CartoDruid

Manual de referencia

Instituto Tecnológico Agrario de
Castilla y León
www.itacyl.es

v0.62.x / Diciembre 2023

José Carlos Peñas López,
Javier Ramos Valle, Gustavo Río Briones

www.cartodruid.es

ÍNDICE

1. Introducción

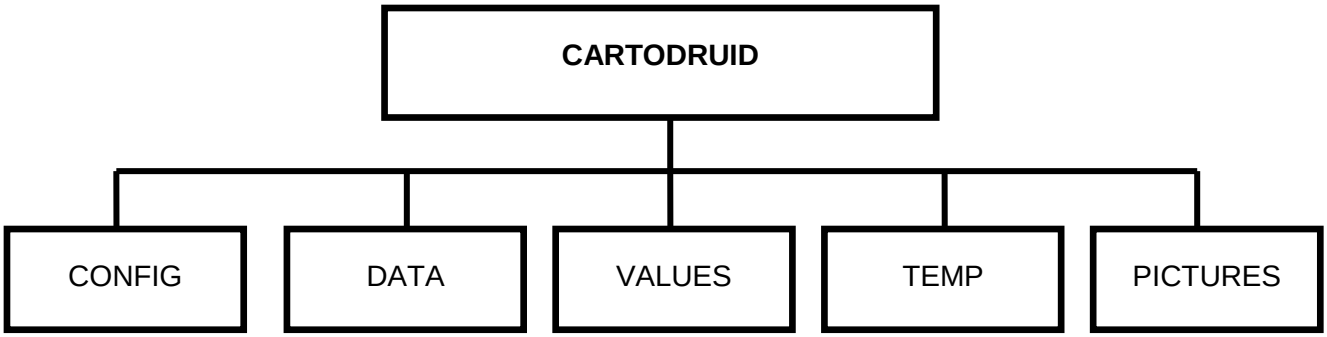
CartoDruid es una aplicación desarrollada en el Instituto Tecnológico Agrario de Castilla y León (ITACyL) pensada como herramienta de apoyo para el trabajo en campo, que busca resolver el problema de la edición de información georeferenciada sin conexión de datos en el dispositivo.

En muchas zonas de campo la cobertura de redes móviles es inexistente o insuficiente para trabajar de forma eficaz, CartoDruid da solución a este problema permitiendo visualizar capas vectoriales y raster descargadas en el propio dispositivo y crear geometrías dibujándolas directamente en la pantalla o utilizando el GPS.

CartoDruid no requiere de conocimientos GIS previos para su manejo y es fácil de configurar y utilizar, lo que abre la puerta a que cualquier persona pueda usarla para manejar información en campo y después exportar los datos grabados para utilizarlos en otras aplicaciones.

2. Estructura de ficheros de la aplicación.

Tras la instalación de CartoDruid en un dispositivo, la herramienta configura la siguiente estructura de directorios:

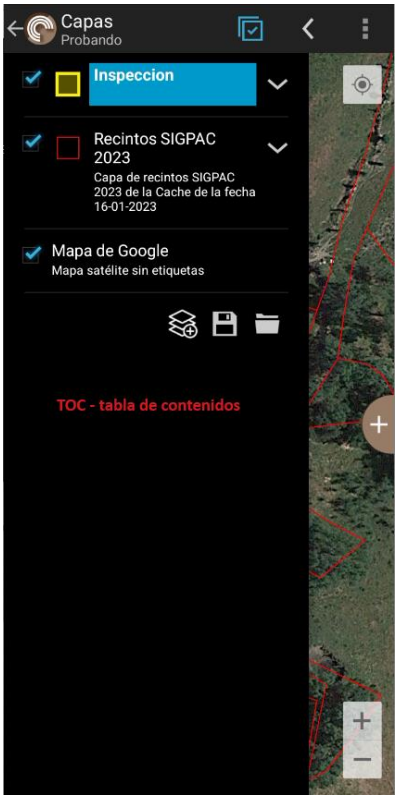


- `config`: contiene los ficheros de configuración de los proyectos, tanto el básico que viene de base con CartoDruid, como de los proyectos creados por usuarios. En el directorio se encuentra:
  - `crtdrdLayers.<id_proyecto>.xml`: ficheros de configuración de proyectos. Por defecto con la instalación de CartoDruid se incluye un fichero `crtdrdLayer.xml` sin contenido para trabajar en el proyecto básico.
  - `crtdrdSymbologies.<id_proyecto>.xml`: fichero de configuración de simbologías personalizadas para el proyecto.
  - `crtdrdStockSymbologies.xml`: fichero de configuración de las simbologías básicas incluidas de base en la instalación.
  - `sigpac.properties`: configuración de las tablas que se utilizarán para la búsqueda de recintos SIGPAC.
- `data`: directorio por defecto para almacenar las las bases de datos sqlite que se crean desde la herramienta.
- `values`: en este directorio se encuentran los archivos para almacenar valores constantes que utilizaremos en la aplicación. (Ej: sistemas de explotación).
- `temp`: directorio con archivos temporales de la aplicación.
- `pictures`: directorio donde se almacenarán las fotos tomadas desde CartoDruid (asociadas a entidades geográficas).

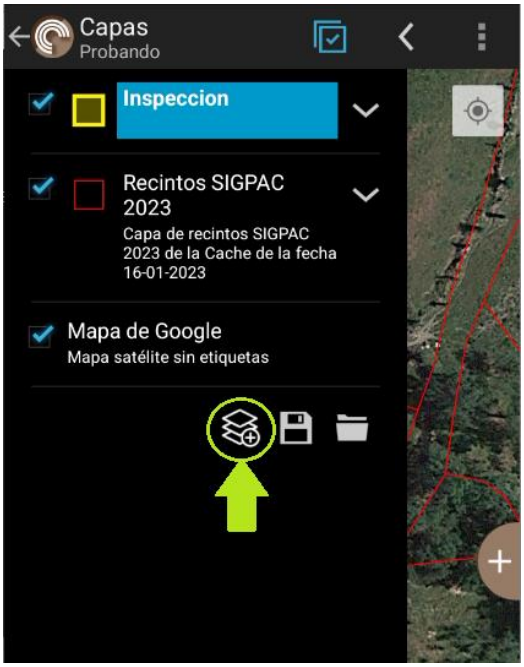
3. Creación de capas desde CartoDruid

En esta sección se indica cómo crear una capa vectorial desde cero con CartoDruid.

1. Para la creación de capas desde CRTDRD, lo primero que debemos hacer es abrir la TOC.

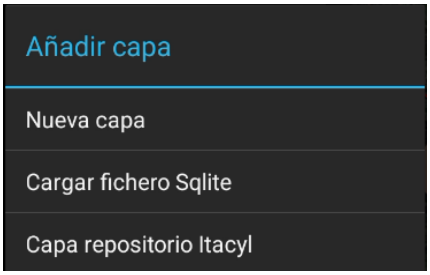


2. Pulsando el botón “Añadir capa” se abrirá el dialogo para la selección del tipo de capa.

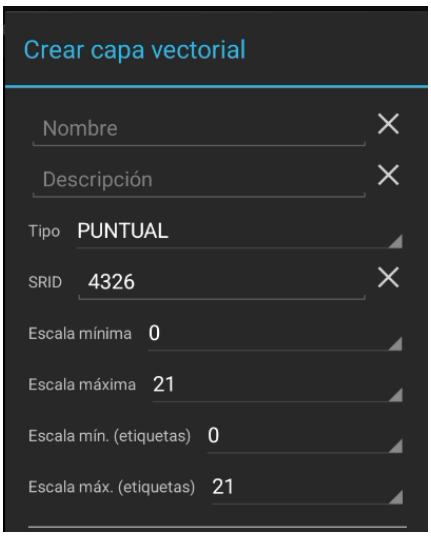
3. 

3.1 Creación de capa vectorial

b. Al seleccionar “Capa vectorial” y tendremos las siguientes opciones:

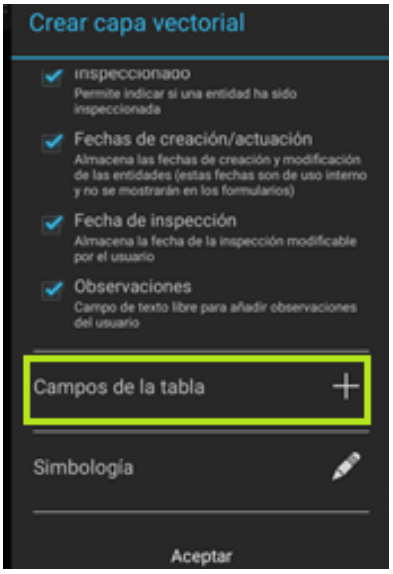


a. Seleccionamos “Nueva capa” y aparecerá el siguiente formulario:



1. En la parte superior se muestran campos básicos de identificación y visualización de la capa:
   - La identificación y el nombre serán los valores que aparecerán identificando a la capa en el menú de la tabla de contenidos.
   - Tipo: debemos seleccionar el tipo de geometría con la que vamos a trabajar: punto (ej: puntos de interés, árboles, cepas, …), línea (tracks, lindes, …) o polígonos (trabajo con superficies)
   - Escala mínima y máxima indican los niveles de zoom a los que la capa estará visible (niveles van del 0 al 21, éste último el más cercano al suelo).
   - Escala mínima y máxima de etiquetas, tiene el mismo significado que el anterior pero para las etiquetas de las geometrías.

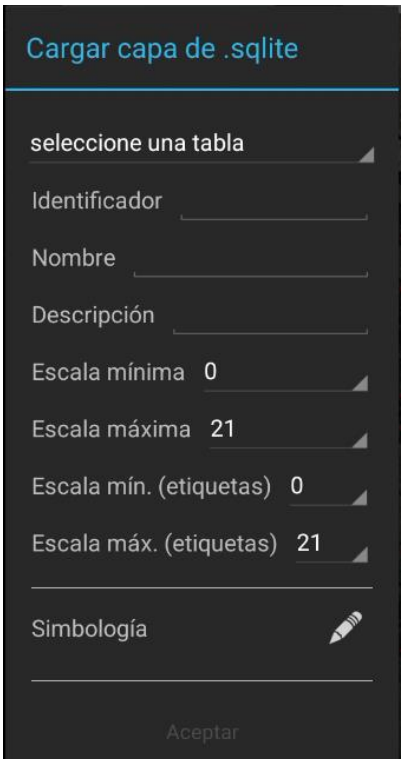
2. Inmediatamente después, aparece una selección de campos “especiales” propuestos, que serán automáticamente añadidos a la capa si se deja marcado el check.
   - Galería de fotos: abre la posibilidad de adjuntar fotos a las entidades de una capa (puntos, líneas y polígonos).
   - Referencia de recinto: añade un campo a la capa para actualizarlo con la referencia del recinto que se encuentra debajo de la entidad.
   - Inspeccionado: marca de trabajo para indicar que esta entidad ha sido inspeccionada.
   - Fechas de creación/actualización: dos campos para indicar cuándo se creó la entidad y cuándo se realizó el último cambio sobre ella.
   - Fecha de inspección: campo de fecha.
   - Observaciones: campo de texto para rellenar en campo.



3. En el siguiente apartado, dentro del mismo formulario, el usuario puede añadir los campos que necesite. Para cada campo se debe indicar el nombre y el tipo. CartoDruid soporta tipos:
   - Text: texto
   - Integer: número entero sin decimales
   - Double: número con decimales.
   - Date: fecha.
   - Boolean: campo booleano (si/no, 1/0).

4. Por último, se debe seleccionar la simbología de la capa. La simbología va a determinar la forma en que se pintará las entidades de la capa (borde, color de fondo, transparencia, etc.)
   - Por defecto CartoDruid trae una serie de simbologías predefinidas para que el usuario elija, pero cada proyecto puede definir sus propias simbologías.
   - La simbología se puede definir tanto a nivel de entidad (geometría), como de etiqueta, si se quiere que el texto identificativo que aparece en el mapa aparezca con un determinado estilo.
   - También es posible configurar simbologías condicionadas, en base a atributos de la entidad o de reglas como veremos más adelante.

b. Si seleccionamos la opción “Cargar fichero sqlite“, nos dará la opción de abrir un explorador de archivos para buscar el fichero .sqlite que contiene la capa que queremos cargar. Una vez seleccionado el fichero, aparecerá el siguiente formulario:



Es muy parecido al anterior, solo que en este caso hay que seleccionar la tabla que vamos a cargar, y además no podemos añadir ningún campo.

