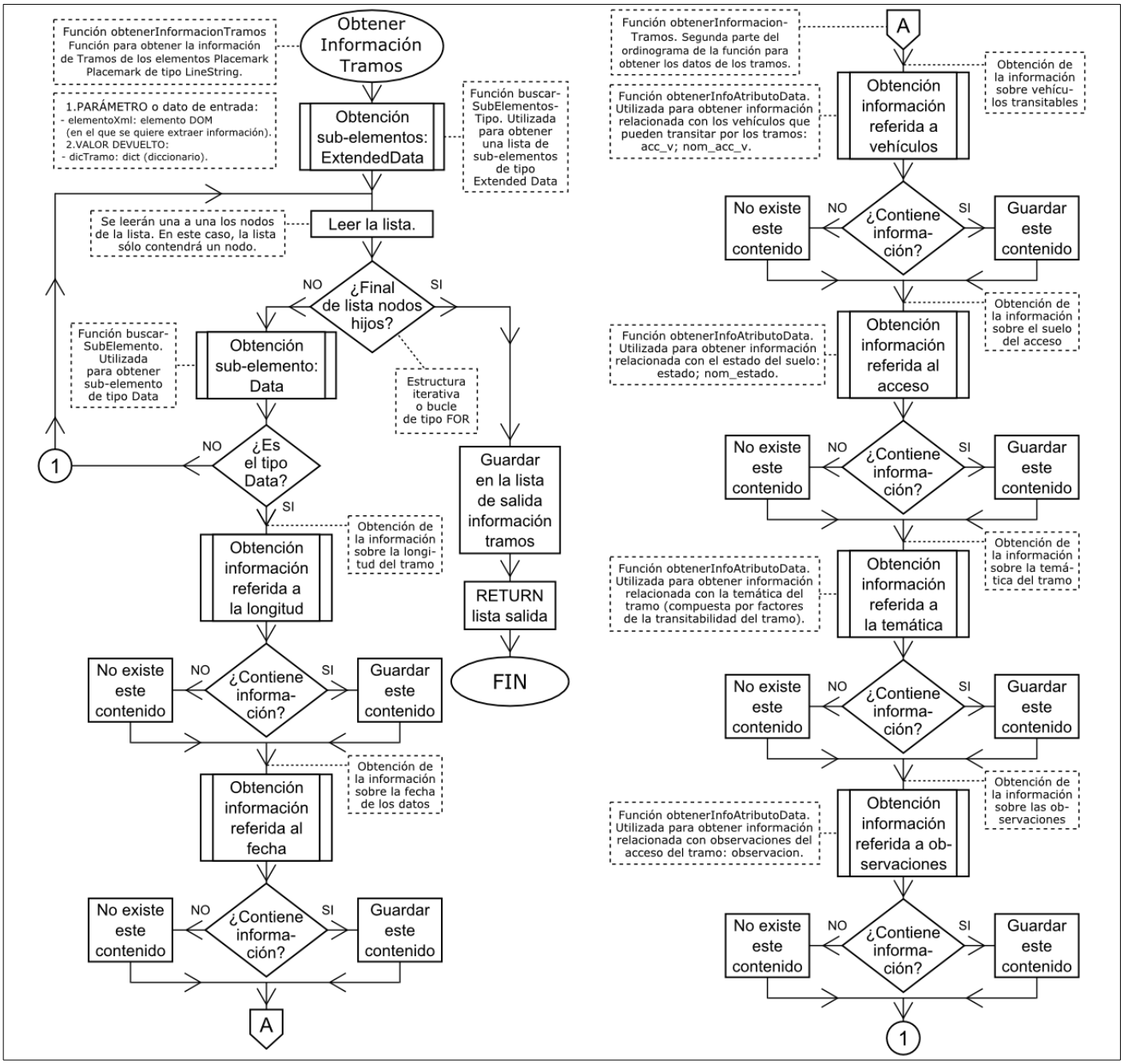

Imagen 119: Ordinograma de la función para obtener la información sobre tramos

2.21. Función obtenerInformacionTramos

Como se aprecia en la Imagen 119 la función *obtenerInformacionTramos* contendrá un parámetro de entrada: un elemento DOM (Placemark de tipo Point).

En este caso, para obtener la información de tramos habrá que acceder al elemento Extended Data y a continuación a sus sub-elementos Data. Cada uno de estos sub-elementos contendrá diferente información, y por ello para obtener cada información se utilizará el valor del atributo name de cada elemento Data. De esta forma, se accederá a cada uno de estos elementos Data para obtener el sub-elemento Value, y posteriormente extraer su contenido. La información obtenida se guardará en su variable correspondiente.

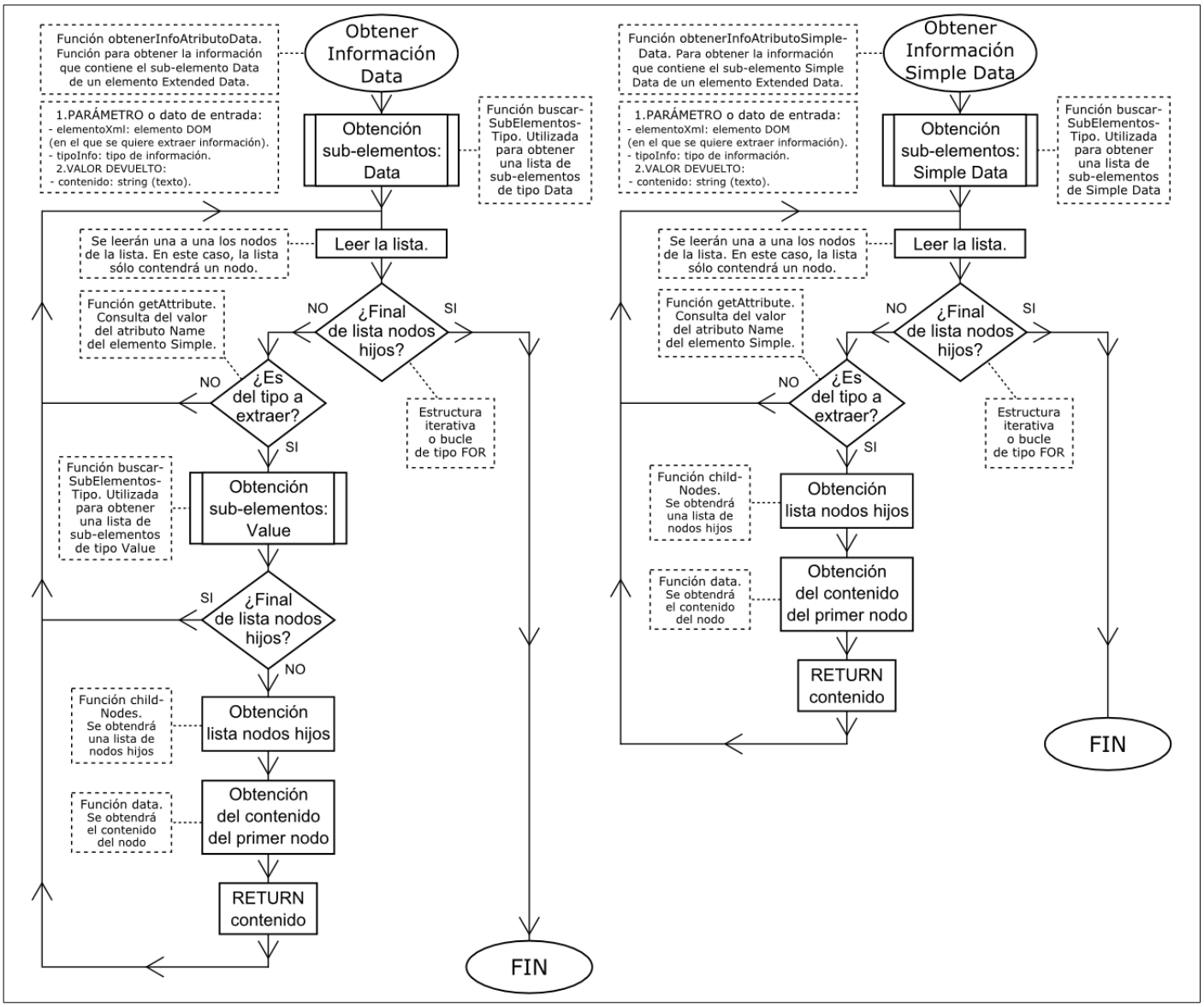

Imagen 120: Ordinograma de las funciones para obtener información de los elementos Data y Simple Data

Para realizar este proceso de extracción se utilizará la función *obtenerInfoAtributoData* (ver apartado 2.22.), y se obtendrá la siguiente información: fecha de la información (fecha); información referida a la distancia a recorrer (longitud); e información relacionada con el acceso y con los vehículos transitables (identificador temático, observaciones; identificador de es-

tado, nombre de estado; identificador de acceso del vehiculo, nombre de acceso del vehiculo).

Al obtener toda la información contenida en cada tramo, se almacenará en una lista, y ésta será el parámetro de salida de la función.

2.22. Función obtenerInfoAtributoData

Como se aprecia en la Imagen 120 la función *obtenerInfoAtributoData* contendrá dos parámetros de entrada: un elemento DOM (de tipo Extended Data) y un tipo de información. Este segundo parámetro será el tipo de contenido que se quiere extraer del elemento DOM (del primer parámetro).

Para obtener la información requerida se consultará el atributo Name del sub-elemento Data (contenido en el elemento DOM especificado). Por lo tanto, al encontrar la información deseada se obtendrá el sub-elemento Value del elemento Data, y se obtendrá su contenido. Esta información será el parámetro de salida.

2.23. Función obtenerInfoAtributoSimpleData.

Como se aprecia en la Imagen 120 la función *obtenerInfoAtributoSimpleData* contendrá dos parámetros de entrada: un elemento DOM (de tipo Schema Data) y un tipo de información. Este segundo parámetro será el contenido que se quiere extraer del elemento DOM (del primer parámetro).

Para obtener la información requerida se consultará el atributo Name del sub-elemento Simple Data (contenido en el elemento DOM especificado). Por lo tanto, al encontrar la información deseada se obtendrá su contenido. Esta información será el parámetro de salida.

