Hackathon Badger

Seller's GrandPrix

Juan González Serrano
Gleiston Guerrero

Abril, 2018

Contents

1 Introducción

En este documento explicaremos, por encima, que funcionalidad tiene nuestro proyecto web y como lo hemos hecho posible.

2 La idea

2.1 Idea principal

La idea principal parte un poco de deportes de competición como puede ser el ciclismo o la formula 1. En ambos casos se puntúa a los competidores que consigue los tiempos mas bajo.
Pero no solo se tiene en cuenta el tiempo total de una carrera, si no que también se contemplan los cronos que realizan los corredores en distintos sectores en los que se divide una carrera.

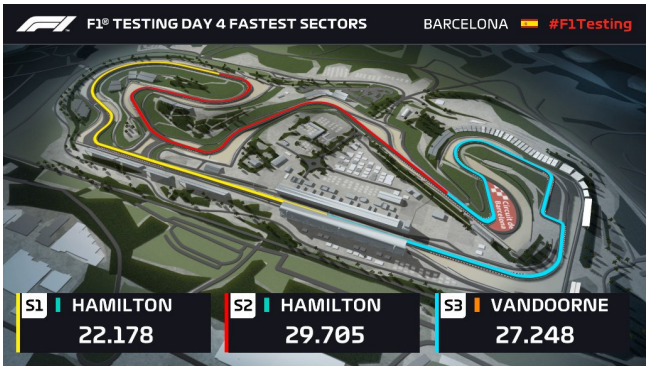

Figure 1: Circuito Formula 1

Ahora cojamos esa idea y llevémosla al día a día de los clientes de Badger. Una carrera completa seria el total de la ruta que debe realizar un trabajador que usa Badger y las distintas etapas en las que se divide una carrera ciclista, por ejemplo, son los distancias que hay entre los distintos clientes que tiene dicho usuario.

2.2 Idea secundaria

Los usuarios de Badger es gente que pasa mucho tiempo en la calle. Por lo que muy seguramente descubrirán nuevos establecimientos antes que sus propios jefes o compañeros de captación de clientes.
Por ello pensamos que cuando un trabajador encuentre un cliente potencial, solo tiene que hacer un *"tap"* sobre el mapa, indicando un posible cliente.
Al final del mes se le dará algún tipo de bonificación a aquel trabajador que haya posteado mayor numero de clientes potenciales.

3 Desarrollo

En la *figura 2* podemos ver nuestro sitio web.
Hemos remarcado con rectángulos de colores las principales áreas que la dividen.

