

## Requisitos

Se pretende agregar la funcionalidad de consultar un punto del mapa y recuperar información asociada. Para ello se seleccionará una capa del desplegable (por ejemplo pois) y pulsará sobre un punto del mapa. Tras pulsar se realizará una petición al servidor humboldt GeoPISTA y este la procesará y devolverá un resultado. El resultado se pintará en cliente en un panel flotante (como las vías y pois) y sobre éste una tabla HTML en la cual puede haber campos que su contenido sea únicamente la dirección de una página web. Si pulsa sobre esa dirección se abrirá una nueva ventana con dicha dirección.

## Implementación

### ***Fichero de configuración de getFeatureInfo servidor***

```
<?xml version="1.0"?>
<relacionesGetFeatureInfo>
  <capaInformacion nombrePublico="Puntos de Interés" idPrivado="Pois">
    <tablasFuente sqlWhereUnionTablas="public.pois.id = ..." sqlJoinTablas="">
      <tabla nombre="public.pois">
        ....
      </tablasFuente>
      <camposFuente>
        <campo nombrePublico="nombre" columnaTablaBD="pois.nombre">
          <campo... >
        </camposFuente>
        <gestionCapaInformacion escalaMinima="10" escalaMaxima="5000"
sensibilidadRadioSeleccionElementosEnPorcentaje="10">
          <campoGeometrico columnaTablaBD="pois.GEOMETRY" tipoGeometrico="POINT"/>
        </gestionCapaInformacion>
      </capaInformacion>
      <capaInformacion ...>
    </capaInformacion>
  </relacionesGetFeatureInfo>
```

En este fichero se establecerá la configuración de dónde obtendrá el servidor humboldt GeoPISTA los datos de la base de datos geopista.

- <relacionesGetFeatureInfo>: Elemento raíz que engloba la configuración.
- <capaInformacion> Establece una capa que será consultable desde el cliente. Ejemplos son puntos de interés, parcelas de urbana... El cliente podrá pedir una única capa cada vez. Tiene dos atributos:
  - nombrePublico: es el que se muestra desde el cliente en el desplegable.
  - NombrePrivado es con el que se realizarán las peticiones. Con esto se da la posibilidad ha en un futuro poder implementar el programa para varios idiomas. Existirá al menos una capaInformacion a consultar, en la instalación vendrá por defecto la de pois.
- <tablasFuente> Agrupa las tablas origen de las cuales se recuperará la información. El atributo:
  - sqlWhere: Especifica el código sql referente al where de la consulta sobre la base de datos. Sirve para deshacer los productos cartesianos al consultar varias tablas.
- <tabla> Dos atributos:
  - nombre es el esquema.nombreTabla.
  - *ColumnaID: es la columna que hace de clave primaria de dicha tabla. Si aparece más de una tabla dentro de tablasFuente entonces se realizará la unión de dichas tablas mediante la columna que hace de identificador de cada una de las tablas especificadas. Deberá haber una tablaFuente para cada capaInformacion.*
- <camposFuente> Agrupa las los campos que se mostrarán al cliente.
- <campo> El campo que se mostrará al cliente. Posee dos atributos:

- nombrePublico: nombre que se mostrará al cliente.
- ColumnaTablaBD: Es el nombre de la columna de la cual se recuperará la información. La notación es NombreTabla.nombreColumna. Deberá haber al menos un campo a consultar.
- <gestionCapaInformacion>: Especifica mediante atributos las propiedades de la capa de información. Estas propiedades comprenden el rango en el cual dicha capa es consultable por el cliente. Para ello se deberá especificar una escala mínima y una escala máxima. Adicionalmente se define el atributo sensibilidadRadioSeleccionElementosEnPorcentaje (cambiar el nombre a otro más breve y comprensible). El origen de este atributo es debido a que los puntos de interés se representa en la base de datos como un punto en el espacio, por lo que es imposible que el usuario acierte a pulsar sobre dicho punto. Por ello es necesario establecer un margen o radio que si el usuario pulsa sobre un punto consultará todo lo que esté dentro de ese alcance. Se podría especificar en metros pero no es lo mismo 50 metros en escala 1:100 que en 1:20000. Por ello algo que lo haga porcentualmente. El valor de este atributo sería entre 1-100. La idea sería si por ejemplo vale 1, es el 1% de la escala actual. Por ejemplo está consultando a escala 1:1000 buscaría sobre un radio de 10 metros, y si estuviera a 1:20000 lo haría con 200 metros.
- <campoGeometrico> Especifica cuál es el campo geométrico de la tabla que posee las coordenadas del objeto a consultar. Como atributos tiene:
  - columnaTablaBD que es el nombre de la tabla con el nombre del campo geométrico.
  - tipoGeometrico, que especifica si es un punto (POINT, como es el caso de POIS), o si fuera una línea o un polygon (parcelas).

## Clases Java

Se creará un servlet que hará de controlador de getFeatureInfo. Para llevar a cabo su labor tendrá clases de apoyo:

- Data Access Objets: Acceso a la base de datos. Debido a que el técnico puede establecer nuevas capas a consultar habrá una clase genérica que recuperará los campos establecidos en el fichero de configuración relacionados con la capa deseada. Para ello lo que hará será concatenar todos los campos como una cadena de texto de forma que el resultado sea una cadena de texto.
- Transfer Objects: Almacenará el resultado de la consulta a la base de datos y será distribuido en las diferentes capas para transformar su información a la representación requerida (se desarrollará en JSON ya que a la aplicación cliente le supondrá menos carga procesar un JSON que se convierte directamente a objetos JavaScript que no recorrer un documento XML).
- Clases que extraigan los parámetros de la petición y los interpreten: por ejemplo que calculen la escala de la petición realizada y comprobar si puede o no realizarse dicha consulta.

## Cliente

Al inicializarse el cliente comprobará la disponibilidad del servidor humboldt GeoPISTA de esta nueva característica. Construirá las peticiones getFeatureInfo para el servidor Humboldt GeoPISTA. El resultado vendrá de la forma:

```
conjuntoResultados
clave: ID
valor=43342134
clave: nombre
valor= Parcela Grande
```

...