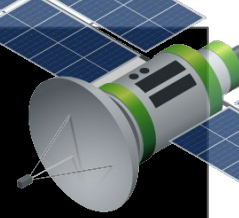
Facultad de Ingeniería

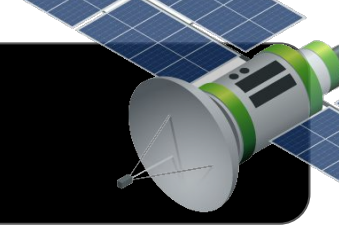


Encontrar y ser encontrado

Localización con javascript y un celular

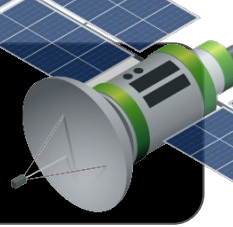
Evans, Felipe - Mutti, Pilar - Mateos, Wenceslao

Índice



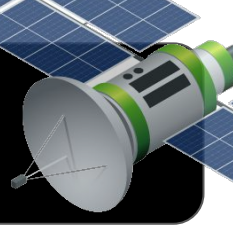
- **Geolocalización de un celular**
  - GPS
  - AGPS
- **Geolocalización en HTML5**
  - Standard html5
  - Ejemplo
  - Objeto devuelto
- **Ajax**
- **Backend**
  - Spatial data type
  - Ejemplo de inserción
  - Algunas de búsquedas
  - Ejemplo de php con inserción en mysql
- **Vinculación en html5**
  - Openlayer.
- **Licencias de mapas**
- **Ejemplos**
  - Una aplicación que suba una foto y su posición. Un lugar donde se visualice lo visto
  - Una aplicación de trackeo

Métodos de Localización



- GPS/GLONASS/Galileo/BEIDOU
- AGPS o AGSS
- GSM Location
- Wifi Location
- Bluetooth Location

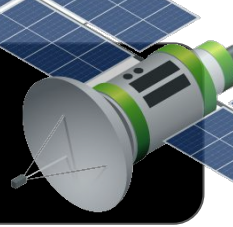
GPS



Sistema construido por el Departamento de defensa de los EEUU. Es un constelación de al menos 24 satélites, para esto la fuerza aérea en el 24 de abril del 2019 (órgano que controla la constelación) tenía 31 satélites funcionales. Sitio oficial https://www.gps.gov.

Hasta el 2000 para actividades civiles se el agregaba un error a la señal civil. Los satélites emiten dos señales una para uso civil y otra para uso militar. El uso militar usa también la civil corrigiendo y mejorando el error de posicionamiento. A partir del 2014 para uso civil se empezó a agregar en los nuevos satélites la señal L2C agregando a la señal actual de L1. Esta señal tiene mas energia, esto permite mejor recepción en interiores y mayor velocidad de adquisición.

GLONASS



Sistema construido por la disuelta Unión Soviética actualmente mantenido por la Federación Rusa. Sitio oficial https://www.glonass-iac.ru/en/.

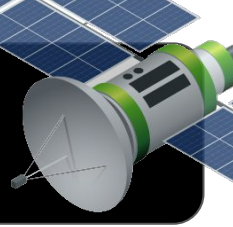
En el 17 de agosto del 2019 la constelación de GLONASS está constituida por 27 satélites de los cuales 23 se encuentran efectivamente operacionales.

Repasando errores

Fuentes de error de los sistemas satelitales.

- Retraso de la señal en las distintas capas de la atmósfera.
- Señal multirruta. Recepción de la señal de mismo GPS producida por el rebote de la señal en edificios y montañas cercanos.
- Errores de en las órbitas de los satélites y vientos solares.
- Número de satélites visibles. Se necesita una visualización de 7 satélites para tener un error inferior al 2,5 m. Esto ocurre casi el 95 % del tiempo.
- Errores locales en el reloj del GPS.

AGPS



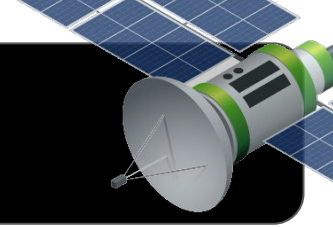
A- GPS o assisted global positioning system, también conocido como GPS asistido consta de una mejora al sistema de GPS tradicional por el problema de primer posicionamiento o posicionamiento inicial, que suele ser muy largo en general, incluso en condiciones óptimas puede llegar a minutos.