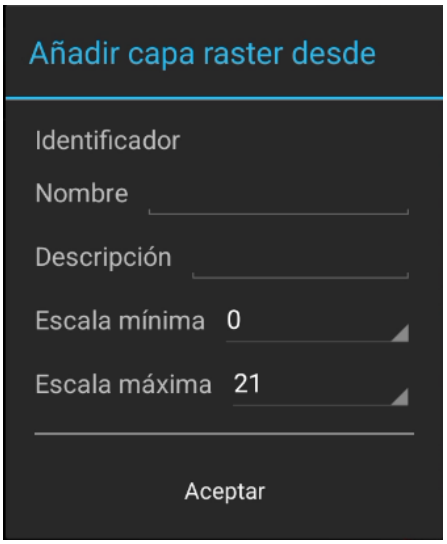
c. Si seleccionamos la opción “Capa repositorio ITACYL“, aparece un diálogo en el cual podemos seleccionar para añadir cualquiera de las capas disponibles en el repositorio del ITACyL.

Al aceptar simplemente se añade la configuración de la capa en el fichero XML de capas. Si no se tienen en el dispositivo habría que descargar los ficheros que contienen los datos del repositorio de Cartografía del ITACyL desde este enlace: https://www.cartodruid.es/cartografia

3.2 Creación de capa raster

Para la creación de “Capa Raster” tenemos las siguientes opciones:

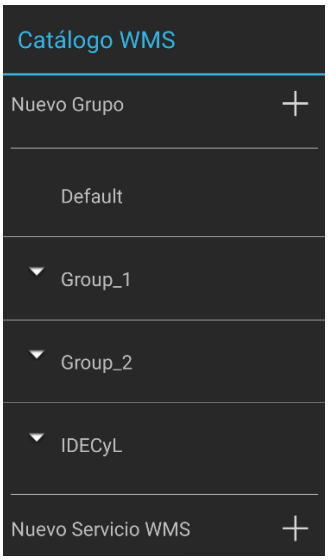
Si seleccionamos fichero MBTiles/Rasterlite se abrirá un explorador de archivos para poder seleccionar el fichero correspondiente. Hecho esto aparecerá el siguiente diálogo:



Indicamos el nombre y la descripción que van a aparecer en la TOC y las escalas de visualización mínima y máxima.

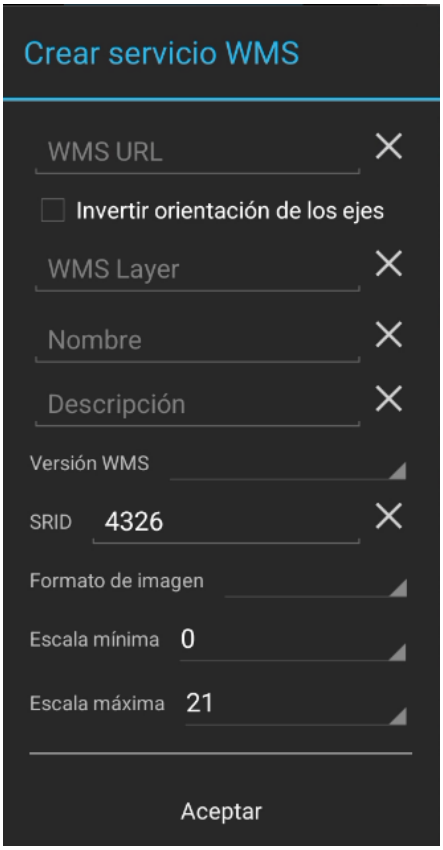
3.2.1 Capa WMS

Cuando queramos añadir una capa WMS habrá que tenerla previamente añadida al catálogo de servicios WMS. Para ello cuando pulsamos en Capa WMS se abre el siguiente diálogo:

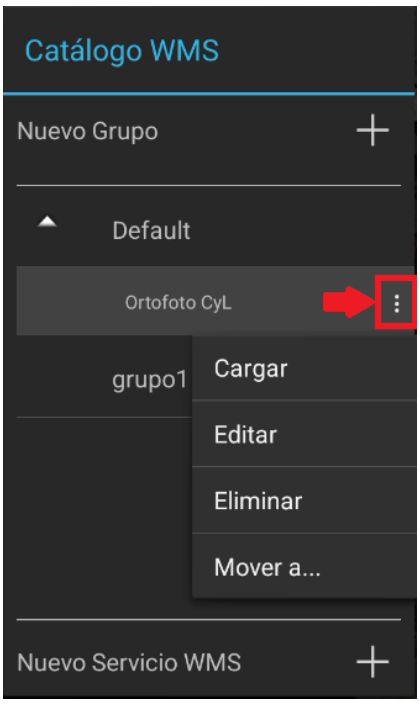


- Aquí tenemos el listado de los diferentes grupos de servicios WMS que tenemos.
- Por defecto solo tenemos el grupo “Default”, que es en el que se van a cargar todos los servicios cuando los vayamos añadiendo.
- Si queremos crear otro grupo pulsamos en “Nuevo Grupo”.
- Si queremos añadir un nuevo servicio pulsamos en “Nuevo Servicio WMS”

Al pulsar en “Nuevo Servicio WMS” nos aparece el siguiente diálogo donde indicamos las características del Servicio:



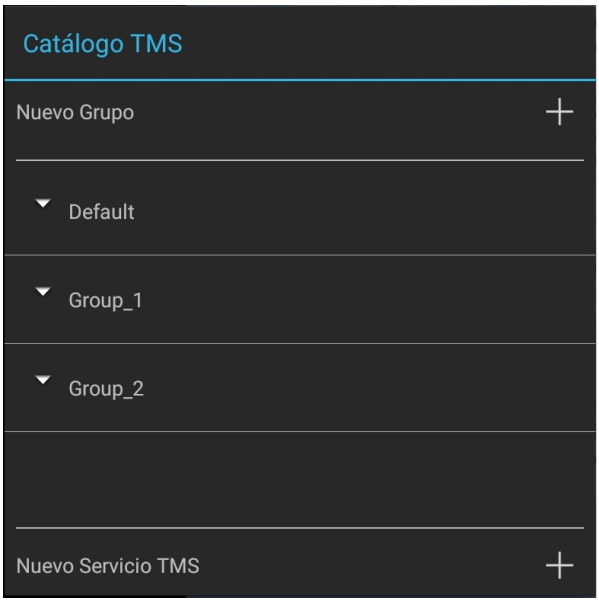
- URL del servicio.
- Nombre de la capa dentro del servicio.
- Nombre con el que vamos a mostrar el servicio en CartoDruid
- Descripción
- Versión WMS: a elegir entre 1.3.0 y 1.1.x
- Sistema de referencia del servicio. Por defecto aparece WGS84 (4326).
- Formato de la imagen: podremos elegir JPG o PNG
- Escalas de visualización mínima y máxima



Una vez añadido el servicio al catálogo (por defecto al grupo “Default”), lo podemos cargar en el mapa, editar, eliminar o moverlo a otro grupo.

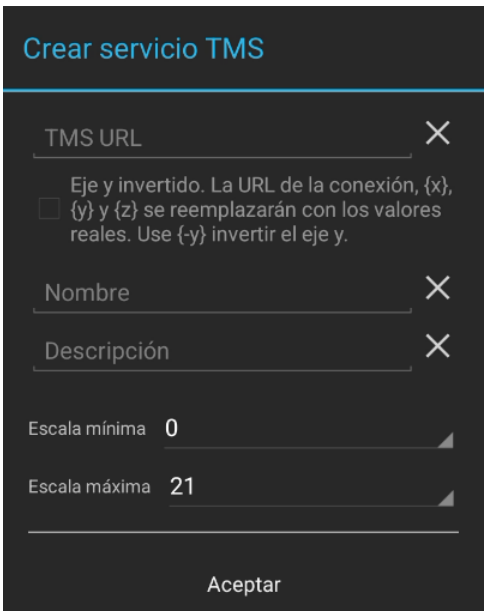
3.2.2 Capa TMS

Igualmente que los servicios WMS, cuando queramos añadir una capa TMS hay que cargarla previamente en el catálogo de servicios TMS. Para ello cuando pulsamos en Capa TMS se abre el siguiente diálogo:

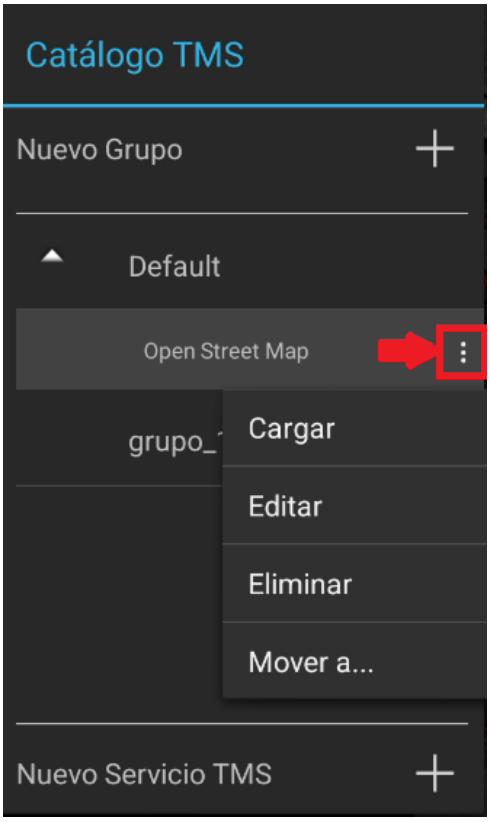


- Aquí tenemos el listado de los diferentes grupos de servicios TMS que tenemos.
- Por defecto solo tenemos el grupo “Default”, que es en el que se van a cargar todos los servicios cuando los vayamos añadiendo.
- Si queremos crear otro grupo pulsamos en “Nuevo Grupo”.
- Si queremos añadir un nuevo servicio pulsamos en “Nuevo Servicio TMS”

Al pulsar en “Nuevo Servicio TMS” nos aparece el siguiente diálogo donde indicamos las características del Servicio:



- URL del servicio.
- Nombre con el que vamos a mostrar el servicio en CartoDruid
- Escalas de visualización mínima y máxima



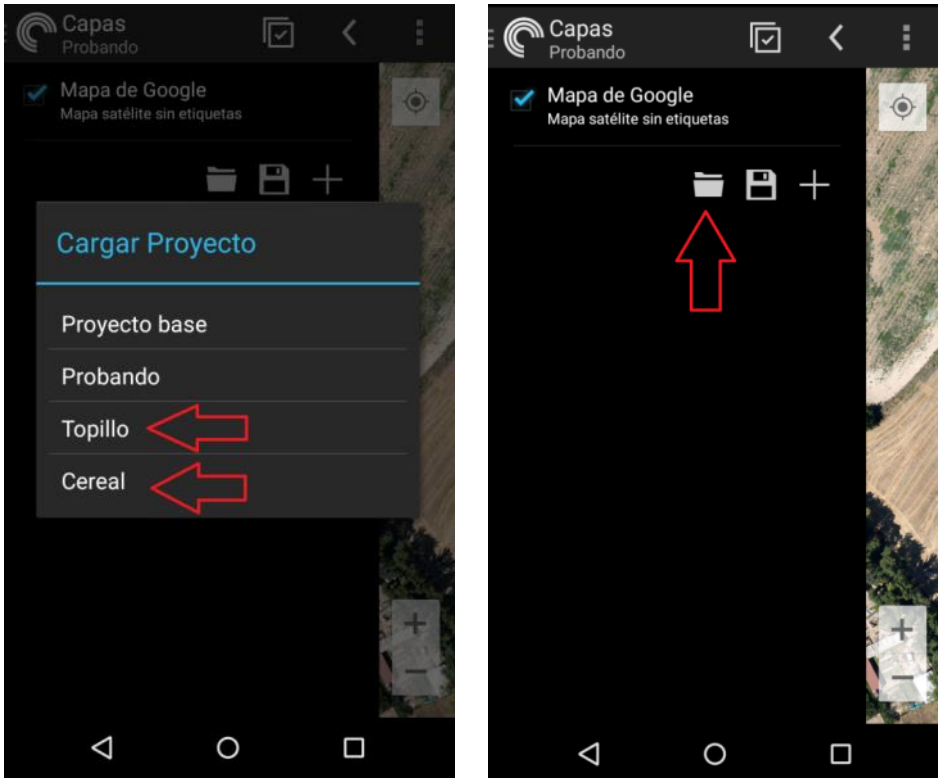
Una vez añadido el servicio al catálogo (por defecto al grupo “Default”), lo podemos cargar en el mapa, editar, eliminar o moverlo a otro grupo.

Con los datos introducidos en el formulario la aplicación creará una base de datos sqlite-spatialite en el directorio `/cartodroid/data` del dispositivo y actualizará el fichero `cartodroid/config/crtdrdLayers.xml` para incluir la definición de la nueva capa.

4. Gestión de proyectos con CRTDRD

CartoDruid es una herramienta multiproyecto que permite tener varias configuraciones de trabajo en una misma instalación. Cada proyecto tiene sus propios ficheros de configuración, todos ellos almacenados en la carpeta `/cartodroid/config`.

