XI Jornadas Ibéricas de Infraestruturas de Datos Espaciales

26 | 30 octubre 2020

- Herramienta para el desarrollo de plugins
- Estructura de directorio
- Fachada e implementación
- Plantillas
- Api.json

- Desarrollo Mapea SIGC o API IGN
- Añadir plugin a un visor
- Plugins desarrollados
- Eventos
- Buenas prácticas
- Documentación

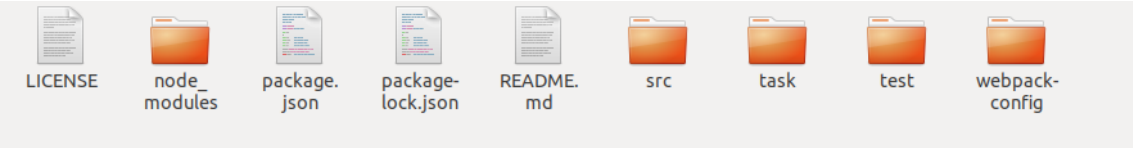
Herramienta para el desarrollo de plugins

- Mediante NPM podemos instalar dicha herramienta:
- $ npm install -g mapea-create-plugin
- Y así poder crear nuestro primera estructura de plugin:
- $ mapea-create-plugin

```
[Mapea-plugins] [WARN] Plugin name cannot be empty or contains spaces.
[Mapea-plugins] What is the name of your plugin?: miPrimerPlugin
?  Choose Mapea version:  5.2.0
```

Herramienta para el desarrollo de plugins

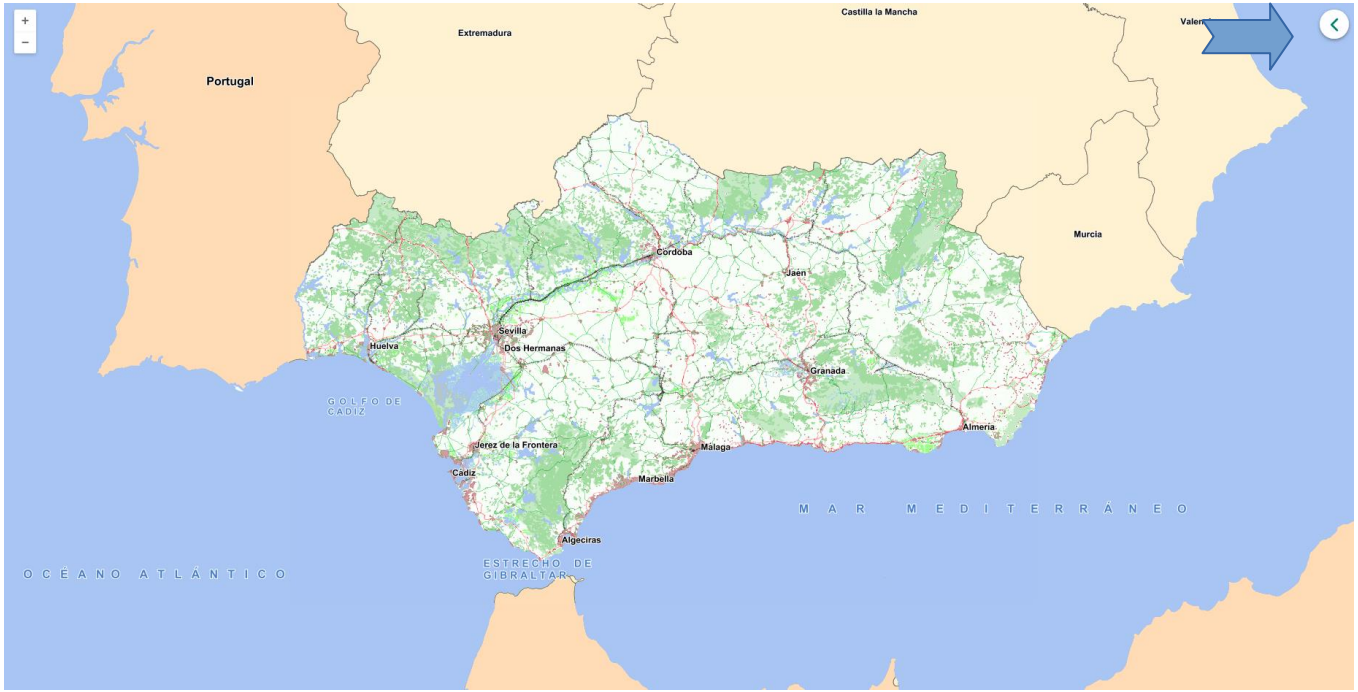
- Creará la siguiente estructura de directorios dentro de la carpeta del plugin:



- Para poder testear el plugin ejecutaremos el siguiente comando dentro del directorio del mismo:

- miPrimerPlugin$ npm start

Herramienta para el desarrollo de plugins



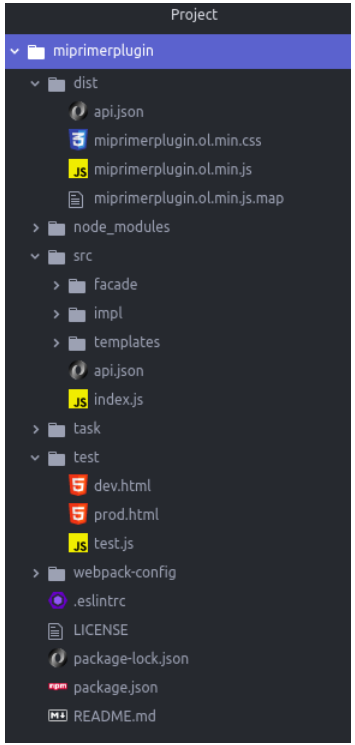
Herramienta para el desarrollo de plugins

Herramienta para el desarrollo de plugins

- Validar el código con ESLint:
  miPrimerPlugin$ npm run check
- Y compila y minimiza el plugin generando un único fichero CSS y JS para usarlos en un visor
- miPrimerPlugin$ npm run build

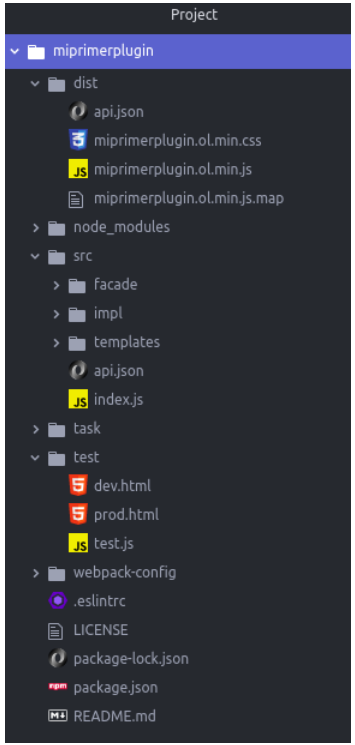
Estructura de directorio

- dist: contine los ficheros minificados y compilados
- Test: tanto de desarrollo como de producción
- Dentro de src se encuentra el código del plugin que se divide en:
  - facade
  - impl
  - templates
  - api.json



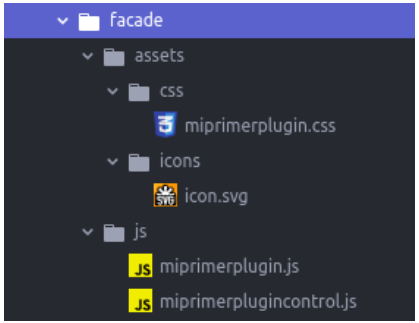
Fachada e implementación

- Objetivo: independizar desarrollos funcionales (fachada) de las librerías de mapas (implementación)
- Fachada: común a todas las implementaciones. Operaciones independientes de la librería de mapas
- Implementación: Openlayers, leaflet, etc