La asistencia se divide en dos categorías:

- Basado en estación móvil (MSB): información utilizada para adquirir satélites más rápidamente. Puede suministrar datos orbitales o almanaques para los satélites GPS al receptor GPS, permitiendo que el receptor GPS se bloquee a los satélites más rápidamente en algunos caso. La red puede proporcionar un tiempo preciso.
- Estación móvil asistida (MSA): cálculo de la posición por parte del servidor utilizando información del receptor GPS. El dispositivo captura una instantánea de la señal GPS, con un tiempo aproximado, para que el servidor la procese posteriormente en una posición. El servidor de asistencia tiene una buena señal de satélite y abundante potencia de cálculo, por lo que puede comparar señales fragmentarias transmitidas a él. Las coordenadas topográficas precisas para las torres del sitio de la celda permiten un mejor conocimiento de las condiciones ionosféricas locales y otras condiciones que afectan la señal GPS que el receptor GPS solo, lo que permite un cálculo más preciso de la posición.

Como beneficio adicional, en las implementaciones asistidas por estación móvil, la cantidad de procesamiento y software necesarios para un receptor GPS se puede reducir descargando la mayor parte del trabajo en el servidor de asistencia.

Geolocalización con HTML5 - Ejemplo



```
<!DOCTYPE html>
<html><body>

<p>Click the button to get your coordinates.</p> <button onclick="getLocation()">Try It</button> <p id="demo"></p>

<script>
var x = document.getElementById("demo");
function getLocation() {
  if (navigator.geolocation) {
          navigator.geolocation.getCurrentPosition(showPosition);
  } else {
          x.innerHTML = "Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>
</body> </html>
```

Origen: https://www.w3schools.com/html/html5_geolocation.asp

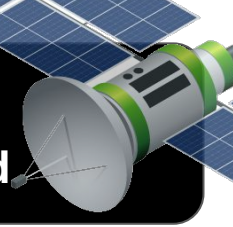
https://www.w3.org/TR/geolocation-API
https://www.w3.org/2008/Talks/0904-fit2008/geoloc-ms/slides.html
https://developer.mozilla.org/es/docs/WebAPI/Using_geolocation

1. API Description
    - 5.1 Geolocation interface
    - 5.2 PositionOptions interface
    - 5.3 Position interface
    - 5.4 Coordinates interface
    - 5.5 PositionError interface

Future

https://w3c.github.io/geolocation-sensor/

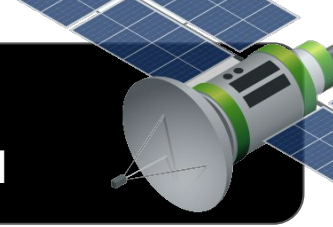
Geolocalizacion con HTML5 - getCurrentPosition() Method



Devolución de propiedades

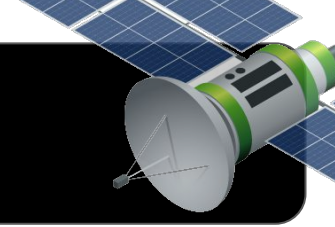
coords.latitude	La latitud como un número decimal (devuelto siempre)
coords.longitude	La longitud como un número decimal (devuelto siempre)
coords.accuracy	La precisión de la posición (devuelto siempre)
coords.altitude	La altitud en metros sobre el nivel del mar (devuelto si está disponible)
coords.altitudeAccuracy	La precisión de la altitud de la posición (devuelto si está disponible)
coords.heading	El rumbo en grados en sentido horario (devuelto si está disponible)
coords.speed	La velocidad en metros por segundo (devuelto si está disponible)
timestamp	La fecha/hora de la respuesta (devuelto si está disponible)

Geolocalizacion con HTML5 - watchPosition() Method



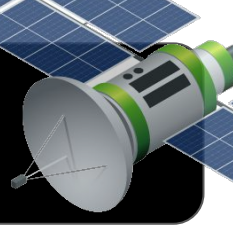
```
<script>
var x = document.getElementById("demo");
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.watchPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}
function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>
```

