Escuela Politécnica Superior de Ávila
Máster en Geotecnologías Cartográficas
en Ingeniería y Arquitectura

Trabajo Final de Master:

Desarrollo de una herramienta de captura de datos para inventario de infraestructura sobre un dispositivo móvil y complemento para extracción y visualización formato geográfico KML.

Autor:
CESAR BERNARDO CORTES RODRIGUEZ

Directores:
Diego González Aguilera y José Antonio Martin

Ávila – España
Junio de 2021

TABLA DE CONTENIDO

RESUMEN 5

ABSTRACT 5

ESTADO DEL ARTE 7

OBJETIVOS DE TRABAJO PROPUESTO 8

MEDIOS EMPLEADOS 8

Entorno de Trabajo 8

Plataforma de Desarrollo 8

Otros recursos para implementación 9

FLUJO DEL PROCESO DE TRABAJO 9

Captura de información en campo 9

Configuración inicial QGIS – QFIELD 10
Creación de una carpeta de trabajo 10
Herramienta de Sincronización en el móvil 10
Complemento de sincronización en QGis 10
Trabajo de campo en QField 11
Descarga del trabajo de campo 11
Sincronización de carpetas de trabajo con el proyecto en centro de operaciones 11

Procesamiento y consolidación en Oficina 11

CONFORMACIÓN DEL ARCHIVO BASE DE TRABAJO 11

Desarrollo Del Formulario De Captura De Datos 11

El formato de inventario 11
Modelo de datos 13
Conformación del Geopackage y niveles de información 13
Creación del Geopackage: 13
Creación de la capa principal de edición: 14
Creación y Asociación de las tablas de dominios: 14
Parametrización de dominios para cada atributo de la tabla: 16
Parametrización de Valores por defecto: 16
Formulario para captura de datos 16
Conformación del Mapa Base de trabajo 18

Aspectos relevantes a tener en cuenta con la generación del MBTiles 19

CONFIGURACIÓN DE ARCHIVOS DE TRABAJO EN EL MÓVIL 19

Instalación de aplicaciones en el dispositivo móvil 19
Configuración de Drive Sync 20
Configuración QField 20

COMPLEMENTO DE SINCRONIZACIÓN DE QGIS 21

EJECUCIÓN DEL PROCESO DE CAPTURA Y SINCRONIZACION 23

Captura en campo 23
Apertura del documento de proyecto 24
Edición y creación de elementos QFIELD 24

Sincronización de información de campo 26
Sincronización móvil – Google Drive 26
Sincronización del Celular a la carpeta compartida Google Drive 27
Consolidación y información de archivo geográfico 27

Extracción de archivo "csv" para generación de KML 28

COMPLEMENTO PARA CONVERSIÓN A KML 28

Estructura del código del complemento 29

Apertura y carga de archivo " csv " 32

Conversión KML por "Tipo Estructura" 34
Visualización de imagen de referencia 37

Conversión a KML por "Estado de la estructura" 40

CREACION DE REPOSITORIO DE DATOS EN GITHUB 43

RESULTADOS 44

DESARROLLOS FUTUROS 45

BIBLIOGRAFÍA 47

ANEXOS 49

Diccionario Estructura principal de datos "InvEstructura" 49

Código de complemento para conversión KML 50
Header "ventanaprincipal.h" 50
Header "puntoscampo.h" 50

Header “generar_kml.h” 51
Header “generar_kml2.h” 51
Header “acercatfm.h” 52
Sources “ventanaprincipal.cpp” 52
Sources “puntoscampo.cpp” 56
Sources “main.cpp” 57
Sources “generar_kml.cpp” 58
Sources “generar_kml2.cpp” 61
Sources “acercatfm.cpp” 64
Principal del proyecto “Kml_conv.pro” 65
Forms ventanaprincipal .ui 65

TABLA DE ILUSTRACIONES

ILUSTRACIÓN 1 - FLUJO DE PROCESO PARA CAPTURA EN CAMPO DE LA INFORMACIÓN 10
ILUSTRACIÓN 2 - ESQUEMA DE IDENTIFICACIÓN DE ELEMENTOS Y ATRIBUTOS 12
ILUSTRACIÓN 3 - FORMULARIO CON PARAMETRIZACIÓN DE ALCANCE 12
ILUSTRACIÓN 4 - ESTRUCTURA DEL MODELO DE DATOS NIVEL PRINCIPAL GEOGRÁFICO 13
ILUSTRACIÓN 5 - ESTRUCTURA DE DATOS CAPA PRINCIPAL 14
ILUSTRACIÓN 6 - EJEMPLO DE TABLA PARA DOMINIO DE DATOS 15
ILUSTRACIÓN 7 - VENTANA DE CREACIÓN DE CAPA NO ESPACIAL EN QGIS 15
ILUSTRACIÓN 8 - VISUALIZACIÓN DE LA ESTRUCTURA DE DATOS COMPLETA EN GEOPACKAGE 15
ILUSTRACIÓN 9 - ASIGNACIÓN DE DOMINIOS POR RELACIÓN DE VALORES. 16
ILUSTRACIÓN 10 - FORMULARIO NATIVO QGIS 17
ILUSTRACIÓN 11 - FORMULARIO GENERADO CON QT DESIGNER 17
ILUSTRACIÓN 12 - DEFINICIÓN DE "ALIAS" PARA ATRIBUTOS 18
ILUSTRACIÓN 13 - DEFINICIÓN DE VALORES POR DEFECTO 18
ILUSTRACIÓN 14 - DEFINICIÓN DE PARÁMETRO PARA TOMA DE REGISTRO FOTOGRÁFICO CON EL MÓVIL 18
ILUSTRACIÓN 15 - CAPA BASE RASTER, PARA REFERENCIACIÓN OFFLINE 19
ILUSTRACIÓN 16 - CONFIGURACIÓN ALOJAMIENTO LOCAL ARCHIVOS DE TRABAJO CON DRIVE SYNC 20
ILUSTRACIÓN 17 - CONFIGURACIÓN DE PROYECTO PARA SINCRONIZACIÓN EN QGIS-QFIELD 21
ILUSTRACIÓN 18 - DEFINICIÓN DE ACCIONES PARA CAPAS 22
ILUSTRACIÓN 19 - CONFIGURACIÓN DE PROYECTO PARA SINCRONIZACIÓN QFIELD 22
ILUSTRACIÓN 20 - CONFIGURACIÓN DE DIRECTORIOS DE ENTRADA Y SALIDA DE ARCHIVOS 23
ILUSTRACIÓN 21 - APERTURA DE PROYECTO EN QFIELD 24
ILUSTRACIÓN 22 - DEFINICIÓN DEL MODO DE EDICIÓN PARA INICIO DE CAPTURA 25
ILUSTRACIÓN 23 - MODO DE EDICIÓN QFIELD 25
ILUSTRACIÓN 24 - VENTANA PARA INGRESO DE ATRIBUTOS 26
ILUSTRACIÓN 25 - PROCESO DE SINCRONIZACIÓN DESDE MÓVIL 27
ILUSTRACIÓN 26 - GENERACIÓN DE DOCUMENTO DE TRABAJO QGIS 28
ILUSTRACIÓN 27 - VENTANA DE COMPLEMENTO PARA CONVERSIÓN DE ARCHIVO A FORMATO KML 29
ILUSTRACIÓN 28 - MENÚ DE UTILIZACIÓN DE LA APLICACIÓN 29
ILUSTRACIÓN 29 - MENSAJE DE ERROR POR ESTRUCTURA DE ARCHIVO " CSV " 30
ILUSTRACIÓN 30 - VENTANA DE CARGA EXITOSA DE FICHERO " CSV ". 30
ILUSTRACIÓN 31 - ESTRUCTURA DE ARCHIVOS QUE HACEN PARTE DEL COMPLEMENTO 31
ILUSTRACIÓN 32 - SIMBOLOGÍA PARA ESTRUCTURAS CON TUBERÍA “TUBOPASANT” 35
ILUSTRACIÓN 33 - SIMBOLOGÍA PARA ESTRUCTURAS TIPO BOX, PUENTE O PONTÓN 35
ILUSTRACIÓN 34 - ESTRUCTURA DEL ENCABEZADO DE ARCHIVO KML. 36
ILUSTRACIÓN 35 - APARTE DE ARCHIVO ESTRUCTURADO KML - ATRIBUTOS E IMAGEN 38

ILUSTRACIÓN 36 - VISUALIZACIÓN DE PUNTO POR TIPO ESTRUCTURA _____39
ILUSTRACIÓN 37 - VENTANA EN GOOGLE EARTH DE ARCHIVO KML GENERADO POR TIPO DE ESTRUCTURA _____39
ILUSTRACIÓN 38 - SIMBOLOGÍA PARA ESTRUCTURAS POR NOVEDAD DE ESTADO _____40
ILUSTRACIÓN 39 - APARTE DE ARCHIVO ESTRUCTURADO KML POR TIPO DE ESTRUCTURA - ATRIBUTOS E IMAGEN__42
ILUSTRACIÓN 40 - VISUALIZACIÓN DE PUNTO POR ESTADO DE LA ESTRUCTURA _____42
ILUSTRACIÓN 41 - VENTANA EN GOOGLE EARTH DE ARCHIVO KML GENERADO POR ESTADO ESTRUCTURA _____43
ILUSTRACIÓN 42 - REPOSITORIO GITHUB CON EL CÓDIGO FUENTE DE CONVERTIDOR CSV A KML._____44
ILUSTRACIÓN 43 - DICCIONARIO DE DATOS "INVESTRUCTURA" _____49

RESUMEN

Con la realización de diferentes trabajos de campo que requieren de agilizar y emplear las herramientas que impliquen facilidad y eficiencia en la ejecución así como la reducción de costos y tiempos de procesamiento de la información, se planteó implementar una herramienta que permitiera la captura de datos para inventarios de infraestructuras hidráulica en campo, para tal efecto en la investigación previa se identificó un número importante de aplicaciones propietarias que prestan servicios similares, no obstante también se encontró el complemento QField desarrollado para QGIS que permite mediante desarrollo de un formulario en QT o directamente desde la aplicación lograr esta implementación de manera “sencilla” por lo que siendo esta una aplicación GNU que viene en constante evolución y requiere una programación mínima para su implementación se determinó que ésta era la mejor ajustada al planteamiento inicial del problema planteado. Dificultades adicionales igualmente serán resueltas con la ejecución de este proyecto como el tema de la sincronización de los datos de campo desde diferentes operarios la cual también se realizará con herramientas ya disponibles y conocidas por la mayoría de usuarios de tecnología, buscando si con el desarrollo de este trabajo simplificar la ejecución y hacer posible la utilización de la herramienta de manera rápida, eficaz y económica para la realización de cualquier trabajo.

Adicionalmente dadas las pruebas de concepto del problema, se encontró que si bien es cierto QGIS permite exportar a en un formato KML éste no es modificable en su presentación, y para el usuario final de la información podrá ser irrelevante una parte de ella y necesaria otra, hecho que no es posible con la exportación directa desde QGI, razón por la que se trabajo un aplicativo que permita la exportación y conversión del archivo principal según la estructura definida en el GeoPackage, a un formato KML de visualización personalizada que permitirá conformar un archivo final para la presentación al usuario final por programación en C++.

Todo lo anterior, no es más que la aplicación de diferentes conceptos y herramientas que fueron expuestas y estudiadas durante la ejecución de la Maestría en las materias de:

- Herramientas informáticas para el Geoprocesado.
- Programación Open Sourse.
- Gestión de la información espacial.

Finalmente, es importante resaltar que la conformación de este conjunto de aplicaciones permitirá contar con información procesada de manera ágil y eficaz, disponible para visualización en cualquier visor geográfico como por ejemplo Google Earth cumpliendo uno de los principales objetivos perseguidos que es la facilidad de manejo para usuarios no tecnológicos.

ABSTRACT

With the realization of different field works that require speeding up and using tools that imply ease and efficiency in the execution as well as the reduction of costs and information processing times, it was proposed to implement a tool that would allow the capture of data for hydraulic infrastructure inventories in the field, for this purpose in the previous research a significant number of proprietary applications that provide similar services were identified, However, we also found the QField complement developed for QGIS that allows through the development of a form in QT or directly from the application to achieve this implementation in a "simple" way, so being this a GNU application that is constantly evolving and requires minimal programming for its implementation, it was

determined that this was the best suited to the initial approach to the problem posed. Additional difficulties will also be solved with the implementation of this project as the issue of synchronization of field data from different operators which will also be done with tools already available and known by most users of technology, seeking if with the development of this work simplify the implementation and make possible the use of the tool quickly, efficiently and economically for the realization of any work.

Additionally, given the proofs of concept of the problem, it was found that although it is true that QGIS allows exporting to a KML format, it is not modifiable in its presentation, and for the end user of the information may be irrelevant one part of it and necessary another, a fact that is not possible with the direct export from QGI, For this reason, we worked on an application that allows the export and conversion of the main file according to the structure defined in the GeoPackage, to a KML format of personalized visualization that will allow to conform a final file for the presentation to the final user by programming in C++.

All the above, is nothing more than the application of different concepts and tools provided with the execution of the Master in the subjects of:

- Computer tools for Geoprocessing.
- Open Source Programming.
- Spatial information management.

Finally, it is important to highlight that the conformation of this set of applications will allow to have information processed in an agile and efficient way, available for visualization in any geographic viewer such as Google Earth, fulfilling one of the main objectives pursued, which is the ease of use for non-technological users.

ESTADO DEL ARTE

Existe un oferta importante en el mercado de aplicaciones que se descargan al celular para la realización de algunas actividades de campo, entre ellas podemos descacar Locus (Assam Software, 2021), MDC (Gis Cloud, 2021), GISMapper (PoloSoftech, 2021), ArcGIS Survey123 (ESRI, 2021), todos ofrecen diferentes alternativas frente a la posible necesidad del cliente, no obstante todos son pagos y algunos son mas robustos que otros, sin embargo la personalización del formulario es inevitable en todos los casos por lo que QField (OpenGIS.ch, 2021) se convierte en una alternativa fácil que a muy bajo costo para el usuario o el cliente, se podrá personalizar el formulario según la conveniencia de los datos requeridos según la conceptualización y planeación del proyecto.

El caso de uso que motivó esta propuesta de TFM fue la realización de un trabajo de campo previo que por métodos convencionales implicó grandes inconvenientes y riesgos frente a la calidad de la información, los costos de procesamiento y el tiempo de entrega de los productos procesaros al cliente, ya que el trabajo se realizó por métodos convencionales o analógicos, es decir, llevando fichas de campo impresas y tomando un registro fotográfico con una cámara o un celular, y luego realizando el procesamiento de esta información en oficina. Como consecuencia de lo anterior, se evidenció que la digitación de la información y el procesamiento para integración de las imágenes tomadas en campo implicó un riesgo alto de alteración (involuntaria) de la información debido a las posibles confusiones frente a la asignación de las imágenes asociadas a cada punto, bien sea por la cantidad de imágenes o por la imposibilidad de determinar si la imagen corresponde al punto de estudio.

Es importante precisar que en el caso particular tomado como ejemplo solo se contó con 6 semanas para la ejecución total del trabajo y para ello se requirió de la conformación de un equipo de 15 comisiones de campo que generaban un volumen diario de imágenes de aproximadamente 1320, y el tiempo de procesamiento completo de esta información fue de 2 semanas, es decir que solo terminado este plazo podíamos identificar posibles inconsistencias o faltantes y para entonces el equipo ya se encontraba muy lejos del sitio de trabajo o realizando actividades adicionales en el mejor de los casos o incluso como normalmente ocurre ya se encuentran desarticulados, implicando reprocesos mayores para la validación y/o obtención de la nueva información de campo para la realización de los ajustes.

Cualquiera de las aplicaciones pagas existentes en el mercado, requiere de acciones complementarias para el procesamiento de la información como son la consolidación, depuración y preparación para la presentación final de la información, que a la postre se convierten en un problema para el usuario final ya que allí se requerirá de validaciones y de las acciones de consolidación necesarias para lograr ajustar, integrar y disponer toda la información recolectada de manera adecuada.

La solución de esta problemática esta orientada a automatizar el proceso de obtención de la información en campo (con herramientas de software libre) y a sistematizar el post procesamiento para la obtención de un producto “rápido” que le permita validar al cliente los avances y la calidad del producto obtenido.

OBJETIVOS DE TRABAJO PROPUESTO

Con la realización de este TFM se pretende lograr la implementación de una herramienta informática como el QGIS – QField que permita hacer lectura de datos visibles a múltiples usuarios para elementos de similares características elemento, el almacenamiento del registro fotográfico y de los datos no visibles de tipo geográfico, con dispositivos móviles en campo aplicando las herramientas de programación como C++ y Python.

Adicionalmente se deberá implementar un flujo de proceso para el procesamiento para la obtención adecuada del trabajo de campo que permita en un proceso de oficina o al cierre del día la extracción de los datos topados para cada dispositivo de manera sencilla y posteriormente la consolidación de la información en un archivo de tipo GeoPackage.

Elaborar una herramienta que permita la generación de un archivo georeferenciado en un archivo KML, que permita el despliegue de imágenes de cada sitio en un visor geográfico y algunos atributos básicos relevantes según la necesidad del cliente.

MEDIOS EMPLEADOS

Entorno de Trabajo

El entorno de trabajo esta fundamentado en el software QGIS – Madeira versión 3.4.14 en el que se desarrolla la parametrización y conceptualización de la estructura de datos para el formulario de captura. Adicionalmente emplearemos algunos complementos de QGIS siendo el principal el denominado (Opengisch, 2021) versión 1.9 lanzada en 2021 con el que se implementará el uso de la herramienta de captura para dispositivos móviles para Android.

Plataforma de Desarrollo

Algunas de las validaciones previstas fueron desarrolladas con las aplicaciones QT Designer y QT Creator 4.12.4 para el desarrollo del formulario y de la aplicación de conversión a KML que se hizo en C++. Se hicieron algunas pruebas de reglas de validación para el formulario implementado con Python empleando las librerías de PyQGis que funciona en el entorno del QGis y otras pruebas con la aplicación Anaconda como entorno de desarrollo para Python.

Otros recursos para implementación

El desarrollo completo del flujo de trabajo previsto, acude al uso de otras aplicaciones de uso libre como el Google Drive y el Drive Sync Pro versión 4.5.6, ambas abiertas para descarga en el Play Store de cualquier dispositivo móvil, sin embargo la última solo trae una limitación de uso para una carpeta cuando es en descarga gratuita y múltiples carpetas a sincronizar cuando se requieren sincronizar múltiples carpetas. Para el fin del trabajo presentado es mas que suficiente la versión no paga dado que solo se requiere la sincronización a una sola carpeta.

Para la conceptualización del modelo de datos, se empleó la aplicación Modelio Open Source 4.1 (Modelio, 2021), que permite la adecuada planificación de la estructura de datos que se implementará en el QGIS, igualmente su utilización ayuda a preveer la definición de tipos de datos y evitar retrocesos significativos en la planificación del trabajo.

FLUJO DEL PROCESO DE TRABAJO

Captura de información en campo

Es importante tener claro el proceso que tendrá que llevarse para lograr la implementación del proceso en campo, esto dará entendimiento pleno de las razones por las que se debe estructurar adecuadamente el trabajo en oficina y las condiciones que deberá tener el “archivo base QGIS”, que es con el que finalmente trabajará el equipo de campo y del que dependerá el éxito en la recolección del trabajo y la minimización de los reprocesos por deficiencias en la calidad de la información.