Desde la propia herramienta se pueden crear y cargar proyectos con los botones que aparecen en la parte inferior de la TOC.



La configuración de un proyecto CartoDruid se apoya principalmente en dos ficheros almacenados en la carpeta `/cartodroid/config`:

- `crtdrdLayers.<id_proyecto>.xml`: contiene la configuración de las capas del proyecto, es obligatorio. Por defecto con la instalación de CartoDruid se incluye un fichero `crtdrdLayer.xml` sin contenido para trabajar en el proyecto básico.
- `crtdrdSymbologies.<id_proyecto>.xml`: fichero de configuración de simbologías personalizadas para el proyecto. Es optativo, si no se incluye, se utilizará el fichero `crtdrdSymbologies.xml` que viene por defecto con la instalación para buscar los estilos y simbologías.

5. Configuración de capas en proyectos

CartoDruid permite al usuario parametrizar un conjunto limitado de opciones sobre las capas y la TOC. Para poder utilizar toda la potencia de la herramienta hay que recurrir a la configuración manual. En este apartado se describe el contenido y estructura de los ficheros de configuración y se incluyen ejemplos prácticos de manejo.

5.1 Estructura general del fichero

Por cada proyecto que tengamos configurado en CartoDruid, existirá un fichero `crtdrdLayer.<id_proyecto>.xml` en la carpeta `cartodroid/config/.` Este fichero almacena la referencia a las capas que se visualizarán en el proyecto y el comportamiento de las mismas (visualización, permisos, operaciones, etc).

Etiqueta	Descripción
`WKSLayerConfiguration`	Elemento raíz del fichero de configuración. Tendrá anidado un elemento `layers`, dentro del cual se configuran las capas con etiquetas `entry`.
`layers`	Elemento que agrupa el listado de capas del proyecto.
`entry`	Representa una entrada en la TOC. Debe tener anidados dos elementos, `string`, que será el identificador de la capa, y otro elemento con la referencia al modelo de capa concreto a utilizar: • `es.jcyl.ita.crtcyl.core.model.VectorialLayer`: para añadir una capa vectorial. • `es.jcyl.ita.crtcyl.core.model.RasterLayer`: para añadir una capa raster.

Un ejemplo de este fichero:

```
<es.jcyl.ita.crtcyl.core.config.WKSLayerConfiguration>
  <layers class="java.util.LinkedHashMap">
    <!-- configuración capa tareas -->
    <entry>
      <string>tareas</string>
      <es.jcyl.ita.crtcyl.core.model.VectorialLayer>
            ...
      </es.jcyl.ita.crtcyl.core.model.VectorialLayer>
    </entry>
    <!-- configuración capa ortofotos -->
    <entry>
      <string>ortofotos_pnoa</string>
      <es.jcyl.ita.crtcyl.core.model.RasterLayer>
            ...
      </es.jcyl.ita.crtcyl.core.model.RasterLayer>
    </entry>
   ...
  </layers>
</es.jcyl.ita.crtcyl.core.config.WKSLayerConfiguration>
```

5.2 Configuración de capa Vectorial

En la siguiente tabla se enumeran las etiquetas que se pueden utilizar dentro del elemento `es.jcyl.ita.crtcyl.core.model.VectorialLayer`.

Etiqueta	Descripción
`id`	Identificador único de la capa. Debe ser una cadena de texto sin espacios en blanco, que contenga únicamente letras y dígitos.
`nombre`	Nombre visible de la capa en la TOC.
`descripcion`	Descripción de la capa a mostrar en la TOC.
`vectorialType`	Tipos de features que almacena la capa: debe tener uno de estos valores: • 10 → para geometrías MULTIPOINT • 20 → para geometrías POLYLINE • 30 → para geometrías MULTIPOLYGON
`showOnTOC`	Determina si la capa se muestra en la lista de capas de la TOC.
Visualización y simbologías	
`orden`	Posición en la capa en el listado de capas de la TOC.
`zOrder`	Determina el orden de solapamiento de las capas a la hora del pintado. La capa con mayor zOrden aparece en el plano más cercano al usuario.

`range`	Indica los niveles de zoom en los que la capa estará visible. Para indicarlo se deben anidar dos etiquetas max y min para indicar el rango. Ej:
	`<range>`
	`<min>15</min>`
	`<max>21</max>`
	`</range>`

`labels`	Indica si las etiquetas deben renderizarse por defecto: `true|false`.
`labelRange`	Indica los rangos de visualización de las etiquetas. Se configura con elementos anidados `min/max` de forma similar a la etiqueta range.
`geometrySizeRestriction`	Indica los rangos en metros cuadrados que se permite para una geometría en esta capa. Controla el tamaño de las nuevas geometrías, de las modificaciones de tamaño y de las divisiones. Se configura con elementos anidados `min/max` de forma similar a la etiqueta range.
`labelExpression`	Expresión SQL a ejecutar para calcular la etiqueta ej: `<labelExpression>campo 1 || campo2</labelExpression>`
`showArrowHeads`	Mostrar flechas de orientación de los puntos de las entidades. Solo aplicable para entidades lineales. `true|false`.
`symbId`	Identificador de simbología a aplicar por defecto a las geometrías de una capa. La simbología debe existir en el fichero del proyecto o en el `crtdrdSymbologies.xml`.

labelSymbId	Identificador de simbología a aplicar por defecto a las etiquetas de una capa. La simbología debe existir en el fichero del proyecto o en el crtdrdSymbologies.xml.
symbologyExpression	Expresión que se utilizará para calcular la simbología de las entidades de la capa.
labelSymbologyExpression	Expresión que se utilizará para calcular la simbología de las etiquetas de las entidades.
Permisos y operaciones a nivel de capa	
allowOverlaps	Si se pueden crear geometrías con solapes dentro de la misma capa. true | false.
canChangeSymbology	Si se permite modificar la simbología de la capa. true|false.
canCopy	Si se pueden copiar los elementos de una capa en otras. true|false.
canCreate	Si se permite crear nuevos elementos en la capa. true|false.
canDeleteAll	Si se debe mostrar el botón “Eliminar todos los elementos de la capa” en la TOC. true|false.
canDieCut	Determina si se pueden troquelar las entidades de la capa. true|false.
canEditVertices	Si se debe mostrar la operación “Editar vértices” durante la edición de una geometría de la capa. true|false.
canExplode	Si se permite utilizar la operación de “Explotar geometrías” en la capa. true|false.
canPaste	Si se permiten pegar geometrías en la capa. true|false.
exportable	Si se permite exportar o compartir la capa. Si no se incluye esta etiqueta, por defecto es true.
canSanitize	Si debe aparecer el botón “Limpiar los datos de la capa” en la TOC. true|false.
canZoomVisibles	Si se debe mostrar el botón “Hacer zoom a elementos visibles” en la TOC. true|false.
deletable	Si se permite eliminar una entidad de la capa. true|false.
editable	Si se pueden editar las entidades de una capa (tanto la geometrías como los atributos). true|false.
identificable	Si se operación de identificación debe estar activa para la capa. true | false.
inspeccionable	Si el botón de “Filtrado de inspecciones” debe estar activo cuando la capa está seleccionado. true|false.
layerEditable	Determina si se puede configurar la capa (nombre, descripción, restricciones de escala). true|false.

`layerRemovable`	Si se puede eliminar la tabla de la TOC. `true|false.`
`searchable`	Si al mostrar el listado de entidades de la capa, se muestran los filtros de búsqueda. `true|false.`
`selectable`	Si las entidades de la capa pueden seleccionarse. `true|false.`
`visible`	Si la capa es visible o no en la TOC. `true|false.`
Origen de información de la capa	
`sources`	Define el origen de datos para localizar la base de datos o fichero en el que se encuentra la capa. Una capa puede incluir varios orígenes de datos. Dentro de esta etiqueta se incluirá uno de los siguientes elementos. Ver más adelante su configuración específica: • `es.jcyl.ita.crtcyl.client.dao.source.SpatiaLiteServiceDescriptor` • `es.jcyl.ita.crtcyl.client.dao.source.RepoSpatiaLiteServiceDescriptor` • `es.jcyl.ita.crtcyl.client.dao.source.SpatiaLiteQueryServiceDescriptor`
Configuración de formularios	
`alphaEditFinisher`	Identificador del formulario que a utilizar para editar los atributos de la capa.
`attributesClassName`	Clase que se utilizará para recuperar la información de la capa. Ver apartado de orígenes de datos para más información.
`editAfterCreation`	Si después de una edición o creación se debe lanzar el formulario de edición de atributos. `true|false`.

`valuesFromList`
(No recomendado)
	Relaciona los campos del formulario que mostrarán un desplegable de selección con el fichero en el que están los valores. Si hay más de uno, se separan con el carácter ";".
	`C_USO_SIGPAC=alegaciones15usoSIGPAC;b_B_SUP_COMPROBADA=siNO;`
	Para el campo `C_USO_SIGPAC` de la capa se mostrarán los valores contenidos den el fichero `/cartodroid/values/alegaciones15usoSIGPAC.properties.`
	Que tendrá el formato <código>=<valor>:
	`TA=TIERRAS ARABLES (TA)`
	`PR=PASTO ARBUSTIVO (PR)`
	`PA=PASTO CON ARBOLADO (PA)`
	Nota: desde la versión 0.56 de cartodroid, se recomienda configurar los desplegables directamente desde la configuración del formulario. Esta configuración se sigue manteniendo para que los proyectos anteriores funcionen, pero la forma recomendada de configurar los desplegables es la indicada en el apartado 7.2.4 - USO DE DESPLEGABLES EN FORMULARIOS.

`sqlIdentify`	Consulta que se va a ejecutar para la identificación.
`sqlAsListView`	Consulta que se ejecutará para mostrar la lista de entidades
`definitionQuery`	Consulta sql que se ejecutará para filtrar las entidades. El texto introducido dentro de la etiqueta se aplica como una cláusula `where` dentro de la consulta de recuperación de datos (filtro).

En el siguiente ejemplo se muestra el mínimo XML que se debe definir para configurar una capa vectorial. En este caso es una capa poligonal (`vectorialType=30`), el fichero de BD se llama `plantaciones.sql` y se encontrará en la carpeta `cartodroid/data`. La tabla que almacena la capa se llama también `plantaciones` y tienen un índice geográfico `idx_plantaciones_Geometry`.

```
<layers class="java.util.LinkedHashMap">
<entry>
  <string>capa1</string>
  <es.jcyl.ita.crtcyl.core.model.VectorialLayer>
    <id>plantaciones</id>
    <name>Plantaciones</name>
    <vectorialType>30</vectorialType>  <!-- capa poligonal -->
    <srs>25830</srs>
    <attributesClassName>es.jcyl.ita.crtdrd.data.DefaultSqlite</attributesClassName>
    <sources>
      <es.jcyl.ita.crtcyl.client.dao.source.SpatiaLiteServiceDescriptor>
       <dbURL>plantaciones.sqlite</dbURL>
       <dataTable>plantaciones</dataTable>
       <indexTable>idx_plantaciones_Geometry</indexTable>
</es.jcyl.ita.crtcyl.client.dao.source.SpatiaLiteServiceDescriptor>
</sources>
<symbId>REBECA</symbId>
    <range>
     <max>21</max>
     <min>0</min>
    </range>
  </es.jcyl.ita.crtcyl.core.model.VectorialLayer>
</entry>
</layers>
```

5.3 Configuración de capa Raster

Etiqueta	Descripción
`id`	Identificador único de la capa. Debe ser una cadena de texto sin espacios en blanco, que contenga únicamente letras y dígitos.
`nombre`	Nombre visible de la capa en la TOC.
`descripcion`	Descripción de la capa a mostrar en la TOC.
`showOnTOC`	Determina si la capa se muestra en la lista de capas de la TOC.
Visualización y simbologías	
`orden`	Posición en la capa en el listado de capas de la TOC.
`zOrder`	Determina el orden de solapamiento de las capas a la hora del pintado. La capa con mayor zOrden aparece en el plano más cercano al usuario.