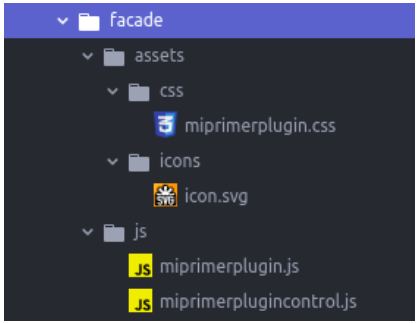
Fachada

- css: hoja de estilos para el plugin
- icons: iconos
- miprimerplugin.js
  Contiene los siguientes métodos:
  - constructor: recibir parámetros de configuración, ...
  - addTo: se llama al añadirlo al mapa y es el método principal ya que recibe como parámetro el mapa, crea el panel, ...
  - destroy: se llama al eliminar el plugin del mapa
  - getAPIRest: para poder llamar al plugin por la API Rest



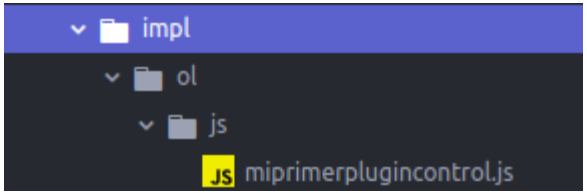
Fachada

- miprimerplugincontrol.js
  Contiene los siguientes métodos:
  - constructor: crea la implementación
  - createView: crea la vista del control llamando a la plantilla
  - activate: activa el plugin
  - deactivate: desactiva el plugin



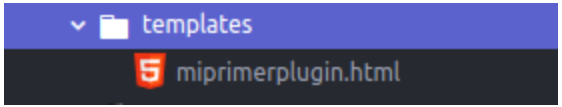
Implementación

- miprimerplugincontrol.js
  Contiene los siguientes métodos:
  - constructor
  - addTo: recibe el mapa y el HTML del control definido en el createView
  - activateClick
  - deactivateClick



Plantillas

- Es habitual que un plugin necesite interfaz para que el usuario pueda interactuar con ella
- Código HTML
- Usa las clases CSS definidas en la fachada



api.json

- En este fichero se encuentra la configuración necesaria para el correcto funcionamiento del plugin, así como los metadatos del mismo:
  - Se especifica el separador usado para indicar cada parámetro de configuración
  - Los parámetros que permite el plugin
  - Ficheros JS y CSS
  - Metadatos
- Es imprescindible para usar el plugin mediante la API REST

Desarrollo Mapea SIGC o API IGN

- En los test del plugin se configuran a los recursos del core a los que queremos apuntar. Ejemplo:
  - SIGC:
    - CSS: http://mapea4-sigc.juntadeandalucia.es/assets/css/mapea-5.2.0.ol.min.css
    - JS: http://mapea4-sigc.juntadeandalucia.es/js/mapea-5.2.0.ol.min.js
    http://mapea4-sigc.juntadeandalucia.es/js/configuration-5.2.0.js
  - API IGN:
    - CSS: https://componentes.cnig.es/api-core/assets/css/apiign.ol.min.css
    - JS: https://componentes.cnig.es/api-core/js/apiign.ol.min.js
    https://componentes.cnig.es/api-core/js/configuration.js

