UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA GEODÉSICA, CARTOGRÁFICA Y TOPOGRAFICA

TRABAJO FIN DE GRADO

Localización y enrutamiento de recursos turísticos y otros puntos de interés del municipio de Santa Eulària des Riu.

Autor: Alejandro de Fuenmayor Satorre

Tutor: José Carlos Martínez Llario

Valencia, Septiembre 2016

ÍNDICE

INDICE DE FIGURAS

INDICE DE TABLAS

LISTADO DE ABREVIATURAS

INE: Instituto Nacional de Estadística.

SGBDR: Sistema de Gestión de Bases de Datos.

GIS: Geopraphic Information System.

SQL: Structured Query Language.

CRUD: Create, Read, Update and Delete.

ACID: Atomicidad, Consistencia, Aislamiento y Durabilidad.

GDB: Bases de Datos basadas en Grafos.

OGC: Open Geospatial Consortium.

SHP: Shapefile.

DXF: Drawing Exchange Format.

SIG: Sistema de Información Geografico.

WMS: Web Map Service.

WFS: Web Feature Service.

HTML: Hyper Text Markup Language

SLP: SpectraLayers Pro

API: Application Programing Interface.

DNG: Digital Negative.

EPSG 25831: Proyección UTM ETRS89 Huso 31 N.

EPSG 4258: Coordenadas Elipsoidales ETRS89 IDEE.

ETRS89: European Terrestrial Reference System 1989.

IDEE: Infraestructura de Datos Espaciales de España.

IGN: Instituto Geográfico Nacional.

CNIG: Centro Nacional de Información Geográfica.

INSPIRE: Insfraestructure for Spatial Information in Europe.

DGC: Dirección General del Catastro.

BTN: Base Topográfica Nacinal.

PDF: Portable Document Format.

URL: Uniform Resource Locator.

RESUMEN

El contenido de este trabajo trata sobre el diseño y desarrollo de una red de caminos y carreteras para permitir el cálculo de rutas del municipio de Santa Eulària des Riu a través de internet, a través de un visualizador web.

Se ha descargado información cartográfica creada por organismos oficiales españoles, como son el Instituto Geográfico Nacional, se ha tratado para obtener una red de caminos y carreteras ajustada al municipio y se ha alojado en PostgreSQL, un sistema de gestión de base de datos relacional, en la cual se han terminado de realizar los ajustes necesarios para obtener una red funcional y adaptada para pgRouting, es una extensión que añade ruteo y otras funcionalidades de análisis de redes a bases de datos PostGIS/PostgreSQL.

Una vez obtenida la red funcional se ha cargado en Geoserver, un servidor web utilizado para publicar la cartografía y permitir acceder a ella a través de los servicios web establecidos por el Open Geospatial Consortium.

Se ha diseñado y programado una página web que incluye un visualizador web interactivo en el cual los usuarios pueden seleccionar un punto de origen y destino de la ruta que desean calcular. En este visor web el usuario podrá visualizar una ortofoto, el límite del municipio, los recursos turísticos y la ruta calculada.

Todo se ha realizado utilizando únicamente software libre y gratuito.

1. JUSTIFICACIÓN Y OBJETIVOS

El objetivo de este proyecto es proporcionar a los vecinos, visitantes y personales interesadas en Santa Eulària des Riu una herramienta con acceso vía internet que permita visualizar la ruta entre dos puntos seleccionados por ellos. Se obtendrá cartografía del municipio y se creará un visor web, en el cual se mostrará la ortofoto del Plan Nacional de Ortofoto Aérea del año 2015, el límite municipal, las rutas de consulta del usuario, el cual obtendrá dos resultados, siendo la ruta más rápida y la ruta más corta y los nombres de las distintas playas o calas, núcleos urbanos, puntas o cabos (estos suelen ser miradores), museos y construcciones antiguas o de interés, como torres de vigía, molinos, pozos e iglesias.

Actualmente existen recursos vía internet que proporcionan ya lo que busca obtener este proyecto, como por ejemplo Google Maps, pero para el caso concreto del municipio de Santa Eulària des Riu estos visores web no contienen todos los caminos que existen actualmente en el municipio y esto provoca que no se pueda mostrar una ruta continua hasta el punto seleccionado como destino por el usuario, dado que muchas zonas de la isla solo son conocidas por sus habitantes locales. También cabe destacar que hoy en día no existe nada parecido en este municipio desarrollado por el ayuntamiento, ya que hace relativamente unos años no existía ningún visor web proporcionado por esta administración pública.

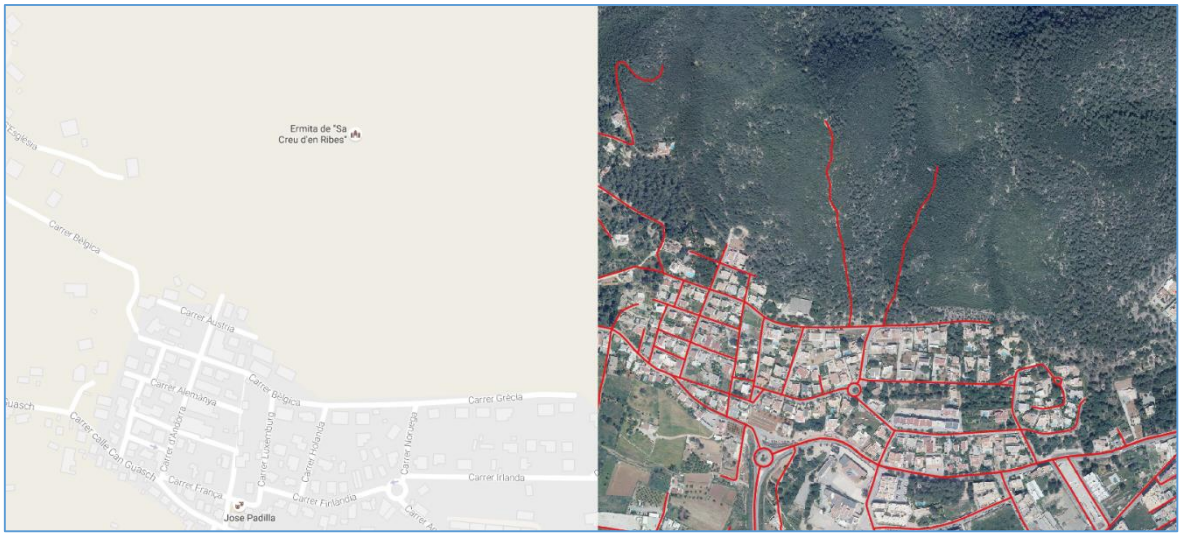


Fig. 1: Imagen comparativa de caminos entre google y mi red de rutas.

Se pretende también la utilización de las versiones más modernas de los softwares comúnmente utilizados en este tipo de aplicaciones, como son las nuevas versiones de PostGIS/PostgreSQL y OpenLayers, para adaptar el proyecto a una versión más actual del programa.

2. ESTRUCTURA DE LA MEMORIA

La memoria de este proyecto se ha estructurado de la siguiente forma:

En primer lugar, se ofrece una introducción del funcionamiento de pgRouting y de la arquitectura cliente-servidor. Además se ha añadido una descripción del municipio de Santa Eulària des Riu y el ámbito geográfico del trabajo, y una descripción de cada uno de los programas que han sido utilizados en este proyecto, junto con la función que cumple cada uno.

En el tercer capítulo se explica el proceso de obtención de la cartografía de los diferentes organismos oficiales y su tratamiento en QuantumGis, preparando las capas para su carga en la base de datos.

El cuarto capítulo trata sobre el proceso de creación y preparación de la base de datos y de los procesos que se han seguido, tras la carga de datos, en la creación de una red valida y funcional para pgRouting. Se mostrarán algunos ejemplos de cálculo de rutas utilizando algoritmos que proporciona pgRouting y se creara una función de cálculo de ruta por medio de uno de los algoritmos de pgRouting más adecuada al objetivo del proyecto.

A continuación se explica la importación de datos en el servidor Geoserver, con sus debidas consultas SQL hacia la base de datos, sus estilos y la información que este ofrece a través de las peticiones GetCapabilities a éstos.

Por último se detalla la creación del visor web, la incorporación de los diferentes servicios comentados y la programación del visualizador cartográfico.

Al final de la memoria se encuentra la conclusión, las aplicaciones futuras, las referencias bibliográficas y los anexos.

3. INTRODUCCIÓN

3.1. Introducción a pgRouting

PgRouting añade la funcionalidad de enrutamiento y otras funcionalidades de análisis de una red para PostGis/PostgreSQL. Un precursor de pgRouting fue pgDijkstra, escrito por Sylvain Pasche de Camptocap, fue extendido más tarde por Orkney y renombrado a pgRouting. El proyecto es soportado y mantenido actualmente por Georepublic, iMaptools y una comunidad de usuarios.

PgRouting es un OSGeo Labs proyecto de la OSGeo Foundation e incluido en OSGeo Live.

PgRouting proporciona funciones para:

- All Pairs Shortest Path, Johnson's Algorithm [1]
- All Pairs Shortest Path, Floyd-Warshall Algorithm [1]
- Shortest Path A*
- Bi-directional Dijkstra Shortest Path [1]
- Bi-directional A* Shortest Path [1]
- Shortest Path Dijkstra
- Driving Distance
- K-Shortest Path, Multiple Alternative Paths [1]
- K-Dijkstra, One to Many Shortest Path [1]
- Traveling Sales Person
- Turn Restriction Shortest Path (TRSP) [1]
- Shortest Path Shooting Star [2]

PgRouting está disponible bajo la licencia GPLv2 y es apoyado por una comunidad cada vez mayor de personas, empresas y organizaciones.

PgRouting página web: http://www.pgrouting.org

[1] Nuevo en 2.0.0 pgRouting

[2] Interrumpidas en pgRouting 2.0.0

3.2. Descripción del entorno de actuación

El municipio de Santa Eulària des Riu ocupa el sector suroriental de la isla de Eivissa, del archipiélago y provincia de Illes Balears. Con una superficie de 153,48 km^2 y un perímetro de 170 km es el segundo mayor de la isla.

Fig. 2: Situación del municipio de Santa Eulària des Riu.

A 1 de enero de 2011, la revisión del padrón municipal mostraba una población de 33.734 habitantes, con una densidad de 219 hab/km^2. La población de Santa Eulària des Riu representa alrededor de un 24% del total de la población ibicenca. Alrededor del 73% de la población vive en los diversos núcleos urbanos del municipio y un 27% es población diseminada.

El territorio municipal está dividido en cinco parroquias: es Puig d'en Valls, Jesús, Sant Carles de Peralta, Santa Eulària des Riu y Santa Gertrudis de Fruitera. Estas constituyen en la actualidad una subdivisión civil del municipio con cierto uso administrativo ya que coinciden con las entidades singulares y los distritos del Instituto Nacional de Estadística (INE). Su origen deriva de las antiguas demarcaciones eclesiásticas denominadas de igual manera.

Cada parroquia está dividida en una zona urbana, con uno o varios núcleos de población, y otra rústica. La delimitación de las zonas urbanas está definida en el planeamiento municipal vigente. La edificación en la zona rústica es principalmente diseminada aunque existen algunas pequeñas agrupaciones o núcleos rurales.

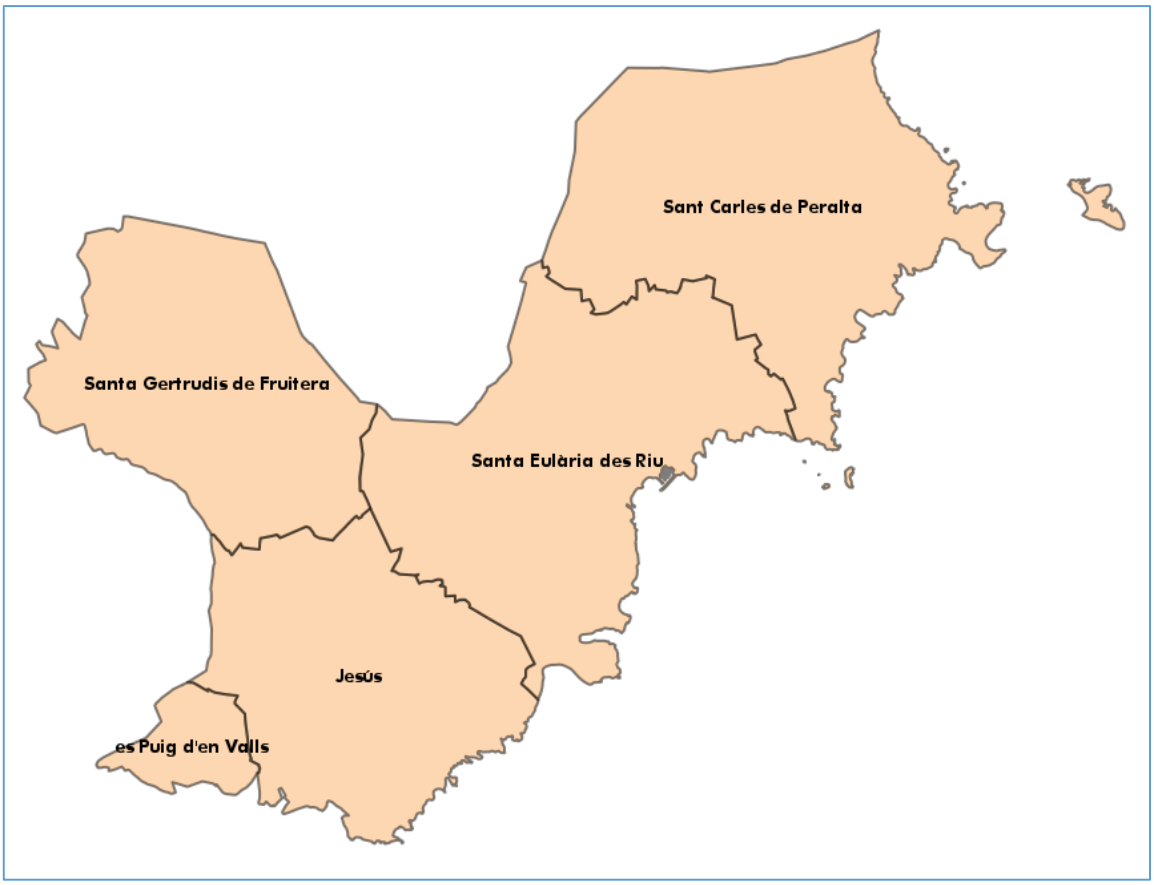


Fig. 3: Delimitación de las parroquias.

Las zonas rústicas de las parroquias están divididas a su vez en “véndas”. Estas son, desde antaño, la división territorial tradicional de la isla de Eivissa. Inicialmente, la “vénda” constituía una agrupación de “casaments” o casas payesas próximas que se organizaban para la realización de trabajos comunitarios y actos festivos. Con el paso del tiempo estas se establecieron como unidades territoriales más o menos definidas, delimitadas por accidentes geográficos y caminos. Esta demarcación se estructuró sobre los antiguos “quartons” en los que se dividió la isla tras la conquista catalana en el siglo XIII. El número de “véndas” y sus límites territoriales han ido variando con el tiempo adaptándose a las variaciones demográficas y a las necesidades de cada época. A finales del siglo XVIII fueron creadas la mayoría de las parroquias ibicencas, constituyéndose como una división territorial de orden superior a las “véndas”, las cuales pasaron a ser una subdivisión del dominio parroquial. Entrado el siglo XIX, se organiza la isla en municipios a partir de la agrupación de las parroquias. A mediados del siglo XX, debido a la implantación del Catastro, cuya división municipal es el polígono, la organización territorial tradicional basada en “véndas” y parroquias cae en desuso, quedando solo para uso administrativo estas últimas. Por ser un patrimonio histórico, cultural y social, tanto del municipio como de toda la isla.

Fig. 4: Delimitación de las "véndas".

3.3. Software utilizado

Para este proyecto se ha requerido utilizar gran variedad de software debido a las diferentes funciones que se han tenido que realizar. Todos estos programas son libres y se presentan como alternativa a otros programas de pago. Uno de los objetivos de este proyecto es mostrar que pueden llevarse a cabo proyectos de calidad sin utilizar caros programas con licencias de pago. La excepción ante esto es el sistema operativo, que ha sido sobre Windows 10.

A continuación se van a explicar los distintos programas utilizados en este proyecto.

3.3.1. Geodatabase: PostgreSQL/Postgis

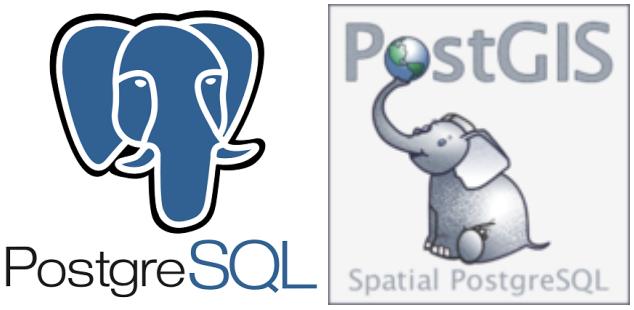


Fig. 5: Logotipos de PostgreSQL y Postgis

PostgreSQL

Es un sistema de gestión de base de datos relacional orientado a objetos. Actualmente es uno de los SGBDR de código abierto más potente del mercado comparable a otras opciones comerciales. A día de hoy se distribuye la versión 9.3.13, siendo esta la versión utilizada en este proyecto. Sus principales características son:

- Permite implementar la arquitectura cliente –servidor.
- Base de datos 100% CRUD (crear, leer, actualizar y eliminar) y ACID (atomicidad, consistencia, aislamiento y durabilidad).
- Integridad referencial.
- Implementación del estándar SQL92/SQL99/SQL2003/SQL2008.
- Soporta múltiples tipos de datos a parte del estándar y permite definir nuevos tipos.
- Incorpora funciones de diversa índole y permite la declaración de funciones propias, así como la definición de "*triggers*" (disparadores).
- Soporta el uso de índices, reglas y vistas.
- Permite la gestión de usuarios y permisos.

Postgist

Es un módulo que añade soporte espacial a PostgreSQL convirtiéndolo en una GDB. Actualmente se distribuye la versión 2.2, siendo esta la utilizada en el presente proyecto. Sus principales características son:

- Compatible con los estándares de Open Geospatial Consortium (OGC).
- Librería GDAL con multitud de sistemas de referencia, proyecciones y tipos de coordenadas.
- Soporta varios tipos de datos espaciales.
- Posee múltiples funciones espaciales.
- Permite importar y exportar datos a otros formatos (shapefile, DXF, OGR
- Multitud de clientes SIG (Quantum SIG, gvSIG, Autocad Map 3D, ArgGIS, etc.), tanto libres como propietarios, permite visualizar y editar datos de Postgist.
- Los principales servidores de mapas web permiten la conexión a datos Postgis (Mapserver, Geoserver, Mapguide, ArcGIS).

La GDB PostgreSQL/Postgis se instala en el equipo servidor (local).

3.3.2. Cliente SIG: Quantum GIS

Un cliente SIG es una aplicación informática que permite visualizar y tratar información geográfica de diversas fuentes a través de sus diferentes herramientas y comandos. En este proyecto representa la aplicación que se conecta con la GDB y permite visualizar, introducir, editar y consultar la información de las capas, tanto alfanumérica como espacial.

De toda la oferta de clientes SIG libres (Quantum GIS, gvSIG, Kosmo, uDig, Tatuk GIS, etc.) se ha seleccionado para este proyecto Quantum GIS por ser una aplicación con múltiples funcionalidades y tener la mejor compatibilidad con PostgreSQL/Postgis.