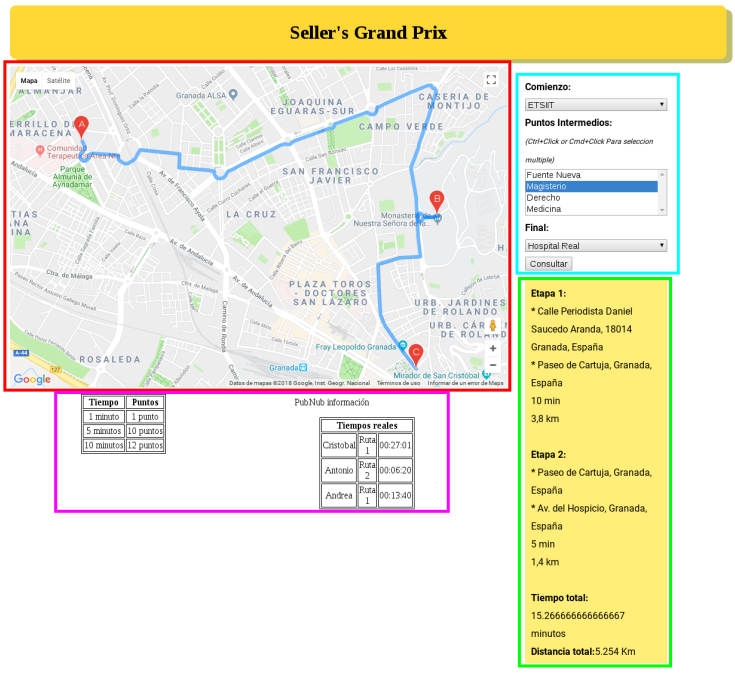

Figure 2: Sellers GrandPrix

- **Rojo** *MAPA*: (3.1) mapa donde se representara la ruta que calculara el sitio web.
- **Turquesa** *PARÁMETROS BÚSQUEDA*: (3.2) aquí indicaremos desde donde iniciaremos nuestra ruta, donde la terminaremos y a que clientes visitaremos.
- **Verde** *RESULTADOS BÚSQUEDA*: (3.2) Nos mostrara las distintas etapas en las que se divide nuestra búsqueda. Podemos ver dirección de comienzo y fin de etapa, distancia a recorrer y tiempo estimado. Estos dos últimos datos los mostramos por etapa y por ruta total.
- **Morado** *TABLAS*: (3.3) Se encuentran dos tablas.
  - Puntos: tabla que muestra los puntos que recibirán los trabajadores según los minutos que consigan mejorar los tiempos que calcula la herramienta.

– Tiempos reales: La idea es que sea una tabla interactiva que muestre en tiempo real la ruta que esta haciendo cada trabajador y sus tiempos. Podría utilizarse PubNub. Interactividad no implementada.

3.1 Mapa

Si consultamos el código html de la pagina veremos un *div* en la **linea 16** que esta vació y tiene como *id "map"*.
Dejemoslo de un lado por un momento y vayamos a la **linea 81** donde encontraremos la magia del sitio web.
Encontramos dos grandes funciones.

3.1.1 initMap()

Función encargada de "dibujar el mapa en el sitio web.

```
84        function initMap() {
85          var directionsService = new google.maps.DirectionsService;
86          var directionsDisplay = new google.maps.DirectionsRenderer;
87          var granada = {lat: 37.1809411, lng: -3.6262914};
88          var map = new google.maps.Map(document.getElementById('map'), {
89            zoom: 13,
90
91            center: granada
92          });
93
94          directionsDisplay.setMap(map);
95
96          document.getElementById('submit').addEventListener('click', function() {
97            calculateAndDisplayRoute(directionsService, directionsDisplay);
98          });
```

Figure 3: "Inicialización" del mapa

No fuimos capaces de implementar la idea de colocar marker sobre la marcha en el mapa. Pero para mostrar la idea establecimos uno en el mapa.
Esta definición también se encuentra dentro de la función *initMap()*

```
100         var contentString = '<div id="content">'+
101             '<div id="siteNotice">'+
102             '</div>'+
103             '<h1>Optica</h1>';
104
105         var marker = new google.maps.Marker({
106           position: {lat: 37.1779517,lng: -3.6102454},
107           map: map,
108           title:'Optica'
109         });
110
111         var infowindow = new google.maps.InfoWindow({
112           content: contentString
113         });
114
115
116         marker.addListener('click', function() {
117           infowindow.open(map, marker);
118         });
```

Figure 4: Definición de "marker"

Por ultimo hablar sobre la *figura 5*. Este código también se encuentra dentro de la función *initMap()*. Cuando pulsemos el botón para calcular una ruta, invocaremos la función *calculateAndDisplayRoute*.

```
96          document.getElementById('submit').addEventListener('click', function() {
97            calculateAndDisplayRoute(directionsService, directionsDisplay);
98          });
```

Figure 5: Invocación a *"calculateAndDisplayRoute"*

3.2 Parámetros de búsqueda y Resultados Búsqueda

3.2.1 calculateAndDisplayRoute()

Función encargada de consultar la ruta y devolver los resultados.
Encontramos la ubicación que deseamos muestre por defecto, el zoom.