Desarrollo Mapea SIGC o API IGN

- SIGC

- API IGN

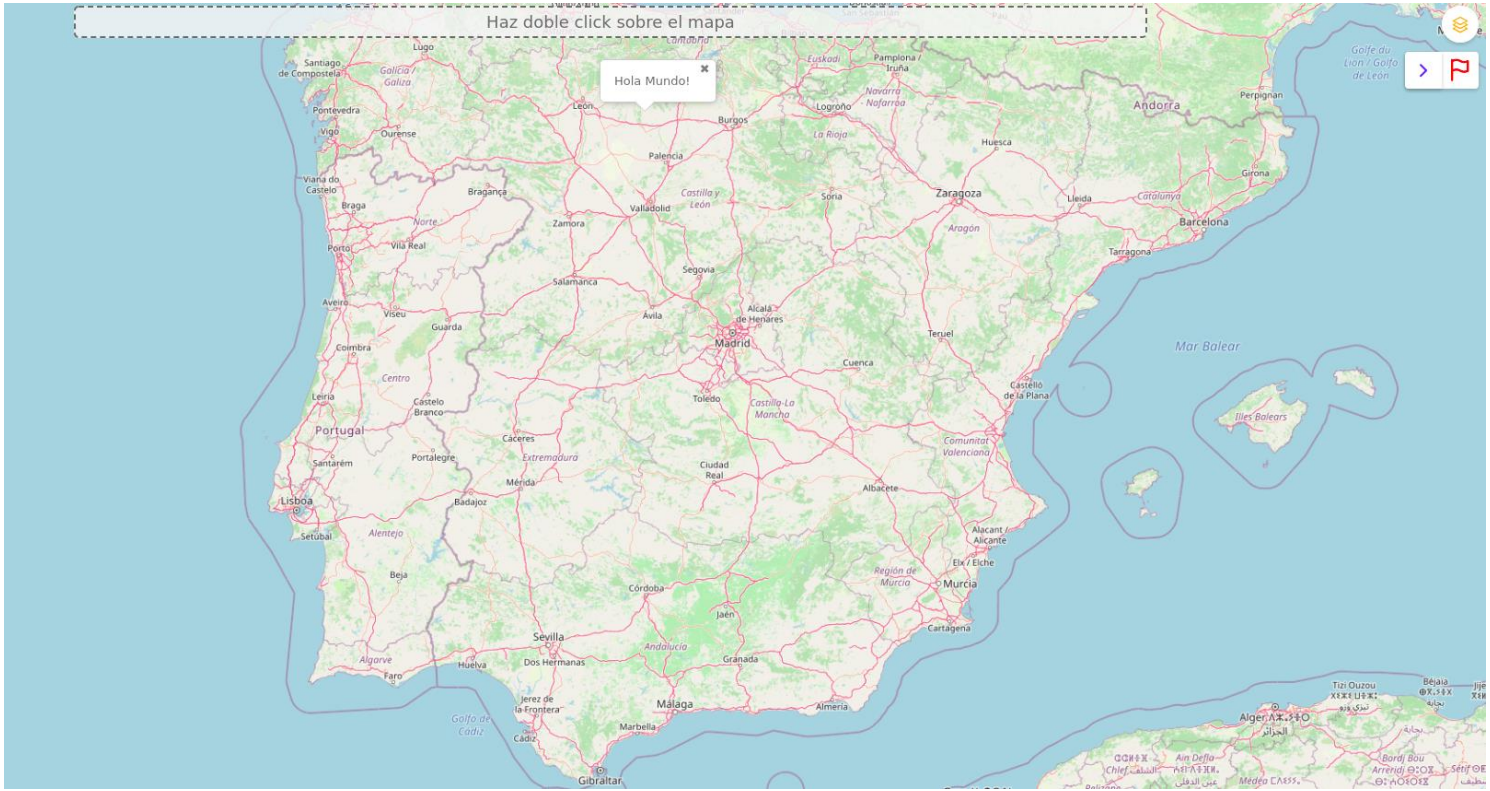
Añadir plugin a un visor

- Creando un fichero HTML con los recursos necesarios tanto de mapas como de plugins podemos obtener un visor con el plugin creado:

```
<html>
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0">
      <meta http-equiv="X-UA-Compatible" content="IE=edge" />
      <meta name="mapea" content="yes">
      <link href="http://mapea4-sigc.juntadeandalucia.es/assets/css/mapea-5.2.0.ol.min.css" rel="stylesheet" />
      <style rel="stylesheet">
         html,
         body {
         margin: 0;
         padding: 0;
         height: 100%;
         overflow: hidden;
         }
      </style>
      <link href="/plugins/css/miprimerplugin.ol.min.css" rel="stylesheet" />
   </head>
   <body>
      <div id="map" class="container"></div>
      <!-- core -->
      <script type="text/javascript" src="http://mapea4-sigc.juntadeandalucia.es/js/mapea-5.2.0.ol.min.js"></script>
      <script type="text/javascript" src="http://mapea4-sigc.juntadeandalucia.es/js/configuration-5.2.0.js"></script>
      <!-- plugins -->
      <script type="text/javascript" src="./plugins/js/miprimerplugin.ol.min.js"></script>
      <script type="application/javascript">
         var map = M.map({
         "container": "map",
         "controls": ["layerswitcher", "navtoolbar"],
         "layers": ["OSM"],
         "projection": "EPSG:3857*m"
         });

         const mp = new M.plugin.MiPrimerPlugin();
         map.addPlugin(mp);
      </script>
   </body>
</html>
```

Añadir plugin a un visor



Desarrollo Mapea SIGC o API IGN

- Dicho plugin también puede ser añadido al core tanto de SIGC como de API IGN y poder ser usado mediante la API REST. Ejemplos:
  - SIGC:
    - Buscador de direcciones
    - Tabla de atributos
  - API IGN:
    - Dibujar geometrías
    - Historial de vistas

Plugins desarrollados