Fig. 6: Logotipo de Quantum GIS

QGIS es una aplicación SIG de escritorio de código libre. La versión 2.14.3 es la que se ha utilizado en este proyecto. Sus principales características son:

- Permite conexión, visualización edición y consulta de datos PostgreSQL/Postgis así como de otras GDB.
- Permite importar y exportar múltiples tipos de datos vectoriales y ráster.
- Permite conexiones WMS y WFS.
- Posee varias herramientas de análisis espacial.
- Interfaz de usuario amigable.

En este proyecto se ha instalado en el equipo servidor. Como aplicación cliente esto no es necesario, pudiendo trabajar desde cualquier equipo conectado al servidor utilizando la red local.

3.3.3. Visor web: HTML, APACHE TOMCAT, GEOSERVER y OPENLAYERS

Una aplicación web es un programa informático que los usuarios pueden emplear accediendo a un servidor web a través de internet o de una intranet mediante un navegador web (Internet Explorer, Mozilla Firefox, Chrome, etc.). Una de las principales ventajas es que actúan como clientes ligeros con independencia del sistema operativo y sin necesidad de distribución ni instalación de software, por lo que son fáciles de actualizar y mantener. Una aplicación web, si así se desea, es accesible desde cualquier equipo conectado a internet, por lo que pueden ser empleadas por multitud de usuarios.

Para el desarrollo del citado visor web se han utilizado los elementos siguientes: el lenguaje HTML, el servidor de mapas web Geoserver, el servidor web Apache y OpenLayers 3.

HTML

HTML significa "*Hypertexted Markup Language*" (Lenguaje de Marcas de Hipertexto). Es el lenguaje de programación predominante en el desarrollo de páginas y aplicaciones web. Este lenguaje permite crear la estructura del visor web. Los documentos HTML son almacenados en servidor y se accede a ellos vía internet o intranet desde el navegador web del equipo cliente donde son ejecutados.

Fig. 7: Logotipo de HTML (versión 5).

Geoserver

Es un servidor ampliamente utilizado para compartir, procesar y editar información geoespacial.

A Geoserver se pueden importar capas en formato vectorial como SHP, acceder a bases de datos como PostGIS, Oracle y otras, subir cartografía en formatos ráster como GeoTiff, ECW y más y añadir información conectando el servidor con servicios WMS o WFS creados por otros organismos.

Además la aplicación permite añadir diferentes estilos a las capas que tengamos alojadas en él en formato sld. Permite asignar diferentes estilos a cada capa, dando la posibilidad de previsualizar desde la misma aplicación. La versión instalada para este proyecto es la 2.9.1.

Fig. 8: Logotipo de Geoserver.

Apache TomCat

Es un servidor de servlets, programas utilizados a través de un navegador web. TomCat permite añadir estas aplicaciones, que el administrador del servidor podrá gestionar desde su Manager, e incluye herramientas para su configuración y manejo, permitiendo acceder a ellas, así como detenerlas o arrancarlas.

Apache TomCat es una aplicación web libre desarrollada por Apache Software Foundation a la que se puede acceder desde el propio navegador web y que utilizaremos para hacer funcionar GeoServer.

Fig. 9: Logotipo del servidor web Apache TomCat

OpenLayers

Es una librería JavaScript Open Source que permite publicar de forma sencilla mapas dinámicos en una página web.

Se trata de un cliente ligero que permite la integración de servicios de visualización de mapas sobre navegadores web. Estos clientes funcionan mediante la interactuación de los siguientes elementos:

- Un navegador web.
- Una API (Application Programming Interface).
- Un documento HTML que contendrá la visualización a modo de página web.
- Una programación JavaScript.

Para este proyecto se ha utilizado OpenLayers 3.

Fig. 10: Logotipo de OpenLayers.

A continuación podemos observar un esquema del flujo de la información durante la utilización del visor web, donde en la Geodatabase estará la capa de la red de viales y como cartografía la ortofoto del municipio, el límite del municipio y sus topónimos.

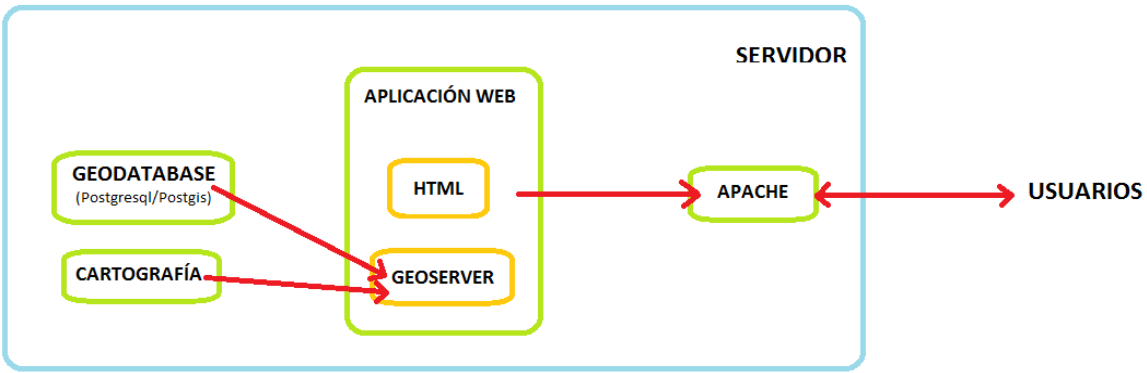


Fig. 11: Esquema del flujo de trabajo del visor web desarrollado.

4. CARTOGRAFÍA

En los siguientes apartados se describe la información geográfica empleada para el servidor WMS, sus principales características como formato, tipo de entidad, proyección, etc., así como la fuente de información de la que provienen.

Se debe tener en cuenta que los WMS tienen la capacidad de leer los datos en sus formatos originales (dgn, shapelife, geotiff, ecw, conexiones a bases de datos Postgis, etc.) y generar como producto de salida una imagen, por lo que esto evita tener que transformar el formato de almacenamiento de los datos.

Es fundamental que las capas de información geográfica se encuentren georreferenciadas, para conseguir superponer capas de distintas fuentes, pero no necesariamente tienen que estar en el mismo Sistema de Referencia de Coordenadas, ya que este se puede definir posteriormente en el servidor WMS, el cual posee la capacidad de reproyectarlas. Pero para evitar cualquier inconveniente toda la cartografía estará en EPSG:25831.

Se ha empleado el software de QGIS para analizar, modificar y crear cada una de las capas que finalmente se añadirán al servidor.

4.1. Origen de los datos

El Instituto Geográfico Nacional proporciona, por medio de su Centro Nacional de Información Geográfica, la posibilidad de descargar cartografía entre una gran variedad de productos. Esta cartografía puede ser descargada gratuitamente si se confirma que no va a ser utilizada para un uso comercial, habiendo aceptado previamente su licencia de uso.

Desde el catálogo de productos ofrecido, se han descargado los siguientes productos:

- Ortofotografía del PNOA de máxima actualidad

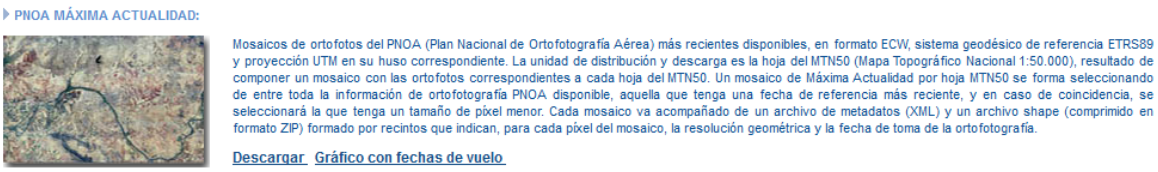


Fig. 12: Descripción del PNOA

El Plan Nacional de Ortofotografía Aérea consta de una serie de fotografías aéreas de gran resolución que realiza el IGN para producir ortofotografías aéreas y modelos digitales del terreno de todo el territorio español. Estas fotografías se utilizan además para la realización de cartografía e información geográfica y son acordes con el espíritu de la Directiva INSPIRE por promover la producción centralizada de datos geográficos y su aprovechamiento entre diferentes organismos.

Se han descargado las fotografías que corresponden con la zona que ocupa el área que abarca el municipio, en este caso son las hojas 772, 773, 798 y 799 del año 2015.

- CartoCiudad

Fig. 13: Descripción de CartoCiudad.

CartoCiudad es un proyecto colaborativo entre distintos organismos de cobertura nacional que contiene cartografía urbana. Sus capas provienen de la DGC y el IGN y se puede acceder a ellas mediante un geoportal llamado CartoVisor y servicios web. La cartografía se descarga en un formato .zip con la cartografía de toda la provincia de Baleares e incluye múltiples capas. Para este caso solo es necesaria la capa de tramos viales, la cual contiene distintos tipos de carreteras, vías urbanas, caminos, etc, y la capa de Municipio, para tener el límite que abarca este proyecto. Aunque también proporciona una capa de topónimos que es requerida para este proyecto, lamentablemente su información es escasa.

- BTN25

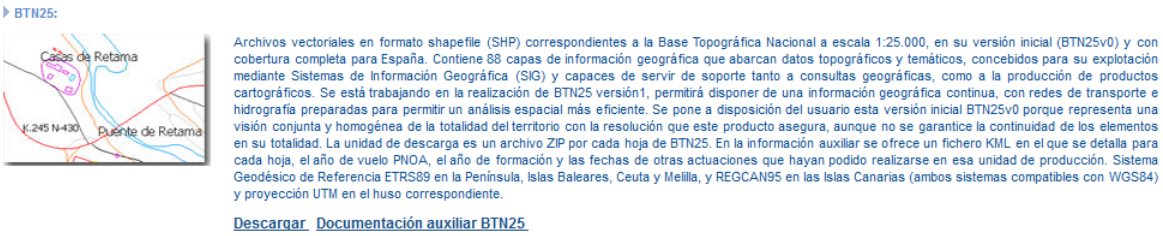


Fig. 14: Descripción de BTN25

La Base Topográfica Nacional es una base de datos geográfica continua que cubre todo el territorio nacional y se crea a partir de las fotografías del PNOA mientras que acuerdos con comunidades autónomas permiten obtener datos actualizados de mayor resolución.

El CNIG nos permite descargar un archivo .zip con una gran cantidad de capas de la BTN con información sobre delimitaciones territoriales, elementos hidrográficos, edificaciones y construcciones, redes e infraestructuras de transporte y más. De entre todas estas nos quedaremos con una, la de topónimos.

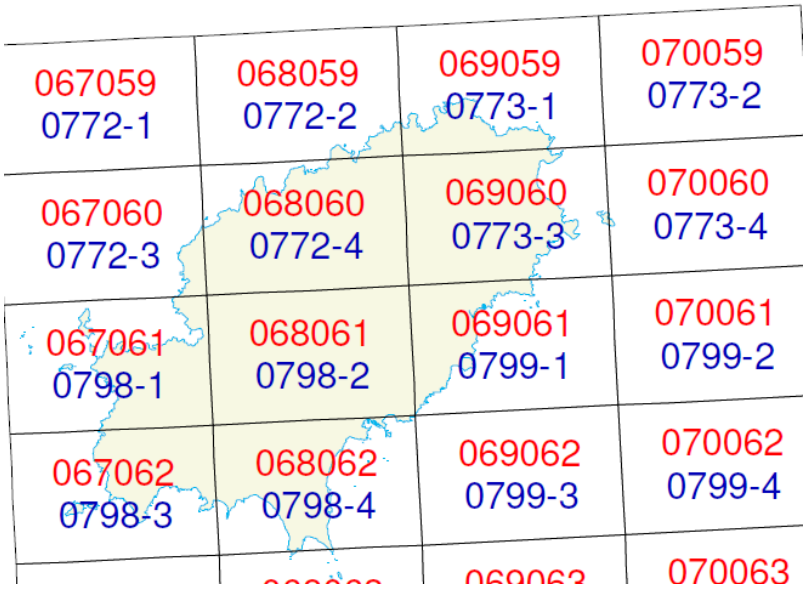


Fig. 15: Numeración de las hojas del BTN25.

4.2. Tratamiento de la cartografía descargada

A continuación se va a explicar los distintos procedimientos y herramientas utilizadas sobre los datos descargados para preparar su carga en el servidor y su posterior visualización en el visor web.

4.2.1. Limite municipal

Es un archivo en formato shp con EPGS:4258 (se debe transformar a 25831), el cual es usado para recortar diferentes capas, delimitar el límite municipal sobre la ortofoto y servirá para señalar al usuario en el visor web dentro de que área puede realizar consultas de rutas.

4.2.2. Ortofoto PNOA 2015

Una vez descargados estos datos obtenemos cuatro imágenes ráster, cada una correspondiente con las hojas ya nombradas, en formato ecw y su sistema de referencia de coordenadas en ETRS89 huso 31. Cargamos estos archivos en QGIS para su visualización, tal y como muestra la siguiente imagen (en amarillo se muestra el límite del municipio):

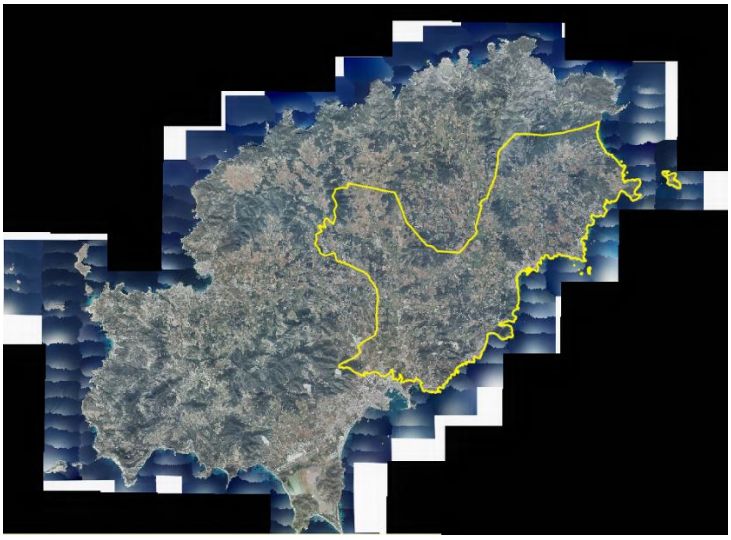


Fig. 16: Ortofoto PNOA 2015 de la isla de Ibiza.

Como se puede observar en la imagen, no se requiere una imagen tan grande. Por lo tanto se procede a unir las cuatro imágenes ráster y posteriormente recortarlas para adaptarlas mejor al área que ocupa nuestro municipio, obteniendo el siguiente resultado:

Fig. 17: Ortofoto PNOA 2015 recortada.

4.2.3. Tramos viales

Es un archivo en formato shp con EPGS:4258 el cual será transformado a EPGS:25831 por medio de QGIS (como ya se mencionó anteriormente el servidor trabajará en EPGS:25831). Al cargar esta capa en QGIS vemos que tenemos la mayoría de viales de toda Baleares, por lo tanto el primer paso será eliminar los viales de las islas de Mallorca, Menorca y Formentera, ya que no entran dentro del municipio, el procedimiento es muy simple, activamos la herramienta de edición de la capa, seleccionamos con la herramienta selección los tramos viales del resto de islas y los eliminamos.

Una vez eliminada la información innecesaria, el siguiente paso es visualizar la capa de tramos viales junto a la capa poligonal del municipio y averiguar cuál es el procedimiento adecuado para recortar los tramos viales que contiene el municipio.

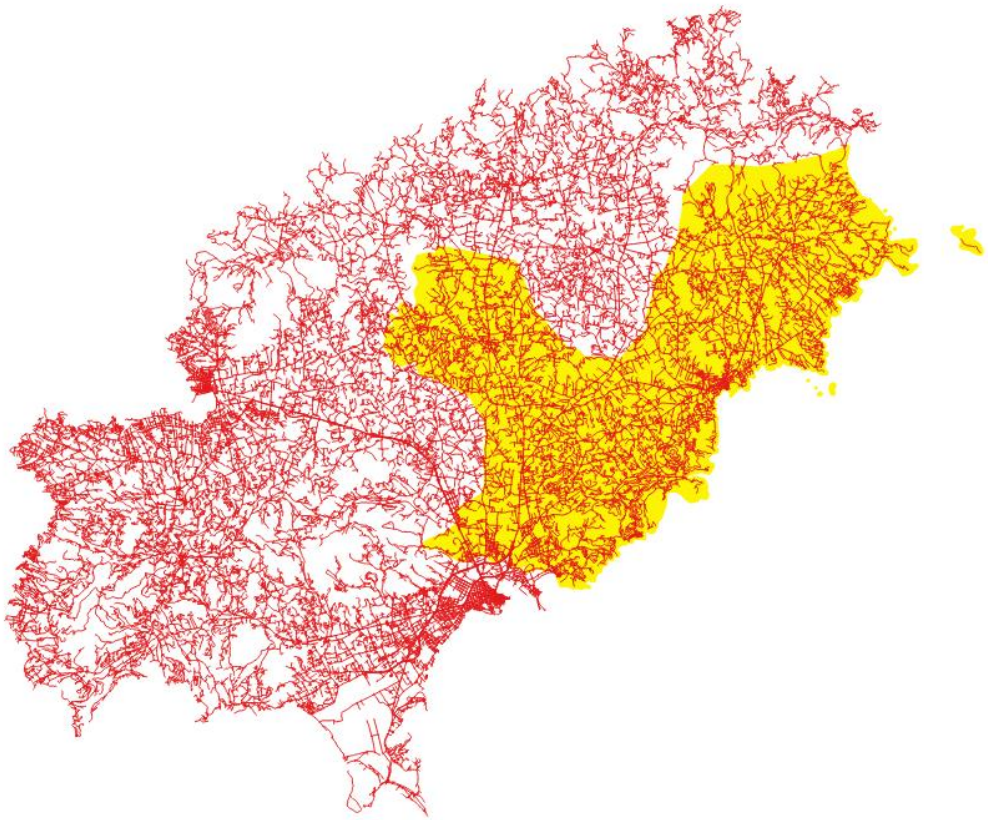


Fig. 18: Visualización en QGIS de la capa municipio y tramos viales.

Estudiando los límites del municipio muchos cruces de caminos se encuentran fuera de este, por lo tanto si recortamos directamente con la capa poligonal del municipio se perdería la continuidad de estos tramos de estos tramos viales.

Fig. 19: Ejemplo de cruce de caminos ubicado fuera del municipio.

Como muestra la imagen anterior, aunque el cruce de caminos este ubicado fuera del municipio no se debe eliminar puesto que se perderían datos necesarios para trazar rutas. La solución adoptada es la ampliación del polígono municipal para así abarcar estos casos. Para generar este nuevo polígono se selecciona la herramienta vectorial de geoproceso "*buffers*" y la aplicamos sobre la capa. La distancia de "*buffer*" seleccionada es de 100 metros (se han realizado muestras aleatorias y ninguna era superior a 100 m).