Para lo anterior, se plantea el siguiente esquema con el que se puede apreciar el flujo a seguir para el trabajo de campo:

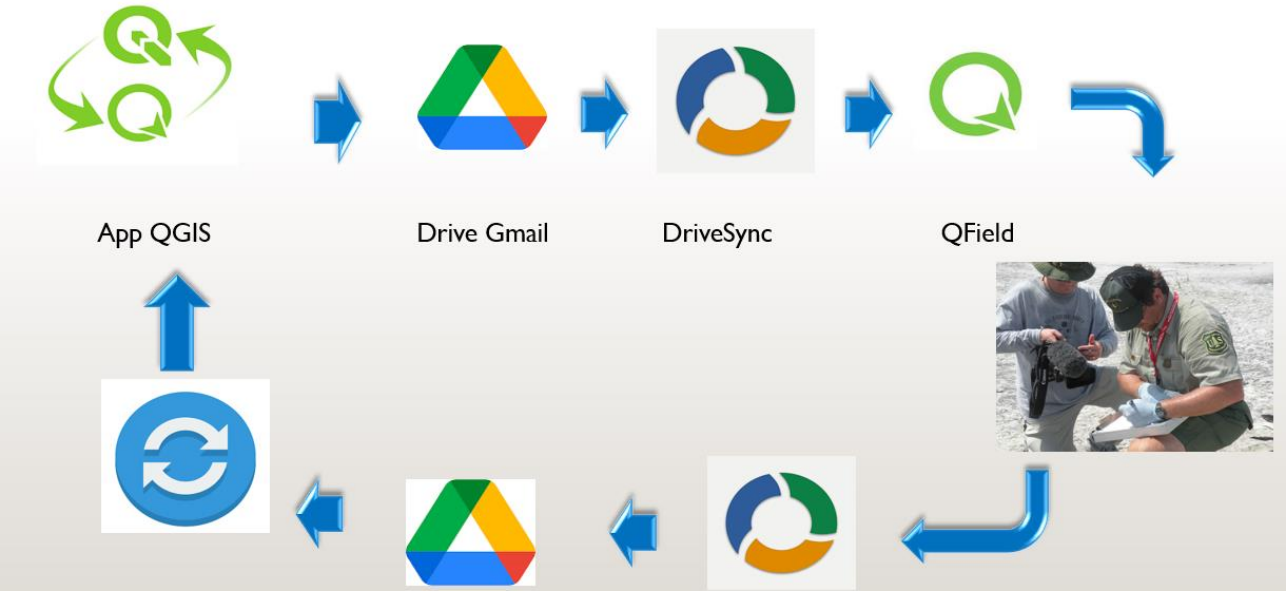


Ilustración 1 - Flujo de proceso para captura en campo de la información

Configuración inicial QGIS – QFIELD

El inicio de todo el proceso esta dado por el QGIS en el que se estructurará todo la configuración de lo documento de trabajo y su parametrización para luego realizar la exportación del archivo base para QFIELD, el intercambio de información entre el las dos herramientas mencionadas es director y transparente por lo que en este primer paso esencial para todo el trabajo a realizar se deberá invertir una buen parte del tiempo para poder realizar adecuadamente el trabajo en campo.

Creación de una carpeta de trabajo

La carpeta de trabajo contendrá toda la documentación base que será exportada a cada operario en campo por lo que se recomienda crear tantas carpetas inicialmente en Google Drive que será el medio para compartir remotamente a los diferentes usuarios y para ello tantas carpetas compartidas como operarios halla esto facilitará la identificación y actualización de la información de ida y de vuelta.

Herramienta de Sincronización en el móvil

Una vez puesta la información en una carpeta compartida por Drive, es necesario ubicar estos archivos en una carpeta de trabajo o en una SD Card en el equipo móvil, para ello existen múltiples aplicaciones tanto para IOs como para Android en sus respectivos centros de recursos, en nuestro caso se escogió Drive Sync V4.6.5, es libre y sin costo para una carpeta lo cual es suficiente porque trabajaremos directamente sobre solo una carpeta.

Complemento de sincronización en QGis

Este complemento permitirá sincronizar los archivos de trabajo tanto para la descarga y distribución en nuestra carpeta compartida desde la base como para la carga una vez se quiera recolectar la información procesada por los operarios de campo, se puede obtener en la carpeta de “Complementos o Plugin” con el nombre de QField Sync desde el QGIS.

Trabajo de campo en QField
El operario recolecta la información deseada empleando la aplicación QField y el proyecto definido para el efecto.

Descarga del trabajo de campo
Esta será realizada del mismo modo que se obtuvo, es decir, se sincroniza la carpeta de trabajo empleando Drive Sync que permite la actualización de la carpeta remota compartida vía Google Drive y la carpeta local del móvil.

Sincronización de carpetas de trabajo con el proyecto en centro de operaciones
El proceso final para el trabajo de campo es la sincronización de las diferentes carpetas por operario y la consolidación de los archivos trabajados en el día, esto se hace con la herramienta de sincronización de QField desde el QGis pero esta vez con la subida de información empleando el botón  .

Procesamiento y consolidación en Oficina

Ya en oficina el procesamiento de la información obtenida en campo que para el efecto obtendremos casi en línea si no hay conexión a internet y podremos allí implementar algunos procesos que no refieren a la práctica de este proyecto pero que refieren a los siguiente tópicos:

- Controles referidos a la calidad de los datos
- Controles asociados a la calidad de las imágenes.

Como estos criterios dependen de las condiciones con las que se defina el trabajo no ahondaremos en ellos pero es importante que se definan algunos lineamientos al respecto.

CONFORMACIÓN DEL ARCHIVO BASE DE TRABAJO

Como se ha mencionado anteriormente todo el desarrollo “base” es realizado en el entorno QGIS, para ello iniciamos el proceso creando un archivo QGZ dentro del área de interés del proyecto, sobre este archivo se realizarán todas las definiciones que serán explicadas en títulos posteriores e igualmente el que será objeto de sincronización con el dispositivo móvil que empleará cada operario de campo.

Desarrollo Del Formulario De Captura De Datos

El formato de inventario
El proceso arranca con la definición de la estructura del formulario de captura de la información que se origina para el caso nuestro en la definición de la estructura los atributos que se desean obtener del elemento a inventariar, en este caso son las estructuras de drenaje transversal sobre vía, la identificación de los elementos principales y la toma de las medidas de estos así como su estado

es lo requerido para el trabajo. Un gráfico de cómo está definido el elemento lo podemos ver a continuación

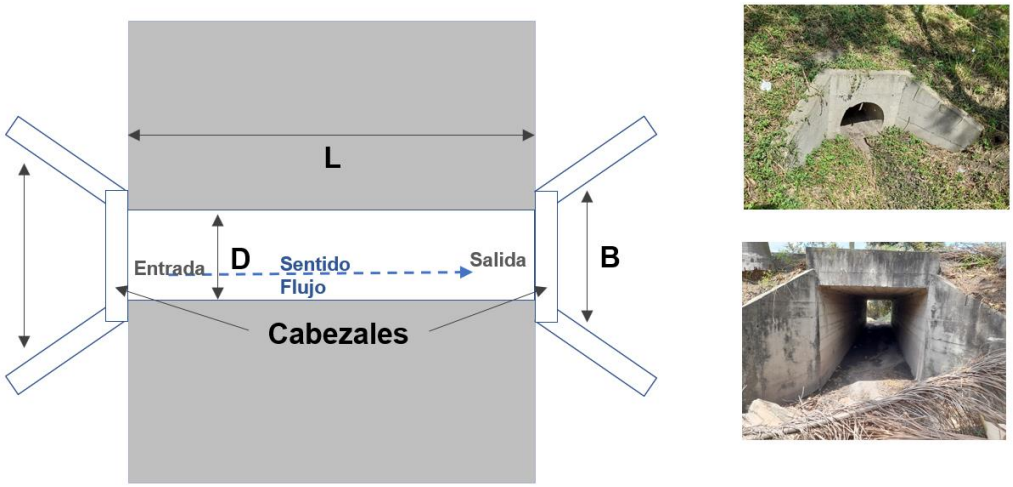


Ilustración 2 - Esquema de identificación de elementos y atributos

Con base en el esquema anterior y en los requerimientos técnicos se puede establecer un formato base con el que se puedan establecer las condiciones técnicas de alcance para la realización del trabajo.

Ilustración 3 - Formulario con parametrización de alcance

Con las condiciones anteriores establecidas y con algún conocimiento de las condiciones para la realización del trabajo tanto en campo como en oficina, podremos ahora estructurar un modelo de datos con la estructura base para la captura de la información.

Modelo de datos

Con el empleo de la herramienta Modelio se determinó un modelo simplificado que permitirá la construcción posterior de la estructura de datos del Geopackage, como se verá a continuación tendremos solo un componente principal asociado a algunas definiciones básicas de dominios por tipo de dato.

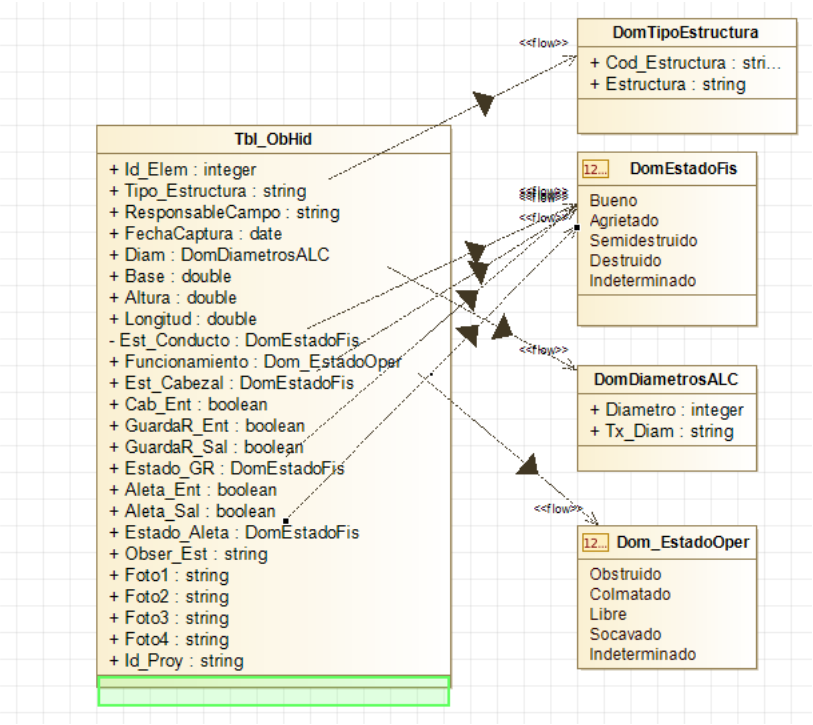


Ilustración 4 - Estructura del modelo de datos nivel principal geográfico

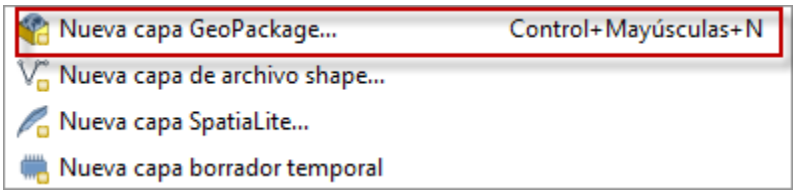
La definición de esta estructura es la que llevamos posteriormente al QGIS para la creación de la estructura de datos, en nuestro caso empleamos la estructura de la capa principal a la que denominamos “InvEstructuras4” con una definición tipo punto, el resto de elementos (Los dominios) serán definidos en el GeoPackage como tablas con atributos mínimos, sin perder de vista que uno de los objetivos principales del trabajo es la simplificación de cada actividad al máximo para que sea asequible a cualquier usuario.

Conformación del Geopackage y niveles de información

Esta configuración la podemos realizar directamente desde el QGIS siguiendo los siguientes pasos.

Creación del Geopackage:

Desde la opción “capa” del menú principal adicionar nuevo geopackage.



Creación de la capa principal de edición:

Se conforma la estructura de la capa principal "Tipo Punto" con la definición de cada uno de los atributos según lo establecido en el modelo de datos, esto es, definiendo la campa con cada uno de los campos establecidos como nombre, tipo de datos y condiciones de tamaño mínimas para mantener reservado el mínimo de espacio, esto garantizará una adecuada velocidad del equipo al momento de utilizarlo.

Tener en cuenta que inicialmente el orden con el que se creen los campos determinará la secuencia de aparición en el formulario automático de QGIS, por lo que es importante tener bien definido el modelo de datos para la capa e irlo organizando secuencialmente.

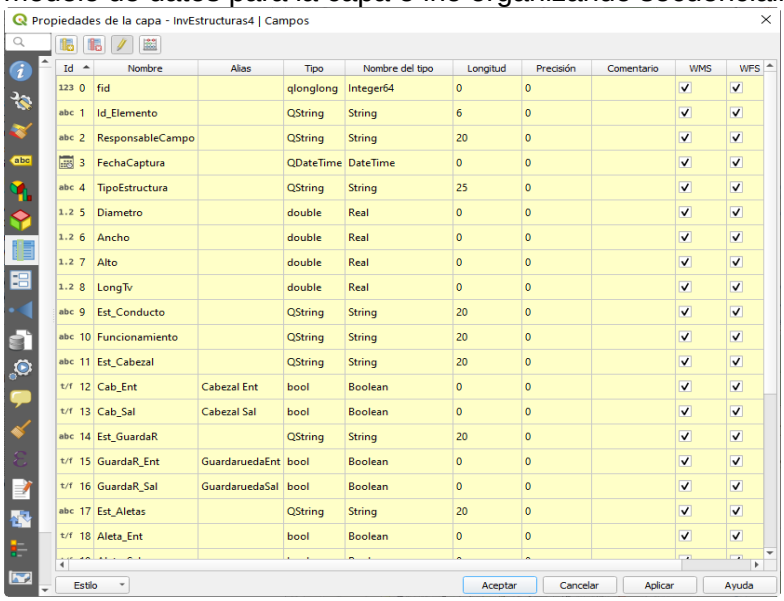


Ilustración 5 - Estructura de datos capa principal

Si se desea se pueden agregar nuevas capas editables siguiendo el mismo procedimiento, en el caso de uso del trabajo realizado solo se requiere una capa pero en otro tipo de trabajo podría ser que se requiera alguna adicional.

Creación y Asociación de las tablas de dominios:

Las tablas de dominios son objetos no espaciales en el QGIS, y se pueden definir con la creación de tablas de datos dentro del QGIS. Esto se realiza mediante la creación de capas no espaciales con los posibles valores que adoptarán según el atributo al que se prevean aplicar, y solo debemos indicar que son objetos sin geometría:

Un ejemplo de tabla para dominio de datos lo podemos observar a continuación:

ido	EstadoFisico
1	Bueno
2	Agrietado
3	Semidestruido
4	Destruido
4	Indeterminado

Ilustración 6 - Ejemplo de tabla para dominio de datos

Estas tablas se crean igualmente sobre el Geopackage y se puede definir una agrupación de capas en el QGIS, esto ayudará posteriormente si se quiere impedir la visualización de estas capas por el usuario.

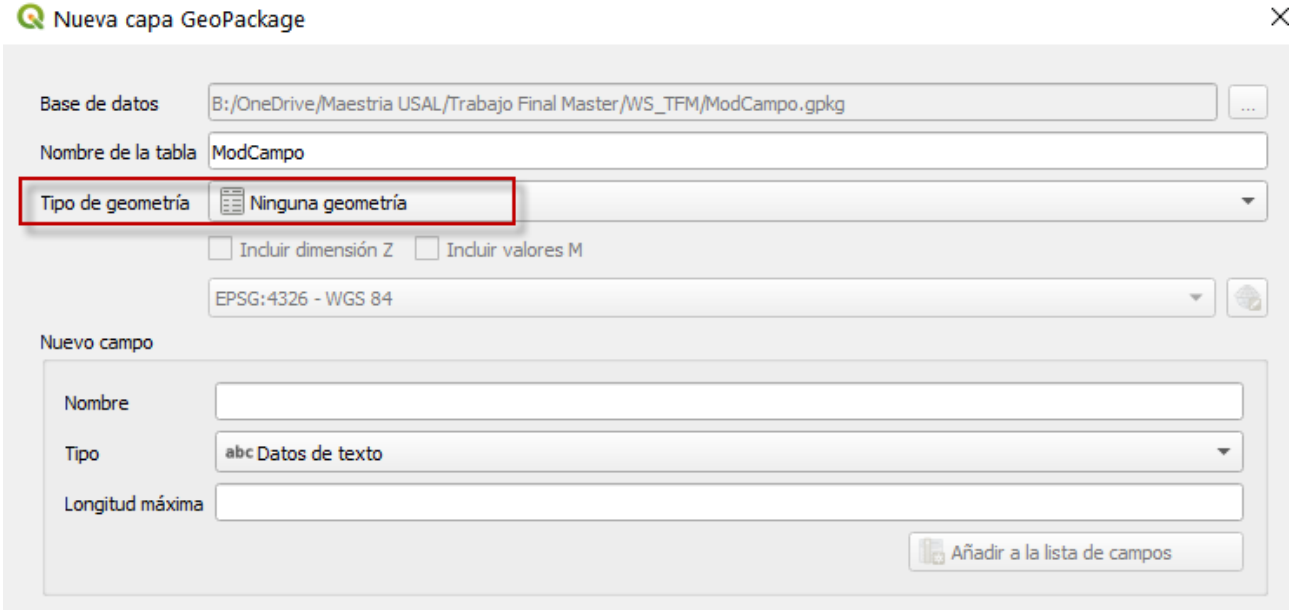


Ilustración 7 - Ventana de creación de capa no espacial en QGIS

Lo que podremos ver hasta este punto es la siguiente estructura en la ventana de capas del QGIS e igualmente en el GeoPackage:

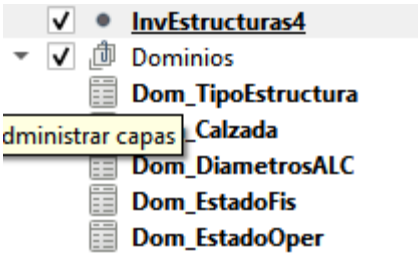


Ilustración 8 - Visualización de la estructura de datos completa en Geopackage

Parametrización de dominios para cada atributo de la tabla:

La asignación de dominios se puede realizar de diferentes maneras (QGIS, 2021), en el caso de estudio se adoptó la relación de valores, que consiste básicamente en una asociación de la tabla dominio a el atributo en referencia, esto se puede lograr entrando por las "propiedades" del elemento geográfico en la opción "Tipo de control", allí se establecerá por "Capa" la tabla de dominios asociada y las columnas clave y de valores que asignarán el dato de visualización y el de entrada en la tabla respectivamente.

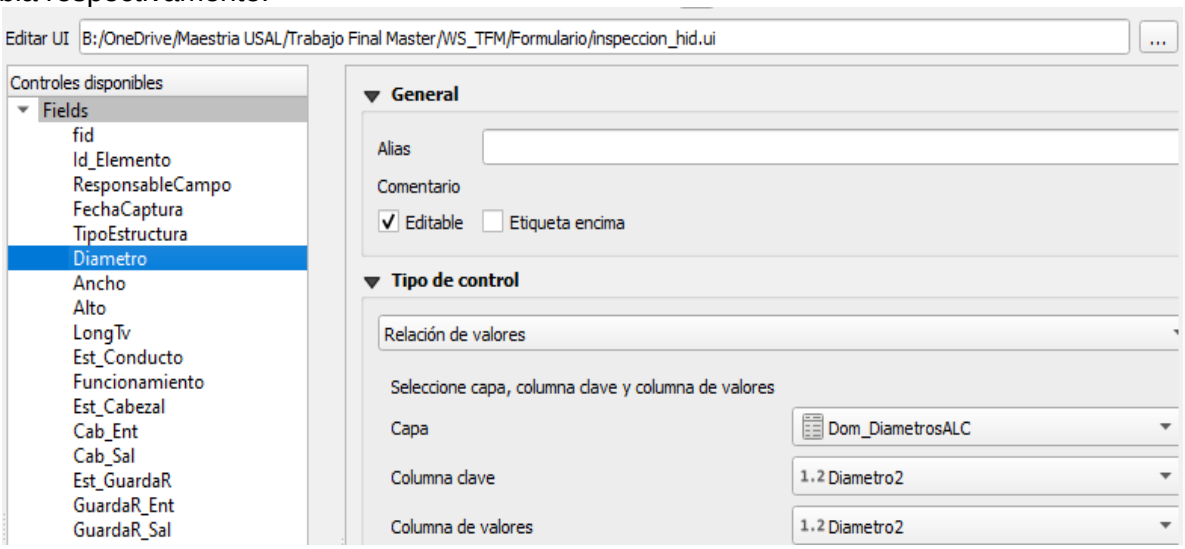


Ilustración 9 - Asignación de dominios por relación de valores.

La configuración de parámetros básicos para cada atributo es fundamental y con el QGIS se realiza el proceso como con cualquier otra herramienta SIG, indicando valores por defecto y condiciones propias para la utilización de cada uno de los atributos. Sin embargo, lo más relevante de esta actividad es establecer valores por defecto que permitan establecer si el operario de campo realizó o no el diligenciamiento así como la validación de algunos valores.

Parametrización de Valores por defecto:

Definir valores por defecto será importante para los controles de calidad que se quieran hacer posteriormente, esto ayudará a evitar problemas en el procesamiento de los datos y adicionalmente medir la calidad y cantidad de datos diligenciados adecuadamente por cada operario de campo, por esta razón existen atributos que indican un valor y también que no han sido diligenciados, por ejemplo "Indeterminado" en los estados o "0" en los diámetros" podrá ser un indicador de esta situación aunque primará el resultado del análisis que realice la persona que realice el procesamiento de la información.

Formulario para captura de datos

Todo el proceso anterior define de fondo el formulario que será empleado en campo para la captura de los datos en campo, QGIS por defecto define un formulario para la captura de los datos (QGIS, 2021) que consolida todo lo que hemos realizado anteriormente, pero recordemos que todo el ejercicio lo estamos realizando para hacerlo en un terminal portátil o en un teléfono móvil y es allí donde debemos centrar nuestro esfuerzo, de validar y realizar las comprobaciones necesarias, esta aclaración se hace porque durante la realización de este trabajo se invirtió bastante tiempo en la elaboración de un formulario de captura que mas adelante tuvo que ser desechado ya que no si bien en el QGIS si se logra la utilización del formulario, en el QFIELD se trabajará con el formato nativo del sistema.