`range`	Indica los niveles de zoom en los que la capa estará visible. Para indicarlo se deben anidar dos etiquetas max y min para indicar el rango. Ej:
	`<rango>`
	`<max>21</max>`

	<min>15</min> </rango>
Permisos y operaciones a nivel de capa	
layerRemovable	Si se puede eliminar la tabla de la TOC. true|false.
visible	Si la imagen raster es visible o no.
Origen de información de la capa	
sources	Define el origen de datos para localizar la base de datos o fichero en el que se encuentra la capa. Una capa puede incluir varios orígenes de datos. Dentro de esta etiqueta se incluirá uno de los siguientes elementos. Ver más adelante su configuración específica: • es.jcyl.ita.crtcyl.client.dao.source.RasterLiteServiceDescriptor • es.jcyl.ita.crtcyl.client.dao.source.RepoRasterLiteServiceDescriptor • es.jcyl.ita.crtcyl.client.dao.source.MBTilesServiceDescriptor • es.jcyl.ita.crtcyl.client.dao.source.RepoMBTilesServiceDescriptor • es.jcyl.ita.crtcyl.client.dao.source.WMSServiceDescriptor

5.4 Configuración de orígenes de datos

Para definir la fuente de datos que debe utilizar CartoDruid para leer la información de una capa se debe anidar una etiqueta <sources> dentro de la etiqueta de definición de la capa, ya sea *VectorialLayer* o *RasterLayer.*

Dentro de la etiqueta sources, vamos a poder definir los parámetros necesarios para localizar la fuente de datos que contiene la cartografía, un *descriptor*. En el siguiente apartado veremos los diferentes tipos de descriptores.

Por otro lado, en la configuración de la capa, en el atributo attributesClassName, se le indica a CartoDruid cómo debe procesar este descriptor y cómo debe tratar las geometrías de la capa cuando las lea.

5.4.1 Orígenes de datos para capas vectoriales

Para establecer un origen de datos contra una capa vectorial, debemos definir dos cosas:

- La implementación que se utilizará (etiqueta attributesClassName): determina cómo se van a leer las geometrías.
- Descriptor del origen de datos: determina cómo se localizan los ficheros de BD de referencia.

Seleccionar la implementación de consulta de geometrías

A la hora de definir el objeto que consultará las geometrías hay que decidir dos parámetros:

a) Tratamiento de las geometrías:

- Geometrías regulares: CartoDruid va a recuperar todas las geometrías que intersecten con la pantalla actual (extend o bbox). Cada geometría se recupera completa.
- Geometrías grandes: si el fichero spatialite tiene geometrías extremadamente grandes (> 10.000 vértices, estamos hablando de geometrías con una amplitud muy superior al nivel municipal, etc), con la aproximación anterior, la navegación se puede ver penalizada. En estos casos se puede utilizar una implementación que recupere solo la parte de las geometrías que se muestra en pantalla y no la geometría completa.

b) Definición de la ruta al fichero de BD:

- Único: en este caso solo queremos utilizar una única base de datos
- Múltiple: si queremos que las entidades que se muestran en una capa provengan de varios ficheros de base de datos, CartoDruid puede consultar todas las BBDD cuyo nombre de fichero empiece por una determinada cadena.

  Utilizando esto, por ejemplo, podemos tener una capa de recintos sigpac, que utilice los ficheros de Valladolid y Zamora poniendo en el descriptor:

  ```
  <dbURL>recintos.sqlite</dbURL>
  ```

  Y dejando en la misma carpeta los ficheros `recintos_va.sqlite` y `recintos_za.sqlite`.

  De esta forma CartoDruid busca en esa misma carpeta todos los ficheros que empiecen por "recintos" y tengan extensión "sqlite" y los utiliza como fuentes de datos para la capa

  Nota: es importante resaltar, que los implentadores *multi* solo se pueden utilizar para operaciones de consulta sobre entidades. Si se utiliza este tipo de implementador y se intenta modificar una geometría o sus atributos, se mostrará un mensaje de error.
- Consulta: si queremos que las entidades sean el resultado de una consulta sobre una o varias tablas de la BD.

Combinando estos dos parámetros, tenemos los cuatro posibles valores que podemos aplicar en la etiqueta <`attributesClassName`>.

	Geometrías sencillas	Geometrías complejas
Fichero único	`DefaultSqlite`	`LargeSqlite`
Múltiples ficheros	`MultiSqlite`	`MultiLargeSqlite`
Consulta	`QuerySqlite`	

Descriptor de origen de datos

A la hora de definir la localización de una capa vectorial, CartoDruid admite tres tipos de descriptor:

- **Descriptor de fichero:** en este caso identificamos la localización exacta del fichero, con una ruta absoluta en el dispositivo o relativa a la carpeta `/cartodroid/data`.

Etiqueta	Descripción
`SpatiaLiteServiceDescriptor`	
`dbURL`	Localización del fichero, con una ruta absoluta en el dispositivo o relativa a la carpeta /cartodroid/data.
`dataTable`	Nombre visible de la capa en la TOC.
`indexTable`	Identifica el índice geográfico asociada a la capa

Ejemplo:

```
<SpatiaLiteServiceDescriptor>
  <dbURL>/storage/emulated/0/cartodroid/data/plantaciones.sqlite</dbURL>
  <dataTable>plantaciones</dataTable>
```

```
    <indexTable>idx_plantaciones_Geometry</indexTable>
  </SpatiaLiteServiceDescriptor>
```

- **Descriptor por referencia a repositorio**: el sistema anterior obliga a identificar el nombre del fichero y complica la reutilización de cartografía entre diferentes proyectos y extensiones. Existe una alternativa en la que se le indica a CartoDruid las características de la capa, y éste se encarga de buscar en todo el sistema de ficheros del dispositivo una capa con un nombre que cumpla esos requisitos.

Etiqueta	Descripción
`RepoSpatiaLiteServiceDescriptor`	
`resourceid`	Localización del fichero, con una ruta absoluta en el dispositivo o relativa a la carpeta `/cartodroid/data`.
`srid`	Sistema de referencia a utilizar en la capa.
`version`	Identificador de la versión del producto cartográfico.
`dataTable`	Nombre visible de la capa en la TOC.
`indexTable`	Identifica el índice geográfico asociada a la capa

Con estos atributos, CartoDruid buscará en todo el dispositivo un fichero con el nombre:

```
<resourceid>_<srid>_<version>.sqlite
```

Por ejemplo para esta definición de capa:

```
< RepoSpatiaLiteServiceDescriptor>
   <resourceid>recintos</resourceid>
   <srid>25830</srid>
   <version>2016</version>
   <dataTable>RECINTOS</dataTable>
   <indexTable>idx_RECINTOS_Geometry</indexTable>
 </RepoSpatiaLiteServiceDescriptor>
```

CartoDruid intentará localizar un fichero de bd con el siguiente nombre:

```
recintos_25830_2016.sqlite
```

Si estamos utilizando un implementador multi (atributo `attributesClassName`), además CartoDruid encontrará todas las BDs que empiecen por “recintos_25830_2016” y tengan extensión .sqlite.

- **Descriptor por consulta**: Si queremos que los datos sean el resultado de una consulta SQL sobre una o varias tablas

Etiqueta	Descripción
SpatiaLiteQueryServiceDescriptor	
fields	Campos que va a devolver la consulta
from	- Tabla o tablas intervienen. - La tabla que contenga las geometrías tendrá asociado el alias “g”. - También se pueden añadir subconsultas SQL anidadas
where (opcional)	Condición de filtrado de la consulta
groupBy (opcional)	Identificador de la versión del producto cartográfico.
orderBy (opcional)	
primaryKey	Campo que va a actuar como identificador en el resultado de la consulta. Debe estar dentro de la lista de campos de <fields>
indexTable	Índice espacial de la tabla con contiene las geometrías
dbURL	Nombre del fichero sqlite que contiene los datos

Con estos atributos, para obtener las entidades CartoDruid generará una consulta como esta:

```
SELECT <fields> FROM <from> WHERE <where> GROUB by <groupBy> ORDER BY <orderBy>
```

Que se ejecutará sobre la base de datos especificada en <dbURL>

Por ejemplo para esta definición de capa:

```
<SpatiaLiteQueryServiceDescriptor>
   <fields>c_parvit, group_concat("c_subparvit"), st_union(geometry) as geometry</fields>
   <from>subparcelas g</from>
   <groupBy>c_parvit</groupBy>
   <dbURL>ribera2023.sqlite</dbURL>
   <primaryKey>c_parvit</primaryKey>
   <indexTable>idx_subparcelas_Geometry</indexTable>
</SpatiaLiteQueryServiceDescriptor>
```

CartoDruid generara la siguiente consulta:

```
SELECT c_parvit, group_concat("c_subparvit"), st_union(geometry) as geometry
FROM subparcelas g
GROUP BY c_parvit;
```

5.4.2 Orígenes de datos para capas raster

En el caso de las capas raster solo es necesario indicar un descriptor de origen de datos. Cada formato o servicio raster soportado por CartoDruid tendrá su propio descriptor.

Configuración de capas rasterlite y MBTiles

De forma similar a las capas vectoriales, tendremos descriptores distintos en función de si queremos referenciar la BD directamente por el nombre del fichero o por la referencia a sus características cartográficas:

	Descriptor de fichero	Descriptor de referencia a repositorio
Rasterlite	es.jcyl.ita.crtcyl.client.dao.source.RasterLiteServiceDescriptor	es.jcyl.ita.crtcyl.client.dao.source.RepoRasterLiteServiceDescriptor
MBTiles	es.jcyl.ita.crtcyl.client.dao.source.MBTilesServiceDescriptor	es.jcyl.ita.crtcyl.client.dao.source.RepoMBTilesServiceDescriptor

Configuración de capa rasterlite con descriptor por referencia

```
<entry>
  <string>ORTOFOTO</string>
  <es.jcyl.ita.crtcyl.core.model.RasterLayer>
    <descripcion>Ortofotos PNOA 2020</descripcion>
    <id>ORTOFOTO</id>
    <name>Ortofotos 2020</name>
    <sources>
      <es.jcyl.ita.crtcyl.client.dao.source.RepoRasterLiteServiceDescriptor>
     <resourceid>ortos</resourceid>
     <rastersTable>ortofotos</rastersTable>
     <srid>4326</srid>
     <version>2020</version>
      </es.jcyl.ita.crtcyl.client.dao.source.RepoRasterLiteServiceDescriptor>
    </sources>
    <zOrder>15</zOrder>
    <range>
   <max>21</max>
   <min>16</min>
    </range>
  </es.jcyl.ita.crtcyl.core.model.RasterLayer>
</entry>
```

Configuración de capa MBTiles con descriptor por referencia

```
<entry>
  <string>MAPASIGN</string>
  <es.jcyl.ita.crtcyl.core.model.RasterLayer>
    <descripcion>Mapas Topográficos del IGN 1/1.250.000, 1/500.000,1/200.000 y
1/25.000</descripcion>
    <id>MAPASIGN</id>
    <name>Mapas IGN</name>
    <sources>
      <es.jcyl.ita.crtcyl.client.dao.source.RepoMBTilesServiceDescriptor>
        <resourceid>ignmapas</resourceid>
        <srid>4326</srid>
        <version>2016</version>
```

```
      </es.jcyl.ita.crtcyl.client.dao.source.RepoMBTilesServiceDescriptor>
    </sources>
    <zOrder>10</zOrder>
    <range>
      <max>15</max>
      <min>6</min>
    </range>
  </es.jcyl.ita.crtcyl.core.model.RasterLayer>
</entry>
```

Configuración de servicio TMS (Tile Map Service)

CartoDruid puede consumir servicios TMS online, pero también puede servir teselas desde el sistema de ficheros del dispositivo, siempre que estén almacenadas con una estructura similar a la que mantienen los servidores TMS (`base_cache/nivel_zoom/x/y`).

En ambos casos el descriptor a utilizar es el mismo, `es.jcyl.ita.crtcyl.client.dao.source.WMSServiceDescriptor`, pero en el caso de hacer referencia a un servicio online, la etiqueta baseURI será una URL que empiece por http, y en el caso de hacer referencia a una caché en el dispositivo, indicaremos la ruta a la base de la caché.

La etiqueta `intertY`, determina el orden en el que debe crecer el eje de coordenadas y a la hora de construir la url de la tesela. En la definición del estándar TMS, la ruta a una tesela se construye como:

```
${baseURI}/nivel_zoom/x/y.${imageExtension}
```

La coordenada y empieza a numerarse en la parte inferior del mapa. En cambio, en el caso de servicios de Google Maps, el eje y se empieza a numerar en la parte superior.

CartoDruid cubre los dos casos simplemente modificando el valor de la etiqueta `invertY`:

- `invertY = true`: acceso a mapa de tiles tipo Google
- `invertY = false`: acceso a mapa TMS estándar.

Para más información sobre estas diferencias: http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/

Configuración acceso TMS online

```
<es.jcyl.ita.crtcyl.client.dao.source.TMSServiceDescriptor>
        <baseURI>http://a.tile.openstreetmap.org</baseURI>
        <imageExtension>png</imageExtension>
        <invertY>false</invertY>
</es.jcyl.ita.crtcyl.client.dao.source.TMSServiceDescriptor>
```

Configuración acceso caché TMS

```
<es.jcyl.ita.crtcyl.client.dao.source.TMSServiceDescriptor>
        <baseURI>/sdcard/cartorepo/osm_cache/</baseURI>
        <imageExtension>png</imageExtension>
        <invertY>false</invertY>
</es.jcyl.ita.crtcyl.client.dao.source.TMSServiceDescriptor>
```

Configuración de servicio WMS (Web Map Service)

CartoDruid soporta una implementación mínima del acceso a servicios WMS. La versión actual no soporta consultas de tipo *GetCapabilities* para comprobar capacidades del servidor, pero se pueden configurar prácticamente todos los parámetros de una petición WMS.