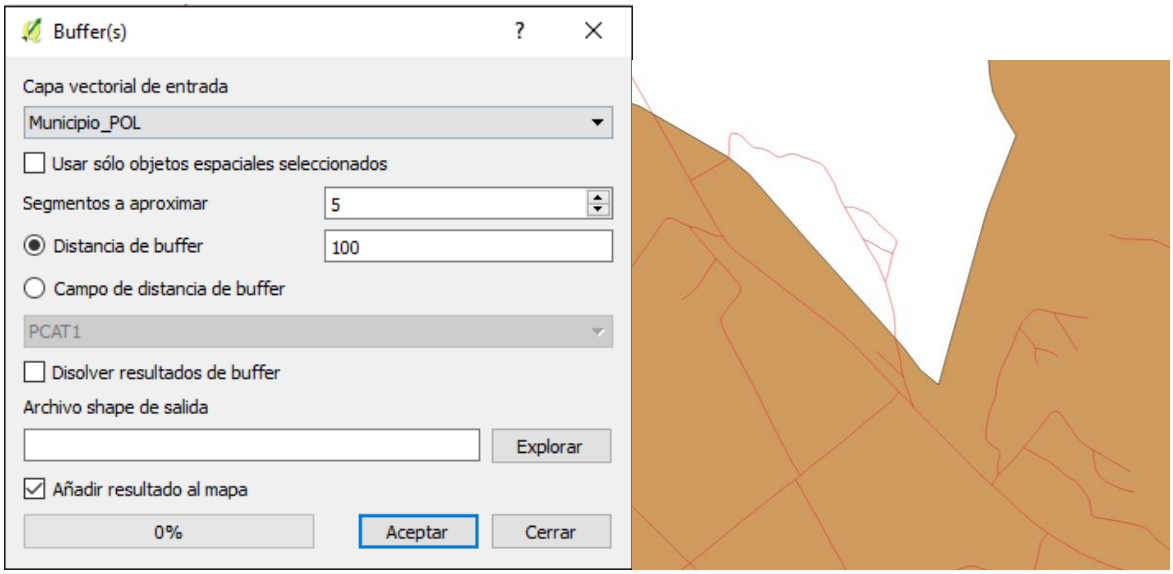


Fig. 20: Opciones de la herramienta "buffer" y resultado obtenido para el ejemplo.

Como resultado se obtiene una nueva capa poligonal mayor que la capa original, como muestra la siguiente imagen:

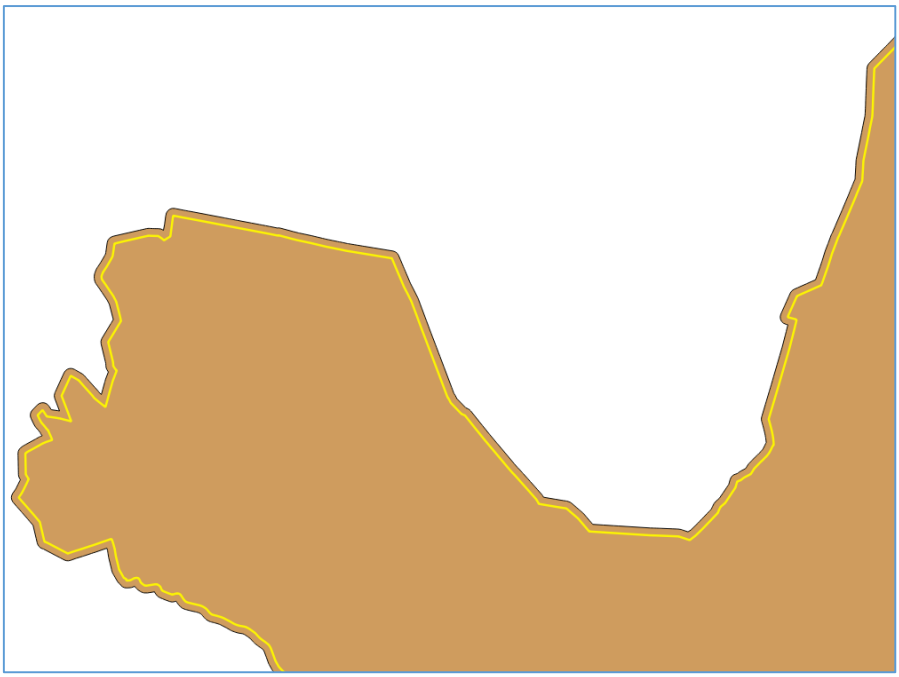


Fig. 21: Nueva capa resultado del "buffer".

Con esta nueva capa poligonal se procede a recortar la capa de tramos viales por medio de la herramienta de intersección.

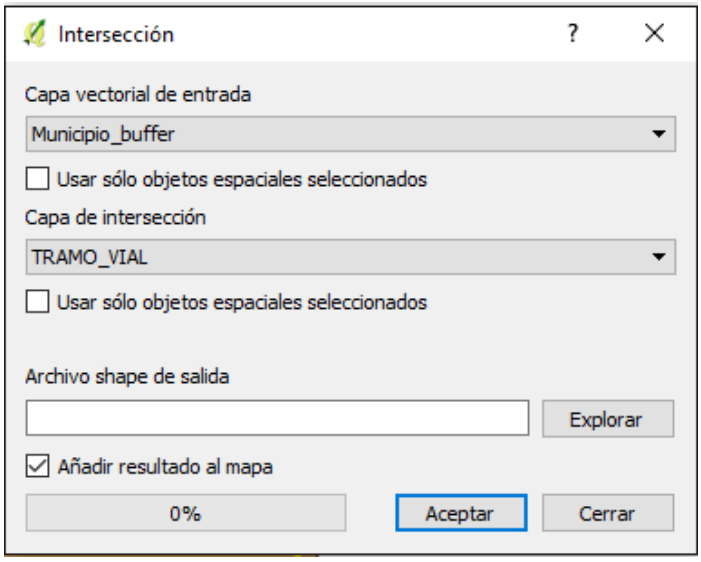


Fig. 22: Opciones de la herramienta intersección.

Obteniendo así una nueva capa de tramos viales la cual solo contiene los viales del municipio. El siguiente paso es revisar esta nueva capa eliminando los caminos que no son públicos, casos concretos donde el camino actúa como acceso a una vivienda diseminada siendo considerado un camino privado, eliminar también los caminos que han

quedado sueltos o están fuera del límite municipal después de la intersección de capas y la digitalización de los caminos que no figuren en la capa.

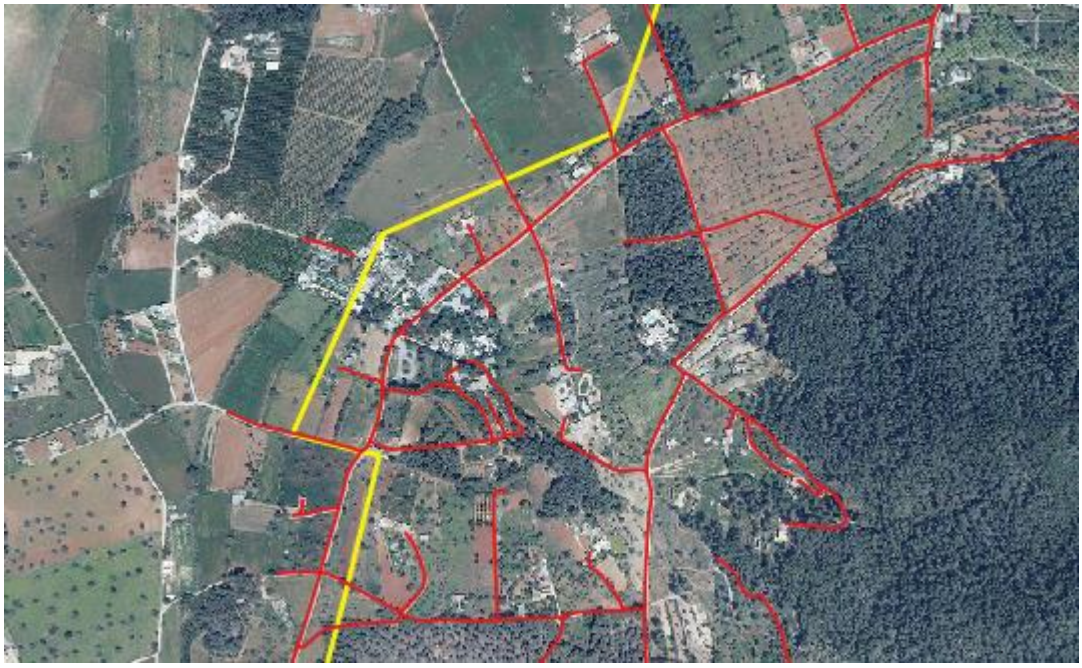


Fig. 23: Ejemplo de caminos que salen del límite municipal.

Para conocer si un camino es o no es privado se deben visualizar la capa de ortofoto y la de tramos viales al mismo tiempo. Estudiando la visualización que nos proporciona QGIS de ambas capas y por conocimiento del terreno se decide si ese camino debe ser eliminado. Actualmente no existe todavía un inventario de caminos públicos proporcionado por el Ayuntamiento de Santa Eulària des Riu. En la siguiente imagen vemos un ejemplo de caminos privados, marcados en lila.

Fig. 23: Ejemplo de caminos privados.

Para la digitalización de los caminos que no figuran en la capa, se procede de una forma parecida, visualizando en QGIS la ortofoto y la capa de tramos viales localizamos aquellos caminos que no estén digitalizados y se digitalizan de forma manual. Para digitalizar un camino se siguen unos pasos, primero se activa la edición en la capa deseada, seguidamente se selecciona la herramienta de añadir objeto espacial y lo vamos dibujando utilizando la ortofoto como guía.

Fig. 24: Ejemplo de camino digitalizado.

Después de digitalizar el camino QGIS nos muestra una ventana con los campos para que sean rellenados, pueden ser rellenados ahora o más tarde. Una vez rellenados los datos, aceptamos y QGIS guarda los datos introducidos en la tabla de la capa, se visualiza el camino digitalizado, se guarda la edición y se sale de edición, de esta forma el camino ya está digitalizado.

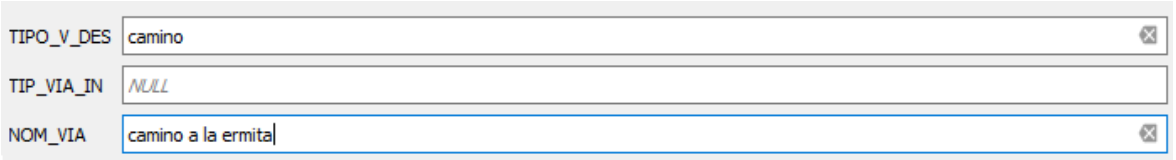


Fig. 23: Ejemplo de introducción de nuevos atributos a la tabla tramos viales.

Una vez eliminados los caminos privados y digitalizados los caminos que no figuraban, se obtiene lo siguiente:

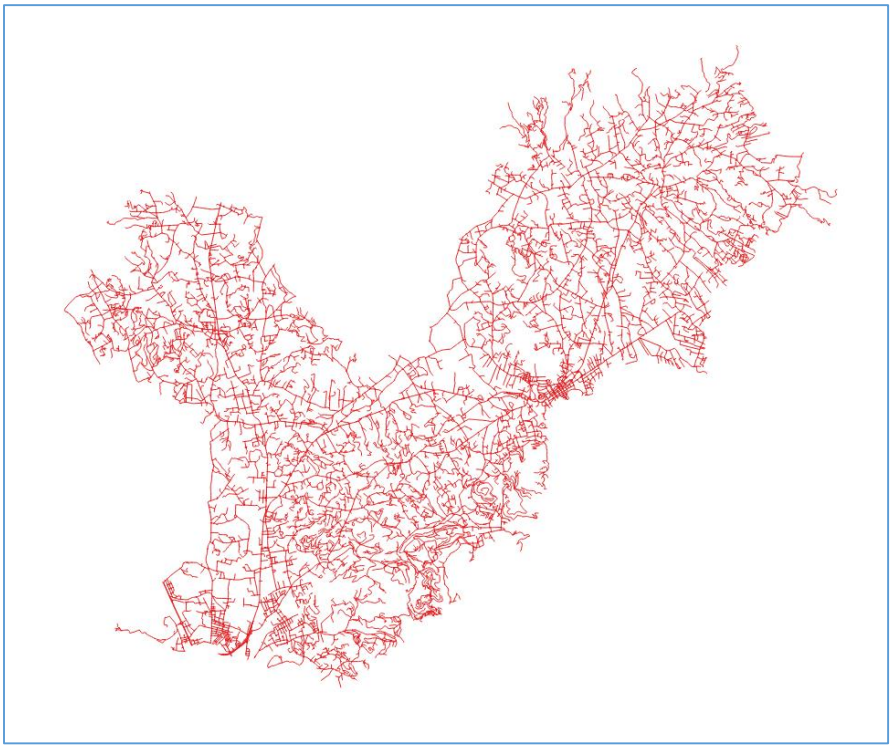


Fig. 24: Capa tramos viales resultado.

Ahora solo queda un paso para acabar con esta capa, corregir los errores que contiene la geometría, para ello se usa la herramienta comprobador de topología. Esta herramienta de QGIS es un complemento que permite encontrar errores de topología en capas vectoriales.

La topología describe las relaciones entre puntos, líneas y polígonos que representan las características de una región geográfica. Con este inspector de topología se pueden mirar los archivos vectoriales y comprobar la topología con varias reglas.

QGIS tiene una función de edición topológica incorporada, que es ideal para la creación de nuevas características sin errores. Pero los errores datos existentes y los errores inducidos por el usuario son difíciles de encontrar. Este plugin ayuda a encontrar este tipo de errores a través de una lista de reglas.

Como estamos editando una capa de líneas vamos a ver las reglas que necesitamos de esta herramienta:

- No debe superponerse: requiere que las líneas no se superpongan con las líneas de la misma clase (o subtipo) de entidad. Esta regla se utiliza en aquellos segmentos de línea que no se deberían duplicar.

- No debe intersectarse: Requiere que las entidades de línea de la misma clase (o subtipo) de entidad no se crucen ni se superpongan entre sí. Las líneas pueden compartir extremos. Esta regla se utiliza para líneas de contorno que nunca se deben de cruzar entre sí o en los casos en los que la intersección de las líneas se debe producir únicamente en extremos.

- No deben quedar nodos colgados: Requiere que una entidad de línea deba tocar las líneas desde la misma clase (o subtipo) de entidad en ambos extremos. Un extremo que no esté conectado con otra línea se llama nodo colgado (dangle). Esta regla se utiliza cuando las entidades de línea deben formar bucles cerrados, como cuando definen los límites de las entidades poligonales. También se podría utilizar en los casos en los que las líneas se conectan generalmente con otras líneas, como con calles. En este caso, las excepciones se pueden utilizar allí donde la regla se viola ocasionalmente, como con segmentos de calle sin salida.

- No deben quedar pseudonodos: Requiere que una línea se conecte, por lo menos, con otras dos líneas en cada extremo. Las líneas que se conectan con otra línea (o con ellas mismas) se dice que tienen pseudonodos. Esta regla se utiliza donde las entidades de línea deben formar bucles cerrados, como cuando definen los límites de los polígonos o cuando las entidades de línea se deben conectar de forma lógica con otras dos entidades de línea en cada extremo, igual que con segmentos en una red de transmisión, marcándose las excepciones para los extremos que originan las transmisiones de primer orden.

Estas son las reglas que se utilizan en esta capa, aunque existen más reglas no son necesarias para este trabajo, pero hay que tener en cuenta unas excepciones. Para la regla de no se debe intersectar no se aplicara en el caso de pasos elevados, donde una carretera pasa por encima de otra, en el municipio únicamente hay un caso.

Fig. 25: Caso paso elevado.

La otra excepción será para la regla no deben quedar nodos colgados, puesto que al recortar la capa de viales con la del municipio creamos nodos colgados y las calles sin salida o finales de caminos serán otro caso de nodo colgado que no debe considerarse un error.

Fig. 26: Caso calle sin salida.

Teniendo estas excepciones en cuenta se seleccionan las reglas topológicas nombradas anteriormente y se validan para que QGIS muestre los errores sobre la capa vectorial como muestra la imagen siguiente:

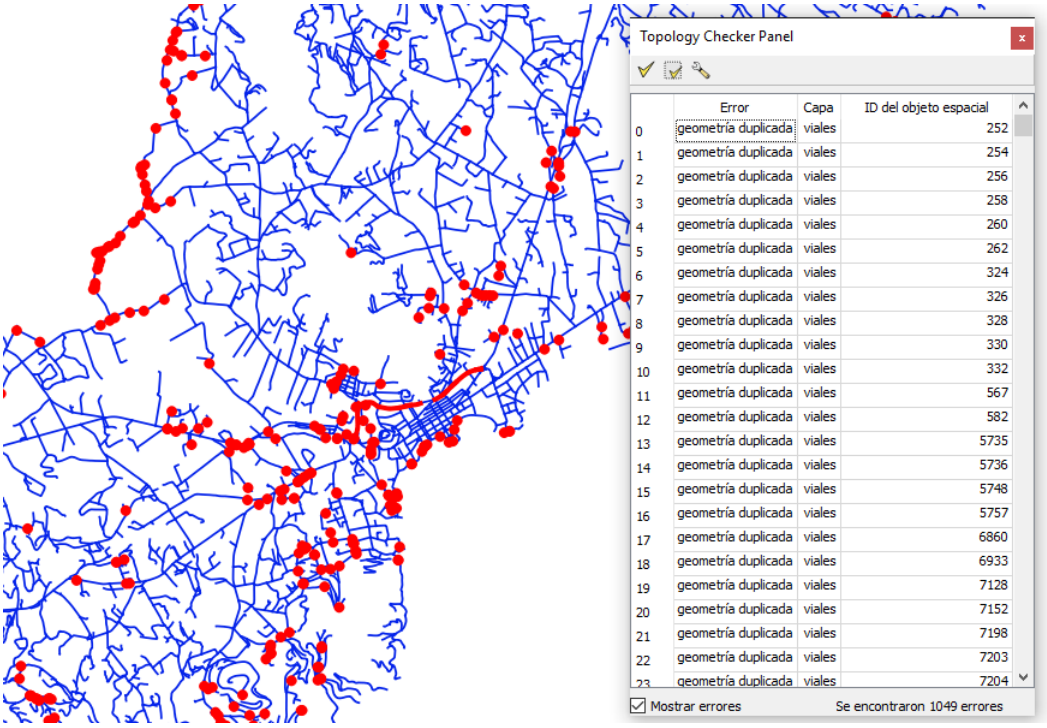


Fig. 27: Validación de la topología

Como podemos ver en la imagen el panel de la derecha nos marca los distintos tipos de error y nos permite realizar un zoom a esa zona donde se produce el error para poder estudiarla y corregirla.

Para corregir los errores se debe activar la edicion de esta e ir corrigiendolos caso por caso. Aunque la herramienta marque mas de 1000 errores, muchos de ellos son por nodos colgados de finales de caminos o calles sin salida. En realidad solo hay 7 errores por solape de lineas, un error de intersección, el cual es el paso elevado y por tanto realmente no es un error, pero en el caso de los pseudonodos si que mostraba bastantes erroes. Para corregirlos se ha tenido que comprobar a que linea conectar cada extremo para asi eliminar los pseudonods.

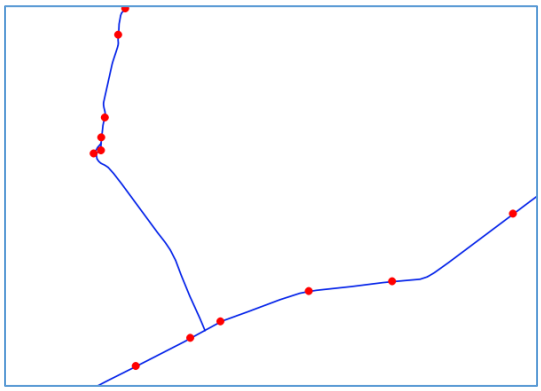


Fig. 28: Ejemplo de error por pseudonodos.