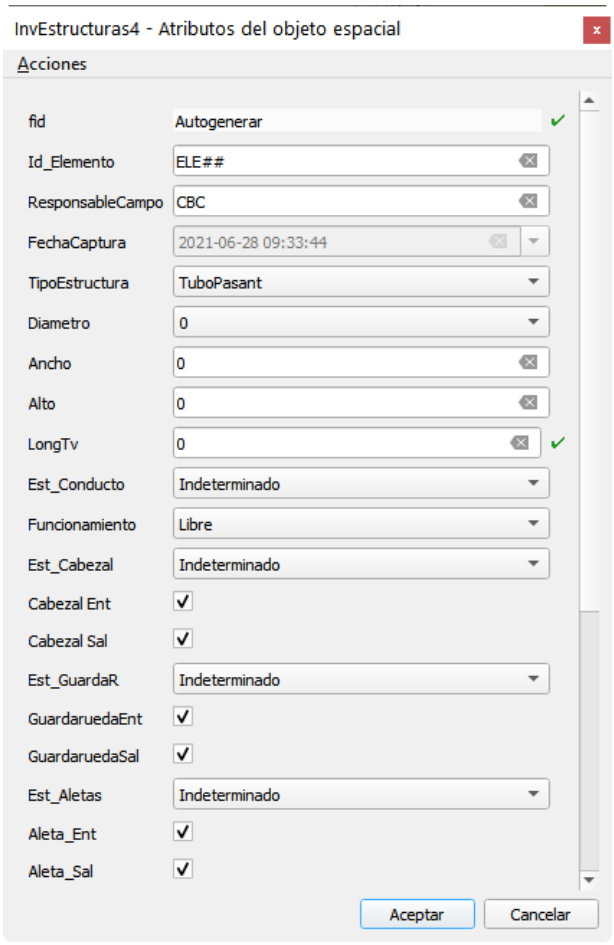


Ilustración 10 - Formulario nativo QGIS

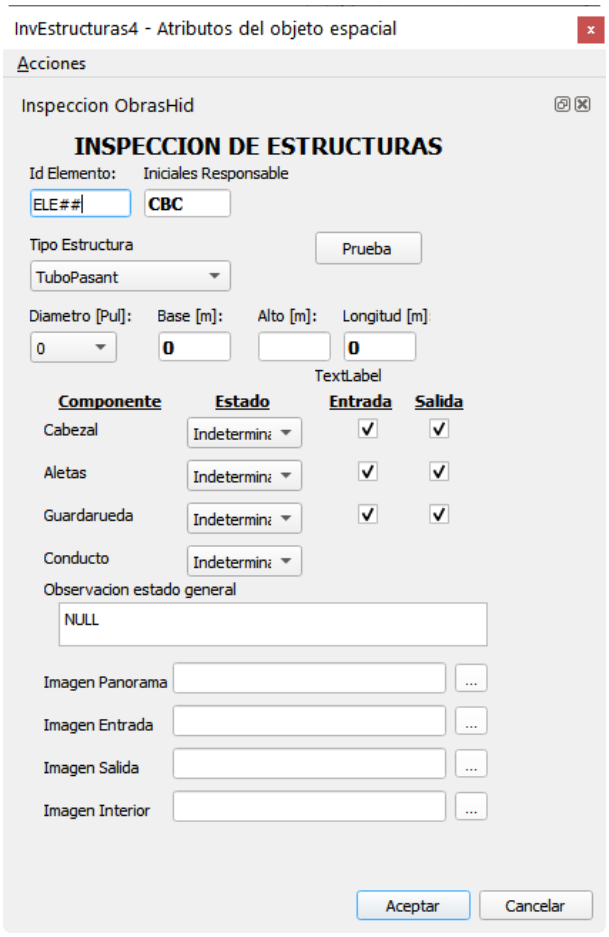


Ilustración 11 - Formulario generado con QT Designer

Como se observa con la comparación de las dos ilustraciones, la ventaja de hacer el formulario personalizado es que en principio podemos dar la visualización que se quiera para el QGIS, sumado al hecho que podremos definir algunas reglas de validación, sin embargo para el trabajo sobre QFIELD esteremos limitados a las limitaciones que por ahora tiene la versión de la herramienta, la cual no permite incluir la configuración del código con reglas de validación y tampoco el formulario.

Aspectos relevantes en la definición del formulario:

- La definición de los alias permitirá la visualización personalizada de los nombres de atributos

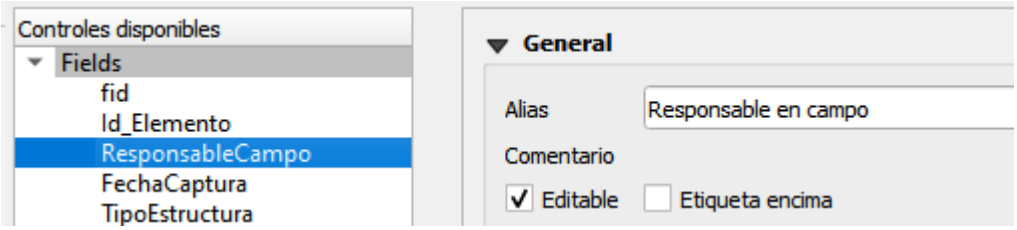


Ilustración 12 - Definición de "alias" para atributos

- Definición de valores por defecto, tener en cuenta también su conceptualización en el modelo de datos para la creación de los dominios.

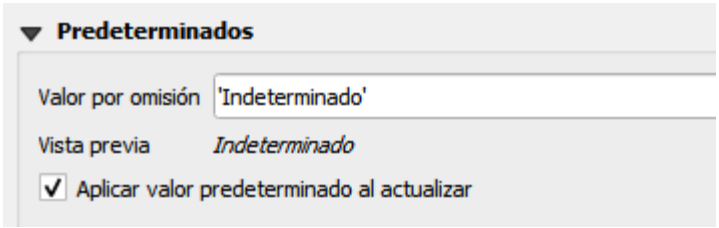


Ilustración 13 - Definición de valores por defecto

- Definición de rutas relativas para imágenes será necesario para que se almacenen adecuadamente las rutas de las imágenes dentro de la cama, pero posteriormente debemos tener en cuenta que con la importación debemos reemplazar la ruta a una relativa para el proyecto.

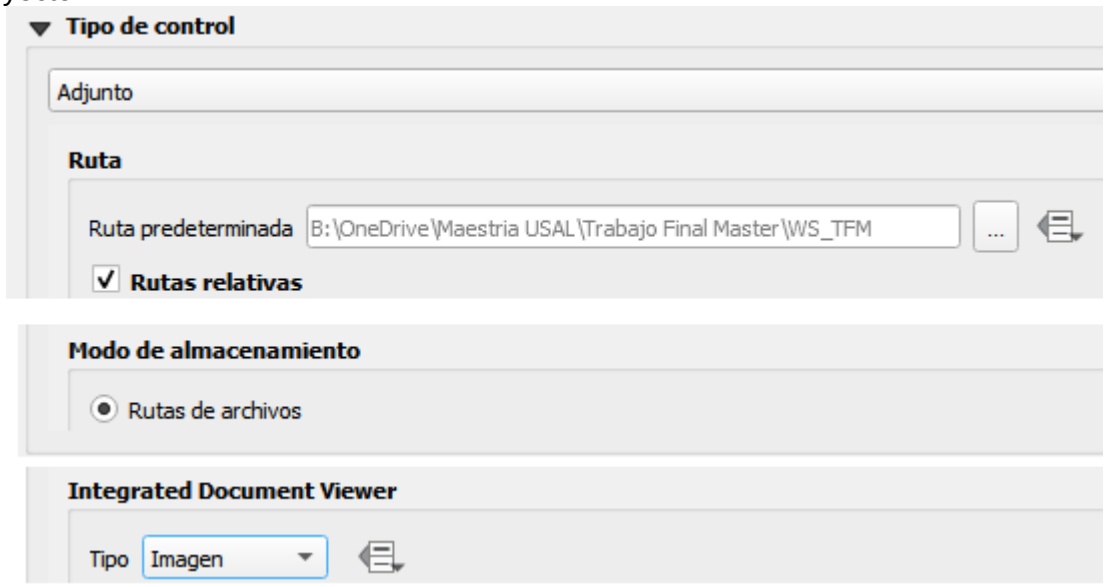


Ilustración 14 - Definición de parámetro para toma de registro fotográfico con el móvil

Conformación del Mapa Base de trabajo

En muchos casos los trabajos de campo se realizan por ser sitios alejados de las ciudades y en general del desarrollo, no contamos con red de internet por esta razón es conveniente establecer una capa base que le permita al operario de campo tener referencias relativas del sitio donde se encuentra, para ello emplearemos la función “Generate Tiles XYZ (MBTiles)” (QGIS, 2021) que se

encuentra en la caja de herramientas del QGIS, esta permite extraer una ventana raster con las referencias relativas del proyecto.

Aspectos relevantes a tener en cuenta con la generación del MBTiles

- Para la generación de esta ventana que denominaremos mapa base, es necesario generar una capa con el polígono del área de interés, mismo que definirá los límites sobre los que se establecerá el raster a extraer que tendrá un formato JPG o MBTiles.
- Considerar que aunque es posible extraer imágenes de cualquiera de los operadores libre que existen (Ej. Bing, Google Maps, OSM), igualmente éstas tienen un tamaño de archivo bastante grande por lo que es necesario considerar la capacidad operativa de los equipos móviles a emplear, la recomendación es la de emplear la capa de referencia predeterminada (eventualmente la de relieve) y no la imagen satelital, esta dará la información necesaria sin complicar su utilización por capacidad de los equipos móviles a emplear en campo.
- Adicionalmente se debe considerar que el raster (Creado con el Tiles XYZ) por efectos de referenciación en el móvil y de la configuración del "setup" de inicio, no es conveniente almacenarlo directamente sobre el GeoPackage sino que se debe manejar aparte como un archivo anexo de la misma carpeta a sincronizar.

Un vez generado el raster podremos tener una nueva capa "raster" en el proyecto QGIS la cual deberá estar guardada en la misma carpeta del GeoPackage y del proyecto QGIS. Una muestra de lo que se debe ver es lo siguiente:

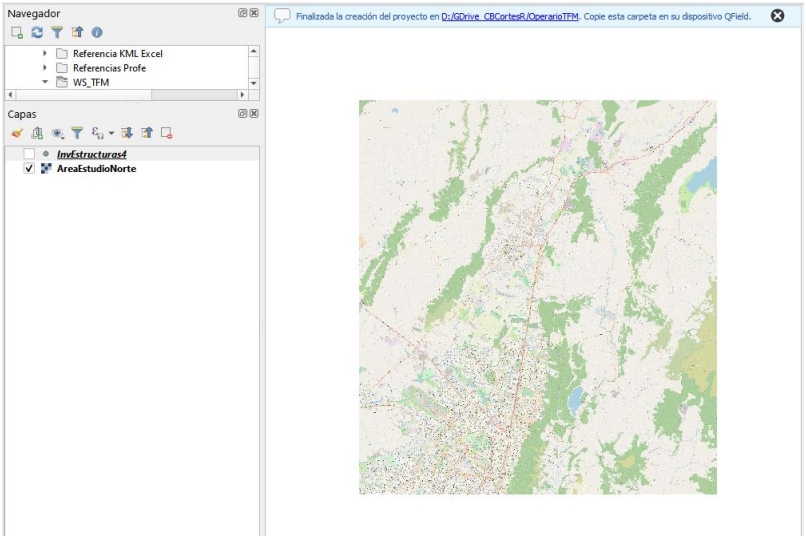


Ilustración 15 - Capa base raster, para referenciación offline

CONFIGURACIÓN DE ARCHIVOS DE TRABAJO EN EL MÓVIL

Instalación de aplicaciones en el dispositivo móvil

La integración entre diferentes aplicaciones existentes en el mercado es la base para poder lograr el éxito en la operación de campo, y para ello se deberán realizar algunas instalaciones de herramientas previas a la salida a campo del personal.

1. Instalar **Google Drive** en los dispositivos móviles, adicionalmente cada operario debe contar con una cuenta de Gmail con la que configurará la cuenta localmente a su celular.
2. Instalación de **Drive Sysc**, esta aplicación permitirá la sincronización de archivos de ingreso para la realización del trabajo y de salida para enviar los resultados diarios del trabajo realizado.
3. Instalación de **Qfied**, que es la herramienta principal para la captura de datos y a su ves la interfase de QGis para la realización del trabajo de campo.

Todas las aplicaciones anteriores están dispuestas en el Play Store para dispositivos móviles Android y en tiene sus equivalentes en el App Store en sistemas IOs.

Configuración de Drive Sync

Como se ha indicado anteriormente el operario debe conectar en su móvil una carpeta compartida a su cuenta de Google desde la central de distribución de trabajo y a su vez esta carpeta se configurará como la carpeta de entrada para que a través de la configuración Drive Sync se indique en qué lugar dentro del móvil alojaremos dicha información.

Esta configuración de los archivos compartidos para sincronización será la que permitirá cargar la estructura básica del archivo de trabajo el en QFIELD, así las cosas de manera simple como se puede observar esta configuración en el siguiente gráfico:

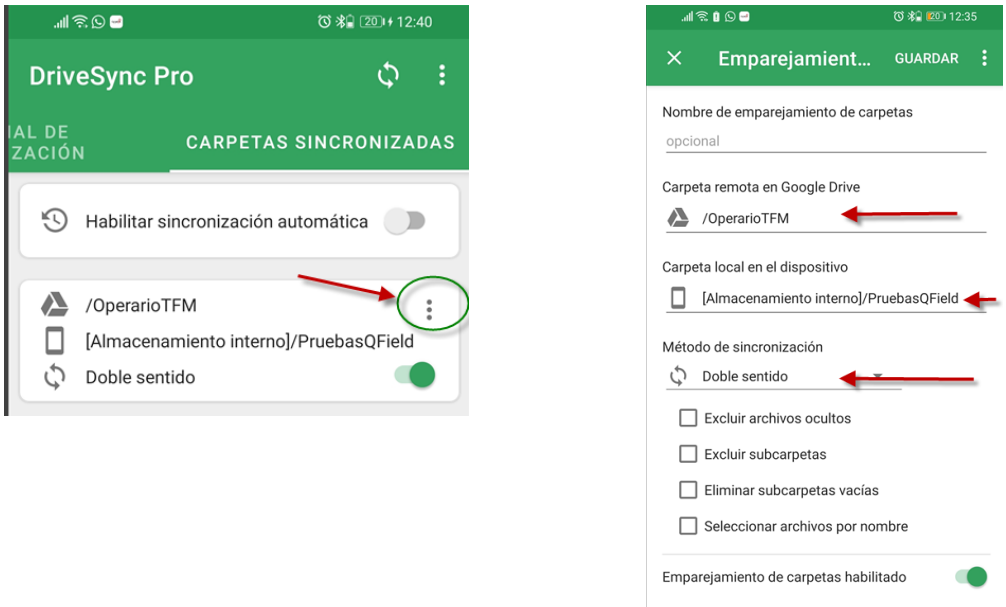


Ilustración 16 - Configuración alojamiento local archivos de trabajo con Drive Sync

Configuración QField

Toda la configuración para poder realizar el trabajo de campo se hace desde la “Central de Operaciones” con el trabajo previo que se ha explicado en los procesos anteriores, ya instalado el QField en el equipo móvil lo único que tendremos es que abrir el archivo de extensión QGS, que se halla definido desde el QGIS, ésta ya carga con toda la información base para la captura de datos en campo.

COMPLEMENTO DE SINCRONIZACIÓN DE QGIS

Es necesario instalar el complemento de QGIS denominado QField Sync, este complemento activará en el menú principal de QGIS dos botones uno para descargar la información base del proyecto de trabajo y el otro para subir nuevamente a nuestro ordenador la información recolectada por el operario

Adicionalmente debemos especificar las carpetas en las que queremos ubicar los archivos con la ejecución de las funciones de carga y descarga, para esto en el menú de complementos de QGIS buscaremos “QFieldSync” y desplegaremos el menú de cascada para seleccionar “Configuración del Proyecto”

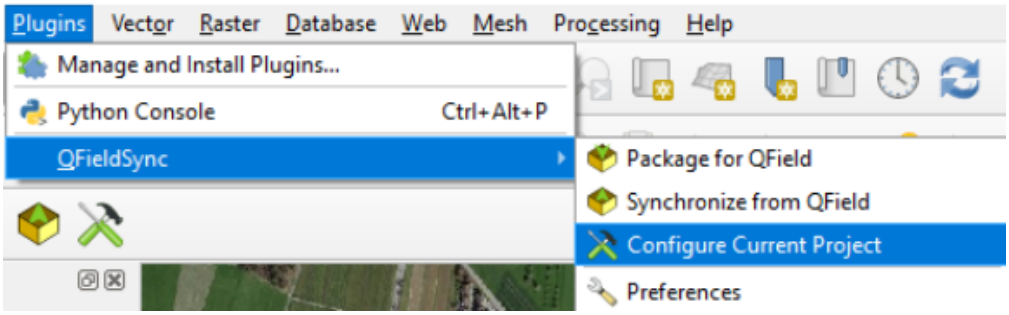


Ilustración 17 - Configuración de proyecto para sincronización en QGIS-QField

La configuración de proyecto nos permitirá definir algunas condiciones especiales con relación a la parametrización del sistema en lo que refiere a que capas se copiarán, se sobreescribirán y sobre todo cuales de ellas podrán ser trabajadas fuera de línea, esto dependerá de las condiciones propias de cada proyecto y es explicado en detalle en el manual de operación de QField (QField -Sync, 2021)

En el caso particular de nuestra plantilla de documento solo tenemos dos niveles de información de definidos, la capa base que hemos denominado “AreaEstudioNorte” y que por supuesto no es sujeto de edición, la otra que es la capa editable es “InvEstructuras4”, que por definición es donde alojaremos los diferente puntos levantados en campo y que por ende es la que deberá contar con la posibilidad de “edición fuera de línea”.

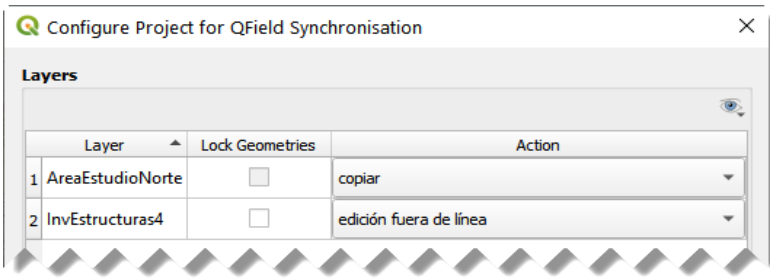


Ilustración 18 - Definición de acciones para capas

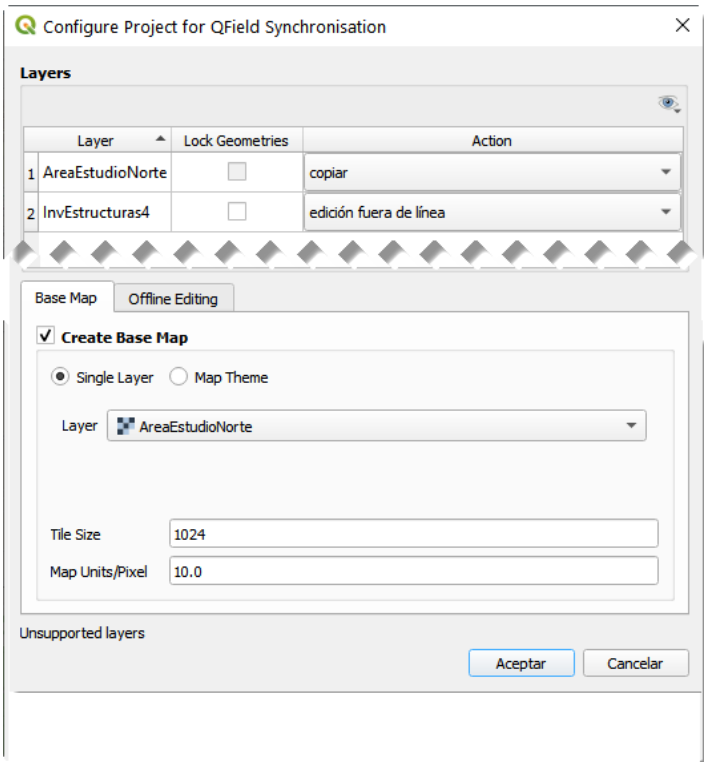


Ilustración 19 - Configuración de proyecto para sincronización QField

Por otra parte, en el mismo menú están las preferencias donde podremos configurar las rutas de entrada y salida para los archivos de trabajo, podremos definir dos carpetas diferentes por orden y procedimiento crear una subcarpeta dentro de la carpeta de sincronización en la que se almacenen los datos que serán objeto de importación una vez re realice el trabajo.

