Sensor Widgets Documentation

Versión 1.0

Oscar Fonts

11 de diciembre de 2017

Índice general

CAPÍTULO 1

Resumen del estándar SOS

Los Sensor Widgets son una herramienta de visualización de datos para servicios que cumplan con el estándar Sensor Observation Service (SOS) de la OGC.

Los widgets implementan un cliente SOS que soporta la versión 2.0 de estándard, y, en estos momentos, necesitan un endpoint en formato JSON, que resulta no ser un requisito del estándar, sino una funcionalidad opcional que proporciona la implementación de servidor SOS de 52 north en su versión 4.0.0 o superior.

Nota: De hecho, el cliente SOS podría ser extendido para soportar la codificación obligatoria KVP/XML, con lo que los Sensor Widgets serían compatibles con otras implementaciones de servidores SOS 2.0.

Un *Sensor Observation Service* ofrece datos procedentes de una colección de sensores. Así es como está organizado un servicio SOS:

1.1 Conceptos

Advertencia: La siguiente es una visión simplificada de los principales conceptos de SOS, así como una referencia rápida para los recién llegados, que probablemente estén más interesados en visualizar algunos datos que en la plena comprensión de los conceptos tras los estándares SWE, SOS y O&M de OGC. Si se va a implementar un servicio SOS, se recomienda no tomar este resumen como referencia, y acudir a las especificaciones estándar de OGC (ver referencias al final de este capítulo).

1.1.1 Offering

Los datos servidos por un servicio SOS se agrupan en diferentes *offerings*. Por ejemplo, un servicio SOS "meteo" podría tener los siguientes offerings: imágenes de satélite, datos de radar, medidas de estaciones meteorológicas, mapas de predicción, etc. Cada offering expone datos de un sensor o una red de sensores, descritos como un *procedure*.

Se pueden considerarlos distintos Offerings como secciones co cajones que clasifican los diferentes datos según su origen o naturaleza.

1.1.2 Procedure

Una procedure describe un sensor, una colección de sensores, or un proceso que produce un conjunto de observaciones. Proporciona metadatos sobre las entradas y salidas del sensor, datos de calibración y procesado, información de contacto, y la disponibilidad de datos (extensiones espacial y temporal), etc.

Normalmente viene descrito en el formato SensorML.

Se puede considerar una Procedure como una ficha de metadatos acerca del (los) sensor(es) o proceso(s) a cargo de generar los datos que ofrece el servicio.

Un Offering está relacionado con una sola Procedure, mientras que una Procedure puede ser usada en diferentes Offerings. Por ejemplo, una Procedure podría ser una "Red de Estaciones Meteorológicas", y ésta misma red de estaciones ser usada en diferentes Offerings, por ejemplo para diferentes períodos de tiempo. El Offerinf sería "Medidas de la Red de Estaciones Meteorológicas para el año 2015".

1.1.3 Feature of Interest

Cada observacion en un servicio SOS está ligada a una *Feature Of Interest* (FoI), que habitualmente determina el lugar donde el fenómeno observado tuvo lugar. Por ejemplo, para imágenes satélite, la FoI podría ser su *footprint* (polígono que determina el área fotografiada sobre la superficie de la tierra), o para una medición de temperatura, la FoI podría ser la ubicación del termómetro (punto).

Las FoI pueden considerarse como el conjunto de lugares a los que están referidos los datos.

1.1.4 Observed Property

La propiedad que se mide, tal que: Temperatura, Dirección del viento, Nubosidad, Número de vehículos... puede ser un valor numérico (una cantidad y una unidad de medida), lógico (toma los valores verdadero o falso), categórico (un valor de entre una lista: soleado, nublado, lluvioso), o descriptivo (un texto).

1.1.5 Observation

Finalmente, una *Observation* es el valor que toma una *Observed Property* en un momento (Phenomenon Time) y un lugar (Feature Of Interest) dados. Por ejemplo: "La temperatura en Barcelona el 22/09/2015 a las 11:52 es de 23 grados centígrados".

1.2 Peticiones

Todas las peticiones SOS han de indicar los siguientes parámetros:

- Service: `SOS`.
- Version: `2.0.0` (es la versión soportada por los Sensor Widgets).
- Request: el nombre de la petición, por ejemplo `GetCapabilities`.

A continuación presentamos las peticiones SOS 2.0 que usadas en los Sensor Widgets:

1.2.1 GetCapabilities

Puesto que la respuesta de un GetCapabilites es bastante prolija, la petición GetCapabilities permite especificar un parámetro `sections` para recuperar sólo parte del documento.

En concreto, la sección `contents` describe el servicio como una colección de Offerings. Cada Offering contiene los siguientes detalles:

- El nombre del Offering (por ejemplo "observaciones diezminutales"),

- El identificador del Offering,
- El identificador del Procedure ligado a éste offering,
- La colección de Observable Properties (sus identificadores),
- La extensión espacial de las observaciones que contiene (el rectángulo contenedor -bbox- de todas las Features of Interest),
- La extensión temporal de las observaciones que contiene (período de tiempo que acota todas las Observations).

Ejemplo de petición GetCapabilities en formato JSON:

```
POST http://sensors.fonts.cat/sos/json
Content-Type: application/json
Contenido:
    {
        "service":"SOS",
        "version":"2.0.0",
        "request":"GetCapabilities",
        "sections":["Contents"]
    }
```

Éste documento de Capabilities (sección contents) es el punto de entada para descubrir cómo está estructurado determinado servicio SOS, así como los datos que contiene. El documento contiene muchos identificadores de los distintos elementos (procedures, properties) pero no sus detalles, que deberán obtenerse mediante otras peticiones como DescribeSensor o GetFeatureOfInterest.

1.2.2 DescribeSensor

La petición DescribeSensor accepta como parámetro un identificador de `procedure`, y devuelve un documento SensorML que contiene metadatos acerta de el (los) sensor(es) o proceso(s) que genera(n) las observaciones.

Los contenidos más relevantes de este documento son:

- El identificador de la Procedure, un nombre corto y un nombre más largo,
- Una colección de palabras clave (útiles para servicios de búsqueda en catálogos de metadatos),
- Información de contacto,
- El período de tiempo de validez (redundante con la respuesta de Capabilities),
- El BBOX observado (redundante con la respuesta de Capabilities),
- La colección de Features of Interest (sus identificadores - nueva información que no se encuentra en el GetCapabilities),
- La colección de Offerings (sus identificadores) que se basan en esta procedure,
- Una lista de salidas (Optputs): Una colección de ObservableProperties y su descripción: IDs, nombres, tipos y unidades de medida.

Esta petición se usa para ampliar detalles que no se ofrecen a través del GetCapabilities, en especial la descripción de las Observable Properties (sus nombres y unidades de medida).

Ejemplo de petición DescribeSensor en formato JSON:

```
POST http://sensors.fonts.cat/sos/json
Content-Type: application/json
Contenido:
    {
        "service":"SOS",
        "version":"2.0.0",
        "request":"DescribeSensor",
        "procedure":"http://sensors.portdebarcelona.cat/def/weather/procedure",
```

```
        "procedureDescriptionFormat":"http://www.opengis.net/sensorML/1.0.1"
    }
```

1.2.3 GetFeatureOfInterest

La operación GetFeatureOfInterest acepta una `procedure` como parámetro, y devuelve todas las Features of Interest relacionadas con dicho procedure. De hecho, las Features of Interest están vinculadas a cada una de las Observation, pero esta operación nos devuelve una suerte de inventario de todos sus posibles valores.