A3.2 Modelo de datos intermedio

Al realizar el proceso de carga se obtendrá un modelo de datos intermedio no exactamente igual al modelo relacional (diseñado previamente). Por lo tanto, se realizarán cambios en la base de datos con el objetivo de obtener un modelo de explotación acorde a este modelo de datos relacional.

1. Cambios en el contenido del modelo intermedio

Para minimizar la diferencia existente en el modelo de datos intermedio y del modelo relacional se han realizado cambios estructurales como cambios en el contenido de la base de datos. En este apartado se explicarán alguna de las consultas e instrucciones en SQL utilizadas y aplicadas a la base de datos.

1.1. Destinos duplicados

Al analizar la relación entre los destinos almacenados en las tablas e_destino y e_destinopendiente, se aprecia que todos los diseminados de la tabla e_destinopendiente están repetidos en la tabla e_destino. Como se aprecia en las siguientes instrucciones en SQL se ha analizado estas coincidencias analizando los nombres de los destinos y la distancia entre ellos.

```
SELECT d.iddestino, d.nombredestino, p.iddestino, p.nombredestino
FROM e_destino d, e_destinopendiente p
WHERE lower (d.nombredestino) = lower (p.nombredestino);
```

```
SELECT d.iddestino, d.nombredestino, p.iddestino, p.nombredestino
FROM e_destino d, e_destinopendiente p
WHERE ST_Distance (d.geom, p.geom) < 20;
```

De la misma forma se analizarán las coincidencias que existen dentro de la tabla e_destino.

```
SELECT d.iddestino, d.nombredestino, dd.iddestino, dd.nombredestino
FROM e_destino d, e_destino dd
WHERE (ST_Distance (d.geom, dd.geom) < 20) and (d.iddestino <> dd.iddestino);
```

Como se ha mencionado en la memoria se ha identificado un destino duplicado almacenado en una carpeta llamada Unificado. Por lo tanto, para eliminar los casos de “destinos unificados” se procederá a eliminar todo el contenido de esta carpeta: primero se eliminarán los tramos (tramos rurales y de carretera) y la ruta asociada a este destino, y a continuación se procederá a eliminar el propio destino. Por ejemplo, esta instrucción en SQL sería la indicada para eliminar la ruta:

```
DELETE FROM e_ruta WHERE idruta IN (
        SELECT r.idruta
        FROM e_ruta r
```

```
        WHERE r.iddestino = (
                SELECT d.iddestino
                FROM e_destino d, e_zona z
                WHERE d.idzona = z.idzona AND lower (z.nombrezona) LIKE '% unifica-
    do'));
```

1.2. Información de habitantes

Como se ha mencionado en la memoria, los datos referentes a los/las habitantes están relacionados con los destinos almacenados en la tabla e_destinopendiente. Por lo tanto, al obtener las coincidencias de los destinos de la tabla e_destino y de la tabla e_destinopendiente se procederá a asociar la información de habitantes a la tabla e_destino. Y de esta manera se conformará la tabla r_destinohabitante.

Por ejemplo, de esta forma añadiremos la información al registro iddestino 77 de la tabla e_destino.

```
UPDATE e_destino
SET (idhabitantes, habitado, observacion, contacto1, contacto2) =
        (SELECT p.idhabitantes, p.habitado, p.observacion, p.contacto1, p.contacto2
        FROM e_destino d, e_destinopendiente p
        WHERE ST_Distance (d.geom, p.geom) < 10 AND p.iddestino = 90)
WHERE iddestino = 77;
```

1.3. Vecindad

Como se ha mencionado en la memoria, se creará la tabla r_destinovecindad considerando vecinos/as a los diseminados existentes a una distancia de 500 metros.

```
INSERT INTO r_destinovecindad
        (SELECT h.iddestino iddestino1, j.iddestino iddestino2,
                ST_Distance (h.geom, j.geom) distanciavecindad
        FROM e_destino h, e_destino j
        WHERE h.iddestino <> j.iddestino AND ST_Distance (h.geom, j.geom) < 500
        ORDER BY (h.iddestino,ST_Distance (h.geom, j.geom)));
```

Como se ha mencionado anteriormente, existen diseminados diferentes en un mismo edificio. Estos diseminados contienen diferentes nombres, por lo tanto no se consideran destinos iguales.

1.4. Destinos pendientes

Los destinos contenidos en la tabla e_destinopendiente que no coincidan con los destinos de la tabla e_destino se incluirán en esta última tabla. De esta manera, se han incluido 89 destinos a la tabla e_destino.

En cambio, se han descartado incluir otros tres casos al estar ubicados a menos de un metro de un destino de la tabla e_destino.

1.5. Destinos y rutas pendientes

Se ha analizado si existe alguna relación entre las tablas e_destinopendiente y e_rutapendiente. Pero no se aprecia ninguna relación entre las dos tablas. Por ejemplo, esta posible relación se ha analizado espacialmente:

```
SELECT d.iddestino, idruta, ST_Distance (d.geom, ST_StartPoint (r.geom))
FROM e_destinopendiente d, e_rutapendiente r
WHERE ST_Distance (d.geom, ST_StartPoint (r.geom)) < 15 LIMIT 10;
```

```
SELECT d.iddestino, idruta, ST_Distance (d.geom, ST_EndPoint (r.geom))
FROM e_destinopendiente d, e_rutapendiente r
WHERE ST_Distance (d.geom, ST_EndPoint (r.geom)) < 15 LIMIT 10;
```

1.6. Puntos de salida

Al realizar consultas en SQL se aprecia que el punto de salida de todas las rutas no está exactamente en el Parque de Bomberos de Oronoz-Mugaire.

```
SELECT ST_Distance (s.geom, ST_StartPoint (t.geom)), *
FROM e_salida s, e_tramocarretera t
WHERE ST_Distance (s.geom, ST_StartPoint (t.geom)) < 250;
```

Por lo tanto, se ve la necesidad de definir varios puntos de salida para este servicio de emergencia. Pero al no tener suficiente información sobre el equipamientos que puedan conformar un parque de bomberos no se ha añadido otro registro a la tabla e_salida.

1.7. Tramos de carretera y tramos rurales duplicados

Como se ha mencionado en la memoria, en la base de datos existen tramos de carretera duplicados. Como se aprecia en la siguiente instrucción en SQL se han analizado estas coincidencias analizando el punto de inicio y el final de cada tramo. De esta forma, se han identificado 254 casos.

```
SELECT t.idtramo, c.idtramo
FROM e_tramocarretera t, e_tramocarretera c
WHERE ST_Distance (ST_StartPoint (t.geom), ST_StartPoint (c.geom)) = 0 AND
```

```
        ST_Distance (ST_EndPoint (t.geom), ST_EndPoint (c.geom)) = 0 AND
        t.idtramo <> c.idtramo AND t.idtramo < c.idtramo;
```

También en la base de datos existen tramos rurales duplicados, y de la misma manera se han analizado estas coincidencias. De esta manera, se han identificado 40 casos.

```
SELECT t.idtramo, r.idtramo
FROM e_tramorural t, e_tramorural r
WHERE ST_Distance (ST_StartPoint (t.geom), ST_StartPoint (r.geom)) = 0 AND
        ST_Distance (ST_EndPoint (t.geom), ST_EndPoint (r.geom)) = 0 AND
        t.idtramo <> r.idtramo AND t.idtramo < r.idtramo;
```

Al analizar los tramos duplicados se ha identificado un tramo de carretera mal duplicado. Es el idtramo 90 y este tramo tendría que ser idéntico al idtramo 88. Por ello, se ha corregido este tramo.

```
UPDATE e_tramocarretera SET (geom) =
        (SELECT geom
        FROM e_tramocarretera t
        WHERE t.iddestino = 88)
        WHERE iddestino = 90;
```

En total se han identificado 255 tramos de carretera y 40 tramos rurales duplicados. Antes de eliminar estos tramos duplicados se ha procedido a crear las composiciones de las rutas.

Hay que mencionar que al obtener el modelo de carga las tablas e_tramocarretera y e_tramorural incluyen el campo iddestino. La razón de ello ha sido facilitar las consultas SQL del modelo intermedio y facilitar su transformación. Después de realizar estos cambios se eliminará este campo.

1.8. Composición de las rutas

Como se ha mencionado en la memoria, para analizar la composición de las rutas, primero se ha identificado las diferentes tipologías de unión de los tramos:

1. Caso: unión de Tramo de Carretera y Destino

Hay que mencionar que todos los tramos de carretera se construyen en la trayectoria de la ruta, es decir, el punto inicial (de la polilínea que conforma la ruta) está en el punto de salida y en punto final es el punto más cercano al Destino.

- Unión de punto final (End Point) del tramo de carretera y Destino: 14 casos.

2. Caso: unión de Tramo de carretera y Tramo rural

Hay que mencionar que todos los tramos rurales no se construyen acorde a la trayectoria de la ruta, y por ello se tendrá que analizar diferentes tipo de trayectorias en cada tramo rural.

- Unión del punto final (End Point) del tramo carretera y del punto inicio (Start Point) del tramo rural: 81 casos.
- Unión del punto final (End Point) del tramo de carretera y del punto final (End Point) del tramo rural: 5 casos.

Se observa que la unión del idtramo 56 del tramo carretera está a 10,7 m del siguiente tramo rural.

```
SELECT c.idtramo, r.idtramo,
        ST_Distance (ST_EndPoint (c.geom), ST_StartPoint (r.geom))
FROM e_tramorural r
WHERE ST_Distance (ST_EndPoint (c.geom), ST_StartPoint (r.geom)) < 15 AND
      c.iddestino = r.iddestino;
```

3. Caso: unión de Tramo rural y Tramo rural

Hay que mencionar que en la construcción de las rutas en referencia a los tramos rurales no en todos los casos el identificador idtramo aumenta de número en la trayectoria de la ruta. Es decir, el idtramo de los tramos rurales que componen una ruta puede no aumentar su número progresivamente cuando se acerque al Destino.

- Unión del punto final (End Point) del tramo rural y del punto inicio (Start Point) del tramo rural, aumentando el idtramo: 32 casos.
- Unión del punto final (End Point) del tramo rural y del punto inicio (Start Point) del tramo rural, disminuyendo el idtramo: 10 casos.
- Unión del punto inicio (Start Point) del tramo rural y del punto inicio (Start Point) del tramo rural, disminuyendo el idtramo: 2 casos.
- Unión del punto inicio (Start Point) del tramo rural y del punto final (End Point) del tramo rural, aumentando el idtramo: 1 casos.