Geolocalizacion con HTML5 - Manejo de errores



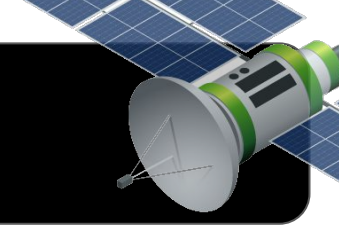
```
function showError(error) {
  switch(error.code) {
        case error.PERMISSION_DENIED:
        x.innerHTML = "User denied the request for Geolocation."
        break;
    case error.POSITION_UNAVAILABLE:
        x.innerHTML = "Location information is unavailable."
        break;
    case error.TIMEOUT:
        x.innerHTML = "The request to get user location timed out."
        break;
        case error.UNKNOWN_ERROR:
      x.innerHTML = "An unknown error occurred."
      break;
  }
}
```

Ejemplo track



https://www.html5rocks.com/tutorials/geolocation/trip_meter/

Ajax



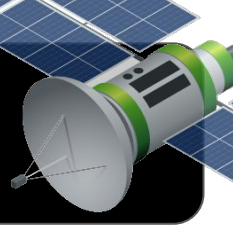
Es una de las tecnologías de HTML5 que permite transmitir información entre el cliente y el servidor sin recargar la página. A diferencia de la tecnología websocket las conexiones no son permanentes y usa protocolo HTTP.

Para hacer uso de esta tecnología en los navegadores se usa el objeto XMLHttpRequest().

Ejemplo:

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML = this.responseText;
            }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

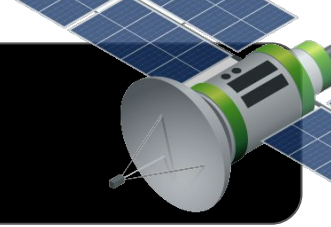
Backend



Para el almacenamiento de datos geoespaciales las bases de datos modernas ofrecen estructuras especiales llamadas Spatial Data Types. Este conjunto de tipos de geometría son propuestos para el entorno SQL con los tipos de geometría propuesto por Open Geospatial Consortium (OGC) basados en el modelo OpenGIS.

- **Geometry (noninstantiable)**
  - **Point (instantiable)**
  - **Curve (noninstantiable**
    - **LineString (instantiable)**
      - **Line**
      - **LinearRing**
  - **Surface (noninstantiable)**
    - **Polygon (instantiable)**
  - **GeometryCollection (instantiable)**
    - **MultiPoint (instantiable)**
    - **MultiCurve (noninstantiable)**
      - **MultiLineString (instantiable)**
    - **MultiSurface (noninstantiable)**
    - **MultiPolygon (instantiable)**

Backend



Agregar un punto a una tabla

```
ALTER TABLE mytable ADD coords Point;
```

Agregar un punto.

```
UPDATE  mytable SET coords = Point(lon, lat);
```

Crear un índice

```
CREATE SPATIAL INDEX sx_mytable_coords ON mytable(coords);
```

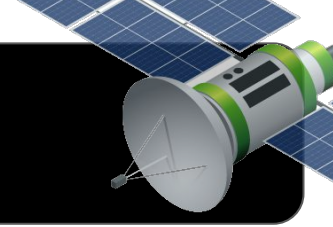
Actualizar un punto desde una posición enviada desde un gps de un celular.
Si lon y lat son tipo tipo texto

```
UPDATE mytable SET coords = GeomFromText(CONCAT('POINT (', lon, ' ', lat, ')'))
```

Si lon y lat son tipo tipo numérico

```
UPDATE mytable SET coords = POINT ( lon, lat)
```

Backend



Recuperar un dato

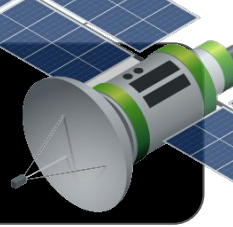
En formato texto → SELECT ST_AsText(g) FROM geom;
En binario → SELECT ST_AsBinary(g) FROM geom;
En formato GeoJSON → SELECT ST_AsGeoJSON(ST_GeomFromText('POINT(11.11111 12.22222)'),2);

```
+-----------------------------------------------------------------------------------------+
| ST_AsGeoJSON(ST_GeomFromText('POINT(11.11111 12.22222)'),2) |
+-----------------------------------------------------------------------------------------+
| {"type": "Point", "coordinates": [11.11, 12.22]}                                        |
+-----------------------------------------------------------------------------------------+
```

```
SET @json = '{ "type": "Point", "coordinates": [102.0, 0.0]}';

SELECT ST_AsText(ST_GeomFromGeoJSON(@json));
+-----------------------------------------------------+
| ST_AsText(ST_GeomFromGeoJSON(@json)) |
+-----------------------------------------------------+
| POINT(102 0)                                        |
+-----------------------------------------------------+
```