Con esto concluye la preparación de la capa de viales, dando como resultado la siguiente red de viales, teniendo una tabla de atributos con 9601 objetos y tres campos:

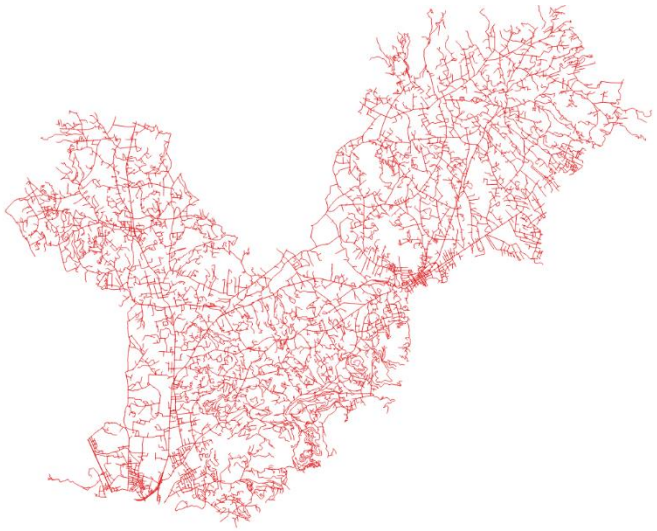
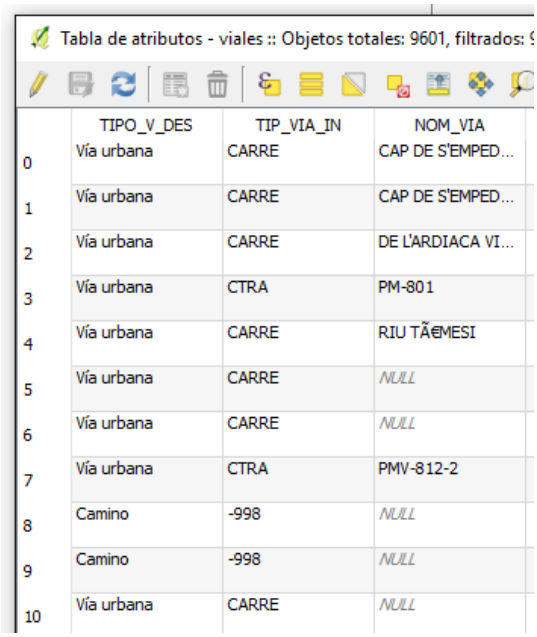


Fig. 29: Capa de viales resultado.



	TIPO_V_DES	TIP_VIA_IN	NOM_VIA
0	Vía urbana	CARRE	CAP DE S'EMPED...
1	Vía urbana	CARRE	CAP DE S'EMPED...
2	Vía urbana	CARRE	DE L'ARDIACA VI...
3	Vía urbana	CTRA	PM-801
4	Vía urbana	CARRE	RIU TÃ€MESI
5	Vía urbana	CARRE	NULL
6	Vía urbana	CARRE	NULL
7	Vía urbana	CTRA	PMV-812-2
8	Camino	-998	NULL
9	Camino	-998	NULL
10	Vía urbana	CARRE	NULL

TIPO_V_DES: Descripción del tipo de vía, vía urbana o camino.

TIP_VIA_IN: Indica el tipo de vía que es, si es carretera convencional, carretera de altas prestaciones, camino, calle, etc.

NOM_VIA: Es el nombre de la vía.

Fig. 30: Tabla de atributos de viales.

Este archivo shp es la red que utilizará pgrouting para calcular los distintos tipos de ruta en nuestro visor web.

4.2.4. Toponimia

Como se vio en la imagen de la *Fig.15* son seis archivos en formato shp con EPGS:4258 el cual será transformado a EPGS:25831 por medio de QGIS (como ya se mencionó anteriormente el servidor trabajará en EPGS:25831). Al cargar esta capa en QGIS vemos que tenemos la mayoría de topónimos de casi toda Ibiza, por lo tanto el primer paso será unir las seis capas puntuales y posteriormente cortarlas con la capa poligonal del municipio. Para unir las capas se emplea la herramienta de geoproceso unión, la cual unirá las seis capas en una sola, haciendo más fácil su recorte con la capa poligonal que delimita el municipio. El resultado de este proceso de unir y recortar es el que muestra la siguiente imagen:

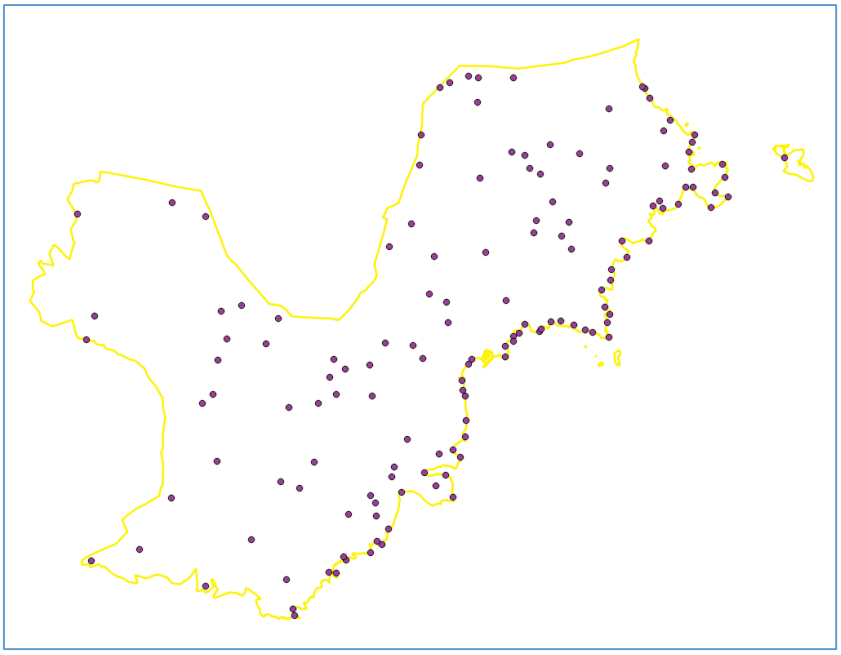


Fig. 31: Capa puntual de topónimos.

Hay que recordar que esta capa se utilizara en el visor web para marcar los distintos recursos turísticos del municipio, los cuales son playas o calas, puntas o cabos, torres, museos, iglesias, ermitas, urbanizaciones y pozos, el resto de topónimos no interesan para este trabajo. Si se comprueba la tabla de atributos vemos solo interesan dos campos, la etiqueta y el tipo_0801.

	ETIQUETA	TIPO_0801
0	Cala Roja	080327
1	Cap Martinet	080330
2	Es Serral	080214

Fig. 32: Tabla de atributos de topónimos.

Para conocer a qué tipo de topónimo le pertenece cada código hay que descargarse el documento proyecto_btn25.pdf y consultar la tabla del apartado 0801 Topónimo Sin Geometría, donde vemos la siguiente tabla

ATRIBUTO:	0801P01	TIPO		TIPO_0801	CodeList
		080111	COMARCAS GRANDES Y MEDIANAS		
		080116	COMARCAS MENORES Y GRANDES PARAJES DE 1		
		080214	PARAJES		
		080322	MARES Y OCÉANOS		
		080325	ESTRECHOS, GOLFOS Y BAHÍAS PRINCIPALES		
		080326	ESTRECHOS, GOLFOS Y BAHÍAS MEDIANAS		
		080327	ESTRECHOS, GOLFOS, BAHÍAS PEQUEÑAS, CALAS, RADAS		
		080328	SALIENTES COSTEROS PRINCIPALES		
		080329	SALIENTES COSTEROS MEDIANOS		
		080330	SALIENTES COSTEROS PEQUEÑOS, CABOS, PUNTAS		
		080337	PLAYAS PRINCIPALES		
		080338	PLAYAS MEDIANAS		
		080339	PLAYAS PEQUEÑAS		
		080520	BARRIOS EN CIUDADES GRANDES		
		080521	DISTRITOS EN CIUDADES GRANDES		

Tabla 1: 0801 Topónimo sin geometría.

Comprobando esta tabla solo nos interesan los topónimos de salientes costeros (suelen ser miradores), playas y barrios. El resto de topónimos serán creados manualmente.

Para no estar consultando continuamente la tabla de topónimos del BTN25 se crea un nuevo campo en la tabla de topónimos, llamado tipo_nom, así se obtiene un campo que indica que tipo de recurso es con letras y no con números. Para crear un nuevo campo en una tabla en QGIS hay que activar la edición de la capa y seleccionar la opción de nuevo campo, darle un nombre al nuevo campo y su tipo, que para este caso es texto (cadena) con una longitud de 100 caracteres.

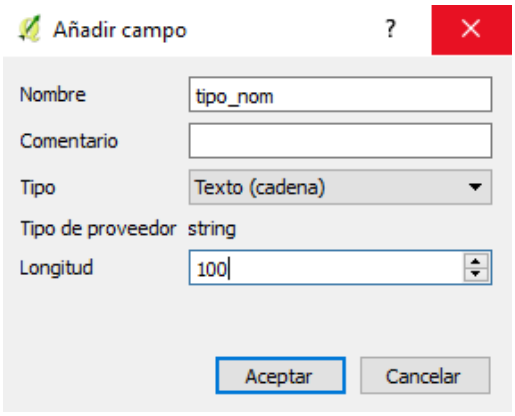
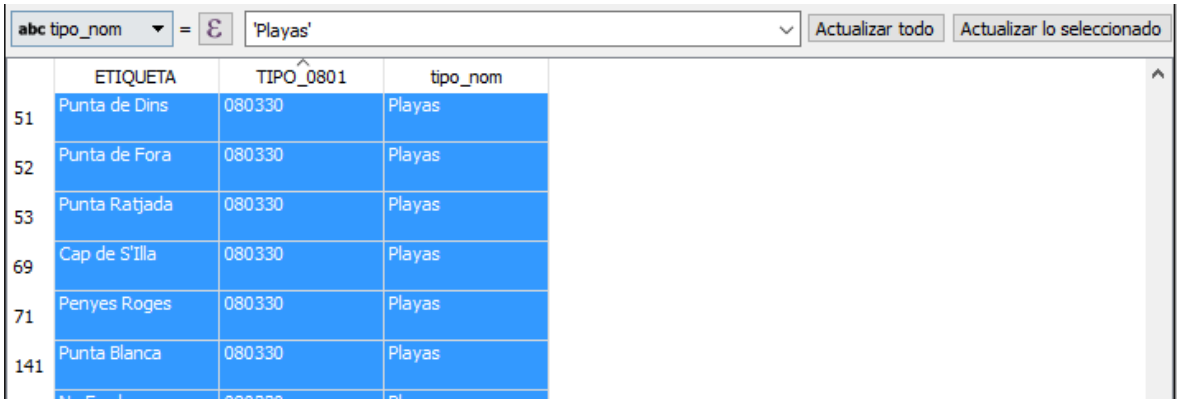


Fig. 33: Creación de un nuevo campo.

Al crear el nuevo campo, tenemos que rellenarlo, puesto que solo tiene valores NULL. Para rellenar el campo rápidamente aprovechando el código del BTN25 se realiza una selección de objetos espaciales por medio de una expresión. Por ejemplo si el campo tipo_0801 tiene como código 080338, que se refiere a playas, se realiza una selección por atributos, que será el código 080338 y con la expresión tipo_nom='Playas' y se hace clic sobre actualizar lo seleccionado. De esta forma se rellena el campo mucho más rápido. Al rellenar todos los objetos se puede eliminar el campo TIPO_0801.



	ETIQUETA	TIPO_0801	tipo_nom
51	Punta de Dins	080330	Playas
52	Punta de Fora	080330	Playas
53	Punta Ratjada	080330	Playas
69	Cap de S'Illa	080330	Playas
71	Penyes Roges	080330	Playas
141	Punta Blanca	080330	Playas

Fig. 34: Rellenar campos.

Solo queda un paso para tener preparada esta capa para el servidor. Generar los puntos de los recursos turísticos que faltan.

Se activa la edición de la capa y se localiza sobre la ortofoto el recurso que queremos introducir, ponemos un punto sobre el recurso y rellenamos los campos ETIQUETA y tipo_nom. Obteniendo esta capa puntual final.

Fig. 35: Capa de recursos turísticos resultado.

La localización de los distintos recursos, ya sean museos, iglesias, pozos, algunas playas, se han realizado desde el conocimiento del propio municipio y con la ayuda de la página web del Ajuntament de Santa Eulària des Riu, el cual proporciona información sobre patrimonio y turismo.

4.3. Estilos

Necesitamos obtener los estilos de las capas en formato .sld para subirlos a GeoServer junto a su capa correspondiente para que cada una se vea con el estilo que queramos. QGIS ofrece la posibilidad de exportar los estilos de las capas utilizadas en el proyecto en dicho formato, lo cual es muy interesante.

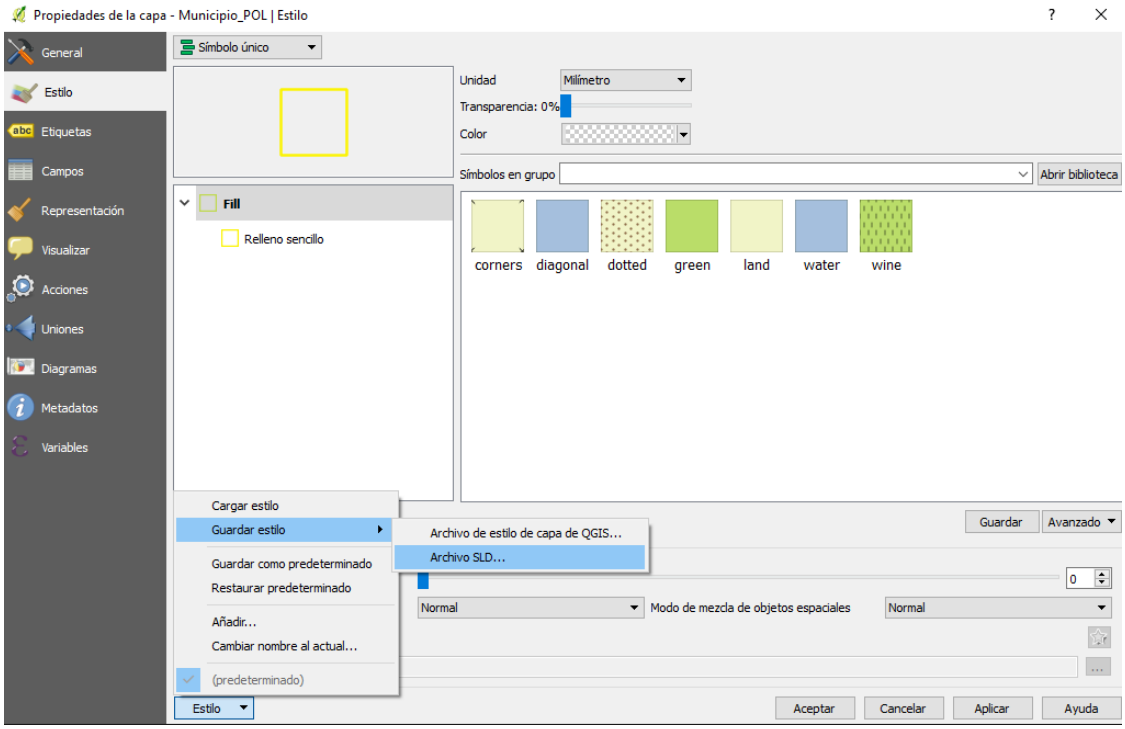


Fig. 36: Exportar estilos desde QGIS en sld.

Sin embargo, existe un pequeño error en la forma en la que QGIS exporta archivos .sld ya que aunque estos muestran correctamente detalles como el color, tipos de borde o rellenos, no incluyen información relacionada con la escala a la que queremos que se muestre cada capa o las etiquetas. Como la capa de recursos turísticos va a mostrar un campo etiqueta, es necesario un programa alternativo para obtener los estilos. La solución planteada es el programa AtlasStyler.

4.3.1. Creación de los estilos en AtlasStyler

AtlasStyler es un programa libre creado por Stefan Tzeggai que permite crear estilos y exportarlos en el formato .sld correctamente, tal y como queremos subirlos a GeoServer. Aunque el desarrollador ha abandonado el proyecto hace un tiempo, aún es posible descargar la aplicación y utilizarla para crear los archivos sld.

La aplicación funciona subiendo los archivos y seleccionando manualmente las condiciones de cada estilo. Tiene una opción de visualizar los estilos de forma simple, cuando el único objetivo es establecer una forma o color determinado o cambiar la visualización en función de un campo concreto.

Fig. 37: Ventana de edición de estilos en AtlasStyler.

Sin embargo, seleccionando estilos avanzados, podemos seleccionar detalles como el rango de las escalas en el que se va a mostrar. Además para añadir etiquetas se debe añadir a cada capa un tipo de estilo.

Los estilos con los que se visualizaran las rutas calculadas se crean desde QGIS, estableciendo una conexión WFS con geoserver cargamos las capas en el proyecto y creamos sus estilos en sld. El motivo de que tenga que crearse así es por un simple motivo, como la capa es una vista SQL, hasta que no se realiza la obtención de las coordenadas no se visualiza ninguna ruta, por ese motivo se debe dar estilo desde el WFS de GeoServer y no desde el archivo shp.

4.3.2. Características de los estilos

La siguiente imagen muestra con la petición GetLegendGraphic la leyenda de los recursos turísticos y se puede ver el estilo aplicado, son de estilo sencillo. En el visor web el punto se ve a cualquier escala, pero el nombre del recurso no aparece hasta que el usuario haga zoom hasta una escala 30.000, entonces el nombre aparece.

El límite del municipio no es más que un polígono sin relleno con un borde amarillo y los distintos tipos de rutas calculadas son de color rojo y azul.

Fig. 38: Estilo de la capa de recursos turísticos.

5. DESARROLLO DE LA GEODATABASE

5.1. Implementación de PostgreSQL y Postgis

El primer paso para implementar la GDB en el equipo servidor ha sido instalar PostgreSQL siguiendo los siguientes pasos:

1. Se ha descargado de la página web oficial de PostgreSQL (https://www.postgresql.org/)el ejecutable para Windows 64 bits con la versión 9.3.13 (ultima actualmente).

2. Una vez descargado el archivo se ha procedido a su instalación cumplimentando la información solicidata.

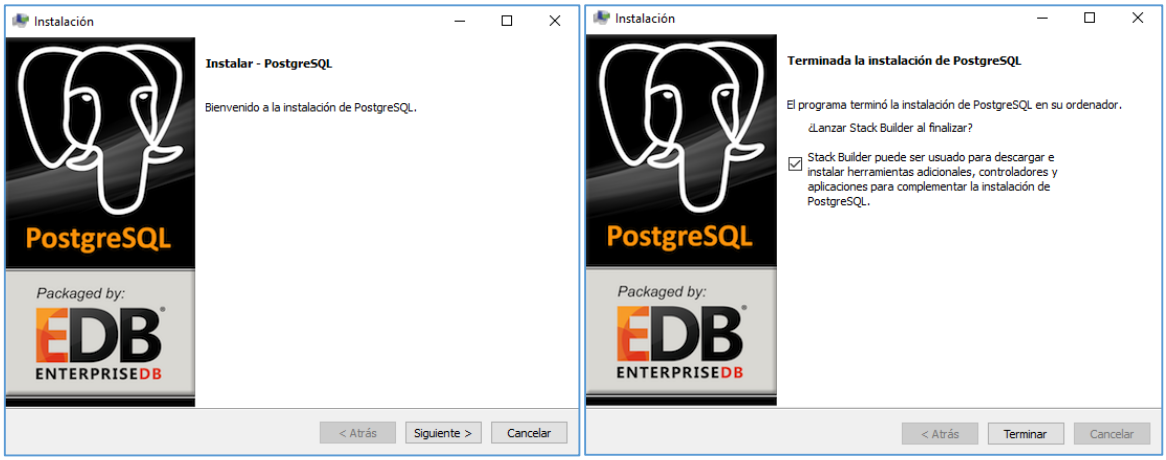


Fig. 39: Ventanas de inicio y final de la instalación.