Por ejemplo, para realizar la primera consulta se ha utilizado la siguiente instrucción en SQL:

```
SELECT t.idtramo, r.idtramo,
        ST_Distance (ST_EndPoint (t.geom), ST_StartPoint (r.geom))
FROM e_tramorural t, e_tramorural r
WHERE ST_Distance (ST_EndPoint (t.geom), ST_StartPoint (r.geom)) < 10 AND
      t.iddestino = r.iddestino AND t.idtramo <> r.idtramo AND t.idtramo < r.idtramo;
```

4. Caso: unión de Tramo rural y Destino

- Unión del punto final (End Point) del tramo rural y Destino: 84 casos.
- Unión del punto inicio (Start Point) del tramo rural y Destino: 3 casos.

De esta forma, al identificar todas estas relaciones o uniones de tramos se podrá guardar en tablas temporales información relacionada con la composición de las rutas como los puntos

de unión o cruces. Primero se crearán las tablas temporales r_carreteradestino, r_carretera-rural, r_ruralrural y r_ruraldestino, y se realizará de la siguiente forma:

```
CREATE TABLE r_carreterarural AS
SELECT c.idtramo idtramoc, r.idtramo idtramor, r.iddestino,
        ST_StartPoint (r.geom) cruce
FROM e_tramocarretera c, e_tramorural r
WHERE ST_Distance (ST_EndPoint (c.geom), ST_StartPoint (r.geom)) < 15 AND
      c.iddestino = r.iddestino;
```

```
INSERT INTO r_carreterarural
 (SELECT c.idtramo idtramoc, r.idtramo idtramor, c.iddestino,
          ST_EndPoint (c.geom) cruce
 FROM e_tramocarretera c, e_tramorural r
 WHERE ST_Distance (ST_EndPoint (c.geom), ST_EndPoint (r.geom)) < 10 AND
      c.iddestino = r.iddestino);
```

A continuación se creará la tabla temporal r_composicionruta, y se incluirá en ella la información asociada a la composición de rutas. Estos son los casos que se han identificado:

- 14 casos de rutas que sólo contienen el tramo de carretera.
- 86 casos de rutas que contienen un tramo de carretera y un tramo rural.
- 37 casos de rutas que contienen un tramo de carretera y dos tramos rurales.
- 29 casos de rutas que contienen un tramo de carretera y tres tramos rurales.
- Ningún caso de rutas que contienen un tramo de carretera y más de tres tramos rurales.

Las rutas que contienen una idruta igual o menor que 102 son las que contienen sólo tramos de carretera.

Por ejemplo, en el cuarto caso para incluir datos en la tabla temporal se utilizaría la siguiente instrucción:

```
INSERT INTO r_composicionruta
 (SELECT r.idruta, v.iddestino, v.idtramoc, v.idtramor1, v.idtramor2, v.idtramor3
 FROM e_ruta r,
       (SELECT u.iddestino, u.idtramoc, u.idtramor1, u.idtramor2,
               w.idtramor2 idtramor3
        FROM r_ruralrural w,
             (SELECT d.iddestino, d.idtramoc, b.idtramor1, b.idtramor2,
                     NULL idtramor3
```

```
                FROM r_carreterarural d, r_ruralrural b
                WHERE d.iddestino = b.iddestino AND d.idtramor = b.idtramor1) u
            WHERE w.idtramor1 = u.idtramor2) v
        WHERE v.iddestino = r.iddestino AND r.idruta < 102);
```

Por último, en esta tabla r_composicionruta se eliminarán las duplicidades de los tramos obtenidas en el apartado anterior. Por lo tanto, se obtendrán una composición de rutas conformada por tramos no duplicados.

Estos datos de la composición de la ruta se utilizarán para conformar la tabla r_rutatramo. También se utilizará estos datos para completar la tabla e_ruta. La tabla e_ruta contiene el número de tramos que conforma cada ruta.

1.9. Rutas alternativas

Antes de realizar los análisis de la duplicidad de los tramos y de la composición de las rutas se ha identificado dos rutas alternativas incompletas. Concretamente estas dos rutas contienen dos tramos rurales incompletas: tramos idtramo 9 y 10. Por lo tanto, se ha procedido a corregir estos dos tramos: primero se han duplicado dos tramos rurales (de la tabla e_tramorural) que contenga parte del recorrido de los tramos rurales incompletos, y posteriormente se han modificado sus componentes espaciales utilizando un software GIS. Por ejemplo, en el registro del tramo idtramo 9 se ha incorporado el componente espacial del tramo idtramo 31, y después se ha modificado el itinerario utilizando el software QGIS:

```
INSERT INTO e_tramoruralalternativo
        (SELECT 9 idtramo, 29 iddestino, 'Ruta sin título' nombretramo, geom
        FROM e_tramorural
        WHERE idtramo = 31);
```

Como se ha mencionado en la memoria, se ha analizado la duplicidad de los tramos de carretera como los tramos rurales en las rutas alternativas. Este análisis se ha realizado de la misma manera que en las tablas e_tramocarretera y e_tramorural: se ha identificado los casos duplicados y posteriormente se han conformado las composiciones de las rutas.

- Tramos de carretera alternativos: se aprecia que en todos los casos se ha utilizado el mismo tramo de carretera.
- Tramos rurales alternativos: el tramo idtramo 1 está duplicado en los tramos idtramo 3, 5 y 7; y el tramo idtramo 6 está duplicado en el tramo idtramo 8.

Por ejemplo, para identificar los casos de tramos rurales duplicados se han utilizando las siguientes instrucciones en SQL:

```
        SELECT t.idtramo, r.idtramo
        FROM e_tramoruralalternativo t, e_tramoruralalternativo r
```

```
WHERE ST_Distance (ST_StartPoint (t.geom), ST_StartPoint (r.geom)) = 0 AND
      ST_Distance (ST_EndPoint (t.geom), ST_EndPoint (r.geom)) = 0 AND
      t.idtramo <> r.idtramo AND t.idtramo < r.idtramo;
```

- Análisis de las tipologías de unión de los tramos: todas las rutas alternativas contienen tramos rurales.
- Creación de las tablas temporales de composición de rutas alternativas.
- Modificación de estas tablas para que no contengan tramos duplicados.

Después de analizar la composición de las rutas alternativas, toda esta información se añadirá de la misma forma que las rutas normales de los diseminados.

- Incorporación de las rutas alternativas a la tabla e_ruta.
- Incorporación de los tramos de carretera alternativos a la tabla e_tramocarretera.
- Incorporación de los tramos rurales alternativos a la tabla e_tramorural.

Por ejemplo, para incorporar un tramo rural alternativo se han utilizando estas instrucciones en SQL:

```
INSERT INTO e_tramorural
    (SELECT 133 idtramo, 22 iddestino, 102 idcarretera,
        ('TRAMO ALTERNATIVO: ' || t.nombretramo) nombretramo,
        t.geom, NULL longitud, NULL tiempo, NULL vehiculotransitable,
        NULL observacionacceso, NULL fecha, NULL idtematico
    FROM e_tramoruralalternativo t
    WHERE t.idtramo = 1);
```

- Incorporación de la composición de la ruta en la tabla r_rutatramo.

1.10. Obtención de puntos de unión o cruces

Como se ha mencionado en la memoria, para obtener los puntos de unión de los tramos se utilizarán las tablas temporales creadas en la composición de rutas. Más concretamente, se utilizarán las tablas r_carreterarural y r_carreteraruralalternativo.

Por ejemplo, para obtener los cruces de las rutas normales se han utilizado las siguientes instrucciones en SQL:

```
INSERT INTO e_cruce
    (SELECT r.idruta idcruce, r.idruta, c.idtramoc, c.idtramor, NULL fotografiacruce,
        NULL observacion, c.cruce geom
    FROM r_carreterarural c, e_ruta r
    WHERE c.iddestino = r.iddestino AND r.idruta < 102);
```

En esta tabla también se analizará la existencia de cruces duplicadas: se identificarán los casos, se cambiarán las relaciones en la tabla r_cruceruta, y por último se eliminarán estos casos en la tabla e_cruce. Por ejemplo, se podría identificar esta duplicidad de la siguiente forma:

```
SELECT t.idcruce, c.idcruce, ST_Distance (t.geom, c.geom)
FROM e_crucetramo t, e_crucetramo c
WHERE ST_Distance (t.geom, c.geom) > 5 AND
        t.idcruce <> c.idcruce AND t.idcruce < c.idcruce
ORDER BY t.idcruce;
```

1.11. Puntos de información

Como se ha mencionado en la memoria, existe un diseminado que se ha almacenado como un punto de información en vez de un destino. Este error de asignación se producía al existir en una misma carpeta dos destinos. Por lo tanto, este punto identificado incorrectamente se ha añadido a la tabla e_destino. Por ejemplo, para añadir este nuevo destino se ha utilizado las siguientes instrucciones en SQL:

```
INSERT INTO e_destino
 (SELECT 101, d.idzona, (u.nombrelookat || ' ' || d.nombredestino) nombredestino,
        2 tipodestino, d.fotografiadestino, u.geom, d.idportal, d.provincia,
        d.municipio, d.cp, d.codmun, d.idpoblacion, d.nombrebarrio,
        d.nombrecalle, d.idenvia, NULL numeroportal, 'NULL' letraportal,
        'NULL' nombrepoligono, 'NULL' nombreparcela, NULL idinventario,
        False depositogas, False depositogasolina, d.idhabitantes,
        False habitado, 'NULL' observacion, d.contacto1, d.contacto2
 FROM e_destino d, (SELECT l.iddestino, l.nombre, l.geom FROM e_puntoinfo l
                    WHERE l.idpuntoinfo = 1) u
 WHERE d.iddestino = u.iddestino);
```

Posteriormente se modificarán las relaciones de los tramos asociados a este nuevo destino. Para ello, se analizarán los tramos que almacena esta carpeta, y posteriormente cada una de ellas se asignarán a una de las dos rutas. Los tramos que pertenezcan a las dos rutas se duplicarán, y posteriormente se asignarán a cada ruta. Por ejemplo, entre otras instrucciones en SQL se ha utilzado las siguientes operaciones en SQL:

```
INSERT INTO e_tramocarretera
        (SELECT 101, 101, c.nombre, c.idsalida, c.geom, c.longitud, c.tiempo
        FROM e_tramocarretera c
```

```
        WHERE c.iddestino = 75);
```

```
    INSERT INTO e_tramorural
        (SELECT 131, 101, 101, r.nombre, r.geom, r.longitud, r.tiempo,
            r.vehiculotransitable, r.observacionacceso, r.fecha, r.idtematico
        FROM e_tramorural r
        WHERE r.idtramo = 91);
```

Por lo tanto, también se creará una nueva ruta en la tabla e_ruta.

```
    INSERT INTO e_ruta (idruta, idsalida, iddestino) VALUES (101, 1, 101);
```

Por último, se borrará este punto de información incorrecto en la tabla e_puntoinfo.

```
    DELETE FROM e_puntoinfo WHERE iddestino IN (
        SELECT l.iddestino
        FROM e_puntoinfo l WHERE l.puntoinfo = 1);
```

Hay que mencionar que al obtener el modelo de carga la tabla e_puntoinfo incluye el campo iddestino. La razón de ello ha sido facilitar las consultas SQL del modelo intermedio y facilitar su cambio. Después de realizar estos cambios se eliminará este campo y se incluirá idruta.

