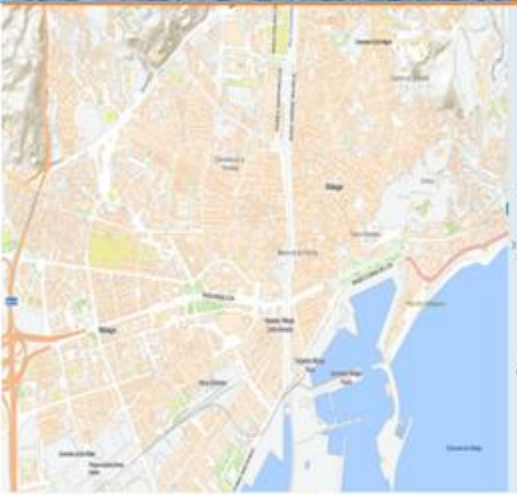


Servicios interoperables desde clientes (QGIS, Iberpix…) y datos abiertos

API del CNIG

José María García Malmierca
jmgmalmierca@mitma.es

Cecilia Poyatos Hernández
cecilia.poyatos@cnig.es

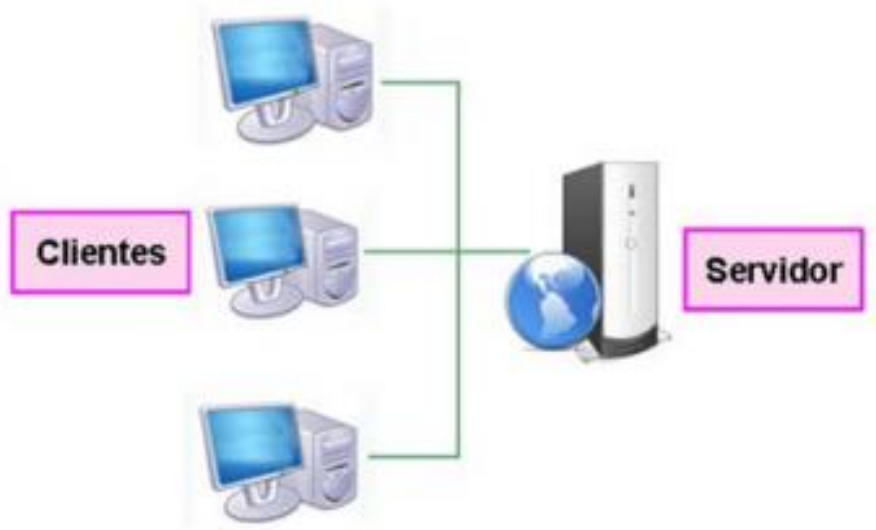
IDE: Infraestructura de Datos Espaciales

Una **Infraestructura de Datos Espaciales**, **IDE**, es un **sistema informático integrado** por un **conjunto de datos y servicios** (descritos a través de sus **metadatos**) que son gestionados a través de **internet**, conforme a **estándares** que regulan y garantizan la **interoperabilidad** de sus datos y a acuerdos políticos que permiten que un **usuario**, por medio de un simple **navegador**, pueda encontrar, **visualizar**, acceder y combinar la Información Geográfica según sus necesidades.

Está integrado por un conjunto de recursos informáticos, como por ejemplo, pueden **ser clientes de catálogos de datos** y de **geoservicios**, **servidores de mapas**, de objetos geográficos o de coberturas, **visualizadores**, **buscadores de nombres geográficos**, etc.

IDE: Infraestructura de Datos Espaciales

- En los proyectos IDE, la arquitectura por excelencia es la denominada arquitectura **cliente - servidor**, en la que una serie de clientes (visualizadores, navegadores web) solicitan una serie de servicios a servidores remotos (ordenadores). Estos últimos procesan las peticiones de los navegadores (realizadas según el protocolo HTTPS) y devuelven una respuesta.



IDE: Infraestructura de Datos Espaciales

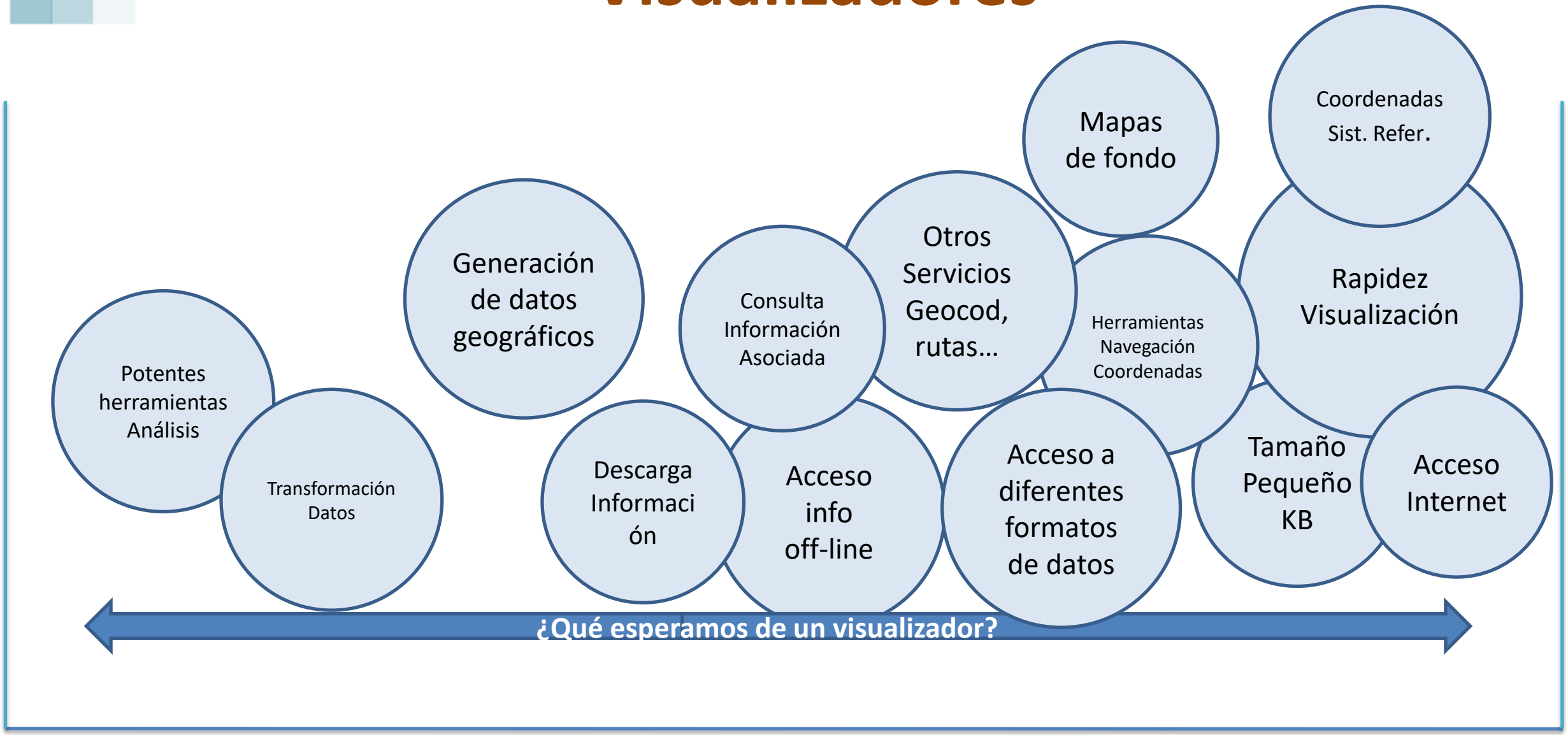
- Por tanto, una IDE está compuesta por **Información Geográfica o Datos geográficos**, que se publican a través de:
  - **Servicios interoperables** o geoservicios de información geográfica distribuidos en diferentes sistemas de información bajo la **responsabilidad y gestión de distintas instancias del sector público o privado.**
  - Los servicios web deben cumplir un mínimo de **protocolos** y **especificaciones normalizadas**, que se establecen con la finalidad de facilitar el acceso a todos esos datos.
  - Tanto los datos geográficos como los servicios web deben estar descritos mediante sus **metadatos** para que se puedan localizar y conocer sus características
  - Los **acuerdos** son necesarios sobre su puesta en común, acceso y utilización entre sus **productores** y entre éstos y los **usuarios**; y los **mecanismos**, procesos y procedimientos de coordinación.
  - …. y sobre todo, debe de disponer de herramientas que facilite al usuario común el uso final de la información, como por ejemplo, visualizadores.

Visualizadores

- Son varios las características que observamos en el momento de definir que es un visualizador. Queda claro que:
  - **Se trata de una herramienta del lado del cliente.**
  - Su objetivo es facilitar el **acceso y la visualización de información geográfica.**
  - Deber primar la **rapidez de visualización** frente a la capacidad de análisis.
    - Diseño sencillo
    - Herramientas accesibles
    - etc
  - Debe permite el acceso de servicios de visualización a través de la **Web** (HTTP)

  También debería permitir:
  - Facilitar la combinación de información espacial.
  - Incorporar otro tipo de servicios, no sólo de visualización.
  - Disponer de distintos tipos de herramientas.

Visualizadores



Visualizador Web. Aspectos

- En un visualizador deben considerarse diferentes **aspectos en función del objetivo**:
  - **Objetivo del visualizador**.
    - Es muy diferente una solución para mostrar un atlas con cartografía temática que para mostrar, por ejemplo, la localización de terremotos.
  - **Diseño**
  - **Temática concreta**.
  - **Herramientas ajustadas al propósito del visualizador.**
  - Soporte estándares que permitan generar valor añadido a través de la **interoperabilidad** de diferentes fuentes y tecnologías.

Visualizador Web. Aspectos

- Aspectos de desarrollo y legales
  - **Tecnología**.
    ¿Usamos fuentes abiertas o propietarias?
  - **Licenciamiento**.
    ¿Permite distribuir libremente nuestros propios desarrollos?
  - Donde vamos a alojarlo (*Hosting*).
  - Accesibilidad.
    Ejemplo aplicación Camino de Santiago.
  - **Reutilización**.
    ¿Existe lo que necesito? ¿Lo puedo tener gratis?