3. Mediante la aplicación Stack Builder, que se instala junto a PostgreSQL, se pueden descargar e instalar herramientas adicionales, controladores y otros complementos para PostgreSQL. A través de esta se han instalado los diferentes controladores que permiten la conexión con otras aplicaciones.

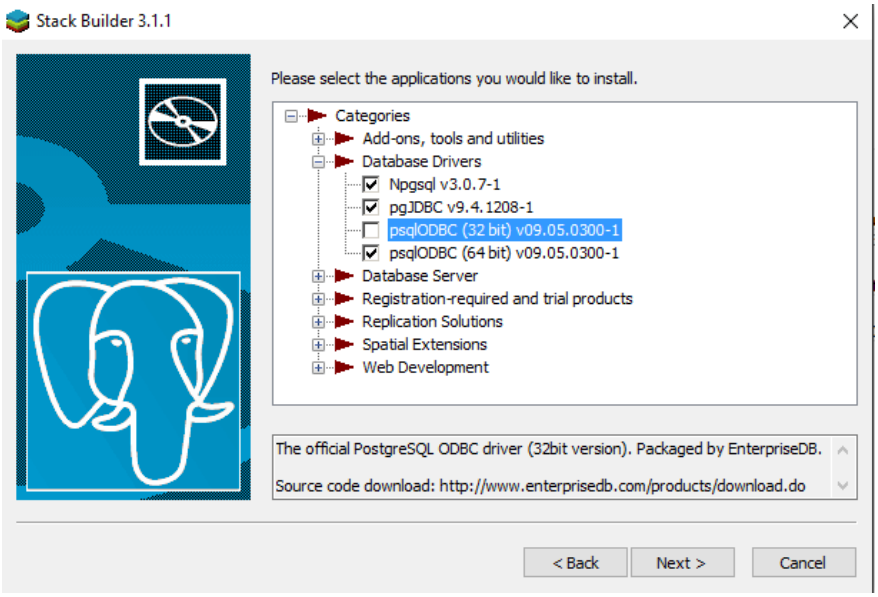


Fig. 40: Instalación de controladores a través de Stack Builder.

Una vez instalado PostgreSQL se ha instalado Postgis. Para ello se ha vuelto a utilizar la aplicación Stack Builder, la cual tiene en su apartado de Spatial Extensions la opción para instalar Postgis versión 2.0.6.

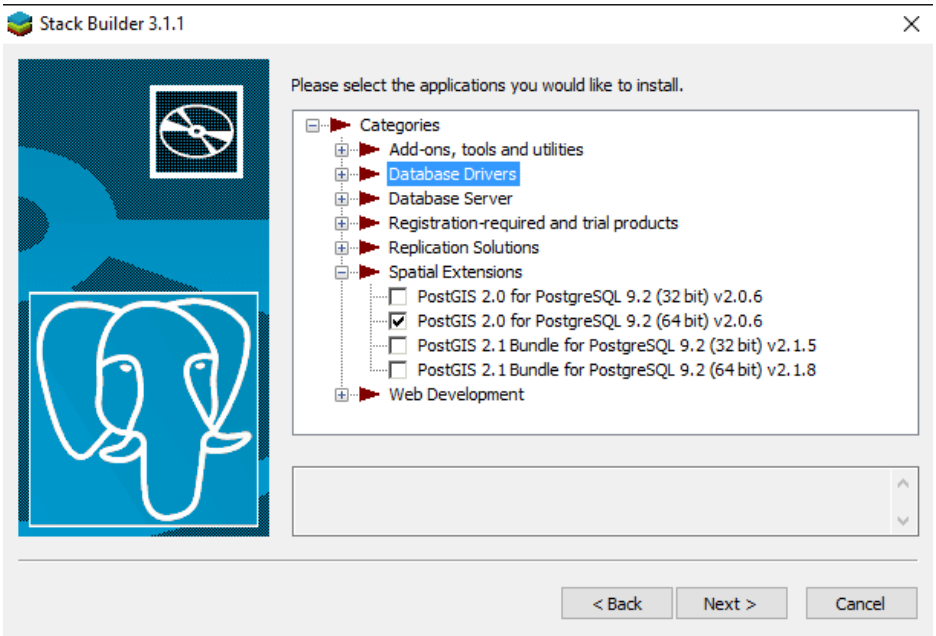


Fig. 41: Instalación de Postgis a través de Stack Builder

Una vez instalada la extensión Postgis sobre PostgreSQL ya tendremos la GDB en el equipo servidor. Su administración se realiza con la aplicación pgAdmin III, que se instala junto a PostgreSQL. Su interfaz gráfica facilita la gestión de la GDB.

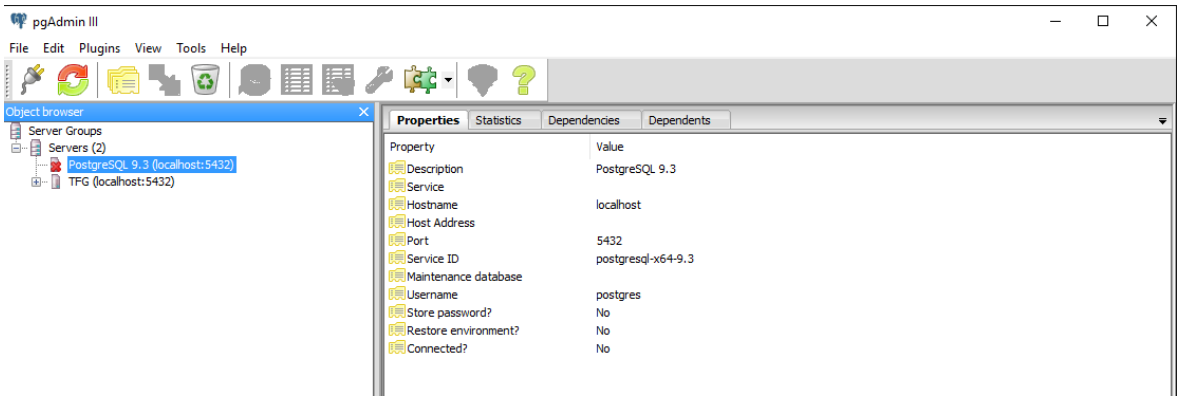


Fig. 42: Interfaz gráfica de pgAdmin III para la administración de PostgreSQL/Postgis.

5.2. Creación de la base de datos e implementación de la extensión pgRouting

Lo primero es iniciar pgAdmin III y crear un nuevo servidor local, para este caso el servidor será llamado TFG (Trabajo Final de Grado) para distinguirlo mejor de otros espacios de trabajo.

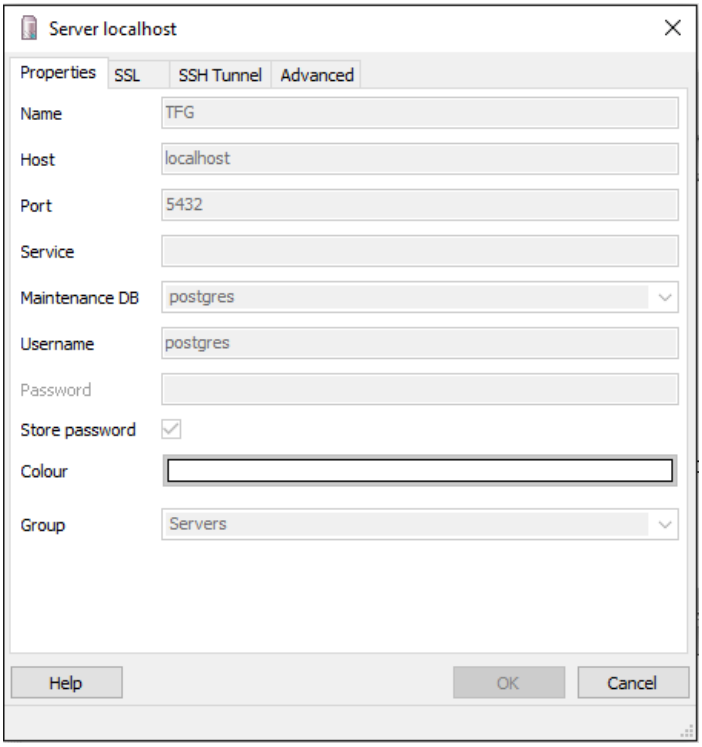


Fig. 43: Propiedades del nuevo servidor local.

Una vez creado el servidor local creamos una base de datos, llamada mydatabase, en la cual hay que cargar las extensiones postgis y pgrouting, tal y como muestra el siguiente código. Pgrouting puede ser descargado desde esta página http://pgrouting.org/download.html

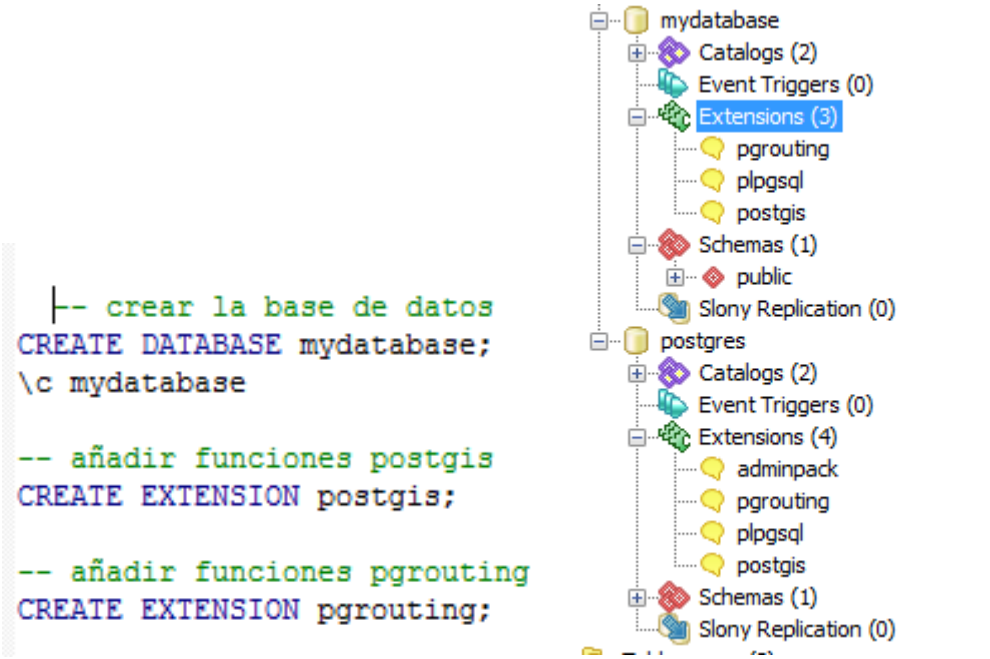


Fig. 44: Implementación de las extensiones postgis y pgrouting en la base de datos.

Una vez realizado este paso ya podemos empezar a trabajar con pgrouting en nuestra base de datos.

5.3. Creación de la red

5.3.1. Cargar los datos de la red

Para poder empezar a trabajar con pgRouting hay que importar a la base de datos la red, que es la capa de viales del municipio que se ha preparado anteriormente. Para simplificar la carga de esta capa se ha instalado un plugin en pgAdmin III, este plugin es Postgis shapefile and DBF loader 2.2.

Abrimos el plugin, se añade el archivo, buscándolo en el directorio donde lo guardamos, introducimos su sistema de coordenadas (EPGS:25831) y en opciones hay que seleccionar la opción de generar geometrías simples en vez de múltiples geometrías.

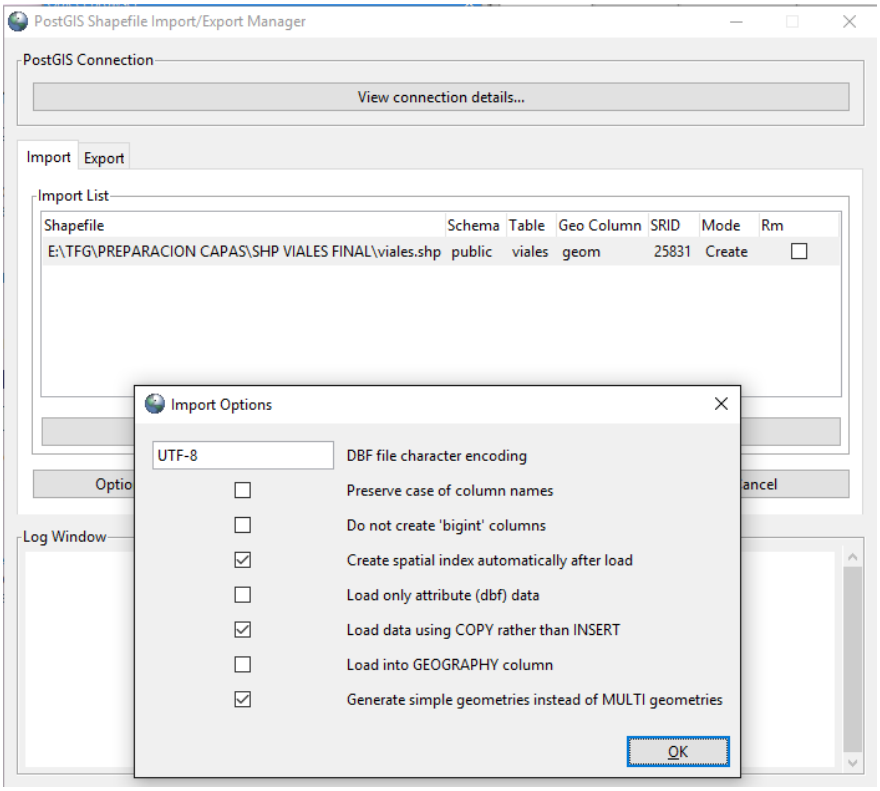


Fig. 45: Importar archivo shp a la base de datos.

Una vez importado se debe haber creado una nueva tabla llamada viales, dentro del apartado de tablas de la base de datos. No hace falta importar ningún archivo más.

En esta imagen se muestran los datos de la tabla viales que son un identificador único por enlace de carretera (gid), la descripción del tipo de vía (tipo_v_des), el tipo de via (tip_via_in), el nombre de la via (nom_via), longitud en kilómetros de las vías (shape_leng) y su geometría (geom, en versiones anteriores se llama the_geom).

```
gid serial NOT NULL,
tipo_v_des character varying(100),
tip_via_in character varying(25),
nom_via character varying(100),
shape_leng numeric,
geom geometry(LineString,25831),
```

Fig. 46: Datos de la tabla viales.

Esto permite visualizar la red de carreteras como una capa PostGIS en software GIS, por ejemplo en QGIS. Aunque no es suficiente para calcular rutas, ya que no contiene información topológica de la red.

5.3.2. Calcular la topología de la red

Para poder empezar a trabajar con pgRouting aún hace falta un paso más, hay que asegurarse que los datos proporcionan una topología de red valida o correcta, que consiste en información sobre la fuente (source) y el destino (target) de cada línea o enlace por carretera.

Si los datos de la red no disponen de dicha información, como es en este caso, se deberá ejecutar la función pgr_createTopology. Esta función asigna un origen (source) y un destino (target) a cada línea y se puede "ajustar" vértices cercanos dentro de una cierta tolerancia.

```
pgr_createTopology('<table>', float tolerance, '<geometry column',
'<gid>')
```

Primero se añaden las columnas de origen y destino, después se ejecuta la función pgr_createTopology y a esperar. Dependiendo del tamaño de la red el proceso podría tardar un par de minutos a horas. También requiere suficiente memoria ram para guardar datos temporales. Dentro del Query de pgAdmin se introduce el siguiente código:

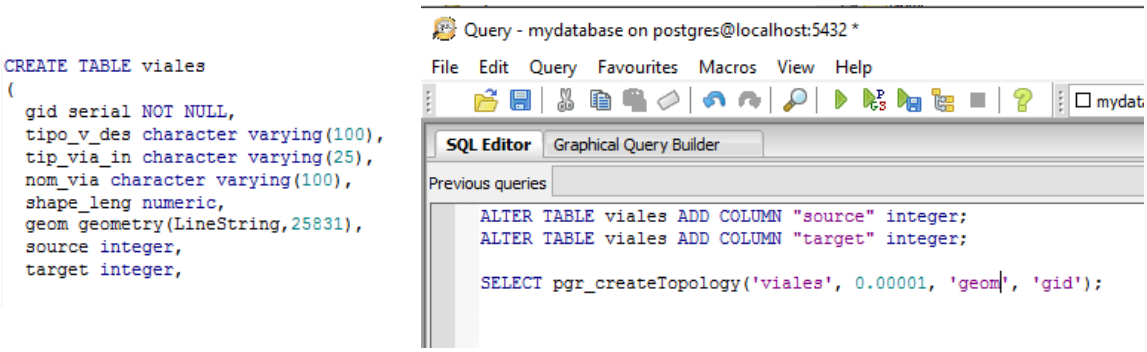


Fig. 47: Creación de los datos origen y destino dentro de la red.

Una vez realizado esto también se ha creado una nueva tabla que contiene los datos de origen y destino de la red (elementos puntuales)

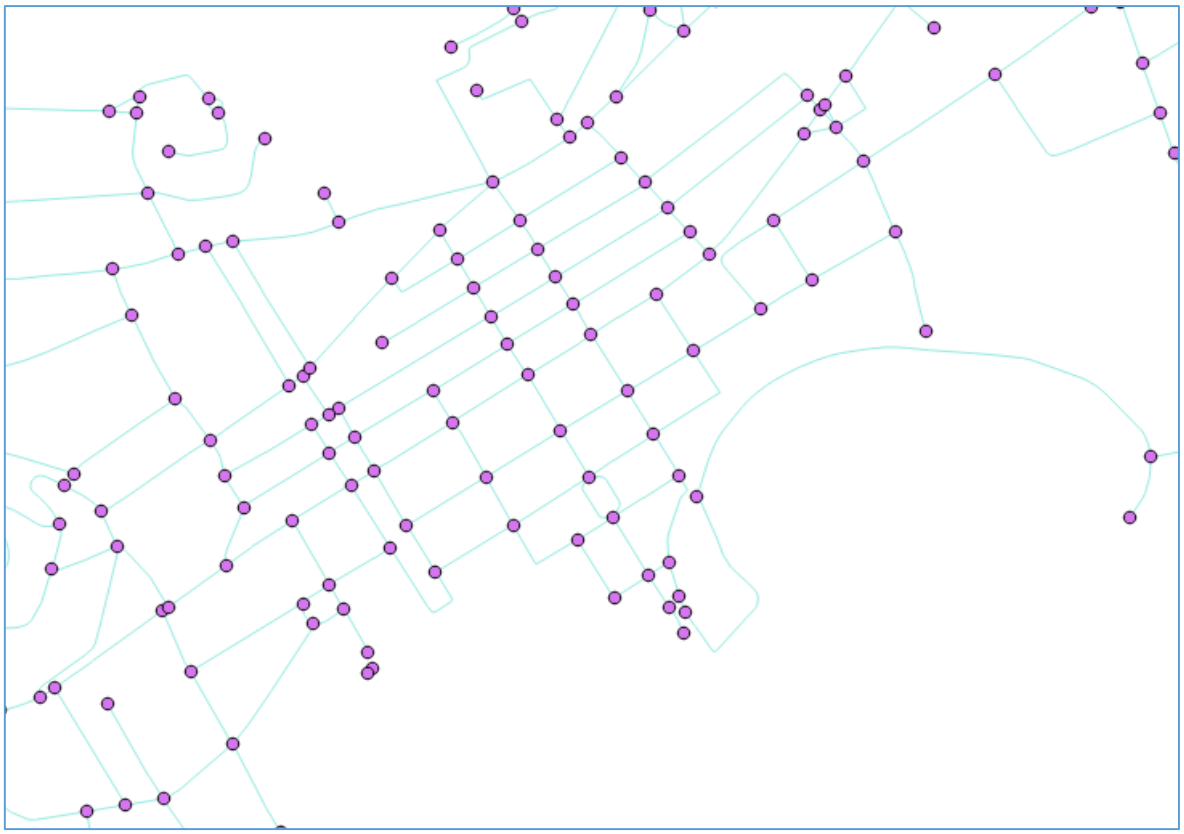


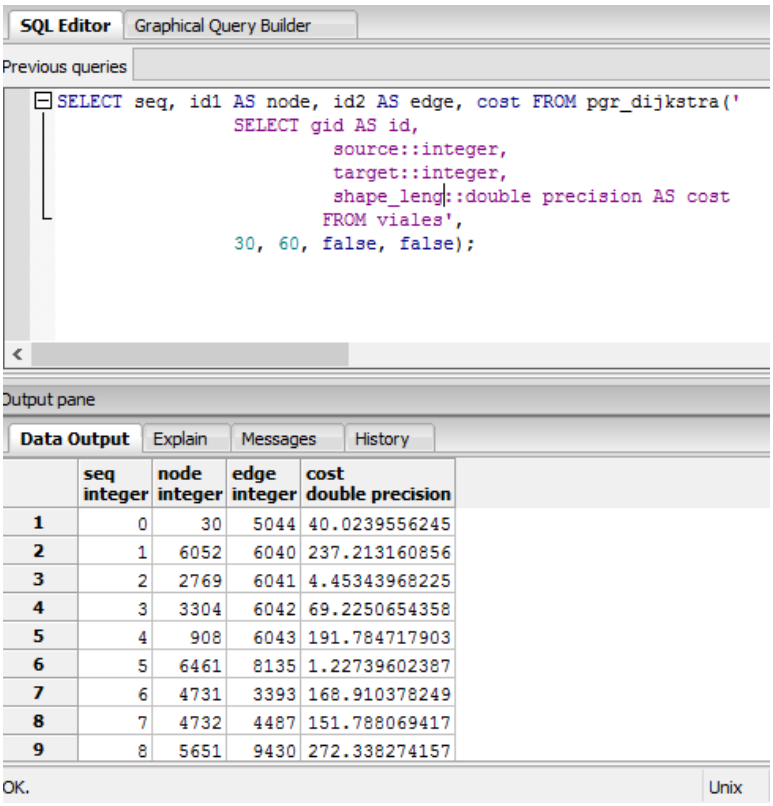
Fig. 48: Visualización de la red como una capa PostGis desde QGIS.