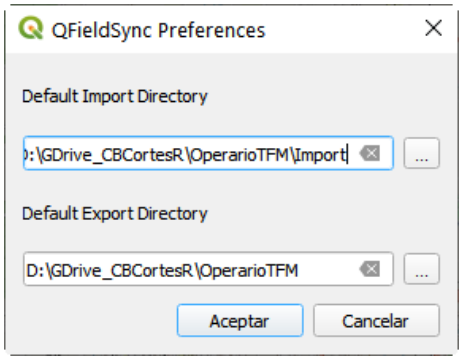


Ilustración 20 - Configuración de directorios de entrada y salida de archivos

EJECUCIÓN DEL PROCESO DE CAPTURA Y SINCRONIZACION

El proceso integral debe ser articulado cuidadosamente cuando se trata de varios operarios, y para ello debemos tener claros todos los pasos a seguir:

partiremos del procesamiento en el QGIS del complemento de sincronización para descarga de los archivos base en la carpeta compartida:

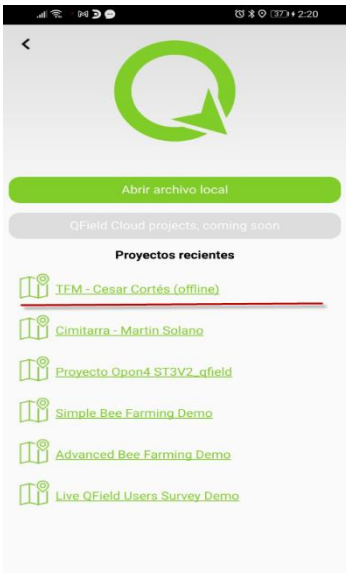
- Sincronización de descarga el proyecto actual desde QGIS a la carpeta compartida.
- Sincronización de descarga los archivos de trabajo de la carpeta compartida al celular a través del DriveSync.
- Ejecución del proyecto de extensión " *.qgs " sobre el QField desde el celular.
- Captura de los datos en campo.
- Sincronización para carga los archivos de trabajo (ejecutado) de la carpeta compartida al celular a través del DriveSync.
- Sincronización de carga de los archivos de trabajo ejecutados hacia la carpeta compartida.
- Consolidación de archivos procesados en oficina

En los títulos anteriores se han detallado todos los pasos de configuración de cada una de las herramientas por lo que la ejecución se limitará a emplear cada una de las herramientas en el orden señalado anteriormente, esto nos permitirá conformar un documento consolidando el trabajo de cada uno de los operarios de campo en un solo archivo dentro del QGIS.

Captura en campo

Para la realización de la captura en campo tendremos que verificar la carga del proyecto en formato QGS en el dispositivo móvil, como se ha realizado la parametrización con un mapa base

Apertura del documento de proyecto



Una vez se activa el programa QFIELD (QField - User documentation, 2021) en el equipo móvil se seleccionará el proyecto que hemos copiado en la carpeta definida del móvil, en el caso de la imagen a la izquierda, se observa que para el caso de nuestro ejemplo el programa guarda en memoria todos los proyectos abiertos, pero si no es así, se podrá realizar por una ventana de búsqueda de directorios convencional según la referencia del sistema operativo con la que cuente el equipo móvil.

Es importante recalcar que por ahora la aplicación no permite alojar los archivos de trabajo en una memoria externa, por lo que será necesario tener cuidado al definir la carpeta de almacenamiento y sincronización para no perder información.

Ilustración 21 - Apertura de proyecto en QField

En los casos que el equipo cuente con poca memoria interna lo que se recomienda es que se integre la memoria externa como un archivo de almacenamiento paralelo del registro fotográfico, es decir que una vez se acabe la jornada o en el momento que se requiera, la carpeta de imágenes se mueva a la memoria externa y se realice la transferencia manualmente, lo importante es que en el archivo geográfico quede el registro de la "ruta" con la identificación del nombre asignado a la imagen, ya que con dicha información se podrá realizar el emparejamiento mediante una proceso de oficina.

Edición y creación de elementos QFIELD

La captura de registros y la edición de sus atributos será muy similar a como se realizan en QGIS, la diferencia aquí es que definitivamente la aplicación esta orientada a ser fácil de emplear por cualquier usuario, por lo que solo tendrá que seleccionar la capa de edición según la condición que encuentre en campo e inicia el diligenciamiento de los atributos. Para efectos de evitar confusiones sobre todo porque en muchas ocasiones el personal de campo no es altamente calificado, se recomienda que los atributos espaciales se procure no sean sujeto de manipulación por el usuario.

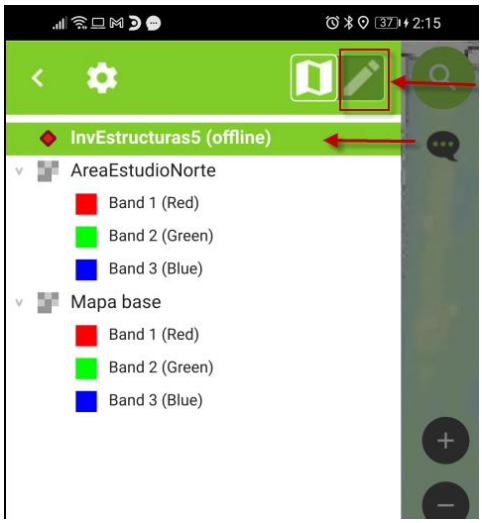
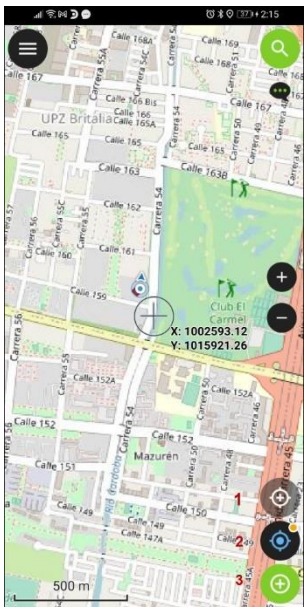


Ilustración 22 - Definición del modo de edición para inicio de captura



En la ilustración anterior, se observan las condiciones para inicio de la edición y captura de información a las que adicionalmente habrá que recalcar que solo serán editables las capas que se configuren como tales desde el QGIS, en el caso del mapa de fondo (Creado con el MBTiles), el sistema directamente creará la capa base y un geopackage adicional que lo separa del que almacena las capas de datos, en el proyecto del ejemplo se dispuso una capa adicional replicada porque esta funcionalidad presentó inconvenientes en algunas ocasiones dependiendo del equipo móvil empleado.

Con lo anterior dispuesto lo que resta es realizar la toma de atributos y para ello el sistema despliega el proyecto con la información cargada en el raster de mapa base, y se activan tres mirillas en la esquina inferior derecha, estas son las pocas herramientas que deberá deberá manipular el operario, y es precisamente lo que hace que esta aplicación sea tan útil, de arriba abajo la mirilla (1) permite la localización del nuevo elemento de acuerdo a la localización del móvil, será directa y el usuario no interviene más que para

Ilustración 23 - Modo de edición QField

confirmar el punto creado, la mirilla (2) permite la localización del nuevo elemento manualmente mediante localización de la mira que muestra el centro de la pantalla arrastrando el dedo sobre ésta y la mirilla (3), agrega el nuevo punto y despliega el menú de entrada de atributos según el formulario definido y será en esta pantalla en la que se realice todo el ingreso de datos visualizados, identificados o medidos en campo por el operario., esto incluye para el caso del formulario planteado el ingreso de un registro fotográfico asociado al punto.

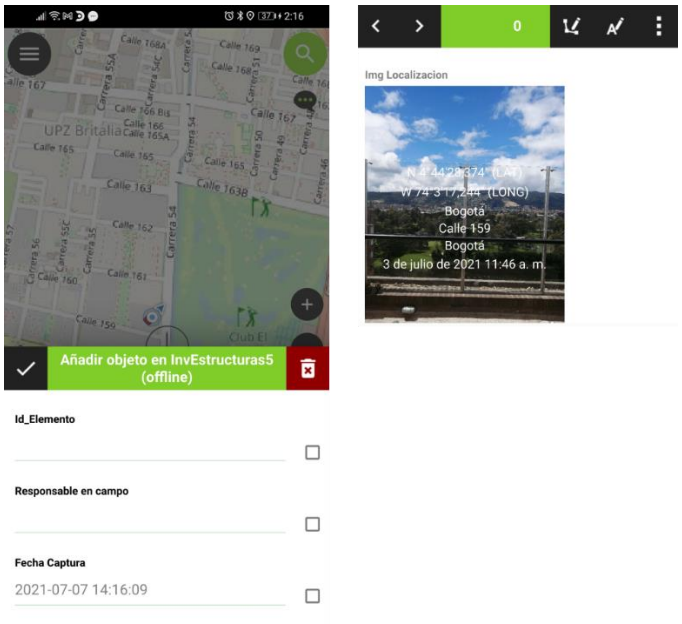


Ilustración 24 - Ventana para ingreso de atributos

Una vez se terminen de agregar los puntos en campo el operario deberá realizar una sincronización de descarga de información con la carpeta compartida de Google Drive de cuerdo al procedimiento que se indica en el siguiente acápite.

Sincronización de información de campo

Sincronización móvil – Google Drive

Una vez se ha concluido con el trabajo de campo, se sugiere la realización de la descarga y sincronización con la carpeta compartida de Google Drive, esto ayudara a liberar espacio de los equipos móviles si se hace diariamente e igualmente permitirá realizar los controles de calidad correspondientes para el trabajo realizado.

Para realizar la sincronización de descarga emplearemos la aplicación Drive Sync que se instaló en el proceso de CONFIGURACIÓN DE ARCHIVOS DE TRABAJO EN EL MÓVIL, y con el ícono de sincronización esperando a que el proceso de sincronización sea concluido satisfactoriamente, el sistema en el celular así lo indicará con un mensaje de estado de sincronización “Listo” como se muestra a continuación:

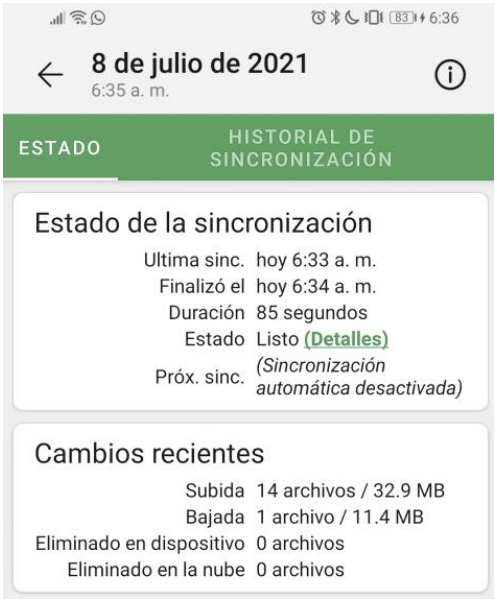


Ilustración 25 - Proceso de sincronización desde móvil

Sincronización del Celular a la carpeta compartida Google Drive

La sincronización desde el celular a la carpeta en solo paso ya que solo tendremos que sincronizar desde DriveSync nuevamente, si se configura la sincronización de doble vía es directa la actualización de esta información a la carpeta del GoogleDrive que se disponga el proceso a seguir es la consolidación de la información en un solo archivo geográfico.

Consolidación y información de archivo geográfico

El proceso anterior, arrojará fragmentos del trabajo de cada equipo de campo, estos fragmentos son agregados a un GeoPackage, por lo que el proceso de oficina consistirá en validar y consolidar los productos obtenidos hasta contar con una capa geográfica con los atributos recolectados para cada elemento y la referenciación relativa de las imágenes, ya con la información consolidada se podrá generar la exportación primero al archivo “ csv “ y luego la generación del archivo “KML” que veremos en los siguientes títulos.

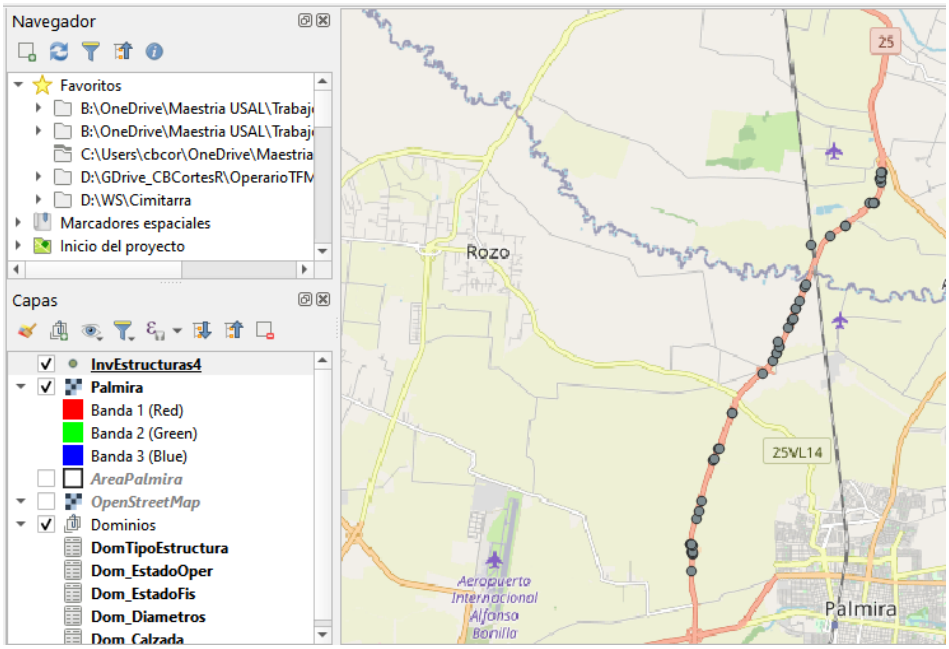


Ilustración 26 - Generación de documento de trabajo QGIS

Extracción de archivo “csv” para generación de KML

Una vez consolidado el conjunto de datos de todos los operarios de procederá a exportar el archivo con todos los atributos en formato de texto con extensión “CSV” éste archivo será la base para la generación del archivo KML con el complemento creado para tal efecto.

COMPLEMENTO PARA CONVERSIÓN A KML

Dado el archivo de trabajo es particular para un tipo de ejecuciones particulares, se pensó en la estructuración de un complemento que permitiera ejecutar un proceso de conversión del archivo “csv” a uno de extensión “kml” que se ceñirá a condiciones particulares de diseño, por lo que ha preferido que éste sea desarrollado de manera alternativa.

Una vez generado el “ csv “, se deberá abrir el archivo y reemplazar las rutas relativas de las imágenes por el texto “RegFotografico\”, que será la misma carpeta donde se alojara el consolidado de imágenes generadas que ha sido importada desde los diferentes equipos móviles. Adicionalmente de deberá suprimir la primera fila con los nombres de la columnas y guardar el archivo nuevamente con los ajustes realizados.

Este complemento ha sido desarrollado en C++ y permite leer el archivo “ csv “, recién generado para convertirlo a un KML “Personalizado”, el complemento permite la lectura del archivo en cuestión e inmediatamente realiza una validación inicial que no se halla alterado el numero columnas del archivo original.

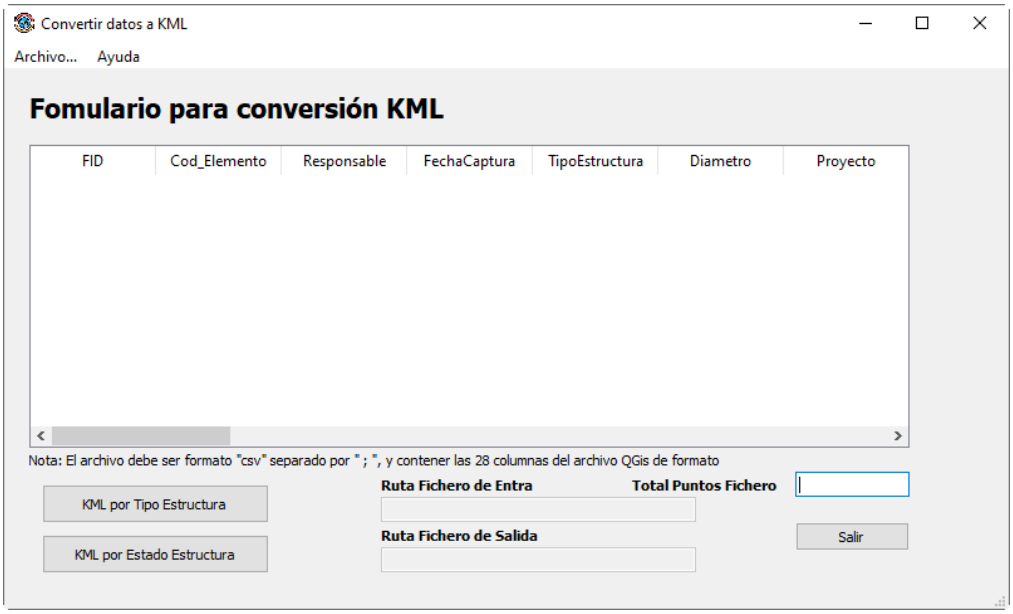


Ilustración 27 - Ventana de complemento para conversión de archivo a formato KML

Estructura del código del complemento

El complemento es simple en su configuración ya que el interés es el de generar únicamente una interfase de conversión que se ajuste a la estructura del documento principal que se estructuró en QGIS, razón por la cual el menú se limita a la apertura del fichero con un menú abrir, un limpiador (Limpiar Archivo) para cuando se ha seleccionado un archivo herrado y la opción de salir del programa que se agrega a los botones de minimizar, ampliar y cerrar con los que cuenta originalmente la ventana.

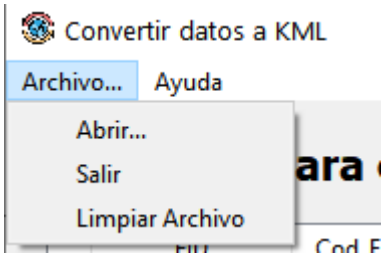


Ilustración 28 - Menú de utilización de la aplicación

Para abrir el fichero se entrará desde el menú “Archivo”, que desplegará inmediatamente una ventana de buscar archivo de Windows, si se selecciona un archivo que no cumpla con la estructura del archivo principal de QGIS, mostrará un error de lectura del fichero, indicando que éste no se ajusta a la estructura principal, es decir, que este tiene en su estructura las 28 columnas indicadas en el Anexo No.1, a continuación el programa no continuará en ejecución.

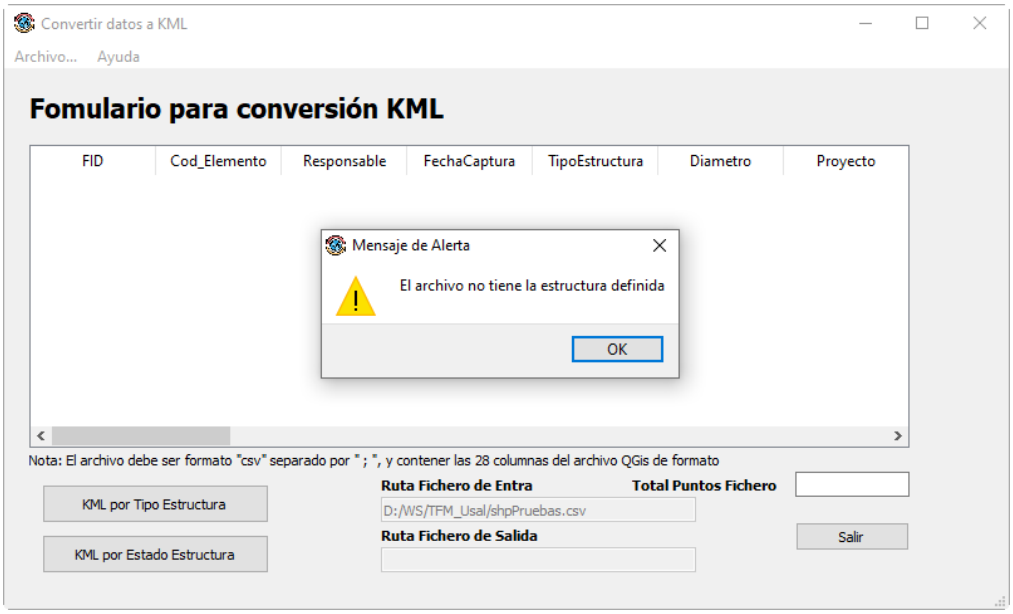


Ilustración 29 - Mensaje de error por estructura de archivo " csv "

Si el fichero cumple con los requisitos de lectura, lo cargará y reflejará en pantalla los datos tomados, adicionalmente indicará la ruta de localización del archivo de entrada en pantalla y el número de registros que contiene el archivo.