Es útil para obtener los detalles de las diversas localizaciones, como sus nombres y geometrías. Así que generalmente se utiliza ésta operación para poder dibujar un mapa o un selector de Features por nombre.

Ejemplo de petición GetFeatureOfInterest en formato JSON:

```
POST http://sensors.fonts.cat/sos/json
Content-Type: application/json
Contenido:
    {
        "service":"SOS",
        "version":"2.0.0",
        "request":"GetFeatureOfInterest",
        "procedure":"http://sensors.portdebarcelona.cat/def/weather/procedure"
    }
```

1.2.4 GetDataAvailability

La petición GetDataAvailability también acepta una `procedure`, y opcionalmente una colección de `FeatureOfInterest` y/o `ObservedProperty` como parámetros.

Devuelve el rango temporal dentro del cual existen datos para cada combinación Procedure-Feature-Property. Así, dado un sensor determinado, sabemos para qué fechas vamos a disponer de datos.

Ejemplo de petición GetDataAvailability en formato JSON:

```
POST http://sensors.fonts.cat/sos/json
Content-Type: application/json
Contenido:
    {
        "service":"SOS",
        "version":"2.0.0",
        "request":"GetDataAvailability",
        "procedure":"http://sensors.portdebarcelona.cat/def/weather/procedure",
        "featureOfInterest":["http://sensors.portdebarcelona.cat/def/weather/
↪features#02"],
        "observedProperty":["http://sensors.portdebarcelona.cat/def/weather/
↪properties#31"]
    }
```

1.2.5 GetObservation

Y, finalmente, los datos de medida.

Una petición GetObservation acepta los siguientes parámetros:

- Un `offering`,
- Una colección de `FeatureOfInterest`,
- Una colección de `ObservedProperties`,

- Filtros espaciales y/o temporales.

El filtrado es especialmente interesante, puesto que pueden restringirse las búsquedas de datos a un período de tiempo o un área geográfica concreta. Los Sensor Widgets existentes hasta la fecha sólo usan el filtrado temporal para obtener, o bien el último dato disponible ("latest"), o bien una serie temporal de datos en un período dado (por ejemplo, últimas 3 horas).

Ejemplo de petición GetObservation en formato JSON:

```
POST http://sensors.fonts.cat/sos/json
Content-Type: application/json
Contenido:
    {
        "service":"SOS",
        "version":"2.0.0",
        "request":"GetObservation",
        "offering":"http://sensors.portdebarcelona.cat/def/weather/offerings#10m",
        "featureOfInterest":["http://sensors.portdebarcelona.cat/def/weather/
↪features#P3"],
        "observedProperty":["http://sensors.portdebarcelona.cat/def/weather/
↪properties#31"],
        "temporalFilter":[{
            "equals":{
                "ref":"om:resultTime",
                "value":"latest"
            }
        }]
    }
```

La respuesta es una colección de observaciones, donde cada observacion consta de:

- El identificador del Offering del que procede,
- El identificador del Procedure que la generó,
- La Feature of Interest a la que se refieren (descripción completa, con su ID, nombre y geometría),
- El identificador de la Property que se ha observado (pero no su nombre),
- Phenomenon time (cuándo ha sucedido lo que se ha medido) y result time (cuándo se ha obtenido el dato),
- Y, por fin, el resultado, que consta de un **valor** y una unidad de medida.

Así, la respuesta completa es tediosamente prolija y redundante, conteniendo centenares o miles de repeticiones sucesivas de algunos de los elementos decriptivos en el mismo documento de respuesta. Imaginemos una serie temporal de 5000 observaciones del mismo sensor. Lo único que cambia es el tiempo y el valor. El resto de contenidos (IDs, Features, etc) se repiten 5000 veces sin necesidad alguna. Esto impacta severamente la agilidad del servicio SOS.

Algunas implementaciones de SOS (en concreto, 52n SOS v.4.0.0+) ofrecen algunas estrategias que extienden el estándar para subsanar esta situación, como la ya mencionada codificación de los mensajes en JSON, y una extensión llamada `MergeObservationsIntoDataArray` que "compactan" todas las observaciones que proceden del mismo procedure, feature of interest y observed property en un `SweArrayObservation` (serie temporal de datos del mismo sensor).

Nota: Los Sensor Widgets no aprovechan aún la extensión `MergeObservationsIntoDataArray`. Es una posible mejora futura.

1.3 Referencias

Especificaciones oficiales del Open Geospatial Consortium:

- OGC® Sensor Web Enablement: Overview And High Level Architecture v. 3 (White Paper). Ref. OGC 07-165.
- OpenGIS® SWE Service Model Implementation Standard v. 2.0. Ref. OGC 09-001.
- OGC® SWE Common Data Model Encoding Standard v. 2.0.0. Ref. OGC 08-094r1.
- Sensor Observation Service v. 1.0. Ref. OGC 06-009r6.
- OGC® Sensor Observation Service Interface Standard v. 2.0. Ref. OGC 12-006.
- OpenGIS® Sensor Model Language (SensorML) Implementation Specification v. 1.0.0. Ref. OGC 07-000.
- OGC Abstract Specification - Geographic information — Observations and measurements v.2.0. Ref. OGC 10-004r3.
- Observations and Measurements - XML Implementation v.2.0. Ref. OGC 10-025r1.

CAPÍTULO 2

Cómo usar los Sensor Widgets

Cada widget tiene una colección de parámetros obligatorios y otros opcionales. Configurar un widget es básicamente definir los valores adecuados de estos parámetros para obtener el resultado deseado.

2.1 El Wizard

La manera más fácil de configurar un widget es usando el Wizard, que nos asistirá en la selección de los parámetros basándose en una lista de posibles valores. Por ejemplo, la mayoría de widgets tienen como parámetros obligatorios un "offering", uno o más "features" y una o más "properties". El wizard inspeccionará los recursos del servicio SOS y nos permitirá elegir de entre una lista de offerings, features y properties existentes.

Otros parámetros habituales son el tiempo de refresco (refresh_interval), para widgets que muestran datos en vivo que deben actualizarse periódicamente, o el rango de tiempo (time_start, time_end), para widgets que muestran una colección de mediciones a lo largo del tiempo. En este último caso, el wizard nos asistirá con un selector de rango de tiempo restringido al período temporal en el que existen datos disponibles.

Parámetros opcionales típicos son la nota a pie ("footnote"), que es un texto que se mostrará junto al widget, y la dirección a una hoja de estilos propia ("custom_css_url"), un mecanismo para adaptar el aspecto de los widgets.

Los detalles acerca de los parámetros que acepta de cada widget y su uso están descritos en el próximo capítulo.

Una vez ajustados los valores en el formulario del wizard, al clicar en el botón "Crear Widget", se visualizará el resultado en el recuadro "Vista del Widget", así como tres maneras de utilizarlo en el recuadro "Para llevar": Como una página HTML ("Enlazar"), como un componente utilizable en otra página ("Incrustar") y como un bloque de código para integrar el widget dentro de un aplicación Javascript de mayor alcance ("Código").

2.2 Para llevar: Enlace e incrustación

Si hacemos clic en el enlace bajo "Enlazar", obtendremos una URL bastante larga. Esta URL abre una página con el widget que hemos configurado. Si lo único que nos interesa es el widget tal cual, no hace falta saber más.