Lo primero que hacemos es "capturar" los puntos intermedios *"waipoints"*.

```
126       function calculateAndDisplayRoute(directionsService, directionsDisplay) {
127         var waypts = [];
128         var checkboxArray = document.getElementById('waypoints');
129         for (var i = 0; i < checkboxArray.length; i++) {
130           if (checkboxArray.options[i].selected) {
131             waypts.push({
132               location: checkboxArray[i].value,
133               stopover: true
134             });
135           }  //if
136         }  //for
```

Figure 6: Captura *"waipoints"*

Tras esta captura comienza a articularse la búsqueda. Establecemos los puntos de inicio y fin, los puntos intermedios, o *waipoints*, que hemos capturado antes, habilitamos la opción de ruta optimizada y el modelo de ruta.

```
137         directionsService.route({
138           origin: document.getElementById('start').value,
139           destination: document.getElementById('end').value,
140           waypoints: waypts,
141           optimizeWaypoints: true,
142           travelMode: 'DRIVING'
143         }, function(response, status) {
144           if (status === 'OK') {
```

Figure 7: Establecemos parámetros de búsqueda

Si nos fijamos en la *figura 7*, en la linea 143, interpretamos que es donde se realiza la consulta. En *response* tendremos la respuesta a nuestra consulta y en *status* el estado de la consulta.
Lo que ya sigo de código es como vinculamos con el elemento html donde imprimiremos los resultados de la búsqueda *(ver figura 8, linea 147 y 150)*, como "inyectamos la información obtenida en nuestro elemento html *(ver imagen 8, lineas 152-160, 164 y 165)*

```
143         }, function(response, status) {
144           if (status === 'OK') {
145             directionsDisplay.setDirections(response);
146             var route = response.routes[0];
147             var summaryPanel = document.getElementById('directions-panel');
148             var distanTotal=0;
149             var timeTotal=0;
150             summaryPanel.innerHTML = '';
151             // For each route, display summary information.
152             for (var i = 0; i < route.legs.length; i++) {
153               var routeSegment = i + 1;
154               summaryPanel.innerHTML += '<b>Etapa '+routeSegment+': '+'</b><br>';
155               summaryPanel.innerHTML += '<b>* </b>'+route.legs[i].start_address + '<br>';
156               summaryPanel.innerHTML += '<b>* </b>'+route.legs[i].end_address + '<br>';
157               summaryPanel.innerHTML += route.legs[i].duration.text + '<br>';
158               summaryPanel.innerHTML += route.legs[i].distance.text + '<br><br>';
159               distanTotal += route.legs[i].distance.value;
160               timeTotal += route.legs[i].duration.value;
161             }
162             distanTotal /=1000;
163             timeTotal /=60;
164             summaryPanel.innerHTML += '<b>Tiempo total: </b>'+timeTotal+' minutos'+'<br>';
165             summaryPanel.innerHTML += '<b>Distancia total:</b>'+distanTotal+' Km'+'<br>';
166             tiempo = timeTotal;
167           } else {
168             window.alert('Directions request failed due to ' + status);
169           }
170         });
171       }  //calculateAndDisplayRoute
```

Figure 8: Uso resultados de consulta

Si nos fijamos en la *figura 8, linea 145,* lo que hacemos es volver a "dibujar" el mapa pero con la información que obtenemos de nuestra consulta.

3.3 Tablas

Tablas estáticas, creadas en código html para que ilustren la idea sobre como recompensar a los empleados, *figura 9*, y de llevar un control de los tiempos que están realizando cada trabajador, *figura 10*.

```
<div id="puntos">
  <table>
    <tr>
      <th>Tiempo</th><th>Puntos</th>
    </tr>
    <tr>
      <td>1 minuto</td><td> 1 punto</td>
    </tr>
    <tr>
      <td>5 minutos</td><td>10 puntos</td>
    </tr>
    <tr>
      <td>10 minutos</td><td>12 puntos</td>
    </tr>
  </table>
</div>
```

Figure 9: Tabla relación puntos - tiempo mejorado

```
<div id="pubnub">
  <p>
    PubNub información
  </p>
  <table>
    <tr>
      <th colspan="3">Tiempos reales</th>
    </tr>
    <tr>
      <td>Cristobal</td><td>Ruta 1</td><td>00:27:01</td>
    </tr>
    <tr>
      <td>Antonio</td><td>Ruta 2</td><td>00:06:20</td>
    </tr>
    <tr>
      <td>Andrea</td><td>Ruta 1</td><td>00:13:40</td>
    </tr>
  </table>
</div>
```

Figure 10: Tabla tiempos cada trabajador

4 Anexo

He publicado el codigo en mi perfil de github.
`https://github.com/naujgs/HACKATHON_Badger`