Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

GIS WEB APLICADO A GESTIÓN DE FLOTAS EN EXPLOTACIONES MINERAS

Trabajo Fin de Máster

Autor: María Suárez Álvarez
Tutor Académico: Cesar Castañón Fernández
julio, 2019

Contenido

Lista de Ilustraciones

1. Resumen

Este trabajo parte de la necesidad de la elaboración de un visor Web a partir de la aplicación MWS4Mining “Sistemas de Control de Flotas Mineras”. Esta aplicación está solamente creada para el sistema Android, por lo que con este proyecto se pretende llegar a todos los usuarios que no tengan disponibilidad de acceso a sistemas Android, mediante un visor web.

En concreto se hará un Gis web de movimientos en tiempo real de la maquinaria minera que se encuentra trabajando en la mina de Riotinto (Huelva).

El resultado final es un código html del visor Web, que introducido en un servidor web, muestra los datos proporcionados por la maquinaria en tiempo real.

Abstract

This work is based on the need to develop a Web viewer for the MWS4Mining application "Mining Fleets Control Systems". This application is only created for the Android system, so this project is intended to reach all users who do not have access to Android systems, through a web viewer.

Specifically, a real-time movement web Gis will be made of the mining machinery that is working in the Riotinto mine (Huelva).

The final result is a html code of the Web viewer, which entered in a web server, shows the data provided by the machinery in real time.

2. Introducción

Los visores Web han llegado a nuestras vidas para quedarse, no solo profesionalmente, sino en nuestro día a día. Google Maps es un gran ejemplo. Esta aplicación usa una cartografía propia que a su vez es usada por otras aplicaciones como OpenLayers, que a su vez contiene su propia cartografía. A continuación, se introducirá el concepto de visor Web y sus orígenes, así como la utilización de la biblioteca OpenLayers entre varios temas más para poder comprender este proyecto.

Visor Web

Origen de los SIG:

El desarrollo de los SIG (Sistemas de Información Geográfica) va de la mano con el desarrollo de la informática, desde sus orígenes allá por el 1960 – 1970, donde se tenían que utilizar voluminosos ordenadores que sólo permitían hacer borradores y no poseían herramientas para hacer una digitalización o almacenamiento alguno.

Antes de la era informática, la necesitad de estos sistemas ya existía, creando diversas soluciones a problemas concretos como la de 1782 del cartógrafo Louis-Alexandre Berthier en la que representó el movimiento de las tropas de la batalla de Yorktown mediante planos superpuestos a una cartografía base. Otro ejemplo es el del doctor John Snow (1854), el cual demostró con un análisis geográfico la distribución de las muertes por cólera, que resultaron estar ligados a los manantiales contaminados de la zona.

En los inicios del desarrollo informático del pasado siglo surgen los primeros SIG creados por norteamericanos (*HarvardLaboratory for Computer Graphics-LGC)*, canadienses (*Canada Geographic Information Systems*-CGIS) y británicos (*Experimental Cartography Unit*-ECU). Estos SIG tenían bastantes problemas respecto a la estructura y organización de la base de datos.

Los SIG fueron modernizándose poco a poco llegando a la representación ráster, celebrándose en 1970, en Canadá, el *Primer Simposio Internacional de Sistemas de Información Geográfica*. Años después ya se empezaron a publicar revistas y a distribuir información en Internet de los SIG. En esa época fue fundada la empresa

Environmental Systems Research Institute (ESRI), pionera en su día hasta el día de hoy.

Este origen de la herramienta ESRI y el aumento del uso de ordenadores personales facilitaron un mayor acceso a los SIG para todo aquel que lo quisiera utilizar para su investigación o trabajo.

En 1985, surge el primer SIG de acceso libre, *Geographic Resources Analysis Support System* (GRASS), siendo en la actualidad un referente para los programas libres de SIG.

A día de hoy, los SIG no se limitan para el uso profesional, sino que ha entrado en nuestra vida cotidiana. Ese es el ejemplo de Google Maps o Google Earth. Estos SIG, acercan al usuario no profesional al mundo SIG.

Origen de los Visores WEB:

World Wide Web (WWW) nace en los finales de 1989, pero no es hasta el 1993 cuando empieza a utilizarse para usos del tipo SIG. Aparece *Xerox PARC*, el primer servidor de mapas, que será sustituido en 1997 por *Mapserver* hasta la actualidad. El primer mapa digital que se desarrolla es el *Atlas Nacional de Canadá*, en 1994. En el 2005 se incorpora Google Maps que permite desarrollar nuevas aplicaciones sobre los servicios ofrecidos por ellos mismos.

OpenLayers:

Es la primera biblioteca Javascript de código abierto que se usa para la visualización de mapas interactivos en los navegadores Web. Este visor es compatible con los WMS (Web Map Service) y WFS (Web Feature Service). Al ser ejecutado con Javascript, los datos se descargan directamente desde el navegador, por lo que no es necesario meterlos el servidor.

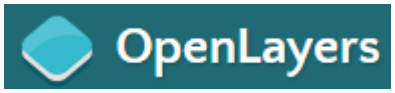


Ilustración 1 Logotipo OpenLayers

En OpenLayers se puede sobreponer una capa sobre otra, añadir puntos o polígonos, los cuales se pueden definir gracias a su API (Application programming interface).

Para funcionar con OpenLayers, se puede trabajar en modo hosting que significa que no tenemos que bajar ninguna librería ni almacenarla en un servidor, o se puede descargar la librería, guardándola en un servidor y accediendo a ella.

Una de las ventajas del OpenLayers es que no se requiere ninguna licencia. Otra es la cantidad de ejemplos que tiene en su página web.

Application programming interface (API):

Interfaz de programación de aplicaciones, es un conjunto de rutinas (permiten llevar a cabo una acción mediante pasos ya definidos por las mismas) que ofrece la biblioteca de OpenLayers para acceder a distintas fuentes de información cartográficas en la red, como los Web Map Services (ejemplo, PNOA o catastro), formatos vectoriales, mapas de Google Maps, etc. En definitiva, son funciones ya elaboradas para facilitar el manejo a los usuarios finales.

Ejemplos:

ol/proj.transform: transforma las coordenadas origen a una nueva proyección. Devuelve unas nuevas coordenadas que no modifican las originales.

ol/map: crea un nuevo mapa

MWS4Mining

En este trabajo se va a crear un visor Web a partir de una aplicación ya creada, el MWS4Mining. Esta aplicación (app) nace del enfoque “Soluciones simples para problemas concretos” y fue creada por las empresas Sadim y RPTec.

Ilustración 2 Logotipo MWS4Mining

La empresa *SADIM Ingeniería* (Sociedad Asturiana de Diversificación Minera), perteneciente al grupo HUNOSA. Por un lado, tiene un área de ingeniería, especializada en la dirección y ejecución de proyectos de minería, industria, ingeniería civil y museística y por otro lado, un área de desarrollo de sistemas de información con personal especializado en áreas con el GIS, desarrollo web y dispositivos móviles.

Ilustración 3 Logotipo empresa SADIM

A su vez, *RPTec* (España) (Recursos y Proyectos Técnicos) está constituido por un equipo de técnicos y profesionales en minería, geología e informática, con gran experiencia en la investigación de yacimientos, recursos geológicos y reservas mineras, diseño y planificación de explotaciones mineras y desarrollo.

Ilustración 4 Logotipo RPTEC

Esta empresa es la responsable del software *RecMin*, un paquete completo de programas, diseñador para gestionar proyectos de investigación y explotación de recursos minerales y de amplia utilización en la industria minera.

Ilustración 5 Logotipo RecMin

La app se divide en tres apartados:

- Partes del trabajo: (MWS *Mobile Working Sheet*) consiste en la toma de datos del trabajo de campo de los trabajadores, evitando la utilización del papel. Cada máquina tiene un dispositivo móvil con la app instalada, en la cual se registra el posicionamiento en tiempo real de la maquinaria minera usando el GPS del dispositivo. Otra de las funcionalidades que tiene este apartado, es el de alertar cuando la maquinaria este en una zona/área prohibida asignada anteriormente para según que carga contenga el camión/máquina. Contiene también un formulario para el registro diario de los partes de trabajo diarios de los trabajadores. Todos estos datos son enviados en tiempo real mediante conexión WIFI o 4G al servidor donde se procesa y gestiona toda la información.

Ilustración 6 Control de acceso al trabajador

- Gestión: (MWSM *Mobile Working Sheet Mining*) se incorpora la herramienta de gestión mediante uso de RecMin. En este apartado se puede crear catálogos, es decir las modificaciones de las rutas, zonas de trabajo, materiales, etc. Se vinculan directamente con el componente

MWS de modo que el sistema de control es cerrado. También elaboran informes con los datos recopilados.