Pero para quien esté interesado en comprender cómo funcionan estos enlaces (por ejemplo, para modificarlos manualmente, sin tener que pasar por el wizard), veamos cómo están formados, descomponiendo los parámetros de uno de los ejemplos:

```
http://sensors.fonts.cat/widget/
    name=compass
    service=http://demo.geomati.co/sos/json
    offering=http://sensors.portdebarcelona.cat/def/weather/offerings#10m
    feature=http://sensors.portdebarcelona.cat/def/weather/features#P3
    property=http://sensors.portdebarcelona.cat/def/weather/properties#31
    refresh_interval=5
    lang=en
```

Nota: Una URL válida debe codificar cada parámetro utilizando la función estándar de javascript `encodeURIComponent` (o su equivalente en otros lenguages). Por claridad, en el ejemplo se muestran los parámetros decodificados.

Como se puede ver, los parámetros de la URL son mayormente los parámetros de entrada del widget. El formulario del wizard nos presenta los nombres de todos los offerings, features y properties, pero los parámetros del widget usan sus correspondientes identificadores. El wizard interrogó el servicio SOS para recuperar todos los posibles pares de nombre e identificador. En caso de querer acceder a los identificadores manualmente, puede hacerse a través de la operación `GetCapabilities`.

La URL también contiene un par de parámetros extra que no son estrictamente parámetros de configuración del widget:

- El primero, "name": Es el nombre del widget a crear.
- The último, "lang": Se utiliza para determinar el idioma de los posibles textos del widget. Es un parámetro opcional, y su valor por defecto es el inglés ("en"). Otras lenguas soportadas son el Español ("es") y el Catalán ("ca").

La opción "incrustar" simplemente incluye el enlace anterior en un elemento HTML <iframe>, de modo que pueda ser usado como componente en otras páginas:

```
<iframe src="..." width="570" height="380" frameBorder="0"></iframe>
```

El ancho y alto del <iframe> vienen determinados en primera instancia por el tamaño inicial indicado en el formulario del wizard, pero pueden cambiarse mediante el wizard simplemente redimensionando el recuadro de "Vista del Wizard" (nótese el control en su esquina inferior izquierda).

2.3 Uso en Javascript

Pro último, la forma más flexible de usar los widgets es por programación. Simplemente ha de incluírse la librería de Sensor Widgets en la página, que está disponible en http://sensors.fonts.cat/js/SensorWidgets.js , e instanciar el widget usando la factoría `SensorWidget`, que toma 3 parámetros:

```
SensorWidget(nombre, configuracion, elemento);
```

El nombre del widget es una cadena de texto, la configuracion es un objeto cuyas propiedades son sus parámetros de configuración, y el elemento es el elemento DOM donde dibujar el widget.

La forma más práctica de crear un widget programáticamente es usando el wizard y copiando y pegando el trozo de código javascript que genera. A partir de este código, se puede añadir dinamismo al código cambiando alguno de sus parámetros de configuración antes de instanciarlo.

Véase un ejemplo de integración práctico en: http://bl.ocks.org/oscarfonts/5ad801cf830d421e55eb

Nota: La función `SensorWidget` no devuelve ningún resultado, pero alguno de los parámetros acepta una función de callback. Los widgets se crean de forma asíncrona. En caso de error, se mostrará un mensaje al usuario en el elemento donde debía dibujarse el widget.

2.3.1 Haciendo llamadas SOS de bajo nivel con Javascript

Advertencia: El acceso directo a las operaciones SOS de bajo nivel es experimental. La API aquí descrita puede cambiar en cualquier momento.

La instancia del cliente SOS se obtiene de forma asíncrona:

```
getSOS(function(SOS) {
    // Debe indicarse una URL de un servicio 52n SOS 4.x con encoding JSON
    SOS.setUrl("http://sensorweb.demo.52north.org/sensorwebtestbed/service");
    // A partir de aquí, se puede llamar al resto de métodos de SOS
});
```

Esta es la API:

```
SOS.getCapabilities(callback, error); // Obtiene la sección de "contents" del␣
↪GetCapabilties.
SOS.describeSensor(procedure, callback, error); // Obtiene el documento SensorML␣
↪convertido a una estructura JSON.
SOS.getFeatureOfInterest(procedure, callback, error); // Obtiene todas las␣
↪FeatureOfInterest del procedure indicado.
SOS.getDataAvailability(procedure, offering, features, properties, callback,␣
↪error); // Obtiene el rango de fechas válido para cada combinación de procedure,␣
↪feature y property.
SOS.getObservation(offering, features, properties, time, callback, error); //␣
↪Obtiene las observaciones para la combinación de parámetros dada.
```

Donde los parámetros son:

- *callback* (función) recogerá la respuesta como un objeto Javascript (JSON parseado).
- *error* (función) de callback que se llamará en caso de que el servicio SOS retorne un error.
- *procedure* (string) identificador de procedure.
- *offering* (string) identificador de offering.
- *features* (array de strings) lista de las Features Of Interest de las que se quiere obtener respuesta.
- *properties* (array de strings) lista de las Observable Properties de las que se quiere obtener respuesta.
- *time* el instante (si es string) o rango de tiempo (si es array de 2 strings) para el que se quiere obtener respuesta. Las fechas se indican en hora UTC, formato "yyyy-mm-ddThh:mm:ssZ". También puede usarse el valor especial "latest" para obtener la observación más reciente disponible.

Y su obligatoriedad es:

- La función de *callback* es siempre obligatoria, y la función de *error* es siempre opcional.
- Para *describeSensor* y *getFeatureOfInterest*, es obligatorio indicar la *procedure*.
- Para *getDataAvailability* y *getObservation* los filtros (procedure, offering, features, properties, time) son opcionales. Indíquese *undefined* en caso de no querer filtrar por uno de estos conceptos.

2.4 Personalización del aspecto gráfico

Todos los widgets admiten un parámetro opcional `custom_css_url`. En él se puede apuntar a una hoja de estilos CSS cuyas reglas sobreescriban el estilo por defecto de los widgets.

Todos los widgets están contenidos en un elemento con dos clases: la clase `widget`, y una clase con el nombre del widget. Por ejemplo, la siguiente regla CSS se aplicará a todos los widgets:

```
.widget {
    border: 2px solid black;
}
```

Y la siguiente se aplicará sólo para widgets del tipo `compass`:

```
.widget.compass {
    background-color: grey;
}
```

Otro elemento común es la nota al pie, que se encuentra bajo un elemento de la clase `footnote`. Puede cambiarse el aspecto de la nota a pie así:

```
.widget .footnote {
    font-color: red;
}
```

Incluso pueden ocultarse ciertos elementos del widget. Por ejemplo, el título principal en un termómetro:

```
.widget.thermometer h1 {
    display: none;
}
```

Para simbolización más específica, una buena práctica es inspeccionar el DOM del widget, y aplicar las reglas CSS según los elementos observados.

CAPÍTULO 3

Los Widgets uno a uno

3.1 Rumbo (compass)

El widget "compass" pertenece a la categoría de "valor instantáneo único". Muestra el último valor disponible para una Property específica, que en este caso expresa un azimut o ángulo respecto al norte. El widget interrogará al servidor periódicamente para actualizar el valor mostrado.

Sus parámetros obligatorios son:

- "service", "offering", "feature" y "property": determinan la propiedad cuyos valores quieren mostrarse. La propiedad debe tomar valores entre 0 y 360 (y la unidad de medida se supone que son grados centesimales).
- "refresh_interval" (en segundos): es el tiempo entre dos interrogaciones al servidor (el widget lanza GetObservations periódicas cada X segundos).