1.12. Reestructuración de los tramos

Los tramos de carretera y los tramos rurales se almacenarán en una misma tabla. Esta nueva tabla se llamará e_tramo. También se creará una segunda tabla p_tipotramo con la finalidad de diferenciar los registros de tramos de e_tramo (para diferenciar si es un tramo de carretera o un tramo rural).

Por ejemplo, para crear la tabla e_tramo se han utilizando las siguientes instrucciones en SQL:

```
    INSERT INTO e_tramo
        (SELECT c.idtramo, 1 tipotramo, c.nombretramo, c.geom, c.longitud, c.tiempo
        FROM e_tramocarretera c);
```

```
    INSERT INTO e_tramo
        (SELECT r.idtramo + 105, 2 tipotramo, r.nombretramo, r.geom, r.longitud,
            r.tiempo
        FROM e_tramorural r);
```

Al insertar todos los datos en la tabla e-tramo se eliminará la tabla e_tramocarretera. Posteriormente, se modificará la estructura de e_tramorural para que contenga sólo la información referida a las características de la calzada como la transitabilidad de dicha vía.

Por último, se modificará la tabla r_rutatramo utilizando la tabla r_rutatramorural (creada anteriormente en la obtención de la composición de las rutas).

Por ejemplo, se han actualizado alguno de los valores de la tabla r_rutatramo de la siguiente manera:

```
INSERT INTO r_rutatramo
 (SELECT r.idrutatramo + 105 idrutatramo, r.idruta, r.idtramor1 + 105 idtramo,
         2 orden
 FROM r_rutatramorural r
 WHERE r.idtramor1 IS NOT NULL ORDER BY idruta);
```

```
INSERT INTO r_rutatramo
 (SELECT r.idrutatramo + 210 idrutatramo, r.idruta, r.idtramor2 + 105 idtramo,
         3 orden
 FROM r_rutatramorural r
 WHERE r.idtramor1 IS NOT NULL AND r.idtramor2 IS NOT NULL ORDER BY idruta);
```

```
INSERT INTO baztanaldea_temp4.r_rutatramo
 (SELECT r.idrutatramo + 315 idrutatramo, r.idruta, r.idtramor3 + 105 idtramo,
         4 orden
 FROM baztanaldea_temp3.r_rutatramorural r
 WHERE r.idtramor1 IS NOT NULL AND r.idtramor2 IS NOT NULL AND
       r.idtramor3 IS NOT NULL ORDER BY idruta);
```

En la tabla r_rutatramo todos los tramos de carretera tendrán un orden de tramos con el valor 1.

1.13. Clasificación de los datos de vehículo, suelo y estado

Al analizar la información contenida en la tabla e_tramorural (en las columnas vehiculotransitable y observacionacceso) y las tablas p_accv, p_estado y p_tematico se crearán las tablas e_vehiculo, e_suelo y e_estado. Para ello, se analizarán los casos existentes, se realizarán nuevas clasificaciones y se almacenarán estos datos en las nuevas tablas. Por último, se reestructurará la información almacenada en la tabla e_tramorural, y se eliminarán las tablas p_accv, p_estado y p_tematico.

El primer caso será crear la tabla p_tipovehiculo relacionada con la tabla e_vehiculo. Y a continuación, se crearán la propia tabla e_vehiculo y la tabla r_tramovehiculo.

Al analizar la información referida a los tipos de vehículos se ha clasificado de la siguiente manera:

- “A pie”: 1 caso. No se considera un tipo de vehículo.

- Tipo 1, “Vehículo B-741”: 4 casos.
- Tipo 2, “Vehículo B-731”: 65 casos.
- Tipo 3, “Vehículo B-751”: 3 casos.
- Tipo 4, “Vehículo 4x4”: 7 casos.
- Tipo 5, “Camión de monte sin remolque”: 23 casos.
- Tipo 6, “Autobomba forestal”: 2 casos.

El segundo paso será crear la tabla p_tiposuelo relacionada con la la tabla e_suelo. Y a continuación, se crearán la tabla e_suelo y la tabla r_suelotramorural.

- Tipo 1, “todo uno” y “todo uno de granulometría muy fina”: 41 casos.
- Tipo 2, “hormigón”: 38 casos.
- Tipo 3, “asfalto”: 26 casos.
- Tipo 4, “tierra”: 10 casos.
- Tipo 5, “hierba”: 4 casos.

Y por último se creará la tabla p_tipoestado relacionada con la la tabla e_estado. Y a continuación, se crearán la tabla e_estado y se asignará el valor a la columna idestado de la tabla e_tramorural.

- Tipo 1, “mal estado” y “mal estado con”: 4 casos y 2 casos (respectivamente).
- Tipo 2, “estado regular” y “estado regular con”: 6 casos y 1 caso (respectivamente).
- Tipo 3, “buen estado”: 62 casos.

1.14. Otros cambios

Al analizar algunos tramos en un software GIS, se ha identificado dos tramos de carretera parecidos: los tramos idtramo 77 y 78. Aunque contengan una longitud con una diferencia de 60 metros, se han definido como tramos duplicados al contener un recorrido común elevado. Por ello, se ha eliminado el tramo idtramo 78.

A3.3 Modelo de explotación

En este apartado se explican dos aproximaciones principales para el análisis y explotación de los datos almacenados en la base de datos espacial del sistema:

- Consultas espaciales SQL desde QGIS.
- Confección de mapas WMS.
- Acceso a través de servicios WFS.

1. Explotación de datos en QGIS mediante consultas espaciales

Los ejemplos desarrollados en este apartado se han realizado con el software QGIS. Esta herramienta es idónea para visualizar y editar el contenido de una base datos espacial PostGIS.

En los ejemplos expuestos a continuación se ha utilizado la herramienta de Administración de Base de Datos contenida en el software QGIS. Con ella se han realizado diferentes consultas espaciales SQL, y se han creado tablas y vistas en la base de datos, y posteriormente se han visualizado en este entorno GIS.

1.1. Ejemplo 1: acceso a las rutas utilizando el nombre de los diseminados

En este ejemplo primero se analizará la composición de las rutas y la relación que contienen las rutas con los tramos. A continuación, se conseguirá el componente espacial de cada ruta. Para ello, se agruparán las geometrías de los tramos que componen cada ruta. Por último, se representarán estas rutas en el mapa.

Análisis de la composición de una ruta:

En este primer paso se combinarán tres tablas para obtener la información que contiene una ruta y los tramos que lo componen. El resultado de esta consulta SQL contendrá un componente espacial.

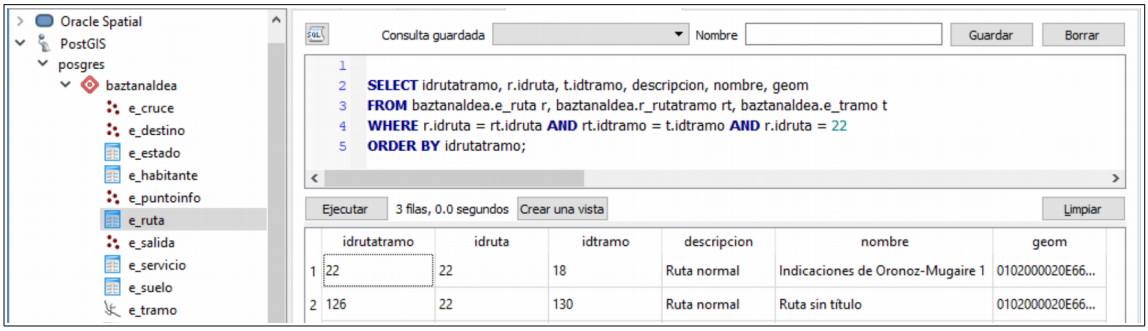

Imagen 121: Consulta SQL realizada para obtener la información de una ruta y los tramos que lo componen

Como se aprecia en la Imagen 121 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener toda la información de los tramos que contiene esta ruta:

```
SELECT idrutatramo, r.idruta, t.idtramo, descripcion, nombre, geom

FROM baztanaldea.e_ruta r, baztanaldea.r_rutatramo rt, baztanaldea.e_tramo t
```

```
WHERE r.idruta = rt.idruta AND rt.idtramo = t.idtramo AND r.idruta = 22

ORDER BY idrutatramo;
```

Análisis de la relación entre rutas y tramos:

En este segundo paso se analizará la relación existente entre las rutas y los tramos. Primero se obtendrá la cantidad de tramos que contiene cada ruta, y a continuación se calculará la cantidad de rutas que pertenece cada tramo.