Etiqueta	Descripción
`es.jcyl.ita.crtcyl.client.dao.source.WMSServiceDescriptor`	
`layerName`	Parámetro LAYERS de la petición WMS indicando la capa o capas a consultar. Si se consulta más de una capa simultáneamente, separar con comas los nombres.
`format`	Parámetro FORMAT, por ejemplo image/png.

`request`	Tipo de petición WMS que se enviará al servicio, generalmente será GetMap.
`EPSG`	Sistema de referencia a utilizar.
`transparent`	Transparencia de la tesela devuelta. `True | false`
`quality`	Parámetro QUALITY para la petición WMS.
`wmsVersion`	Versión de wms a utilizar.
`endpointList`	Configuración de la ruta al servicio online.

Un ejemplo de configuración de una capa que consulta un servicio WMS:

```
<entry>
  <string>pnoa_wms</string>
  <es.jcyl.ita.crtcyl.core.model.RasterLayer>
    <descripcion>PNOA WMS ITACyL</descripcion>
    <id>pnoa_wms</id>
    <name>pnoa_wms</name>
    <sources>
   <es.jcyl.ita.crtcyl.client.dao.source.WMSServiceDescriptor>
     <layerName>Ortofoto_CyL</layerName>
     <format>image/png</format>
     <request>GetMap</request>
     <EPSG>4326</EPSG>
     <quality>50</quality>
     <transparent>true</transparent>
     <wmsVersion>1.0.0</wmsVersion>
     <endPointList>
       <es.jcyl.ita.crtcyl.core.model.source.EndPoint>
      <URL>http://orto.wms.itacyl.es/WMS</URL>
       </es.jcyl.ita.crtcyl.core.model.source.EndPoint>
     </endPointList>
   </es.jcyl.ita.crtcyl.client.dao.source.WMSServiceDescriptor>
    </sources>
    <zOrder>14</zOrder>
    <range>
   <max>21</max>
   <min>0</min>
    </range>
  </es.jcyl.ita.crtcyl.core.model.RasterLayer>
</entry>
```

6. Configuración de simbologías en proyectos

La parametrización de cómo se visualizan las geometrías y las etiquetas de las entidades de capas vectoriales se establece en el fichero `cartodroid/config/crtdrdSymbologies.xml.`

6.1 Estructura general del fichero

A la hora de definir una simbología, independientemente del tipo de geometría, se configura por separado el estilo de la simbología en sí:

- Estilo: determina colores, trazo y transparencias de figuras geométricas.
- Simbología: permite establecer el estilo a aplicar para la visualización de la geometría y para el de las etiquetas.

Etiqueta	Descripción
`symbologyConf`	Elemento raíz del fichero de configuración. Tendrá anidado un elemento `estilos` y un elemento `simbologias`.
`estilos`	Elemento que agrupa los elementos `entry` para definir los estilos.
`simbologias`	Elemento que agrupa los elementos `entry` para definir las simbologías.
`entry`	Dentro de los elementos estilos y simbologías, se pueden anidar uno más elementos `entry`. En el caso de `estilos`, una entrada se define creando una etiqueta `estilo`, y en el caso de `simbologias`, con la etiqueta `symb`. Además cada elemento entry debe llevar asociada una etiqueta `string` con un valor único que identifica al estilo/simbología.

De forma general el fichero `crtdrdSymbologies.xml` tendrá la siguiente estructura:

```
<symbologyConf>
   <estilos>
      <entry>
         <string>estilo1</string>
         <estilo class="estiloPoligono">
…
         </estilo>
      </entry>
      <entry>
         <string>estilo2</string>
         <estilo class="estiloPoligono">
…
         </estilo>
      </entry>
   </estilos>
   <simbologias>
      <entry>
         <string>simbologia1</string>
         <symb class="poligono">
            …
         </symb>
      </entry>
```

```
    </simbologias>
</symbologyConf>
```

En los siguientes apartados se describe cómo crear una visualización a medida para cada tipo de Geometría (punto, línea, polígono).

6.2 Relación entre estilos y simbologías

Existen cuatro tipo de estilos: texto, marker, línea y polígono.

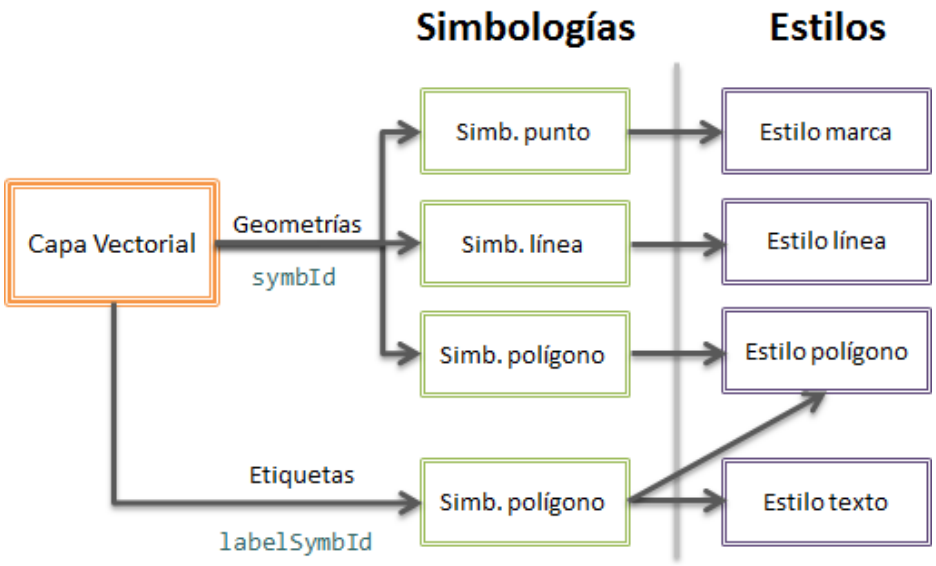
- Estilo texto (`estiloTexto`): define color y tamaño de la fuente.
- Estilo marker (`estiloMarca`): define color y opcionalmente imagen de fondo a utilizar como marca de posición.
- Estilo línea (`estiloLinea`): color, ancho de trazo de la línea, terminador, etc.
- Estilo polígono (`estiloPoligono`): además de la información asociada al estilo de línea, color de fondo.

Estos cuatro estilos se combinan para definir las simbologías de las entidades vectoriales. En el caso de CartoDruid, se puede definir simbologías diferentes para las geometrías y para las etiquetas.

Cada tipo de capa vectorial tiene asociado un tipo de simbología que se corresponde directamente con el tipo de estilo. Así, la simbología polígono, tendrá asociado un estilo de tipo `estiloPoligono`, la simbología de línea un `estiloLinea` y la simbología de punto un `estiloMarca`. Estas simbologías se referencian en la capa con la etiqueta `<symbId>` (o utilizando una expresión definida en la etiqueta `<symbologyExpression>`)

Para el caso de las etiquetas, la simbología debe ser de tipo polígono, y tendrá asociado un `estiloTexto` para establecer el formato del texto, y un `estiloPoligono`, para dar formato al cuadro que rodea al texto en sí.

En el siguiente esquema se muestra la relación entre las simbologías y los estilos.



Relación entre estilos y simbologías

6.3 Estilos y simbologías para puntos

6.3.1 Definición de estilos

Etiqueta	Descripción
`id`	Identificador del estilo. Debe coincidir con el atributo `string` definido en la etiqueta `entry`.
`transparencia`	Grado de transparencia a aplicar.
`color`	Color de la geometría. Debe llevar anidado un elemento rgb `<color><rgb>0,255,255,255</rgb></color>`
`urlImagen`	Nombre del fichero de la imagen que se mostrará para sustituir el marker por defecto de google. La imagen debe estar en el directorio `/cartodroid/symbol`.
`rotation`	Rotación a aplicar sobre la imagen (grados 0-360). Opcional.

6.3.2 Definición de simbología

Etiqueta	Descripción
`id`	Identificador de la simbologia. Debe coincidir con el atributo string definido en la etiqueta `entry`.
`idEstiloPunto`	Identificador de estilo que se utilizará para el marker.

```
<estilos>
   <entry>
      <string>symbSelectedPoint</string>
      <estilo class="estiloMarca">
         <id>symbSelectedPoint</id>
         <transparencia>1.0</transparencia>
         <color>
            <rgb>0,255,255,255</rgb>
         </color>
      </estilo>
   </entry>
</estilos>
```

Definición de estilo de Marca

```
<simbologias>
   <entry>
      <string>symbSelectedPoint</string>
      <symb class="punto">
         <id>symbSelectedPoint</id>
         <idEstiloPunto>symbSelectedPoint
         </idEstiloPunto>
      </symb>
   </entry>
</simbologias>
```

Definición de simbología de punto

6.4 Estilos y simbologías para líneas

6.4.1 Definición de estilos

Etiqueta	Descripción
`id`	Identificador del estilo. Debe coincidir con el atributo string definido en la etiqueta `entry`.
`transparencia`	Grado de transparencia a aplicar.
`colorLinea`	Color de la geometría. Debe llevar anidado un elemento rgb `<color><rgb>0,255,255,255</rgb></color>`

`trazo`	Configuración del estilo del trazo de la línea. En la actualidad se permite modificar únicamente el ancho de la línea, pero se prevé añadir otras posibilidades de visualización soportadas por el api de google-maps.
	https://developers.google.com/maps/documentation/android-api/releases
	`<trazo class=”basicStroke”>`
	`  <anchura>3</anchura>`
	`</trazo>`

6.4.2 Definición de simbología

Etiqueta	Descripción
`id`	Identificador de la simbología. Debe coincidir con el atributo `string` definido en la etiqueta `entry`.
`idEstiloLinea`	Identificador de estilo que se utilizará para la línea.

```
<estilos>
<entry>
<string>symbInspeccionLinea</string>
   <estilo class="estiloLinea">
      <id>symbInspeccionLinea</id>
      <transparencia>1.0</transparencia>
      <colorLinea>
         <rgb>0,255,255,255</rgb>
      </colorLinea>
      <trazo class=”basicStroke”>
         <anchura>3</anchura>
      </trazo>
   </estilo>
</entry>
</estilos>
```

Definición de estilo de línea

```
<simbologias>
<entry>
<string>symbInspeccionLinea</string>
   <symb class="linea">
      <id>symbInspeccionLinea</id>
      <idEstiloLinea>symbInspeccionLinea</
idEstiloLinea>
   </symb>
</entry>
</simbologias>
```

Definición de simbología de línea

6.5 Estilos y simbologías para polígonos

6.5.1 Definición de estilos

Etiqueta	Descripción
`id`	Identificador del estilo. Debe coincidir con el atributo `string` definido en la etiqueta `entry`.
`transparencia`	Grado de transparencia a aplicar.
`colorLinea`	Color de la geometría. Debe llevar anidado un elemento `rgb`: `<color><rgb>0,255,255,255</rgb></color>`
`colorRelleno`	Color de fondo de la geometría. Debe llevar anidado un elemento `rgb`: `<colorRelleno><rgb>`255,145,0,80`</rgb></colorRelleno>`

`trazo`	Configuración del estilo del trazo de la línea. En la actualidad se permite modificar únicamente el ancho de la línea, pero se prevé añadir otras posibilidades de visualización soportadas por el api de google-maps.
	https://developers.google.com/maps/documentation/android-api/releases
	`<trazo class=”basicStroke”>`
	`  <anchura>3</anchura>`
	`</trazo>`

6.5.2 Definición de simbología

Etiqueta	Descripción
`id`	Identificador de la simbologia. Debe coincidir con el atributo `string` definido en la etiqueta `entry`.
`texto`	Identificador de estilo que se utilizará para la etiqueta de la geometría. Opcional (solo se utiliza cuando la simbología se aplica a las etiquetas.
`idEstiloPoligono`	Identificador de estilo que se utilizará para la línea.

```
<estilos>
   <entry>
      <string>pend50</string>
      <estilo class="estiloPoligono">
         <id>pend50</id>
         <transparencia>1.0</transparencia>
         <colorLinea>
            <rgb>0,255,255,255</rgb>
         </colorLinea>
         <trazo class=”basicStroke”>
            <anchura>3</anchura>
         </trazo>
         <colorRelleno>
            <rgb>255,145,0,80</rgb>
         </colorRelleno>
      </estilo>
   </entry>
</estilos>
```