A modo de comprobación se va a utilizar dos funciones de pgRouting para calcular la ruta más corta y así asegurar que la red está correctamente creada.

Dijktra

La función Dijktra calcula la ruta más corta entre un punto de origen (source) y un punto destino (target) de la propia red, aparte de requerir esos dos datos también necesita el gid y un coste. El coste hay muchas formas de definirlo pero para este caso concreto será la distancia en kilómetros (shape_leng).

Ejemplo del cálculo de ruta con la función Dijktra:



	seq integer	node integer	edge integer	cost double precision
1	0	30	5044	40.0239556245
2	1	6052	6040	237.213160856
3	2	2769	6041	4.45343968225
4	3	3304	6042	69.2250654358
5	4	908	6043	191.784717903
6	5	6461	8135	1.22739602387
7	6	4731	3393	168.910378249
8	7	4732	4487	151.788069417
9	8	5651	9430	272.338274157

Fig. 49: Calculo de ruta más corta con la función Dijktra.

En el panel de salida, si se toma la primera fila, se obtiene la secuencia de la fila 0, el identificador del nodo 60, identificación del borde 5044 (-1 para la última fila) y el coste del traslado desde 60 a 5044. Esta función no devuelve una geometría, devuelve únicamente una lista ordenada de nodos.

Camino más corto A*

A-Star es un algoritmo de enrutamiento conocido. Añade información geográfica al origen y destino de cada línea de la red. Esto permite la consulta de enrutamiento a partir del nodo más cercado al objetivo de la búsqueda en el cálculo del camino más corto.

Para poder utilizar esta función se requiere de una preparación previa, hay que añadir las columnas de coordenadas, donde x1, y1 son las coordenadas del punto de inicio de la arista y x2, y2 son las coordenadas del punto extremo del borde.

```
ALTER TABLE viales ADD COLUMN x1 double precision;
ALTER TABLE viales ADD COLUMN y1 double precision;
ALTER TABLE viales ADD COLUMN x2 double precision;
ALTER TABLE viales ADD COLUMN y2 double precision;

UPDATE viales SET x1 = ST_x(ST_PointN(geom, 1));
UPDATE viales SET y1 = ST_y(ST_PointN(geom, 1));

UPDATE viales SET x2 = ST_x(ST_PointN(geom, ST_NumPoints(geom)));
UPDATE viales SET y2 = ST_y(ST_PointN(geom, ST_NumPoints(geom)));
```

Output pane

Data Output | Explain | **Messages** | History

```
Query returned successfully: 9600 rows affected, 1704 ms execution time.
```

Fig. 50: Creación de las columnas X1, Y1 y X2, Y2 y cálculo de sus valores.

La función de la ruta más corta A-Star es muy similar a la función Dijkstra, aunque esta busca los nodos más cercanos al punto de búsqueda.

Para realizar el cálculo con A-Star se realiza de la siguiente forma.

```
SELECT seq, id1 AS node, id2 AS edge, cost FROM pgr_astar('
                SELECT gid AS id,
                         source::integer,
                         target::integer,
                         shape_leng::double precision AS cost,
                         x1, y1, x2, y2
                        FROM viales',
                30, 60, false, false);
```

Output pane

Data Output | Explain | Messages | History

	seq integer	node integer	edge integer	cost double precision
1	0	30	5044	40.0239556245
2	1	6052	6040	237.213160856
3	2	2769	6041	4.45343968225
4	3	3304	6042	69.2250654358
5	4	908	6043	191.784717903
6	5	6461	8135	1.22739602387
7	6	4731	3393	168.910378249
8	7	4732	4487	151.788069417
9	8	5651	9430	272.338274157

Fig. 51: Calculo de ruta más corta con la función A-Star.

Si este resultado lo guardamos en una tabla y generamos una vista en QGIS se obtiene una ruta entre dos puntos:

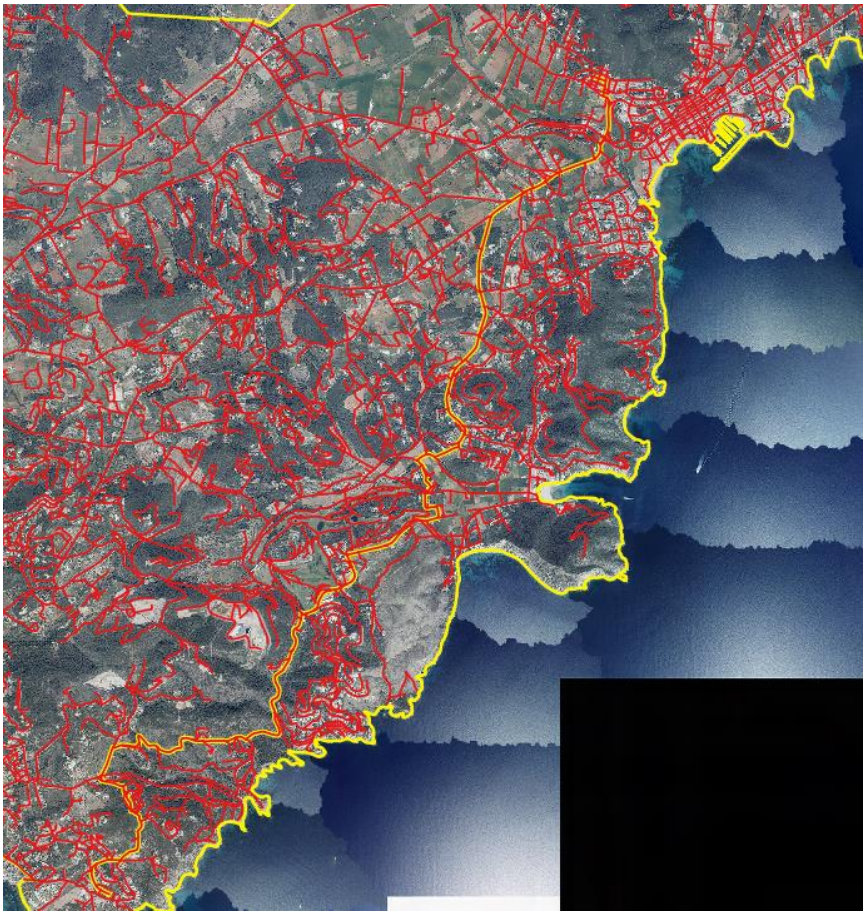


Fig. 52: Visualización de una ruta desde QGIS.

Con esta vista por medio de QGIS se comprueba que la topología de la red está correctamente creada.

5.4. Creación de una función para el cálculo de rutas

Muchas funciones de pgRouting proporcionan una interfaz de bajo nivel de los algoritmos por ejemplo devuelven identificadores ordenados en vez de geometrías de rutas. Las funciones wrapper ofrecen diferentes parámetros de entrada, así como transformar el resultado devuelto en un formato que puede ser más fácil de leer para las aplicaciones.

La desventaja de este tipo de funciones es que a menudo hacen suposiciones que solo son útiles en ciertos casos de usos específicos. Para ello pgRouting ha decidido únicamente admitir funciones de bajo nivel y dejar al usuario escribir sus propias funciones para sus propios casos de uso.

Por lo tanto la función que hay que desarrollar tiene que cumplir ciertos puntos. Primero que reconozca el punto de origen y destino (x1, y1, x2, y2) que introduzca el usuario y segundo que sea capaz de calcular la ruta más rápida y la más corta. Para calcular la ruta más corta se puede utilizar la función Dijktra, de tal forma que la función busque el nodo de la ruta más cercano al origen elegido por el usuario y el nodo más cercano al destino elegido por el usuario. Por otro lado, para calcular la ruta más rápida, hay que tener en cuenta los puntos introducidos por el usuario y la velocidad de cada tipo de vía.

Para poder implementar la velocidad en la red hay que tomar el dato de coste (shape_leng) y el tipo de vial, porque no es lo mismo recorrer 100km a una velocidad de 40km/h que 100km a 80km/h. Así que una solución sencilla seria añadir un multiplicador al dato de coste.

Consultando los datos de la red se han dividido los viales en tres tipos:

- Caminos, con una velocidad media entre 30-50km/h dependiendo del estado del camino, se opta por unos 40 km/h.

- Carreteras, con tramos máximos de 80 km/h y algunos tramos de 60 km/h, se opta por 80 km/h.

- Calles, la velocidad máxima en zona urbana de todo el municipio es de 50 km/h.

Teniendo en cuenta todo esto, los caminos tendrán su coste original, las carreteras tendrán su coste original multiplicado por 0.5, reduciendo así su coste y las zonas urbanas su coste original multiplicado por 0.8, reduciendo, pero en menor medida, su coste.

Se va a crear una nueva columna en la tabla de viales llamada coste_vel (coste en función de la velocidad) y actualizar sus valores tomando el coste original por el multiplicador ya explicado, obteniendo así una tabla de este estilo.

gid	tipo_v_des	tip_via_in	nom_via	shape_leng	coste_vel	source	target	x1	y1	x2	y2
6541	Camino	-998	NULL	50.8800245682	50.8800245682	1972	4602	371153.9589999...	4318343.801999...	371133.0649999...	4318390.193999...
6662	Camino	-998	NULL	7.41064342749	7.41064342749	6342	2833	360867.442999853	4318946.543999...	360872.9489998...	4318951.503999...
5363	Camino	-998	NULL	22.4482459226	22.4482459226	6245	5468	362722.7119998...	4320458.591999...	362745.0759998...	4320456.648999...
6819	Camino	-998	NULL	27.6473957449	27.6473957449	6436	3614	363780.4411998...	4311960.107899...	363785.7079998...	4311987.24899989
5364	Camino	-998	NULL	8.18597837711	8.18597837711	6246	2087	372719.1009999...	4314016.914999...	372710.9399999...	4314017.55399993
40	Camino	-998	NULL	7.16361375248	7.16361375248	358	360	370213.2429998...	4318380.831999...	370210.9039998...	4318374.060999...
370	Vía urbana	CARRE	MESTRAL	20.4123793808	16.32990350464	759	779	363940.6179998...	4310441.7179999	363941.69399987	4310462.101999...
413	Vía urbana	CARRE	25	47.4072909182	37.92583273456	824	263	369693.1074998...	4310337.69789992	369694.9899998...	4310290.327999...
432	Vía urbana	PL	TITI	6.09300927217	4.874407417736	765	734	371252.8719999...	4312796.066999...	371258.6129999...	4312794.025999...
5365	Camino	-998	NULL	5.1633679888	5.1633679888	2360	3162	367571.7159998...	4310390.322999...	367568.2039998...	4310394.107999...
5366	Vía urbana	CTRA	PM-810	19.8895586925	9.94477934625	6247	2216	376922.9769999...	4323738.823999...	376938.7379999...	4323726.691999...
5367	Camino	-998	NULL	13.3560201404	13.3560201404	4334	6248	377040.0489999...	4322643.896999...	377047.613999923	4322654.903999...
5387	Camino	-998	NULL	7.80047754938	7.80047754938	1925	1926	362007.478999859	4318382.584999...	362015.2679998...	4318382.161999...
5368	Camino	-998	NULL	6.55375373342	6.55375373342	1262	6249	370836.6449999	4318135.234999...	370834.9029999	4318141.552999...
2745	Camino	-998	NULL	2.19439825208	2.19439825208	99	2897	371767.425999904	4321712.804999...	371767.937299904	4321714.938999...
2839	Camino	-998	NULL	3.0578234639	3.0578234639	3443	4117	361583.5869998...	4318643.756999...	361583.2281998...	4318646.793699...
3786	Camino	-998	NULL	4.63443711139	4.63443711139	5085	5086	367380.4221998...	4314707.34029991	367380.6719998...	4314711.967999...

Tabla 2: Tabla para cálculo de rutas acabada.

Ahora que ya está acabada la tabla se va a explicar la función implementada para cálculo de ruta más corta y cálculo de ruta más rápida. La función calcula la ruta entre dos puntos y devuelve la geometría ordenada. Por lo tanto, los parámetros de entrada serán las coordenadas que seleccione el usuario y devuelve una ruta que se puede mostrar en los servicios QGIS o WMS como GeoServer.

Lo que hace la función:

- Parámetros de entrada: nombre de la taba, punto de inicio y punto final.

```
IN tbl varchar,
IN x1 double precision,
IN y1 double precision,
IN x2 double precision,
IN y2 double precision,
```

- Columnas de salida: una secuencia (para ordenar los resultados), gid (para vincular los resultados de nuevo a la tabla original), la dirección (el acimut entre los dos puntos), el coste (para la ruta más corta en kilómetros y para la más rápida en función de la velocidad) y la geometría de la ruta.

```
OUT seq integer,
OUT gid integer,
OUT heading double precision,
OUT cost double precision,
OUT geom geometry
```

- La función debe encontrar los nodos más cercanos al punto inicial introducido y al punto final introducido.

```
EXECUTE 'SELECT id::integer FROM viales_vertices_pgr
        ORDER BY the_geom <-> ST_GeometryFromText(''POINT('
        || x1 || ' ' || y1 || ')'',25831) LIMIT 1' INTO rec;
source := rec.id;

EXECUTE 'SELECT id::integer FROM viales_vertices_pgr
        ORDER BY the_geom <-> ST_GeometryFromText(''POINT('
        || x2 || ' ' || y2 || ')'',25831) LIMIT 1' INTO rec;
```

- Obtenidos los nodos se ejecuta la ruta más corta de Dijkstra (dependiendo del valor de coste que se introduzca calculara la más corta o la más rápida). Donde pone shape_leng (ruta más corta) cambiarlo por coste_vel (ruta más rápida) y renombrar la función. Así se obtienen dos funciones una para la ruta más corta (CalcularRuta) y otra para la ruta más rápida (CalcularRutaRapida).

```
seq := 0;
sql := 'SELECT gid, geom, cost, source, target,
        ST_Reverse(geom) AS flip_geom FROM ' ||
                'pgr_dijkstra(''SELECT gid as id, source::int, target::int, '
                                || 'shape_leng::float AS cost FROM '
                                || quote_ident(tbl) || ''', '
                                || source || ', ' || target
                                || ' , false, false), '
                        || quote_ident(tbl) || ' WHERE id2 = gid ORDER BY seq';
```

- Luego calcular el azimut desde el principio hasta el final de cada nodo de la red de carreteras.

```
EXECUTE 'SELECT degrees( ST_Azimuth(
        ST_StartPoint(''' || rec.geom::text || '''),
        ST_EndPoint(''' || rec.geom::text || ''') ) )'
    INTO heading;
```

- Y finalmente devolver el resultado como un conjunto de resultados.

```
        seq     := seq + 1;
        gid     := rec.gid;
        cost    := rec.cost;
        geom    := rec.geom;
        RETURN NEXT;
END LOOP;
RETURN;
```

Lo que no hace la función:

- No restringe la red de carreteras seleccionadas al BBOX (necesario para las grandes redes).
- No devuelve clases de carreteras u otros atributos.
- No toma en cuenta las calles de un único sentido.

Para crear la función se introduce el siguiente código en el Query de pgAdmin, seguido de todo el código de la función.

```
CREATE OR REPLACE FUNCTION pgr_CalcularRuta(
```

Realizando una consulta desde pgAdmin y visualizando el resultado en QGIS se obtiene lo siguiente:

SQL Editor | Graphical Query Builder

Previous queries | Delete | Delete

```
SELECT * FROM pgr_CalcularRuta('viales',372995.638,4316103.936,375783.391,4321511.002);
```

Output pane

Data Output | Explain | Messages | History

	seq integer	gid integer	heading double precision	cost double precision	geom geometry
1	1	6769	.3179061164103	26.0252247645	01020000
2	2	6768	.2198712863647	86.8488295927	01020000
3	3	6767	.7688387411968	83.2813940381	01020000
4	4	6766	.081187931263	71.0704266768	01020000
5	5	6765	.8027673636849	71.9010199689	01020000
6	6	6764	.6615687539562	63.597750753	01020000
7	7	6030	.2546238113554	144.379970678	01020000

Fig. 53: Consulta sobre la función creada.

Fig. 54: Visualización en QGIS de la ruta calcula desde la función creada.

El código completo de la función aparece en los anexos.

6. PREPARACIÓN DE GEOSERVER Y CARGA DE DATOS

6.1. Instalación de GeoServer en el servidor Apache TomCat

Se procede con la instalación de la versión 7 de TomCat, junto con sus extensiones docs, manager y ejemplos. La más importante es la Manager, que nos permite acceder a la pantalla de TomCat donde se desplegará GeoServer.

TomCat se instala inicialmente sin un listado de usuarios que puedan acceder a su Manager, por lo que se debe buscar el fichero control de usuarios y editarlo, añadiéndonos con el usuario y contraseña con el que deseemos acceder al programa. Este listado se encuentra en C:\Tomcat 7.0\conf\tomcat-users.

El niel de usuario que permite tener acceso al Manager y añadir los diferentes programas es el manager-gui, por lo que hay que añadir una cuenta con el nombre y contraseña que se desee y ese nivel de acceso, será el utilizado a partir de ahora para acceder a la aplicación.

```
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="tomcat" roles="tomcat,manager-gui"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>

</tomcat-users>
```

Fig. 55: Edición de los usuarios de acceso a TomCat.