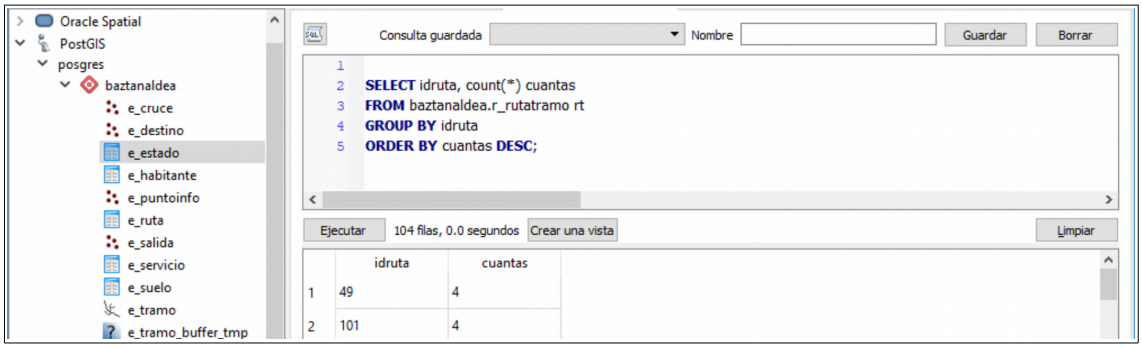

Imagen 122: Consulta SQL realizada para obtener la cantidad de tramos que contiene cada ruta

Como se aprecia en la Imagen 122 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener la cantidad de tramos que contiene cada ruta:

```
SELECT rt.idruta, count(*) cuantas

FROM baztanaldea.r_rutatramo rt

GROUP BY rt.idruta

ORDER BY cuantas DESC;
```

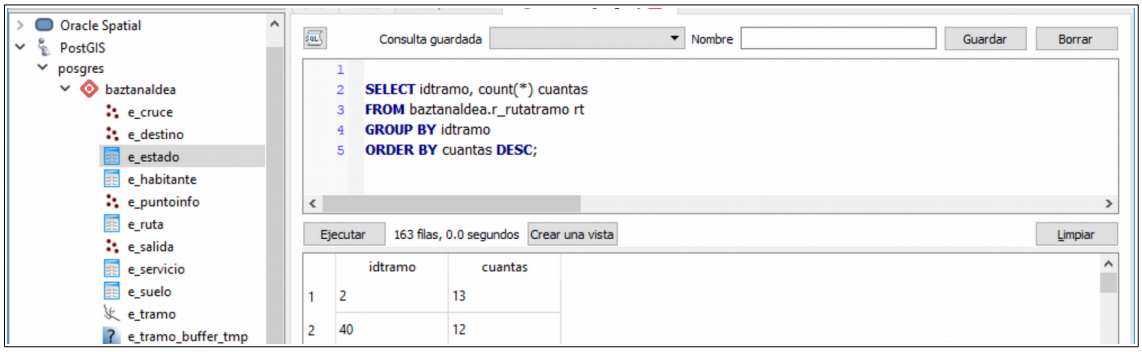

Imagen 123: Consulta SQL realizada para obtener la cantidad de rutas que pertenece cada tramo

Como se aprecia en la Imagen 123 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener la cantidad de rutas que pertenece cada tramo:

```
SELECT rt.idtramo, count(*) cuantas

FROM baztanaldea.r_rutatramo rt
```

```
    GROUP BY rt.idtramo

    ORDER BY cuantas DESC;
```

Agrupación de la geometría de los tramos por cada ruta

En este tercer paso se analizará combinarán cuatro tablas de la base de datos para obtener el resultado relacional que contenga por cada ruta la información espacial de los tramos y los nombres de los destinos. Por lo tanto, se podrá visualizar una ruta en el mapa partiendo del nombre un diseminado.

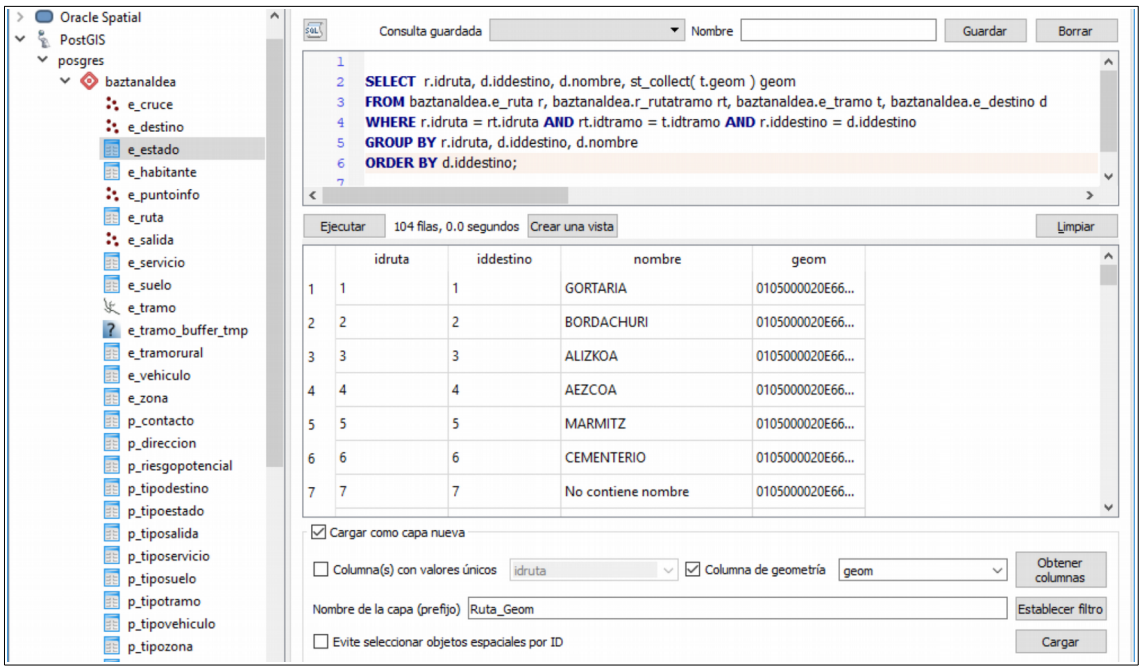

Imagen 124: Consulta realizada para obtener información de tramos y del destino que contiene cada ruta

Como se aprecia en la Imagen 124 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener la información de los tramos y del destino que contiene cada ruta:

```
    CREATE OR REPLACE VIEW baztanaldea.Ruta_Destino_geom AS

     SELECT  r.idruta, d.iddestino, d.nombre, st_collect ( t.geom ) geom

     FROM baztanaldea.e_ruta r, baztanaldea.r_rutatramo rt, baztanaldea.e_tramo t,

                    baztanaldea.e_destino d

     WHERE r.idruta = rt.idruta AND rt.idtramo = t.idtramo AND r.iddestino = d.iddestino

     GROUP BY r.idruta, d.iddestino, d.nombre

     ORDER BY d.iddestino;
```

En este caso, con el resultado de esta consulta se ha creado una vista en SQL, y a continuación se ha añadido al mapa de QGIS (como se puede observar en la Imagen 125).

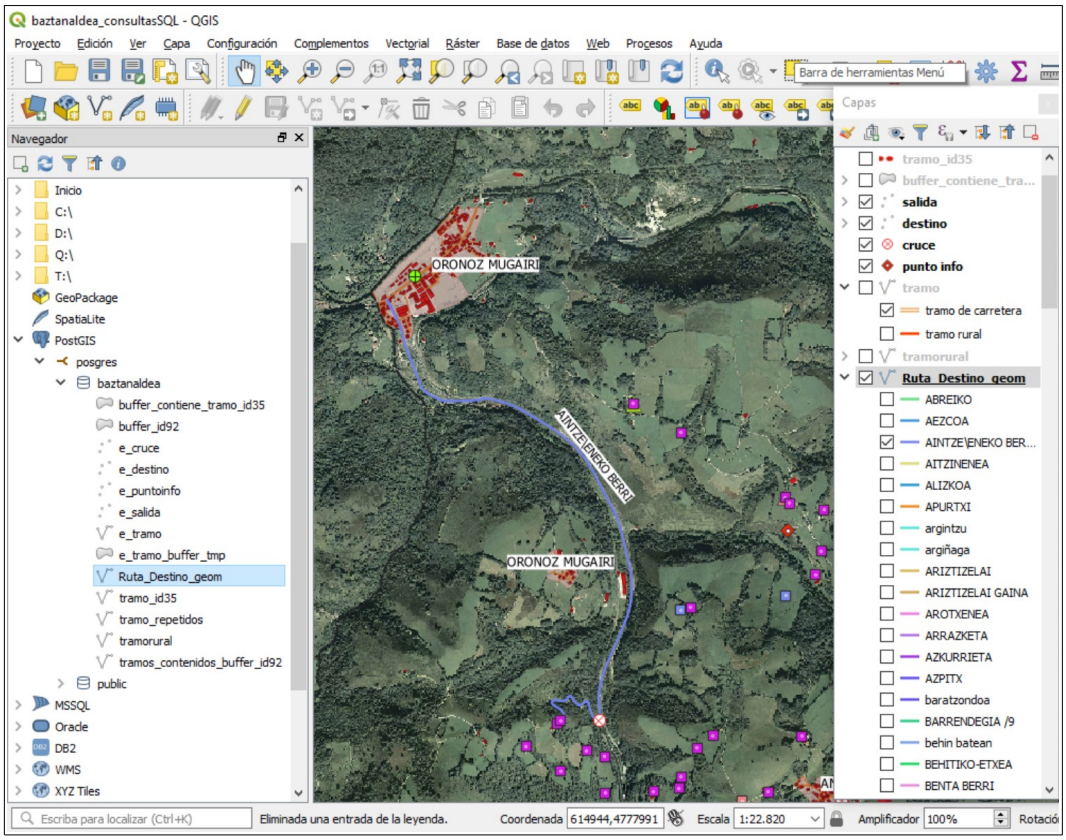

Imagen 125: Visualización de las rutas a partir del nombre del destino que contiene cada ruta

Como se aprecia en este ejemplo las geometrías de tipo LinseString de los tramos se han agrupado o recolectado utilizando la función espacial ST_Collect de PostGIS. El resultado de esta función serán una geometría de tipo MultiLineString, y por lo tanto se podrá visualizar en el mapa.

1.2. Ejemplo 2: identificación de tramos con itinerarios parcialmente repetidos

En este ejemplo se analizará los tramos que contienen itinerarios parcialmente repetidos con otros tramos. Para ello, se desarrollarán dos ejemplos donde se apreciará visualmente la existencia de datos redundantes en la base de datos (en referencia al componente espacial de los tramos) .

Creación de una tabla con un buffer de los tramos

En este primer paso se creará una tabla temporal para agilizar los cálculos internos del software QGIS. Es decir, al realizar una consulta espacial en SQL con varias funciones espaciales el software requiere mucho tiempo para realizar todos los cálculos y operaciones que exige una consulta SQL de este tipo. Por lo tanto, se creará esta tabla temporal para agilizar este proceso de cálculos. Esta tabla contendrá toda la información de los tramos pero con una geometría diferente. Contendrá como componente espacial un buffer de 5 metros de la polilínea del tramo a que pertenece. A esta tabla se le añadirán una clave primaria y un índice espacial.