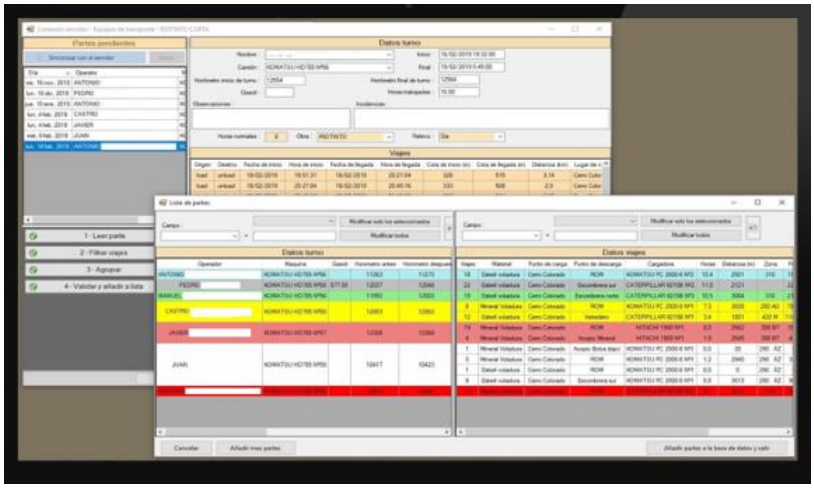


Ilustración 7 Elaboración de informes

- Supervisión: (MWS&S *Mobile Working Sheet and Supervision*) herramienta específica de supervisión dirigida a Gestores. Este apartado es para el encargado de obra para la supervisión de la flota. Consta de la visualización del posicionamiento en tiempo real sobre un mapa de los camiones y maquinaria. También es al que le llegan las alertas cuando una máquina/camión se encuentra en una zona de peligro o prohibida. Otra alerta posible, es cuando un camión va a una velocidad más alta que la permitida.

Ilustración 8 Imagen de la app con los equipos trabajando

Minas de Riotinto (Huelva)

Uno de los lugares donde se aplica este nuevo sistema es en la empresa Atalaya Mining que explota los recursos mineros de Riotinto.

Ilustración 9 Logotipo Atalaya Mining

Las Minas de Riotinto se explotan desde la Edad de Cobre, y está en el origen y auge del mítico reino de Tartessos, de la Edad del Hierro. Los recursos codiciados por los diferentes imperios que dominaron el sur de la península (Fenicios, Romanos, etc.) no fueron explotados de manera masiva hasta el siglo XVIII. El 1873, un empresario británico compra las minas al estado español propietario entonces, y funda la *Rio Tinto Company Limited*. Años después se construye la línea de ferrocarril entre Huelva y

Riotinto para el transporte de minerales hasta el puerto de Huelva. Con estos avances, se creó mucho empleo, por lo que la mano de obra procedía de todos los puntos de España y Portugal.

El aumento de la mina obligó a trasladar el pueblo Riotinto a El Valle, construyéndose viviendas inglesas para el personal inglés que trabajaba allí. Un dato anecdótico de esta Mina de Riotinto es que fue la cuna del futbol de España, el cual en 1889 se creó el actual Recreativo de Huelva.

Durante el periodo franquista, en 1954, las minas se nacionalizan, pero quedando la gestión bajo varias empresas que la siguen explotando. Con altibajos de producción, actualmente la mina se encuentra en activo contando con una flota potente de maquinaria, como 43 volquetes, varias retroexcavadoras, excavadoras, etc.

Ilustración 10 Mina de Riotinto

3. Objetivos

El objetivo principal de este proyecto es conseguir hacer un visor web, a través de la biblioteca de OpenLayers, que permita a los usuarios autorizados visualizar una web la cual ya tiene una aplicación en sistema Android.

Dentro de este visor, hay varios objetivos:

- Crear un archivo KML a partir de un archivo php.
- Visualizar el recorrido en formato KML de la maquinaria minera localizada.
- Actualizar la visualización de ese KML, actualizando para ello la ejecución del archivo php y la recarga del KML al visor.
- Mostrar la información que tiene el KML al pasar el ratón por encima.
- Mostrar a través de una función el tiempo que queda para actualizar el kml.
- Entrar en el visor con un usuario y contraseña para que no sea público.

Una vez concluido el proyecto, uno de los objetivos también será el conocimiento adquirido a lo largo del trabajo realizado.

4. Metodología

Para crear este visor Web, se utilizaron los lenguajes de programación, HTML (HyperTxt Markup Language), JavaScript, PHP utilizando para su edición el editor de texto NotePad++.

NotePad++

Es un editor de texto igual que el Bloc de notas simple, pero con la particularidad de que este soporta varios lenguajes de programación, (HTML, XML, PHP, JavaScript entre otros). Una de las características más importantes que tiene este editor es la forma de colorear las expresiones propias del lenguaje que se esté utilizando, como se muestra en la siguiente imagen:

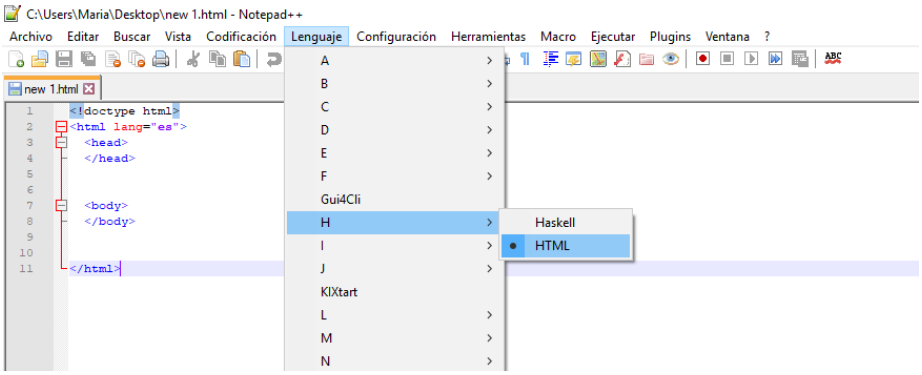


Ilustración 11 Ejemplo lenguaje HTML en NotePad++

También ayuda a tener controlados los cierres de paréntesis, llaves o corchetes, ya que tiene la capacidad de mostrar el correspondiente cierre o apertura cuando el cursor se coloca encima de alguno de estos. Ejemplo:

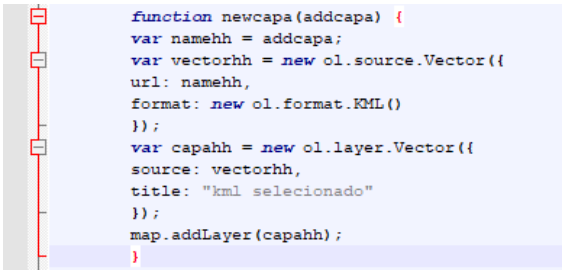


Ilustración 12 Paréntesis en rojo marcando el inicio y fin

Lenguaje HTML

Ilustración 13 Logotipo HTML 5

HTML, HyperText Markup Language o Lenguaje de Marcas de Hipertexto en español, es un lenguaje para la elaboración de páginas web. La sintaxis del lenguaje se basa en etiquetas de apertura y de cierre que definen lo que va entre ellas. Un ejemplo es las dos etiquetas <body> </body>, que indicarían que lo que hay entre esas dos etiquetas pertenecen a esa parte del código. Existen algunas etiquetas que no utilizan la etiqueta de cierre como la etiqueta <img> que define una imagen o
 que define un salto de línea.

Dentro de una etiqueta se pueden definir atributos propios como, por ejemplo:

```
<input type="button" class="more"  value="kml_area1"
onClick=newcapa("https://recmin.com/CF_SYL/RIOTINTO%20CORTA/resources/area1.kml
")>
```

Ilustración 14 Ejemplo etiqueta HTML

La etiqueta <*input*> tiene como atributos un *type* que tiene como valor "*button*" o *value*. Como se puede ver en los ejemplos, el propio NotePad++, diferencia las etiquetas, los atributos y los valores de los atributos por colores. El HTML se usa para la ordenación del contenido de la página, no el diseño de esta. Para ello existe el lenguaje CSS.