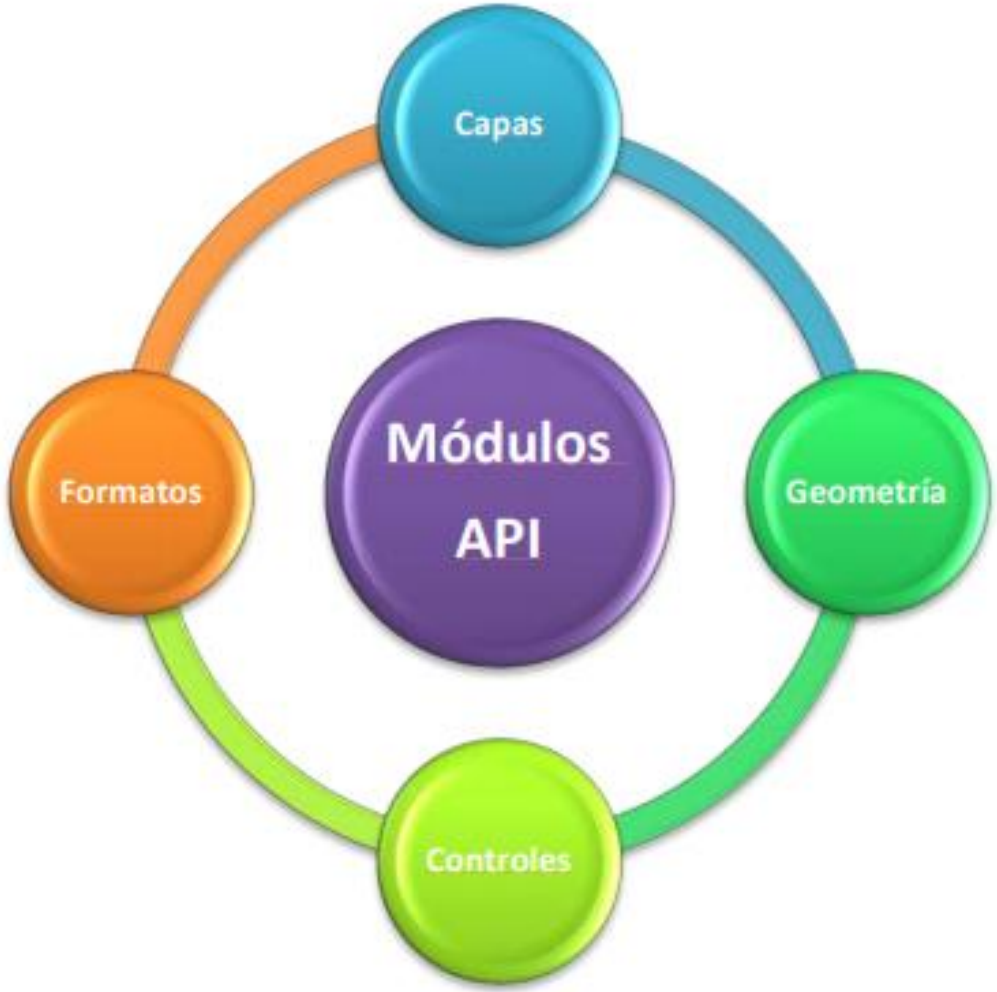
API (Application Programming Interface)

- El concepto hace referencia a los **procesos**, las **funciones** y los **métodos** que brinda una determinada **biblioteca de programación** a modo de capa de abstracción para que sea empleada por otro programa informático.
  https://definicion.de/api/

- Su principal característica es que **proporciona un conjunto de funciones de uso general que evitan tener que programar todo desde el principio**.
  - Así, por ejemplo, si una API dispone de funciones que permiten hacer zoom en un ámbito geográfico (cosa muy común), podemos incluir una llamada a dichas funciones en el código de nuestra aplicación y no tener que programar dicha funcionalidad de forma particular en cada aplicación que queramos desarrollar.

API (Application Programming Interface)

- **Capas**: se corresponden con todos los objetos y funciones que permiten incorporar diferentes tipos de información: WMS, KML, Google, imágenes, etc. –
- **Formatos**: se corresponden con los formatos de información que es posible incluir en una API: WMS, WFS, OSM, KML, GeoJSON, GeoRSS, shp,...
- **Controles**: Los controles se utilizan para interactuar con el mapa. Permiten hacer zoom, moverse por el mapa, conocer las coordenadas del cursor, medir distancias, etc.
- **Geometría**: Se refiere a la que corresponda a los objetos que se van a poder incorporar en la aplicación: punto, línea, superficie, etc.



API (Application Programming Interface). Ejemplos

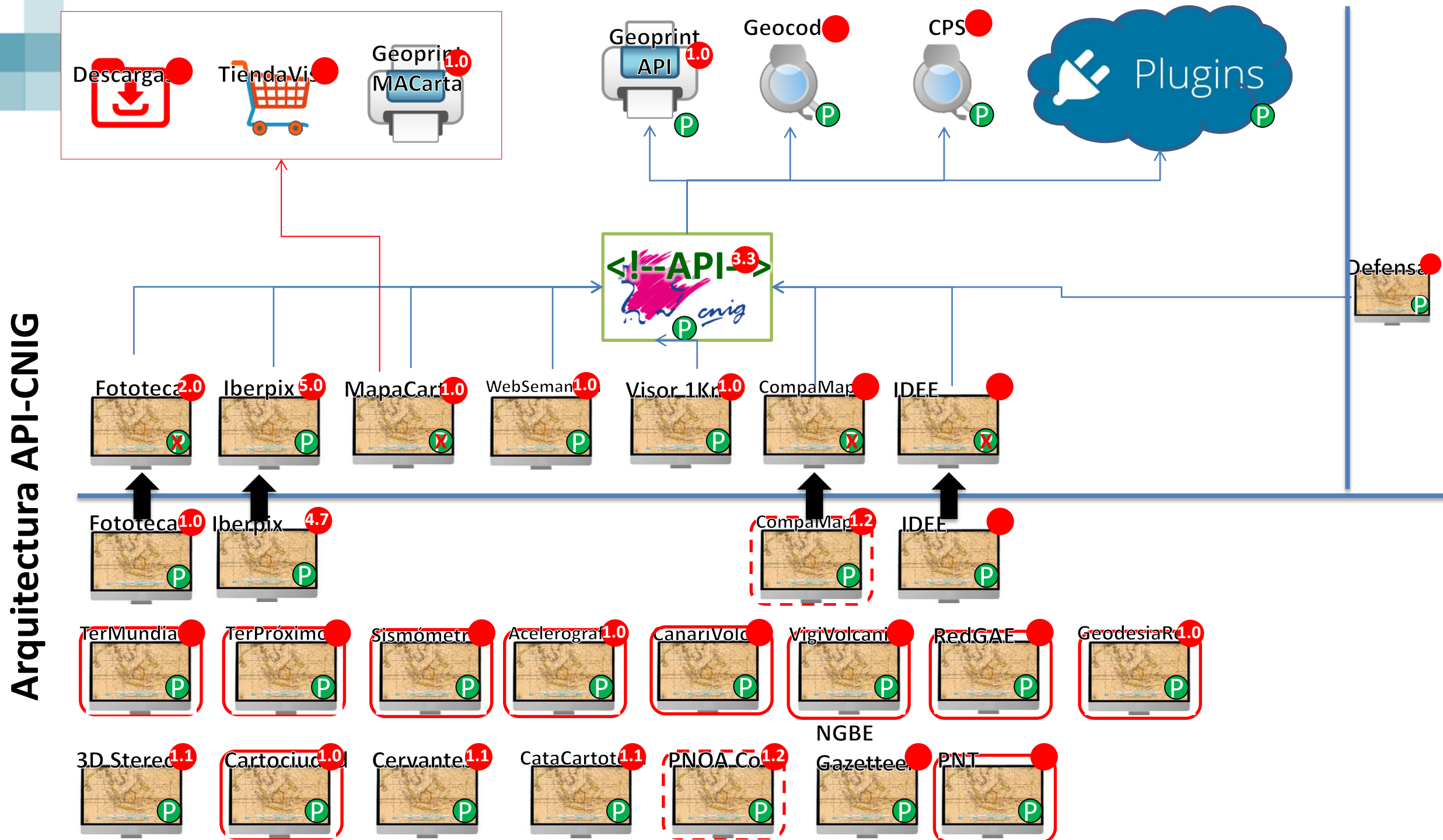
- **OpenLayers**
  es una biblioteca de JavaScript de código abierto para mostrar mapas interactivos en los navegadores web. OpenLayers ofrece un API para acceder a diferentes fuentes de información cartográfica en la red: Web Map Services, Mapas comerciales (tipo Google Maps, Bing, Yahoo), Web Features Services, distintos formatos vectoriales, mapas de OpenStreetMap, etc.

- **Leaflet**
  Leaflet es una librería JavaScript de código abierto para crear mapas interactivos en un entorno móvil.

- Carto, GoogleMaps API, MAPBOX

API-CNIG 3.0. Contexto

- **Han existido otras API en el CNIG**
  - No consiguen soportar todos los visualizadores
- **API-CNIG 3.0 en desarrollo desde 2019**
  - Objetivo: Dar soporte a gran parte de los visualizadores del IGN-CNIG
- **Primeros visualizadores con esta API en julio de 2019**


Arquitectura API-CNIG
Descarga
TiendaVis
Geoprint 1.0
MACarta
Geoprint
API 1.0
Geocod
CPS
Plugins
<!--API-> 3.3
cnig
Defensa
Fototeca 2.0
Iberpix 5.0
MapaCar 1.0
WebSeman 1.0
Visor 1Kn 1.0
CompaMap
IDEE
Fototeca 1.0
Iberpix 4.7
CompaMap 1.2
IDEE
TerMundia
TerPróximo
Sismómetr
Acelerograf 1.0
CanariVolc
VigiVolcani
RedGAE
GeodesiaRe 1.0
NGBE
3D Stereo 1.1
Cartociud 1.0
Cervantes 1.1
CataCartot 1.1
PNOA Co 1.2
Gazettee
PNT

API-CNIG 3.0

- **Rama de la API MAPEA de la Junta de Andalucia.**
  - Diferencias:
    - CORE más ligero.
    - Extensión mediante plugins/componentes.