Definición de estilo de polígono

```
<simbologias>
   <entry>
      <string>pend50</string>
      <symb class="poligono">
         <id>pend50</id>

   <idEstiloPoligono>pend50</idEstiloPoligono>
      </symb>
   </entry>
</simbologias>
```

Definición de simbología de polígono

6.6 Estilos y simbologías para etiquetas

6.6.1 Definición de estilos

Etiqueta	Descripción
`id`	Identificador del estilo. Debe coincidir con el atributo string definido en la etiqueta `entry`.
`transparencia`	Grado de transparencia a aplicar.
`fuente`	Etiqueta que permite definir atributos tipográficos de la etiqueta. Ahora mismo solo está soportado el tamaño. Ej: `<fuente><tamano>36</tamano></fuente>`
`colorTexto`	Color de la letra a utilizar. Debe llevar anidado un elemento `rgb`: `<colorTexto><rgb>255,145,0,80</rgb></colorTexto>`

```
<entry>
   <string>idTextRojo</string>
   <estilo class="estiloTexto">
      <id>idTextRojo</id>
      <transparencia>1.0</transparencia>
      <fuente>
         <tamano>36</tamano>
      </fuente>
      <colorTexto>
         <rgb>255,0,0,255</rgb>
      </colorTexto>
      <rotacion>0.0</rotacion>
   </estilo>
</entry>
```

Definición de estilo de texto

```
<entry>
   <string>label_rojas_symb</string>
   <symb class="poligono">
      <id>Default_label</id>
      <texto>idTextRojo</texto>
      <idEstiloPoligono>label_rojas_pol
      </idEstiloPoligono>
   </symb>
</entry>
```

Definición de simbología de etiqueta

En este caso definimos una simbología de tipo polígono "`label_rojas_symb`" que referencia a un estilo de tipo texto `idTextoRojo` y a un estilo de tipo polígono `label_rojas_pol`. Para utilizar esta simbología, tendremos que referenciarla en la definición de la capa del siguiente modo: `<labelSymbId>label_rojas_symb</labelSymbId>.`

6.7 Estilos y simbologías por defecto

Con la instalación de CartoDruid se incluye un fichero con simbologías y estilos por defecto configurados en el fichero `/Cartodroid/config/crtdrdSymbologies.xml`. Esta configuración puede ser sobrescrita por el proyecto sin más que definir en el fichero de simbologías del proyecto un estilo/simbología con el mismo identificador.

Por ejemplo, la siguiente simbología es utilizada por CartoDruid para definir la presentación del polígono del sketch de un corte válido. Pegando este trozo de código en el fichero de simbologías del proyecto y modificándolo se modificaría la visualización que por defecto da CartoDruid a este tipo de sketch.

```
<entry>
   <string>symbSketchLineOK</string>
   <estilo class="estiloPoligono">
      <id>symbSketchLineOK</id>
      <transparencia>0.8</transparencia>
      <colorLinea>
         <rgb>110,230,0,255</rgb>
      </colorLinea>
      <trazo class="basicStroke">
        <anchura>2</anchura>
      </trazo>
      <colorRelleno>
         <rgb>0,0,0,0</rgb>
      </colorRelleno>
      <imagenFondo></imagenFondo>
   </estilo>
</entry>
```

6.8 Simbologías condicionales

CartoDruid permite definir reglas para calcular la simbología que se debe aplicar a una entidad, tanto a su geometria como a su *label*, utilizando las etiquetas `symbologyExpression` y `labelSymbologyExpression` en la definición de la capa en el fichero `crtdrdLayers.xml` se puede definir una expresión SQL para determinar el identificador de la simbología a utilizar (Ver apartado de casos prácticos para ejemplos concretos).

Existe un método alternativo para simbolizar en función de atributos, en la configuración de la capa se puede anidar un elemento `symbologies`, en el que definimos un elemento `symbology` por cada simbología diferente que queremos aplicar, y la expresión SQL para su cálculo.

Por ejemplo en este caso vamos a simbolizar la capa de recintos SIGPAC por el capo de c_uso_sigpac, pero en el caso del viñedo queremos hacer diferencias utilizando el campo del campo cap_auto.

```
<es.jcyl.ita.crtcyl.core.model.VectorialLayer>
        …
   <symbologies>
      <es.jcyl.ita.crtcyl.core.model.style.ConditionalSymbology>
         <id>ALICIA</id>
         <name>Uso Forestal</name>
         <condition>c_uso_sigpac = 'FO'</condition>
      </es.jcyl.ita.crtcyl.core.model.style.ConditionalSymbology>
      <es.jcyl.ita.crtcyl.core.model.style.ConditionalSymbology>
         <id>JAIME</id>
         <name>Improductivo</name>
         <condition>c_uso_sigpac = 'IM'</condition>
      </es.jcyl.ita.crtcyl.core.model.style.ConditionalSymbology>
      <es.jcyl.ita.crtcyl.core.model.style.ConditionalSymbology>
         <id>CAMILO</id>
         <name>Tierra arable</name>
         <condition>c_uso_sigpac = 'TA'</condition>
      </es.jcyl.ita.crtcyl.core.model.style.ConditionalSymbology>
      <es.jcyl.ita.crtcyl.core.model.style.ConditionalSymbology>
         <id>PABLO</id>
         <name>Viñedo con CAP &gt; 50%</name>
         <condition>c_uso_sigpac = 'VI' AND CAP_AUTO &gt; 50</condition>
      </es.jcyl.ita.crtcyl.core.model.style.ConditionalSymbology>
      <es.jcyl.ita.crtcyl.core.model.style.ConditionalSymbology>
         <id>SOFIA</id>
         <name>Viñedo con CAP &lt;= 50%</name>
         <condition>c_uso_sigpac = 'VI' AND CAP_AUTO &lt;= 50</condition>
      </es.jcyl.ita.crtcyl.core.model.style.ConditionalSymbology>
   </symbologies>
      …
</es.jcyl.ita.crtcyl.core.model.VectorialLayer>
```

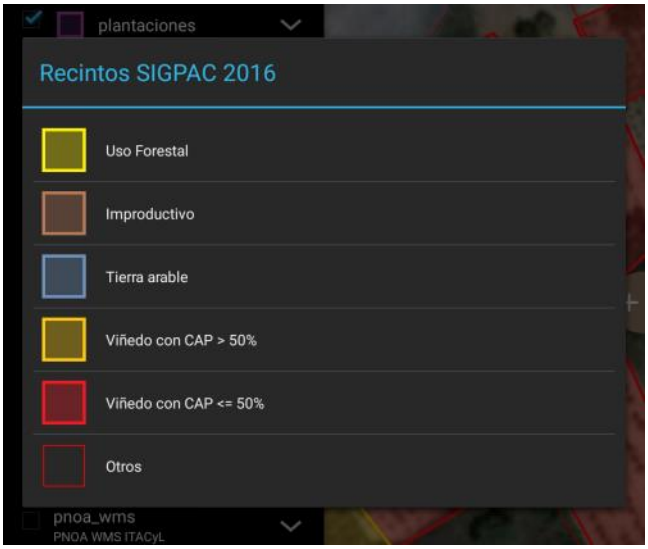
Estas reglas se aplican en el orden en el que son definidas en el XML, es decir, para cada geometría se evalúan las reglas en secuencia y se aplica la simbología de la primera regla satisfecha. En caso de no cumplir ninguno de los casos, se utiliza la simbología por defecto de la capa, la definida en el atributo `symbId`.

La ventaja respecto a utilizar la etiqueta `symbologyExpression`, es que al poder describir por separado cada regla, y darle un nombre, desde CartoDruid se puede utilizar esta información para mostrar

una leyenda del mapa. Al pulsar sobre el cuadro de la simbología en la TOC, se abre una pantalla con la descripción de los estilos.

En las siguientes pantallas se puede ver cómo se aplican estas reglas sobre las capas de recintos y la leyenda que se genera a partir de las reglas.

Capa de recintos sigpac simbolizada por uso y por coeficiente CAP



Leyenda calculada a partir de reglas de simbología

7. Configuración de la visualización de formularios

CartoDruid permite definir la visualización que se le quiere dar a un formulario, y definir reglas para determinar qué campos puede editar el usuario o qué reglas de validación aplicar.

Para poder crear un formulario de edición alfanumérica propio para una capa es necesario establecer el atributo `alphaEditFinisher` de la capa como sigue:

```
<alphaEditFinisher>userFormAlphaEditFinisher</alphaEditFinisher>
```

Estableciendo este parámetro, CartoDruid buscará dentro de la carpeta `cartodroid/values` un fichero con el nombre `identificadordecapa_AlphaEdit.xml`. Por ejemplo, para configurar un formulario para la capa “aforos”, habrá que crear un XML `cartodroid/values`/`aforos_AlphaEdit.xml`.

Nota: el fichero se almacena en un sistema de ficheros Android que es sensible a minúsculas/mayúsculas, el nombre del fichero debe empezar con exactamente el identificador de la capa, luego si la capa se ha identificado como INSPECCIONES, el fichero debe llamarse `INSPECCIONES_AlphaEdit.xml`.

7.1 Estructura general del fichero

El XML de configuración del formulario tendrá la siguiente organización:

- Formulario (`<form>`): pantalla de edición alfanumérica
  - Grupo de pestañas (`<tabs>`), y para cada pestaña anidada (`<tab>`):
    - Grupo de campos de la pestaña (`<fields>`), y dentro de ésta, por cada campo a mostrar, tendremos una etiqueta `<field>.`

Un ejemplo de la organización del fichero XML:

```
<form>
  <id>aforosForm</id>
  <name>Aforos (Edición)</name> <!-- nombre a mostrar como título de pantalla -->
  <tabs class="linked-list">
    <tab>
      <id>tab1</id>
      <name>Edición</name>  <!-- nombre de la pestaña -->
      <fields class="linked-list">
        <field>
          …
        </field>
        …
      </fields>
    </tab>
    …
  </tabs>
</form>
```

7.2 Configuración de campos del formulario

Etiqueta	Descripción
`id`	Identificador único del campo.
`name`	Nombre que se mostrará para el campo.
`type`	Tipo de campo, puede ser `TEXT, DROPDOWN, DATE, BOOLEAN, SEPARATOR, INFO, SIGN`. Nota: el tipo `COMBO` ha desaparecido en favor de `DROPDOWN`.
`inputType`	Tipo de componente del campo `TEXT`: configurable en función de http://developer.android.com/reference/android/text/InputType.html. Por ejemplo, para configurar un campo de tipo numérico, con signo y decimal, se pondría `12290` (2 + 4096 + 8192), equivalente a sumar los valores de: • `InputType.TYPE_CLASS_NUMBER (2)` • `InputType.TYPE_NUMBER_FLAG_SIGNED (4096)` • `InputType.TYPE_NUMBER_FLAG_DECIMAL (8192)`
`regexp`	Expresión regular que debe cumplir el valor introducido por el usuario para que valide correctamente. Sólo es aplicable para campos de tipo `TEXT`. Puede utilizarse para validar que el usuario introduzca valores numéricos, emails, teléfonos, etc…
`hint`	Texto de ayuda para rellenar el campo.
`extendedHint`	Texto extendido de ayuda para rellenar el campo.
`persistedField`	Nombre del campo de la BD que se utilizará para mostrar/almacenar la información.
`required`	Indica si el campo es obligatorio rellenarlo. Si este parámetro está a true, no se guardarán los datos hasta que sea rellenado. En caso de no incluir esta etiqueta, por defecto es false.
`deletable`	Indica si se permite al usuario eliminar el valor del campo.
`editable`	Indica si el campo es editable.
`defaultValue`	Valor por defecto del campo: • Para campos de tipo `TEXT` el valor por defecto será un texto • Para campos de tipo `DROPDOWN` el valor por defecto será el código del combo a establecer (por ejemplo ‘VI’ para ‘Viñedo’), no el texto visible del combo • Para campos de tipo `DATE` el valor por defecto deberá tener el formato ‘dd/MM/yyyy‘ • Para campos de tipo `BOOLEAN` el valor por defecto será ‘true’ o ‘false’ (cualquier otro valor se interpretará como ‘false’)

`choices`	Permite definir una lista de `<ítems>` de un desplegable asociado al campo. Por cada ítem se puede indicar el código y descripción del elemento con las etiquetas `key` y `value`. (ver ejemplo más adelante).
`choicesByFile`	Identifica el fichero que contiene los valores del desplegable que se asocia al campo. El fichero tiene que estar almacenado en el ruta `/cartodroid/values`.
`choicesByQuery`	Consulta SQL que se utilizará para cargar los valores del desplegable. Se debe indicar el fichero de BD que se va a utilizar con la etiqueta `<dbFile>` (ruta relativa al directorio data, o ruta absoluta), y la consulta con la etiqueta `<query>`.

Nota: Utilizando el tipo `SEPARATOR` es posible crear cabeceras intermedias para los datos, o bien vacías (una línea horizontal), o bien con un texto (rellenando el atributo `name`).