Otros parámetros opcionales:

- "title": Si no se especifica, por defecto se usa como título el nombre de la Feature.
- "display_utc_times": Las fechas del servicio SOS se obtienen en UTC, y por defecto se convierten al huso horario local del navegador. Si se quieren mostrar en UTC, poner este parámetro a *true*.
- "footnote": Texto opcional que aparecerá como una pequeña nota al pie.
- "custom_css_url": hoja de estilos css que se aplicará al widget.

3.2 Manómetro (gauge)

Otro widget de "valor instantáneo único", en esta ocasión para presentar valores de porcentaje entre 0 y 100.

Parámetros obligatorios:

- "service", "offering", "feature" y "property": determinan la propiedad cuyos valores quieren mostrarse. La propiedad debe tomar valores entre 0 y 100 (y la unidad de medida se supone que es un tanto porciento).
- "refresh_interval" (en segundos): es el tiempo entre dos interrogaciones al servidor (el widget lanza GetObservations periódicas cada X segundos).

Parámetros opcionales:

- "footnote": Texto opcional que aparecerá como una pequeña nota al pie.

- "display_utc_times": Las fechas del servicio SOS se obtienen en UTC, y por defecto se convierten al huso horario local del navegador. Si se quieren mostrar en UTC, poner este parámetro a *true*.
- "custom_css_url": hoja de estilos css que se aplicará al widget.

3.3 Tabla jQuery (jqgrid)

Muestra una tabla jqGrid con un conjunto de observaciones para un período de tiempo determinado, donde cada registro es una observación. La lista de resultados se muestra paginada y puede ser ordenada por los valores de cualquiera de las columnas (Hora, Feature, Propiedad, Valor y Unidad de medida).

Parámetros obligatorios:

- "service", "offering", un conjunto de "features" y un conjunto de "properties": selecciona el conjunto de combinaciones feature-property a mostrar.
- "time_start" and "time_end": Selecciona las observaciones que caen dentro de este rango de tiempo.
- "title": el título del widget.

Parámetros opcionales:

- "footnote": Texto opcional que aparecerá como una pequeña nota al pie.
- "display_utc_times": Las fechas del servicio SOS se obtienen en UTC, y por defecto se convierten al huso horario local del navegador. Si se quieren mostrar en UTC, poner este parámetro a *true*.
- "custom_css_url": hoja de estilos css que se aplicará al widget. Nótese que el aspecto de jqGrid se toma del tema jQuery-ui subyacente.

Nota: este widget depende de jQuery, jQuery UI y el plugin jgGrid. Es un widget bastante pesado y no muy personalizable (se desarrolló como un ejercicio de integración con una aplicación existente). Se recomienda el uso de otros widgets como el "table", que van más en la línea de lo que pretenden ser los Sensor Widgets: ligeros, compactos y flexibles.

3.4 Mapa (map)

Este widget es especial en varios sentidos. En primer lugar, muestra la respuesta a una petición GetFeatureOfInterest, en lugar del caso más habitual en que un widget representa la respuesta a un GetObservation.

En segundo lugar, es un widget muy configurable, a través de algunos parámetros complejos. Afortunadamente la mayoría de parámetros son opcionales, de modo que su uso más elemental es de hecho muy sencillo.

Está basado en la librería de mapas Leaflet.

Los únicos parámetros estrictamente obligatorios son:

- "service" y "offering": Determinan el offering cuyos Features of Interest quieren mostrarse sobre el mapa.

Esto mostrará un mapa con las Features. Pasando el ratón por encima de una Feature, se mostrará una pequeña etiqueta con el nombre de la feature.

Hay otro par de parámetros formalmente obligatorios (aunque pueden dejarse vacíos):

- "features": Podemos seleccionar sólo algunas features para mostrar en el mapa. Si no se indica ninguna, de hecho se mostrarán *todas* (no se aplica ningún filtrado). Pero este parámetro debe existir, aunque sea como una lista vacía.
- "properties": Si se indican una o más properties, la etiqueta de cada feature se convertirá de hecho en un pequeño widget de tipo "panel", mostrando los últimos valores de cada property para cada una de las features. Nuevamente, puede indicarse una ista vacía de properties, en cuyo caso, NO se mostrará ningún valor.

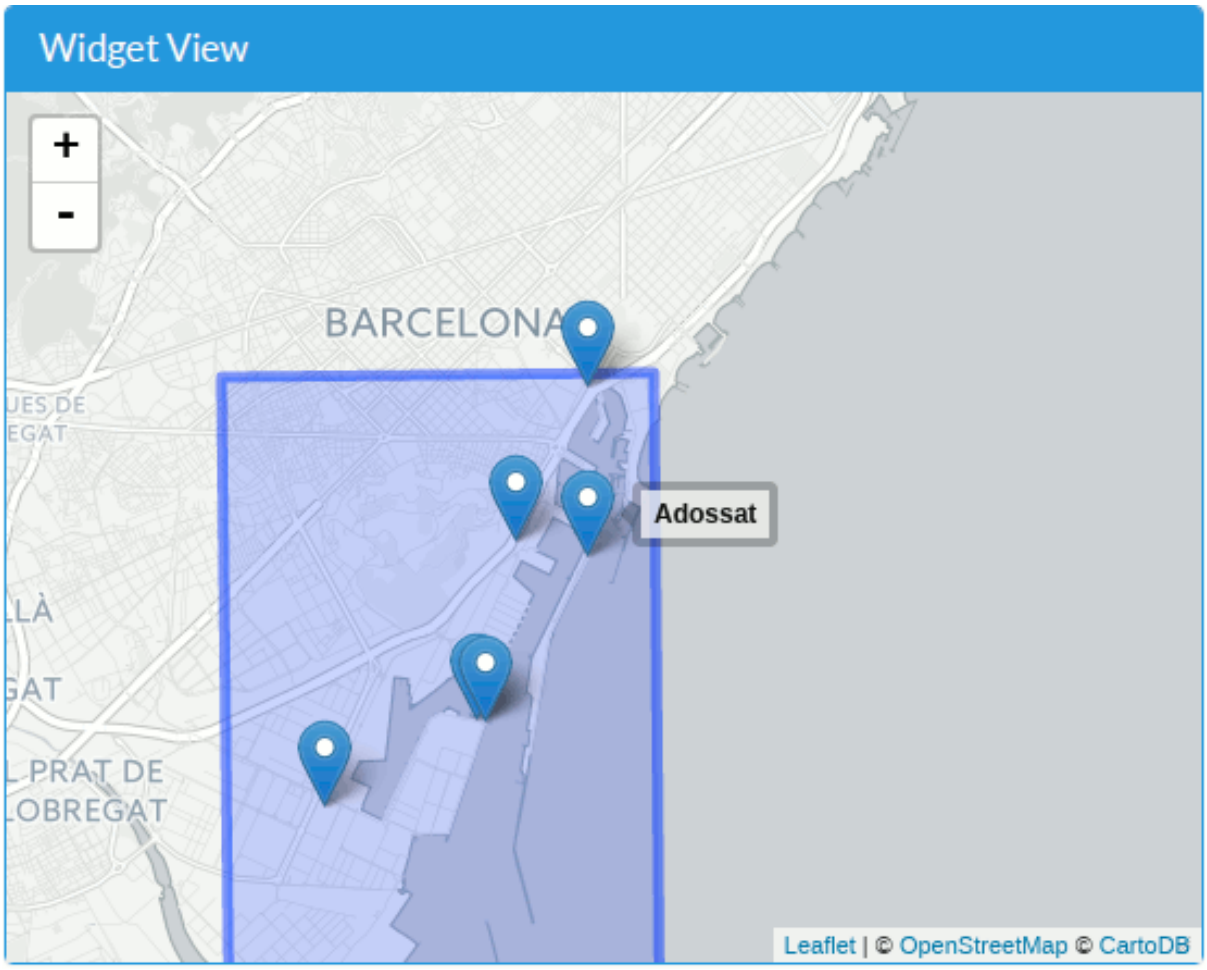

Figura 3.1: Mapa simple donde no se han indicado ni features ni properties.