- **Desarrollada en JavaScript usa Openlayers y utiliza HTML5 y CSS3.**
  - Contiene un módulo proxy para realizar peticiones a servicios externos.
  - API Rest para generar visores sencillos a partir de parámetros.

API-CNIG 3.0

- **Desarrollo pensado en su utilización en el IGN–CNIG y para reutilización por la comunidad IDE.**
  - Repositorio en GitHub
  - Wiki
  - Extensión de la API

- **Destinada para:**
  - usuarios que quieren desarrollar un navegador de forma rápida aprovechando las funcionalidades y extensiones que ofrece la API.
  - desarrolladores que deseen o necesiten extender la API en función de sus necesidades.

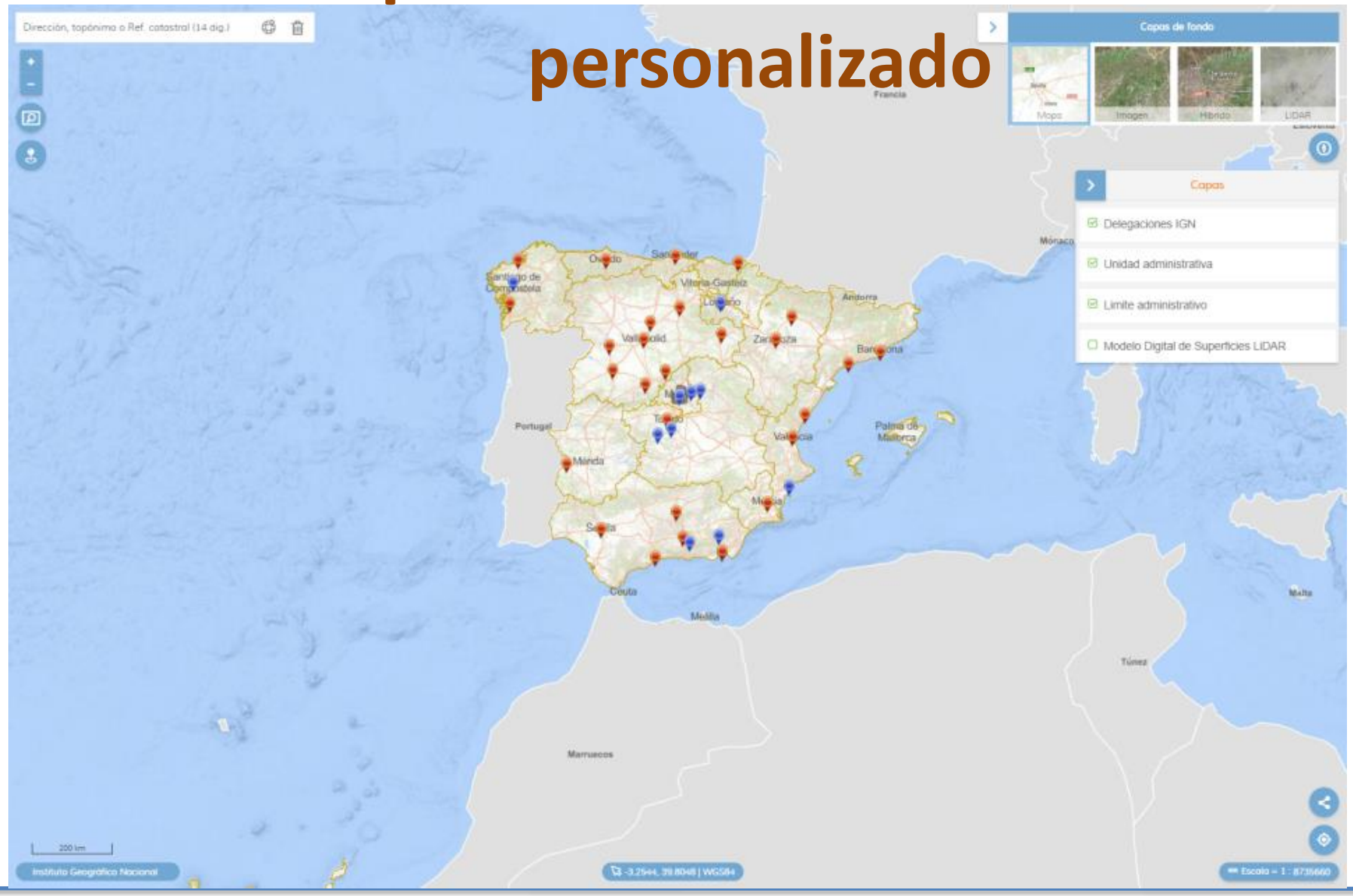
API-CNIG 3.0

- **Repositorio y documentación de la API:**
  - https://github.com/IGN-CNIG/API-CNIG
- **URL API-CNIG 3.0:**
  - https://componentes.cnig.es/api-core/
- **Principales elementos personalizables en la API para usuarios:**
  - Capas: información espacial que mostrar en el mapa.
    - Ráster: WMS, WMTS
    - Vectoriales: WFS, KML, GeoJSON
  - Controles básicos
  - Plugins :son funcionalidades que ofrecen utilidades específicas

API-CNIG 3.0

- **La API-CNIG cuenta con dos APIs.**
  - API JavaScript que permite crear desde visualizadores de mapas básicos hasta otros de mayor complejidad.
  - Una API REST muy sencilla que permite incluir un visualizador interactivo en cualquier página web sin necesidad de disponer de conocimientos específicos en programación ni de SIG (Sistemas de Información Geográfico).
- **A continuación se desarrolla ejemplos de visualizadores con ambas API.**

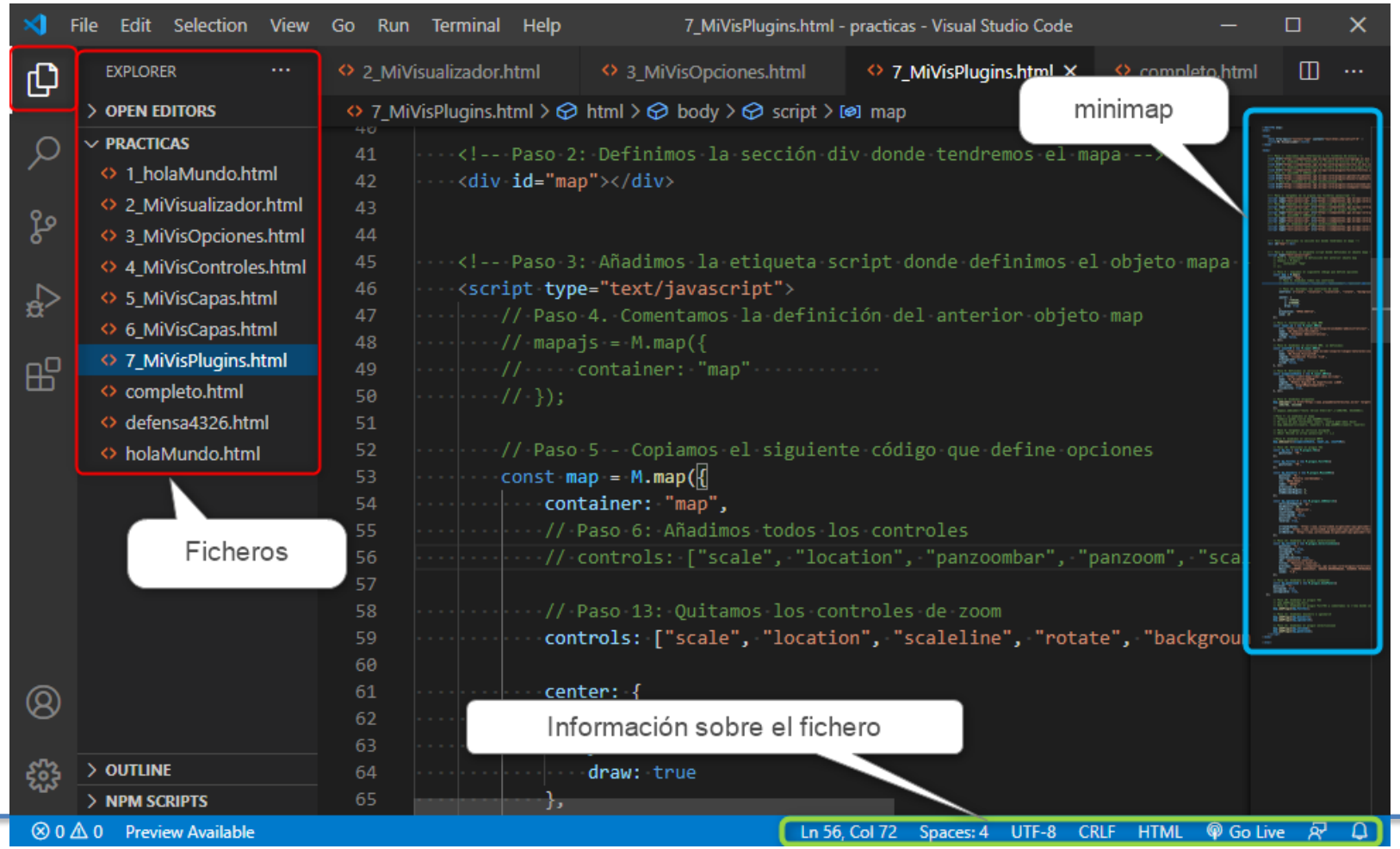
API JavaScript. Desarrollo de un visualizador personalizado