Lenguaje CSS

CSS (Cascading Style Sheets u "Hojas de estilo en cascada"), es un lenguaje para el diseño de páginas web. En este proyecto, al partir de la base de

OpenLayers, ya hay unos estilos definidos (*ol.css*). Para que el HTML “coja” esos estilos pondremos la siguiente línea en la cabecera (“head”) del código:

```
<link rel="stylesheet"
href="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/css/ol.css
" type="text/css">
```

Ilustración 15 Código estilos de OpenLayers

También se ha creado un archivo .css para pequeñas cosas que se mostraran a lo largo del proyecto. Para insertarlo en el HTML se hará como el anterior archivo .css, cambiando solo el origen del archivo que estará en la misma carpeta que el archivo final del proyecto.

Lenguaje JavaScript

Ilustración 16 Logotipo JavaScript

Es un lenguaje de programación interpretado para la creación de aplicaciones que se ejecutan en el navegador. Para insertar un código de JavaScript en HTML, se debe de utilizar la etiqueta <script> </script>, como se muestra en el ejemplo:

```
<script src="ol/ol.js"></script>
```

Ilustración 17 Ejemplo inserción script

JavaScript tiene la principal función de proporcionar efectos dinámicos a las páginas para hacerlas más atractivas a la vista. Este lenguaje no requiere clasificar el tipo de dato que se almacena. Para este proyecto se ha usado para recibir información del servidor.

Es un lenguaje muy complicado, ya que requiere muchas líneas de código para llegar al resultado final. Para ello existen librerías que simplifican el código como la librería jQuery.

Lenguaje PHP

PHP, *Hypertext Preprocessor*, lenguaje de programación interpretado que se almacena en un servidor Web. Para insertar el PHP en el HTML tiene una etiqueta menos común que las usadas normalmente:

```
<?php
   include("f_kml.php");
?>
```

Ilustración 18 PHP en HTML

En este proyecto se usa para conectar con la base de datos de las maquinas mineras ese archivo se llama *f_kml.php.*

Código HTML-PHP paso a paso

En esta parte, se va a explicar lo más significante que tiene el código, el cual se encuentra en el apartado de resultados.

Inicio:

Empezamos con el código que tiene OpenLayers en su página web https://openlayers.org/en/latest/doc/quickstart.html.

```
<!doctype html>
<html lang="en">
  <head>
    <link rel="stylesheet"
href="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/css/ol.css
" type="text/css">
    <style>
      .map {
        height: 400px;
        width: 100%;
      }
    </style>
    <script
src="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/build/ol.js"
></script>
```

```
  <title>OpenLayers example</title>
 </head>
 <body>
  <h2>My Map</h2>
  <div id="map" class="map"></div>
  <script type="text/javascript">
   var map = new ol.Map({
    target: 'map',
    layers: [
     new ol.layer.Tile({
      source: new ol.source.OSM()
     })
    ],
    view: new ol.View({
     center: ol.proj.fromLonLat([37.41, 8.82]),
     zoom: 4
    })
   });
  </script>
 </body>
</html>
```

Ilustración 19 Código inicial de OpenLayers

Así se mostraría en el navegador:

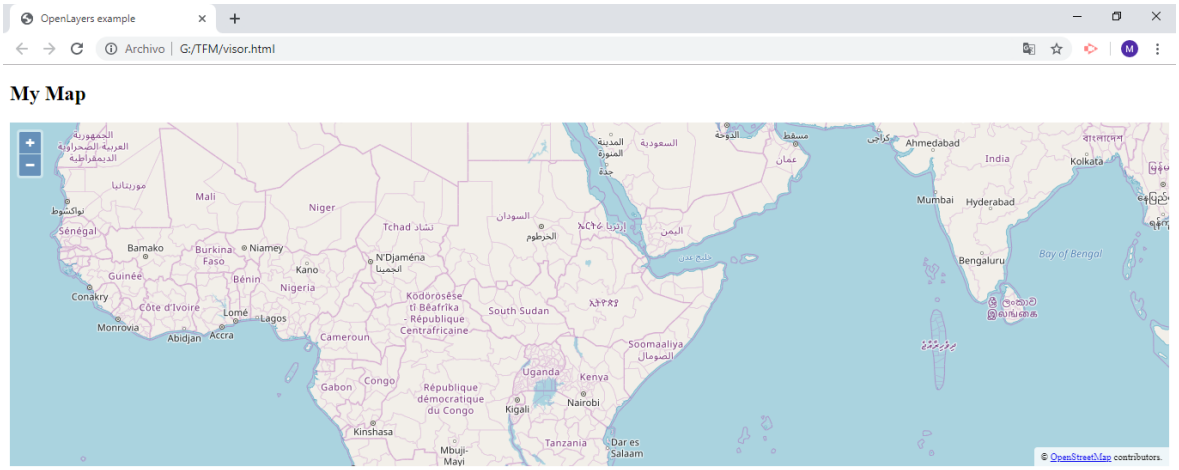


Ilustración 20 Visor básico de OpenLayers

1. ***Pequeños cambios del código básico***

El primer cambio es modificar las coordenadas para que se ajusten a la zona de la mina. Aumentar el zoom, es el siguiente cambio ya que no se necesita tener un zoom tan amplio.

En la etiqueta <style> nos encontramos con *.map*, que define las características visuales que tendrá el mapa. La altura está definida por pixeles, por

lo que tiene un tamaño fijo. Se debe cambiar por el 100% para que se haga en pantalla completa sin tener en cuenta el tamaño de la pantalla.

También cambiaremos el idioma en el que está definido (línea 2), el título que se muestra por “TFM Maria 2019”. Este título es el que aparecerá en la pestaña del navegador junto a un icono. Ese icono también se modifica con el siguiente código:

```
<link rel="icon" href="imagen/epm.png">
```

Ilustración 21 Logo de EPM en la pestaña del navegador

El cual se mostrará la imagen del logo de la escuela.

Estos pequeños cambios quedarían de la siguiente manera:

```
<!doctype html>
<html lang="es">
  <head>
    <link rel="stylesheet"
href="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/css/ol.css
" type="text/css">
     <link rel="icon" href="imagen/epm.png">
    <style>
     .map {
      height: 100%;
      width: 100%;
     }
    </style>
    <script
src="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/build/ol.js"
></script>
    <title> TFM Maria 2019 </title>
  </head>
  <body>
    <h2>Minas de Riotinto</h2>
    <div id="map" class="map"></div>
    <script type="text/javascript">
     var map = new ol.Map({
      target: 'map',
      layers: [
       new ol.layer.Tile({
        source: new ol.source.OSM()
       })
      ],
      view: new ol.View({
       center: ol.proj.fromLonLat([-6.58, 37.7005]),
       zoom: 14.7
      })
     });
    </script>
  </body>
</html>
```

Ilustración 22 Cambios básicos

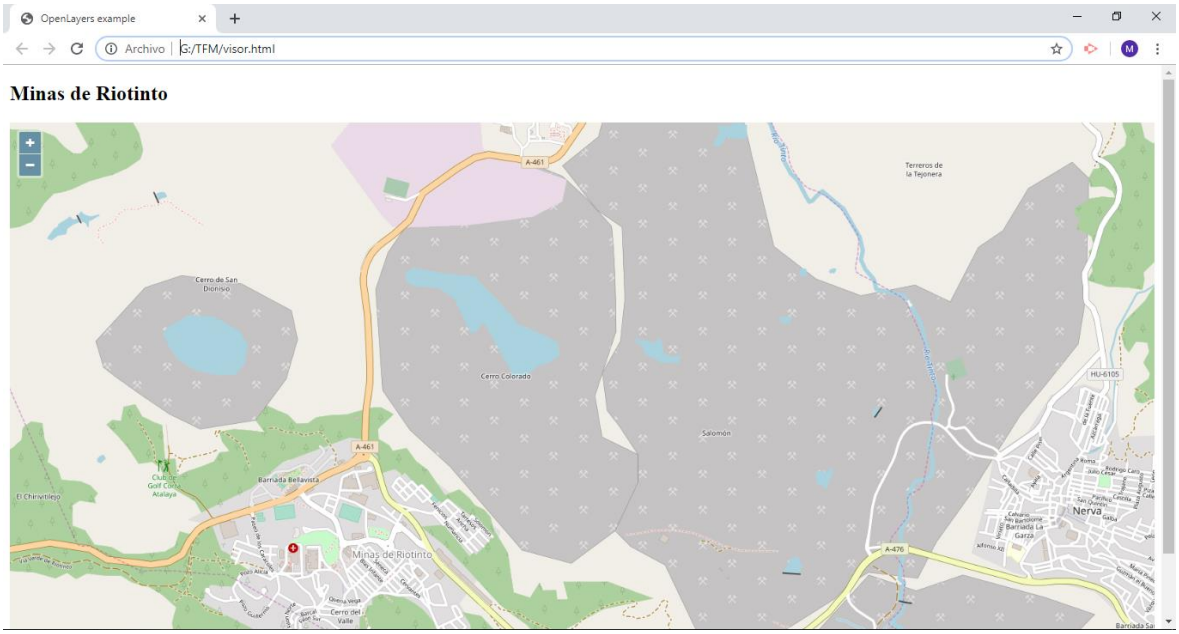


Ilustración 23 Visor con los cambios básicos

2. ***Insertar un kml de una imagen***

En Google Earth, podemos ver una imagen tomada en el 2013, pero con una aeronave no tripulada, se ha obtenido una imagen más reciente de la zona:

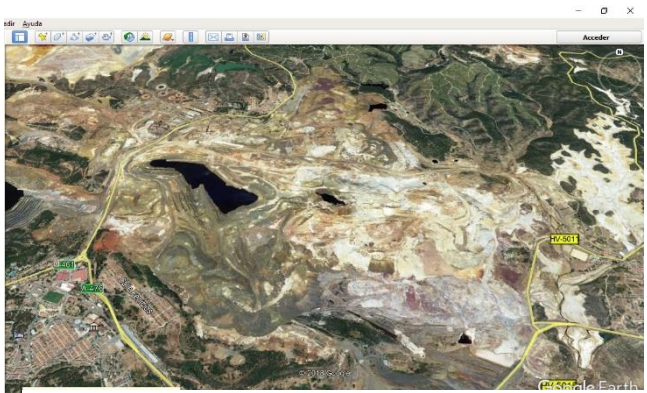


Ilustración 24 Imagen Google Earth 2013

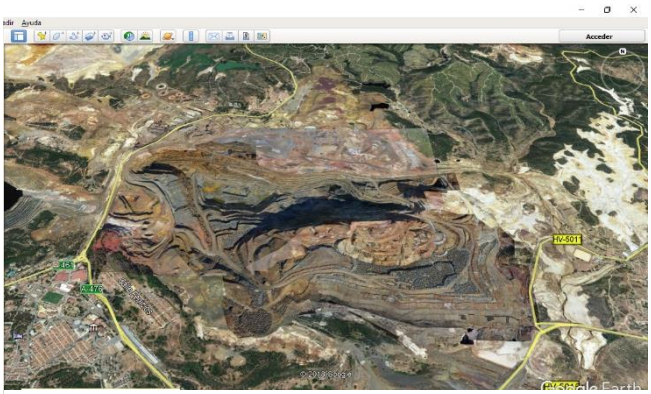


Ilustración 25 Ortofoto hecha con Dron

A continuación, se muestra el código de la ortofoto hecha por el Dron de la mina de Riotinto:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:gx="http://www.google.com/kml/ext/2.2"
xmlns:kml="http://www.opengis.net/kml/2.2" xmlns:atom="http://www.w3.org/2005/Atom">
<GroundOverlay>
<name>orto_nov_18_02.tif</name>
<description>Source Image: orto_nov_18_02.tif</description>
<Icon>

<href>https://recmin.com/CF_SYL/RIOTINTO%20CORTA/resources/orto.png</href>
<viewBoundScale>0.75</viewBoundScale>
</Icon>
<LatLonBox>
<north>37.71267661344199</north>
<south>37.6924337111975</south>
<east>-6.561162254675501</east>
<west>-6.593721369220825</west>
</LatLonBox>
</GroundOverlay>
</kml>
```

Ilustración 26 Código Ortofoto

Como se puede ver, las coordenadas que definen la imagen son coordenadas geográficas en forma de longitud y latitud en el sistema WGS 84. Cuando queremos meter esta imagen en OpenLayers mediante una función hay que meter las coordenadas transformadas en coordenadas UTM 30N, quedando la función en nuestro código de la siguiente manera:

```
var graphic = new ol.layer.Image({
source: new ol.source.ImageStatic({
url: 'https://recmin.com/CF_SYL/RIOTINTO CORTA/resources/orto.png',
imageExtent: [-734009.7051522139, 4536067.661023562, -730385.2411008531,
4538915.790758088]
})
});
map.addLayer(graphic);
```

Ilustración 27 Código de Ortofoto

Este trozo de código lo añadiremos en el código de la Ilustración 22 para que se inserte la imagen directamente cuando se ejecuta el visor.

3. ***Insertar botón para que muestre un kml:***

En el área de trabajo de la mina hay dos zonas, en las que los camiones tienen el acceso restringido. Estas dos zonas están definidas por una kml, los cuales están guardados en el servidor. Para incorporarlos en el visor, añadiremos dos botones, uno para cada zona, con el siguiente código:

```
<input type="button" class="more"  value="kml_area1"
onClick=newcapa("https://recmin.com/CF_SYL/RIOTINTO%20CORTA/resources/area1.kml
")>
```

Ilustración 28 Código insertar botón

La función que pide el botón se llama "*newcapa*", que se define con el código:

```
function newcapa(addcapa) {
        var namehh = addcapa;
        var vectorhh = new ol.source.Vector({
        url: namehh,
        format: new ol.format.KML()
        });
        var capahh = new ol.layer.Vector({
        source: vectorhh,
        title: "kml selecionado"
        });
        map.addLayer(capahh);
        }
```

Ilustración 29 Función añadir kml

4. ***Insertar capa PNOA:***

Otra opción que se puede meter en el código es hacer un botón para insertar la capa PNOA, creando un botón exclusivo para ello:

```
<input type="button" class="more"  value="Añadir capa PNOA"
onClick=newcapapnoa()>
```

Ilustración 30 Botón para la capa PNOA

La llamada a la función de la capa de PNOA es distinta a la llamada de un kml porque la llamamos con un enlace URL.

La función para mostrarlo es el siguiente:

```
function newcapapnoa(){
    var capapnoa = new ol.layer.Tile({
    title: 'image PNOA',
    source: new ol.source.TileWMS({
    url: 'http://www.idee.es/wms/PNOA/PNOA?',
    params: {
    'LAYERS': 'PNOA',
    'FORMAT': 'image/png',
    'type': 'base'
    //'SRS': 'epsg:25830'
    }
    }),
    visible:true
});

  map.addLayer(capapnoa);

  }
```

Ilustración 31 Función para insertar capa PNOA

5. ***Insertar PHP***

El PHP contiene el código que se puede ver en el apartado de resultados: Código creación del kml: f_kml.php. Este código se conecta al servidor y a una base de datos y después lanza la siguiente sentencia SQL:

select maquina, tiempo, lon, lat, material, conductor, origen, destino, velocidad, peso from flotas where tiempo >= ADDTIME(CURRENT_TIMESTAMP,-200) order by maquina, tiempo ASC

Esta sentencia SQL devuelve los campos que hemos pedido a los equipos mineros que enviaron los datos en los últimos dos minutos.

Luego crea un kml llamado pos.kml y que se guarda en la carpeta del servidor "f_kml". Antes de ejecutarlo, el archivo que tenemos para el visor que tiene extensión .html, al contener un archivo ejecutable de php, pasará a tener una nueva extensión de .php.

Para que el archivo se ejecute en el HTML hay que poner el siguiente código:

```
<?php
    include("f_kml.php");
?>
```

Ilustración 32 Incluir un PHP en HTML

Este archivo es ejecutado una vez, por lo que hay que decirle que lo ejecute cada cierto tiempo para que cree un nuevo archivo pos.kml. Para ello utilizamos el siguiente comando:

```
<script>
$(document).ready(function() {
    var refreshId =  setInterval( function(){
   $('#kml').load('f_kml.php');//actualizas el div
  }, 5000 );
});
</script>
```

Ilustración 33 Ejecución de archivo PHP cada cierto timepo

Al abrir el visor, según está, nos crea un archivo .kml en la carpeta “f_kml”. Este archivo es el que hay que insertar en nuestro visor ya que es el que muestra las rutas que llevan la maquinaria que tiene el dispositivo para control de flotas.

El archivo tiene el siguiente código:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.1">
  <Document>
    <Style id="ima_rojo">
      <IconStyle>
        <scale>1.1</scale>
        <Icon>
          <href>https://image.flaticon.com/icons/png/512/89/89140.png</href>
        </Icon>
        <hotSpot x="20" Y="2" xunits="pixels" yunits="pixels"/>
      </IconStyle>
    </Style>

    <Placemark id="KOMATSU HD785 Nº56">
      <name>ALEJANDRO REVALIENTE CASILLA</name>
      <description>- </description>
      <LineString>
        <coordinates>-6.5862386,37.7082624 -6.5862386,37.7082624 -
6.5862386,37.7082624 -6.5862386,37.7082624 -6.5862386,37.7082624 -
6.5862386,37.7082624 </coordinates>
      </LineString>
    </Placemark>

    <Placemark>
      <styleUrl>ima_rojo</styleUrl>
      <Point>
        <gx:drawOrder>1</gx:drawOrder>
        <coordinates>-6.5862386,37.7082624 </coordinates>
      </Point>
    </Placemark>

    <Placemark id="KOMATSU HD785 Nº57">
```

```
            <name>SERGIO LOPEZ DE PABLO</name>
            <description>- Esteril voladura</description>
            <LineString>
              <coordinates>-6.5714995,37.7065131 -6.5701934,37.7060879 -
6.5699435,37.7055337 -6.5702846,37.705163 -6.5703507,37.7052449 -6.5703167,37.7052322
</coordinates>
            </LineString>
          </Placemark>

          <Placemark>
            <styleUrl>ima_rojo</styleUrl>
            <Point>
              <gx:drawOrder>1</gx:drawOrder>
              <coordinates>-6.5703167,37.7052322 </coordinates>
            </Point>
          </Placemark>

          <Placemark id="KOMATSU HD785 Nº58">
            <name>JOSÉ MANUEL ROJAS</name>
            <description>- Esteril voladura</description>
            <LineString>
              <coordinates>-6.5864495,37.709642 -6.5854503,37.7093775 -
6.5846299,37.7096361 -6.5851159,37.7099717 -6.5849672,37.7104449 -
6.5847245,37.7104341 </coordinates>
            </LineString>
          </Placemark>

          <Placemark>
            <styleUrl>ima_rojo</styleUrl>
            <Point>
              <gx:drawOrder>1</gx:drawOrder>
              <coordinates>-6.5847245,37.7104341 </coordinates>
            </Point>
          </Placemark>
        </Document>
      </kml>
```

Ilustración 34 KML generado a partir del PHP

Para que se vea en el visor, sin falta de darle a ningún botón, tenemos el siguiente parte del código:

```
var vector = new ol.source.Vector({
      url: "f_kml/pos.kml",
      format: new ol.format.KML()
      });
      var capa = new ol.layer.Vector({
      source: vector,
      title: "kml php"
      });
      map.addLayer(capa);
```

Ilustración 35 Código para mostrar el KML

Al probar nuestro código, vemos que el estilo del kml es el que usa Google Earth. Para cambiarlo definimos un nuevo estilo con el siguiente código:

```
var icono8 = new ol.style.Style({
      stroke: new ol.style.Stroke({
         color: 'yellow',
         width: 2
      }),
      image: new ol.style.Icon({
         anchor: [1, 1],
         size: [483, 296],
         offset: [0, 0],
         opacity: 1,
         scale: 0.08,
         src: 'https://recmin.com/GISWEB/SAM/RT/imagen/dumper.png'
      })
   });
```

Ilustración 36 Definición del estilo del KML

Donde definimos el estilo de la línea y la imagen que queremos que se muestre en nuestro kml, en este caso una imagen de un dumper:

Ilustración 37 Imagen que se muestra en el KML

Código CSS

Paralelamente al desarrollo del código PHP, se ha hecho el código “estilos.css” para la mejor apariencia de la página web.

Se ha pensado en hacer un encabezado con el título del trabajo y los logos de la Universidad de Oviedo y de la Escuela Politécnica de Mieres con enlaces a sus respectivas páginas web, poner un color al fondo de pantalla y un pie de página.

Para ello, en el código PHP se crea la etiqueta de <header> (encabezamiento):

```
<header>
```

```
    <a href="http://www.uniovi.es/" target="_blank"> <img class="izquierda" src=
"imagen/logoUniovi.png" alt=""></a>
    <a href="https://epm.uniovi.es/" target="_blank"><img class="derecha" src=
"imagen/epm.png" alt=""></a>
    <h2><img class="izquierda"><img class="derecha">TITULO DEL TRABAJO Minas de
Riotinto</h2>
  </header>
```

Ilustración 38 Header en el HTML

Y la etiqueta <footer> (pie de página):

```
<footer>
  <p> Web desarrollada por María Suárez Álvarez</p>
</footer>
```

Ilustración 39 Footer en el HTML

De este paso, pasamos al .css para definir las características de cada etiqueta (véase Código estilos.css)

Usuario y Contraseña

El visor Web, contiene datos confidenciales de la explotación por lo que es necesario insertarle un Usuario y Contraseña para que sea de acceso limitado.

Para ello contamos con varios archivos php (Véase en el apartado de resultados o en los siguientes enlaces):

- bloqueDeSeguridad.php
- login.php
- Autenticacion.php
- salir.php

La idea consiste en que nos pida el usuario y la contraseña antes de iniciar el visor, por lo que al principio de nuestro *index.php*, tenemos que poner la siguiente línea:

```
<?include ("bloqueDeSeguridad.php");?>
```

Ilustración 40 Código para ejecutar el archivo bloqueDeSeguridad.php

Este php llama al código el bloqueDeSeguridad.php, el cual se pregunta si ya hay una sesión abierta, en caso contrario ejecuta el código login.php, que muestra la siguiente pantalla en nuestro navegador:

Ilustración 41 Inicio Sesión

Los datos de usuario y contraseña se encuentran en el código Autenticacion.php, así que este código comprueba que esos datos son escritos correctamente. Si se escribe mal la contraseña o el usuario nos muestra un error en rojo:

Ilustración 42 Datos introducidos incorrectos

Una vez introducidos los datos correctos, se ejecutará el visor que hemos creado.

Para cerrar sesión, no vale con cerrar la pestaña del navegador, ya que queda guardada la sesión. Para ello, se ha incluido un botón:

Ilustración 43 Botón para cerrar sesión

Este botón ejecuta el archivo salir.php el cual nos indica que la sesión está cerrada y nos da la opción de iniciar la sesión otra vez.

Esta opción se ve así:

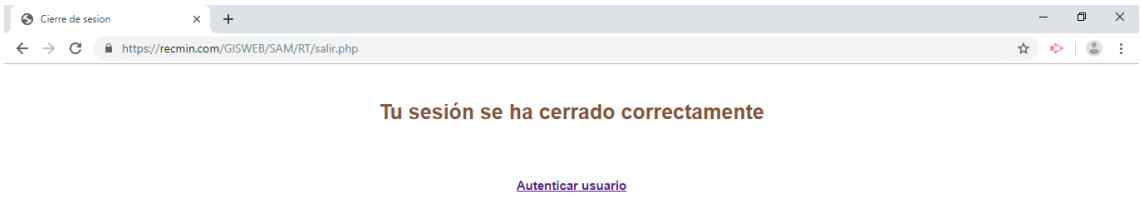


Ilustración 44 Cierre de sesión

Contador

El KML se debe actualizar cada cierto tiempo para que se pueda ver el movimiento de la maquinaria. Para saber cuánto falta para que se actualice, ponemos un contador, usando un código de JavaScript (véase: Código contador.js)

En este código, hace una cuenta atrás desde 10 a 0. Cuando llega a 0, se para, por lo que se debe de volver a reiniciar el contador. Para ello utilizamos la "*función actualiza"* del Código funciones.js y el comando "*setInterval*".

En el php se mete el siguiente código para que muestre en pantalla el contador:

```
<table border="0" align="right">
```

```
            <tr><td style="color:#845237">Faltan: </td><td> <span id="reloj"
style="color:#845237"></span> </td><td style="color:#845237"> segundos para
actualizar</td></tr>
        </table>

        <script src="contador.js"></script>
```

Ilustración 45 Código para mostrar el contador

El resultado podemos observarlo en la siguiente ilustración:

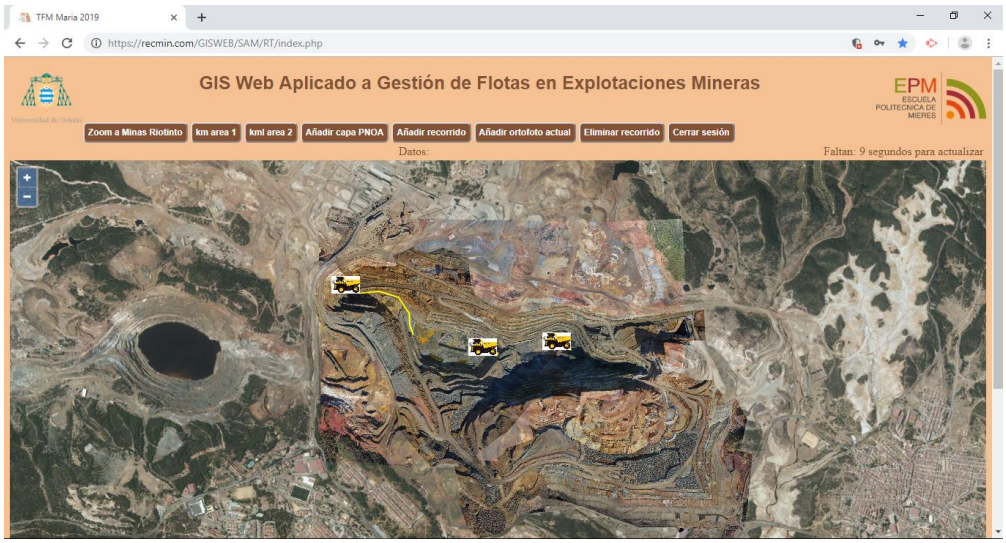


Ilustración 46 Contador

Mostrar datos

Para mostrar los datos que obtenemos del kml, tenemos que hacer una función que interactúe con el kml. Para ello, tenemos varios ejemplos en OpenLayers el cual se ha decidido escoger el Código lectura.js.

```
<script src="lectura.js"></script>
```

Ilustración 47 Código para llamar al código

Esta línea de comando se debe de poner después de nombrar la función que muestra el mapa, sino no funciona el código.