Backend

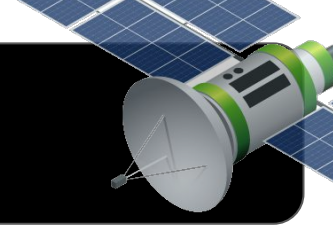


Funciones:

- ST_Distance(g1, g2) Distancia entre dos geometrías. Si son geometrías complejas devuelve la distancia más corta entre todas las combinaciones.
- ST_Contains(*g1*, *g2*) Devuelve 1 o 0 para indicar si la geometría *g1* está completamente contenida en *g2.*
- ST_Intersects(*g1*, *g2*) Devuelve 1 o 0 para indicar si la geometría *g1* intersecta *g2.*
- ST_Touches(*g1*, *g2*) Devuelve 1 o 0 para indicar si la geometría si se tocan los bordes de *g2 y g1.*

https://dev.mysql.com/doc/refman/5.7/en/spatial-relation-functions-object-shapes.html

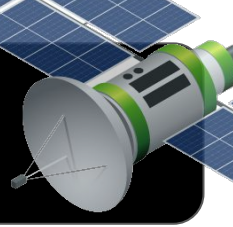
Ejemplo con php



```php
$nombre = $_REQUEST['nombre'];
$apellido = $_REQUEST['apellido'];
$descripcion = $_REQUEST['descripcion'];
$longitud = $_REQUEST['longitud'];
$latitud = $_REQUEST['latitud'];
$filename = $_FILES['adjunto']['tmp_name'];

exec("mv '$filename' fotos/$filename");
require('conexion.php');
$query = "  INSERT INTO sociedad (foto, posicion, nombre, apellido, descripcion)
            VALUES ('$filename', ST_GeomFromText('POINT($longitud $latitud)', 4326), '$nombre',
'$apellido', '$descripcion');";
mysqli_query($db, $query);
header("location: visualizacion.php");
```

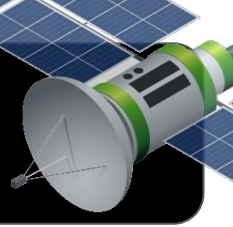
Ejemplo con php - Recuperación



```
$query = '  SELECT foto, ST_X(posicion) as lat, ST_Y(posicion) as lon, nombre, apellido, descripcion
            FROM sociedad';
$sql = mysqli_query($db, $query);
```

- ST_X(p): Devuelve el valor de la coordenada X para el objeto Point ***p*** como un número de precisión doble.
- ST_Y(p): Devuelve el valor de la coordenada Y para el objeto Point ***p*** como un número de precisión doble.

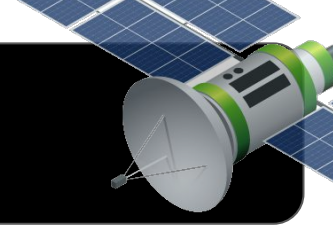
OpenLayers



OpenLayers hace fácil utilizar un mapa dinámico en cualquier página web. Puede mostrar map tiles, datos vectoriales y marcadores cargados de cualquier fuente.

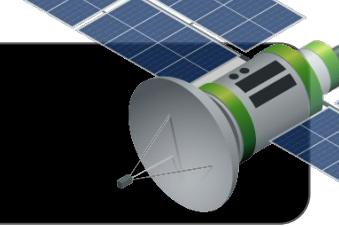
OpenLayers ha sido desarrollado para fomentar el uso de información geográfica de todo tipo. Es completamente libre, Open Source JavaScript, publicado bajo la licencia BSD de 2 cláusulas (también conocida como FreeBSD).