2020-11-25

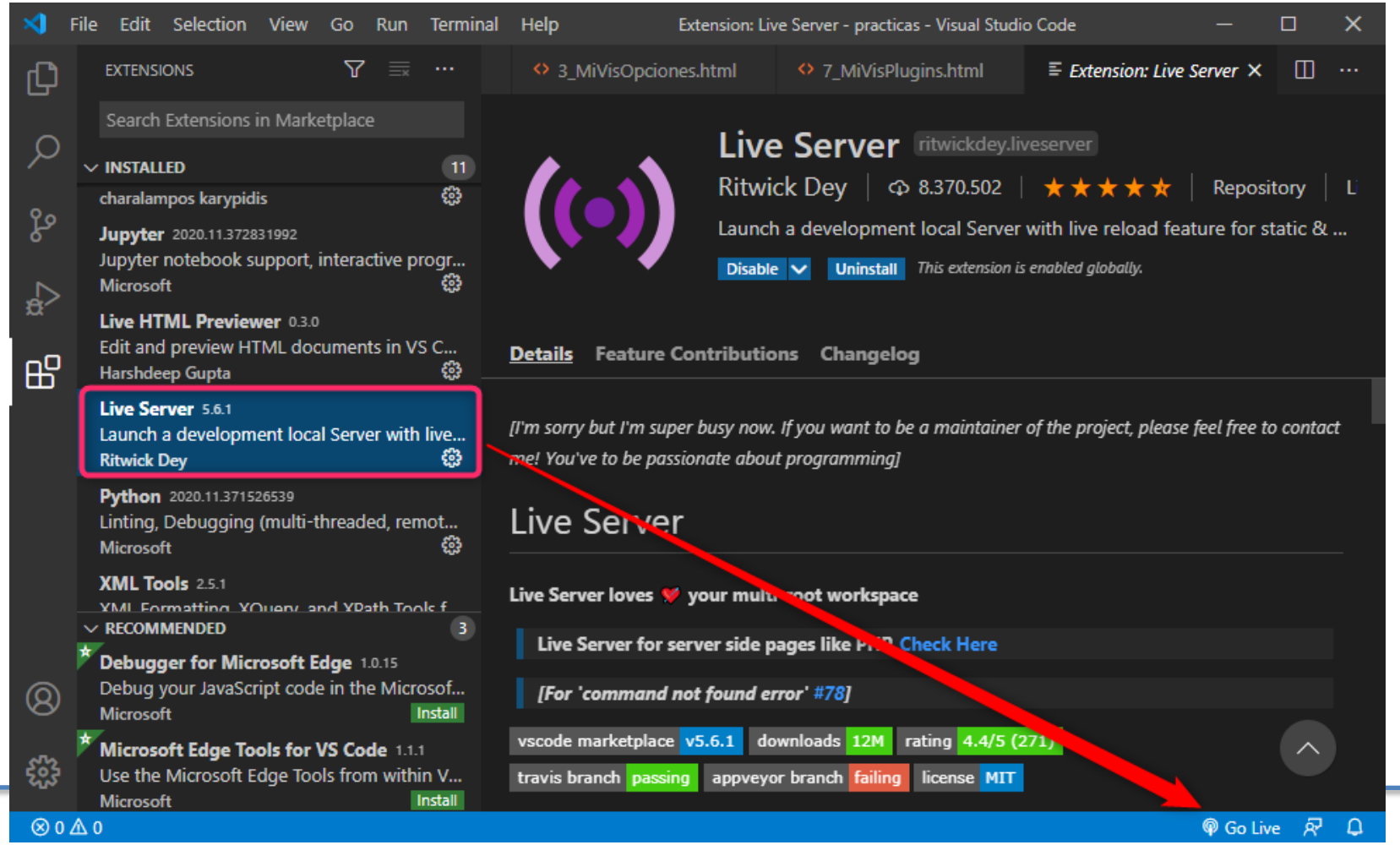
API JavaScript. Desarrollo de un visualizador personalizado

- Utilizaremos el editor de código Visual Studio Code.



API JavaScript. Desarrollo de un visualizador personalizado

- Instalar el complemento Live Server.



API JavaScript. Desarrollo de un visualizador personalizado

- Copiamos a nuestro PC desde el directorio compartido la carpeta *"resultados"* que contiene los diferentes pasos del código del visualizador.
- Una vez la tenemos en nuestro PC, arrastramos la carpeta *"resultados"* sobre una ventana limpia de Visual Studio Code.
- Nos aparecerá el código en la pantalla.
- Compilamos
- Se abrirá en nuestro navegador predeterminado una URL tal como http://127.0.0.1:5500/visor.html

API del CNIG. Práctica

Necesitaremos:

- El editor de código *Visual Studio Code* junto con el plugin *Live Server*.
- Acceso al repositorio de la Wiki:
  https://github.com/IGN-CNIG/API-CNIG/wiki
- URL de acceso a la API-CNIG:
  https://componentes.ign.es/api-core/

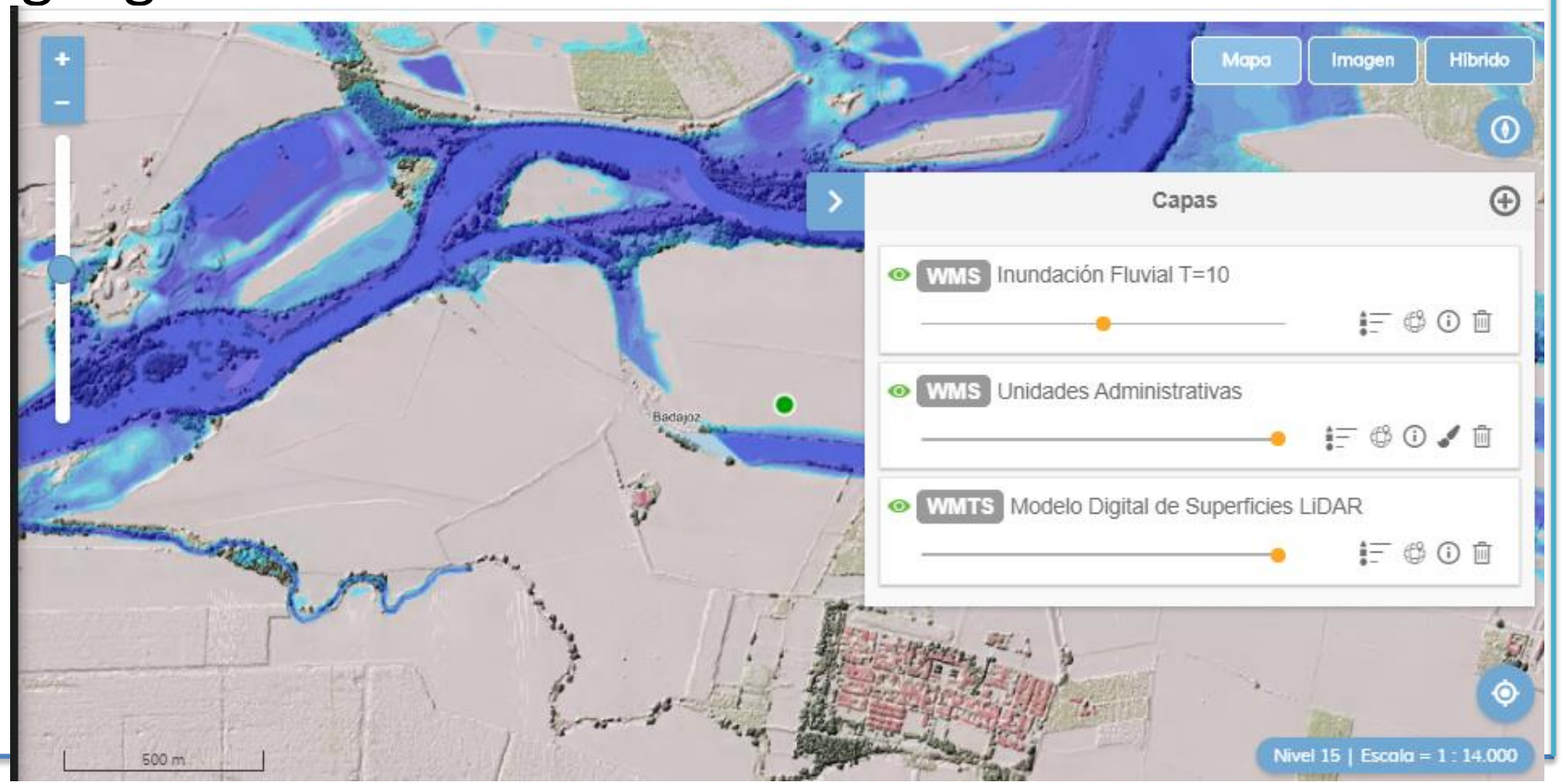
Soporte:

- Presentación de Power Point con el guion de las prácticas.
- Ficheros de resultado de cada uno de los pasos.

Objetivo

Crear un visualizador personalizado con:

- Capas de información geográfica de su interés.
- Herramientas personalizadas.



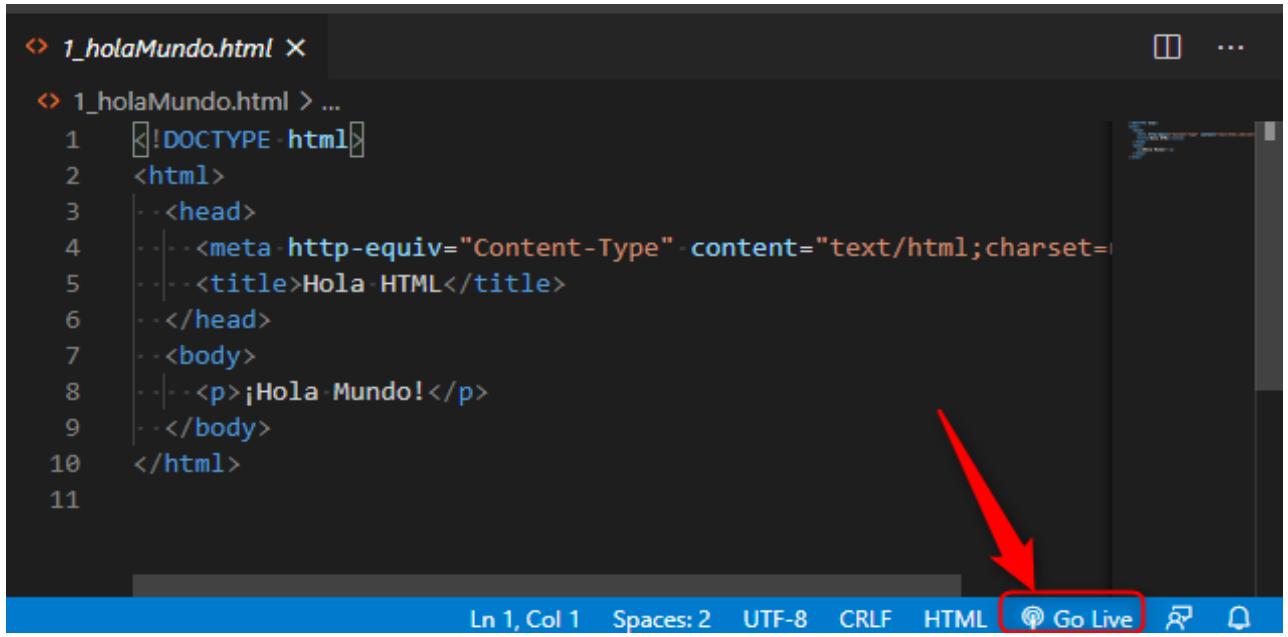
Creación de una página HTML

- Copia en tu equipo desde la carpeta compartida la carpeta “practicas”.
- Esta carpeta será el espacio de trabajo a lo largo de la práctica.
- La carpeta ya contiene un fichero con el nombre: “1_holaMundo.html”, con el siguiente contenido:

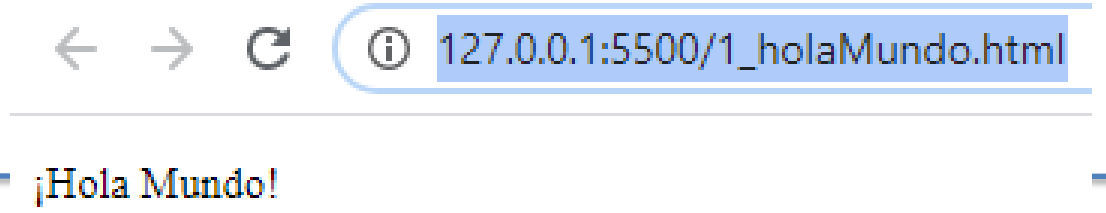
```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    <title>Hola HTML</title>
  </head>
  <body>
    <p>¡Hola Mundo!</p>
  </body>
</html>
```

Creación de una página HTML

- Abre Visual Studio Code
- Arrastra la carpeta prácticas a Visual Studio Code
- Abre el fichero "1_holaMundo.html"
- Ejecuta el código en Visual Studio Code.



Resultado:
Se abrirá una ventana del navegador por defecto con la siguiente la página



Visualizador básico

- Antes de empezar, crearemos un nuevo fichero al que llamaremos "2_MiVisualizador.html" con el contenido del anterior.

- Del nuevo fichero eliminamos y modificamos el contenido no necesario, dejando sólo la estructura de la página HTML.

  Para ello:

  - Modificamos el título de nuestra página y ponemos "Mi Visualizador".
  - Eliminamos la etiqueta `<p>¡Hola Mundo!</p>` de la sección `<body>`.

Visualizador básico

- Seguiremos los pasos descritos en la wiki de la API-CNIG primeros pasos. https://github.com/IGN-CNIG/API-CNIG/wiki/Primeros-pasos

1 **Paso 1** : Copia dentro del *<head>* el siguiente código.

1.- Importa los siguientes ficheros en tu html:

```
<!-- fichero estilos -->
<link href="https://componentes.ign.es/api-core/assets/css/apiign.ol.min.css" rel="stylesheet" />

<!-- ficheros javascript -->
<script type="text/javascript" src="https://componentes.ign.es/api-core/js/apiign.ol.min.js"></script>
<script type="text/javascript" src="https://componentes.ign.es/api-core/js/configuration.js"></script>
```

Visualizador básico

2 **Paso 2:** Copia dentro del *<body>*

2.- Añade un elemento *div* donde quieras que se muestre el mapa, asignándole un *id*:

```
<div id="map"></div>
```

3 **Paso 3:** Copia dentro del *<body>* después del * *.

3.- Crea el mapa mediante la siguiente instrucción javascript, especificando el *id* del *div* que hemos creado:

```
<script type="text/javascript">
mapajs = M.map({
   container:"map"
});
</script>
```

- Guardamos los cambios y lanzamos Live Server en Visual Studio Code.

Visualizador básico

Resultado:
Observamos nuestro visualizador que se habrá abierto en el navegador y hacemos zoom sobre la península.

Como no hemos especificado qué datos o capas queremos ver, por defecto la API-CNIG muestra el servicio del IGN-Base. Durante la sesión vamos a ir añadiendo otras capas personalizadas y herramientas, siguiendo los primeros pasos.

Personalizar el visualizador

La personalización del mapa que queremos construir recae en las siguientes categorías:

• **Opciones:** Permiten establecer características concretas del mapa o del área a visualizar, como son el centrado en un punto, nivel de zool inicial, sistema de referencia, etc. Consulta las Opciones existentes

• **Controles:** Son las herramientas que incluye el mapa y que permiten trabajar con él, tales como herramientas de medición, de coordenadas, etc. Se dividen en dos categorías: básicos y plugins. Más información sobre los Controles Básicos

• **Capas:** Son los datos a mostrar, y conforman el mapa que se dibujará. Actualmente, acepta los principales orígenes de datos OGC: WMS, WFS, KML, WMTS, y Geojson. Más información sobre las Capas

Personalizar el visualizador. Opciones

Opciones

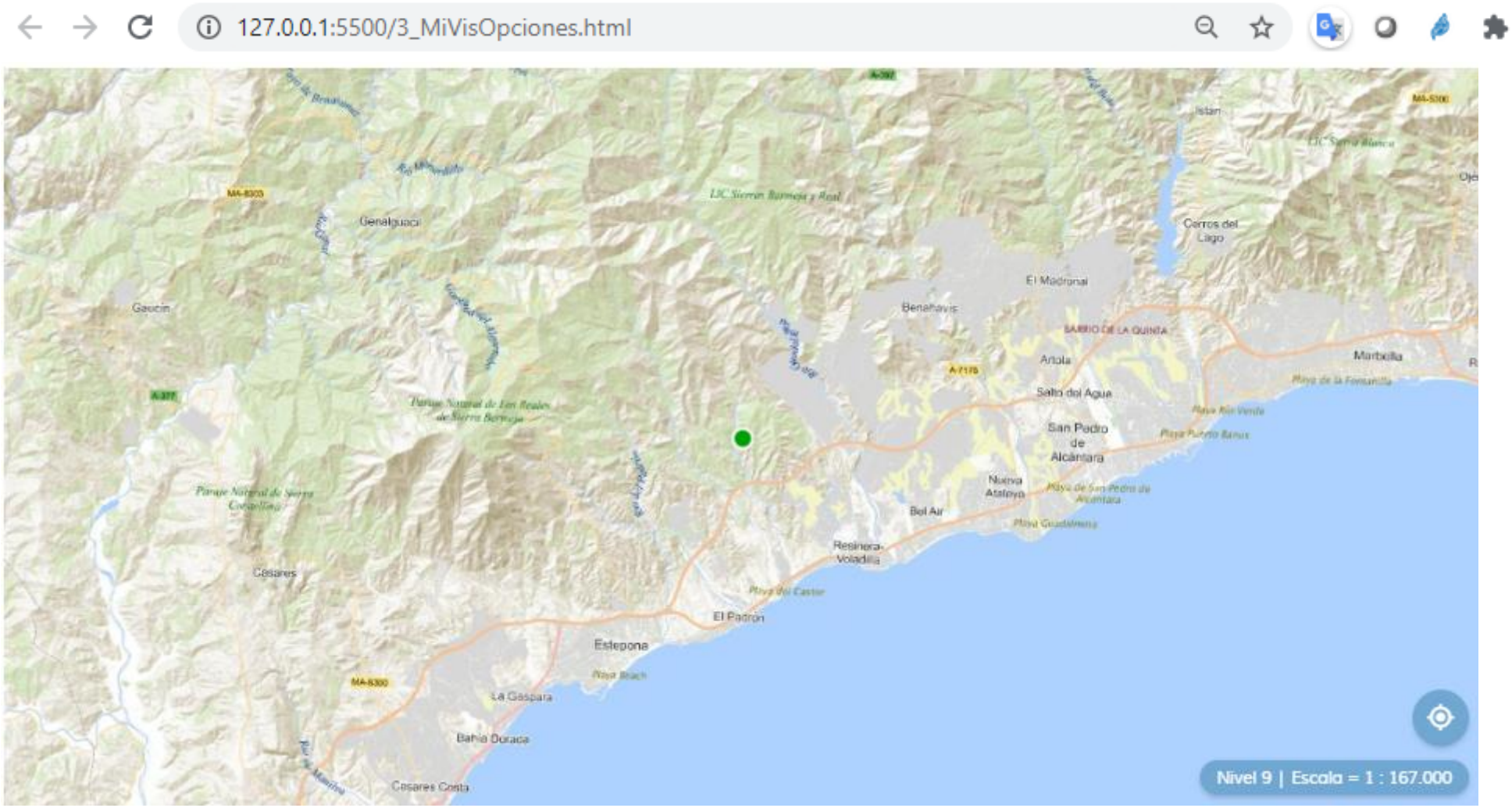
- Seguiremos las instrucciones de la wiki, apartado Opciones: https://github.com/IGN-CNIG/API-CNIG/wiki/Opciones
- Creamos un nuevo fichero con el contenido del que tenemos o guardamos el actual con el nombre "3_MiVisOpciones.html"

4 **Paso 4:** Pon atención en que el objeto Map ya está definido.

5 **Paso 5:** Copiar el contenido que nos ofrece el ejemplo de la wiki: *"En la construcción del objeto Mapa"*.

Personalizar el visualizador. Opciones

- **Opciones**
- **Resultado:**



2020-11-25

Personalizar el visualizador. Opciones

Opciones : Centrar visualizador en una ubicación

Abre el nuevo visualizador de Iberpix: https://www.ign.es/iberpix y desplázate en el mapa con las opciones de zoom hasta un punto donde te gustaría estar disfrutando de unos días de descanso.

- Cambia el sistema de coordenadas que se visualizan en la parte inferior-central del visualizador a EPSG:3857 y apunta las coordenadas.

- Modifica el código de tu visualizador **centrándolo en estas coordenadas y haciendo un zoom más cercano al terreno**. **Ten en cuenta el cambio en el sistema de coordenadas.**

Personalizar el visualizador. Opciones

6 **Paso 6**: Añade una etiqueta de texto con la propiedad **`addlabel`** tal y como se muestra en la wiki.