La razón de añadir un índice espacial a la tabla será el de agilizar los procesos de cálculo espaciales. Con la misma finalidad, también se añadirán este tipo de índices a todas las tablas

de la base de datos que contengan un componente espacial. De esta manera, al realizar consultas espaciales complejas se facilitará la obtención del resultado.

Imagen 126: Consulta realizada para obtener la tabla del buffer de los tramos

Como se aprecia en la Imagen 126 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener la nueva tabla del buffer de los tramos:

```
CREATE TABLE baztanaldea.e_tramo_buffer_tmp AS
        SELECT idtramo, ST_Buffer (geom, 5) geom
        FROM baztanaldea.e_tramo;
```

```
ALTER TABLE ONLY baztanaldea.e_tramo_buffer_tmp
        ADD CONSTRAINT tramo_buffer_tmp_pk PRIMARY KEY (idruta);
```

```
CREATE INDEX e_tramo_tmp_geom_gix
        ON baztanaldea.e_tramo_buffer_tmp
        USING gist (geom);
```

Identificación de tramos que contienen parte del itinerario común

En este segundo paso se identificarán los casos donde las geometrías de los tramos están enteramente dentro de la geometría de la tabla buffer creada temporalmente. Es decir, se identificarán los casos de los tramos que conforman una polilínea acorde a una parte de la polilínea de otro tramo.

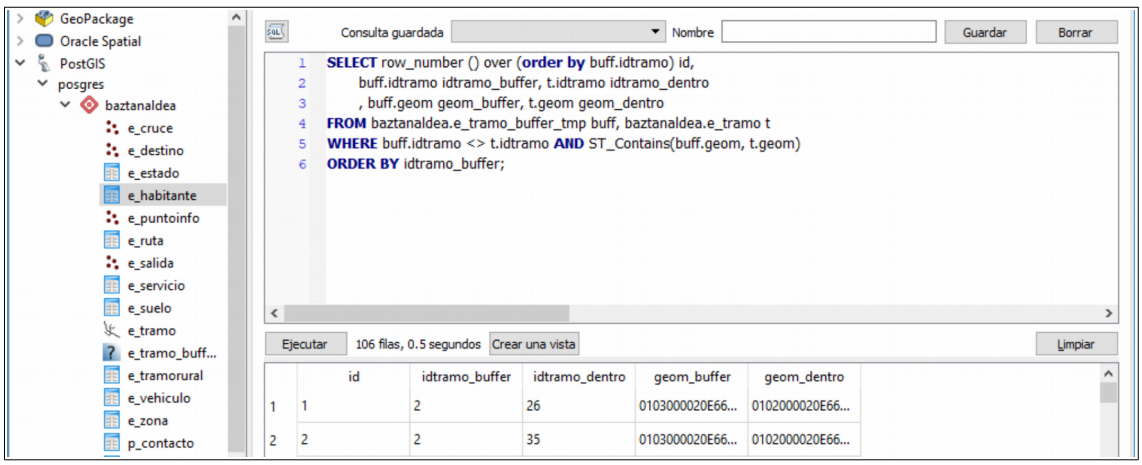

Imagen 127: Consulta realizada para obtener tramos que contienen parte del itinerario común

Para definir la condición geométrica de la combinación de estas dos tablas se ha utilizado la función espacial ST_Contains (geometry g1, geometry g2) de PostGIS. Esta función devolverá verdadero (True) en los casos donde la geometría de los tramos estén espacialmente dentro de la geometría del buffer de los tramos.

Como se aprecia en la Imagen 127 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener la lista de tramos que cumplen esta condición:

Tramoen bufferren barruan zein tramo sartzen diren ikusten da.

```
SELECT row_number () over (order by buff.idtramo) id,
        buff.idtramo idtramo_buffer, t.idtramo idtramo_dentro,
        buff.geom geom_buffer, t.geom geom_dentro
FROM baztanaldea.e_tramo_buffer_tmp buff, baztanaldea.e_tramo t
WHERE buff.idtramo <> t.idtramo AND ST_Contains (buff.geom, t.geom)
        ORDER BY idtramo_buffer;
```

Como se puede apreciar en esta consulta se ha utilizado la función row_number ( ). Esta función se utilizará para crear un número secuencial en una columna de la tabla resultante de la consulta. Esta columna servirá como identificador de cada registro de la tabla, y será indispensable para incorporar al mapa del software QGIS.

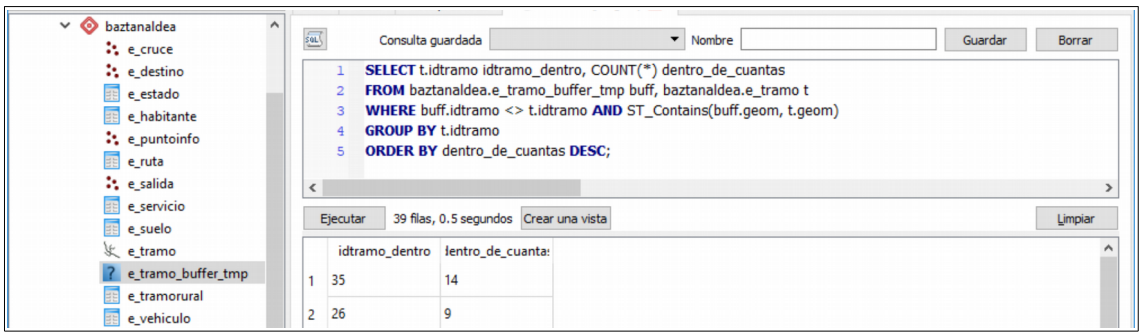

Imagen 128: Consulta realizada para obtener la cantidad de buffers que contienen la geometría del tramo

Análisis de los casos identificados que cumplen la condición geométrica

En este tercer paso se analizará la relación existente entre los tramos y el buffer de los tramos. Primero se obtendrá para cada tramo la cantidad de buffers que enteramente contienen la geometría del tramo, y en segundo lugar se calculará para cada buffer la cantidad de tramos que están enteramente dentro de la geometría del buffer.

Como se aprecia en la Imagen 128 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener la cantidad de buffers que enteramente contienen la geometría del tramo.

```
SELECT t.idtramo idtramo_dentro, COUNT(*) dentro_de_cuantas
```

```
FROM baztanaldea.e_tramo_buffer_tmp buff, baztanaldea.e_tramo t

WHERE buff.idtramo <> t.idtramo AND ST_Contains (buff.geom, t.geom)

GROUP BY t.idtramo

ORDER BY dentro_de_cuantas DESC;
```

Por ejemplo, la polilínea del tramo con el identificador idtramo 35 está dentro de 14 buffers de tramos. Esto significa que la geometría de este tramo está repetida en 14 tramos.

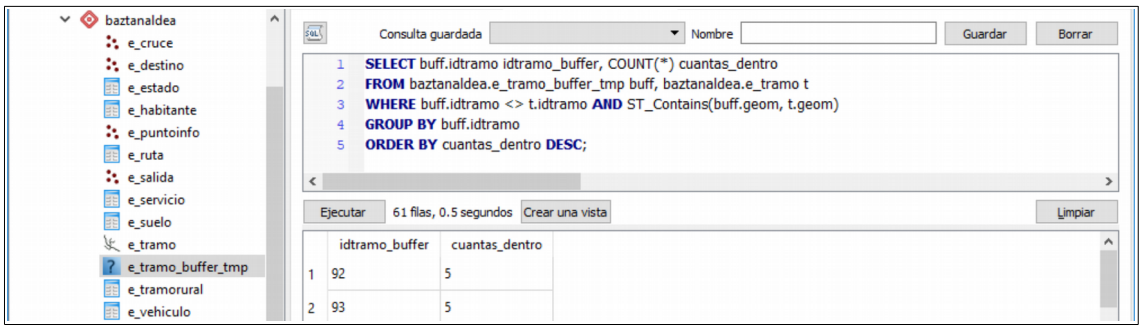

Imagen 129: Consulta realizada para obtener cantidad de tramos que están dentro de geometría del buffer

Como se aprecia en la Imagen 129 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener la cantidad de tramos que están enteramente dentro de la geometría del buffer.

```
SELECT buff.idtramo idtramo_buffer, COUNT(*) cuantas_dentro

FROM baztanaldea.e_tramo_buffer_tmp buff, baztanaldea.e_tramo t

WHERE buff.idtramo <> t.idtramo AND ST_Contains (buff.geom, t.geom)

GROUP BY buff.idtramo

ORDER BY cuantas_dentro DESC;
```

Por ejemplo, el buffer del tramo con el identificador idtramo 92 contiene 5 tramos. Esto significa que este tramo (del que se ha creado un buffer de cinco metros) contiene la geometría repetida de 5 tramos.

Visualización de los casos identificados

En este cuarto paso se visualizarán alguno de los casos que se han identificado en el paso anterior. Más concretamente se visualizarán dos casos: primero se creará por cada caso una vista en la base de datos, y a continuación se añadirá al mapa del software QGIS.

1. **Caso: los buffers que contiene el tramo idtramo 35**

Como se aprecia en la Imagen 130 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener los buffers que enteramente contiene la geometría del tramo con el iden-

tificador idtramo 35. Este resultado con 14 registros de buffers se añadirán a una vista creada en la base de datos.