7.2.1 Definición de pestañas en el formulario

El siguiente trozo de código define tres pestañas vacías, el usuario podrá moverse de una a otra desplazando lateralmente la pantalla.

```
<form>
  <id>dopfigurasForm</id>
  <name>Inspecciones DOP/IGP</name>
  <tabs class="linked-list">
     <tab>
      <id>general</id>
      <name>1.- General</name>
    </tab>
    <tab>
      <id>obligatorios</id>
      <name>2.- Obligatorios</name>
    </tab>
    <tab>
      <id>facultativos</id>
      <name>3.- Facultativos</name>
    </tab>
  </tabs>
</form>
```

Formulario con tres pestañas

7.2.2 Uso de campos booleanos

Con el siguiente trozo de XML, configuramos un apartado con tres capos booleanos para un formulario de inspecciones. Para mostrar un control de tipo booleano el campo de la tabla debe empezar por "B_", ver el apartado de 9.2 -NOMENCLATURA DE LOS NOMBRES DE CAMPOS DE LAS TABLAS.

```
        <field>
          <id>obl_separador2</id>
```

```
          <name>Nombre registrado</name>
          <type>SEPARATOR</type>
        </field>
        <field>
          <id>b_oblig_nombre_aparece</id>
          <name>Aparece en el etiquetado</name>
          <hint>El nombre de la DOP/IGP debe aparecer en el etiquetado.</hint>
          <type>BOOLEAN</type>
          <persistedField>b_oblig_nombre_aparece</persistedField>
          <editable>true</editable>
        </field>
        <field>
          <id>b_oblig_nombre_logo_ue</id>
          <name>Nombre registrado junto a logo UE</name>
          <hint>El nombre de la DOP/IGP está en el mismo campo visual que el símbolo de la
Unión.</hint>
          <type>BOOLEAN</type>
          <persistedField>b_oblig_nombre_logo_ue</persistedField>
          <editable>true</editable>
        </field>
        <field>
          <id>b_oblig_nombre_mencion_igp</id>
          <name>Va acompañado de mención DOP/IGP</name>
          <type>BOOLEAN</type>
          <persistedField>b_oblig_nombre_mencion_igp</persistedField>
          <editable>true</editable>
        </field>
```

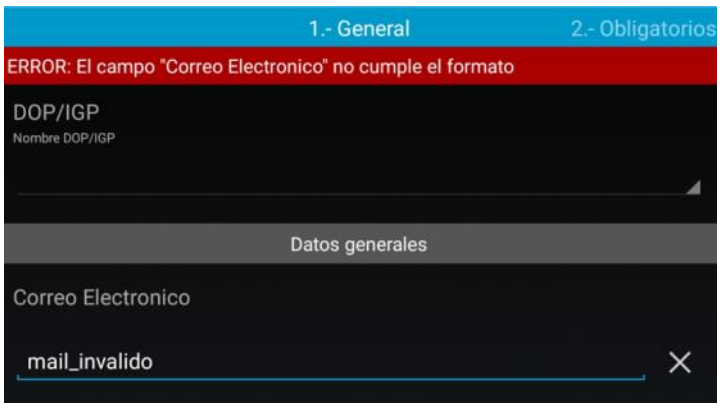
Uso de separador y componentes booleanos

7.2.3 Validación de datos con expresión regular

El siguiente código muestra cómo definir una validación de un campo de correo electrónico:

```
<field>
   <id>d_info_correo</id>
   <name>Correo Electronico</name>
   <hint></hint>
   <type>TEXT</type>
   <persistedField>d_info_correo</persistedField>
   <editable>true</editable>
   <regexp>^[_A-Za-z0-9-\+]+(\.[_A-Za-z0-9-]+)*@[A-Za-z0-9-]+(\.[A-Za-z0-9]+)*(\.[A-Za-
z]{2,})$</regexp>
</field>
```

Si el usuario no introduce un valor válido, se mostrará un mensaje de error en el formulario.



Mensaje de error tras validación de formulario

Otros ejemplos de expresiones regulares que pueden ser de utilidad:

- Introducir un valor numérico de 2 decimales permitiendo separador de "." o ",":

```
<regexp>\d+([\.,]\d{1,2})?</regexp>
```

- Validación básica de DNI (solo formato no código de control):

```
<regexp>[0-9]{7,8}[a-zA-Z]</regexp>
```

- Validar la entrada para permitir solo values numéricos 1, 2 o 3.

```
<field>
  <id>d_tipo_instalacion</id>
  <name>Tipo instalacion</name>
  <hint></hint>
  <type>TEXT</type>
  <inputType>2</inputType>
  <persistedField>d_tipo_instalacion</persistedField>
  <editable>true</editable>
  <regexp>(1|2|3)</regexp>
</field>
```

Para más información sobre expresiones regulares:

https://docs.oracle.com/javase/tutorial/essential/regex/

7.2.4 Uso de desplegables en formularios

Cartodruid permite definir desplegables en formularios de tres formas diferentes:

1) Incluyendo directamente en el fichero de configuración del formulario los valores a utilizar.
2) Utilizando los valores definidos en un fichero de propiedades.
3) Cargando los valores desde una tabla de Base de Datos.

En primer lugar, hemos definido el campo en el fichero xml de configuración del formulario, para ello indicando que es de tipo `DROPDOWN`.

```
<field>
  <id>variedad</id>
  <name>Variedad</name>
  <type>DROPDOWN</type>
  <persistedField>c_variedad</persistedField>
  <editable>true</editable>
    …
</field>
```

Para cargar el contenido del desplegable tenemos tres opciones que se configurarán con un atributo dentro de `<field></field>`.

Valores directamente en el fichero de configuración del formulario

En este caso, dentro de la configuración del campo, añadimos el listado de opciones de la siguiente forma:

```
<field>
      <id>variedad</id>
      <name>Variedad</name>
      <type>DROPDOWN</type>
      <persistedField>variedad</persistedField>
      <editable>true</editable>
      <deletable>true</deletable>
      <choices>
        <item>
          <key>CA</key>
          <value>Cabernet</value>
        </item>
        <item>
          <key>GA</key>
          <value>Garnacha</value>
        </item>
        <item>
          <key>ME</key>
          <value>Merlot</value>
        </item>
        …
      </choices>
</field>
```

Cada `<item>` se corresponde con una de las opciones del desplegable. El valor de la etiqueta `<key>` es lo que se guardará en base de datos y el de `<value>` lo que aparecerá en el desplegable.

Mediante un fichero de propiedades

En este caso, utilizamos la etiqueta `<choicesByFile>variedades.properties</choicesByFile>`, donde `variedades.properties` es el nombre del fichero que contiene las diferentes opciones que se cargarán en el desplegable.

```
<field>

      …

      <choicesByFile>variedades.properties</choicesByFile>,

</field>
```

Dicho fichero debe estar en la carpeta `cartodroid/values` y su estructura debe ser la siguiente

```
CA=Cabernet
GA=Carnacha
ME=Merlot
PA=Palomino
SA=Sauvignon Blanc
TE=Tempranillo
VE=Verdejo
VI=Viura
```

`CA` es el valor que se almacenará en base de datos y `Cabernet` el contenido que aparecerá en el desplegable

A partir de base de datos

En el tercer caso, se debe indicar en el fichero de configuración del formulario una consulta SQL y la base de datos sobre la que ejecutarla:

```
<field>

       …

       <choicesByQuery>

         <dbFile>plantaciones.sqlite</dbFile>

         <query>Select id, nombre from variedades</query>

       </choicesByQuery>
</field>
```

En este caso, la base de datos es la misma que contiene las capas del proyecto, pero podría ser cualquier otra y no es necesario que sea una capa gráfica, basta con que sea una tabla alfanumérica. La ruta al fichero de base de datos puede ser relativa, y se buscará la BD en la carpeta `/cartodroid/data` o absoluta.

En la consulta, el primer valor se utilizará como clave (lo que se almacenará en la base de datos) y el segundo es el texto que aparecerá en el desplegable.

7.2.5 Uso de campos de firma

Con el siguiente trozo de XML, configuramos un campo de firma en el que se queda registrada la firma que dibujemos en una pantalla emergente. En el bloque que aparece en el formulario podemos definir el título (name), un texto de aclaración (hint), así como exigir que la firma sea realizada obligatoriamente (required). Nos aparecerá un botón para indicar que queremos firmar y la firma en pequeño en caso de que ya se haya firmado. Para mostrar un control de tipo firma el campo de la tabla debe empezar por “S_”, ver el apartado de 9.2 -NOMENCLATURA DE LOS NOMBRES DE CAMPOS DE LAS TABLAS.

```
<field>
         <id>S_FIRMA_TITULAR</id>
         <name>Firma</name>
         <hint>Antes de iniciarse la actuación he sido informado sobre la forma en que se va a
desarrollar la misma.</hint>
        <type>SIGN</type>
        <required>true</required>
         <persistedField>S_FIRMA_ TITULAR</persistedField>
</field>
```

8. Casos prácticos de parametrización de capas vectoriales

8.1 Configuración de capa vectorial con múltiples ficheros

En este ejemplo utilizamos un implementador `MultiSqlite` para consultar varias bases de datos de recintos. Con esta configuración, CartoDruid buscará todas las bases de datos con el nombre `recintos_25830_2016_*.sqlite` y mostrará la información como una única capa.

```
<entry>
   <string>rec_2016</string>
   <es.jcyl.ita.crtcyl.core.model.VectorialLayer>
     <attributesClassName>es.jcyl.ita.crtdrd.data.MultiSqlite</attributesClassName>
     <id>recintos_2016</id>
     <name>Recintos SIGPAC 2016</name>
     <srs>25830</srs>
     <vectorialType>30</vectorialType>
     <sources>
       <es.jcyl.ita.crtcyl.client.dao.source.RepoSpatiaLiteServiceDescriptor>
         <resourceid>recintos</resourceid>
         <srid>25830</srid>
         <version>2016</version>
         <dataTable>RECINTOS</dataTable>
         <indexTable>idx_RECINTOS_Geometry</indexTable>
       </es.jcyl.ita.crtcyl.client.dao.source.RepoSpatiaLiteServiceDescriptor>
     </sources>
     <symbId>GREGORIO</symbId>
     <range>
       <max>21</max>
       <min>17</min>
     </range>
   </es.jcyl.ita.crtcyl.core.model.VectorialLayer>
</entry>
```

8.2 Configuración de capa vectorial con restricciones para la edición

En este caso definimos una capa en la que no queremos que se pueda modificar la información, para ello utilizamos las etiquetas `editable, deletable,` etc. de forma que desaparezcan del menú las opciones de modificación de datos.

```
<entry>
   <string>rec_2016</string>
   <es.jcyl.ita.crtcyl.core.model.VectorialLayer>
     <attributesClassName>es.jcyl.ita.crtdrd.data.MultiSqlite</attributesClassName>
     <id>recintos_2016</id>
     <name>Recintos SIGPAC 2016</name>
     <srs>25830</srs>
     <vectorialType>30</vectorialType>
     <sources>
       <es.jcyl.ita.crtcyl.client.dao.source.RepoSpatiaLiteServiceDescriptor>
         <resourceid>recintos</resourceid>
         <srid>25830</srid>
         <version>20160229</version>
         <dataTable>RECINTOS</dataTable>
         <indexTable>idx_RECINTOS_Geometry</indexTable>
       </es.jcyl.ita.crtcyl.client.dao.source.RepoSpatiaLiteServiceDescriptor>
     </sources>
     <symbId>GREGORIO</symbId>
         <!-- la capa se visualiza en los niveles [17,21] -->
     <range>
       <max>21</max>
```

```
          <min>17</min>
        </range>
        <editable>false</editable>
        <deletable>false</deletable>
        <canCreate>false</canCreate>
        <canPaste>false</canPaste>
        <canSanitize>false</canSanitize>
      </es.jcyl.ita.crtcyl.core.model.VectorialLayer>
  </entry>
```

8.3 Definir una etiqueta dinámica

En el texto que se muestra asociado a una geometría en el mapa (etiqueta), CartoDruid por defecto muestra el camo `pk_uid` de la tabla. El valor a mostrar se puede configurar para que se calcule en base a los campos de la capa utilizando una expresión SQL.

Por ejemplo en este caso se mostraría el valor del campo “variedad”.

```
<labelExpression>variedad</labelExpression>
```

Pero también podemos aplicar funciones para que este valor se calcule en función de diferentes. En este caso se utilizará como etiqueta de la geometría del municipio el código y su descripción separados por un valor.

```
<labelExpression>cod_municipio || '-' || desc_municipio</labelExpression>
```

8.4 Cambiar la vista de identificación de una entidad.

En este caso vamos a determinar qué campos queremos mostrar en el formulario de identificación, para ello podemos definir una expresión SQL directamente en la etiqueta `<sqlIdentity>`.