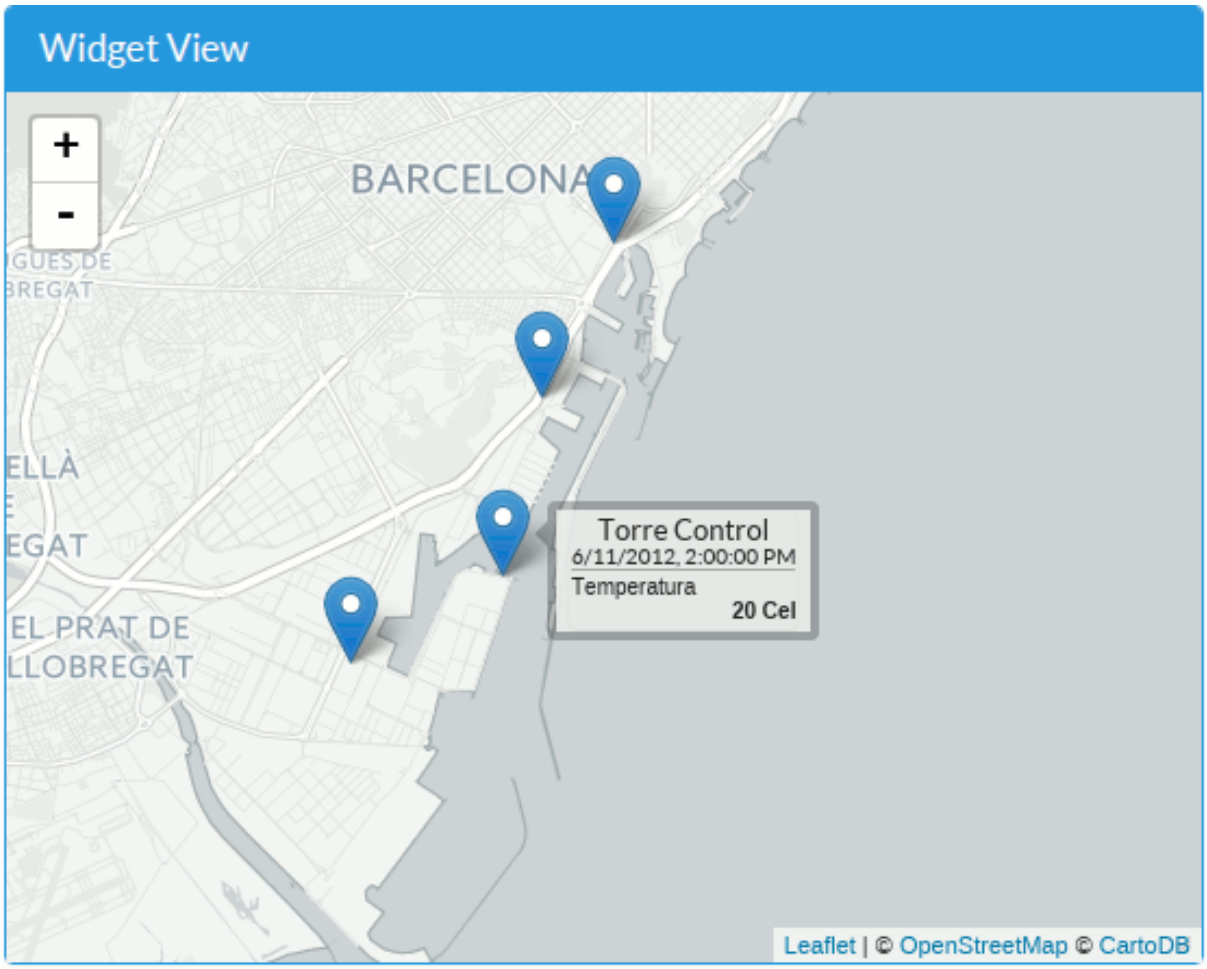

Figura 3.2: Mapa donde se han seleccionado cuatro features y una property, cuyo valor se muestra en la etiqueta.

El parámetro opcional “permanent_tooltips”, si toma el valor “true”, hará que se muestren todas las etiquetas permanentemente, no sólo cuando se pase el ratón por encima.