Este código muestra los datos en la parte superior de la página, donde se llama a la función:

```
<table border="0" align="center">
  <tr><td style="color:#845237">Datos: </td><td><span id="info"
style="color:#845237"></span></td></tr>
</table>
```

Ilustración 48 Código para mostrar los datos

Ilustración 49 Muestra de datos de la maquinaria minera

5. Resultados

Códigos resultantes

1. ***Código principal: index.php***

```
<?include ("bloqueDeSeguridad.php");?>
<!doctype html>
<html lang="es">
  <head>

    <meta charset="utf-8">

    <link rel="stylesheet"
href="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/css/ol.css"
type="text/css">
    <link rel="stylesheet" href="estilos.css">
    <link rel="icon" href="imagen/epm.png">

    <script src="https://code.jquery.com/jquery-3.2.1.js"></script>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
    <script src="ol/ol.js"></script>

    <title>TFM Maria 2019</title>

    <a href="http://www.uniovi.es/" target="_blank"> <img class="izquierda" src=
"imagen/logoUniovi.png" alt=""></a>
    <a href="https://epm.uniovi.es/" target="_blank"><img class="derecha" src=
"imagen/epm.png" alt=""></a>
    <h2><img class="izquierda"><img class="derecha">GIS Web Aplicado a Gestión de Flotas
en Explotaciones Mineras</h2>

  </head>

  <body>

 </br>
    <input type="button" class="pointer"  value="Zoom a Minas Riotinto"
onClick=zoomriotinto()>
    <input type="button" class="pointer"  value="km area 1"
onClick=newcapa("https://recmin.com/CF_SYL/RIOTINTO%20CORTA/resources/area1.kml")>
    <input type="button" class="pointer"  value="kml area 2"
onClick=newcapa("https://recmin.com/CF_SYL/RIOTINTO%20CORTA/resources/area2.kml")>
    <input type="button" class="pointer"  value="Añadir capa PNOA" onClick=newcapapnoa()>
    <input type="button" class="pointer"  value="Añadir recorrido" onClick=newkml()>
    <input type="button" class="pointer"  value="Añadir ortofoto actual" onClick=orto()>
    <input type="button" class="pointer"  value="Eliminar recorrido" onClick=elimina()>
    <input type="button" class="pointer"  value="Cerrar sesión" onClick=salir()>
    </br>
    <table border="0" align="right">
        <tr><td style="color:#845237">Faltan: </td><td> <span id="countdown"
style="color:#845237"></span> </td><td style="color:#845237"> segundos para
actualizar</td></tr>
    </table>
```

```
    <table border="0" align="center">
        <tr><td style="color:#845237">Datos: </td><td><span id="info"
style="color:#845237"></span></td></tr>
    </table>

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


    <div id="kml">
    <?php
        include("f_kml.php");
    ?>
    </div>

        <script>
        $(document).ready(function() {
            var refreshId =  setInterval( function(){
            $('#kml').load('f_kml.php');//actualizas el div
          }, 5000 );
        });

    </script>

    <div id="salir">
        <script>
            function salir(){
                window.location="salir.php";
            }
        </script>
    </div>


    <script src="contador.js"></script>
    <script src="funciones.js"></script>


    <script type="text/javascript">
    //DEFINICIÓN DE VARIABLES
//Variable de capa KML de camiones
    var capakml;
//Variable para poder usar el contador
    var totalTime = 10;

//INSERCCIONES DIRECTAS:
//Inserción directa de  capa base de OSM
    var map = new ol.Map({
        target: 'map',
        layers: [
            new ol.layer.Tile({
            source: new ol.source.OSM()
          })
        ],
        view: new ol.View({
            center: ol.proj.fromLonLat([-6.58, 37.7005]),
            zoom: 14.7
        })
    });
    var icono8 = new ol.style.Style({
        stroke: new ol.style.Stroke({
```

```
            color: 'yellow',
            width: 2
        }),
        image: new ol.style.Icon({
            anchor: [1, 1],
            size: [483, 296],
            offset: [0, 0],
            opacity: 1,
            scale: 0.08,
            src: 'https://recmin.com/GISWEB/SAM/RT/imagen/dumper.png'
        })
    });

// Insercción directa de  capa PNOA:
    newcapapnoa();

// Insercción directa de  capa Ortofoto:
     orto();

// Insercción directa de capa KML de camiones  :
    newkml();
    </script>

    <script src="lectura.js"></script>



    <footer>
        <p> Web desarrollada por María Suárez Álvarez</p>
    </footer>


  </body>

</html>
```

2. Código estilos.css

```
img.izquierda {
    float: left;
    width: 100px;
}

img.derecha {
    float: right;
    width: 150px;
}


header {
    background: #F7C293;
}

footer {
    background: #F7C293;
 }


body {
    background:#F7C293;
}

h2{
    text-align: center;
    color: #845237;
    font-size: 25px;
    font-family: Arial, Helvetica, sans-serif;
}

p{
    text-align: center;
    color: #845237;
    font-size: 25px;
}

.map {
    height: 100%;
    width: 100%;
}

.pointer {
    cursor:pointer;
    text-decoration: none;
    padding: 5px;
    font-weight: 600;
    font-size: 12px;
    color: #ffffff;
    background-color: #845237;
    border-radius: 6px;
    border: 2px solid #0000000;
 }
 .pointer:hover{
    color: #845237;
    background-color: #ffffff;
 }
```

3. Código funciones.js

```
// FUNCIONES:
// Función para meter cualquier capa KML
   function newcapa(addcapa) {
         var namehh = addcapa;
         var vectorhh = new ol.source.Vector({
         url: namehh,
         format: new ol.format.KML()
         });
         var capahh = new ol.layer.Vector({
         source: vectorhh,
         title: "kml selecionado"
         });
      map.addLayer(capahh);
   }


// Función para meter capa KML de camiones
   function newkml() {
         var vector = new ol.source.Vector({
            url: "f_kml/pos.kml",
            format: new ol.format.KML({
            extractStyles: false
            })
         });
         capakml = new ol.layer.Vector({
            source: vector,
            title: "kml php",
            style: icono8
         });
      map.addLayer(capakml);
   }

// Función para eliminar la capa de KML
   function elimina(){
      map.removeLayer(capakml);
   }

// Función para meter capa ortofoto
   function orto(){
      var graphic = new ol.layer.Image({
         source: new ol.source.ImageStatic({
            url: 'https://recmin.com/CF_SYL/RIOTINTO CORTA/resources/orto.png',
            imageExtent: [-734009.7051522139,  4536067.661023562, -
730385.2411008531, 4538915.790758088]
         })
      });
      map.addLayer(graphic);
   }

// Función para hacer un zoom a la zona de la Mina
   function zoomriotinto() {
      map.getView().setCenter(ol.proj.fromLonLat([-6.58, 37.7005]));
      map.getView().setZoom(14.7);
   }

// Función para meter capa PNOA
```

```
    function newcapapnoa(){
        var capapnoa = new ol.layer.Tile({
        title: 'image PNOA',
        source: new ol.source.TileWMS({
        url: 'http://www.idee.es/wms/PNOA/PNOA?',
        params: {
        'LAYERS': 'PNOA',
        'FORMAT': 'image/png',
        'type': 'base'
        //'SRS': 'epsg:25830'
        }
        }),
        visible:true
        });
        map.addLayer(capapnoa);
    }

//Función para actualizar la capa de KML y el contador
    function actualiza() {
        map.removeLayer(capakml);
        newkml();
        totalTime = 10;
    }

// Comando setInterval para que repita la función "actualiza" cada 10 segundos
    setInterval("actualiza()",10000);
```