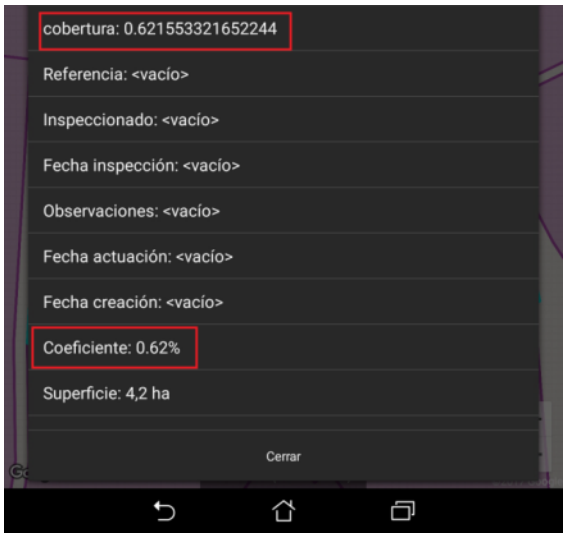
Por ejemplo, en este caso, mostramos un campo numérico formateado a 2 decimales y añadimos al final el texto “%”.

```
<sqlIdentify>*, ROUND(cobertura,2) || '%' as cobertura</sqlIdentify>
```

La etiqueta `sqlIdentity` sobrescribe la forma en que CartoDruid muestra la información de la entidad, por ello, la configuración anterior añade “*” al principio, para que se muestren todos los campos de la entidad y al final se añada el nuevo campo (ver en siguiente figura).

Si queremos mostrar solo una parte de los campos, podemos seleccionarlos directamente con la expresión SQL. En este caso solo mostramos cuatro campos al identificar:

```
<sqlIdentify>codigo, variedad,
ROUND(cobertura,2)|| '%' as cobertura
</sqlIdentify>
```



8.5 Cambiar la vista de la lista de entidades.

De igual forma que `sqlIdentify` permite modificar la información a mostrar al identificar una entidad, `sqlAsListView` permite modificar la presentación del listado de entidades.

Introduciendo la expresión del ejemplo anterior, veremos que en el listado solo se muestran las cuatro columnas seleccionadas.

```
<sqlAsListView>
codigo, variedad, ROUND(cobertura,2)|| '%' as
cobertura
</sqlAsListView>
```

De forma similar, podemos mostrar información en el listado (o en la identificación) de una tabla relacionada incluyendo una subquery como parte de la expresión.

```
<sqlAsListView>
*, (select nombre || ' ' || apellidos
from propietario p
where p.pk_uid = r.propietario_id)
as propietario
</sqlAsListView>
```

Con la expresión anterior, añadimos una consulta para calcular el campo `propietario` obteniéndolo de una tabla `Propietario` que existe en la misma BD que la tabla de la capa.

CartoDruid establece **"r"** como alias de la tabla actual, por lo que se puede utilizar este alias para hacer el join en la subconsulta.

8.6 Configurar filtros reutilizables para una capa

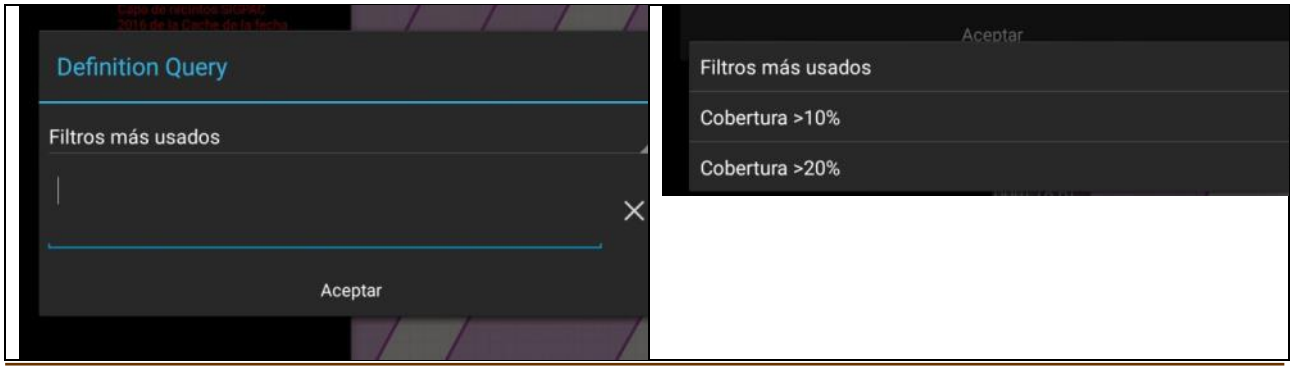
A la hora de trabajar con un proyecto es común tener una serie de filtros reutilizables para mostrar distintos grupos de información.

CartoDruid permite definir estos filtrados a nivel de capa estableciendo lo que habitualmente se conoce como se *definition queries*. Para ello, definimos el fichero `cartodroid/values/<ID_DE_CAPA>DefinitionQueries.properties`.

El fichero contendrá el nombre que aparecerá en el desplegable de filtros y la definition query que se establecerá. Por ejemplo si establecemos el fichero `/cartodroid/values/plantacionesDefinitionQueries.properties` el siguiente contenido:

```
Cobertura >10% = cobertura>0.1
Cobertura >20% = cobertura>0.2
```

Veremos que en las opciones de filtrado de la capa aparece un desplegable de "Filtros más usados", y al abrirlo aparecerán las opciones incluidas en el fichero.



En definitiva el texto introducido en el fichero de *definition queries*, se aplica directamente a la cláusula *where* de la consulta de las entidades, por lo que se pueden hacer consultas más complejas, que por ejemplo apliquen expresiones booleanas sobre los campos:

```
Condicion especial=(variedad = 'PA' and sistcond = 'V') or (variedad = 'VE' and marco1 >= 3)
```

O cruzar con otra tabla para establecer la condición en base a campos de la tabla referida, por ejemplo para mostrar solo las parcelas de los propietarios asociados:

```
Asociados=exists(select 1 from propietario p where p.pk_uid = r.propietario_id and p.asociado =
'S')
```

8.7 Configurar nombres de las imágenes tomadas con CartoDruid

Es posible configurar los nombres de las imágenes de las entidades mediante una expresión SQL sobre la entidad. Para ello, se puede usar la etiqueta `<labelPictureExpression>`, esta etiqueta define el prefijo que se asociará al archivo de la imagen.

```
<labelPictureExpression>pk_uid||'-'||c_refrec</labelPictureExpression>
```

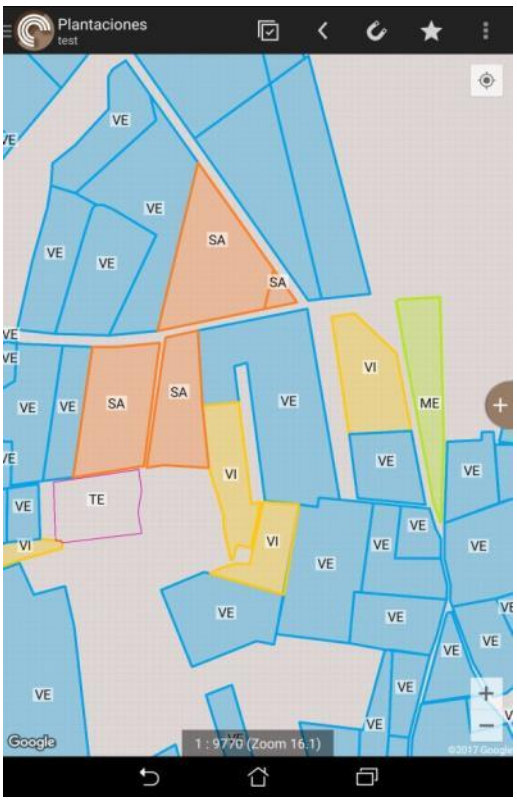
8.8 Simbolización de entidades por expresión

En este caso utilizamos el atributo `symbologyExpression` para calcular el estilo que se debe aplicar a cada entidad en función del valor variedad.

```
<entry>
  <string>insp</string>
  <es.jcyl.ita.crtcyl.core.model.VectorialLayer>
    <id>insp</id>
    <nombre>CAP</nombre>
    <descripcion>CAP 2015</descripcion>
    <vectorialType>30</vectorialType>
    <simbologia>REBECA</simbologia>
    <origenes>
      <es.jcyl.ita.crtcyl.client.dao.source.SpatiaLiteServiceDescriptor>
        <dbURL>dorueda.sqlite</dbURL>
        <dataTable>plantaciones</dataTable>
        <indexTable>idx_plantaciones_Geometry</indexTable>
      </es.jcyl.ita.crtcyl.client.dao.source.SpatiaLiteServiceDescriptor>
    </origenes>
    <rango>
      <max>21</max>
      <min>0</min>
    </rango>
    <srs>25830</srs>
    <attributesClassName>es.jcyl.ita.crtdrd.data.DefaultSqlite</attributesClassName>
<!-- la etiqueta para la entidad será el código de variedad -->
    <labelExpression>variedad</labelExpression>
    <symbologyExpression>case when variedad = 'PA' then 'ALICIA'
         when variedad = 'VI' then 'CARLA'
         when variedad = 'VE' then 'CAMILO'
         when variedad = 'TE' then 'ANA'
         when variedad = 'SA' then 'JUAN'
         when variedad = 'GA' then 'JAIME'
```

```
        when variedad = 'CA' then 'IRENE'
        when variedad = 'ME' then 'SOFIA'
        else 'REBECA' end</symbologyExpression>
   </es.jcyl.ita.crtcyl.core.model.VectorialLayer>
</entry>
```

Este sería el resultado, una capa en la que las geometrías se muestran con distinta apariencia en función del atributo “variedad”.



8.9 Cambiar los markers por defecto de las simbologías puntuales

Es posible modificar la simbología por defecto de los markers (entidades puntuales).

Para ello, se debe meter el icono deseado en el directorio symbol del directorio CartoDruid del dispositivo, y configurar la simbología puntual de la siguiente manera en el archivo `crtdrdSymbologies*.xml`.

```
<symbologyConf>
<estilos>
   <entry>
        <string>symbTopilloOK</string>
        <estilo class="estiloMarca">
           <id>symbTopilloOK</id>
           <urlImagen>topillo_ok.png</urlImagen>
           <transparencia>1.0</transparencia>
           <color>
              <rgb>0,255,255,255</rgb>
           </color>
        </estilo>
   </entry>
</estilos>
```

```
<simbologias>
   <entry>
         <string>symbTopilloOK</string>
         <symb class="punto">
            <id>symbTopilloOK</id>
            <idEstiloPunto>symbTopilloOK</idEstiloPunto>
         </symb>
   </entry>
</simbologias>
</symbologyConf>
```

8.10 Mostrar puntas de flecha en las entidades lineales

Para mostrar puntas de flecha en las entidades lineales (para marcar el sentido de grabación), se debe añadir la siguiente etiqueta en la capa:

```
<showArrowHeads>true</showArrowHeads>
```

9. Sobre las capas SQLite-Spatialite en CartoDruid

9.1 Requisitos

Requisitos que debe cumplir una capa spatialite para poder ser utilizada en CartoDruid:

- La capa debe tener correctamente definido el sistema de referencia.
- La tabla debe tener un campo de clave primaria con el nombre `pk_uid`.
- El campo que contiene la geometría en la tabla se debe llamar `Geometry` (con "G" en mayúscula).
- Para dar robustez a la edición, es conveniente utilizar tipos de datos multiparte: `MULTIPOLYGON, MULTILINESTRING, MULTIPOINT`. Esto se puede forzar con la siguiente sentencia:

  ```
  update geometry_columns set type = 'MULTIPOLYGON' where f_table_name = 'inspeccion'
  ```

  Si ya existen entidades en la capa, podemos modificar su tipo utilizando la siguiente sentencia:

  ```
  update inspeccion set Geometry = CastToMultiPolygon(Geometry);
  ```

9.2 Nomenclatura de los nombres de campos de las tablas

- Para mostrar un componente de tipo checkbox en un formulario, el nombre del campo deberá comenzar por "B_", y se definirá en la tabla como tipo entero. CartoDruid almacenará el valor del checkbox como 1 o 0. En este caso el tipo del campo de la Base de datos será INTEGER.
- Para mostrar un componente de tipo fecha, el nombre del campo deberá de comenzar por "F_", y se almacenará en la base de datos como un DOUBLE (timestamp). Si la tabla está previamente creada, para este tipo de campos también se puede utilizar el tipo INTEGER.
- Para mostrar u componente de tipo firma, el nombre del campo deberá de comenzar por "S_", y se almacenará en la base de datos como un tipo BLOB.