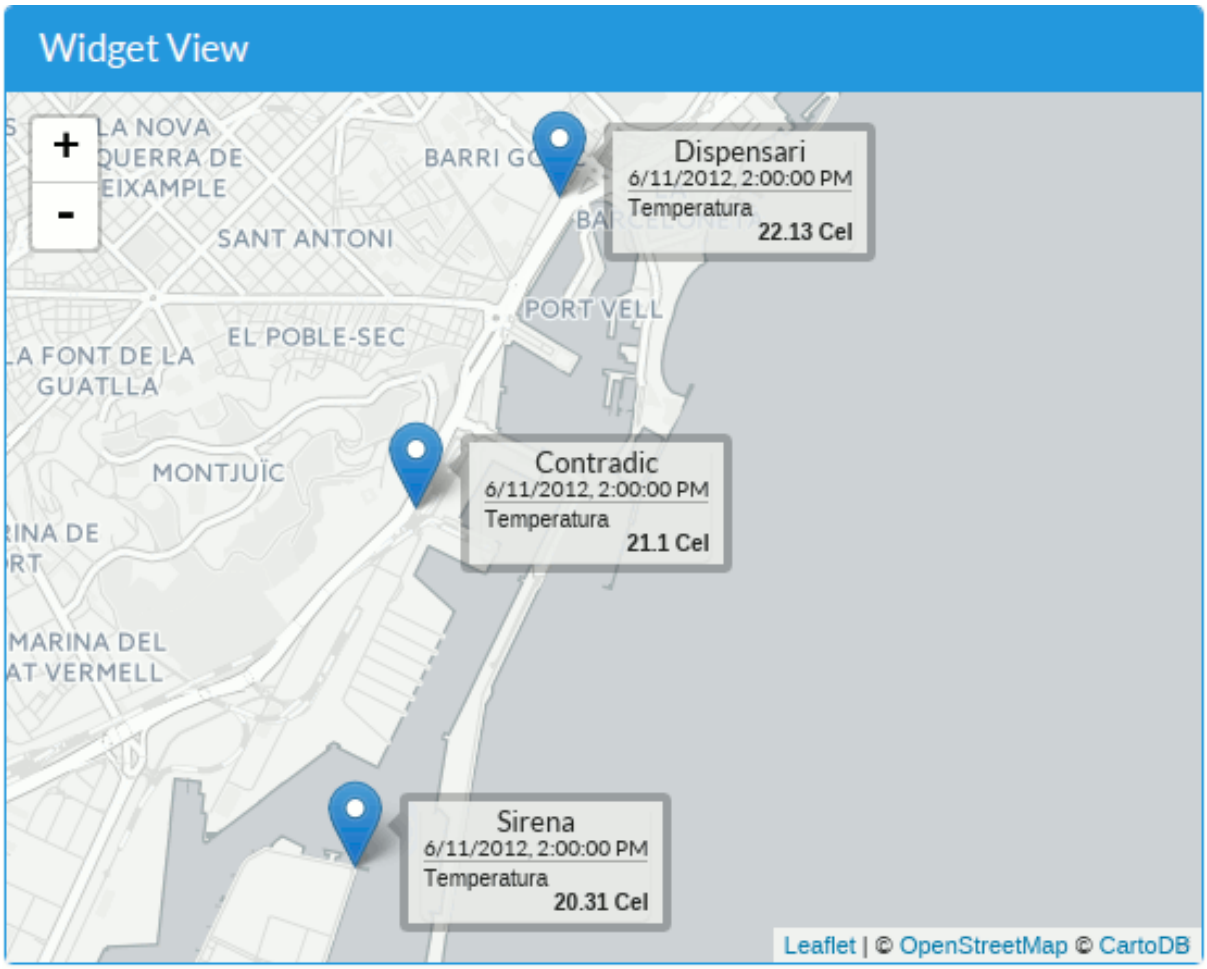

Figura 3.3: Mapa con etiquetas permanentes.

Si los elementos sobre el mapa aparecen en la otra punta del mundo, es probable que haya que cambiar el orden de los ejes de coordenadas. Añadiendo el parámetro opcional “swap_axis”=true, se intercambiarán latitud y longitud, y se corregirá este efecto.

Además de las etiquetas, también podemos vincular un sub-widget a cada feature, que se mostrará en un globo al hacer clic sobre ella. El parámetro “popup_widget” toma como valor un JSON de configuración de dicho sub-widget. En esta configuracion, los parámetros “service”, “offering” y “feature(s)” se obtienen del widget *padre* (el mapa), así que no deben indicarse. La propiedad “name” indica qué clase de widget queremos incrustar.

Por ejemplo, si queremos que se abra un globo conteniendo una gráfica temporal al hacer clic en cada feature, debemos indicar:

- “name”: “timechart”,
- ...todos los parámetros del widget timechart, excepto “service” y “offering”.

Es decir:

```
{
    "name": "timechart",
    "title": "Temperatures",
    "properties": [
        "http://sensors.portdebarcelona.cat/def/weather/properties#32M",
        "http://sensors.portdebarcelona.cat/def/weather/properties#32",
        "http://sensors.portdebarcelona.cat/def/weather/properties#32N"
```

```
    ],
    "time_start": "2015-09-03T05:05:40Z",
    "time_end": "2015-09-03T08:05:40Z"
}
```

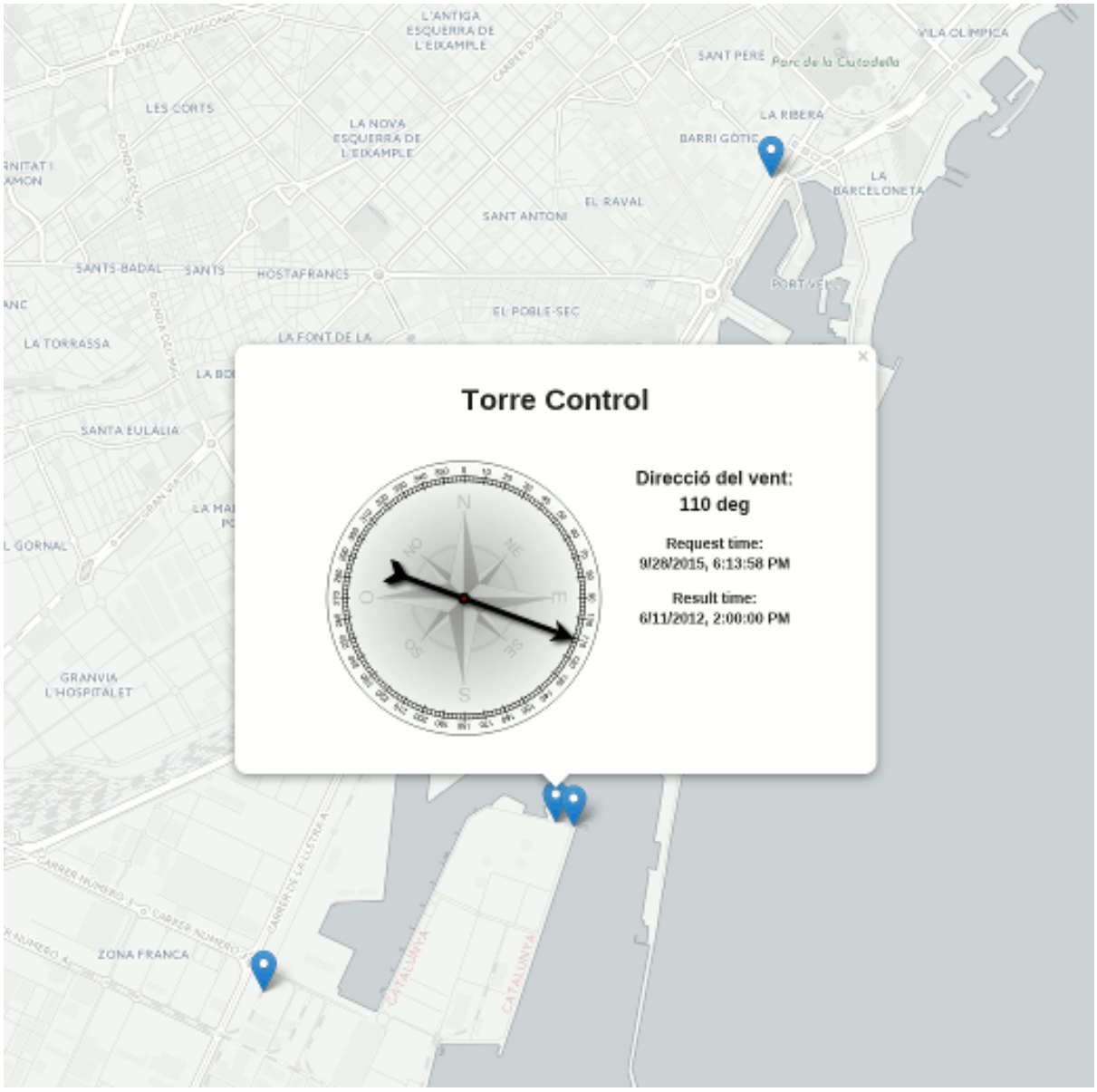

Figura 3.4: Mapa con un “popup_widget” de tipo “compass”.

Además de personalizar las etiquetas y los globos con detalles acerca de cada feature, podemos cambiar la cartografía de base del mapa mediante el parámetro “base_layer”. Se pueden especificar dos tipos de capa base:

- Una capa de teselas: Debe indicarse una “url” y un conjunto de “options”. Por ejemplo:

```
{
   "url": "http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
   "options": {
          "maxZoom": 19,
          "attribution": "&copy; <a href='http://www.openstreetmap.org/
↪copyright'>OpenStreetMap contributors</a>"
       }
}
```

Los parámetros “url” y “options” se corresponden con los parámetros del constructor TileLayer de Leaflet “urlTemplate” y “TileLayer_options” respectivamente.