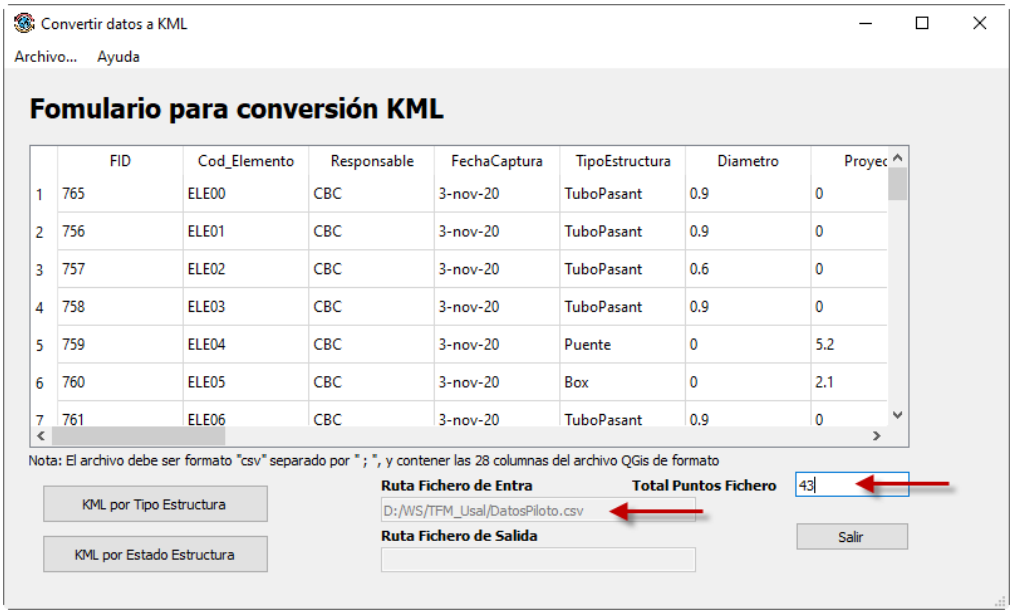


Ilustración 30 - Ventana de carga exitosa de fichero " csv ".

La estructura de los archivos de generación de esta ventana esta definida como se dijo anteriormente en C++ con QT, para el caso se establecen similares archivos de títulos (HEADERS)

" *.h" con la definición de los procedimientos en archivos fuente de los procedimientos que están alojados la subcarpeta SOURCES finalmente la parte gráfica o visual de la aplicación queda alojada en la subcarpeta FORMS, allí básicamente se establece el formulario que pudimos visualizar en la Ilustración 28 - Menú de utilización de la aplicación, esta estructura vista desde el Qt es la siguiente:

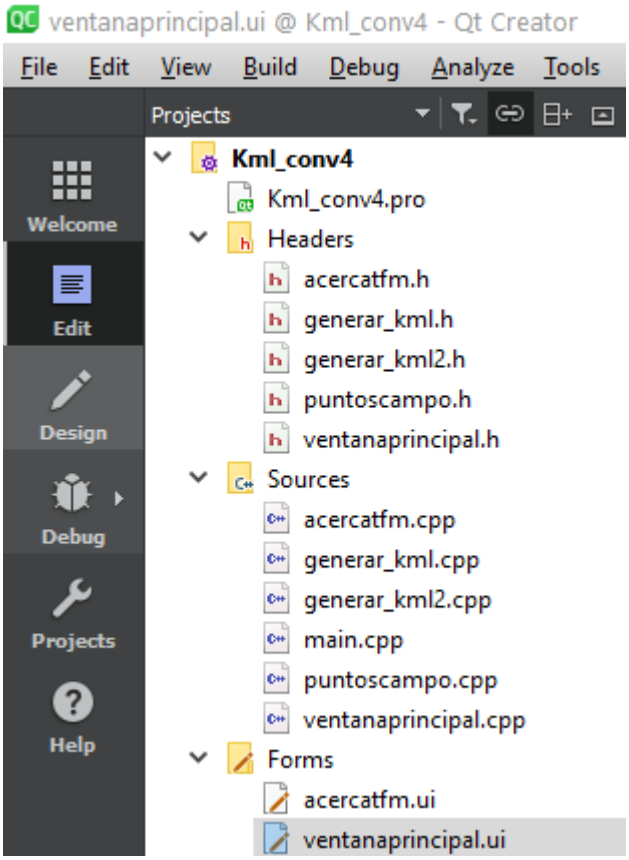


Ilustración 31 - Estructura de archivos que hacen parte del complemento

Ya en el detalle de los "módulos" que lo componen, y lo que hace cada uno tenemos el siguiente diccionario de set de archivos:

- *ventanaprincipal.cpp:* Este fichero integra todos los procedimientos que se llevan a cabo con la ejecución de la aplicación, desde éste se invocan a los demás procedimientos para lograr el producto deseado.
- *puntoscampo.cpp:* Con este fichero se parametrizan los métodos constructores de los tipos de datos que contiene la lista o tabla de datos que se leen con la carga del CSV, esto se hace a través del constructor por defecto de parámetros, que hemos llamado PuntosCampo que lleva en paralelo el descructor que es el método que se invoca automáticamente cuando el objeto se destruye y por último el constructor de copia inicializa el objeto recién creado.
- *main.cpp: Es la función de punto de inicio del programa, no contiene programación alguna a la que se establece por defecto con la creación del proyecto en QT.*
- *Generar_kml.ccp: Define la estructura del archivo KML en el encabezado, el cuerpo (proceso iterativo para cada punto) y el final del archivo, en este caso es el encargado de escribir o crear el archivo que se selecciona con la opción "KML POR TIPO ESTRUCTURA".*
- *Generar_kml2.ccp: Define la estructura del archivo KML en el encabezado, el cuerpo (proceso iterativo para cada punto) y el final del archivo, en este caso es el encargado de*

escribir o crear el archivo que se selecciona con la opción "KML POR ESTADO ESTRUCTURA".

- *Acerca_tfm.ccp: Ejecuta el segundo formulario creado con los metadatos del desarrollo.*

Dependiendo del archivo de desarrollo se emplearon algunas librerías QT5 que permiten la utilización de funcionalidades particulares de la programación en QT, cada una de ellas permite la invocación de comandos en la programación del código para realizar determinadas funcionalidades

```
#include <QMainWindow>
#include <QFileDialog>
#include <QFile>
#include <QTextStream>
#include <stdlib.h>
#include <QList>
#include <QTableWidget>
```

Apertura y carga de archivo " csv "

La apertura o carga del archivo permite alojar datos iniciales en la aplicación como la ruta de entrada o "rutaoriginal" con el path del archivo seleccionado, la cantidad de puntos "contarpuntos", igualmente activa un procedimiento de lectura del archivo a manera de lista del conjunto de datos, esto lo hace invocando la *QTableWidget* definida en el formulario principal Ui definido con el QT Deginer

```
1. void VentanaPrincipal::on_actionAbrir_triggered()
2. {
3. QString formatos;
4. formatos = "Archivos de campo (*.csv)";
5. QFileDialog *dialogoabrir=new QFileDialog(this,"Abrir
   archivos",".",formatos);
6. QString ruta, linea;
7. QStringList partes; //Creo una lista de cadenas para almacenar los datos
8. Generar_KML convertir;  // Creo un varible de tipo clase Generar_KML
9. //Creo una lista de cadenas para almacenar los atributos
10.     PuntosCampo *unpunto; //Defino un puntero para puntoscampo
11.     this->setCursor(Qt::WaitCursor);//Pongo el reloj de Arena

12.     int i, contarpuntos=0; //Inicializó los contadores a 0

13.     if(dialogoabrir->exec()==QDialog::Accepted){
      a. ruta = dialogoabrir->selectedFiles().first();
      b. // Guardo la ruta del archivo
      c. QFile leer(ruta);
      d. leer.open(QIODevice::ReadOnly);//Abro el archivo para lectura
      e. ui->txt_rutaoriginal->setText(ruta);
      f. QTextStream leerstream(&leer);//Para poder leer linea a linea

14.     while (!leerstream.atEnd()){
          i.  linea=leerstream.readLine(); // Leo la linea
         ii.  partes=linea.split(';',QString::SkipEmptyParts); // Arranca en la
              posición " ; "
```

```
15.     // Arranca en la posición " ; " y recorre la linea cada que encuentre
   ese carácter.
16.     // y se salta los espacios.
            i.  if(partes.length()==28){
           ii.  unpunto=new
                PuntosCampo(partes[0].toInt(),partes[26].toDouble(),partes[25].to
                Double());
17.     // Asigno a PuntosCampo los 3 atributos principales fid, longitud y
   latitud
            i.  unpunto-> idelem=partes[1];
           ii.  unpunto-> tipoelemento=partes[4];
          iii.  unpunto-> diametro=partes[5].toDouble();
           iv.  unpunto-> ancho=partes[6].toDouble();
            v.  unpunto-> alto=partes[7].toDouble();
           vi.  unpunto-> longitud=partes[8].toDouble();
          vii.  unpunto-> responsable=partes[2];
         viii.  unpunto-> observacion=partes[20];
           ix.  unpunto-> estconducto=partes[9];
            x.  unpunto-> imagen1=partes[21];

           xi.  if(contarpuntos<200){
18.     //Limito el almacenamiento de puntos en la tabla a 200 registros
         (1)    ui->tw_tabla->setRowCount(ui->tw_tabla->rowCount()+1); //Añade
                fila nueva en la tabla
         (2)    // Ahorar creo un bucle que recorra cada campo de vector y lo
                integre a la lista
         (3)    for (i=0;i<partes.size();i++) {
            (a) ui->tw_tabla->setItem(ui->tw_tabla->rowCount()-1,i,new
                QTableWidgetItem(partes[i]));
            (b) }
         (4)    }// Fin del " if " anidado dentro del while
         (5)    lista_puntos.append(*unpunto);//Añado el punto a la lista
         (6)    contarpuntos++; //Incremento el contador
            (a) }else{
         (7)    validacion=0;//si alguna fila no tiene los 28 campos lo pongo a 0
         (8)    }//fin comprobación 28 columnas
      ii)       } // Fin del While principal

   b) leer.close();//Cierra el fichero

      a. ui->txt_puntos->setText(QString::number(contarpuntos));
19.     // Agrego el numero de puntos en el cuadro
      a. if(validacion==0){
                        a. QMessageBox::warning (this,"Mensaje de Alerta","El
                           archivo no tiene la estructura definida");
                        b. ui->tw_tabla->setRowCount(0); //Limpia la tabla;
                    2. }
20.     } // Fin del If Principal
21. this->setCursor(Qt::ArrowCursor);//Dejo el cursor normal
```

En el código expuesto, las líneas 1 a 9 permiten emplear la ventana de apertura de archivos de windows para que el usuario interactúe en la búsqueda y definición de la misma, en la línea 10 inicializo la lista de datos que esta alojada en el constructor de parámetros y la apunto a una variable que tendrá la estructura predefinida desde el constructor denominada *"unpunto"*, luego ponemos un reloj de arena con la línea 11 en el evento que se demore mucho la carga, en las 12 y 13 se hace

la lectura del fichero línea por línea siempre y cuando el fichero se ajuste a la estructura de la tabla principal definida con las 28 columnas, pero para cada línea del fichero se hace la división de cada atributo empleando el separador definido, en este caso " ; ", ya en la línea 14.g.v. se crea una lista denominada "unpunto" que mediante un bucle tipo "while" alojará los datos principales de registro (línea) según la siguiente estructura.

```
PuntosCampo(int fid, double longitud, double latitud);

//Constructor de parámetros
```

Esto lo interpretamos en el código de la siguiente manera:

```
PuntosCampo(partes[0].toInt(),partes[26].toDouble(),partes[25].toDouble()
);
```

Los siguientes literales almacenan en la lista los datos de cada atributo en la variable establecida según el orden definido en el QTableWidget inicializando el contador "validación" en 1, que permitirá asegurar que la estructura del archivo se ajuste a la del modelo definido para que no se generen errores posteriormente con la escritura del archivo KML. La estructura de cada punto fue definida en el constructor (Header "PuntosCampo.h"), precisando que no se requiere la configuración de todos los atributos de la tabla y como lo vemos en la definición solo hacemos la configuración de 3 valores, el identificador del punto y sus coordenadas longitud y latitud

Los datos almacenados para cada variable son alojados en una lista "lista_puntos" mediante de procedimiento *"append"*, (Linea 18.(5)) en un proceso iterativo que se repite hasta que se termine de leer cada una de las líneas del fichero "csv" siempre y cuando este no contenga mas de 200 puntos. A la terminación de la lectura se cerrará el fichero y se guardarán en memoria los datos registrados en la lista y el contador de iteraciones que permite establecer cuando puntos (registros) hicieron parte del proceso.

Con el fichero cargado adecuadamente lo que sigue es la escritura del archivo nuevo que se desea crear que tendrá extensión KML, para ello el usuario definirá la rutina que desea aplicar, esta parte del código será explicada a continuación:

Conversión KML por "Tipo Estructura"

La primera tipología de conversión dará prelación a mostrar gráficamente el elemento principal "TuboPasant" que es la estructura que regularmente encontramos en vías de Colombia, esta es la que tiene un tubo circular en su interior:

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

Ilustración 32 - Simbología para estructuras con tubería "TuboPasant"

	 http://maps.google.com/mapfiles/kml/paddle/ltblu-blank.png

Ilustración 33 - Simbología para estructuras tipo Box, Puente o Pontón

La definición de la estructura del código es simple si se siguen las indicaciones que para tal efecto se establecen en la documentación del lenguaje (Google Developers - KML, 2021), para ello se identifican tres componentes principales en la estructura que se repetirá con algunas variaciones independientemente de si estamos trabajando un archivo tipo punto, línea o polígono, la cabecera del archivo, el cuerpo (repetitivo para cada elemento) y el fin del documento. Tendremos variaciones particulares según la personalización que pretendamos realizar.

Para el caso particular de nuestro proyecto, tenemos la conformación de un archivo tipo "**punto**", con la definición de algunos íconos especiales, que formarán parte del estilo en el formato del archivo, para ello agregaremos inmediatamente después de la definición de la cabecera los "estilos" propios con los que se desea visualizar la mostrar la información final como lo vemos en el código a continuación que refleja la estructura del archivo "Generar_KML.cpp" dentro de la estructura del programa para clasificación por tipo de estructura:

```
1. void Generar_KML::agregar_comienzo(){
```

```
2. ListaFichero.append(QString("<?xml version=\"1.0\" encoding=\"UTF-
   8\"?>"));
3. ListaFichero.append("<kml xmlns=\"http://www.opengis.net/kml/2.2\">");
4. ListaFichero.append(QString("<Document>"));
5. ListaFichero.append("<Folder>");
6. ListaFichero.append("    <name>""Inventario Infraestructura""</name>");
7. // el nombre de carpeta que será abierta una vez se cargue el archivo en Google
   Earth
8. ListaFichero.append("    <visibility>0</visibility>");
9. ListaFichero.append("    <description>""Obras de
   drenaje""</description>");

10.  //Definición de estilos de indicadores para estructuras
11.  //http://kml4earth.appspot.com/icons.html... Ruta para tipos de iconos

12.  //Estilo 2: globeIcon
13.  ListaFichero.append("    <Style id=\"globeIcon\">");
14.  ListaFichero.append("       <IconStyle>");
15.  ListaFichero.append("       <scale>1.1</scale>");
16.  ListaFichero.append("         <Icon>");
17.  ListaFichero.append("
   <href>http://maps.google.com/mapfiles/kml/paddle/ylw-blank.png</href>");
18.  ListaFichero.append("         </Icon>");
19.  ListaFichero.append("       </IconStyle>");
20.  ListaFichero.append("    </Style>");
```

La cabecera del archivo esta estructurada en las líneas 1 a 9, dado que el lenguaje nativo del formato KML es el XML y en este caso el Estándar que lo se escribe en el encabezado corresponde a versión 2.2. y actualmente ésta referencia invoca la V2.3 automáticamente (Open Geospatial Consortium, 2021),.Es importante recalcar que debemos ser muy rigurosos en la colocación apropiada de cada línea del archivo, ya que de no cumplirse estrictamente la forma de escritura, el archivo no funcionará. Visualmente esta parte del código queda escrita así:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<Folder>
    <name>Inventario Infraestructura</name>
    <visibility>0</visibility>
    <description>Obras de drenaje</description>
```

Ilustración 34 - Estructura del encabezado de archivo KML.

Luego definiremos los estilos de íconos para mas adelante en el código poder hacer el llamado según la casuística de clasificación que se desee, en nuestro caso como en esta tipología solo se quiere resaltar el tipo de estructura, para el programa en definición se empleó un condicional " if " que permite categorizar según valor del dato de "TipoEstructura" de la tabla, esta parte del código queda como se muestra a continuación:

```
//Estilo 2: globeIcon
    ListaFichero.append("    <Style id=\"globeIcon\">");
    ListaFichero.append("       <IconStyle>");
    ListaFichero.append("       <scale>1.1</scale>");
    ListaFichero.append("         <Icon>");
```

```
    ListaFichero.append("
<href>http://maps.google.com/mapfiles/kml/paddle/ylw-blank.png</href>");
    ListaFichero.append("         </Icon>");
    ListaFichero.append("       </IconStyle>");
    ListaFichero.append("    </Style>");
```

Es importante resaltar que la definición de estilos es personalizada para cada archivo KML por lo que es necesario que vaya contenida en cada generación, como se observa en el código propuesto el ícono es invocado a través de la sentencia "href", que direcciona a la lectura del mismo en una dirección de internet provista por el mismo Google (Google Earth - Iconos , 2021)

Para el caso particular del fichero con clasificación por Tipo de Estructura, lo que se hizo fue incluir un condicional que haga la clasificación según el valor de este atributo y de acuerdo al resultado muestre un tipo de ícono o el otro.

```
//Clasificación según el tipo de la estructura
            if (lista_puntos.at(i).tipoelemento == "TuboPasant") {
                ListaFichero.append("   <styleUrl>#globeIcon</styleUrl>");
//Tipo de Marcador 1 para tuberías
            }
            else {
                ListaFichero.append("   <styleUrl>#blueglobeIcon</styleUrl>");
//Tipo de Marcador 5 para otras estructuras
      }
```

Visualización de imagen de referencia

Una de las intenciones principales que motivaron a la realización de este complemento fue la de poder estructurar los atributos que se quería mostrar caprichosamente, y principalmente que el cliente pudiese ver la imagen de cada sitio con la visualización tipificada de los atributos principales o de mayor interés.

Mostrar una imagen cuando se pinche cacha punto por el usuario, implica que estas imágenes estar estén alojadas en una carpeta local o también podrán referirse a una dirección en un servidor, por practicidad en el manejo eficiente de la información se decidió alojarlas localmente en una carpeta compartida (RegFotografico), esta carpeta que contiene las imágenes deben ubicarse en el directorio raíz donde se ubique posteriormente el archivo KML, esto facilitará la lectura y visualización de la información contenida en ellas.

El código para visualización de la imagen esta contenido dentro de las descripción de cada elemento, luego éste código se sitúa en el iterador para conformación de la estructura de datos por elemento, y se establece de la siguiente manera:

```
1. ListaFichero.append("   <description>");//Inicio etiqueta observación

2. ListaFichero.append("    <![CDATA[<h3>Inventario Infraestructura TFM
   USAL</h3><br>");
```

```
    // Ruta de imagen a Visualizar;
3.  ListaFichero.append(QString("    <img style=\"max-width:300px;\"")
    .append(" src=\"").append(QString(lista_puntos.at(i).imagen1
          )).append("\">"))
    // Conformación de una tabla para desplegar algunos atributos
4.  ListaFichero.append("     <br><table border=\"2\" padding=\"1\">");
5.  ListaFichero.append(QString("     <tr><td>Estado de la estructura
    </td><td>").append(QString(lista_puntos.at(i).observacion))
    .append("</td></tr>"));
6.  ListaFichero.append(QString("     <tr><td>Estado del conducto</td><td>")
      .append(QString(lista_puntos.at(i).estconducto)).append("</td></tr>"));
7.  ListaFichero.append(QString("     <tr><td>Responsable</td><td>")
      .append(QString(lista_puntos.at(i).responsable)).append("</td></tr>"));
8.  ListaFichero.append(QString("     <tr><td>FechaRegistro</td><td>")
      .append(lista_puntos.at(i).fechareg).append("</td></tr>"));
9.  ListaFichero.append("     </table>]]>");
10.  ListaFichero.append("   </description>");//Fin etiqueta observación
```

La línea en la que se incluye la ruta de la imagen es la linea macada con el numeral 3. que como podemos ver adicionalmente en ella se define también el tamaño de la imagen, resto de los literales permiten establecer un formato de tabla de datos que será el que veamos cuando se realice el despliegue de visualización al momento de pinchar el punto. En la ejecución del archivo ya estructurado podremos ver algo como lo que se muestra a continuación:.