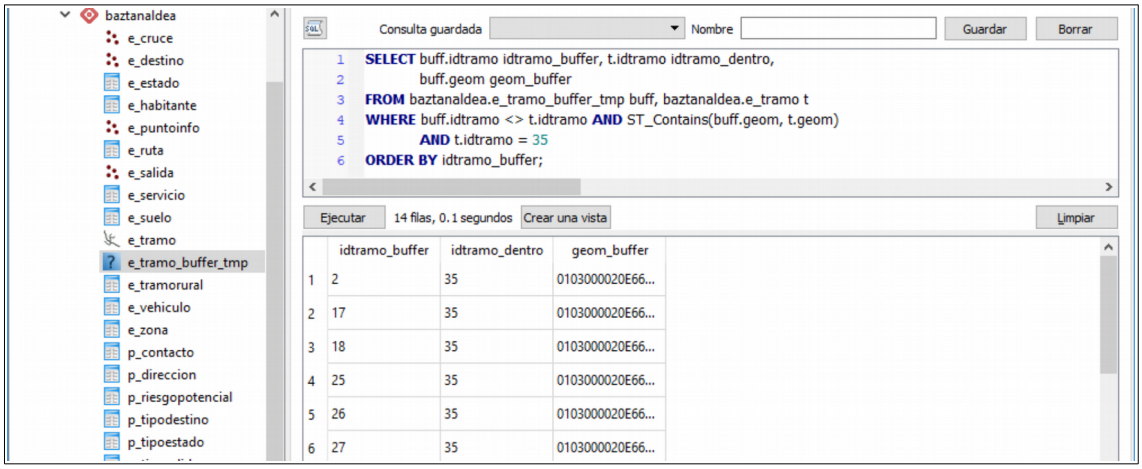

Imagen 130: Consulta realizada para obtener los buffers que contienen la geometría del tramo idtramo 35

```
CREATE OR REPLACE VIEW baztanaldea.v_buffer_contiene_tramo_id35 AS
 SELECT buff.idtramo idtramo_buffer, t.idtramo idtramo_dentro,
        buff.geom geom_buffer
 FROM baztanaldea.e_tramo_buffer_tmp buff, baztanaldea.e_tramo t
 WHERE buff.idtramo <> t.idtramo AND ST_Contains (buff.geom, t.geom)
          AND t.idtramo = 35
 ORDER BY idtramo_buffer;
```

Como se aprecia en la Imagen 131 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener el tramo con el identificador idtramo 35.

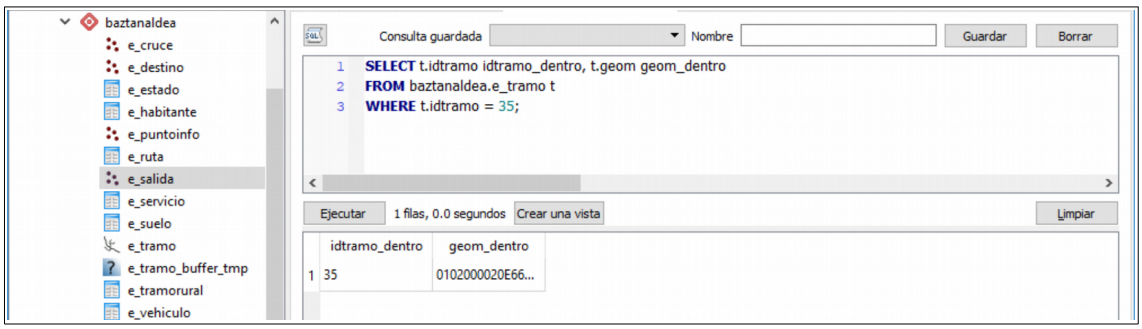

Imagen 131: Consulta realizada para obtener el tramo idtramo 35

```
CREATE OR REPLACE VIEW baztanaldea.tramo_id35 AS
        SELECT t.idtramo idtramo_dentro, t.geom geom_dentro
        FROM baztanaldea.e_tramo t
```

```
        WHERE t.idtramo = 35;
```

A continuación, estas dos vistas creadas se añadirán al mapa de QGIS. Como se puede observar en la Imagen 132, se activará y desactivará la visualización de cada buffer para observar que el tramo idtramo 35 está enteramente dentro ellas.

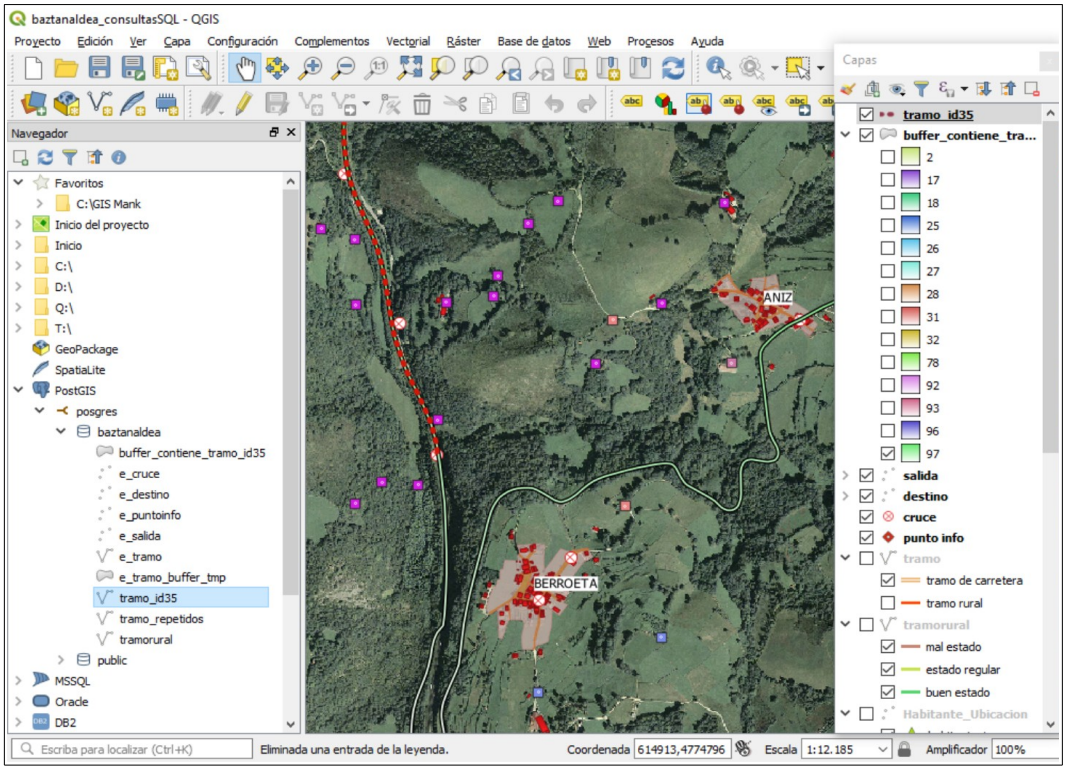

Imagen 132: Visualización de los buffers que contienen la geometría del tramo idtramo 35

2. Caso: los tramos que están dentro del buffer idtramo 92

Como se aprecia en la Imagen 133 se ha realizado esta consulta SQL en el Administrador de BBDD para obtener los tramos que están enteramente dentro de la geometría del buffer con el identificador idtramo 92. Este resultado con 5 registros de tramos se añadirán a una vista creada en la base de datos.

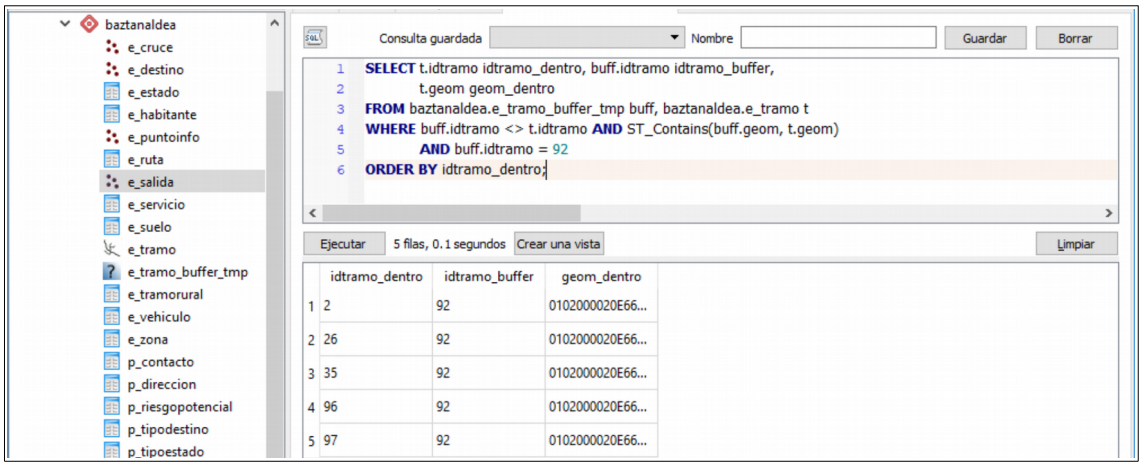

Imagen 133: Consulta realizada para obtener los tramos que están dentro de la geometría del buffer idtramo 92

```
CREATE OR REPLACE VIEW baztanaldea.tramos_contenidos_buffer_id92 AS
 SELECT t.idtramo idtramo_dentro, buff.idtramo idtramo_buffer,
        t.geom geom_dentro
 FROM baztanaldea.e_tramo_buffer_tmp buff, baztanaldea.e_tramo t
 WHERE buff.idtramo <> t.idtramo AND ST_Contains (buff.geom, t.geom)
          AND buff.idtramo = 92
 ORDER BY idtramo_dentro;
```

Como se aprecia en la Imagen 134 se ha realizado esta consulta SQL en el Administrador de BBD para obtener el buffer con el identificador idtramo 92.

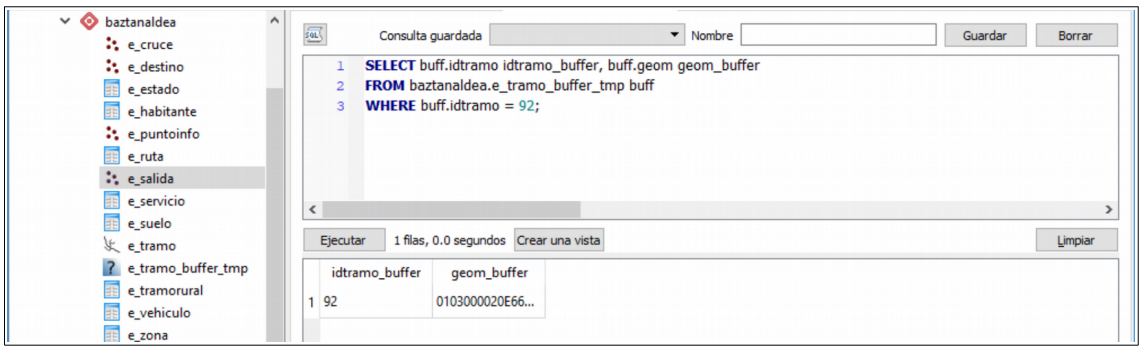

Imagen 134: Consulta realizada para obtener el tramo idtramo 92

```
CREATE OR REPLACE VIEW baztanaldea.buffer_id92 AS
        SELECT buff.idtramo idtramo_buffer, buff.geom geom_buffer
        FROM baztanaldea.e_tramo_buffer_tmp buff
        WHERE buff.idtramo = 92;
```

A continuación, estas dos vistas creadas se añadirán al mapa de QGIS. Como se puede observar en la Imagen 135, se activará y desactivará la visualización de cada tramo para observar que el tramo idtramo 92 contiene cada uno de estos tramos.