OpenLayers - Objetos Básicos



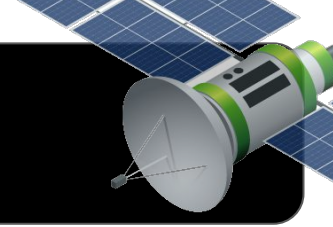
- Map - Mapa
  - Un mapa está hecho de layers, un view para poder visualizarlo, interacciones para modificar el contenido del mapa y controles con los componentes de la Interfaz de Usuario.
- View - Vista
  - La vista maneja los parámetros visuales del mapa como la resolución o rotación.
- Layer - Capas
  - Los layers son contenedores livianos que obtienen datos de sus fuentes. Estos pueden ser:
    - Tile
    - Image
    - Vector
    - Vector Tile
- Controles: Un control es un widget visible con un elemento DOM en una posición fija en la pantalla. Estos pueden involucrar entradas de usuarios como botones, o solo ser de carácter informativo, su posición está determinada por CSS.

OpenLayers - Objetos Básicos



- Interacciones
  - Interacciones por defecto del mapa, como dibujar, seleccionar, drag and drop, etc.
- Fuentes y formatos
  - Tile: Para layers que proveen imágenes en mosaico pre-renderizadas en cuadrícula y que están organizadas por niveles de zoom para resoluciones específicas.
  - Image: Imágenes renderizadas en un servidor que están disponibles en resoluciones y grados arbitrarios.
  - Vector: Datos vectorizados que son renderizados del lado del cliente.
  - Vector Tile: Layer para los datos vectorizados en mosaico del lado del cliente.
- Proyecciones: Son los sistemas de representación gráfica que establece una relación ordenada entre los puntos de la superficie curva de la Tierra y los de una superficie plana (mapa)
  - Por defecto EPSG:3857 Utiliza la proyección de Mercator (cilíndrica)
  - EPSG:4326 Proyección cilíndrica equidistante (Latitud Longitud)

OpenLayers - Ejemplo



```
var map = new ol.Map({
   layers: [openStreetMap, vectorLayer],
   target: 'map',
   loadTilesWhileAnimating: true,
   view: new ol.View({
     maxZoom: 18,
     center: [-6409852, -4571211],
     zoom: 2
     })
});
```

```
var vectorLayer = new ol.layer.Vector({
   source: new ol.source.Vector(),
   style: new ol.style.Style({
       image: new ol.style.Icon({
           src: 'recursos/marker.png',
           anchor: [0.5, 1],
           scale: 0.1
       })
   })
})
```

Licencias de uso de mapas

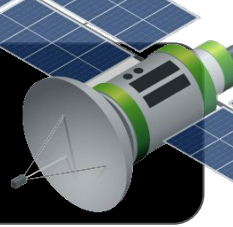
OpenStreetMap® es Open Data (un servicio de datos de acceso libre), con licencia Open Data Commons Open Database License (ODbL) de la Fundación OpenStreetMap (OSMF).

Puedes copiar, distribuir, transmitir y adaptar sus mapas e información libremente siempre y cuando des reconocimiento a OpenStreetMap y sus colaboradores. Si alteras o generas contenido sobre sus mapas e información, solo podrás distribuir estos cambios bajo la misma licencia.

En cambio Google Maps se puede utilizar a un costo muy elevado y sin modificar el mapa de forma drástica, cualquier cambio que se haga, debe ser lo suficientemente pequeño para no alterar la estética de google.

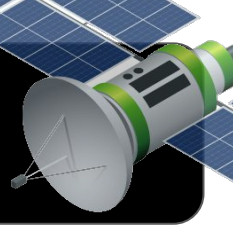
La version gratuita de Google Maps solo permite la visualización de los mismos y pequeñas alteraciones.

Ejemplos - Visualización



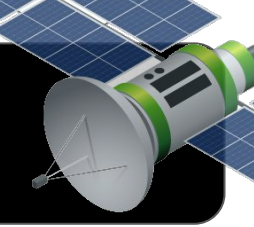
```
function makeFeatures(datos) {
   var vecAux = [];
   var i;
   for (i = 0; i < datos.length; i++) {
       vecAux.push(new ol.Feature({
           type: 'point',
           geometry: new ol.geom.Point(ol.proj.transform([datos[i].lat, datos[i].lon], 'EPSG:4326', 'EPSG:3857')),
           foto: datos[i].foto,
           lat: datos[i].lat,
           lon: datos[i].lon,
           nombre: datos[i].nombre,
           apellido: datos[i].apellido,
           descripcion: datos[i].descripcion
       }));   }
   vectorLayer.getSource().addFeatures(vecAux);
}
makeFeatures(datos);
```