```
<Placemark>
  <name>ELE02</name>
   <visibility>1</visibility>
   <LookAt>
      <latitude>-74.0051</latitude>
      <longitude>4.94517</longitude>
   </LookAt>
   <styleUrl>#flechaRoja</styleUrl>
   <Point>
      <coordinates>-74.0051,4.94517</coordinates>
   </Point>
   <description>
    <![CDATA[<h3>Inventario Infraestructura TFM USAL</h3><br>
    <img style="max-width:300px;" src="RegFotografico/1/1602332979930.jpg">
     <br><table border="2" padding="1">
     <tr><td>Estado de la estructura</td><td>Estructura para reemplazar</td></tr>
     <tr><td>Estado del conducto</td><td>Destruido</td></tr>
     <tr><td>Responsable</td><td>CBC</td></tr>
     </table>]]>
   </description>
</Placemark>
```

Ilustración 35 - Aparte de archivo estructurado KML - Atributos e imagen

El resultado de este código ya en la lectura del punto sería:

Ilustración 36 - Visualización de punto por tipo estructura

Adicionalmente la visualización general del archivo KML en Google Earth

Ilustración 37 - Ventana en Google Earth de archivo KML generado por Tipo de Estructura

Conversión a KML por "Estado de la estructura"

Para la segunda tipología de conversión dará prelación a mostrar gráficamente el estado de las estructuras sobre los cuales se establecieron cinco dominios iniciales (Bueno, Destruido, Agrietado, Semidestruido e Indeterminado), según el caso de uso y sus condiciones de análisis se agruparon las categorías irrelevantes en un solo símbolo y las que requieren se una análisis e identificación particular con una simbología diferente, así se verán en el mapa:

- Bueno Estructuras en buen estado -	 http://maps.google.com/mapfiles/kml/pushpin/grn-pushpin.png
- Destruido - Semidestruido - Agrietado Estructuras que requieren algún tipo de intervención.	 http://maps.google.com/mapfiles/kml/shapes/arrow.png
- Indeterminado Indicador de control para ajustar procesos de campo.	 http://maps.google.com/mapfiles/kml/paddle/I.png

Ilustración 38 - Simbología para estructuras por novedad de estado

En la estructura definida para el código C++ del convertidor se mantiene, la visualización de la imagen y se ajustan los criterios para la definición de las puntos de marca, así mismo ajustamos los atributos que se visualizan en la tabla, todo esto después de la definición del encabezado de archivo en un "for" iterativo para cada uno de los puntos.

```
estado_conducto = lista_puntos.at(i).estconducto;

    if (estado_conducto == "Destruido" || estado_conducto == "Semidestruido" ||
estado_conducto== "Agrietado"){
    ListaFichero.append("   <styleUrl>#flechaRoja</styleUrl>");//Marcador por
estado "1" para estructuras con falla
    }
    else if (estado_conducto == "Bueno")
    {
       ListaFichero.append("   <styleUrl>#normalPlacemark</styleUrl>");//Tipo de
Marcador 2 para estructuras en buen estado
    }
    else
    {
    ListaFichero.append("   <styleUrl>#highlightPlacemark</styleUrl>");//Tipo de
Marcador 3 para que no se identifico estado adecuadamente.
```

```
    }
ListaFichero.append("   <Point>");//Marcador abierto para coordenadas del punto
ListaFichero.append(QString("
<coordinates>").append(QString::number(lista_puntos.at(i).x)).append(",").append
(QString::number(lista_puntos.at(i).y)).append("</coordinates>"));
//Que sigue el indicador de entidad
ListaFichero.append("   </Point>");
//Cierre de marcador para coordenadas del punto

ListaFichero.append("   <description>");
//Inicio etiqueta observacion
ListaFichero.append("    <![CDATA[<h3>Inventario Infraestructura TFM
USAL</h3><br>");
ListaFichero.append(QString("    <img style=\"max-width:300px;\"").append("
src=\"").append(QString(lista_puntos.at(i).imagen1)).append("\">"));
ListaFichero.append("     <br><table border=\"2\" padding=\"1\">");
ListaFichero.append(QString("     <tr><td>Estado del conducto</td><td>")
.append(QString(lista_puntos.at(i).estconducto)).append("</td></tr>"));
ListaFichero.append(QString("     <tr><td>Observación de
estado</td><td>").append(QString(lista_puntos.at(i).observacion)).append("</td><
/tr>"));
ListaFichero.append(QString("     <tr><td>Responsable</td><td>").append(QString(
lista_puntos.at(i).responsable)).append("</td></tr>"));
ListaFichero.append(QString("     <tr><td>Fecha de registro</td><td>").append(
QString(lista_puntos.at(i).fechareg)).append("</td></tr>"));
ListaFichero.append("     </table>]]>");
ListaFichero.append("   </description>");//Fin etiqueta observacion
```

Aspectos relevantes del código adicionales a los ya señalados anteriormente, que se identifican con la ejecución, "*ListaFichero"* es una lista a la que se van agregando valores mediante del método *"append"*, internamente la especificación que se da es lo que corresponde a cada línea del archivo que estamos conformando. Por otra parte el condicional "If" podría ser complementado si requerimos desplegar mas elementos incluyendo algunos condicionales anidados adicionales a los que se muestran en el ejemplo anterior.

Así las cosas, el mismo elemento del ejemplo anterior, lo veríamos en esta nueva estructura para lenguaje KML de la siguiente manera:

```
<Placemark>
  <name>ELE02</name>
   <visibility>1</visibility>
   <LookAt>
      <latitude>-76.3381</latitude>
      <longitude>3.54399</longitude>
   </LookAt>
   <styleUrl>#globeIcon</styleUrl>
   <Point>
      <coordinates>-76.3381,3.54399</coordinates>
   </Point>
   <description>
    <![CDATA[<h3>Inventario Infraestructura TFM USAL</h3><br>
    <img style="max-width:300px;" src="RegFotografico\2\1602333511295.jpg">
     <br><table border="2" padding="1">
     <tr><td>Tipo de estructura</td><td>TuboPasant</td></tr>
     <tr><td>Estado del conducto</td><td>Agrietado</td></tr>
     <tr><td>Responsable</td><td>CBC</td></tr>
     <tr><td>FechaRegistro</td><td>3-nov-20</td></tr>
     </table>]]>
   </description>
</Placemark>
```

Ilustración 39 - Aparte de archivo estructurado KML por tipo de estructura - Atributos e imagen

El resultado visual de esta codificación podríamos verlo así:

Ilustración 40 - Visualización de punto por estado de la estructura

El resultado general del archivo KML con visualización por tipo de estructura sería así:

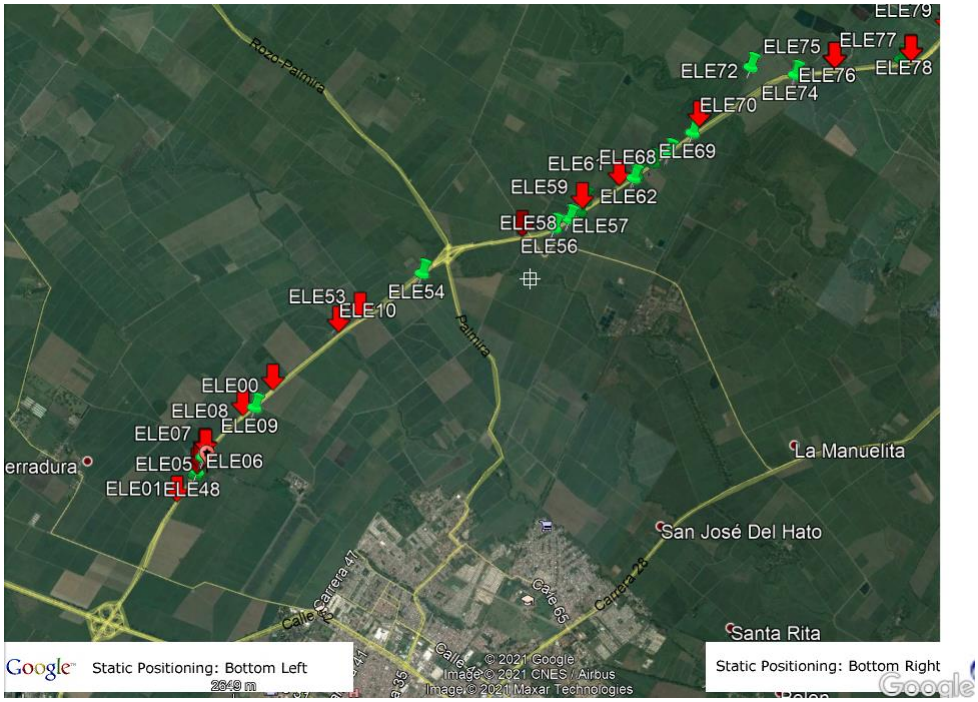


Ilustración 41 - Ventana en Google Earth de archivo KML generado por Estado Estructura

CREACION DE REPOSITORIO DE DATOS EN GITHUB

El código fuente del convertidor KML ha sido compartido en la cuenta de GitHub que creamos con la realización de la maestría, y podrá estar disponible para quienes quieran hacer sus propias adaptaciones para proyectos propios, esto estará alojado en la ruta URL https://github.com/cbcortesr/convert2KML, con una configuración pública para que cualquier usuario la pueda descargar

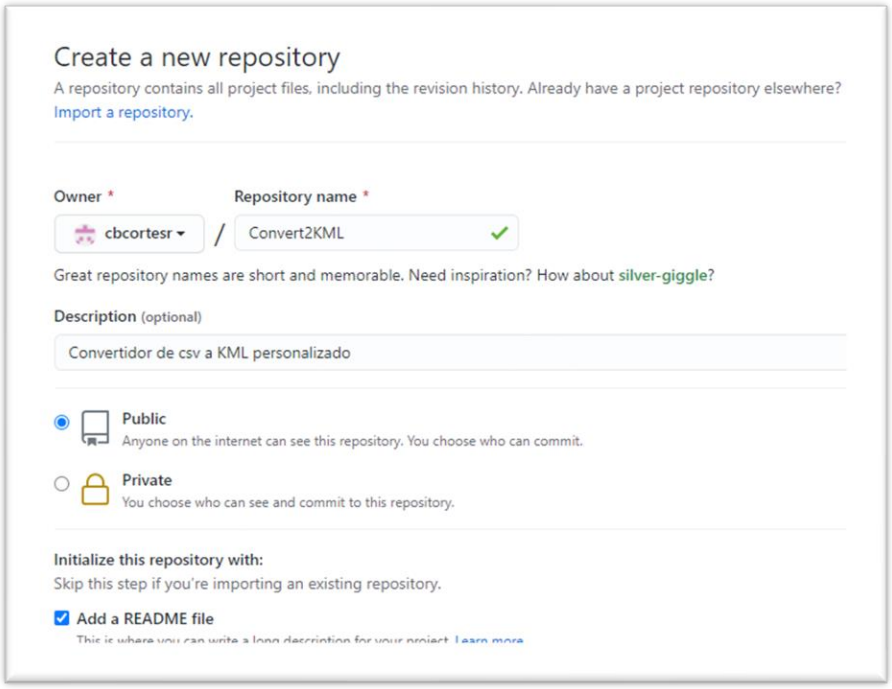


Ilustración 42 - Repositorio GitHub con el código fuente de convertidor csv a kml.

Pare efectos de poder lograr un entendimiento de las funcionalidades de programa se hizo un archivo descriptivo en el que adicionalmente se dan las indicaciones básicas para correrlo a través de QT.

RESULTADOS

1) La validación del éxito del proceso nos ha permitido establecer una metodología a seguir para la realización de trabajos campo con la captura de los datos y el procesamiento de la información obtenida que la podemos resumir en la siguiente secuencia de pasos:

Paso del proceso	Resultado
Proyecto QGIS	Estructura de proyecto para QFIELD en carpeta compartida de Google drive
Sincronización en Drive Sync	Paso la información base del proyecto al equipo móvil.
Captura de información en campo	Archivo geográfico con atributos de la realidad en terreno y registro fotográfico (QField crea carpeta adjunta con el registro de imágenes tomadas).
Sincronicación con Drive Sync	Al final de cada día de trabajo se sincroniza el equipo móvil con la carpeta compartida de Google Drive para descargar registro de atributos para todos los elementos y el registro fotográfico.

Consolidación de información	En QGIS consolidaremos toda la información de campo y se realizada el proceso de control de calidad QC y ajustes previo a la exportación del archivo "csv" para generación del KML.
Conversión de KML	Se genera el archivo KML según la necesidad para entrega de resultados parciales al cliente.

2) Una vez sincronizado el proyecto al equipo móvil encontramos que los tiempos de procesamiento se redujeron en un 60% comparado con los resultados de un proyecto anterior que se trabajó con formatos en papel, pasando de 12 minutos por elemento a 5 minutos como máximo.
3) Con relación al procesamiento de la información en oficina, la ganancia de la ejecución de un proyecto empleando la metodología propuesta se logra la reducción en tiempos pasando de 15 días a 3 días para la entrega de resultados, con el valor agregado que se suprimen trabajos adicionales que se debieron realizar en el trabajo con papel así:
   a) Se suprime escaneo de las fichas de campo, con reducción de costos y tiempos de procesamiento (1 semana), en trabajos sobre vía se debe esperar a que el equipo vuelva de campo y esto podría implicar una semana adicional.
   b) Se suprime la digitación de fichas de campo, reducción de costos, tiempos de procesamiento y personal de control de calidad de digitación (1 semana).
   c) Se identifican en el QC problemas o falencias de la información que pueden ser corregidas oportunamente ya que los resultados se tienen al final de cada día de trabajo en campo.
   d) Se elimina el proceso de organización de imágenes y asignación a puntos ya que los datos vienen directamente capturados en campo. Consecuentemente se suprime el alto margen de error que éste implicaba ya que solo se podía realizar por lectura de la hora de toma del registro fotográfico.
4) Con la utilización del "convertidor KML" se logra satisfacer una necesidad inmediata de control de avances que se tenía, así como la posibilidad de contar con un termómetro de resultados sobre el avance que se tenía de los trabajos en campo.
5) La presentación de avances a través de un mapa generado en un archivo formato KML personalizado, permite ilustrar y caracterizar los resultados con información visual y fresca, generando un gran valor agregado para los clientes, ya que esto se realiza a través de Google Earth que es una herramienta asequible, sin costo y de fácil manejo, garantizando que los resultados lleguen directo a la cabeza del proyecto sin complicar la forma de hacerlo.

DESARROLLOS FUTUROS

Las posibilidades que se abren para la realización de cualquier tipo de trabajo son innumerables con el QIELD, aunque durante la realización de este trabajo se intentó implementar validaciones adicionales programadas con Python, éstas no fluyen con naturalidad dentro de la interface actual, presentan errores o no son adoptadas en el equipo móvil, por ello, es este aspecto al que sin duda se dedicará en el futuro cercano mayor esfuerzo y dedicación, esto acompañado en la profundización en los conocimientos de programación de Python, hecho que seguro influyó en el logro de resultados en este aspecto en los temas que se trabajaron bajo esta plataforma.

El proceso de implementación del módulo de conversión KML es totalmente factible como un plugin en Python para QGIS, por lo que de manera inmediata se considera pertinente hacer la migración del complemento que inicialmente fue desarrollado en C++ ahora en PyQGIS, esto podría reducir el intercambio a otra interfase y evitar la manipulación del archivo “csv” generado con la exportación.

La publicación del registro fotográfico en un servidor corporativo que permita mantener por un tiempo limitado el registro fotográfico para cada uno de los puntos generados se consideraría una mejora interesante para facilitar y reducir el flujo de información, no adiciona funcionalidades a lo que se ha implementado pero permite evitar inconvenientes en la lectura del archivo KML cuando el usuario final de éste no es una persona experta en la manipulación de información con rutas relativas. Sería una alternativa a esta propuesta manejar ficheros KMZ que permitirían la integración del registro fotográfico como un conjunto dentro del mismo archivo, sin embargo, debe evaluarse la conveniencia de esta alternativa considerando que el volumen de información en formatos de imagen podría ser bastante alto y esto podría ir en detrimento de la implementación cuando se pierde la facilidad en la transferencia de un fichero liviano como lo es el KML

BIBLIOGRAFÍA

Assam Software, 2021. *Locus.* [En línea] Available at: https://play.google.com/store/apps/details?id=menion.android.locus.gis&hl=es_CO&showAllReviews=true [Último acceso: 15 1 2021].

ESRI, 2021. *ArcGIS Survey123.* [En línea] Available at: https://play.google.com/store/apps/details?id=com.esri.survey123&hl=es_CO [Último acceso: 18 01 2021].

Gis Cloud, 2021. *Mobile data Collection, Google Play.* [En línea] Available at: https://giscloud.com [Último acceso: 18 01 2021].

Google Developers - KML, 2021. *Keyhole Markup Language.* [En línea] Available at: https://developers.google.com/kml/documentation/kml_tut?hl=es [Último acceso: 25 03 2021].

Google Earth - Iconos , 2021. *Google Earth / Maps Icons.* [En línea] Available at: http://kml4earth.appspot.com/icons.html#pal2 [Último acceso: 15 06 2021].

Modelio, 2021. *Modelio - Open Source Modeling Enviroment.* [En línea] Available at: http://www.modelio.org [Último acceso: 15 02 2021].

Open Geospatial Consortium, 2021. *OGC KML 2.3.* [En línea] Available at: http://docs.opengeospatial.org/is/12-007r2/12-007r2.html#8 [Último acceso: 1 06 2021].

OpenGIS.ch, 2021. *QField para QGIS.* [En línea] Available at: https://play.google.com/store/apps/details?id=ch.opengis.qfield&hl=es_CO [Último acceso: 18 01 2021].

Opengisch, 2021. *QField.org.* [En línea] Available at: https://github.com/opengisch/QField/releases/ [Último acceso: 2 03 2021].

PoloSoftech, 2021. *Gis Mapper.* [En línea] Available at: https://play.google.com/store/apps/details?id=com.globalgnss.gismapper&hl=es_CO [Último acceso: 18 01 2021].

QField - User documentation, 2021. *QField.org.* [En línea] Available at: https://qfield.org/docs/prepare/index.html [Último acceso: 15 05 2021].

QField -Sync, 2021. *Soporte para QField.* [En línea] Available at: https://qfield.org/docs/synchronise/qfieldsync.html [Último acceso: 30 04 2021].

QGIS, 2021. *Manual de aprendizaje QGIS.* [En línea] Available at: https://docs.qgis.org/3.16/es/docs/training_manual/create_vector_data/forms.html [Último acceso: 15 03 2021].

ANEXOS

Diccionario Estructura principal de datos “InvEstructura”

Numero	Nombre Atributo	Variable	Tipo de dato	Longitud	Presisión	Descriptor
0	fid	id	Int	3	0	Consecutivo de captura (Automático)
1	Id_Elemento	idelem	String	6	0	Identificador o nombre de elemento en campo
2	ResponsableCampo	responsable	String	20	0	Nombre o iniciales de quien toma la información
3	FechaCaptura	fechacap	date	0	0	Fecha con hora de captura (Automático)
4	TipoEstructura	tipoestructura	String	25	0	Tipo de estructura
5	Diametro	diametro	double	0	0	Diametro de tubería
6	Ancho	ancho	double	0	0	Ancho de ducto
7	Alto	ancho	double	0	0	Altura de ducto
8	LongTv	alto	double	0	0	Longitud del tramo de ducto
9	Est_Conducto	est_conducto	String	20	0	Estado general del conducto
10	Funcionamiento	funcionamiento	String	20	0	Estado de funcionamiento del conducto
11	Est_Cabezal	est_cabezal	String	20	0	Estado general de los cabezales
12	Cab_Ent	cab_ent	bool	0	0	Confirmación de existencia de cabezal a la entrada
13	Cab_Sal	cab_sal	bool	0	0	Confirmación de existencia de cabezal a la salida
14	Est_GuardaR	est_guardar	String	20	0	Estado general de los guardaruedas
15	GuardaR_Ent	guardar_ent	bool	0	0	Confirmación de existencia de guardaruedas a la entrada
16	GuardaR_Sal	guardar_sal	bool	0	0	Confirmación de existencia de guardaruedas a la salida
17	Est_Aletas	est_aletas	String	20	0	Estado general de las aletas
18	Aleta_Ent	aleta_ent	bool	0	0	Confirmación de existencia de aletas a la entrada
19	Aleta_Sal	aleta_sal	bool	0	0	Confirmación de existencia de aletas a la salida
20	Observaciones	observaciones	String	100	0	Observaciones relevantes determinantes
21	Foto1	foto1	String	100	0	Ruta del registro de la imagen localización general
22	Foto2	foto2	String	100	0	Ruta del registro de la imagen lateral entrada
23	Foto3	foto3	String	100	0	Ruta del registro de la imagen lateral salida
24	Foto4	foto4	String	100	0	Ruta del registro de la imagen ducto
25	Latitud	latitud	double	0	0	Coordenada de Latitud de registro del punto (Automático)
26	Longitud	longitud	double	0	0	Coordenada de Longitud de registro del punto (Automático)
27	Proyecto	proyecto	String	30	0	Nombre del proyecto (Automático)

Ilustración 43 - Diccionario de datos "InvEstructura"

Código de complemento para conversión KML

Header “ventanaprincipal.h”