- **Resultado:**

Personalizar el visualizador. Controles

Controles

- La API dispone de una serie de **controles básicos**, que se especifican mediante el parámetro '*controls*' en el constructor del mapa, o llamando al método *'addControls'* del mismo.

- Podemos encontrarlo en Github en: https://github.com/IGN-CNIG/API-CNIG/wiki/Opciones

Personalizar el visualizador. Controles

Controles

- Al igual que en pasos anteriores creamos un nuevo fichero al que llamaremos "4_MiVisControles.html"

- Presta atención en el código, en el script de construcción del mapa y verás como ya contamos con 2 controles. Observa que están entre comillas y separados por comas.

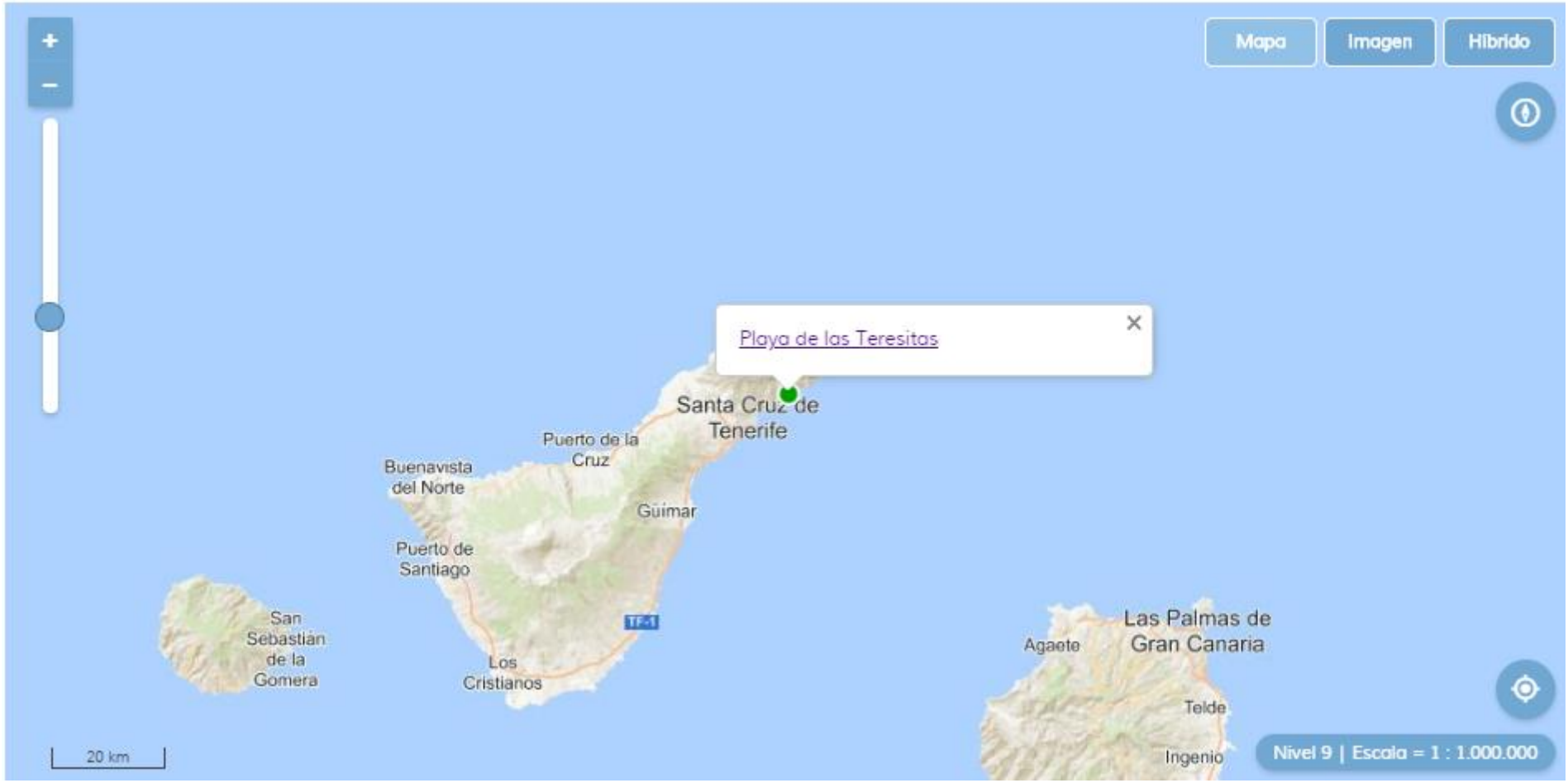
6

- **Paso 6-bis.** Navega en la wiki hasta la sección controles básicos y añade todos los controles básicos disponibles a tu visualizador.

Personalizar el visualizador. Controles

Controles

Resultado:



2020-11-25

Personalizar el visualizador. Capas

Capas

- Las capas son para API CNIG los proveedores de información espacial que mostrar en el mapa. Existen diversos formatos tanto ráster como vectoriales bajo los que ofrecer esas capas.

- Podemos encontrarlo en Github en: https://github.com/IGN-CNIG/API-CNIG/wiki/Capas

Personalizar el visualizador. Capas

Capas. Servicios de Mapas. WMS

- Guardamos nuestro fichero como “5_MiVisCapas.html” y continuamos trabajando sobre este.

- Navegamos en la Wiki hasta el apartado WMS https://github.com/IGN-CNIG/API-CNIG/wiki/WMS

- Vamos a cargar una capa del servicio de unidades administrativas del IGN tal y como se indica en el método 2 de la página utilizando la función addLayers.

Personalizar el visualizador. Capas

Capas. Servicios de Mapas. WMS

- Ten en cuenta que en el ejemplo el objeto Mapa se llama Map en lugar de Mapjs como se llama en nuestro visualizador.
  - Por comodidad, vamos a cambiar el nombre del objeto en nuestro mapa a Map.

7

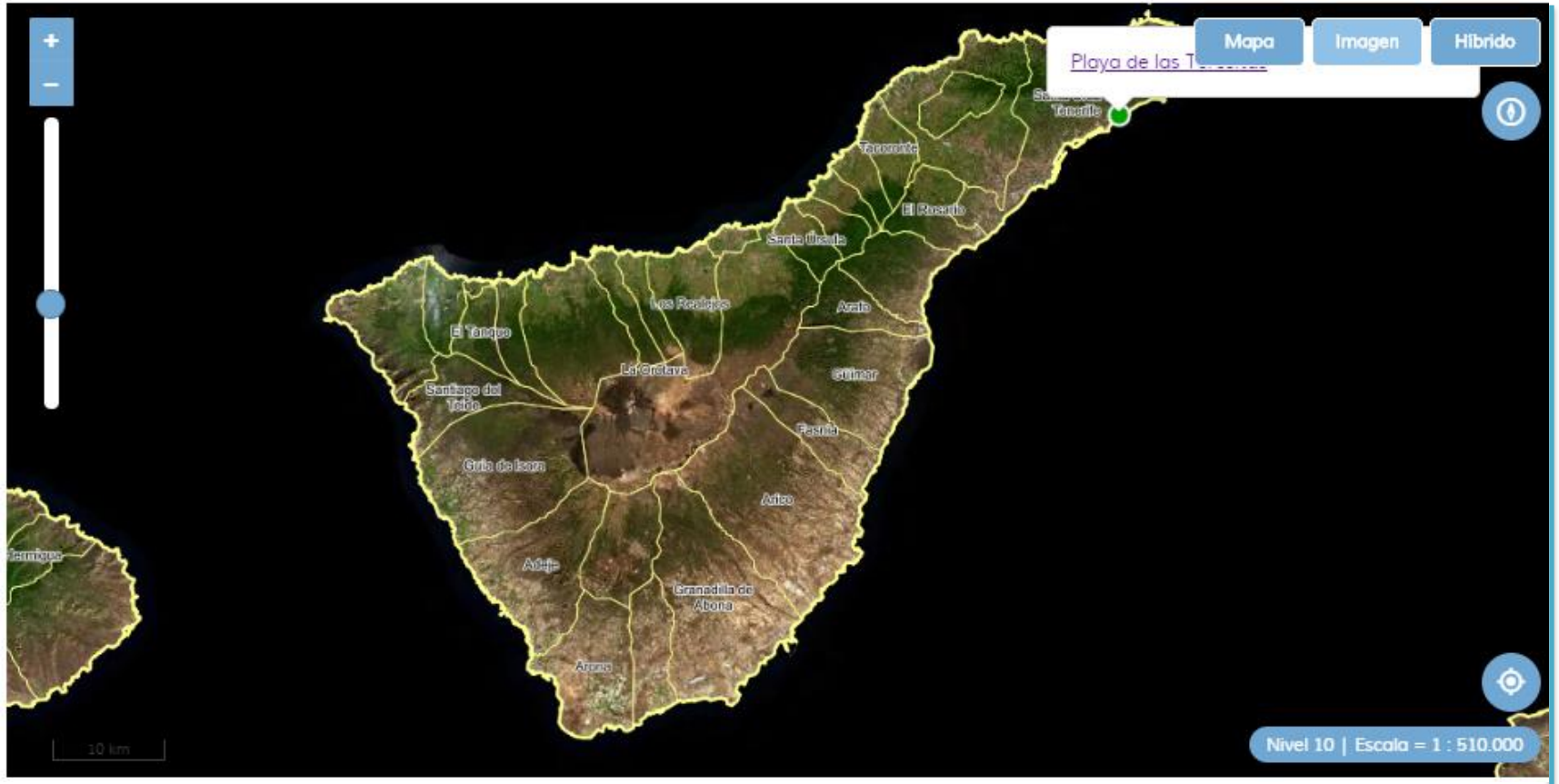
- **Paso 7:** Como vamos a añadir más capas conviene que el identificador de esta capa no se llame “`Layer`” y le ponemos otro más descriptivo como “`Layer_ua`” o “`UnidadAdministrativa`”.

Personalizar el visualizador. Capas

Capas. Servicios de Mapas. WMS

Resultado:

Podemos ver las unidades administrativas en amarillo, por encima de la capa de fondo imagen.