Ejemplos - Visualización



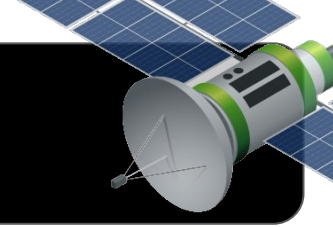
```
var select = new ol.interaction.Select({ condition: ol.events.condition.click });
map.addInteraction(select);
select.on('select', hacerCuandoSeleccione, this);
```

Ejemplos - Visualización



```
function hacerCuandoSeleccione(that) {
   if (that.selected.length >= 1) {
       var geometria = that.selected[0].getGeometry();
       var posicion = geometria.getFirstCoordinate();
       var propiedades = that.selected[0].getProperties()
       var aux = "";
       $("#popup").fadeIn();
       overlay.setPosition(posicion);
       content.innerHTML = "";
       var claves = Object.keys(propiedades);
       claves = claves.filter(item => item != "geometry");
       claves = claves.filter(item => item != "type");
       claves = claves.filter(item => item != "foto");
       claves = claves.filter(item => item != "lat");
       claves = claves.filter(item => item != "lon");
       claves = claves.filter(item => item != "styleUrl");
       claves.forEach((clave) => {
         if (propiedades[clave] != "") {
           content.innerHTML += clave +": "+propiedades[clave] + "<br>";
         }
       });
       aux +='<a href="fotos/' + propiedades.foto + '">';
       aux +=‘<img class="imagen" src="fotos/' + propiedades.foto + '">'
       aux += ' </a>';
       content.innerHTML += aux;
   }
   select.getFeatures().clear();
}
```

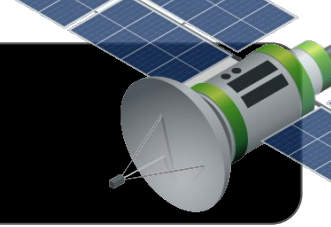
Ejemplos - Tracking



```
var vectorLayer = new ol.layer.Vector({
   source: new ol.source.Vector()
})
var miView = new ol.View({
   maxZoom: 18,
   center: [-6409852, -4571211],
   zoom: 2
})
var map = new ol.Map({
   layers: [openStreetMap, vectorLayer],
   target: 'map',
   view: miView
});
```

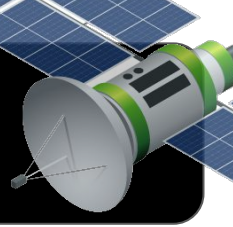
```
function getLocation() {
   if (navigator.geolocation) {
       navigator.geolocation.watchPosition(mark);
   } else {
       alert("Geolocation is not supported by this
browser.");
   }
}
getLocation();
```

Ejemplos - Tracking



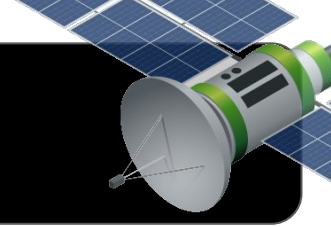
```
function mark(position) {
   var miPos = ol.proj.transform([position.coords.longitude, position.coords.latitude], 'EPSG:4326', 'EPSG:3857');
   vectorLayer.getSource().addFeature(new ol.Feature({ geometry: new ol.geom.Point(miPos) }))
   miView.animate({
       center: miPos,
       zoom: 15,
       duration: 1000
   });
};
```

Síntesis



1. HTML5 tiene una serie de elementos para el uso de geolocalización muy versátiles y variados.
2. Las implementaciones de SQL tienen una serie de funciones y objetos que nos permiten tanto guardar como recuperar y manipular elementos georeferenciados.
3. OpenLayers es una herramienta muy útil y poderosa, que en poco tiempo nos permite hacer mapas completos muy fácilmente y con una capacidad de crecimiento muy grande.
4. OpenStreetMaps es libre y posee muchísima funcionalidad, es una alternativa muy importante a la hora de decidir qué mapas utilizar.
5. Para localización necesitamos un dispositivo con GPS y en este caso, con un cliente web, un servidor con un backend y servidor de base de datos.

Github



https://github.com/WenceslaoMateos/CharlaGeolocalizacion

Preguntas

Evans, Felipe - Mutti, Pilar - Mateos, Wenceslao