```
#ifndef VENTANAPRINCIPAL_H
#define VENTANAPRINCIPAL_H

#include <QMainWindow>
#include <QFileDialog>
#include <QFile>
#include <QTextStream>
#include "puntoscampo.h"

QT_BEGIN_NAMESPACE
namespace Ui { class VentanaPrincipal; }
QT_END_NAMESPACE

class VentanaPrincipal : public QMainWindow
{
    Q_OBJECT
public:
    QList<PuntosCampo> lista_puntos;
    explicit VentanaPrincipal(QWidget *parent = 0);
    ~VentanaPrincipal();
private slots:
    void on_actionSalir_triggered();
    void on_actionAbrir_triggered();
    void on_actionCerrar_triggered();
    void on_btn_salir_clicked();
    void on_btn_proceso_clicked();
    void on_btn_archiv_sal_clicked();
    void on_actionAcerca_de_triggered();
    void on_btn_proceso_2_clicked();
    void on_actionLimpiar_Archivo_triggered();
    void on_pushButton_clicked();

private:
    Ui::VentanaPrincipal *ui;
};
#endif // VENTANAPRINCIPAL_H
```

Header “puntoscampo.h”

```
#ifndef PUNTOSCAMPO_H
#define PUNTOSCAMPO_H
#include <QString>

class PuntosCampo // Constructor
{
public:
    //Variables de la clase Punto son publicas, permitiran acceder a ellas.
        int id, ii;
        double x,y,ancho, alto, diametro, longitud;
        QString proyecto,idelem,responsable, tipoelemento, estconducto,
observacion, imagen1, fechareg;
        //Defino las variables con las que predendemos generar alguna salida
adiocional
```

```
        PuntosCampo();                              //Constructor por defecto
        PuntosCampo(const PuntosCampo &otropunto);//Constructor de copia
        PuntosCampo(int fid, double longitud, double latitud);//Constructor de
parametros
};

#endif // PUNTOSCAMPO_H
```

Header "generar_kml.h"

```
#ifndef GENERAR_KML_H
#define GENERAR_KML_H
#include <QStringList>
#include "puntoscampo.h"

class Generar_KML
{
public:
    Generar_KML();
    QStringList ListaFichero; //Variable que almacenará el contenido del KML
linea a linea
    void escribir_KML(QString ruta_fichero); //Almacenará la lista en el fichero
    void agregar_comienzo();  //Escribe en el KML los renglones del comienzo
según protocolo
    void agregar_fin();       //Escribe la parte final del final
    void agregar_Lista_Puntos(QList<PuntosCampo> lista_puntos);
    //Agrega los puntos de la lista al fichero.
};

#endif // GENERAR_KML_H
```

Header "generar_kml2.h"

```
#ifndef GENERAR_KML2_H
#define GENERAR_KML2_H
#include <QStringList>
#include "puntoscampo.h"

class Generar_KML2
{
public:
    Generar_KML2();
    QStringList ListaFichero; //Variable que almacenará el contenido del KML
linea a linea
    void escribir_KML2(QString ruta_fichero); //Almacenará la lista en el
fichero
    void agregar_comienzo();
//Escribe en el KML los renglones del comienzo según protocolo
    void agregar_fin();       //Escribe la parte final del final
    void agregar_Lista_Puntos(QList<PuntosCampo> lista_puntos);
    //Agrega los puntos de la lista al fichero.
};
```

```
#endif // GENERAR_KML_H
```

Header “acercatfm.h”

```
#ifndef ACERCATFM_H
#define ACERCATFM_H

#include <QDialog>

namespace Ui {
class acercaTFM;
}

class acercaTFM : public QDialog
{
    Q_OBJECT

public:
    explicit acercaTFM(QWidget *parent = nullptr);
    ~acercaTFM();

private slots:
    void on_pushButton_clicked();

private:
    Ui::acercaTFM *ui;
};

#endif // ACERCATFM_H
```

Sources “ventanaprincipal.cpp”

```
#include "ventanaprincipal.h"
#include "ui_ventanaprincipal.h"
#include "generar_kml.h"
#include "generar_kml2.h"
#include "acercatfm.h"
#include <QTextStream>
#include <QMessageBox>
#include <QFileDialog>      // Para trabajar con cuadros de dialogo predefinidos
para archivos
#include <stdlib.h>
#include <QList>
#include <QTableWidget>

VentanaPrincipal::VentanaPrincipal(QWidget *parent)
    : QMainWindow(parent),
      ui(new Ui::VentanaPrincipal)
{
    ui->setupUi(this);
    this->setWindowTitle("Convertir datos a KML");
}
```

```
VentanaPrincipal::~VentanaPrincipal()
{
    delete ui;
}

void VentanaPrincipal::on_actionSalir_triggered()
{
    QApplication::quit(); //Para salir de la aplicación
}
void VentanaPrincipal::on_actionAbrir_triggered()
{
    QString formatos;
    formatos = "Archivos de campo (*.csv)";
    QFileDialog *dialogoabrir=new QFileDialog(this,"Abrir
archivos",".",formatos);
    QString ruta, linea;
    QStringList partes; //Creo una lista de cadenas para almacenar los datos
    Generar_KML convertir;  // Creo un varible de tipo clase Generar_KML
    //Creo una lista de cadenas para almacenar los atributos
    PuntosCampo *unpunto; //Defino un puntero para puntoscampo
    this->setCursor(Qt::WaitCursor);//Pongo el reloj de Arena

    int i, contarpuntos=0;  //Inicialización de contadores a 0

    if(dialogoabrir->exec()==QDialog::Accepted){
        ruta = dialogoabrir->selectedFiles().first(); // Guardo la ruta del
archivo
        QFile leer(ruta);
        leer.open(QIODevice::ReadOnly);//Abro el archivo para lectura
        ui->txt_rutaoriginal->setText(ruta);
        QTextStream leerstream(&leer);//Para poder leer linea a linea

while (!leerstream.atEnd()){
            linea=leerstream.readLine(); // Leo la linea
            partes=linea.split(';',QString::SkipEmptyParts); // Arranca en la
posición " ; "
            // Arranca en la posición " ; " y recorre la linea cada que encuentre
ese carácter.
            // y se salta los espacios.
            if(partes.length()==28){
                unpunto=new
PuntosCampo(partes[0].toInt(),partes[26].toDouble(),partes[25].toDouble());
                // Asigno a PuntosCampo los 3 atributos principales fid,longitud
y latitud
                unpunto-> idelem=partes[1];
                unpunto-> tipoelemento=partes[4];
                unpunto-> diametro=partes[5].toDouble();
                unpunto-> ancho=partes[6].toDouble();
                unpunto-> alto=partes[7].toDouble();
                unpunto-> longitud=partes[8].toDouble();
                unpunto-> responsable=partes[2];
                unpunto-> observacion=partes[20];
                unpunto-> estconducto=partes[9];
                unpunto-> imagen1=partes[21];

                if(contarpuntos<200){ //Limito el almacenamiento de puntos en la
tabla a los 200 registros
```

```
                    ui->tw_tabla->setRowCount(ui->tw_tabla->rowCount()+1);
//Añade fila nueva en la tabla
// Ahorar creo un bucle que recorra cada campo de vector y lo integre a la lista
                    for (i=0;i<partes.size();i++) {
                        ui->tw_tabla->setItem(ui->tw_tabla->rowCount()-1,i,new
QTableWidgetItem(partes[i]));
                        }
                }// Fin del " if " anidado dentro del while
                    lista_puntos.append(*unpunto);//Añado el punto a la lista
                    contarpuntos++; //Incremento el contador
            }else{
                 validacion=0;
//si alguna fila no tiene los 28 campos lo pongo a 0
                 }//fin comprobación 28 columnas
        } // Fin del While principal

        leer.close();//Cierra el fichero

        ui->txt_puntos->setText(QString::number(contarpuntos));
// Agrego el numero de puntos en el cuadro
        if(validacion==0){
                    QMessageBox::warning (this,"Mensaje de Alerta","El
archivo no tiene la estructura definida");
                    ui->tw_tabla->setRowCount(0); //Limpia la tabla;
                }
   } // Fin del If Principal
    this->setCursor(Qt::ArrowCursor);//Dejo el cursor normal
}
void VentanaPrincipal::on_actionCerrar_triggered()
{
    this->close();
}


void VentanaPrincipal::on_btn_salir_clicked()
{
    QApplication::quit(); //Para salir de la aplicación
}

void VentanaPrincipal::on_btn_proceso_clicked()
{
    QString formatos, ruta_guardar;
    Generar_KML convertir;  // Creo un varible de tipo clase Generar_KML
    formatos="Archivos KML (*.kml)";
    //indico formatos soportados y con ;;separa un formato de otro
    QFileDialog *dialogoguardar=new QFileDialog(this, "Guardar como KML",
".",formatos);
    dialogoguardar->setAcceptMode(QFileDialog::AcceptSave); //Para que sea el
cuadro de diálogo como de guardar
    if (dialogoguardar->exec()==QDialog::Accepted){
        //si seleccionan un archivo y pulsan abrir hago el resto
        ruta_guardar=dialogoguardar->selectedFiles().first();
        ui->txt_rutasalida->setText(ruta_guardar);

//==============Corro los procesos para la generación del KML====

        convertir.agregar_comienzo(); //Incorpora la cabecera del archivo
```

```
        convertir.agregar_Lista_Puntos(lista_puntos); //Genera el listado de
puntos
        convertir.agregar_fin(); //Incorpora el final del archivo
        convertir.escribir_KML(ui->txt_rutasalida->text());
        QMessageBox::information(this,"Mensaje Informativo","Archivo KML
generado satisfactoriamente");
    }

    else {
        QMessageBox::warning(this,"Mensaje de Error","Primero debe indicar la
ruta de almacenamiento del archivo de salida");
        //mensajeerror.setText()
    }
    //fin del if
}

void VentanaPrincipal::on_btn_archiv_sal_clicked()
{
    QString formatos, ruta_guardar;
    Generar_KML convertir;  // Creo un varible de tipo clase Generar_KML
    formatos="Archivos KML (*.kml)";
    //indico formatos soportados y con ;;separa un formato de otro
    QFileDialog *dialogoguardar=new QFileDialog(this, "Guardar como KML",
".",formatos);
    dialogoguardar->setAcceptMode(QFileDialog::AcceptSave); //Para que sea el
cuadro de diálogo como de guardar
    if (dialogoguardar->exec()==QDialog::Accepted){
        //si seleccionan un archivo y pulsan abrir hago el resto
        ruta_guardar=dialogoguardar->selectedFiles().first();
        ui->txt_rutasalida->setText(ruta_guardar);
    }
}

void VentanaPrincipal::on_actionAcerca_de_triggered()
{
    acercaTFM *ayuda;
    ayuda=new acercaTFM(this);
    ayuda->setWindowTitle("Acerca de Convertior KML");
    ayuda->exec();//Con exec muestra el formulario y hasta que no se cierra
    //no sigo con el principal
    //ayuda->show();//Show nos permitira seguir trabajando en el principal
}


void VentanaPrincipal::on_btn_proceso_2_clicked()
{
    QString formatos, ruta_guardar;
    Generar_KML2 convertir2;  // Creo un varible de tipo clase Generar_KML
    formatos="Archivos KML (*.kml)";
    //indico formatos soportados y con ;;separa un formato de otro
    QFileDialog *dialogoguardar=new QFileDialog(this, "Guardar como KML",
".",formatos);
    dialogoguardar->setAcceptMode(QFileDialog::AcceptSave); //Para que sea el
cuadro de diálogo como de guardar
    if (dialogoguardar->exec()==QDialog::Accepted){
        //si seleccionan un archivo y pulsan abrir hago el resto
        ruta_guardar=dialogoguardar->selectedFiles().first();
```

```
        ui->txt_rutasalida->setText(ruta_guardar);

//==============Corro los procesos para la generación del KML======

        convertir2.agregar_comienzo(); //Incorpora la cabecera del archivo
        convertir2.agregar_Lista_Puntos(lista_puntos); //Genera el listado de
puntos
        convertir2.agregar_fin(); //Incorpora el final del archivo
        convertir2.escribir_KML2(ui->txt_rutasalida->text());
        QMessageBox::information(this,"Mensaje Informativo","Archivo KML
generado satisfactoriamente");
    }

    else {
        QMessageBox::warning(this,"Mensaje de Error","Primero debe indicar la
ruta de almacenamiento del archivo de salida");
        //mensajeerror.setText()
    }
    //fin del if
}

void VentanaPrincipal::on_actionLimpiar_Archivo_triggered()
{
    ui->tw_tabla->setRowCount(0); //Limpia la tabla
    ui->txt_rutaoriginal->setText(""); // Limpia la ruta original
    ui->txt_rutasalida->setText(""); // Limpia la ruta se salida
}


void VentanaPrincipal::on_pushButton_clicked()
{
    ui->tw_tabla->setRowCount(0); //Limpia la tabla
    ui->txt_rutaoriginal->setText(""); // Limpia la ruta original
    ui->txt_rutasalida->setText(""); // Limpia la ruta se salida
    ui->txt_puntos->setText(""); // Limpia cuadro cantidad de puntos de la
ultima generación

}
```

Sources "puntoscampo.cpp"

```
#include "puntoscampo.h"
#include <QtMath>       //Si vamos a calcular algunas operaciones


//Constructor por defecto ============================================
PuntosCampo::PuntosCampo()   //defino valores de inicialización del constructor
{
    id=0;
    idelem="";
    fechareg="";
    ancho=0;
    alto=0;
    diametro=0;
    proyecto="";
    tipoelemento="";
```

```
    responsable="";
    observacion="";
    estconducto="";
    x=0;
    y=0;
    imagen1="";

    }

//Constructor de copia ================================================
PuntosCampo::PuntosCampo(const PuntosCampo &p2)
{   // Asigno los valores del punto recibido a las variables del punto que
estamos creando
    id=p2.id;
    idelem=p2.idelem;
    fechareg=p2.fechareg;
    ancho=p2.ancho;
    alto=p2.alto;
    diametro=p2.diametro;
    proyecto=p2.proyecto;
    tipoelemento=p2.tipoelemento;
    responsable=p2.responsable;
    observacion=p2.observacion;
    estconducto=p2.estconducto;
    x=p2.x;
    y=p2.y;
    imagen1=p2.imagen1;
}

//Constructor de parametros ===========================================
PuntosCampo::PuntosCampo(int fid, double longitud,double latitud )
{
    id=fid;
    idelem="";
    fechareg="";
    ancho=0;
    alto=0;
    diametro=0;
    proyecto=""; //Nombre del proyecto
    tipoelemento="";
    responsable="";
    observacion="";
    estconducto="";
    x=longitud;
    y=latitud;
    imagen1="";
}
```

Sources "main.cpp"

```
#include "ventanaprincipal.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
```

```
    VentanaPrincipal w;
    w.show();
    return a.exec();
}
```

Sources "generar_kml.cpp"

```
#include "generar_kml.h"
#include <QTextStream>
#include <QMessageBox>
#include <QFile>
#include "puntoscampo.h"
#include <QString>

//Generador KML por TIPO DE ESTRUCTURA ====================================
//=========================================================================
===
Generar_KML::Generar_KML()
{
// Constructor por defecto
}

//Escribe en el KML los renglones del comienzo del KML
============================

void Generar_KML::agregar_comienzo(){
    ListaFichero.append(QString("<?xml version=\"1.0\" encoding=\"UTF-8\"?>"));
    ListaFichero.append("<kml xmlns=\"http://www.opengis.net/kml/2.2\">");
    ListaFichero.append(QString("<Document>"));
    ListaFichero.append("<Folder>");
    ListaFichero.append("    <name>""Inventario Infraestructura""</name>");
    ListaFichero.append("    <visibility>0</visibility>");
    ListaFichero.append("    <description>""Obras de drenaje""</description>");

    //Definición de estilos de indicadores para estructuras
    //http://kml4earth.appspot.com/icons.html... Ruta para tipos de iconos

    //Estilo 1: flechaRoja
    ListaFichero.append("    <Style id=\"flechaRoja\">");
    ListaFichero.append("       <IconStyle>");
    ListaFichero.append("       <color>ff0000ff</color>");
    ListaFichero.append("       <scale>1.0</scale>");
    ListaFichero.append("         <Icon>");
    ListaFichero.append("
<href>http://maps.google.com/mapfiles/kml/pal4/icon28.png</href>");
    ListaFichero.append("         </Icon>");
    ListaFichero.append("       </IconStyle>");
    ListaFichero.append("    </Style>");

    //Estilo 2: globeIcon
    ListaFichero.append("    <Style id=\"globeIcon\">");
    ListaFichero.append("       <IconStyle>");
    ListaFichero.append("       <scale>1.1</scale>");
    ListaFichero.append("         <Icon>");
    ListaFichero.append("
<href>http://maps.google.com/mapfiles/kml/paddle/ylw-blank.png</href>");
```

```
    ListaFichero.append("         </Icon>");
    ListaFichero.append("       </IconStyle>");
    ListaFichero.append("    </Style>");

    //Estilo 3: normalPlacemark
    ListaFichero.append("    <Style id=\"normalPlacemark\">");
    ListaFichero.append("       <IconStyle>");
    ListaFichero.append("       <scale>1.0</scale>");
    ListaFichero.append("         <Icon>");
    ListaFichero.append("
<href>http://maps.google.com/mapfiles/kml/pushpin/grn-pushpin.png</href>");
    ListaFichero.append("         </Icon>");
    ListaFichero.append("       </IconStyle>");
    ListaFichero.append("    </Style>");

    //Estilo 4: highlightPlacemark
    ListaFichero.append("    <Style id=\"highlightPlacemark\">");
    ListaFichero.append("       <IconStyle>");
    ListaFichero.append("       <scale>1.0</scale>");
    ListaFichero.append("         <Icon>");
    ListaFichero.append("
<href>http://maps.google.com/mapfiles/kml/paddle/red-stars.png</href>");
    ListaFichero.append("         </Icon>");
    ListaFichero.append("       </IconStyle>");
    ListaFichero.append("    </Style>");

    //Estilo 5: blueglobeIcon
    ListaFichero.append("    <Style id=\"blueglobeIcon\">");
    ListaFichero.append("       <IconStyle>");
    ListaFichero.append("       <scale>1.0</scale>");
    ListaFichero.append("         <Icon>");
    ListaFichero.append("
<href>http://maps.google.com/mapfiles/kml/paddle/ltblu-blank.png</href>");
    ListaFichero.append("         </Icon>");
    ListaFichero.append("       </IconStyle>");
    ListaFichero.append("    </Style>");

}

//Escribe la parte final del KML
=================================================
void Generar_KML::agregar_fin(){
    ListaFichero.append("</Folder>");
    ListaFichero.append("</Document>");
    ListaFichero.append("</kml>");

}

 //Almacenará la lista en el fichero
=============================================
void Generar_KML::escribir_KML(QString ruta_fichero){
    QFile guardar(ruta_fichero);
    guardar.open(QIODevice::ReadWrite); //Abro para lectura y escritura
    QTextStream guardarstream(&guardar);
    for(int i=0;i<ListaFichero.count();i++){//Recorro la lista de lineas
        guardarstream<<ListaFichero.at(i)<<Qt::endl;//Guardo cada linea
    }
```

```
    guardar.close();//Cierro el fichero
}

//=========================================================================
=====
//Agrega los puntos dela lista al fichero. Los coloca en la capa indicada como
parametro.
void Generar_KML::agregar_Lista_Puntos(QList<PuntosCampo> lista_puntos)
 {
    //QMessageBox::information(this,"Mensaje de Prueba","Alerta de proceso
puntos");

    for(int i=0;i<lista_puntos.count();i++){//Recorro la lista de lineas
            ListaFichero.append("<Placemark>"); //Inicia definicion de datos
punto

            ListaFichero.append(QString("
<name>").append(lista_puntos.at(i).idelem).append("</name>"));
//Marcador para identificador de elemento
            ListaFichero.append("   <visibility>1</visibility>");
//Definición de visibilidad del punto 1->si 0->no
            ListaFichero.append("   <LookAt>");
//Marcador abierto para localizacion elemento
            ListaFichero.append(QString("
<latitude>").append(QString::number(lista_puntos.at(i).x).append("</latitude>"))
);
            ListaFichero.append(QString("
<longitude>").append(QString::number(lista_puntos.at(i).y).append("</longitude>"
)));
            ListaFichero.append("   </LookAt>");
//Marcador abierto para localizacion elemento agrega las coordenadas del punto,
con zoom al sitio