Personalizar el visualizador. Capas

Capas

Resultado:

Prueba ahora a hacer clic dentro de cualquier área administrativa y podrás comprobar la utilidad de uno de los controles que hemos añadido anteriormente, el getfeatureinfo, que nos proporciona información del servicio WMS que hemos cargado.

Personalizar el visualizador. Capas

Capas. Servicios de Mapas. WMS

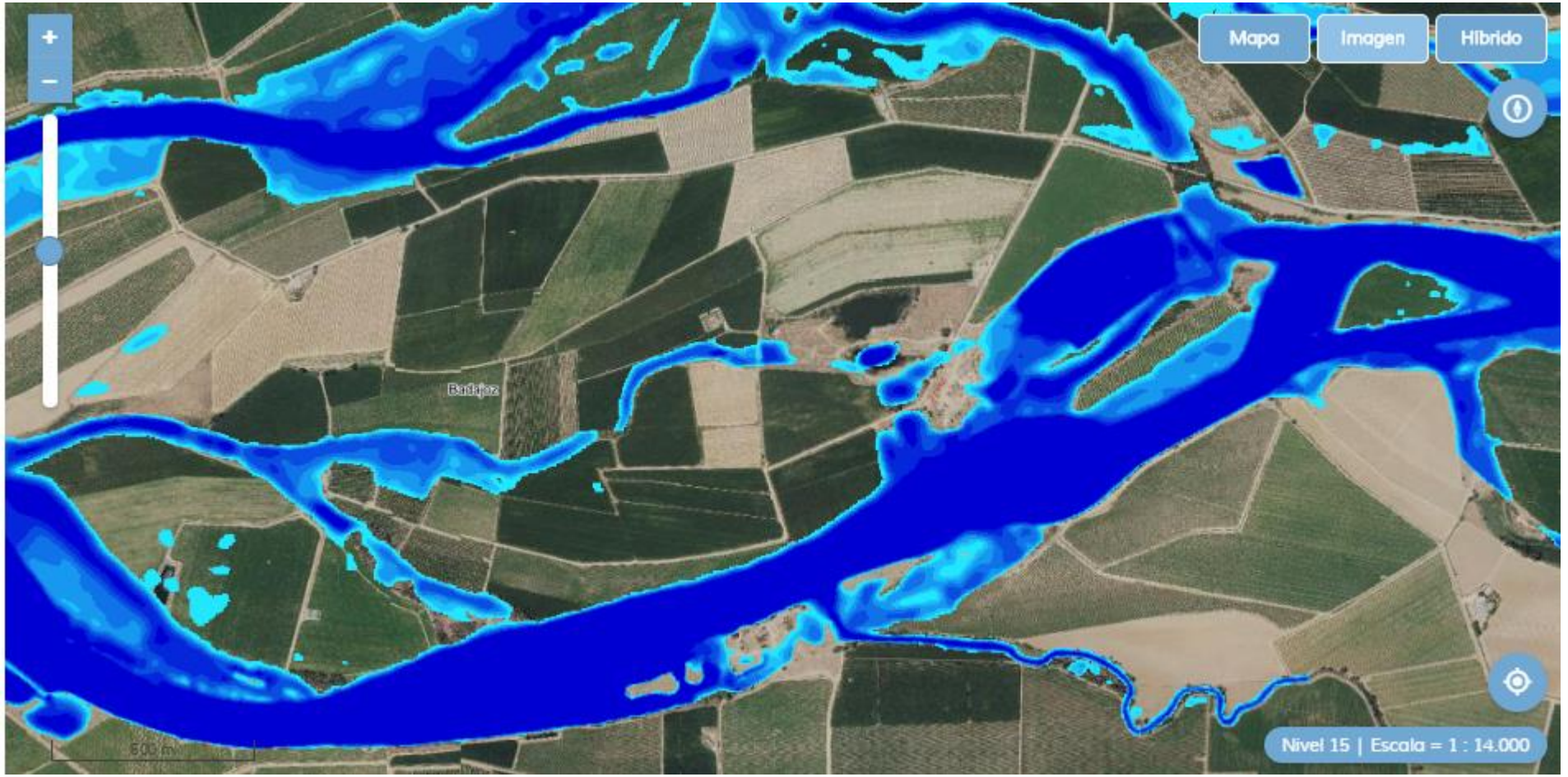
8 Paso 8: Escoge un servicio de mapas WMS que conozcas o bien dirígete al directorio de servicios de la IDEE y selecciona uno https://www.idee.es/web/guest/directorio-de-servicios.

- Asegúrate de que el servicio al menos responde a la operación GetCapabilities, o mejor aún, asegúrate de que puedes cargarlo en el visualizador de la IDEE https://www.idee.es/visualizador/ o en QGIS.
  - **Intenta cargarlo en tu visualizador.**
  - Ten en cuenta que necesitarás cierta información sobre el contenido del servicio que solo el propio servicio te puede proporcionar:
    - Nos referimos al fichero de capacidades que puedes obtener con las siguientes operaciones sobre el servicio WMS: https://servicios.idee.es/wms-inspire/riesgos-naturales/inundaciones?**request=GetCapabilities&service=WMS&Version=1.3.0**

Personalizar el visualizador. Capas

Capas. Servicios de Mapas. WMS

Resultado:



Personalizar el visualizador. Capas

Capas. Servicios de Mapas. WMTS

- Guardamos nuestro fichero como "6_MiVisCapas.html" y continuamos trabajando sobre este.

- Navegamos en la Wiki hasta el apartado WMTS
  https://github.com/IGN-CNIG/API-CNIG/wiki/WMTS

9

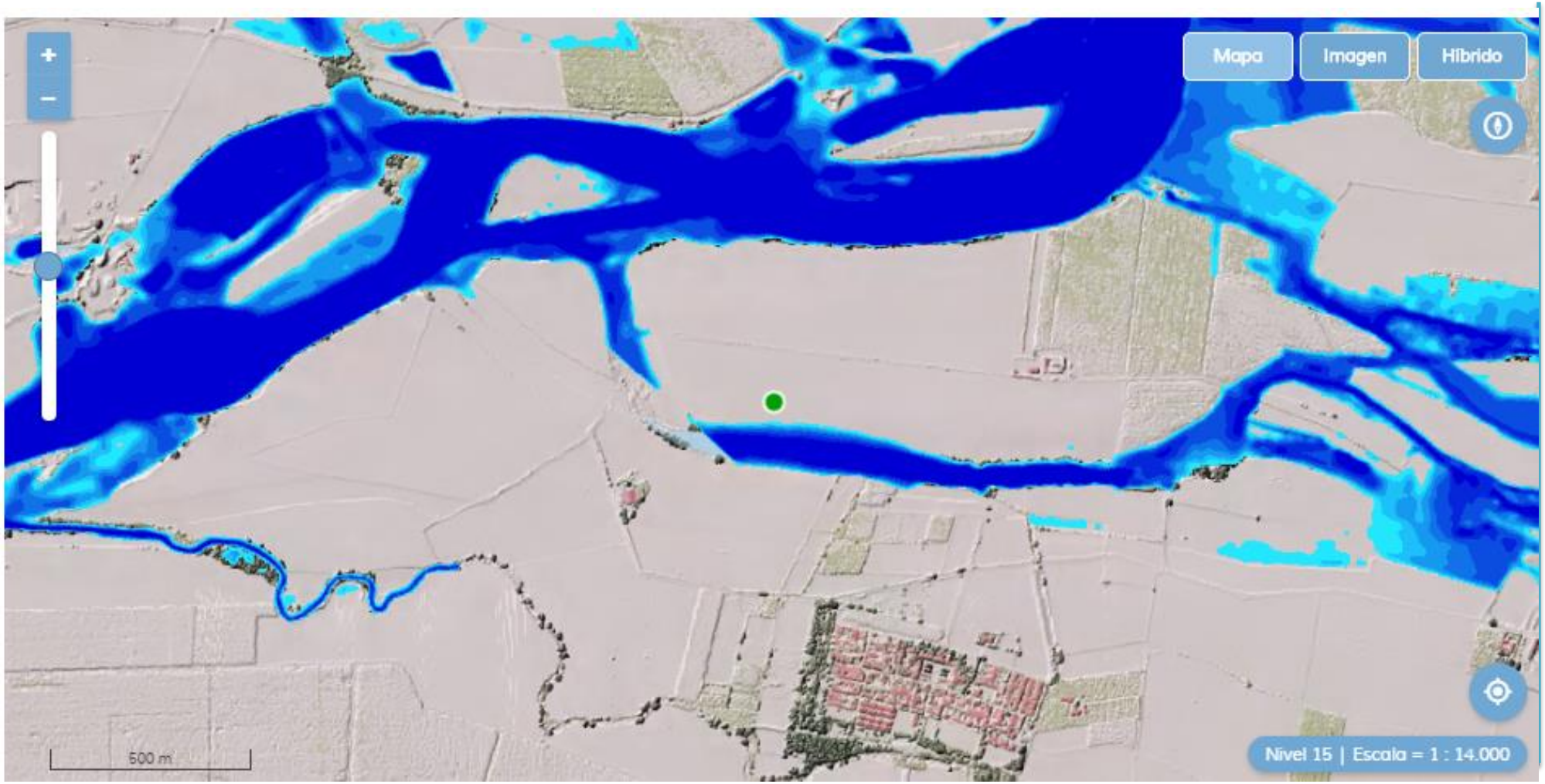
- **Paso 9:** De los diferentes ejemplos, por comodidad y porque comparte el método *AddLayers* que hemos venido usando, utilizaremos la opción 3.

- Al igual que hemos hecho con el servicio WMS escoge un servicio WMTS y añádelo a tu visualizador

Personalizar el visualizador. Capas

Capas. Servicios de Mapas. WMTS

Resultado:



2020-11-25

Plugins

Los **plugins** o componentes son funcionalidades que ofrecen utilidades específicas que, a diferencia de los controles básicos, suelen usarse en escenarios concretos, por lo que se separan del core o núcleo principal de la API para no penalizar con su descarga a quien no los necesite.