4. Códigos auxiliares:

a. Código contador.js

```
// Función para el contador:

window.onload = updateClock;


function updateClock() {
  document.getElementById('countdown').innerHTML = totalTime;

  if(totalTime==0){
    totalTime+=10;
   // console.log('Final');
  }else{
    totalTime-=1;
    setTimeout("updateClock()",1000);
  }
}
```

b. Código lectura.js

```
var displayFeatureInfo = function(pixel) {
      var features = [];
      map.forEachFeatureAtPixel(pixel, function(feature) {
        features.push(feature);
      });
      if (features.length > 0) {
        var info = [];
        var i, ii;
        for (i = 0, ii = features.length; i < ii; ++i) {
        info.push(features[i].get('name'));
        info.push(features[i].get('description'));

        }
        document.getElementById('info').innerHTML = info.join(', ') || '';
        map.getTarget().style.cursor = 'pointer';
      } else {
        document.getElementById('info').innerHTML = ' ';
        map.getTarget().style.cursor = '';
      }
    };

map.on('pointermove', function(evt) {
      if (evt.dragging) {
        return;
      }
      var pixel = map.getEventPixel(evt.originalEvent);
      displayFeatureInfo(pixel);
});


map.on('click', function(evt) {
   displayFeatureInfo(evt.pixel);
});
```

c. Código creación del kml: f_kml.php

```
<?php
$servername = "server:3307";
$database = "basededatos";
$username = "nombre";
$password = "contrasena";
try {
   $conn = new PDO("mysql:host=$servername;dbname=$database", $username, $password);
   // set the PDO error mode to exception
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
   // Conectado correctamente
   $stmt = $conn->prepare("select maquina, tiempo, lon, lat, material, conductor, origen, destino, velocidad, peso from flotas where tiempo >= ADDTIME(CURRENT_TIMESTAMP,-200) order by maquina, tiempo ASC");
   $stmt->execute();
   // set the resulting array to associative
   $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
   try {
```

```
        //INTERPRETAMOS LOS REGISTROS DEVUELTOS EN EL SQL Y GENERAMNOS UN FICHERO kml
        // Creates the Document.
        $dom = new DOMDocument('1.0', 'UTF-8');
        // Creates the root KML element and appends it to the root document.
        $node = $dom->createElementNS('http://earth.google.com/kml/2.1', 'kml');
        $parNode = $dom->appendChild($node);
        // Creates a KML Document element and append it to the KML element.
        $dnode = $dom->createElement('Document');
        $docNode = $parNode->appendChild($dnode);
        $node1 = $dom->createElement('Style');
        $placeNode1 = $docNode->appendChild($node1);
        // Creates an id attribute and assign it the value of id column.
        $node1->setAttribute('id', "ima_rojo");
        $descNode1 = $dom->createElement('IconStyle');
        $placeNode1->appendChild($descNode1);
        $nameNode2 = $dom->createElement('scale',1.1);
        $descNode1->appendChild($nameNode2);
        $descNode3 = $dom->createElement('Icon');
        $descNode1->appendChild($descNode3);
        //$nameNode4 = $dom->createElement('href',"https://image.flaticon.com/icons/png/512/89/89140.png");
        $nameNode4 = $dom->createElement('href',"https://image.flaticon.com/icons/png/512/89/89140.png");
        $descNode3->appendChild($nameNode4);
        $nameNode5 = $dom->createElement('hotSpot');
        $nameNode5->setAttribute('x',"20");
        $nameNode5->setAttribute('Y',"2");
        $nameNode5->setAttribute('xunits',"pixels");
        $nameNode5->setAttribute('yunits',"pixels");
        $descNode1->appendChild($nameNode5);
        $cam = "";
        $van = 0;
        $vv = 0;
        $coorStr1 = "";
        foreach($result as $row) {
           $van = $van + 1;
           if ($cam == $row['maquina']) {
              $coorStr = $coorStr . $row['lon'] . ','  . $row['lat'] . " ";
              $coorStr1 = $row['lon'] . ','  . $row['lat'] . " ";
           } else {
              if ($coorStr <> "") {
                 $coorNode = $dom->createElement('coordinates', $coorStr);
                 $lineNode->appendChild($coorNode);
                 $node8 = $dom->createElement('Placemark');
                 $placeNode8 = $docNode->appendChild($node8);
                 $node9 = $dom->createElement('styleUrl',"ima_rojo");
                 $placeNode8->appendChild($node9);
                 $node10 = $dom->createElement('Point');
                 $placeNode8->appendChild($node10);
                 $node11 = $dom->createElement('gx:drawOrder',"1");
                 $node10->appendChild($node11);
                 $coorNode1 = $dom->createElement('coordinates', $coorStr1);
                 $node10->appendChild($coorNode1);
                 $van = 0;
                 $coorStr = "";
              }
              $node = $dom->createElement('Placemark');
              $placeNode = $docNode->appendChild($node);
```

```
                // Creates an id attribute and assign it the value of id column.
                $placeNode->setAttribute('id', $row['maquina']);
                // Create name, and description elements and assigns them the values of the name
and address columns from the results.
                //$nameNode = $dom->createElement('name',htmlentities($row['name']));
                $nameNode = $dom->createElement('name',$row['conductor']);
                $placeNode->appendChild($nameNode);
                $vv = $vv + 1;
                $descNode = $dom->createElement('description', "- ".$row['material']);
                $placeNode->appendChild($descNode);
                //$styleUrl = $dom->createElement('styleUrl', '#' . 'restaurantStyle' . 'Style');
                //$placeNode->appendChild($styleUrl);
                // Creates a Point element.
                $lineNode = $dom->createElement('LineString');
                $placeNode->appendChild($lineNode);
                // Creates a coordinates element and gives it the value of the lng and lat columns
from the results.
                $cam = $row['maquina'];
                $coorStr = $coorStr . $row['lon'] . ','  . $row['lat'] . " ";
                $coorStr1 = $row['lon'] . ','  . $row['lat'] . " ";
            }
        }
        $coorNode = $dom->createElement('coordinates', $coorStr);
        $lineNode->appendChild($coorNode);
        $node8 = $dom->createElement('Placemark');
        $placeNode8 = $docNode->appendChild($node8);
        $node9 = $dom->createElement('styleUrl',"ima_rojo");
        $placeNode8->appendChild($node9);
        $node10 = $dom->createElement('Point');
        $placeNode8->appendChild($node10);
        $node11 = $dom->createElement('gx:drawOrder',"1");
        $node10->appendChild($node11);
        $coorNode1 = $dom->createElement('coordinates', $coorStr1);
        $node10->appendChild($coorNode1);
        $kmlOutput = $dom->saveXML();
        $nombre_fichero = "f_kml/pos.kml";
        file_put_contents($nombre_fichero, $kmlOutput);
        $conn = null;
    } catch (\Throwable $th) {
    }
    }
catch(PDOException $e)
    {
    //    echo "Fallo conectando: " . $e->getMessage();
    }
?>
```

d. ***Códigos para el inicio y fin de sesión:***

i. ***Autenticacion.php***

```
<?
//vemos si el usuario y contraseña son válidos
if ($_POST["usuario"]=="XXXXXX" && $_POST["contrasena"]=="XXXXXX"){
```

```
//usuario y contraseña válidos
//se define una sesión y se guarda el dato
session_start();
$_SESSION["autenticado"]= "SI";
header ("Location: index.php");
}else {
//si no existe se va a login.php
header("Location: login.php?errorusuario=si");
}
?>
```

ii. ***bloqueDeSeguridad.php***

```
<?
//Inicio la sesión
session_start();
//COMPRUEBA QUE EL USUARIO ESTA AUTENTICADO
if ($_SESSION['autenticado'] != 'SI') {
//si no existe, va a la página de autenticación
header('Location: login.php');
//salimos de este script
exit();
}
?>
```