Para poder instalar GeoServer es necesario realizar un cambio más y es aumentar la memoria de los archivos con los que el Manager de TomCat puede trabajar. Es necesario ampliar el valor de estas etiquetas para poder instalar GeoServer dentro de TomCat. C:\Tomcat 7.0\webapps\manager\WEB-INF\web.xml

```
<!-- 50MB max -->
<max-file-size>73400320</max-file-size>
<max-request-size>73400320</max-request-size>
<file-size-threshold>0</file-size-threshold>
```

Fig. 56: Aumento de la memoria de archivos.

GeoServer se instala dentro de Apache TomCat. Para ello, se descarga el archivo .war de su página web (http://geoserver.org/release/stable/), se descarga la versión 2.9.1.

Aplicaciones					
Trayectoria	Versión	Nombre a Mostrar	Ejecutándose	Sesiones	Comandos
/	Ninguno especificado	Welcome to Tomcat	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/docs	Ninguno especificado	Tomcat Documentation	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/geoserver	Ninguno especificado	GeoServer	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/manager	Ninguno especificado	Tomcat Manager Application	true	1	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos

Fig. 57: Aplicaciones del Manager de TomCat.

Se puede acceder a GeoServer a través del propio Manager de TomCat o directamente a través de la URL http://localhost:8080/geoserver/.

6.2. Carga de capas en GeoServer

Una vez instalado GeoServer, iniciamos sesión directamente con la cuenta de administrador definida por defecto. El primer paso será crear dos espacios de trabajo, uno para la base de datos (pgRouting) y un segundo para la cartografía. Al primer espacio de trabajo se ha llamado pgrouting, y al segundo capasvisor. Se dejan el resto de valores por defecto y guardamos. Dentro del espacio de trabajo de pgrouting se han rellenado los apartados para aportar información de contacto.

Fig. 58: Creación del espacio de trabajo.

La información introducida de contacto se mostrara al realizar una petición GetCapabilities.

Dentro del espacio de trabajo para la cartografía (capas visor) se deben crear almacenes de capas para poder crear después las propias capas. Los almacenes de capas contienen la información geográfica en su formato original hasta su publicación en el

servidor. Hay que elegir en que formato se encuentra la capa e indicarle donde se encuentra y como obtenerla.

Para el caso de las capas de límite municipal y topónimos, se selecciona el formato shapefile. Pero para la ortofoto es necesario instalar la extensión GDAL para poder seleccionar el formato ECW, ya que GeoServer por defecto no tiene ese formato.

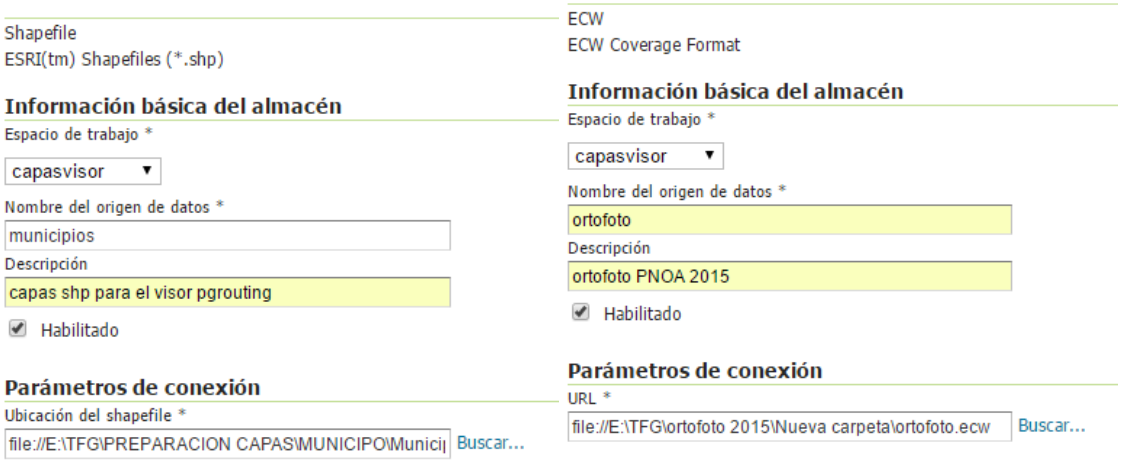


Fig. 59: Importar capas en SHP y ECW.

Para el almacén de datos de pgRouting no se seleccionara un formato, sino origen de datos de PostGist Database. En el cual se deben establecer los parámetros de la conexión con la base de datos.

Fig. 60: Origen de datos PostGis.

Al realizar la conexión se obtienen las capas que hay en la base de datos, en este caso son dos viales y viales_vertices_pgr (índice source/target).

De esta forma se han definido los origines de los datos que se van a requerir para generar el visor web.

Tipo de datos	Espacio de trabajo	Nombre del almacén	Tipo	¿Habilitado?
	capasvisor	municipio	Shapefile	✔
	capasvisor	ortofoto	ECW	✔
	pgrouting	pgrouting	PostGIS	✔
	capasvisor	toponimia	Shapefile	✔

Fig. 61: Listado de almacenes de capas.

Las capas ortofoto, municipio y topónimos se publican de igual forma, en cambio pgrouting no, en vez de crear un nuevo feature type se crea una nueva vista SQL. De esta forma llama a la función de calcular la ruta e introduce en la función las coordenadas, que son sus parámetros de entrada y con st_makeline crea una línea (la ruta). Con esta misma configuración se genera una segunda vista SQL pero en vez de llamar a la función de cálculo de ruta más corta, llamara a la función de ruta más rápida.

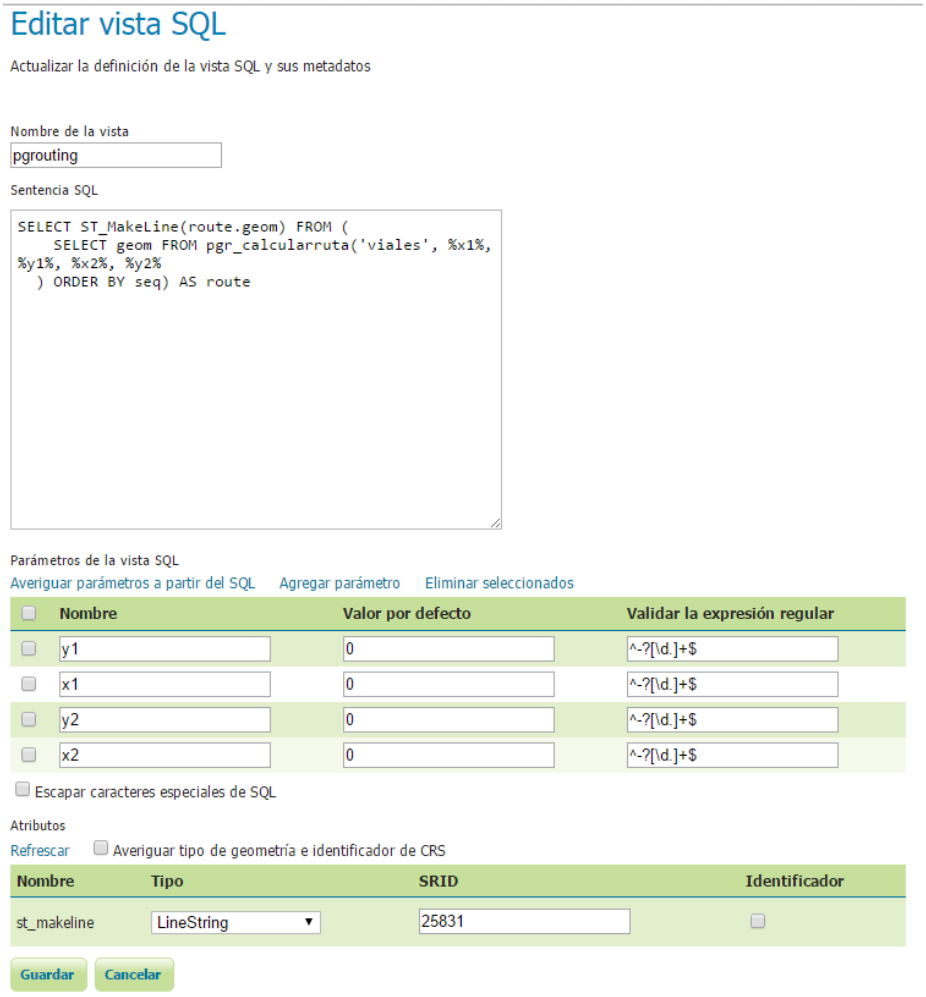


Fig. 62: Configuración de la vista SQL.

Tipo	Title	Nombre de la capa	Almacén	Habilitada?	SRS nativo
	pgrouting	pgrouting:pgrouting	pgrouting	✓	EPSG:25831
	pgroutingrapida	pgrouting:pgroutingrapida	pgrouting	✓	EPSG:25831
	toponimia	capasvisor:toponimia	toponimia	✓	EPSG:25831
	Municipio_POL	capasvisor:Municipio_POL	municipio	✓	EPSG:25831
	ortofoto	capasvisor:ortofoto	ortofoto	✓	EPSG:25831

Fig. 63: Capas publicadas en GeoServer.

6.3. Carga de los estilos de las capas

En la pestaña estilos añadimos a la aplicación los diferentes estilos que se han creado previamente en QGIS y AtlasStyler, en formato sld. Para ello, simplemente se debe seleccionar cada archivo sld y cargarlo (subir) a la aplicación. El programa se encarga de importarlo y solamente deberemos seleccionar nuestro espacio de trabajo y confirmar que es correcto. GeoServer nos da la opción de validad el código para comprobar si existe algún error en él. Si el estilo esta generado en su totalidad por AtlasStyler no dará ningún error, pero si esta genererado desde QGIS, no hará falta validar puesto que siempre dará algún error.

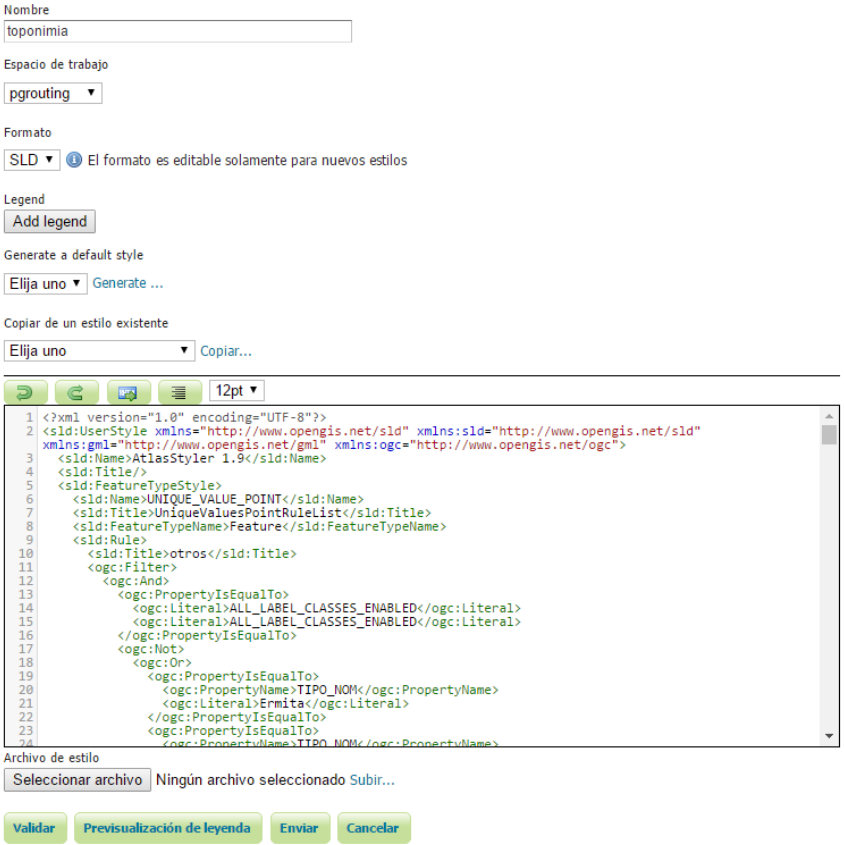


Fig. 64: Carga de estilos en Geoserver.

En el código se puede observar claramente información sobre la capa, color o relleno. En las capas en las que se ha decidido mostrar como etiquetas la información de alguno de sus atributos se puede ver como el código SLD contiene además información acerca del tamaño o tipo de letra, anchura del halo o del desplazamiento relativo al elemento.

Por ultimo solo resta asignar cada uno de los estilos con su correspondiente capa. Para ellos, se entra dentro de la pestaña de capas, se selecciona una de ellas y se accede al apartado de publicación, donde se puede seleccionar el tipo de estilo con el que se desea mostrar la capa. Desde el propio Geoserver existe una opción de previsualización de capas con el estilo seleccionado.

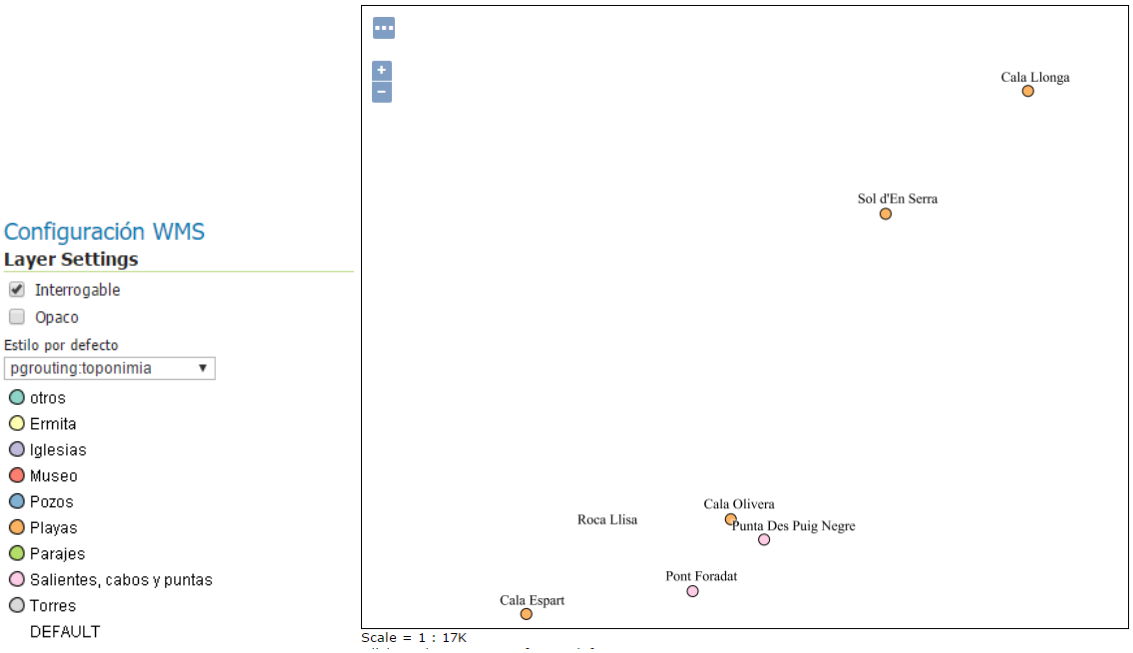


Fig. 65: Previsualización de la capa de recursos turísticos.

Se han previsualizado todas las capas para comprobar que los estilos han sido asignados correctamente.

7. DESARROLLO DEL VISOR WEB DE CALCULO DE RUTAS

7.1. Funcionamiento del visor

Un usuario interactúa con el visor accediendo desde su navegador web vía internet o intranet a las paginas HTML que lo forman, aunque en este caso va a ser un acceso local. Estas están almacenadas en el servidor y son publicadas a través del servidor. Las solicitudes de mapas que un usuario realiza a través de la aplicación web (el visor) a GeoServer se realizan vía OpenLayers.

7.2. Plantilla HTML

La plantilla HTML es el elemento que permite interactuar con el usuario. A través de esta se realizan las peticiones de mapas, las cuales son enviadas a Geoserver (GetMap) y devueltas al usuario incrustadas en el código HTML de la plantilla para ser visualizadas en su navegador web. Al archivo plantilla se le ha nombrado de la misma forma plantilla.html. A continuación se explica las peticiones GetMap que tiene y como permite al usuario introducir los puntos que delimitaran la ruta.

Para empezar la petición GetMap es realizada conjuntamente con OpenLayers como muestra el siguiente código, este código se repite para las capas de ortofoto, municipio y topónimos, la diferencia entre estas será la capa que vaya dentro de la petición GetMap.

```
//wms ortofoto PNOA
var wmsOrtofoto = new ol.layer.Image({
    source: new ol.source.ImageWMS({
        url: 'http://localhost:8080/geoserver/capasvisor/wms',
        params: {
                'LAYERS': 'capasvisor:ortofoto',
                'STYLES': '',
                'VERSION': '1.1.1',
                'FORMAT': 'image/png',
        }
    })
```

Fig. 66: Petición GetMap para la capa de la ortofoto.

```
//wms poligono del municipio
var wmsMunicipio = new ol.layer.Image({
    source: new ol.source.ImageWMS({
        url: 'http://localhost:8080/geoserver/capasvisor/wms',
        params: {
            'LAYERS':'capasvisor:Municipio_POL',
            'STYLES':'',
            'VERSION':'1.1.1',
            'FORMAT':'image/png',
            },
        })
    });
```

Fig. 67: Petición GetMap para la capa del municipio.

```
//wms playas
var wmstoponimos = new ol.layer.Image({
    source: new ol.source.ImageWMS({
        url: 'http://localhost:8080/geoserver/capasvisor/wms',
        params: {
            'LAYERS':'capasvisor:recursosturisticos',
            'STYLES':'',
            'VERSION':'1.1.1',
            'FORMAT':'image/png',
            },
        })
    });
```

Fig. 68: Petición GetMap para la capa de topónimos.

Seguidamente se genera la proyección EPSG:25831 y se crea el mapa con su vista.

```
var projection = new ol.proj.Projection({
    code: 'EPSG:25831',
    units: 'm',
    axisOrientation: 'neu'
});
//creacion mapa
var map = new ol.Map({
    controls: ol.control.defaults().extend([
        new ol.control.ScaleLine()
    ]),
    target:'map',
    view: new ol.View({
        projection: projection,
    })
});
```

Fig. 69: Creación del mapa y su proyección.

A continuación está el código para registrar los clics del usuario y generar las rutas (petición GetMap a una vista SQL con ST_MakeLine, devuelve una línea).

```
// Punto origuen y destino
var startPoint = new ol.Feature();
var destPoint = new ol.Feature();

// Capa vectorial para mostar el punto origen y destino
var vectorLayer = new ol.layer.Vector({
    source: new ol.source.Vector({
        features: [startPoint, destPoint]
    })
});
```

Fig. 70: Mostrar puntos definidos por el usuario.

Este código genera las variables de punto inicial y punto final, y los muestra en el mapa utilizando la librería de OpenLayers 3.

```
map.on('click', function(event) {
    if (startPoint.getGeometry() == null) {
// Primer clic
        startPoint.setGeometry(new ol.geom.Point(event.coordinate));
    } else if (destPoint.getGeometry() == null) {
// Segundo clic
        destPoint.setGeometry(new ol.geom.Point(event.coordinate));
//Obtencion de coordenadas

            var startCoord = startPoint.getGeometry().getCoordinates();
            var destCoord = destPoint.getGeometry().getCoordinates();
```

Fig. 71: Obtención de las coordenadas.