Plugins

- Guardamos nuestro fichero como “7_MiVisPlugins.html” y continuamos.
- Navegamos en la Wiki hasta el apartado WMS https://github.com/IGN-CNIG/API-CNIG/wiki/Plugins
- Como podemos ver, son muchos los plugins o funcionalidades que tenemos. Vamos a empezar por añadir un plugin muy útil, el TOC o tabla de contenidos.
- Navegamos y comprobamos que tiene las siguientes dependencias:

Dependencias

- toc.ol.min.js
- toc.ol.min.css

```
<link href="../../plugins/toc/toc.ol.min.css" rel="stylesheet" />
<script type="text/javascript" src="../../plugins/toc/toc.ol.min.js"></script>
```

Plugins

- Cada plugin tiene sus propios ficheros de recursos que deben vincularse junto a los del CORE.
- Si nos fijamos, vemos que en el ejemplo las rutas son relativas. Al estar trabajando en local y no en la estructura definitiva que tendría dentro del servidor de aplicaciones, tenemos que hacer que las rutas sean absolutas.
  - La API se encuentra disponible en la URL:

    https://componentes.ign.es/api-core/
  - y en este caso, sustituyendo `../../` por esta, tendremos las rutas absolutas:

    https://componentes.ign.es/api-core/plugins/toc/toc.ol.min.css
    https://componentes.ign.es/api-core/plugins/toc/toc.ol.min.js

Plugins

10

- **Paso 10.** Copiamos en el código las anteriores referencias.

- Añadimos al visualizador el código del plugin de forma que quede arriba a la derecha siguiendo las instrucciones de “Ejemplos de uso” del plugin

  https://github.com/IGN-CNIG/API-CNIG/blob/master/api-ign-js/src/plugins/toc/README.md

Plugins

Resultado:

Plugins

11

- **Paso 11.** Sustituye el anterior plugin por el plugin *FullToc*

- Busca la documentación en : https://github.com/IGN-CNIG/API-CNIG/wiki/Plugins

Plugins

Resultado:

2020-11-25

Plugins

12

- **Paso 12.** Vamos a añadir algún plugin más, por ejemplo:
  - MouseSRS
  - IGNSearch

- Busca la documentación en : https://github.com/IGN-CNIG/API-CNIG/wiki/Plugins

Plugins

Resultado:

Plugins

13

- **Paso 13**. Carga los siguientes plugins:
    - Selectionzooms
    - Zoompanels

- Busca la documentación en : https://github.com/IGN-CNIG/API-CNIG/wiki/Plugins

13

- **Paso 13.** Elimina los controles relativos al zoom.

Plugins

Resultado:

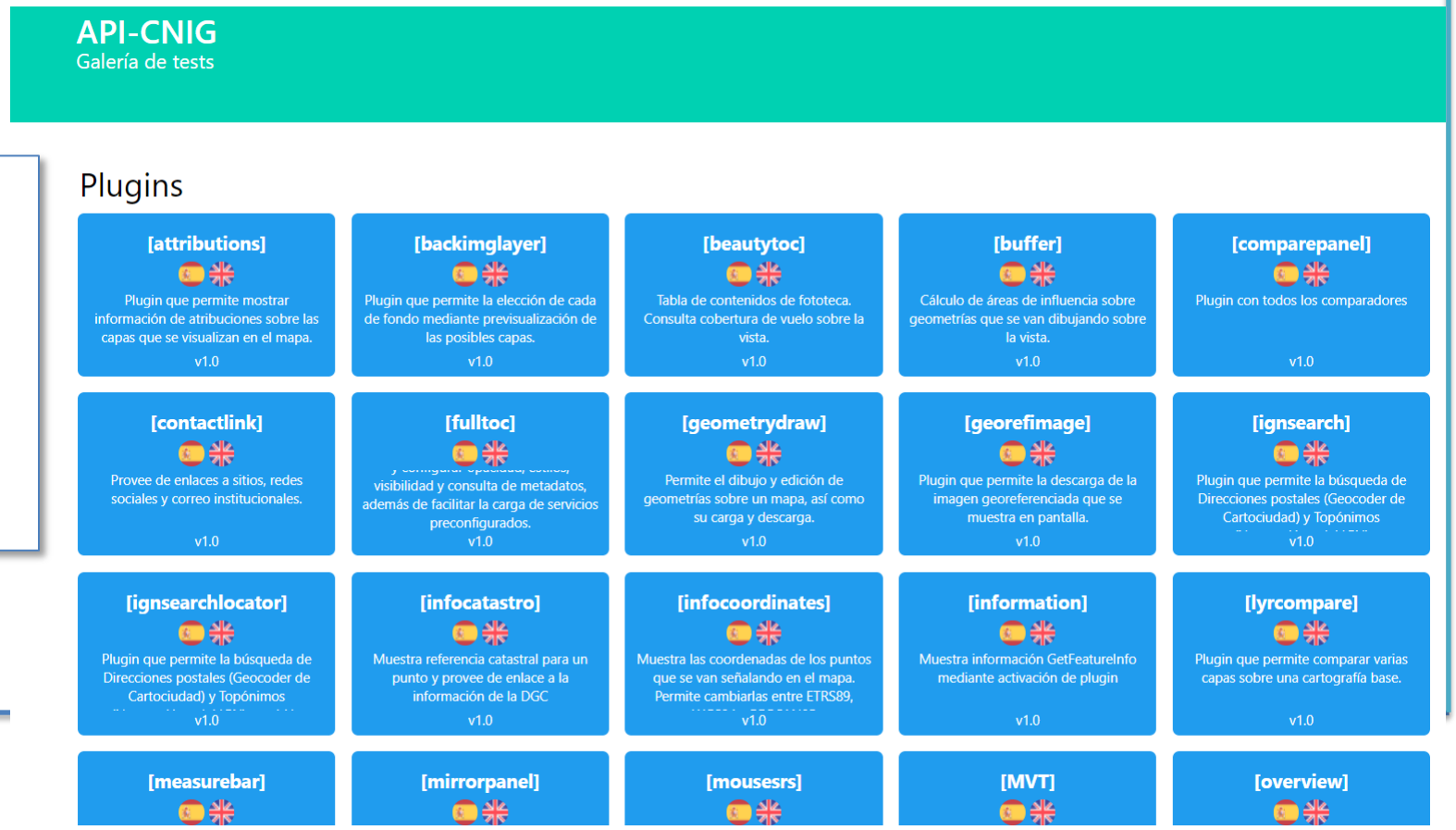
2020-11-25

Plugins

Por último navega por la página de Test de los diferentes Plugins, pruébalos y trata de completar tu visualizador con alguno de ellos:
http://componentes.cnig.es/api-core/test.html

Ten en cuenta que es un entorno de prueba donde puedes encontrar algunos plugins que no están depurados.

Los que ofrecen más confianza son los que están listados en la wiki.



API-CNIG 3.0 API-Rest

- **Ejemplos API – REST**

  https://github.com/IGN-CNIG/API-CNIG/wiki/API-REST

  - 1.- Visualizador básico para saber si está funcionando
    http://componentes.cnig.es/api-core/
  - 2.- Visualizador con un control
    http://componentes.cnig.es/api-core/?controls=panzoom
  - 3.- Visualizador con todos los controles
    http://componentes.cnig.es/api-core/?controls=panzoom,scale,scaleline,panzoombar,panzoom,location,getfeatureinfo,rotate,backgroundlayers

API-CNIG 3.0 API-Rest

- **Ejemplos API - REST**
  - 4.- Visualizador con controles y parámetros de localización y proyección:

    https://componentes.cnig.es/api-core/?
    controls=panzoom,scale,scaleline,panzoom,location, rotate,backgroundlayers
    &center=-118114.81,4397718.76
    &zoom=6
    &projection=EPSG:3857*m

API-CNIG 3.0 API-Rest

- **Ejemplos API - REST**
  - 5.-Visualizador con controles, parámetros de localización y capas procedentes de servicios:

    https://componentes.cnig.es/api-core/?controls=panzoom,scale,scaleline,panzoom,location,rotate,backgroundlayers&center=-118114.81,4397718.76&zoom=6&projection=EPSG:3857*m&layers=WMTS*http://wmts-mapa-lidar.idee.es/lidar*EL.GridCoverageDSM*GoogleMapsCompatible*Modelo%20Digital%20de%20Superficies%20LiDAR*true*image/png*true*true*true

API-CNIG 3.0 API-Rest

- **Ejemplos API - REST**
  - 6.- Visualizador con controles, parámetros de localización, capas procedentes de servicios y plugins:
    https://componentes.cnig.es/api-core/?controls=panzoom,scale,scaleline,panzoom,location,rotate,backgroundlayers&center=-118114.81,4397718.76&zoom=6&projection=EPSG:3857*m&plugins=vectors,measurebar,overviewmap&layers=WMTS*http://wmts-mapa-lidar.idee.es/lidar*EL.GridCoverageDSM*GoogleMapsCompatible*Modelo%20Digital%20de%20Superficies%20LiDAR*true*image/png*true*true*true

API-CNIG 3.0 API-Rest

- **Ejemplos API - REST**
  - 7.- Visualizador con controles, parámetros de localización, capas procedentes de servicios más vectorial KML y plugins:
    - https://componentes.cnig.es/api-core/?controls=panzoom,scale,scaleline,panzoom,location,rotate,backgroundlayers&center=-118114.81,4397718.76&zoom=6&projection=EPSG:3857*m&plugins=vectors,measurebar,overviewmap&layers=WMTS*https://www.ign.es/wmts/ign-base?*IGNBaseTodo*GoogleMapsCompatible*IGNBaseTodo*false*image/png*true*true*true,WMS*Limite%20administrativo*http://www.ign.es/wms-inspire/unidades-administrativas?*AU.AdministrativeBoundary*true*false**1.1.1*true*true*true,WMS*Unidad%20administrativa*http://www.ign.es/wms-inspire/unidades-administrativas?*AU.AdministrativeUnit*true*true**1.3.0*true*true*true,KML*Delegaciones*https://www.ign.es/web/resources/delegaciones/delegacionesIGN.kml*false*false*true

Gracias

José María García Malmierca
jmgmalmierca@mitma.es

Cecilia Poyatos Hernández
cecilia.poyatos@cnig.es