- Infocatastro: Muestra referencia catastral para un punto y provee de enlace a la información de la DGC.

```
activate() {
  if (this.activated) {
    this.deactivate();
  } else {
    this.facadeMap_.on(M.evt.CLICK, this.buildUrl_, this);
    this.activated = true;
    this.element_.querySelector('#m-infocatastro-btn').classList.add('activated');
  }
}
```

```
buildUrl_(evt) {
  const options = {
    jsonp: true,
  };

  const srs = this.facadeMap_.getProjection().code;
  M.remote.get(this.catastroWMS, {
    SRS: srs,
    Coordenada_X: evt.coord[0],
    Coordenada_Y: evt.coord[1],
  }).then((res) => {
    this.showInfoFromURL_(res, evt.coord);
  }, options);
}
```

```
showInfoFromURL_(response, coordinates) {
  if ((response.code === 200) && (response.error === false)) {
    const infos = [];
    const info = response.text;
    const formatedInfo = this.formatInfo_(info);
    infos.push(formatedInfo);

    const tab = {
      icon: 'g-cartografia-pin',
      title: this.POPUP_TITLE,
      content: infos.join(''),
    };

    let popup = this.facadeMap_.getPopup();

    if (M.utils.isNullOrEmpty(popup)) {
      popup = new M.Popup();
      popup.addTab(tab);
      this.facadeMap_.addPopup(popup, coordinates);
```

Eventos

- ADDED_KML, ADDED_WFS, ADDED_WMS...
- ADDED_TO_PANEL
- CHANGE_PROJ
- CHANGE_ZOOM
- HOVER_FEATURES
- ...

Plugins desarrollados

- Measurebar: Herramienta de medición de áreas y distancias.

```
formatGeometry(geometry) {
  ...
  if (codeProj === 'EPSG:3857') {
    length = Math.round(ol.sphere.getLength(geometry) * 100) / 100;
  } else if (unitsProj === 'd') {
    const coordinates = geometry.getCoordinates();
    for (let i = 0, ii = coordinates.length - 1; i < ii; i += 1) {
      length += ol.sphere.getDistance(ol.proj.transform(coordinates[i], codeProj, 'EPSG:4326'), ol.proj.transform(coordinates[i + 1], codeProj, 'EPSG:4326'));
    }
  } else {
    length = Math.round(geometry.getLength() * 100) / 100;
  }
  ...
}
```