Se puede escoger entre una buena colección de capas de teselas: http://leaflet-extras.github.io/leaflet-providers/preview/

- Una capa WMS: Debe especificarse “type”=”wms”, una “url” y un conjunto de “options”. Por ejemplo:

```
{
   "type": "wms",
   "url": "http://geoserveis.icc.cat/icc_mapesbase/wms/service",
   "options": {
      "layers": "orto5m",
      "format": "image/jpeg",
      "attribution": "Ortofoto 1:5.000: CC-by <a href='http://www.icc.cat'␣
↪target='_blank'>Institut Cartogràfic de Catalunya</a>"
   }
}
```

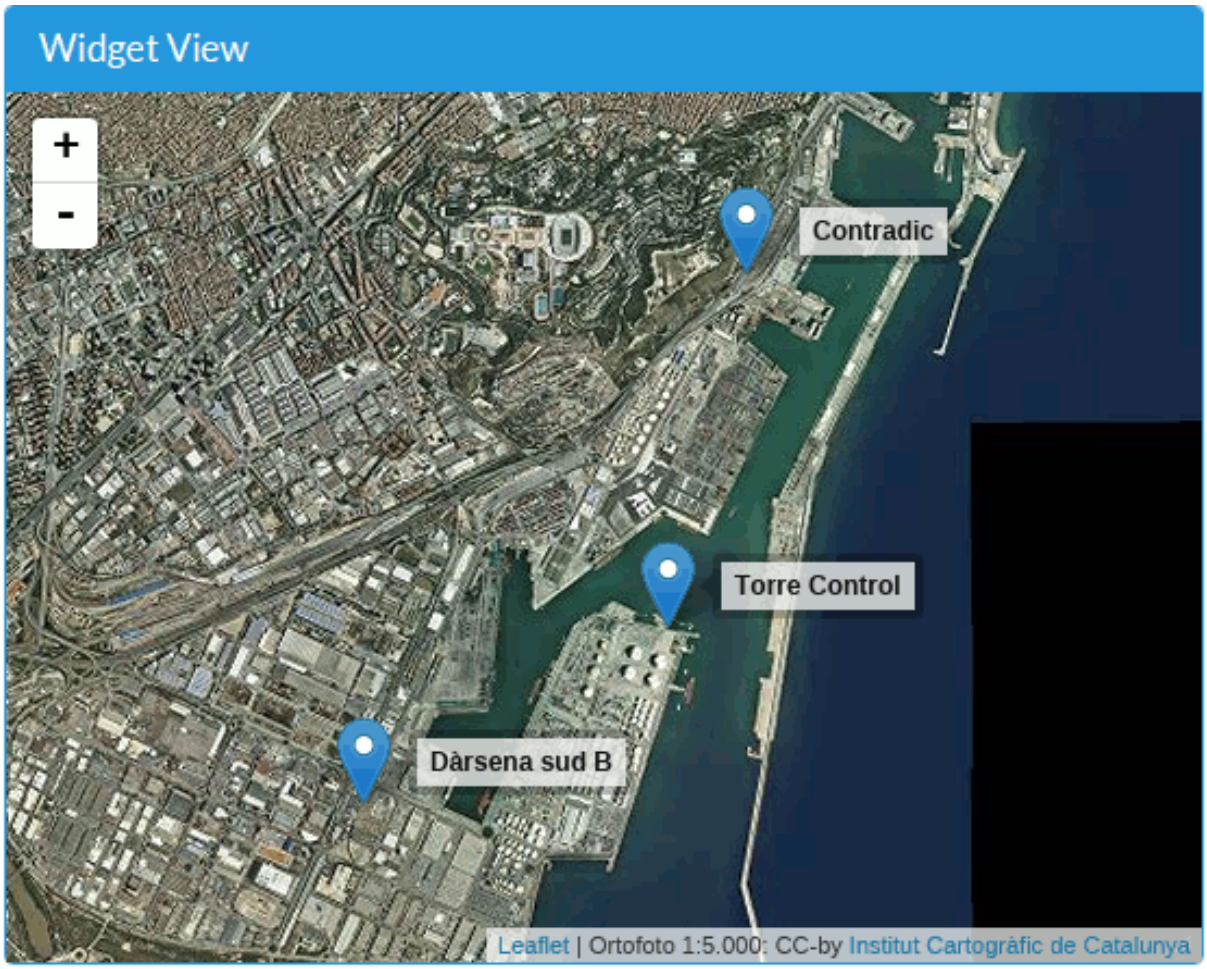

Figura 3.5: Mapa con cartografía WMS.

Los parámetros “url” y “options” se corresponden con los parámetros del constructor TileLayer.WMS de Leaflet “baseUrl” y “TileLayer.WMS_options” respectivamente.

Otro parámetro opcional es “max_initial_zoom”: Indica el nivel de zoom máximo a utilizar en la vista inicial del mapa. Esto evita acercarse demasiado y perder contexto cartográfico, especialmente cuando se muestra una única feature puntual.

Cuando existen muchos marcadores sobre el mapa, se aplica automáticamente una función de agrupación de los mismos (clustering). Si no se quiere aplicar el clustering de forma automática, debe ponerse a *true* el parámetro

opcional “no_clustering”.

Si se está usando el widget con Javascript, es posible capturar el “click” sobre los marcadores y obtener sus detalles:

```
"on_click": function(marker) {
   console.log(marker.feature);
}
```

Por último, los parámetros opcionales habituales “display_utc_times”, “footnote” y “custom_css_url” también están disponibles.

Véase un **ejemplo funcional completo** aquí: http://bl.ocks.org/oscarfonts/265d734349396cf4372c

3.5 Panel (panel)

El widget “panel” se usa para mostrar los útimos valores de un conjunto de propiedades de una Feature dada. Está construido como una Lista de Definiciones (<dl>) de HTML, compatible con las clases CSS de Bootstrap. El contenido del widget se actualizará automáticamente de forma periódica.

Sus parámetros obligatorios son:

- Los habituales “service”, “offering” y “feature”.
- Una lista de “properties” a mostrar.
- El “refresh_interval”, en segundos.

Y los parámetros opcionales: “title”, “display_utc_times”, “footnote” y “custom_css_url”.

El panel también mostrará la fecha de las observaciones como subtitulo. En caso de que alguno de los valores sea de una fecha anterior a la fecha común, se mostará el valor en color rojo y se mostrará la fecha para dicha observación en particular.

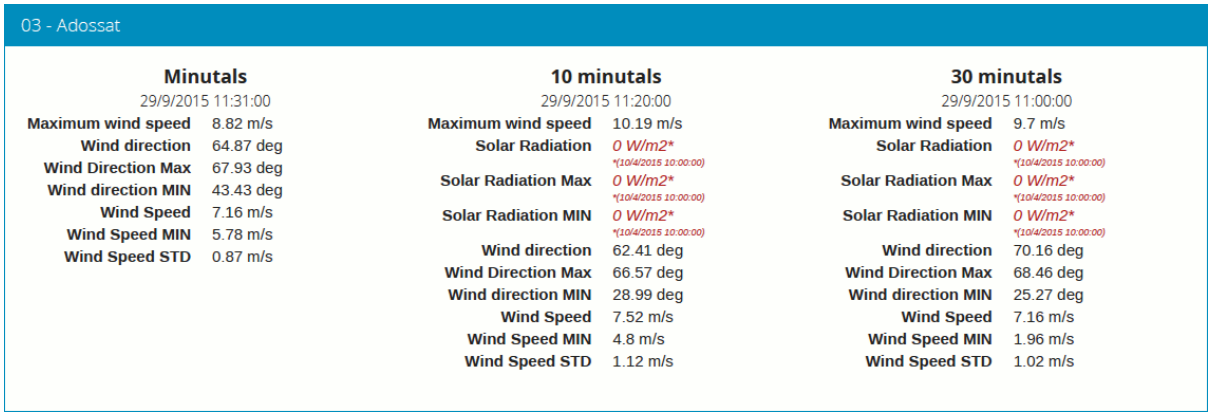

Figura 3.6: Tres widgets de tipo Panel, algunos de ellos mostrando valores antiguos.