Esta parte del código identifica el evento de clic sobre el mapa y da las coordenadas a las variables creadas anteriormente. Hay un condicional para que la función reconozca cuando se realiza el segundo clic, que es el que especifica el destino o punto final.

```
        var startCoord = startPoint.getGeometry().getCoordinates();
        var destCoord = destPoint.getGeometry().getCoordinates();
        //params ruta mas corta
        var params = {
                'LAYERS': 'pgrouting:pgrouting',
                'FORMAT': 'image/png',
                VERSION: '1.1.1'
            };
        //params ruta mas rapida
        var params2 = {
                'LAYERS': 'pgrouting:pgroutingrapida',
                'FORMAT': 'image/png',
                VERSION: '1.1.1'
            };
        var viewparams = [
            'x1:' + startCoord[0], 'y1:' + startCoord[1],
            'x2:' + destCoord[0], 'y2:' + destCoord[1]
        ];
```

Fig. 72: Parámetros para la petición GetMap.

Aquí, las variables almacenan las coordenadas y pasan a formar parte de los parámetros que se utilizaran en la petición GetMap, puesto que las coordenadas x1, y1, x2, y2 son las variables que requiere la función de calcular la ruta, que está establecida en la base de datos.

```
params.viewparams = viewparams.join(';');
resultado = new ol.layer.Image({
        source: new ol.source.ImageWMS({
            url: 'http://localhost:8080/geoserver/pgrouting/wms',
            params: params
        })
    })
map.addLayer(resultado);
//rapida
params2.viewparams = viewparams.join(';');
resultado2 = new ol.layer.Image({
        source: new ol.source.ImageWMS({
            url: 'http://localhost:8080/geoserver/pgrouting/wms',
            params: params2
        })
    })
map.addLayer(resultado2);
```

Fig. 73: Petición GetMap para el cálculo de ruta.

Aquí se muestran las dos peticiones GetMap que envían las coordenadas a la vista SQL creada anteriormente para obtener y mostrar la ruta en el mapa.

Al final del código se ha añadido un pequeño botón que permite borrar las dos rutas que se muestran y así poder realizar otra consulta y obtener más rutas. Las dos rutas (corta y rápida) se visualizan al mismo tiempo con colores distinto para poder comparar los dos resultados obtenidos.

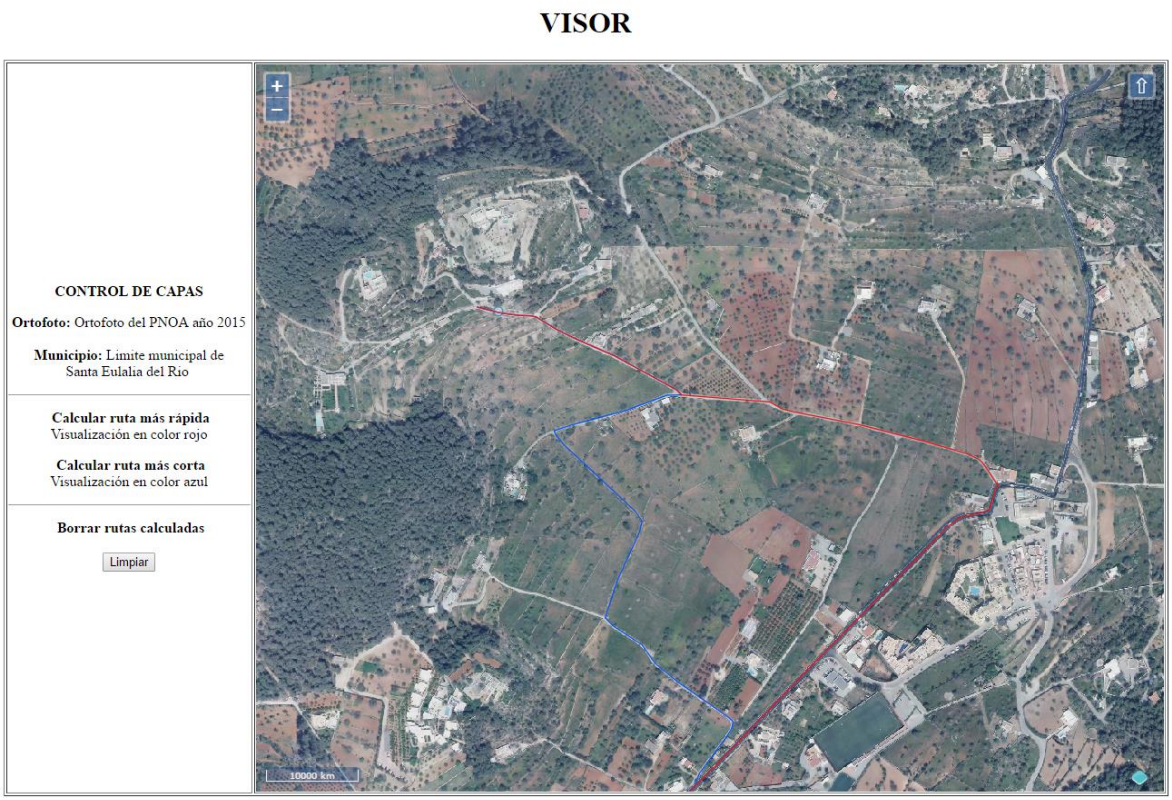


Fig. 74: Visor web de cálculo de rutas.

Aunque el visor tenga un estilo sencillo, el cual no es objetivo de este proyecto, se ha conseguido el objetivo de este proyecto, que era el conseguir un visor funcional con OpenLayers 3 que permitiera al usuario introducir dos puntos para trazar una ruta entre estos.

El código completo del visor está en el Anexo II.

8. CONCLUSIONES

Se ha demostrado que no es necesario el uso de aplicaciones de pago o con licencia temporal para ejecutar este proyecto. Todas las tareas realizadas, ya sea preparar la cartografía junto con sus estilos, crear el servidor, desarrollar el visor web y la base de datos se han desarrollado con aplicaciones de libre distribución. Ofreciendo un buen resultado para este proyecto. Además muchas instituciones públicas utilizan estos programas.

Se ha comprobado que es posible utilizar lenguajes de programación como JavaScript para producir aplicaciones geoespaciales, como el visualizador en OpenLayers. La limitación en estas funciones está marcada por la propia limitación dentro del conocimiento del programador. No obstante se debe tener en cuenta estos lenguajes dada a su gran versatilidad y ayuda por parte de las distintas comunidades que los sostienen.

Me ha sorprendido que a día de hoy no se encuentren todos los caminos digitalizados de Ibiza en cartociudad o BTN, puesto que en estos años todo se está informatizando e incluido en una base de datos.

Aunque en este proyecto se ha conseguido trazar dos tipos de rutas, pgRouting ofrece muchas más opciones, no solo existen esas dos. Como ya se ha comentado en este proyecto, pgRouting proporciona pequeñas funciones de cálculo de rutas, es el usuario el que las adapta a sus necesidades y las amplía. Lo más seguro es que todas las posibilidades giren en torno a crear una base de datos con un diseño lógico relacional el cual enlace tablas de calles urbanas, para así obtener sus direcciones (unidireccional), con tablas con los diferentes caminos y carreteras para tener en cuenta las variaciones de velocidad máxima que existen en este municipio.

En este proyecto no se ha realizado nada parecido puesto que mis conocimientos de postgis son bastante limitados. No obstante ya había trabajado en la realización de inventarios para el ayuntamiento donde resido.

9. LÍNEAS FUTURAS

Este proyecto no es otra cosa que una demostración y prueba para una aplicación real y futra. Actualmente he realizado el inventariado de viviendas diseminadas del municipio de Santa Eulària des Riu, durante mi periodo de trabajo en el Ayuntamiento.

La idea original de este proyecto consistía en trazar rutas hacia viviendas diseminadas para proporcionar servicios públicos, sea policía, ambulancias, bomberos, etc., un modo fácil, rápido y seguro de acceder a dichas viviendas. Dado que mucha gente cuando llama a estos servicios tiene que dar una explicación compleja de cómo llegar a su vivienda, como por ejemplo, “coger la carretera dirección Santa Eulària – Ibiza y al pasar de largo la gasolinera tomar el primer desvió a mano derecha, el cual es un camino de tierra y girar en el segundo cruce donde hay una piedra pintada de azul” esa es una de las muchas definiciones que te puede dar un residente que reside en una vivienda diseminada. Pero con un trazador de rutas y un inventario de viviendas te ahorras este tipo de explicaciones un tanto confusas, que para un caso de urgencias en vez de ayudar causan problemas.

Con este proyecto se ha demostrado la posibilidad de trazar rutas con el software que se dispone en el ayuntamiento, solo faltaría depurar las condiciones, es decir, que tenga en cuenta las direcciones únicas, definir mas tipo se carreteras y caminos (mas caminos de velocidad) y obviamente la única ruta que importaría en estos casos, sería la ruta más rápida.

El visor online actual del ayuntamiento se conecta a una base de datos postgis al igual que este proyecto, la diferencia que existe es que se han decantado por utilizar MapServer en vez de GeoServer.

Básicamente espero que en un futuro no muy lejano se puedan calcular rutas desde el visor web Geoxarc.

10. BIBLIOGRAFÍA Y REFERENCIAS

- José C. Martínez Llario. (2012-2013). *PostGis2 Análisis Espacial Avanzado*.
- Santiago, Antonio (2015) *The book of OpenLayers 3*. *Leanpub.*
- Gabriel Huecas, Ignacio Vázquez Zapata, Juan Quemada Vives, Joaquín Salvachúa Rodríguez, Eugenio Vega Pinado y Santiago Pavon. *Curso Desarrollo en HTML5, CSS y Javascript de WebApps, incl. Móviles FirefoxOS* (3ªed).* https://miriadax.net/web/html5mooc
- Anita Graser (2013). *Public transport isochrones with pgRouting. Qgis Planet.* http://planet.qgis.org/planet/tag/pgrouting/
- PosgreSQL Wiki. https://wiki.postgresql.org/wiki/Main_Page
- pgRouting Manual (2.0.0). http://docs.pgrouting.org/2.0/es/doc/index.html.
- pgRouting Workshops. http://workshop.pgrouting.org/
- GeoServer Documentation. http://docs.geoserver.org/
- Open Source Geosp3atial Foundation (2015), *Geoserver User manual.*
- OpenLayers 3. *Tutorials.* http://openlayers.org/en/latest/doc/tutorials/
- OpenLayers 3. *API Docs.* http://openlayers.org/en/latest/apidoc/
- Boundlessgeo (2013) *How to publish GDAL/MrSID image formats on a production Geoserver on Windows.* http://boundlessgeo.com/2013/03/how-to-publish-gdalmrsid-image-formats-on-a-production-geoserver-on-windows/

11. ANEXOS

ANEXO I: Código de la función de cálculo de rutas.

```
--
--FUNCION pgr_CalcularRuta(varchar, double precision, double precision,
--                          double precision, double precision);

CREATE OR REPLACE FUNCTION pgr_CalcularRuta(
                IN tbl varchar,
                IN x1 double precision,
                IN y1 double precision,
                IN x2 double precision,
                IN y2 double precision,
                OUT seq integer,
                OUT gid integer,
                OUT heading double precision,
                OUT cost double precision,
                OUT geom geometry
        )
        RETURNS SETOF record AS
$BODY$
DECLARE
        sql     text;
        rec     record;
        source  integer;
        target  integer;
        point   integer;

BEGIN
    -- Buscar el nodo mas cercano
    EXECUTE 'SELECT id::integer FROM viales_vertices_pgr
            ORDER BY the_geom <-> ST_GeometryFromText(''POINT('
            || x1 || ' ' || y1 || ')'',4258) LIMIT 1' INTO rec;
    source := rec.id;

    EXECUTE 'SELECT id::integer FROM viales_vertices_pgr
            ORDER BY the_geom <-> ST_GeometryFromText(''POINT('
            || x2 || ' ' || y2 || ')'',4258) LIMIT 1' INTO rec;
    target := rec.id;

    -- Camino mas corto (TODO: limite de la extensión by BBOX)
        seq := 0;
        sql := 'SELECT gid, geom, cost, source, target,
                ST_Reverse(geom) AS flip_geom FROM ' ||
                        'pgr_dijkstra(''SELECT gid as id, source::int, target::int, '
                                        || 'shape_leng::float AS cost FROM '
                                        || quote_ident(tbl) || ''', '
                                        || source || ', ' || target
                                        || ' , false, false), '
                                || quote_ident(tbl) || ' WHERE id2 = gid ORDER BY seq';

    -- Guardar punto de inicio
        point := source;

        FOR rec IN EXECUTE sql
        LOOP
        -- Dar la vuelta a la geometria (si es necesario)
        IF ( point != rec.source ) THEN
            rec.geom := rec.flip_geom;
            point := rec.source;
        ELSE
            point := rec.target;
        END IF;
```

```
61
62          -- Calcular la dirección (simplificado)
63          EXECUTE 'SELECT degrees( ST_Azimuth(
64                  ST_StartPoint(''' || rec.geom::text || '''),
65                  ST_EndPoint(''' || rec.geom::text || ''') ) )'
66              INTO heading;
67
68          -- Devolver registros
69                  seq     := seq + 1;
70                  gid     := rec.gid;
71                  cost    := rec.cost;
72                  geom    := rec.geom;
73                  RETURN NEXT;
74          END LOOP;
75          RETURN;
76  END;
77  $BODY$
78  LANGUAGE 'plpgsql' VOLATILE STRICT;
```

ANEXO II: Código del visor web html.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Plantilla</title>
        <meta charset="utf8">
        <link rel="stylesheet" href="http://openlayers.org/en/v3.0.0/css/ol.css" type="text/css">
        <!--<script src="http://openlayers.org/en/v3.15.1/build/ol.js" type="text/javascript"></script>-->
        <script src="http://localhost:8080/geoserver/openlayers3/ol.js" type="text/javascript"></script>
    </head>
    <body>
        <center>
            <h1>VISOR</h1>
            <table class="normal" border="1px">
                <tr>
                    <td align="center">
                        <p> <b> CONTROL DE CAPAS </b> </p>
                        <p>
                             <b>Ortofoto:</b> Ortofoto del PNOA año 2015 </br>
                            </br>
                             <b>Municipio:</b> Limite municipal de </br>Santa Eulalia del Rio 
                        </p>
                        <hr size="1">
                        <p>
                             <b>Calcular ruta más rápida</b></br>
                            Visualización en color rojo
                        </p>
                        <p>
                             <b>Calcular ruta más corta</b></br>
                            Visualización en color azul
                        </p>
                        <hr size="1">
                        <p>
                             <b>Borrar rutas calculadas</b></br></br>
                            <button id="limpiar" class="pure-button">Limpiar</button>
                        </p>
                    </td>
                    <td>
                        <div style="width:1000px; height:800px" id="map"></div>
                        <script>
                            var extent = [359916.3745, 4308205.3435,
                                          383164.849, 4325429.351];
                            //wms ortofoto PNOA
                            var wmsOrtofoto = new ol.layer.Image({
                                source: new ol.source.ImageWMS({
                                    url: 'http://localhost:8080/geoserver/capasvisor/wms',
                                    params: {
                                            'LAYERS': 'capasvisor:ortofoto',
                                            'STYLES': '',
                                            'VERSION': '1.1.1',
                                            'FORMAT': 'image/png',
                                    }
                                })
                            });
                            //wms poligono del municipio
                            var wmsMunicipio = new ol.layer.Image({
                                source: new ol.source.ImageWMS({
                                    url: 'http://localhost:8080/geoserver/capasvisor/wms',
                                    params: {
                                            'LAYERS':'capasvisor:Municipio_POL',
                                            'STYLES':'',
                                            'VERSION':'1.1.1',
                                            'FORMAT':'image/png',
                                            },
                                    })
                                });
                            //wms playas
                            var wmstoponimos = new ol.layer.Image({
                                source: new ol.source.ImageWMS({
                                    url: 'http://localhost:8080/geoserver/capasvisor/wms',
                                    params: {
                                            'LAYERS':'capasvisor:toponimia',
                                            'STYLES':'',
                                            'VERSION':'1.1.1',
                                            'FORMAT':'image/png',
                                            },
                                    })
                                });
```

```
                        //seleccion proyeccion UTM etrs89 huso31
                        var projection = new ol.proj.Projection({
                            code: 'EPSG:25831',
                            units: 'm',
                            axisOrientation: 'neu'
                        });
                        //creacion mapa
                        var map = new ol.Map({
                            controls: ol.control.defaults().extend([
                                new ol.control.ScaleLine()
                            ]),
                            target:'map',
                            view: new ol.View({
                                projection: projection,
                            })
                        });
                        map.getView().fit(extent, map.getSize());
                        map.addLayer(wmsOrtofoto);
                        map.addLayer(wmsMunicipio);
                        //map.addLayer(wmstoponimos);

 //trazar la ruta en el mapa ------------------------------------------------------------------------------

                        // Punto origuen y destino
                        var startPoint = new ol.Feature();
                        var destPoint = new ol.Feature();

                        // Capa vectorial para mostar el punto origen y destino
                        var vectorLayer = new ol.layer.Vector({
                            source: new ol.source.Vector({
                                features: [startPoint, destPoint]
                            })
                        });
                        map.addLayer(vectorLayer);

                        // A transform function to convert coordinates from EPSG:25831
                        // to EPSG:4258.
                    //  var transform = ol.proj.getTransform('EPSG:25831', 'EPSG:4258');

                        // Registrar un clic en el mapa

                        map.on('click', function(event) {
                            if (startPoint.getGeometry() == null) {
                        // Primer clic
                                startPoint.setGeometry(new ol.geom.Point(event.coordinate));
                            } else if (destPoint.getGeometry() == null) {
                        // Segundo clic
                                destPoint.setGeometry(new ol.geom.Point(event.coordinate));
                        //Obtencion de coordenadas

                                    var startCoord = startPoint.getGeometry().getCoordinates();
                                    var destCoord = destPoint.getGeometry().getCoordinates();
                                    //params ruta mas corta
                                    var params = {
                                            'LAYERS': 'pgrouting:pgrouting',
                                            'FORMAT': 'image/png',
                                            VERSION: '1.1.1'
                                        };
                                    //params ruta mas rapida
                                    var params2 = {
                                            'LAYERS': 'pgrouting:pgroutingrapida',
                                            'FORMAT': 'image/png',
                                            VERSION: '1.1.1'
                                        };
                                    var viewparams = [
                                        'x1:' + startCoord[0], 'y1:' + startCoord[1],
                                        'x2:' + destCoord[0], 'y2:' + destCoord[1]
                                    ];
                                    //corta
                                    params.viewparams = viewparams.join(';');
                                    resultado = new ol.layer.Image({
                                            source: new ol.source.ImageWMS({
                                                url: 'http://localhost:8080/geoserver/pgrouting/wms',
                                                params: params
                                            })
                                        })
                                    map.addLayer(resultado);
                                    //rapida
                                    params2.viewparams = viewparams.join(';');
                                    resultado2 = new ol.layer.Image({
                                            source: new ol.source.ImageWMS({
                                                url: 'http://localhost:8080/geoserver/pgrouting/wms',
                                                params: params2
                                            })
                                        })
                                    map.addLayer(resultado2);
                                }
                        });
```

```
                                    map.addLayer(resultado);
                                    //rapida
                                    params2.viewparams = viewparams.join(';');
                                    resultado2 = new ol.layer.Image({
                                            source: new ol.source.ImageWMS({
                                                url: 'http://localhost:8080/geoserver/pgrouting/wms',
                                                params: params2
                                            })
                                        })
                                    map.addLayer(resultado2);
                                }
                            });

                            //Añadir boton limpiar
                            var clearButton = document.getElementById('limpiar');
                            clearButton.addEventListener('click', function(event) {
                            // Reinicia el punto origen y destino
                                startPoint.setGeometry(null);
                                destPoint.setGeometry(null);
                            // Limpia el resultado de la capa
                                map.removeLayer(resultado);
                                map.removeLayer(resultado2);
                            });
                        </script>
                    </td>
                </tr>
            </table>
        </center>
    </body>
</html>
```