Buenas prácticas

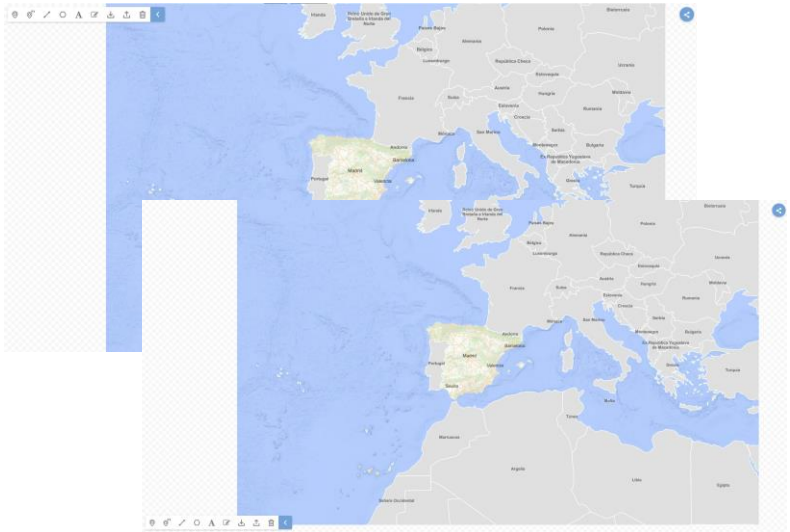
- Hacer uso de los métodos disponibles en M.utils (isNullOrEmpty, iString, …)

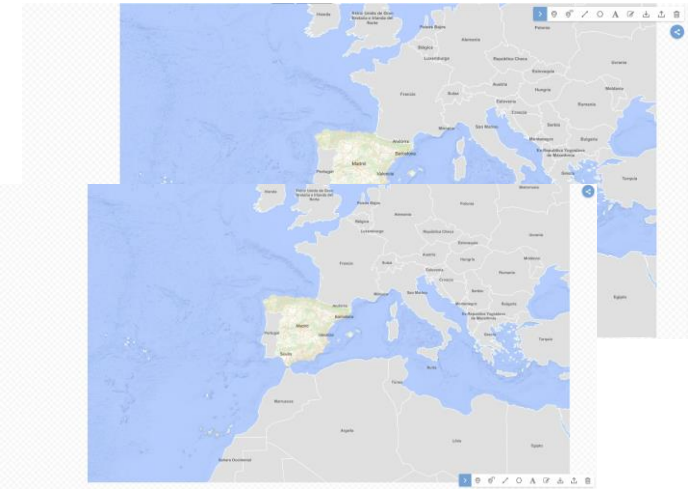
```
> M.utils.isNullOrEmpty("")
<- true
> M.utils.isString(2)
<- false
```

- No hacer uso de librerías de mapas en la fachada.
  Ejemplo: no usar OL en la fachada del plugin

Buenas prácticas

- Asegurar la correcta visualización de los plugins en las diferentes posiciones:





Buenas prácticas

- Incorporar valores por defecto para los parámetros del plugin:
  M.map({"container":"map"}).addPlugin(new M.plugin.XYLocator({"position":"TL"}))
  M.map({"container":"map"}).addPlugin(new M.plugin.XYLocator({}))

- No sobrescribir estilos del core en los plugins, ni de otros plugins

- Indicar en el fichero README ejemplos de uso, dependencias, descripción, parámetros...

Documentación

- https://github.com/sigcorporativo-ja/Mapea4/wiki
- https://www.npmjs.com/package/mapea-create-plugin/v/1.0.4
- https://github.com/IGN-CNIG/API-CNIG/wiki

XI Jornadas Ibéricas de Infraestruturas de Datos Espaciales

26 | 30 octubre 2020