3.6 Barra (progressbar)

Otro widget que muestra un valor instantáneo, esta vez mostrado como una barra proporcional entre dos valores. Es útil para mostrar gráficamente dónde cae un valor respecto a sus valores límite. Se puede usar para mostrar un porcentaje si se ajustan los valores mínimo y máximo a 0 y 100 respectivamente, en cuyo caso sería muy similar a un widget de tipo “gauge” pero mostrando el valor linealmente, pero “progressbar” también puede tomar otros valores límite distintos, con lo que es más flexible que “gauge”. Además el contenido es HTML, cuyo aspecto es más fácil de personalizar mediante CSS.

Parámetros obligatorios:

- Los habituales “service”, “offering”, “feature” y “property”.
- “min_value” y “max_value”, que determinan los valores extremos.
- “refresh_interval” en segundos.

Y los parámetros opcionales habituales: “display_utc_times”, “footnote” y “custom_css_url”.

3.7 Status (status)

El widget “status” muestra el estado global de todo un offering de un vistazo. Dado un offering, construye una tabla cuyas celdas representan todas las possibles combinaciones de feature-property. Para cada una, se muestra el último valor observado y su antigüedad. Es una buena forma de inspeccionar el estado de salud de un offering: Si están llegando nuevas observaciones, y para qué sensores.

Este widget está pensado como una herramienta de supervisón (una especie de hiper-tabla), y es más útil si se muestra a pantalla completa.

Sus únicos parámetros obligatorios son “service” y “offering”.

Y los parámetros opcionales habituales: “display_utc_times”, “footnote” y “custom_css_url”.

3.8 Tabla (table)

Dados un feature y un período de tiempo, un widget “table” muestra las observaciones de un conjunto de propiedades a lo largo del tiempo. Es similar a “jqgrid” pero proporciona una vista más compacta. El widget es una simple tabla HTML con clases CSS compatibles con Bootstrap.

Parámetros:

- Los habituales “service”, “offering” y “feature”.
- Una lista de “properties” a mostrar.
- “time_start” y “time_end”: Período de tiempo del que quieren obtenerse observaciones.
- Y el “title”.

Además de los parámetros opcionales comunes: “display_utc_times”, “footnote” y “custom_css_url”.

3.9 Termómetro (thermometer)

Otro widget de tipo “valor instantáneo único”, tal como Compass y Gauge, pero para mostrar una temperatura ambiental en grados Celsius.

Muestra el dibujo de un termómetro que puede tomar valores de los -24ºC a los 56ºC. También se muestra el valor numérico. Como otros widgets de su categoría, incorpora un mecanismo de actualización periódica.

Parámetros obligatorios:

- “service”, “offering”, “feature” y “property”: Determinan la property cuyos valores quieren mostrarse. Se le supone grados centígrados como unidad de medida.
- “refresh_interval” (en segundos): el tiempo entre actualizaciones del valor.

Otros parámetros opcionales:

- “footnote”: Texto opcional que aparecerá como una pequeña nota al pie.
- “display_utc_times”, para mostrar la hora en tiempo universal y no en el huso horario local.
- “custom_css_url”: hoja de estilos css que se aplicará al widget.

3.10 Serie tiempo (timechart)

Dados una feature y un rango de tiempo, muestra los valores que van tomando ciartas propiedades a lo largo del tiempo. Su interfaz es la misma que el widget "table", pero los resultados se muestran en una gráfica.

Las gráficas se construyen gracias a la librería Flot, que a su vez depende de jQuery.

Parámetros:

- Los habituales "service", "offering" y"feature".
- La lista de "properties" a mostrar.
- "time_start" y "time_end": Período de tiempo del que quieren obtenerse observaciones.
- Y el "title".

Parámetros opcionales:

- "colors": Array de colores en formato *#rrggbb*, que se aplicarán al dibujar las líneas de cada una de las properties indicadas.
- "callback": Función que se llamará tras instanciar el widget. Recoge la instancia del Flot chart como parámetro.

Además de los parámetros opcionales comunes: "display_utc_times", "footnote" y "custom_css_url".

3.11 Rosa vientos (windrose)

Este es un widget para un caso de uso muy específico: mostrar estadísticas del régimen de vientos, donde se puede apreciar de un vistazo la dirección y velocidad predominante del viento, y también su variabilidad sobre un período de tiempo.

Nota: La gráfica polar está basada en la librería Highcharts. Esta librería es gratuíta para usos no comerciales, pero **debe adquirirse una licencia para su uso comercial**.

Parámetros obligatorios:

- "service", "offering", "feature": determinan una localización, de la que deben existir datos de dirección y velocidad del viento.
- "properties": admite un array de dos (y sólo dos) properties. Una será la velocidad del viento en `m/s`, y la otra su dirección en `deg`. Las observaciones para ambas properties deben producirse a intervalos regulares y de forma síncrona.
- "time_start" y "time_end": el período de tiempo sobre el que se descargarán datos y se extraerán estadísticas.
- "refresh_interval" (en segundos): tiempo entre actualizaciones del widget. Se recomiendan valores de varios minutos para no saturar el servidor, puesto que la cantidad de datos a descargar es grande, pero las estadísticas sobre un período de tiempo largo no cambian bruscamente.
- "title" el título del widget.

Parámetros opcionales:

- "subtitle".
- "display_utc_times", "footnote" y "custom_css_url".

Así es como se agrupan los datos para construír la gráfica de la rosa de los vientos:

1. Los valores de dirección del viento se clasifican en 16 sectores: N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, NNW and N.

2. Para cada sector, las velocidades del viento correspondeoentes se clasifican en rangos: 0-2 m/s, 2-4 m/s, 4-6 m/s, 6-8 m/s , 8-10 m/s y > 10 m/s.

Se dibuja entonces una gráfica polar con 16 columnas, en cada una de las cuales apilan diferentes segmentos coloreados, proporcionales al conteo de observaciones para cada rango de velocidades.

Nota: A diferencia de otros widgets, más ligeros y flexibles, este requiere que el servicio SOS del que se alimenta exponga los datos de una forma muy concreta. Además, depende de una librería de gráficos no exactamente libre. Pero los resultados para el caso de uso que cubre son excelentes. Así pues, tómese éste widget no como uno genérico y reusable, sino como un ejemple de la *especialización* a la que se puede llegar programando widgets propios. Para desarrollar widgets propios que le ayuden a expresar mejor sus propios datos, consulte el capítulo sobre cómo contribuir al proyecto (en ingles).

CAPÍTULO 4

Cómo contribuir al proyecto

La documentación para desarrolladores está disponible únicamente en inglés.