2. Explotación de datos en Geoserver (WMS y WFS)

Los ejemplos desarrollados en este apartado se han realizado con el servidor Geoserver. Es decir, se ha utilizado este software para visualizar información geográfica de la base de datos.

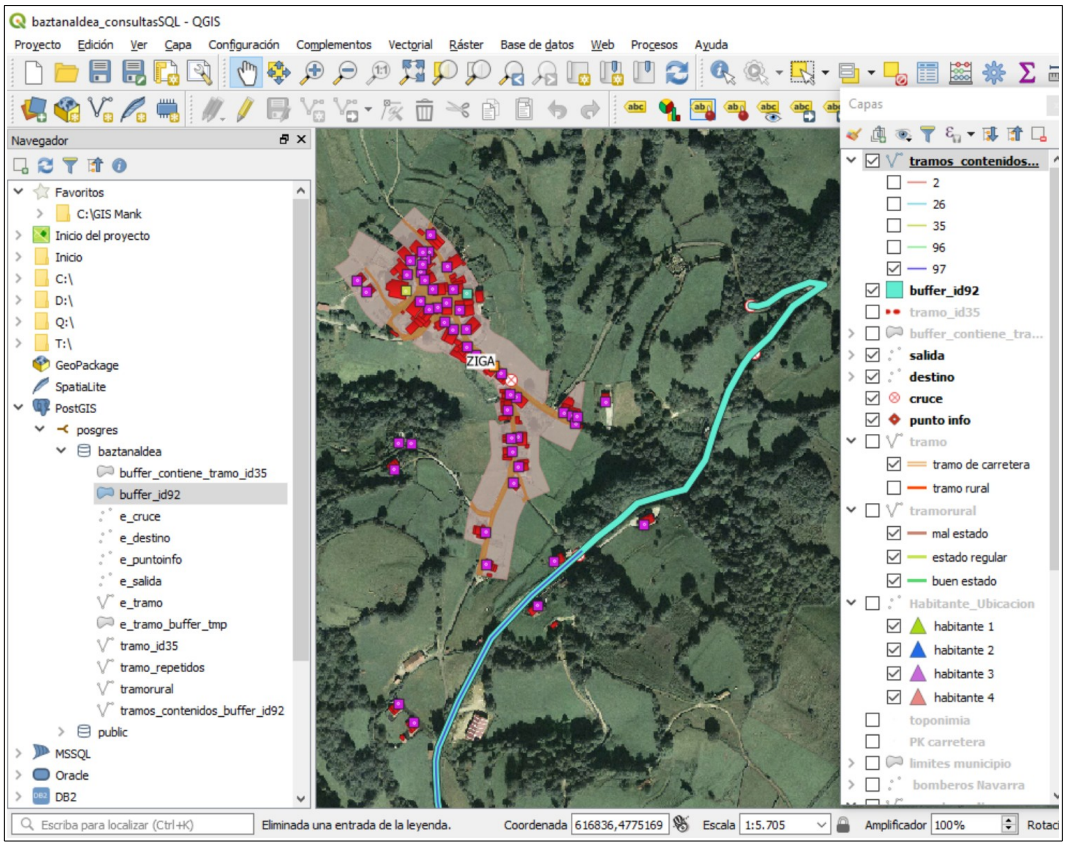

Imagen 135: Visualización de los tramos que están dentro de la geometría del buffer idtramo 92

2.1. Ejemplo 3: asignación de diferentes pesos a tramos

En este ejemplo se asignará diferentes pesos a los tramos dependiendo de los vehículos que pueden transitar por ella. Por lo tanto, se pretende clasificar los tramos que contengan menos dificultades para transitar.

Asignación de los pesos según el vehículo

En este primer paso se ha creado un tabla temporal donde se guardarán los tramos con un peso determinado. La asignación de pesos se realizará de la siguiente forma: contendrá peso 1 los tramos que indiquen que hay que transitar con un vehículo 4X4; contendrá peso 2 los tramos que recomiendan transitar con los vehículos B-741, B-731 y B-751; el peso 3 se le asignará a los tramos que pueden transitar camiones de monte sin remolque; y el peso 4 se le asignará a los que pueden transitar con una autobomba forestal.

Como se puede apreciar en el siguiente código SQL se ha utilizado el Ejecutor de consultas SQL de PgAdmin III para crear una tabla temporal que contenga una lista de tramos de carretera con diferentes pesos.

```
CREATE TABLE baztanaldea.e_tramo_peso_tmp AS

    SELECT t.idtramo, t.geom, 1 peso

    FROM baztanaldea.e_tramo t, baztanaldea.r_tramovehiculo tv,
```

```
            baztanaldea.e_vehiculo v, baztanaldea.p_tipovehiculo p
        WHERE t.idtramo = tv.idtramo AND tv.idvehiculo = v.idvehiculo
            AND v.idtipovehiculo = p.idtipovehiculo
            AND p.descripciontipo::text ~~ '%4X4%'::text;
```

```
INSERT INTO baztanaldea.e_tramo_peso_tmp
    (SELECT t.idtramo, t.geom, 4 peso
    FROM baztanaldea.e_tramo t, baztanaldea.r_tramovehiculo tv,
        baztanaldea.e_vehiculo v, baztanaldea.p_tipovehiculo p
    WHERE t.idtramo = tv.idtramo AND tv.idvehiculo = v.idvehiculo
        AND v.idtipovehiculo = p.idtipovehiculo
        AND p.descripciontipo::text ~~ '%autobomba%'::text);
```

Suma de los pesos para obtener los tramos con diferente transitabilidad

Como se aprecia en la Imagen 136 se ha utilizado el Ejecutor de consultas SQL de PgAdmin III para crear una vista SQL con la lista de tramos de rurales con la suma de pesos de transitabilidad.

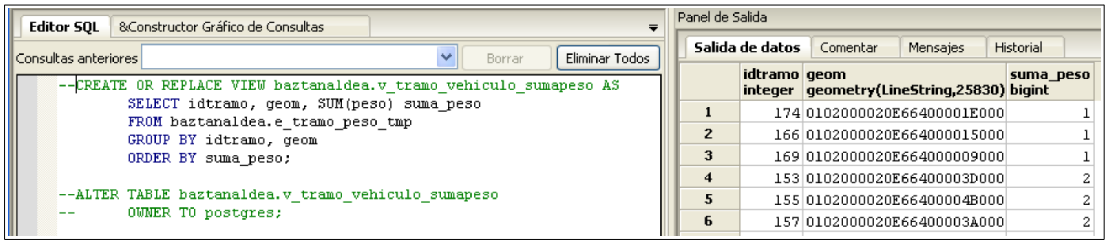

Imagen 136: Consulta realizada para obtener los tramos de carretera importantes

```
CREATE OR REPLACE VIEW baztanaldea.v_tramo_vehiculo_sumapeso AS
    SELECT idtramo, geom, SUM (peso) suma_peso
    FROM baztanaldea.e_tramo_peso_tmp
    GROUP BY idtramo, geom
    ORDER BY suma_peso;
```

A continuación, en el servidor Geoserver se creará un almacén de datos y una capa con esta información. Como se puede observar en la Imagen 137 se ha especificado un estilo SLD para esta capa.

Finalmente se ha añadido como un servicio de WMS al mapa del software QGIS. Como se puede observar en la Imagen 138, esta vista SQL también se ha añadido directamente de la base de datos (para ver la diferencia de visualización).

Imagen 137: Asignación del estilo a la capa de servicios WMS de tramos con pesos de transitabilidad

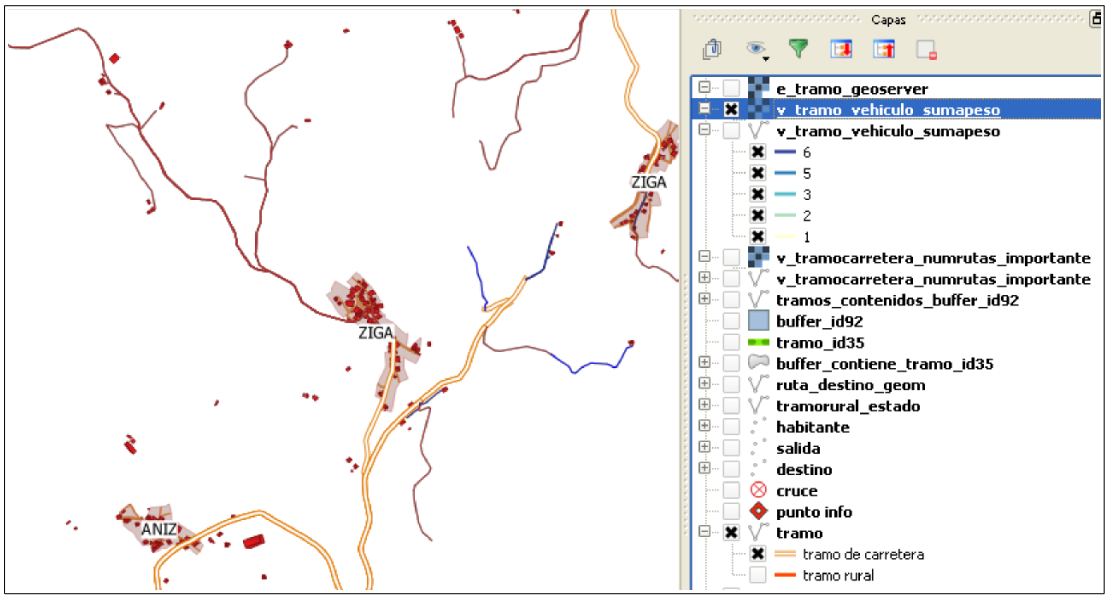

Imagen 138: Visualización de los tramos con pesos de transitabilidad en el mapa del software QGIS