            //Clasificación según el tipo de la estructura
            if (lista_puntos.at(i).tipoelemento == "TuboPasant") {
                ListaFichero.append("   <styleUrl>#globeIcon</styleUrl>");//Tipo
de Marcador 1 para tuberías
            }
            else {
                ListaFichero.append("
<styleUrl>#blueglobeIcon</styleUrl>");//Tipo de Marcador 5 para otras
estructuras
            }
            ListaFichero.append("   <Point>");//Marcador abierto para
coordenadas del punto
            ListaFichero.append(QString("
<coordinates>").append(QString::number(lista_puntos.at(i).x)).append(",").append
(QString::number(lista_puntos.at(i).y)).append("</coordinates>"));
            //Que sigue el indicador de entidad
            ListaFichero.append("   </Point>");//Cierre de marcador para
coordenadas del punto

            ListaFichero.append("   <description>");//Inicio etiqueta
observacion
            ListaFichero.append("    <![CDATA[<h3>Inventario Infraestructura TFM
USAL</h3><br>");
            // Ruta de imagen a Visualizar
```

```
            ListaFichero.append(QString("    <img style=\"max-
width:300px;\"").append("
src=\"").append(QString(lista_puntos.at(i).imagen1)).append("\">"));
            // Conformación de una tabla para desplegar algunos atributos
            ListaFichero.append("     <br><table border=\"2\" padding=\"1\">");
            ListaFichero.append(QString("     <tr><td>Tipo de
estructura</td><td>").append(QString(lista_puntos.at(i).tipoelemento)).append("<
/td></tr>"));
            ListaFichero.append(QString("     <tr><td>Estado del
conducto</td><td>").append(QString(lista_puntos.at(i).estconducto)).append("</td
></tr>"));
            ListaFichero.append(QString("
<tr><td>Responsable</td><td>").append(QString(lista_puntos.at(i).responsable)).a
ppend("</td></tr>"));
            ListaFichero.append(QString("     <tr><td>Fecha de
Registro</td><td>").append(lista_puntos.at(i).fechareg).append("</td></tr>"));
            ListaFichero.append("     </table>]]>");
            ListaFichero.append("   </description>");
//Fin etiqueta observacion
        ListaFichero.append("</Placemark>");
//Fin definicion de datos punto
    }
}
```

Sources "generar_kml2.cpp"

```
#include "generar_kml2.h"
#include <QTextStream>
#include <QMessageBox>
#include <QFile>
#include "puntoscampo.h"
#include <QString>

//Generador KML por ESTADO DE LA ESTRUCTURA ==================================
//============================================================================
Generar_KML2::Generar_KML2()
{
// Constructor por defecto
}

//Escribe en el KML los renglones del comienzo del KML
============================

void Generar_KML2::agregar_comienzo(){
    ListaFichero.append(QString("<?xml version=\"1.0\" encoding=\"UTF-8\"?>"));
    //ListaFichero.append(QString("<?xml
version=").append(QString(""1.0"").append(" encoding=").append("'UTF-
8'").append("?>")));
    ListaFichero.append("<kml xmlns=\"http://www.opengis.net/kml/2.2\">");
    ListaFichero.append(QString("<Document>"));
    ListaFichero.append("<Folder>");
    ListaFichero.append("    <name>""Inventario Infraestructura""</name>");
    ListaFichero.append("    <visibility>0</visibility>");
    ListaFichero.append("    <description>""Obras de drenaje""</description>");

    //Definición de estilos de indicadores para estructuras
```

```
    //http://kml4earth.appspot.com/icons.html... Ruta para tipos de iconos
    //Estilo 1: flechaRoja
    ListaFichero.append("    <Style id=\"flechaRoja\">");
    ListaFichero.append("       <IconStyle>");
    ListaFichero.append("       <color>ff0000ff</color>");
    ListaFichero.append("       <scale>1.0</scale>");
    ListaFichero.append("         <Icon>");
    ListaFichero.append("
<href>http://maps.google.com/mapfiles/kml/pal4/icon28.png</href>");
    ListaFichero.append("         </Icon>");
    ListaFichero.append("       </IconStyle>");
    ListaFichero.append("    </Style>");

    //Estilo 2: globeIcon
    ListaFichero.append("    <Style id=\"globeIcon\">");
    ListaFichero.append("       <IconStyle>");
    ListaFichero.append("       <scale>1.1</scale>");
    ListaFichero.append("         <Icon>");
    ListaFichero.append("
<href>http://maps.google.com/mapfiles/kml/paddle/ylw-blank.png</href>");
    ListaFichero.append("         </Icon>");
    ListaFichero.append("       </IconStyle>");
    ListaFichero.append("    </Style>");

    //Estilo 3: normalPlacemark
    ListaFichero.append("    <Style id=\"normalPlacemark\">");
    ListaFichero.append("       <IconStyle>");
    ListaFichero.append("       <scale>1.0</scale>");
    ListaFichero.append("         <Icon>");
    ListaFichero.append("
<href>http://maps.google.com/mapfiles/kml/pushpin/grn-pushpin.png</href>");
    ListaFichero.append("         </Icon>");
    ListaFichero.append("       </IconStyle>");
    ListaFichero.append("    </Style>");

    //Estilo 4: highlightPlacemark
    ListaFichero.append("    <Style id=\"highlightPlacemark\">");
    ListaFichero.append("       <IconStyle>");
    ListaFichero.append("       <scale>1.0</scale>");
    ListaFichero.append("         <Icon>");
    ListaFichero.append("
<href>http://maps.google.com/mapfiles/kml/paddle/red-stars.png</href>");
    ListaFichero.append("         </Icon>");
    ListaFichero.append("       </IconStyle>");
    ListaFichero.append("    </Style>");

}

//Escribe la parte final del KML
================================================
void Generar_KML2::agregar_fin(){
    ListaFichero.append("</Folder>");
    ListaFichero.append("</Document>");
    ListaFichero.append("</kml>");

}
```

```
 //Almacenará la lista en el fichero
==============================================
void Generar_KML2::escribir_KML2(QString ruta_fichero){
    QFile guardar(ruta_fichero);
    guardar.open(QIODevice::ReadWrite); //Abro para lectura y escritura
    QTextStream guardarstream(&guardar);
    for(int i=0;i<ListaFichero.count();i++){//Recorro la lista de lineas
        guardarstream<<ListaFichero.at(i)<<Qt::endl;//Guardo cada linea
    }
    guardar.close();//Cierro el fichero
}

//=============================================================================
//Agrega los puntos dela lista al fichero. Los coloca en la capa indicada como
parametro.
void Generar_KML2::agregar_Lista_Puntos(QList<PuntosCampo> lista_puntos)

 {
    QString estado_conducto;
    //QMessageBox::information(this,"Mensaje de Prueba","Alerta de proceso
puntos");

    for(int i=0;i<lista_puntos.count();i++){//Recorro la lista de lineas
            ListaFichero.append("<Placemark>"); //Inicia definicion de datos
punto

            ListaFichero.append(QString("
<name>").append(lista_puntos.at(i).idelem).append("</name>"));//Marcador para
identificador de elemento
            ListaFichero.append("   <visibility>1</visibility>"); //Definición
de visibilidad del punto 1->si 0->no
            ListaFichero.append("   <LookAt>");//Marcador abierto para
localizacion elemento
            ListaFichero.append(QString("
<latitude>").append(QString::number(lista_puntos.at(i).x).append("</latitude>"))
);
            ListaFichero.append(QString("
<longitude>").append(QString::number(lista_puntos.at(i).y).append("</longitude>"
)));
            ListaFichero.append("   </LookAt>");//Marcador abierto para
localizacion elemento
            //Agrega las coordenadas del punto, con zoom al sitio

            estado_conducto = lista_puntos.at(i).estconducto;

            if (estado_conducto == "Destruido" || estado_conducto ==
"Semidestruido" || estado_conducto== "Agrietado"){
                ListaFichero.append("
<styleUrl>#flechaRoja</styleUrl>");//Marcador por estado "1" para estructuras
con falla
                }
            else if (estado_conducto == "Bueno")
                {
                ListaFichero.append("
<styleUrl>#normalPlacemark</styleUrl>");//Tipo de Marcador 2 para estructuras en
buen estado
                }
```

```
            else
                {
                ListaFichero.append("
<styleUrl>#highlightPlacemark</styleUrl>");//Tipo de Marcador 3 para que no se
identifico estado adecuadamente.
                }
            ListaFichero.append("   <Point>");//Marcador abierto para
coordenadas del punto
            ListaFichero.append(QString("
<coordinates>").append(QString::number(lista_puntos.at(i).x)).append(",").append
(QString::number(lista_puntos.at(i).y)).append("</coordinates>"));
            //Que sigue el indicador de entidad
            ListaFichero.append("   </Point>");//Cierre de marcador para
coordenadas del punto

            ListaFichero.append("   <description>");//Inicio etiqueta
observacion
            //ListaFichero.append(QString("    <![CDATA[<img style=\"max-
width:300px;\"").append("
src=\"").append(QString(lista_puntos.at(i).imagen1)).append("\">").append("]]>")
);
            ListaFichero.append("    <![CDATA[<h3>Inventario Infraestructura TFM
USAL</h3><br>");
            ListaFichero.append(QString("    <img style=\"max-
width:300px;\"").append("
src=\"").append(QString(lista_puntos.at(i).imagen1)).append("\">"));
            ListaFichero.append("     <br><table border=\"2\" padding=\"1\">");
            ListaFichero.append(QString("     <tr><td>Estado del
conducto</td><td>").append(QString(lista_puntos.at(i).estconducto)).append("</td
></tr>"));
            ListaFichero.append(QString("     <tr><td>Observación de
estado</td><td>").append(QString(lista_puntos.at(i).observacion)).append("</td><
/tr>"));
            ListaFichero.append(QString("
<tr><td>Responsable</td><td>").append(QString(lista_puntos.at(i).responsable)).a
ppend("</td></tr>"));
            ListaFichero.append(QString("     <tr><td>Fecha de
registro</td><td>").append(QString(lista_puntos.at(i).fechareg)).append("</td></
tr>"));
            ListaFichero.append("     </table>]]>");
            ListaFichero.append("   </description>");//Fin etiqueta observacion

        ListaFichero.append("</Placemark>"); //Fin definicion de datos punto
    }
}
```

Sources “acercatfm.cpp”

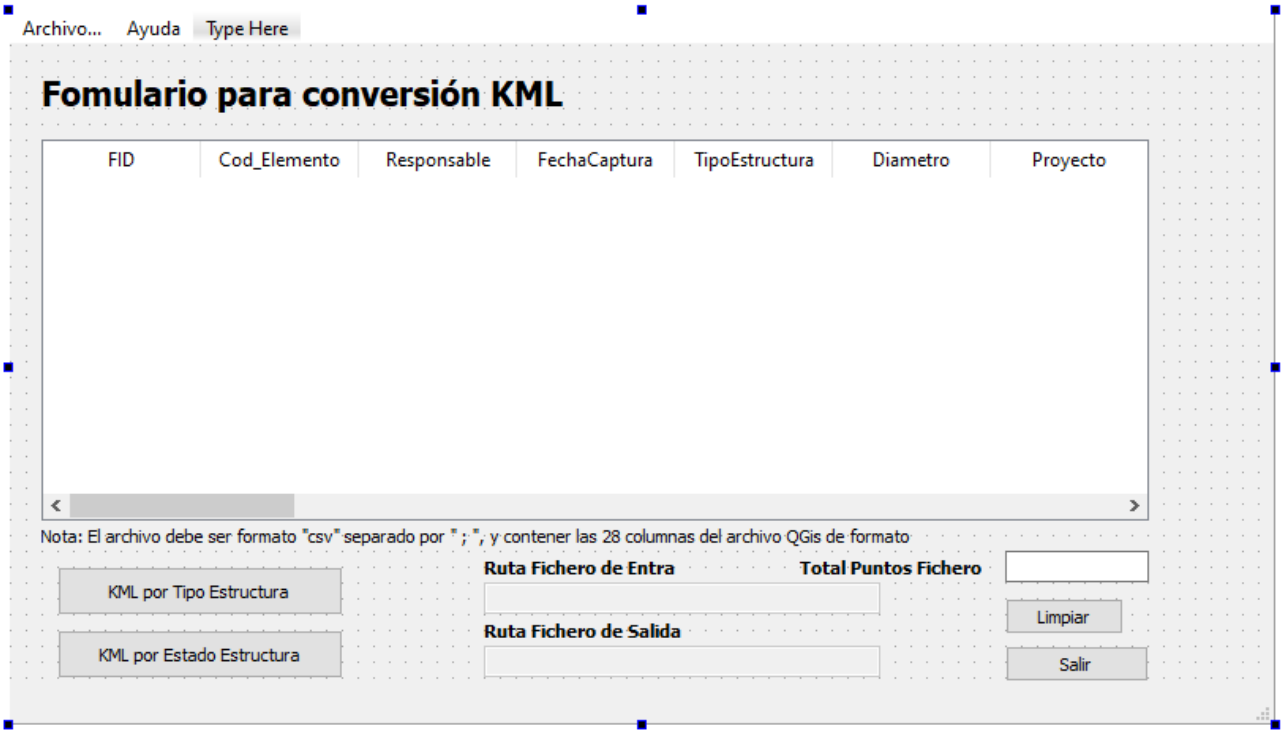
```
#include "acercatfm.h"
#include "ui_acercatfm.h"

acercaTFM::acercaTFM(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::acercaTFM)
{
    ui->setupUi(this);
```

```
}

acercaTFM::~acercaTFM()
{
    delete ui;
}

void acercaTFM::on_pushButton_clicked()
{
    this->close(); //Cierro el formulario.;
}
```

Principal del proyecto "Kml_conv.pro"

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs
deprecated before Qt 6.0.0

SOURCES += \
    acercatfm.cpp \
    generar_kml.cpp \
    generar_kml2.cpp \
    main.cpp \
    puntoscampo.cpp \
    ventanaprincipal.cpp

HEADERS += \
    acercatfm.h \
    generar_kml.h \
    generar_kml2.h \
    puntoscampo.h \
    ventanaprincipal.h

FORMS += \
    acercatfm.ui \
    ventanaprincipal.ui

RC_ICONS = icono.ico

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

Forms ventanaprincipal .ui



Esta ventana en lenguaje XML la podríamos estructurar así:

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>VentanaPrincipal</class>
 <widget class="QMainWindow" name="VentanaPrincipal">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>800</width>
    <height>450</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>VentanaPrincipal</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QLineEdit" name="txt_puntos">
    <property name="enabled">
     <bool>true</bool>
    </property>
    <property name="geometry">
     <rect>
      <x>630</x>
      <y>320</y>
      <width>91</width>
      <height>20</height>
     </rect>
    </property>
   </widget>
   <widget class="QLabel" name="label">
```

```
 <property name="geometry">
  <rect>
   <x>500</x>
   <y>320</y>
   <width>121</width>
   <height>20</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <weight>75</weight>
   <bold>true</bold>
  </font>
 </property>
 <property name="text">
  <string>Total Puntos Fichero</string>
 </property>
</widget>
<widget class="QLabel" name="label_2">
 <property name="geometry">
  <rect>
   <x>20</x>
   <y>10</y>
   <width>381</width>
   <height>41</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <pointsize>16</pointsize>
   <weight>75</weight>
   <bold>true</bold>
  </font>
 </property>
 <property name="text">
  <string>Fomulario para conversión KML</string>
 </property>
</widget>
<widget class="QTableWidget" name="tw_tabla">
 <property name="geometry">
  <rect>
   <x>20</x>
   <y>60</y>
   <width>701</width>
   <height>241</height>
  </rect>
 </property>
 <column>
  <property name="text">
   <string>FID</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>Cod_Elemento</string>
  </property>
 </column>
```

```
<column>
 <property name="text">
  <string>Responsable</string>
 </property>
</column>
<column>
 <property name="text">
  <string>FechaCaptura</string>
 </property>
</column>
<column>
 <property name="text">
  <string>TipoEstructura</string>
 </property>
</column>
<column>
 <property name="text">
  <string>Diametro</string>
 </property>
</column>
<column>
 <property name="text">
  <string>Proyecto</string>
 </property>
</column>
<column>
 <property name="text">
  <string>Altura</string>
 </property>
</column>
<column>
 <property name="text">
  <string>Longitud</string>
 </property>
</column>
<column>
 <property name="text">
  <string>Est_Conducto</string>
 </property>
</column>
<column>
 <property name="text">
  <string>Funcionamiento</string>
 </property>
</column>
<column>
 <property name="text">
  <string>Est_Cabezal</string>
 </property>
</column>
<column>
 <property name="text">
  <string>Cab_Salida</string>
 </property>
</column>
<column>
 <property name="text">
```

```
   <string>Cab_Entrada</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>Est_Guardarueda</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>GuardaR_Ent</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>GuardaR_Sal</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>Est_Aletas</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>Aleta_Ent</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>Aleta_Sal</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>Observaciones</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>Img_General</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>Img_Entrada</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>Img_Salida</string>
  </property>
 </column>
 <column>
  <property name="text">
   <string>Img_Conducto</string>
  </property>
```

```
    </column>
    <column>
     <property name="text">
      <string>Longitud</string>
     </property>
    </column>
    <column>
     <property name="text">
      <string>Latitud</string>
     </property>
    </column>
    <column>
     <property name="text">
      <string>Proyecto</string>
     </property>
    </column>
   </widget>
   <widget class="QLineEdit" name="txt_rutasalida">
    <property name="enabled">
     <bool>false</bool>
    </property>
    <property name="geometry">
     <rect>
      <x>300</x>
      <y>380</y>
      <width>251</width>
      <height>20</height>
     </rect>
    </property>
   </widget>
   <widget class="QLabel" name="label_3">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>300</y>
      <width>571</width>
      <height>21</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <pointsize>8</pointsize>
      <weight>50</weight>
      <bold>false</bold>
     </font>
    </property>
    <property name="text">
     <string>Nota: El archivo debe ser formato &quot;csv&quot; separado por
&quot; ; &quot;, y contener las 28 columnas del archivo QGis de formato</string>
    </property>
   </widget>
   <widget class="QPushButton" name="btn_salir">
    <property name="geometry">
     <rect>
      <x>630</x>
      <y>380</y>
      <width>91</width>
```

```
    <height>23</height>
   </rect>
  </property>
  <property name="text">
   <string>Salir</string>
  </property>
 </widget>
 <widget class="QLineEdit" name="txt_rutaoriginal">
  <property name="enabled">
   <bool>false</bool>
  </property>
  <property name="geometry">
   <rect>
    <x>300</x>
    <y>340</y>
    <width>251</width>
    <height>20</height>
   </rect>
  </property>
 </widget>
 <widget class="QLabel" name="label_4">
  <property name="geometry">
   <rect>
    <x>300</x>
    <y>320</y>
    <width>121</width>
    <height>20</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <weight>75</weight>
    <bold>true</bold>
   </font>
  </property>
  <property name="text">
   <string>Ruta Fichero de Entrada</string>
  </property>
 </widget>
 <widget class="QPushButton" name="btn_proceso">
  <property name="geometry">
   <rect>
    <x>30</x>
    <y>330</y>
    <width>181</width>
    <height>31</height>
   </rect>
  </property>
  <property name="text">
   <string>KML por Tipo Estructura</string>
  </property>
 </widget>
 <widget class="QPushButton" name="btn_proceso_2">
  <property name="geometry">
   <rect>
    <x>30</x>
    <y>370</y>
```

```
      <width>181</width>
      <height>31</height>
     </rect>
    </property>
    <property name="text">
     <string>KML por Estado Estructura</string>
    </property>
   </widget>
   <widget class="QLabel" name="label_5">
    <property name="geometry">
     <rect>
      <x>300</x>
      <y>360</y>
      <width>141</width>
      <height>20</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <weight>75</weight>
      <bold>true</bold>
     </font>
    </property>
    <property name="text">
     <string>Ruta Fichero de Salida</string>
    </property>
   </widget>
   <widget class="QPushButton" name="pushButton">
    <property name="geometry">
     <rect>
      <x>630</x>
      <y>350</y>
      <width>75</width>
      <height>23</height>
     </rect>
    </property>
    <property name="text">
     <string>Limpiar</string>
    </property>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>800</width>
     <height>21</height>
    </rect>
   </property>
   <widget class="QMenu" name="menuAbrir">
    <property name="title">
     <string>Archivo...</string>
    </property>
    <addaction name="actionAbrir"/>
    <addaction name="actionLimpiar_Archivo"/>
    <addaction name="actionSalir"/>
```

```
   </widget>
   <widget class="QMenu" name="menuAyuda">
    <property name="title">
     <string>Ayuda</string>
    </property>
    <addaction name="actionAcerca_de"/>
   </widget>
   <addaction name="menuAbrir"/>
   <addaction name="menuAyuda"/>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
  <action name="actionAbrir">
   <property name="text">
    <string>Abrir...</string>
   </property>
  </action>
  <action name="actionSalir">
   <property name="text">
    <string>Salir</string>
   </property>
  </action>
  <action name="actionAcerca_de">
   <property name="text">
    <string>Acerca de...</string>
   </property>
  </action>
  <action name="actionLimpiar_Archivo">
   <property name="text">
    <string>Limpiar Archivo</string>
   </property>
  </action>
 </widget>
 <resources/>
 <connections/>
</ui>
```