iii. ***login.php***

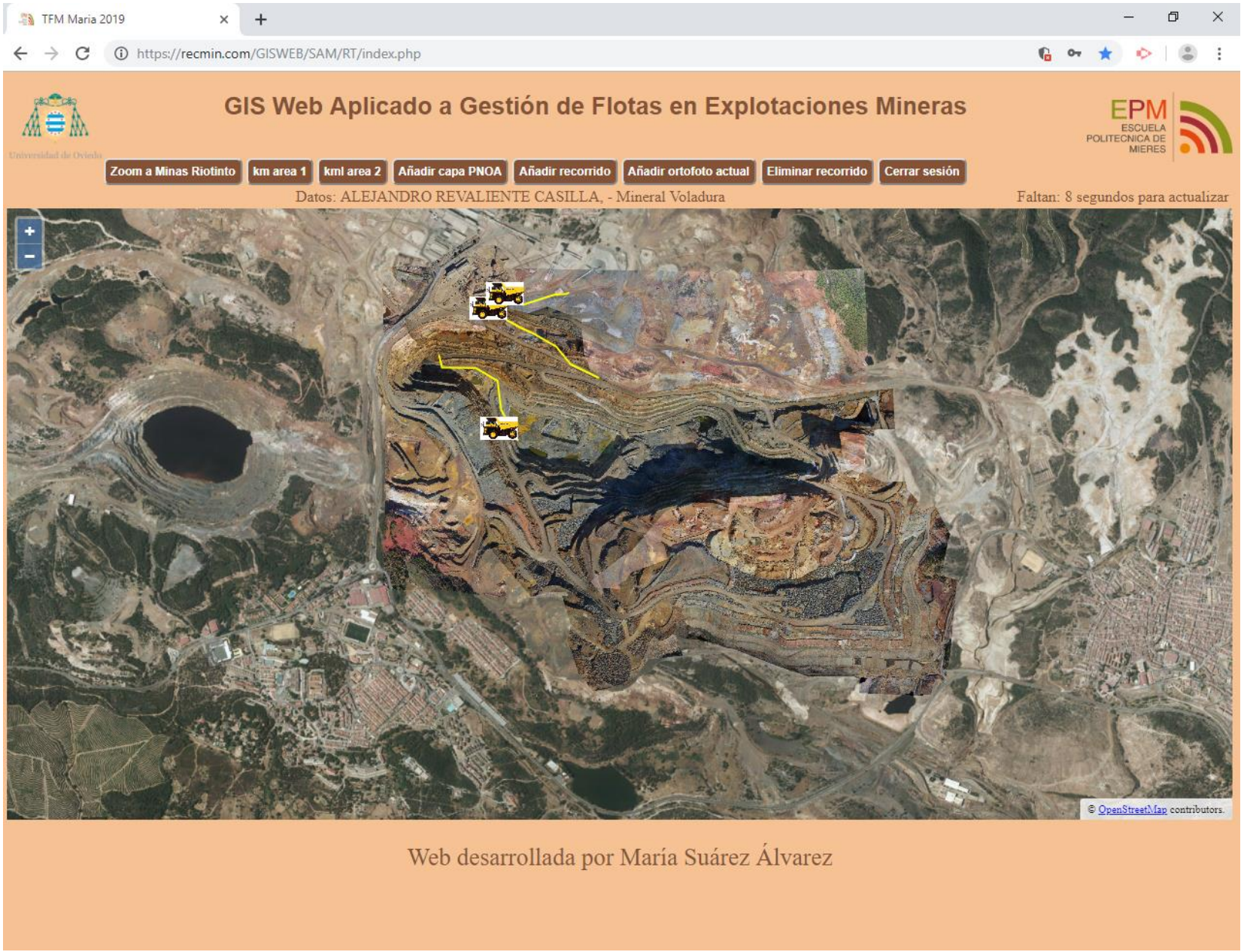
```
<html>
   <head>
   <title>Inicio de Sesion</title>
   <style>
   h1{
   text-align: center;
   color: #845237;
   font-size: 25px;
   font-family: Arial, Helvetica, sans-serif;
   }
   h2{
   text-align: center;
   color: #845237;
   font-size: 15px;
   font-family: Arial, Helvetica, sans-serif;
   }
   h3{
   text-align: center;
   color: red;
      font-size: 15px;
   font-family: Arial, Helvetica, sans-serif;
   }
   .pointer {
   cursor:pointer;
   text-decoration: none;
   padding: 5px;
   font-weight: 600;
   font-size: 10px;
   color: #ffffff;
```

```
        background-color: #845237;
        border-radius: 6px;
        border: 2px solid #0000000;
      }
      </style>

      </head>

      <body>
        <h1>Inicio de Sesión</h1>

        <?if ($_GET['errorusuario']=="si"){?>

          <b><h3>¡¡Datos incorrectos!! Vuelva a intentarlo</h3></b>

        <?}else{?>
        <h2>Introduce tu nombre de usuario y contraseña:</h2>
        <?}?>

        <form action="autenticacion.php" method="POST" >
          <table border="0" align="center">
            <tr><td style="color:#845237">Nombre de usuario:</td><td><input
name="usuario" size="25" value=""/></td></tr>

            <tr><td style="color:#845237">Contraseña:</td><td><input
name="contrasena" size="25" type="password"/></td></tr>

            <tr><td colspan="3"style="text-align:center;"><input type="submit"
class="pointer" value="Inicio de sesión"/></td></tr>
          </table>
        </form>

      </body>
    </html>
```

iv. salir.php

```
    <?
    session_start();
    session_destroy();

    ?>
    <html>
    <head>
      <title>Cierre de sesion</title>
      <style>
        h1{
        text-align: center;
        color: #845237;
        font-size: 25px;
        font-family: Arial, Helvetica, sans-serif;
        }
        h2{
        text-align: center;
        color: #845237;
        font-size: 15px;
        font-family: Arial, Helvetica, sans-serif;
```

```
      }
    </style>
    <script>
      function cerrar()
       {
        window.close();
       }
    </script>

</head>

<br>
<body  onLoad="cerrar()">
<h1>Tu sesión se ha cerrado correctamente</h1>
<br>
<br>
<h2><a href='login.php'>Autenticar usuario</a></h2>

</body>
</html>
```


TFM Maria 2019
https://recmin.com/GISWEB/SAM/RT/index.php
GIS Web Aplicado a Gestión de Flotas en Explotaciones Mineras
EPM
ESCUELA POLITECNICA DE MIERES
Zoom a Minas Riotinto
km area 1
kml area 2
Añadir capa PNOA
Añadir recorrido
Añadir ortofoto actual
Eliminar recorrido
Cerrar sesión
Datos: ALEJANDRO REVALIENTE CASILLA, - Mineral Voladura
Faltan: 8 segundos para actualizar
© OpenStreetMap contributors.
Web desarrollada por María Suárez Álvarez

6. Conclusiones

Finalizada la elaboración de este trabajo se han cumplidos los objetivos que nos habíamos marcado para este proyecto:

- Se ha creado un archivo KML mediante un archivo php a partir de los datos de la maquinaria minera.
- Se ha visualizado la información en formato KML de la maquinaria minera localizada.
- Se ha programado una actualización de ese KML a través de la ejecución del archivo php, cada diez segundos.
- Se codifica una función para que el cursor muestre la información de la maquinaria a pasar por encima de la misma.
- Se ha añadido al visor web una cuenta atrás para conocer el tiempo de espera hasta la siguiente actualización automática.
- Se bloquea el acceso al visor web mediante una clave de usuario y contraseña.

Pero este visor no acaba aquí, ya que se pueden incluir bastantes mejoras, como son:

- Alertas de aviso en zonas prohibidas. En este proyecto se han incluido dos polígonos que son zonas de acceso restringido. Una de las mejoras sería crear una alerta de aviso que se activa cuando una maquinaria no autorizada entra en una de esas zonas.
- Actualización de ortofoto. Cada mes, se hace una ortofoto con un dron en la zona de la mina. Esta actualización mensual, sería otro objetivo que mejorar.
- Aumento de equipos móviles. Una mejora por parte de la empresa es aumentar el número de equipos móviles ya que no todos tienen esta mejora en sus maquinarias.

Concluido la elaboración de este trabajo hacemos balance de lo aprendido durante este curso académico. Haciendo hincapié en este trabajo, se ha concluido la gran utilidad que tienen los visores web sin falta de tener conocimientos de GIS, ni instalación de programas GIS para su visualización.

7. Bibliografía

- http://www.mws4mining.com/
- https://openlayers.org/
- https://recmin.com/WP/
- https://volaya.github.io/libro-sig/chapters/Historia.html
- https://es.ccm.net/faq/494-como-colocar-un-icono-en-la-barra-de-direcciones-favicon
- http://mappinggis.com/2013/04/como-crear-un-mapa-con-openlayers-3/
- https://www.lawebdelprogramador.com/foros/PHP/1654530-Actualizar-funcion-PHP-cada-cierto-tiempo.html
- https://es.stackoverflow.com/questions/55668/actualizar-div-autom%C3%A1ticamente
- https://codeday.me/es/qa/20190416/507809.html
- http://www.forosdelweb.com/f18/ejecutar-codigo-php-con-boton-841529/
- https://www.bufa.es/javascript-cuenta-atras-de-x-segundos/